wildbook-ia

Release latest

Feb 12, 2022

Contents:

For details about the Wildbook project see the Wild Me website.

Wildbook's Image Analysis is colloquially known as Wildbook-IA and by developers as wbia (wib-ee-A). Any references to WBIA in this documentation should be assumed to therefore mean Wildbook-IA.

The Wildbook-IA application is used for the storage, management and analysis of images and derived data used by computer vision algorithms. It aims to compute who an animal is, what species an animal is, and where an animal is with the ultimate goal being to ask important why biological questions.

This project is the Machine Learning (ML) / computer vision component of the WildBook project. This project is an actively maintained fork of the popular IBEIS (Image Based Ecological Information System) software suite for wildlife conservation. The original IBEIS project is maintained by Jon Crall (@Erotemic) at https://github.com/Erotemic/ibeis. The IBEIS toolkit originally was a wrapper around HotSpotter, which original binaries can be downloaded from: http://cs.rpi.edu/hotspotter/

Currently the system is build around and SQLite database, a web UI, and matplotlib visualizations. Algorithms employed are: convolutional neural network detection and localization and classification, hessian-affine keypoint detection, SIFT keypoint description, LNBNN identification using approximate nearest neighbors.

CHAPTER 1

API

1.1 wbia.algo package

1.1.1 Subpackages

1.1.1.1 wbia.algo.detect package

1.1.1.1.1 Subpackages

1.1.1.1.1.1 wbia.algo.detect.nms package

1.1.1.1.1.2 Submodules

1.1.1.1.1.3 wbia.algo.detect.nms.py_cpu_nms module

wbia.algo.detect.nms.py_cpu_nms.**py_cpu_nms**(*dets*, *scores*, *thresh*)
Pure Python NMS baseline.

1.1.1.1.1.4 Module contents

1.1.1.2 Submodules

1.1.1.3 wbia.algo.detect.azure module

Interface to Azure object proposals.

wbia.algo.detect.azure.**detect**(*gpath_list*, *config_filepath*, *verbose=False*, ***kwargs*)
Detect image filepaths with azure.

Parameters **gpath_list** (*list of str*) – the list of image paths that need proposal candidates

Kwargs (optional): refer to the Azure documentation for configuration settings

Returns iter

wbia.algo.detect.azure.**detect_gid_list**(*ibs*, *gid_list*, *verbose=False*, ***kwargs*)
Detect gid_list with azure.

Parameters **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection

Kwargs (optional): refer to the Azure documentation for configuration settings

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection

Kwargs: detector, config_filepath, weights_filepath, verbose

Yields *tuple* – (gid, gpath, result_list)

wbia.algo.detect.azure.**label**(*chip_filepath_list*, *labeler_weight_filepath*, *verbose=False*, ***kwargs*)
Classify aid_list with azure.

wbia.algo.detect.azure.**label_aid_list**(*ibs*, *aid_list*, *verbose=False*, ***kwargs*)
Classify aid_list with azure.

Parameters **aid_list** (*list of int*) – the list of IBEIS annotation rowids that need classifying

Kwargs (optional): refer to the Azure documentation for configuration settings

Yields *tuple* – (gid, gpath, result_list)

1.1.1.1.4 wbia.algo.detect.canonical module

Interface to Lightnet object proposals.

class wbia.algo.detect.canonical.**Augmentations**
Bases: object

class wbia.algo.detect.canonical.**ImageFilePathList**(*filepaths*, *targets=True*, *transform=None*, *target_transform=None*)
Bases: torch.utils.data.dataset.Dataset

class wbia.algo.detect.canonical.**TrainAugmentations**
Bases: *wbia.algo.detect.canonical.Augmentations*

class wbia.algo.detect.canonical.**ValidAugmentations**
Bases: *wbia.algo.detect.canonical.Augmentations*

wbia.algo.detect.canonical.**finetune**(*model*, *dataloaders*, *optimizer*, *scheduler*, *device*, *num_epochs=128*, *under=1.0*, *over=1.0*)

wbia.algo.detect.canonical.**test**(*gpath_list*, *canonical_weight_filepath=None*, ***kwargs*)

wbia.algo.detect.canonical.**test_ensemble**(*filepath_list*, *weights_path_list*, ***kwargs*)

wbia.algo.detect.canonical.**test_single**(*filepath_list*, *weights_path*, *batch_size=512*)

wbia.algo.detect.canonical.**train**(*data_path*, *output_path*, *batch_size=32*)

wbia.algo.detect.canonical.**visualize_augmentations**(*dataset*, *augmentation*, *tag*, *num=20*)

1.1.1.1.5 wbia.algo.detect.darknet module

Interface to Darknet object proposals.

wbia.algo.detect.darknet.**detect**(*gpath_list*, *config_filepath*, *weight_filepath*, *class_filepath*, *sensitivity*, *verbose=False*, *use_gpu=True*, *use_gpu_id=0*, ***kwargs*)

Parameters **gpath_list** (*list of str*) – the list of image paths that need proposal candidates

Kwargs (optional): refer to the Darknet documentation for configuration settings

Returns iter

wbia.algo.detect.darknet.**detect_gid_list**(*ibs*, *gid_list*, *downsample=True*, *verbose=False*, ***kwargs*)

Parameters

- **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection
- **downsample** (*bool, optional*) – a flag to indicate if the original image sizes should be used; defaults to True

  True: ibs.get_image_detectpaths() is used False: ibs.get_image_paths() is used

Kwargs (optional): refer to the Darknet documentation for configuration settings

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection
- **downsample** (*bool, optional*) – a flag to indicate if the original image sizes should be used; defaults to True

Kwargs: detector, config_filepath, weights_filepath, verbose

Yields *tuple* – (gid, gpath, result_list)

CommandLine: python -m wbia.algo.detect.darknet detect_gid_list –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.detect.darknet import *  # NOQA
>>> from wbia.core_images import LocalizerConfig
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids()
>>> config = {'verbose': True}
```

(continues on next page)

(continued from previous page)

```
>>> downsample = False
>>> results_list = detect_gid_list(ibs, gid_list, downsample, **config)
>>> results_list = list(results_list)
>>> print('result lens = %r' % (map(len, list(results_list))))
>>> print('result[0] = %r' % (len(list(results_list[0][2]))))
>>> config = {'verbose': True}
>>> downsample = False
>>> results_list = detect_gid_list(ibs, gid_list, downsample, **config)
>>> results_list = list(results_list)
>>> print('result lens = %r' % (map(len, list(results_list))))
>>> print('result[0] = %r' % (len(list(results_list[0][2]))))
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

Yields results (list of dict)

1.1.1.1.6 wbia.algo.detect.densenet module

Interface to Lightnet object proposals.

class wbia.algo.detect.densenet.**Augmentations**
Bases: object

class wbia.algo.detect.densenet.**ImageFilePathList**(*filepaths*, *targets=None*, *transform=None*, *target_transform=None*)
Bases: torch.utils.data.dataset.Dataset

class wbia.algo.detect.densenet.**StratifiedSampler**(*dataset*, *phase*, *multiplier=1.0*)
Bases: torch.utils.data.sampler.Sampler

class wbia.algo.detect.densenet.**TrainAugmentations**(*blur=True*, *flip=False*, *rotate=10*, *shear=10*, ***kwargs*)
Bases: *wbia.algo.detect.densenet.Augmentations*

class wbia.algo.detect.densenet.**ValidAugmentations**(***kwargs*)
Bases: *wbia.algo.detect.densenet.Augmentations*

wbia.algo.detect.densenet.**features**(*filepath_list*, *batch_size=512*, *multi=True*, ***kwargs*)

wbia.algo.detect.densenet.**finetune**(*model*, *dataloaders*, *criterion*, *optimizer*, *scheduler*, *device*, *num_epochs=128*)

wbia.algo.detect.densenet.**test**(*gpath_list*, *classifier_weight_filepath=None*, *return_dict=False*, *multiclass=False*, ***kwargs*)

wbia.algo.detect.densenet.**test_dict**(*gpath_list*, *classifier_weight_filepath=None*, *return_dict=None*, ***kwargs*)

wbia.algo.detect.densenet.**test_ensemble**(*filepath_list*, *weights_path_list*, *classifier_weight_filepath*, *ensemble_index*, *ibs=None*, *gid_list=None*, *multiclass=False*, ***kwargs*)

wbia.algo.detect.densenet.**test_single**(*filepath_list*, *weights_path*, *batch_size=1792*, *multi=True*, ***kwargs*)

wbia.algo.detect.densenet.**train**(*data_path*, *output_path*, *batch_size=48*, *class_weights={}*, *multi=True*, *sample_multiplier=4.0*, *allow_missing_validation_classes=False*, ***kwargs*)

wbia.algo.detect.densenet.**visualize_augmentations**(*dataset*, *augmentation*, *tag*, *num_per_class=10*, ***kwargs*)

1.1.1.1.7 wbia.algo.detect.fasterrcnn module

Interface to Faster R-CNN object proposals.

wbia.algo.detect.fasterrcnn.**detect**(*gpath_list*, *config_filepath*, *weight_filepath*, *class_filepath*, *sensitivity*, *verbose=False*, *use_gpu=True*, *use_gpu_id=0*, ***kwargs*)

Parameters **gpath_list** (*list of str*) – the list of image paths that need proposal candidates

Kwargs (optional): refer to the Faster R-CNN documentation for configuration settings

Returns iter

wbia.algo.detect.fasterrcnn.**detect_gid_list**(*ibs*, *gid_list*, *downsample=True*, *verbose=False*, ***kwargs*)

Parameters

- **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection
- **downsample** (*bool, optional*) – a flag to indicate if the original image sizes should be used; defaults to True

  True: ibs.get_image_detectpaths() is used False: ibs.get_image_paths() is used

Kwargs (optional): refer to the Faster R-CNN documentation for configuration settings

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection
- **downsample** (*bool, optional*) – a flag to indicate if the original image sizes should be used; defaults to True

Kwargs: detector, config_filepath, weights_filepath, verbose

Yields *tuple* – (gid, gpath, result_list)

CommandLine: python -m wbia.algo.detect.fasterrcnn detect_gid_list –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.detect.fasterrcnn import *  # NOQA
>>> from wbia.core_images import LocalizerConfig
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids()
>>> config = {'verbose': True}
>>> downsample = False
>>> results_list = detect_gid_list(ibs, gid_list, downsample, **config)
>>> results_list = list(results_list)
```

(continues on next page)

(continued from previous page)

```
>>> print('result lens = %r' % (map(len, list(results_list))))
>>> print('result[0] = %r' % (len(list(results_list[0][2]))))
>>> config = {'verbose': True}
>>> downsample = False
>>> results_list = detect_gid_list(ibs, gid_list, downsample, **config)
>>> results_list = list(results_list)
>>> print('result lens = %r' % (map(len, list(results_list))))
>>> print('result[0] = %r' % (len(list(results_list[0][2]))))
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

Yields results (list of dict)

1.1.1.1.8 wbia.algo.detect.grabmodels module

wbia.algo.detect.grabmodels.**assert_models**(*modeldir='default'*, *verbose=True*)

wbia.algo.detect.grabmodels.**ensure_models**(*modeldir='default'*, *verbose=True*)

Parameters **modeldir** (*str*) –

CommandLine: python -m wbia.algo.detect.grabmodels –test-ensure_models

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.detect.grabmodels import *  # NOQA
>>> modeldir = 'default'
>>> result = ensure_models(modeldir)
>>> print(result)
```

wbia.algo.detect.grabmodels.**get_species_trees_paths**(*species*, *modeldir='default'*)

Parameters

- **species** –
- **modeldir** (*str*) –

Returns trees_path

Return type

?

CommandLine: python -m wbia.algo.detect.grabmodels –test-get_species_trees_paths

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.detect.grabmodels import *  # NOQA
>>> import wbia
>>> # build test data
```

(continues on next page)

(continued from previous page)

```
>>> species = wbia.const.TEST_SPECIES.ZEB_PLAIN
>>> modeldir = 'default'
>>> # execute function
>>> trees_path = get_species_trees_paths(species, modeldir)
>>> # verify results
>>> result = str(trees_path)
>>> print(result)
```

wbia.algo.detect.grabmodels.**iter_algo_modeldirs**(*modeldir='default'*, *ensure-base=False*)

wbia.algo.detect.grabmodels.**redownload_models**(*modeldir='default'*, *verbose=True*)

Parameters

- **modeldir** (*str*) – (default = 'default')
- **verbose** (*bool*) – verbosity flag(default = True)

CommandLine: python -m wbia.algo.detect.grabmodels –test-redownload_models

Example

```
>>> # SCRIPT
>>> from wbia.algo.detect.grabmodels import *  # NOQA
>>> result = redownload_models()
```

1.1.1.1.9 wbia.algo.detect.lightnet module

Interface to Lightnet object proposals.

wbia.algo.detect.lightnet.**detect**(*gpath_list*, *orient_list*, *config_filepath=None*, *weight_filepath=None*, *classes_filepath=None*, *sensitivity=0.0*, *verbose=False*, *flip=False*, *batch_size=192*, ***kwargs*)

Detect image filepaths with lightnet.

Parameters **gpath_list** (*list of str*) – the list of image paths that need proposal candidates

Kwargs (optional): refer to the Lightnet documentation for configuration settings

Returns iter

wbia.algo.detect.lightnet.**detect_gid_list**(*ibs*, *gid_list*, *verbose=False*, ***kwargs*)

Detect gid_list with lightnet.

Parameters **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection

Kwargs (optional): refer to the Lightnet documentation for configuration settings

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection

Kwargs: detector, config_filepath, weight_filepath, verbose

Yields *tuple* – (gid, gpath, result_list)

1.1.1.1.10 wbia.algo.detect.orientation module

Interface to Lightnet object proposals.

class wbia.algo.detect.orientation.**Augmentations**
Bases: object

class wbia.algo.detect.orientation.**ImageFilePathList**(*filepaths*, *targets=None*, *transform=None*, *target_transform=None*)
Bases: torch.utils.data.dataset.Dataset

class wbia.algo.detect.orientation.**StratifiedSampler**(*dataset*, *phase*, *multiplier=1.0*)
Bases: torch.utils.data.sampler.Sampler

class wbia.algo.detect.orientation.**TrainAugmentations**(*blur=True*, *flip=False*, *rotate=10*, *shear=10*, ***kwargs*)
Bases: *wbia.algo.detect.orientation.Augmentations*

class wbia.algo.detect.orientation.**ValidAugmentations**(***kwargs*)
Bases: *wbia.algo.detect.orientation.Augmentations*

wbia.algo.detect.orientation.**features**(*filepath_list*, *batch_size=512*, *multi=True*, ***kwargs*)

wbia.algo.detect.orientation.**finetune**(*model*, *dataloaders*, *criterion*, *optimizer*, *scheduler*, *device*, *num_epochs=128*)

wbia.algo.detect.orientation.**test**(*gpath_list*, *classifier_weight_filepath=None*, *return_dict=False*, *multiclass=False*, ***kwargs*)

wbia.algo.detect.orientation.**test_dict**(*gpath_list*, *classifier_weight_filepath=None*, *return_dict=None*, ***kwargs*)

wbia.algo.detect.orientation.**test_ensemble**(*filepath_list*, *weights_path_list*, *classifier_weight_filepath*, *ensemble_index*, *ibs=None*, *gid_list=None*, *multiclass=False*, ***kwargs*)

wbia.algo.detect.orientation.**test_single**(*filepath_list*, *weights_path*, *batch_size=1792*, *multi=True*, ***kwargs*)

wbia.algo.detect.orientation.**train**(*data_path*, *output_path*, *batch_size=48*, *class_weights={}*, *multi=True*, *sample_multiplier=1.0*, ***kwargs*)

wbia.algo.detect.orientation.**visualize_augmentations**(*dataset*, *augmentation*, *tag*, *num_per_class=10*, ***kwargs*)

1.1.1.1.11 wbia.algo.detect.randomforest module

Interface to pyrf random forest object detection.

wbia.algo.detect.randomforest.**detect**(*ibs*, *gpath_list*, *tree_path_list*, ***kwargs*)

Parameters

- **gpath_list** (*list of str*) – the list of image paths that need detection
- **tree_path_list** (*list of str*) – the list of trees to load for detection

Kwargs (optional): refer to the PyRF documentation for configuration settings

Returns iter

wbia.algo.detect.randomforest.**detect_gid_list**(*ibs*, *gid_list*, *tree_path_list*, *downsample=True*, ***kwargs*)

Parameters

- **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection
- **tree_path_list** (*list of str*) – the list of trees to load for detection
- **downsample** (*bool, optional*) – a flag to indicate if the original image sizes should be used; defaults to True

  True: ibs.get_image_detectpaths() is used False: ibs.get_image_paths() is used

Kwargs (optional): refer to the PyRF documentation for configuration settings

Yields results (list of dict)

wbia.algo.detect.randomforest.**detect_gid_list_with_species**(*ibs*, *gid_list*, *species*, *downsample=True*, ***kwargs*)

Parameters

- **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection
- **species** (*str*) – the species that should be used to select the pre-trained random forest model
- **downsample** (*bool, optional*) – a flag to indicate if the original image sizes should be used; defaults to True

  True: ibs.get_image_detectpaths() is used False: ibs.get_image_paths() is used

Kwargs (optional): refer to the PyRF documentation for configuration settings

Returns iter

CommandLine: python -m wbia.algo.detect.randomforest –test-detect_gid_list_with_species

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.detect.randomforest import *  # NOQA
>>> from wbia.algo.detect.randomforest import _get_models  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> species = wbia.const.TEST_SPECIES.ZEB_PLAIN
>>> gid_list = ibs.get_valid_gids()
>>> downsample = True
>>> kwargs = {}
>>> # execute function
>>> result = detect_gid_list_with_species(ibs, gid_list, species, downsample)
>>> # verify results
>>> print(result)
```

wbia.algo.detect.randomforest.**detect_gpath_list_with_species**(*ibs*, *gpath_list*, *species*, ***kwargs*)

Parameters

- **gpath_list** (*list of str*) – the list of image paths that need detection
- **species** (*str*) – the species that should be used to select the pre-trained random forest model
- **downsample** (*bool, optional*) – a flag to indicate if the original image sizes should be used; defaults to True

  True: ibs.get_image_detectpaths() is used False: ibs.get_image_paths() is used

Kwargs (optional): refer to the PyRF documentation for configuration settings

Yields iter

wbia.algo.detect.randomforest.**train_gid_list**(*ibs*, *gid_list*, *trees_path=None*, *species=None*, *setup=True*, *teardown=False*, ***kwargs*)

Parameters

- **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection
- **trees_path** (*str*) – the path that the trees will be saved into (along with temporary training inventory folders that are deleted once training is finished)
- **species** (*str*) – the species that should be used to assign to the newly trained trees

Kwargs (optional): refer to the PyRF documentation for configuration settings

Returns None

wbia.algo.detect.randomforest.**train_gpath_list**(*ibs*, *train_pos_cpath_list*, *train_neg_cpath_list*, *trees_path=None*, ***kwargs*)

Parameters

- **train_pos_cpath_list** (*list of str*) – the list of positive image paths for training
- **train_neg_cpath_list** (*list of str*) – the list of negative image paths for training
- **trees_path** (*str*) – the path that the trees will be saved into (along with temporary training inventory folders that are deleted once training is finished)
- **species** (*str, optional*) – the species that should be used to assign to the newly trained trees

Kwargs (optional): refer to the PyRF documentation for configuration settings

Returns None

1.1.1.1.12 wbia.algo.detect.rf module

Interface to Darknet object proposals.

wbia.algo.detect.rf.**classify**(*vector_list*, *weight_filepath*, *verbose=False*, ***kwargs*)

Parameters **thumbail_list** (*list of str*) – the list of image thumbnails that need classifying

Returns iter

wbia.algo.detect.rf.**classify_helper**(*weight_filepath*, *vector_list*, *index_list=None*, *verbose=False*)

1.1.1.1.13 wbia.algo.detect.selectivesearch module

Interface to Selective Search object proposals.

wbia.algo.detect.selectivesearch.**detect**(*gpath_list*, *matlab_command='selective_search'*, *verbose=False*, ***kwargs*)

Parameters **gpath_list** (*list of str*) – the list of image paths that need proposal candidates

Kwargs (optional): refer to the Selective Search documentation for configuration settings

Returns iter

wbia.algo.detect.selectivesearch.**detect_gid_list**(*ibs*, *gid_list*, *downsample=True*, *verbose=False*, ***kwargs*)

Parameters

- **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection
- **downsample** (*bool, optional*) – a flag to indicate if the original image sizes should be used; defaults to True

  True: ibs.get_image_detectpaths() is used False: ibs.get_image_paths() is used

Kwargs (optional): refer to the Selective Search documentation for configuration settings

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection
- **downsample** (*bool, optional*) – a flag to indicate if the original image sizes should be used; defaults to True

Kwargs: detector, config_filepath, weights_filepath, verbose

Yields *tuple* – (gid, gpath, result_list)

CommandLine: python -m wbia.algo.detect.selectivesearch detect_gid_list –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.detect.selectivesearch import *  # NOQA
>>> from wbia.core_images import LocalizerConfig
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids()
>>> config = {'matlab_command': 'selective_search', 'verbose': True}
>>> downsample = False
>>> results_list = detect_gid_list(ibs, gid_list, downsample, **config)
>>> results_list = list(results_list)
>>> print('result lens = %r' % (map(len, list(results_list))))
>>> print('result[0] = %r' % (len(list(results_list[0][2]))))
```

(continues on next page)

(continued from previous page)

```
>>> config = {'matlab_command': 'selective_search_rcnn', 'verbose': True}
>>> downsample = False
>>> results_list = detect_gid_list(ibs, gid_list, downsample, **config)
>>> results_list = list(results_list)
>>> print('result lens = %r' % (map(len, list(results_list))))
>>> print('result[0] = %r' % (len(list(results_list[0][2]))))
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

Yields results (list of dict)

1.1.1.1.14 wbia.algo.detect.ssd module

Interface to SSD object proposals.

wbia.algo.detect.ssd.**detect**(*gpath_list*, *config_filepath*, *weight_filepath*, *class_filepath*, *sensitivity*, *verbose=False*, *use_gpu=True*, *use_gpu_id=0*, ***kwargs*)

Parameters **gpath_list** (*list of str*) – the list of image paths that need proposal candidates

Kwargs (optional): refer to the SSD documentation for configuration settings

Returns iter

wbia.algo.detect.ssd.**detect_gid_list**(*ibs*, *gid_list*, *downsample=True*, *verbose=False*, ***kwargs*)

Parameters

- **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection
- **downsample** (*bool, optional*) – a flag to indicate if the original image sizes should be used; defaults to True

  True: ibs.get_image_detectpaths() is used False: ibs.get_image_paths() is used

Kwargs (optional): refer to the SSD documentation for configuration settings

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection
- **downsample** (*bool, optional*) – a flag to indicate if the original image sizes should be used; defaults to True

Kwargs: detector, config_filepath, weights_filepath, verbose

Yields *tuple* – (gid, gpath, result_list)

CommandLine: python -m wbia.algo.detect.ssd detect_gid_list –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.detect.ssd import *  # NOQA
>>> from wbia.core_images import LocalizerConfig
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids()
>>> config = {'verbose': True}
>>> downsample = False
>>> results_list = detect_gid_list(ibs, gid_list, downsample, **config)
>>> results_list = list(results_list)
>>> print('result lens = %r' % (map(len, list(results_list))))
>>> print('result[0] = %r' % (len(list(results_list[0][2]))))
>>> config = {'verbose': True}
>>> downsample = False
>>> results_list = detect_gid_list(ibs, gid_list, downsample, **config)
>>> results_list = list(results_list)
>>> print('result lens = %r' % (map(len, list(results_list))))
>>> print('result[0] = %r' % (len(list(results_list[0][2]))))
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

Yields results (list of dict)

1.1.1.1.15 wbia.algo.detect.svm module

Interface to Darknet object proposals.

wbia.algo.detect.svm.**classify**(*vector_list*, *weight_filepath*, *verbose=False*, ***kwargs*)

Parameters **thumbail_list** (*list of str*) – the list of image thumbnails that need classifying

Returns iter

wbia.algo.detect.svm.**classify_helper**(*weight_filepath*, *vector_list*, *index_list=None*, *verbose=False*)

1.1.1.1.16 wbia.algo.detect.yolo module

Interface to pydarknet yolo object detection.

wbia.algo.detect.yolo.**detect**(*gpath_list*, *detector=None*, *config_filepath=None*, *weights_filepath=None*, ***kwargs*)

Parameters **gpath_list** (*list of str*) – the list of image paths that need detection

Kwargs (optional): refer to the PyDarknet documentation for configuration settings

Returns iter

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.detect.yolo import *  # NOQA
>>> from wbia.core_images import LocalizerConfig
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='WS_ALL')
>>> gid_list = ibs.images()._rowids[0:1]
>>> gpath_list = ibs.get_image_paths(gid_list)
>>> dpath = '/media/raid/work/WS_ALL/localizer_backup/'
>>> weights_filepath = join(dpath, 'detect.yolo.2.39000.weights')
>>> config_filepath = join(dpath, 'detect.yolo.2.cfg')
>>> config = LocalizerConfig(
>>>     weights_filepath=weights_filepath,
>>>     config_filepath=config_filepath,
>>> )
>>> kwargs = config.asdict()
>>> ut.delete_dict_keys(kwargs, ['weights_filepath', 'config_filepath'])
>>> ut.delete_dict_keys(kwargs, ['thumbnail_cfg', 'species', 'algo'])
```

wbia.algo.detect.yolo.**detect_gid_list**(*ibs*, *gid_list*, *downsample=False*, ***kwargs*)

Parameters

- **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection
- **downsample** (*bool, optional*) – a flag to indicate if the original image sizes should be used; defaults to True

  True: ibs.get_image_detectpaths() is used False: ibs.get_image_paths() is used

Kwargs (optional): refer to the PyDarknet documentation for configuration settings

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **gid_list** (*list of int*) – the list of IBEIS image_rowids that need detection
- **downsample** (*bool, optional*) – a flag to indicate if the original image sizes should be used; defaults to True

Kwargs: detector, config_filepath, weights_filepath, verbose

Yields *tuple* – (gid, gpath, result_list)

CommandLine: python -m wbia.algo.detect.yolo detect_gid_list –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.detect.yolo import *  # NOQA
>>> from wbia.core_images import LocalizerConfig
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='WS_ALL')
>>> gid_list = ibs.images()._rowids[0:1]
>>> kwargs = config = LocalizerConfig(**{
>>>     'weights_filepath': '/media/raid/work/WS_ALL/localizer_backup/detect.yolo.
↪2.39000.weights',
```

(continues on next page)

(continued from previous page)

```
>>>     'config_filepath': '/media/raid/work/WS_ALL/localizer_backup/detect.yolo.
↪2.cfg',
>>> })
>>> exec(ut.execstr_dict(config), globals())
>>> #classes_fpath = '/media/raid/work/WS_ALL/localizer_backup/detect.yolo.2.cfg.
↪classes'
>>> downsample = False
>>> (gid, gpath, result_list) = detect_gid_list(ibs, gid_list, downsample,␣
↪**config)
>>> result = ('(gid, gpath, result_list) = %s' % (ut.repr2((gid, gpath, result_
↪list)),))
>>> print(result)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

Yields results (list of dict)

1.1.1.1.17 Module contents

wbia.algo.detect.**IMPORT_TUPLES = [('grabmodels', None), ('randomforest', None), ('yolo', No**

cd /Users/bluemellophone/code/wbia/wbia/algo/detect makeinit.py –modname=wbia.algo.detect

Type Regen Command

wbia.algo.detect.**reassign_submodule_attributes**(*verbose=True*)

why reloading all the modules doesnt do this I don't know

wbia.algo.detect.**reload_subs**(*verbose=True*)

Reloads wbia.algo.detect and submodules

wbia.algo.detect.**rrrr**(*verbose=True*)

Reloads wbia.algo.detect and submodules

1.1.1.2 wbia.algo.graph package

1.1.1.2.1 Subpackages

1.1.1.2.1.1 wbia.algo.graph.tests package

1.1.1.2.1.2 Submodules

1.1.1.2.1.3 wbia.algo.graph.tests.dyn_cases module

wbia.algo.graph.tests.dyn_cases.**case_all_types**()

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_all_types –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
>>> case_all_types()
```

wbia.algo.graph.tests.dyn_cases.**case_flag_merge**()

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_flag_merge –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
>>> case_flag_merge()
```

wbia.algo.graph.tests.dyn_cases.**case_incon_removes_inference**()

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_incon_removes_inference –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
>>> case_incon_removes_inference()
```

wbia.algo.graph.tests.dyn_cases.**case_inconsistent**()

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_inconsistent –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
>>> case_inconsistent()
```

wbia.algo.graph.tests.dyn_cases.**case_inferable_notcomp1**()
make sure notcomparable edges can be inferred

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_inferable_notcomp1 –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
>>> case_inferable_notcomp1()
```

wbia.algo.graph.tests.dyn_cases.**case_inferable_update_notcomp**()
make sure inference updates for nocomparable edges

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_inferable_update_notcomp –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
>>> case_inferable_update_notcomp()
```

wbia.algo.graph.tests.dyn_cases.**case_keep_in_cc_infr_post_negative**()

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_keep_in_cc_infr_post_negative –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
>>> case_keep_in_cc_infr_post_negative()
```

wbia.algo.graph.tests.dyn_cases.**case_keep_in_cc_infr_post_notcomp**()

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_keep_in_cc_infr_post_notcomp –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
>>> case_keep_in_cc_infr_post_notcomp()
```

wbia.algo.graph.tests.dyn_cases.**case_match_infr**()

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_match_infr –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
```

```
>>> case_match_infr()
```

wbia.algo.graph.tests.dyn_cases.**case_negative_infr**()

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_negative_infr –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
>>> case_negative_infr()
```

wbia.algo.graph.tests.dyn_cases.**case_notcomp_remove_cuts**()

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_notcomp_remove_cuts –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
>>> case_notcomp_remove_cuts()
```

wbia.algo.graph.tests.dyn_cases.**case_notcomp_remove_infr**()

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_notcomp_remove_infr –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
>>> case_notcomp_remove_infr()
```

wbia.algo.graph.tests.dyn_cases.**case_out_of_subgraph_modification**()

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_out_of_subgraph_modification –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
>>> case_out_of_subgraph_modification()
```

wbia.algo.graph.tests.dyn_cases.**case_override_inference**()

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_override_inference –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
>>> case_override_inference()
```

wbia.algo.graph.tests.dyn_cases.**case_redo_incon**()

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_redo_incon –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
>>> case_redo_incon()
```

wbia.algo.graph.tests.dyn_cases.**case_undo_match**()

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_undo_match –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
>>> case_undo_match()
```

wbia.algo.graph.tests.dyn_cases.**case_undo_negative**()

CommandLine: python -m wbia.algo.graph.tests.dyn_cases case_undo_negative –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.tests.dyn_cases import *  # NOQA
>>> case_undo_negative()
```

wbia.algo.graph.tests.dyn_cases.**do_infr_test**(*ccs*, *edges*, *new_edges*)
Creates a graph with *ccs* + *edges* and then adds *new_edges*

1.1.1.2.1.4 wbia.algo.graph.tests.mst_debug module

1.1.1.2.1.5 wbia.algo.graph.tests.test_graph_iden module

1.1.1.2.1.6 wbia.algo.graph.tests.test_neg_metagraph module

TODO: These tests are good and important to run. Ensure they are run via run_tests even though they are not doctests.

Consider moving to pytest and using xdoctest (because regular doctest does not accept the syntax of IBEIS doctests)

wbia.algo.graph.tests.test_neg_metagraph.**test_neg_metagraph_simple_add_remove**()
Test that the negative metagraph tracks the number of negative edges between PCCs through non-label-changing operations

wbia.algo.graph.tests.test_neg_metagraph.**test_neg_metagraph_split_and_merge**()
Test that the negative metagraph tracks the number of negative edges between PCCs through label-changing split and merge operations

wbia.algo.graph.tests.test_neg_metagraph.**test_neg_metagraph_split_incomp**()

wbia.algo.graph.tests.test_neg_metagraph.**test_neg_metagraph_split_neg**()
Test that the negative metagraph tracks the number of negative edges between PCCs through label-changing split operations

1.1.1.2.1.7 Module contents

1.1.1.2.2 Submodules

1.1.1.2.3 wbia.algo.graph.__main__ module

wbia.algo.graph.__main__.**main**()

1.1.1.2.4 wbia.algo.graph.core module

class wbia.algo.graph.core.**AltConstructors**
Bases: object

classmethod from_netx(*G*, *ibs=None*, *verbose=False*, *infer=True*)

classmethod from_pairs(*aid_pairs*, *attrs=None*, *ibs=None*, *verbose=False*)

classmethod from_qreq_(*qreq_*, *cm_list*, *autoinit=False*)
Create a AnnotInference object using a precomputed query / results

status(*extended=False*)

class wbia.algo.graph.core.**AnnotInference**(*ibs*, *aids=[]*, *nids=None*, *autoinit=True*, *verbose=False*)
Bases: utool.util_dev.NiceRepr, *wbia.algo.graph.core.AltConstructors*, *wbia.algo.graph.core.MiscHelpers*, *wbia.algo.graph.core.Feedback*, *wbia.algo.graph.core.NameRelabel*, *wbia.algo.graph.mixin_dynamic.NonDynamicUpdate*, *wbia.algo.graph.mixin_dynamic.Recovery*, *wbia.algo.graph.mixin_dynamic.Consistency*, *wbia.algo.graph.mixin_dynamic.Redundancy*, *wbia.algo.graph.mixin_dynamic.DynamicUpdate*, *wbia.algo.graph.mixin_priority.Priority*, *wbia.algo.graph.mixin_matching.CandidateSearch*, *wbia.algo.graph.mixin_matching.InfrLearning*, *wbia.algo.graph.mixin_matching.AnnotInfrMatching*, *wbia.algo.graph.mixin_helpers.AssertInvariants*, *wbia.algo.graph.mixin_helpers.DummyEdges*, *wbia.algo.graph.mixin_helpers.Convenience*, *wbia.algo.graph.mixin_helpers.AttrAccess*, *wbia.algo.graph.mixin_simulation.SimulationHelpers*, *wbia.algo.graph.mixin_loops.InfrReviewers*, *wbia.algo.graph.mixin_loops.InfrLoops*, *wbia.algo.graph.mixin_viz.GraphVisualization*, *wbia.algo.graph.mixin_groundtruth.Groundtruth*, *wbia.algo.graph.mixin_wbia.IBEISIO*, *wbia.algo.graph.mixin_wbia.IBEISGroundtruth*

class for maintaining state of an identification

Terminology and Concepts:

CommandLine: wbia make_qt_graph_interface –show –aids=1,2,3,4,5,6,7 wbia AnnotInference:0 –show wbia AnnotInference:1 –show wbia AnnotInference:2 –show

wbia AnnotInference:0 –loginfr

Doctest:

```
>>> from wbia.algo.graph.core import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> aids = [1, 2, 3, 4, 5, 6]
>>> infr = AnnotInference(ibs, aids, autoinit=True, verbose=1000)
>>> result = ('infr = %s' % (infr,))
>>> print(result)
>>> ut.quit_if_noshow()
>>> use_image = True
>>> infr.initialize_visual_node_attrs()
>>> # Note that there are initially no edges
>>> infr.show_graph(use_image=use_image)
>>> ut.show_if_requested()
infr = <AnnotInference(nNodes=6, nEdges=0, nCCs=6)>
```

Example

```
>>> # SCRIPT
>>> from wbia.algo.graph.core import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> aids = [1, 2, 3, 4, 5, 6, 7, 9]
>>> infr = AnnotInference(ibs, aids, autoinit=True)
>>> result = ('infr = %s' % (infr,))
>>> print(result)
>>> ut.quit_if_noshow()
>>> use_image = False
>>> infr.initialize_visual_node_attrs()
>>> # Note that there are initially no edges
>>> infr.show_graph(use_image=use_image)
>>> # But we can add nodes between the same names
>>> infr.ensure_mst()
>>> infr.show_graph(use_image=use_image)
>>> # Add some feedback
>>> infr.add_feedback((1, 4), NEGTV)
>>> infr.apply_feedback_edges()
>>> infr.show_graph(use_image=use_image)
>>> ut.show_if_requested()
```

Example

```
>>> # SCRIPT
>>> from wbia.algo.graph.core import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> aids = [1, 2, 3, 4, 5, 6, 7, 9]
>>> infr = AnnotInference(ibs, aids, autoinit=True)
>>> result = ('infr = %s' % (infr,))
>>> print(result)
>>> ut.quit_if_noshow()
>>> use_image = False
>>> infr.initialize_visual_node_attrs()
>>> infr.ensure_mst()
>>> # Add some feedback
>>> infr.add_feedback((1, 4), NEGTV)
>>> try:
>>>     infr.add_feedback((1, 10), NEGTV)
>>> except ValueError:
>>>     pass
>>> try:
>>>     infr.add_feedback((11, 12), NEGTV)
>>> except ValueError:
>>>     pass
>>> infr.apply_feedback_edges()
>>> infr.show_graph(use_image=use_image)
>>> ut.show_if_requested()
```

Ignore:

```
>>> import wbia
>>> import utool as ut
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> infr = wbia.AnnotInference(ibs, 'all')
>>> class_ = infr
>>> fpath = None
>>> static_attrs = ut.check_static_member_vars(class_, fpath)
>>> uninitialized = set(infr.__dict__.keys()) - set(static_attrs)
```

copy()

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

set_config(*config*, ***kw*)

subgraph(*aids*)
Makes a new inference object that is a subset of the original.

Note, this is not robust, be careful. The subgraph should be treated as read only. Do not commit any reviews made from here.

subparams(*prefix*)
Returns dict of params prefixed with <prefix>. The returned dict does not contain the prefix

Doctest:

```
>>> from wbia.algo.graph.core import *
>>> import wbia
>>> infr = wbia.AnnotInference(None)
>>> result = ut.repr2(infr.subparams('refresh'))
>>> print(result)
{'method': 'binomial', 'patience': 72, 'thresh': 0.052, 'window': 20}
```

class wbia.algo.graph.core.**Feedback**
Bases: object

add_feedback(*edge*, *evidence_decision=None*, *tags=None*, *user_id=None*, *meta_decision=None*, *confidence=None*, *timestamp_c1=None*, *timestamp_c2=None*, *timestamp_s1=None*, *timestamp=None*, *verbose=None*, *priority=None*)

Doctest:

```
>>> from wbia.algo.graph.core import *  # NOQA
>>> infr = testdata_infr('testdb1')
>>> infr.add_feedback((5, 6), POSTV)
>>> infr.add_feedback((5, 6), NEGTV, tags=['photobomb'])
>>> infr.add_feedback((1, 2), INCMP)
>>> print(ut.repr2(infr.internal_feedback, nl=2))
>>> assert len(infr.external_feedback) == 0
>>> assert len(infr.internal_feedback) == 2
>>> assert len(infr.internal_feedback[(5, 6)]) == 2
>>> assert len(infr.internal_feedback[(1, 2)]) == 1
```

add_feedback_from(*items*, *verbose=None*, ***kwargs*)

add_node_feedback(*aid*, ***attrs*)

all_feedback()

all_feedback_items()

apply_feedback_edges()
Transforms the feedback dictionaries into nx graph edge attributes

CommandLine: python -m wbia.algo.graph.core apply_feedback_edges

Doctest:

```
>>> from wbia.algo.graph.core import *  # NOQA
>>> infr = testdata_infr('testdb1')
>>> infr.reset_feedback()
>>> infr.params['inference.enabled'] = False
>>> #infr.add_feedback((1, 2), 'unknown', tags=[])
>>> infr.add_feedback((1, 2), INCMP, tags=[])
>>> infr.apply_feedback_edges()
>>> print('edges = ' + ut.repr4(dict(infr.graph.edges)))
>>> result = str(infr)
>>> print(result)
<AnnotInference(nNodes=6, nEdges=3, nCCs=4)>
```

clear_edges()
Removes all edges from the graph

clear_feedback(*edges=None*)
Delete all edges properties related to feedback

clear_name_labels()
Sets all annotation node name labels to be unknown

edge_decision(*edge*)
Gets a decision on an edge, either explicitly or implicitly

CommandLine: python -m wbia.algo.graph.core edge_decision

Doctest:

```
>>> from wbia.algo.graph.core import *  # NOQA
>>> from wbia.algo.graph import demo
>>> infr = demo.demodata_infr(num_pccs=1, p_incon=1)
>>> decision = infr.edge_decision((1, 2))
>>> print('decision = %r' % (decision,))
>>> assert decision == POSTV
>>> decision = infr.edge_decision((199, 299))
>>> print('decision = %r' % (decision,))
>>> assert decision == UNREV
```

edge_decision_from(*edges*)
Gets a decision for multiple edges

feedback_data_keys = ['evidence_decision', 'tags', 'user_id', 'meta_decision', 'timest

feedback_keys = ['evidence_decision', 'tags', 'user_id', 'meta_decision', 'timestamp_c

reset(*state='empty'*)
Removes all edges from graph and resets name labels.

Ignore:

```
>>> from wbia.algo.graph.core import *  # NOQA
>>> from wbia.algo.graph import demo
>>> infr = demo.demodata_infr(num_pccs=5)
>>> assert len(list(infr.edges())) > 0
```

(continues on next page)

(continued from previous page)

```
>>> infr.reset(state='empty')
>>> assert len(list(infr.edges())) == 0
```

reset_feedback(*mode='annotmatch'*, *apply=True*)
Resets feedback edges to state of the SQL annotmatch table

reset_name_labels()
Resets all annotation node name labels to their initial values

class wbia.algo.graph.core.**MiscHelpers**
Bases: object

add_aids(*aids*, *nids=None*)

CommandLine: python -m wbia.algo.graph.core add_aids –show

Doctest:

```
>>> from wbia.algo.graph.core import *  # NOQA
>>> aids_ = [1, 2, 3, 4, 5, 6, 7, 9]
>>> infr = AnnotInference(ibs=None, aids=aids_, autoinit=True)
>>> aids = [2, 22, 7, 9, 8]
>>> nids = None
>>> infr.add_aids(aids, nids)
>>> result = infr.aids
>>> print(result)
>>> assert len(infr.graph) == len(infr.aids)
...
[1, 2, 3, 4, 5, 6, 7, 9, 22, 8]
```

dump_logs()

initialize_graph(*graph=None*)

latest_logs(*colored=False*)

log_message(*msg*, *level=1*, *color=None*)

print(*msg*, *level=1*, *color=None*)

remove_aids(*aids*)
Remove annotations from the graph. :returns: split: indicates which PCCs were split by this action. :rtype: dict

---

Note: This may cause unintended splits!

Ignore:

```
>>> from graphid import demo, util
>>> infr = demo.demodata_infr(num_pccs=5, pos_redun=1)
>>> infr.refresh_candidate_edges()
>>> infr.pin_node_layout()
>>> before = infr.copy()
>>> aids = infr.aids[::5]
>>> splits = infr.remove_aids(aids)
>>> assert len(splits['old']) > 0
>>> infr.assert_invariants()
>>> # xdoc: +REQUIRES(--show)
```

(continues on next page)

(continued from previous page)

```
>>> util.qtensure()
>>> after = infr
>>> before.show(fnum=1, pnum=(1, 2, 1), pickable=True)
>>> after.show(fnum=1, pnum=(1, 2, 2), pickable=True)
```

update_node_attributes(*aids=None*, *nids=None*)

class wbia.algo.graph.core.**NameRelabel**

Bases: object

connected_component_status()

Returns num_inconsistent, num_names_max

Return type dict

CommandLine: python -m wbia.algo.graph.core connected_component_status

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.graph.core import *  # NOQA
>>> infr = testdata_infr('testdb1')
>>> infr.add_feedback_from([(2, 3, NEGTV), (5, 6, NEGTV), (1, 2, POSTV)])
>>> status = infr.connected_component_status()
>>> print(ut.repr3(status))
```

node_label(*aid*)

node_labels(**aids*)

relabel_using_reviews(*graph=None*, *rectify=True*)

Relabels nodes in graph based on positive connected components

This will change all of the names on the nodes to be consistent while preserving any existing names as best as possible. If rectify=False, this will be faster, but the old names will not be preserved and each PCC will be assigned an arbitrary name.

Note: if something messes up you can call infr.reset_labels_to_wbia() to reset node labels to their original values — this will almost always put the graph in an inconsistent state — but then you can this with rectify=True to fix everything up.

Parameters

- **graph** (*nx.Graph, optional*) – only edges in *graph* are relabeled defaults to current graph.
- **rectify** (*bool, optional*) – if True names attempt to remain consistent otherwise there are no restrictions on name labels other than that they are distinct.

wbia.algo.graph.core.**testdata_infr**(*defaultdb='PZ_MTEST'*)

1.1.1.2.5 wbia.algo.graph.demo module

TODO: separate out the tests and make this file just generate the demo data

class wbia.algo.graph.demo.**DummyVerif**(*infr*)

Bases: object

generates dummy scores between edges (not necesarilly in the graph)

CommandLine: python -m wbia.algo.graph.demo DummyVerif:1

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.demo import *  # NOQA
>>> from wbia.algo.graph import demo
>>> import networkx as nx
>>> kwargs = dict(num_pccs=6, p_incon=.5, size_std=2)
>>> infr = demo.demodata_infr(**kwargs)
>>> infr.dummy_verif.predict_edges([(1, 2)])
>>> infr.dummy_verif.predict_edges([(1, 21)])
>>> assert len(infr.dummy_verif.infr.task_probs['match_state']) == 2
```

dummy_ranker(*u*, *K=10*)

simulates the ranking algorithm. Order is defined using the dummy vsone scores, but tests are only applied to randomly selected gt and gf pairs. So, you usually will get a gt result, but you might not if all the scores are bad.

find_candidate_edges(*K=10*)

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.demo import *  # NOQA
>>> from wbia.algo.graph import demo
>>> import networkx as nx
>>> kwargs = dict(num_pccs=40, size=2)
>>> infr = demo.demodata_infr(**kwargs)
>>> edges = list(infr.dummy_verif.find_candidate_edges(K=100))
>>> scores = np.array(infr.dummy_verif.predict_edges(edges))
```

predict_edges(*edges*)

predict_proba_df(*edges*)

CommandLine: python -m wbia.algo.graph.demo DummyVerif.predict_edges

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.demo import *  # NOQA
>>> from wbia.algo.graph import demo
>>> import networkx as nx
>>> kwargs = dict(num_pccs=40, size=2)
>>> infr = demo.demodata_infr(**kwargs)
```

(continues on next page)

(continued from previous page)

```
>>> verif = infr.dummy_verif
>>> edges = list(infr.graph.edges())
>>> probs = verif.predict_proba_df(edges)
>>> #print('scores = %r' % (scores,))
>>> #hashid = ut.hash_data(scores)
>>> #print('hashid = %r' % (hashid,))
>>> #assert hashid == 'cdlkytilfeqgmtsihvhqwffmhczqmpil'
```

show_score_probs()

CommandLine: python -m wbia.algo.graph.demo DummyVerif.show_score_probs –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.demo import *  # NOQA
>>> import wbia
>>> infr = wbia.AnnotInference(None)
>>> verif = DummyVerif(infr)
>>> verif.show_score_probs()
>>> ut.show_if_requested()
```

wbia.algo.graph.demo.**apply_dummy_viewpoints**(*infr*)

wbia.algo.graph.demo.**demo2**()

CommandLine: python -m wbia.algo.graph.demo demo2 –viz python -m wbia.algo.graph.demo demo2

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.graph.demo import *  # NOQA
>>> result = demo2()
>>> print(result)
```

wbia.algo.graph.demo.**demodata_infr**(***kwargs*)

kwargs = {}

CommandLine: python -m wbia.algo.graph.demo demodata_infr –show python -m wbia.algo.graph.demo demodata_infr –num_pccs=25 python -m wbia.algo.graph.demo demodata_infr –profile –num_pccs=100

Ignore:

```
>>> from wbia.algo.graph.demo import *  # NOQA
>>> from wbia.algo.graph import demo
>>> import networkx as nx
>>> kwargs = dict(num_pccs=6, p_incon=.5, size_std=2)
>>> kwargs = ut.argparse_dict(kwargs)
>>> infr = demo.demodata_infr(**kwargs)
>>> pccs = list(infr.positive_components())
>>> assert len(pccs) == kwargs['num_pccs']
>>> nonfull_pccs = [cc for cc in pccs if len(cc) > 1 and nx.is_empty(nx.
↪complement(infr.pos_graph.subgraph(cc)))]
>>> expected_n_incon = len(nonfull_pccs) * kwargs['p_incon']
>>> n_incon = len(list(infr.inconsistent_components()))
```

(continues on next page)

(continued from previous page)

```
>>> # TODO can test that we our sample num incon agrees with pop mean
>>> #sample_mean = n_incon / len(nonfull_pccs)
>>> #pop_mean = kwargs['p_incon']
>>> print('status = ' + ut.repr4(infr.status(extended=True)))
>>> ut.quit_if_noshow()
>>> infr.show(pickable=True, groupby='name_label')
>>> ut.show_if_requested()
```

Ignore:

kwargs = { 'ccs': [[1, 2, 3], [4, 5]]

}

wbia.algo.graph.demo.**demodata_infr2**(*defaultdb='PZ_MTEST'*)

wbia.algo.graph.demo.**demodata_mtest_infr**(*state='empty'*)

wbia.algo.graph.demo.**get_edge_truth**(*infr*, *n1*, *n2*)

wbia.algo.graph.demo.**make_demo_infr**(*ccs*, *edges=[]*, *nodes=[]*, *infer=True*)
Depricate in favor of demodata_infr

wbia.algo.graph.demo.**make_dummy_infr**(*annots_per_name*)

wbia.algo.graph.demo.**randn**(*mean=0*, *std=1*, *shape=[]*, *a_max=None*, *a_min=None*, *rng=None*)

1.1.1.2.6 wbia.algo.graph.mixin_dynamic module

Todo: Negative bookkeeping, needs a small re-organization fix. MOVE FROM neg_redun_metagraph TO neg_metagraph

Instead of maintaining a graph that contains PCCS which are neg redundant to each other, the graph should maintain PCCs that have ANY negative edge between them (aka 1 neg redundant). Then that edge should store a flag indicating the strength / redundancy of that connection. A better idea might be to store both neg_redun_metagraph AND neg_metagraph.

TODO: this (all neg-redun functionality can be easilly consolidated into the neg-metagraph-update. note, we have to allow inconsistent pccs to be in the neg redun graph, we just filter them out afterwords)

---

class wbia.algo.graph.mixin_dynamic.**Consistency**
Bases: object

consistent_components(*graph=None*)
Generates consistent PCCs. These PCCs contain no internal negative edges.

Yields *cc* – set: nodes within the PCC

inconsistent_components(*graph=None*)
Generates inconsistent PCCs. These PCCs contain internal negative edges indicating an error exists.

is_consistent(*cc*)
Determines if a PCC contains inconsistencies

Parameters **cc** (*set*) – nodes in a PCC

Returns bool: returns True unless cc contains any negative edges

Return type flag

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph import demo
>>> infr = demo.demodata_infr(num_pccs=1, p_incon=1)
>>> assert not infr.is_consistent(next(infr.positive_components()))
>>> infr = demo.demodata_infr(num_pccs=1, p_incon=0)
>>> assert infr.is_consistent(next(infr.positive_components()))
```

positive_components(*graph=None*)

Generates the positive connected compoments (PCCs) in the graph These will contain both consistent and inconsinstent PCCs.

Yields *cc* – set: nodes within the PCC

class wbia.algo.graph.mixin_dynamic.**DynamicUpdate**

Bases: object

12 total possible states

details of these states. POSITIVE, WITHIN, CONSISTENT

- pos-within never changes PCC status
- never introduces inconsistency
- might add pos-redun

POSITIVE, WITHIN, INCONSISTENT

- pos-within never changes PCC status
- might fix inconsistent edge

POSITIVE, BETWEEN, BOTH_CONSISTENT

- pos-between edge always does merge

POSITIVE, BETWEEN, ANY_INCONSISTENT

- pos-between edge always does merge
- pos-between never fixes inconsistency

NEGATIVE, WITHIN, CONSISTENT

- might split PCC, results will be consistent
- might causes an inconsistency

NEGATIVE, WITHIN, INCONSISTENT

- might split PCC, results may be inconsistent

NEGATIVE, BETWEEN, BOTH_CONSISTENT

- might add neg-redun

NEGATIVE, BETWEEN, ANY_INCONSISTENT

- might add to incon-neg-external
- neg-redun not tracked for incon.

UNINFERABLE, WITHIN, CONSISTENT

- might remove pos-redun

- might split PCC, results will be consistent

UNINFERABLE, WITHIN, INCONSISTENT

- might split PCC, results may be inconsistent

UNINFERABLE, BETWEEN, BOTH_CONSISTENT

- might remove neg-redun

UNINFERABLE, BETWEEN, ANY_INCONSISTENT

- might remove incon-neg-external

add_review_edge(*edge*, *decision*)
Adds edge to the dynamically connected graphs and updates dynamically inferrable edge attributes.

ensure_edges_from(*edges*)
Finds edges that don't exist and adds them as unreviwed edges. Returns new edges that were added.

on_between(*edge*, *decision*, *prev_decision*, *nid1*, *nid2*, *merge_nid=None*)
Callback when a review is made between two PCCs

on_within(*edge*, *decision*, *prev_decision*, *nid*, *split_nids=None*)
Callback when a review is made inside a PCC

class wbia.algo.graph.mixin_dynamic.**NonDynamicUpdate**
Bases: object

apply_nondynamic_update(*graph=None*)
Recomputes all dynamic bookkeeping for a graph in any state. This ensures that subsequent dyanmic inference can be applied.

CommandLine: python -m wbia.algo.graph.mixin_dynamic apply_nondynamic_update

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.mixin_dynamic import *  # NOQA
>>> from wbia.algo.graph import demo
>>> num_pccs = 250
>>> kwargs = dict(num_pccs=100, p_incon=.3)
>>> infr = demo.demodata_infr(infer=False, **kwargs)
>>> graph = None
>>> infr.apply_nondynamic_update()
>>> infr.assert_neg_metagraph()
```

categorize_edges(*graph=None*, *ne_to_edges=None*)
Non-dynamically computes the status of each edge in the graph. This is can be used to verify the dynamic computations and update when the dynamic state is lost.

CommandLine: python -m wbia.algo.graph.mixin_dynamic categorize_edges –profile

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.mixin_dynamic import *  # NOQA
>>> from wbia.algo.graph import demo
>>> num_pccs = 250 if ut.get_argflag('--profile') else 100
```

(continues on next page)

(continued from previous page)

```
>>> kwargs = dict(num_pccs=100, p_incon=.3)
>>> infr = demo.demodata_infr(infer=False, **kwargs)
>>> graph = None
>>> cat = infr.categorize_edges()
```

collapsed_meta_edges(*graph=None*)
Collapse the grah such that each PCC is a node. Get a list of edges within/between each PCC.

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

class wbia.algo.graph.mixin_dynamic.**Recovery**
Bases: object

recovery funcs

hypothesis_errors(*pos_subgraph*, *neg_edges*)

is_recovering(*edge=None*)
Checks to see if the graph is inconsinsistent.

Parameters **edge** (*None*) – If None, then returns True if the graph contains any inconsistency. Otherwise, returns True if the edge is related to an inconsistent component via a positive or negative connection.

Returns flag

Return type bool

CommandLine: python -m wbia.algo.graph.mixin_dynamic is_recovering

Doctest:

```
>>> from wbia.algo.graph.mixin_dynamic import *  # NOQA
>>> from wbia.algo.graph import demo
>>> infr = demo.demodata_infr(num_pccs=4, size=4, ignore_pair=True)
>>> infr.ensure_cliques(meta_decision=SAME)
>>> a, b, c, d = map(list, infr.positive_components())
>>> assert infr.is_recovering() is False
>>> infr.add_feedback((a[0], a[1]), NEGTV)
>>> assert infr.is_recovering() is True
>>> assert infr.is_recovering((a[2], a[3])) is True
>>> assert infr.is_recovering((a[3], b[0])) is True
>>> assert infr.is_recovering((b[0], b[1])) is False
>>> infr.add_feedback((a[3], b[2]), NEGTV)
>>> assert infr.is_recovering((b[0], b[1])) is True
>>> assert infr.is_recovering((c[0], d[0])) is False
>>> infr.add_feedback((b[2], c[0]), NEGTV)
>>> assert infr.is_recovering((c[0], d[0])) is False
>>> result = ut.repr4({
>>>     'pccs': sorted(list(infr.positive_components())),
>>>     'iccs': sorted(list(infr.inconsistent_components())),
>>> }, nobr=True, si=True, itemsep='')
>>> print(result)
iccs: [{1,2,3,4}],
pccs: [{5,6,7,8},{9,10,11,12},{13,14,15,16},{1,2,3,4}],
```

maybe_error_edges()

class wbia.algo.graph.mixin_dynamic.**Redundancy**
Bases: wbia.algo.graph.mixin_dynamic._RedundancyComputers

methods for dynamic redundancy book-keeping

filter_edges_flagged_as_redun(*edges*)
Returns only edges that are not flagged as redundant. Uses bookkeeping structures

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.mixin_dynamic import *  # NOQA
>>> from wbia.algo.graph import demo
>>> infr = demo.demodata_infr(num_pccs=1, size=4)
>>> infr.clear_edges()
>>> infr.ensure_cliques()
>>> infr.clear_feedback()
>>> print(ut.repr4(infr.status()))
>>> nonredun_edges = list(infr.filter_edges_flagged_as_redun(
>>>     infr.unreviewed_graph.edges()))
>>> assert len(nonredun_edges) == 6
```

is_flagged_as_redun(*edge*)
Tests redundancy against bookkeeping structure against cache

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

update_extern_neg_redun(*nid*, *may_add=True*, *may_remove=True*, *force=False*)
Checks if *nid* is negative redundant to any other *cc* it has at least one negative review to. (TODO: NEG REDUN CAN BE CONSOLIDATED VIA NEG-META-GRAPH)

update_neg_redun_to(*nid1*, *other_nids*, *may_add=True*, *may_remove=True*, *force=False*)
Checks if nid1 is neg redundant to other_nids. Edges are either removed or added to the queue appropriately. (TODO: NEG REDUN CAN BE CONSOLIDATED VIA NEG-META-GRAPH)

update_pos_redun(*nid*, *may_add=True*, *may_remove=True*, *force=False*)
Checks if a PCC is newly, or no longer positive redundant. Edges are either removed or added to the queue appropriately.

1.1.1.2.7 wbia.algo.graph.mixin_groundtruth module

class wbia.algo.graph.mixin_groundtruth.**Groundtruth**
Bases: object

apply_edge_truth(*edges=None*)

edge_attr_df(*key*, *edges=None*, *default=NoParam*)
constructs DataFrame using current predictions

is_comparable(*aid_pairs*, *allow_guess=True*)
Guesses by default when real comparable information is not available.

is_photobomb(*aid_pairs*)

is_same(*aid_pairs*)

match_state_df(*index*)
Returns groundtruth state based on wbia controller

match_state_gt(*edge*)

1.1.1.2.8 wbia.algo.graph.mixin_helpers module

class wbia.algo.graph.mixin_helpers.**AssertInvariants**
Bases: object

assert_consistency_invariant(*msg=''*)

assert_disjoint_invariant(*msg=''*)

assert_edge(*edge*)

assert_invariants(*msg=''*)

assert_neg_metagraph()
Checks that the negative metgraph is correctly book-kept.

assert_recovery_invariant(*msg=''*)

assert_union_invariant(*msg=''*)

class wbia.algo.graph.mixin_helpers.**AttrAccess**
Bases: object

Contains non-core helper functions

edges(*data=False*)

gen_edge_attrs(*key*, *edges=None*, *default=NoParam*, *on_missing=None*)
maybe change to gen edge items

gen_edge_values(*key*, *edges=None*, *default=NoParam*, *on_missing='error'*, *on_keyerr='default'*)

gen_node_attrs(*key*, *nodes=None*, *default=NoParam*)

gen_node_values(*key*, *nodes*, *default=NoParam*)

get_annot_attrs(*key*, *aids*)
Wrapper around get_node_attrs specific to annotation nodes

get_edge_attr(*edge*, *key*, *default=NoParam*, *on_missing='error'*)
single edge getter helper

get_edge_attrs(*key*, *edges=None*, *default=NoParam*, *on_missing=None*)
Networkx edge getter helper

get_edge_data(*edge*)

get_edge_dataframe(*edges=None*, *all=False*)

get_edge_df_text(*edges=None*, *highlight=True*)

get_edges_where_eq(*key*, *val*, *edges=None*, *default=NoParam*, *on_missing=None*)

get_edges_where_ne(*key*, *val*, *edges=None*, *default=NoParam*, *on_missing=None*)

get_node_attrs(*key*, *nodes=None*, *default=NoParam*)
Networkx node getter helper

get_nonvisual_edge_data(*edge*, *on_missing='filter'*)

has_edge(*edge*)

set_edge_attr(*edge*, *attr*)
single edge setter helper

set_edge_attrs(*key*, *edge_to_prop*)
Networkx edge setter helper

set_node_attrs(*key*, *node_to_prop*)
Networkx node setter helper

class wbia.algo.graph.mixin_helpers.**Convenience**
Bases: object

static e_(*u*, *v*)

edge_tag_hist()

incomp_graph

neg_graph

node_tag_hist()

pair_connection_info(*aid1*, *aid2*)
Helps debugging when ibs.nids has info that annotmatch/staging do not

Example

```
>>> # # FIXME failing-test (22-Jul-2020) GZ_Master1 doesn't exist
>>> # xdoctest: +SKIP
>>> from wbia.algo.graph.mixin_helpers import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='GZ_Master1')
>>> infr = wbia.AnnotInference(ibs, 'all', autoinit=True)
>>> infr.reset_feedback('staging', apply=True)
>>> infr.relabel_using_reviews(rectify=False)
>>> aid1, aid2 = 1349, 3087
>>> aid1, aid2 = 1535, 2549
>>> infr.pair_connection_info(aid1, aid2)
```

```
>>> aid1, aid2 = 4055, 4286
>>> aid1, aid2 = 6555, 6882
>>> aid1, aid2 = 712, 803
>>> aid1, aid2 = 3883, 4220
>>> infr.pair_connection_info(aid1, aid2)
```

pos_graph

print_graph_connections(*label='orig_name_label'*)
label = 'orig_name_label'

print_graph_info()

print_within_connection_info(*edge=None*, *cc=None*, *aid=None*, *nid=None*)

unknown_graph

unreviewed_graph

class wbia.algo.graph.mixin_helpers.**DummyEdges**
Bases: object

ensure_cliques(*label='name_label'*, *meta_decision=None*)
Force each name label to be a clique.

Parameters

- **label** (*str*) – node attribute to use as the group id to form the cliques.
- **meta_decision** (*str*) – if specified adds clique edges as feedback items with this decision. Otherwise the edges are only explicitly added to the graph.
- **infr** –
- **label** – (default = 'name_label')
- **decision** (*str*) – (default = 'unreviewed')

CommandLine: python -m wbia.algo.graph.mixin_helpers ensure_cliques

Doctest:

```
>>> from wbia.algo.graph.mixin_helpers import *  # NOQA
>>> from wbia.algo.graph import demo
>>> label = 'name_label'
>>> infr = demo.demodata_infr(num_pccs=3, size=5)
>>> print(infr.status())
>>> assert infr.status()['nEdges'] < 33
>>> infr.ensure_cliques()
>>> print(infr.status())
>>> assert infr.status()['nEdges'] == 33
>>> assert infr.status()['nUnrevEdges'] == 12
>>> assert len(list(infr.find_clique_edges(label))) > 0
>>> infr.ensure_cliques(meta_decision=SAME)
>>> assert infr.status()['nUnrevEdges'] == 0
>>> assert len(list(infr.find_clique_edges(label))) == 0
```

ensure_full()
Explicitly places all edges, but does not make any feedback items

ensure_mst(*label='name_label'*, *meta_decision='same'*)
Ensures that all names are names are connected.

Parameters

- **label** (*str*) – node attribute to use as the group id to form the mst.
- **meta_decision** (*str*) – if specified adds clique edges as feedback items with this decision. Otherwise the edges are only explicitly added to the graph. This makes feedback items with user_id=algo:mst and with a confidence of guessing.

Ignore: annots = ibs.annots(infr.aids) def fix_name(n):

import re n = re.sub(' ', ' ', n) return re.sub(' *-? *BBQ[0-9]', '', n)

ut.fix_embed_globals() new_names = [fix_name(n) for n in annots.names] set(new_names)

annots.names = new_names

infr.set_node_attrs('name_fix', ut.dzip(infr.aids, new_names)) label = 'name_fix' infr.ensure_mst(label)

infr.set_node_attrs('name_label', ut.dzip(infr.aids, annots.nids))

Ignore: label = 'name_label'

Doctest:

```
>>> from wbia.algo.graph.mixin_dynamic import *  # NOQA
>>> from wbia.algo.graph import demo
>>> infr = demo.demodata_infr(num_pccs=3, size=4)
>>> assert infr.status()['nCCs'] == 3
>>> infr.clear_edges()
>>> assert infr.status()['nCCs'] == 12
>>> infr.ensure_mst()
>>> assert infr.status()['nCCs'] == 3
```

Doctest:

```
>>> from wbia.algo.graph.mixin_dynamic import *  # NOQA
>>> import wbia
>>> infr = wbia.AnnotInference('PZ_MTEST', 'all', autoinit=True)
>>> infr.reset_feedback('annotmatch', apply=True)
>>> assert infr.status()['nInconsistentCCs'] == 0
>>> assert infr.status()['nCCs'] == 41
>>> label = 'name_label'
>>> new_edges = infr.find_mst_edges(label=label)
>>> assert len(new_edges) == 0
>>> infr.clear_edges()
>>> assert infr.status()['nCCs'] == 119
>>> infr.ensure_mst()
>>> assert infr.status()['nCCs'] == 41
```

find_clique_edges(*label='name_label'*)

Augmenting edges that would complete each the specified cliques. (based on the group inferred from *label*)

Parameters **label** (*str*) – node attribute to use as the group id to form the cliques.

find_connecting_edges()

Searches for a small set of edges, which if reviewed as positive would ensure that each PCC is k-connected. Note that in somes cases this is not possible

find_mst_edges(*label='name_label'*)

Returns edges to augment existing PCCs (by label) in order to ensure they are connected with positive edges.

CommandLine: python -m wbia.algo.graph.mixin_helpers find_mst_edges –profile

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.mixin_helpers import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> infr = wbia.AnnotInference(ibs, 'all', autoinit=True)
>>> label = 'orig_name_label'
>>> label = 'name_label'
>>> infr.find_mst_edges()
>>> infr.ensure_mst()
```

Ignore:

old_mst_edges = [e for e, d in infr.edges(data=True) if d.get('user_id', None) == 'algo:mst'

] infr.graph.remove_edges_from(old_mst_edges) infr.pos_graph.remove_edges_from(old_mst_edges) infr.neg_graph.remove_edges_from(old_mst_edges) infr.incomp_graph.remove_edges_from(old_mst_edges)

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

1.1.1.2.9 wbia.algo.graph.mixin_loops module

class wbia.algo.graph.mixin_loops.**InfrLoops**
Bases: object

Algorithm control flow loops

hardcase_review_gen()
Subiterator for hardcase review

Re-review non-confident edges that vsone did not classify correctly

incon_recovery_gen()
Subiterator for recovery mode of the mainm algorithm

Iterates until the graph is consistent

Note: inconsistency recovery is implicitly handled by the main algorithm, so other phases do not need to call this explicitly. This exists for the case where the only mode we wish to run is inconsistency recovery.

---

init_refresh()

main_gen(*max_loops=None*, *use_refresh=True*)
The main outer loop.

This function is designed as an iterator that will execute the graph algorithm main loop as automatically as possible, but if user input is needed, it will pause and yield the decision it needs help with. Once feedback is given for this item, you can continue the main loop by calling next. StopIteration is raised once the algorithm is complete.

Parameters

- **max_loops** (*int*) – maximum number of times to run the outer loop, i.e. ranking is run at most this many times.
- **use_refresh** (*bool*) – allow the refresh criterion to stop the algo

Notes

Different phases of the main loop are implemented as subiterators

CommandLine: python -m wbia.algo.graph.mixin_loops main_gen

Doctest:

```
>>> # xdoctest: +REQUIRES(--slow)
>>> from wbia.algo.graph.mixin_loops import *
>>> from wbia.algo.graph.mixin_simulation import UserOracle
>>> import wbia
>>> infr = wbia.AnnotInference('testdb1', aids='all',
>>>                            autoinit='staging', verbose=4)
```

(continues on next page)

(continued from previous page)

```
>>> infr.params['manual.n_peek'] = 10
>>> infr.params['ranking.ntop'] = 1
>>> infr.oracle = UserOracle(.99, rng=0)
>>> infr.simulation_mode = False
>>> infr.reset()
>>> #infr.load_published()
>>> gen = infr.main_gen()
>>> while True:
>>>     try:
>>>         reviews = next(gen)
>>>         edge, priority, data = reviews[0]
>>>         feedback = infr.request_oracle_review(edge)
>>>         infr.add_feedback(edge, **feedback)
>>>     except StopIteration:
>>>         break
```

main_loop(*max_loops=None*, *use_refresh=True*)
DEPRICATED

use list(infr.main_gen) instead or assert not any(infr.main_gen()) maybe this is fine.

neg_redun_gen()
Subiterator for phase3 of the main algorithm.

Searches for decisions that would commplete negative redundancy

pos_redun_gen()
Subiterator for phase2 of the main algorithm.

Searches for decisions that would commplete positive redundancy

Doctest:

```
>>> from wbia.algo.graph.mixin_loops import *
>>> import wbia
>>> infr = wbia.AnnotInference('PZ_MTEST', aids='all',
>>>                            autoinit='staging', verbose=4)
>>> #infr.load_published()
>>> gen = infr.pos_redun_gen()
>>> feedback = next(gen)
```

ranked_list_gen(*use_refresh=True*)
Subiterator for phase1 of the main algorithm

Calls the underlying ranking algorithm and prioritizes the results

start_id_review(*max_loops=None*, *use_refresh=None*)

class wbia.algo.graph.mixin_loops.**InfrReviewers**
Bases: object

accept(*feedback*)
Called when user has completed feedback from qt or web

emit_manual_review(*edge*, *priority=None*)
Emits a signal containing edges that need review. The callback should present them to a user, get feedback, and then call on_accpet.

qt_edge_reviewer(*edge=None*)

qt_review_loop()
TODO: The loop parts should be a non-mixin class

Qt review loop entry point

CommandLine: python -m wbia.algo.graph.mixin_loops qt_review_loop –show

Example

```
>>> # SCRIPT
>>> import utool as ut
>>> import wbia
>>> ibs = wbia.opendb('PZ_MTEST')
>>> infr = wbia.AnnotInference(ibs, 'all', autoinit=True)
>>> infr.ensure_mst()
>>> # Add dummy priorities to each edge
>>> infr.set_edge_attrs('prob_match', ut.dzip(infr.edges(), [1]))
>>> infr.prioritize('prob_match', infr.edges(), reset=True)
>>> infr.params['redun.enabled'] = False
>>> win = infr.qt_review_loop()
>>> import wbia.guitool as gt
>>> gt.qtapp_loop(qwin=win, freq=10)
```

request_oracle_review(*edge*, ***kw*)

resume()

skip(*edge*)

try_auto_review(*edge*)

1.1.1.2.10 wbia.algo.graph.mixin_matching module

class wbia.algo.graph.mixin_matching.**AnnotInfrMatching**
Bases: object

Methods for running matching algorithms

apply_match_edges(*review_cfg={}*)
Adds results from one-vs-many rankings as edges in the graph

apply_match_scores()
Applies precomputed matching scores to edges that already exist in the graph. Typically you should run infr.apply_match_edges() before running this.

CommandLine: python -m wbia.algo.graph.core apply_match_scores –show

Example

```
>>> # xdoctest: +REQUIRES(--slow)
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.core import *  # NOQA
>>> infr = testdata_infr('PZ_MTEST')
>>> infr.exec_matching()
>>> infr.apply_match_edges()
>>> infr.apply_match_scores()
>>> infr.get_edge_attrs('score')
```

exec_matching(*qaids=None*, *daids=None*, *prog_hook=None*, *cfgdict=None*, *name_method='node'*, *use_cache=True*, *invalidate_supercache=False*, *batch_size=None*, *ranks_top=5*)
Loads chip matches into the inference structure Uses graph name labeling and ignores wbia labeling

exec_vsone_subset(*edges*, *prog_hook=None*)

Parameters **prog_hook** (*None*) – (default = None)

CommandLine: python -m wbia.algo.graph.core exec_vsone_subset

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.core import *  # NOQA
>>> infr = testdata_infr('testdb1')
>>> infr.ensure_full()
>>> edges = [(1, 2), (2, 3)]
>>> result = infr.exec_vsone_subset(edges)
>>> print(result)
```

lookup_cm(*aid1*, *aid2*)
Get chipmatch object associated with an edge if one exists.

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

class wbia.algo.graph.mixin_matching.**CandidateSearch**
Bases: wbia.algo.graph.mixin_matching._RedundancyAugmentation

Search for candidate edges

add_candidate_edges(*candidate_edges*)

ensure_prioritized(*priority_edges*)

ensure_priority_scores(*priority_edges*)
Ensures that priority attributes are assigned to the edges. This does not change the state of the queue.

Doctest:

```
>>> import wbia
>>> ibs = wbia.opendb('PZ_MTEST')
>>> infr = wbia.AnnotInference(ibs, aids='all')
>>> infr.ensure_mst()
>>> priority_edges = list(infr.edges())[0:1]
>>> infr.ensure_priority_scores(priority_edges)
```

Doctest:

```
>>> import wbia
>>> ibs = wbia.opendb('PZ_MTEST')
>>> infr = wbia.AnnotInference(ibs, aids='all')
>>> infr.ensure_mst()
>>> # infr.load_published()
>>> priority_edges = list(infr.edges())
>>> infr.ensure_priority_scores(priority_edges)
```

Doctest:

```
>>> from wbia.algo.graph import demo
>>> infr = demo.demodata_infr(num_pccs=6, p_incon=.5, size_std=2)
>>> edges = list(infr.edges())
>>> infr.ensure_priority_scores(edges)
```

ensure_task_probs(*edges*)

Ensures that probabilities are assigned to the edges. This gaurentees that infr.task_probs contains data for edges. (Currently only the primary task is actually ensured)

CommandLine: python -m wbia.algo.graph.mixin_matching ensure_task_probs

Doctest:

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.graph.mixin_matching import *
>>> import wbia
>>> infr = wbia.AnnotInference('PZ_MTEST', aids='all',
>>>                            autoinit='staging')
>>> edges = list(infr.edges())[0:3]
>>> infr.load_published()
>>> assert len(infr.task_probs['match_state']) == 0
>>> infr.ensure_task_probs(edges)
>>> assert len(infr.task_probs['match_state']) == 3
>>> infr.ensure_task_probs(edges)
>>> assert len(infr.task_probs['match_state']) == 3
```

Doctest:

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.graph.mixin_matching import *
>>> from wbia.algo.graph import demo
>>> infr = demo.demodata_infr(num_pccs=6, p_incon=.5, size_std=2)
>>> edges = list(infr.edges())
>>> infr.ensure_task_probs(edges)
>>> assert all([np.isclose(sum(p.values()), 1)
>>>             for p in infr.task_probs['match_state'].values()])
```

find_lnbnn_candidate_edges(*desired_states=['unreviewed']*, *can_match_samename=False*, *can_match_sameimg=False*, *K=5*, *Knorm=5*, *requery=True*, *prescore_method='csum'*, *score_method='csum'*, *sv_on=True*, *cfgdict_=None*, *batch_size=None*)

Example

```
>>> # DISABLE_DOCTEST
>>> # xdoctest: +REQUIRES(--slow)
>>> from wbia.algo.graph import demo
>>> infr = demo.demodata_mtest_infr()
>>> cand_edges = infr.find_lnbnn_candidate_edges()
>>> assert len(cand_edges) > 200, len(cand_edges)
```

refresh_candidate_edges()

Search for candidate edges. Assign each edge a priority and add to queue.

class wbia.algo.graph.mixin_matching.**InfrLearning**

Bases: object

learn_deploy_verifiers(*publish=False*)
Uses current knowledge to train verifiers for new unseen pairs.

Example

```
>>> # DISABLE_DOCTEST
>>> import wbia
>>> ibs = wbia.opendb('PZ_MTEST')
>>> infr = wbia.AnnotInference(ibs, aids='all')
>>> infr.ensure_mst()
>>> publish = False
>>> infr.learn_deploy_verifiers()
```

Ignore: publish = True

learn_evaluation_verifiers()
Creates a cross-validated ensemble of classifiers to evaluate verifier error cases and groundtruth errors.

CommandLine: python -m wbia.algo.graph.mixin_matching learn_evaluation_verifiers

Doctest:

```
>>> # xdoctest: +REQUIRES(module:wbia_cnn, --slow)
>>> import wbia
>>> infr = wbia.AnnotInference(
>>>     'PZ_MTEST', aids='all', autoinit='annotmatch',
>>>     verbose=4)
>>> verifiers = infr.learn_evaluation_verifiers()
>>> edges = list(infr.edges())
>>> verif = verifiers['match_state']
>>> probs = verif.predict_proba_df(edges)
>>> print(probs)
```

load_latest_classifiers(*dpath*)

load_published()
Downloads, caches, and loads pre-trained verifiers. This is the default action.

photobomb_samples()

1.1.1.2.11 wbia.algo.graph.mixin_priority module

class wbia.algo.graph.mixin_priority.**Priority**
Bases: object

Handles prioritization of edges for review.

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.mixin_priority import *  # NOQA
>>> from wbia.algo.graph import demo
>>> infr = demo.demodata_infr(num_pccs=20)
```

confidently_connected(*u*, *v*, *thresh=2*)
Checks if u and v are conneted by edges above a confidence threshold

confidently_separated(*u*, *v*, *thresh=2*)
Checks if u and v are conneted by edges above a confidence threshold

Doctest:

```
>>> from wbia.algo.graph.mixin_priority import *  # NOQA
>>> from wbia.algo.graph import demo
>>> infr = demo.make_demo_infr(ccs=[(1, 2), (3, 4), (5, 6), (7, 8)])
>>> infr.add_feedback((1, 5), NEGTV)
>>> infr.add_feedback((5, 8), NEGTV)
>>> infr.add_feedback((6, 3), NEGTV)
>>> u, v = (1, 4)
>>> thresh = 0
>>> assert not infr.confidently_separated(u, v, thresh)
>>> infr.add_feedback((2, 3), NEGTV)
>>> assert not infr.confidently_separated(u, v, thresh)
```

generate_reviews(*pos_redun=None*, *neg_redun=None*, *data=False*)
Dynamic generator that yeilds high priority reviews

peek()

peek_many(*n*)
Peeks at the top n edges in the queue.

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.mixin_priority import *  # NOQA
>>> from wbia.algo.graph import demo
>>> infr = demo.demodata_infr(num_pccs=7, size=5)
>>> infr.refresh_candidate_edges()
>>> infr.peek_many(50)
```

pop()
Main interface to the priority queue used by the algorithm loops. Pops the highest priority edge from the queue.

prioritize(*metric=None*, *edges=None*, *scores=None*, *force_inconsistent=True*, *reset=False*)
Adds edges to the priority queue

Doctest:

```
>>> from wbia.algo.graph.mixin_priority import *  # NOQA
>>> from wbia.algo.graph import demo
>>> infr = demo.demodata_infr(num_pccs=7, size=5)
>>> infr.ensure_cliques(meta_decision=SAME)
>>> # Add a negative edge inside a PCC
>>> ccs = list(infr.positive_components())
>>> edge1 = tuple(list(ccs[0])[0:2])
>>> edge2 = tuple(list(ccs[1])[0:2])
>>> infr.add_feedback(edge1, NEGTV)
>>> infr.add_feedback(edge2, NEGTV)
>>> num_new = infr.prioritize(reset=True)
>>> order = infr._peek_many(np.inf)
```

(continues on next page)

(continued from previous page)

```
>>> scores = ut.take_column(order, 1)
>>> assert scores[0] > 10
>>> assert len(scores) == num_new, 'should prioritize two hypotheis edges'
>>> unrev_edges = set(infr.unreviewed_graph.edges())
>>> err_edges = set(ut.flatten(infr.nid_to_errors.values()))
>>> edges = set(list(unrev_edges - err_edges)[0:2])
>>> edges.update(list(err_edges)[0:2])
>>> num_new = infr.prioritize(edges=edges, reset=True)
>>> order2 = infr._peek_many(np.inf)
>>> scores2 = np.array(ut.take_column(order2, 1))
>>> assert np.all(scores2[0:2] > 10)
>>> assert np.all(scores2[2:] < 10)
```

Example

import wbia infr = wbia.AnnotInference('PZ_MTEST', aids='all', autoinit='staging') infr.verbose = 1000 infr.load_published() incon_edges = set(ut.iflatten(infr.nid_to_errors.values())) assert len(incon_edges) > 0 edges = list(infr.find_pos_redun_candidate_edges()) assert len(set(incon_edges).intersection(set(edges))) == 0 infr.add_candidate_edges(edges)

infr.prioritize() logger.info(ut.repr4(infr.status()))

push(*edge*, *priority=None*)
Push an edge back onto the queue

reinstate_between_priority(*cc1*, *cc2*)

reinstate_external_priority(*cc*)

reinstate_internal_priority(*cc*)

remaining_reviews()

remove_between_priority(*cc1*, *cc2*)

remove_external_priority(*cc*)

remove_internal_priority(*cc*)

1.1.1.2.12 wbia.algo.graph.mixin_simulation module

Mixin functionality for experiments, tests, and simulations. This includes recordings measures used to generate plots in JC's thesis.

class wbia.algo.graph.mixin_simulation.**SimulationHelpers**
Bases: object

init_simulation(*oracle_accuracy=1.0*, *k_redun=2*, *enable_autoreview=True*, *enable_inference=True*, *classifiers=None*, *match_state_thresh=None*, *pb_state_thresh=None*, *max_outer_loops=None*, *name=None*)

init_test_mode()

measure_error_edges()

measure_metrics()

class wbia.algo.graph.mixin_simulation.**UserOracle**(*accuracy*, *rng*)
Bases: object

review(*edge*, *truth*, *infr*, *accuracy=None*)

1.1.1.2.13 wbia.algo.graph.mixin_viz module

class wbia.algo.graph.mixin_viz.**GraphVisualization**
Bases: object

contains plotting related code

debug_edge_repr()

draw_aids(*aids*, *fnum=None*)

get_colored_edge_weights(*graph=None*, *highlight_reviews=True*)

get_colored_weights(*weights*)

initialize_visual_node_attrs(*graph=None*)

static make_viz_config(*use_image*, *small_graph*)

repr_edge_data(*all_edge_data*, *visual=True*)

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

show(*graph=None*, *use_image=False*, *update_attrs=True*, *with_colorbar=False*, *pnum=(1, 1, 1)*, *zoomable=True*, *pickable=False*, ***kwargs*)

Parameters

- **infr** –
- **graph** (*None*) – (default = None)
- **use_image** (*bool*) – (default = False)
- **update_attrs** (*bool*) – (default = True)
- **with_colorbar** (*bool*) – (default = False)
- **pnum** (*tuple*) – plot number(default = (1, 1, 1))
- **zoomable** (*bool*) – (default = True)
- **pickable** (*bool*) – (de = False)
- ****kwargs** – verbose, with_labels, fnum, layout, ax, pos, img_dict, title, layoutkw, framewidth, modify_ax, as_directed, hacknoedge, hacknode, node_labels, arrow_width, fontsize, fontweight, fontname, fontfamilty, fontproperties

CommandLine: python -m wbia.algo.graph.mixin_viz GraphVisualization.show_graph –show

Example

```
>>> # xdoctest: +REQUIRES(module:pygraphviz)
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.mixin_viz import *  # NOQA
>>> from wbia.algo.graph import demo
>>> import wbia.plottool as pt
>>> infr = demo.demodata_infr(ccs=ut.estarmap(
>>>     range, [(1, 6), (6, 10), (10, 13), (13, 15), (15, 16),
```

(continues on next page)

(continued from previous page)

```
>>>             (17, 20)]))
>>> pnum_ = pt.make_pnum_nextgen(nRows=1, nCols=3)
>>> infr.show_graph(show_cand=True, simple_labels=True, pickable=True, fnum=1,
↪ pnum=pnum_())
>>> infr.add_feedback((1, 5), INCMP)
>>> infr.add_feedback((14, 18), INCMP)
>>> infr.refresh_candidate_edges()
>>> infr.show_graph(show_cand=True, simple_labels=True, pickable=True, fnum=1,
↪ pnum=pnum_())
>>> infr.add_feedback((17, 18), NEGTV)  # add inconsistency
>>> infr.apply_nondynamic_update()
>>> infr.show_graph(show_cand=True, simple_labels=True, pickable=True, fnum=1,
↪ pnum=pnum_())
>>> ut.show_if_requested()
```

show_edge(*edge*, *fnum=None*, *pnum=None*, ***kwargs*)

show_error_case(*aids*, *edge=None*, *error_edges=None*, *colorby=None*, *fnum=1*)
Example

show_graph(*graph=None*, *use_image=False*, *update_attrs=True*, *with_colorbar=False*, *pnum=(1, 1, 1)*, *zoomable=True*, *pickable=False*, ***kwargs*)

Parameters

- **infr** –
- **graph** (*None*) – (default = None)
- **use_image** (*bool*) – (default = False)
- **update_attrs** (*bool*) – (default = True)
- **with_colorbar** (*bool*) – (default = False)
- **pnum** (*tuple*) – plot number(default = (1, 1, 1))
- **zoomable** (*bool*) – (default = True)
- **pickable** (*bool*) – (de = False)
- ****kwargs** – verbose, with_labels, fnum, layout, ax, pos, img_dict, title, layoutkw, framewidth, modify_ax, as_directed, hacknoedge, hacknode, node_labels, arrow_width, fontsize, fontweight, fontname, fontfamilty, fontproperties

CommandLine: python -m wbia.algo.graph.mixin_viz GraphVisualization.show_graph –show

Example

```
>>> # xdoctest: +REQUIRES(module:pygraphviz)
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.mixin_viz import *  # NOQA
>>> from wbia.algo.graph import demo
>>> import wbia.plottool as pt
>>> infr = demo.demodata_infr(ccs=ut.estarmap(
>>>     range, [(1, 6), (6, 10), (10, 13), (13, 15), (15, 16),
>>>             (17, 20)]))
>>> pnum_ = pt.make_pnum_nextgen(nRows=1, nCols=3)
>>> infr.show_graph(show_cand=True, simple_labels=True, pickable=True, fnum=1,
↪ pnum=pnum_())
```

(continues on next page)

(continued from previous page)

```
>>> infr.add_feedback((1, 5), INCMP)
>>> infr.add_feedback((14, 18), INCMP)
>>> infr.refresh_candidate_edges()
>>> infr.show_graph(show_cand=True, simple_labels=True, pickable=True, fnum=1,
↪ pnum=pnum_())
>>> infr.add_feedback((17, 18), NEGTV)  # add inconsistency
>>> infr.apply_nondynamic_update()
>>> infr.show_graph(show_cand=True, simple_labels=True, pickable=True, fnum=1,
↪ pnum=pnum_())
>>> ut.show_if_requested()
```

simplify_graph(*graph=None*, *copy=True*)

start_qt_interface(*loop=True*)

update_node_image_attribute(*use_image=False*, *graph=None*)

update_node_image_config(***kwargs*)

update_visual_attrs(*graph=None*, *show_reviewed_edges=True*, *show_unreviewed_edges=False*, *show_inferred_diff=True*, *show_inferred_same=True*, *show_recent_review=False*, *highlight_reviews=True*, *show_inconsistency=True*, *wavy=False*, *simple_labels=False*, *show_labels=True*, *reposition=True*, *use_image=False*, *edge_overrides=None*, *node_overrides=None*, *colorby='name_label'*, ***kwargs*)

visual_edge_attrs
all edge visual attrs

visual_edge_attrs_appearance
attrs that pertain to edge color and style

visual_edge_attrs_space
attrs that pertain to edge positioning in a plot

visual_node_attrs

wbia.algo.graph.mixin_viz.**on_pick**(*event*, *infr=None*)

1.1.1.2.14 wbia.algo.graph.mixin_wbia module

class wbia.algo.graph.mixin_wbia.**IBEISGroundtruth**
Bases: object

Methods for generating training labels for classifiers

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

wbia_guess_if_comparable(*aid_pairs*)
Takes a guess as to which annots are not comparable based on scores and viewpoints. If either viewpoints is null assume they are comparable.

wbia_is_comparable(*aid_pairs*, *allow_guess=True*)
Guesses by default when real comparable information is not available.

wbia_is_photobomb(*aid_pairs*)

wbia_is_same(*aid_pairs*)

class wbia.algo.graph.mixin_wbia.**IBEISIO**
Bases: object

Direct interface into wbia tables and delta statistics

add_annots(*aid_list*)

find_unjustified_splits()

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.mixin_helpers import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='GZ_Master1')
>>> ibs = wbia.opendb(defaultdb='PZ_Master1')
>>> infr = wbia.AnnotInference(ibs, 'all', autoinit=True)
>>> infr.reset_feedback('staging', apply=True)
>>> infr.relabel_using_reviews(rectify=False)
>>> unjustified = infr.find_unjustified_splits()
>>> review_edges = []
>>> for cc1, cc2 in unjustified:
>>>     u = next(iter(cc1))
>>>     v = next(iter(cc2))
>>>     review_edges.append(nxu.e_(u, v))
>>> infr.verbose = 100
>>> infr.prioritize(
>>>     edges=review_edges, scores=[1] * len(review_edges),
>>>     reset=True,
>>> )
>>> infr.qt_review_loop()
```

get_wbia_name_delta(*ignore_unknown=True*, *relabel=True*)
Rectifies internal name_labels with the names stored in the name table.

Return a pandas dataframe indicating which names have changed for what annotations.

Parameters

- **ignore_unknown** (*bool*) – if True does not return deltas for unknown annotations (those with degree 0).
- **relabel** (*bool*) – if True, ensures that all nodes are labeled based on the current PCCs.

Returns

pd.DataFrame - name_delta_df - data frame where each row specifies an aid and its *old_name* which is in the wbia database and the *new_name* which is what we infer it should be renamed to.

Example

infr.write_wbia_name_assignment

CommandLine: python -m wbia.algo.graph.mixin_wbia get_wbia_name_delta

Doctest:

```
>>> from wbia.algo.graph.mixin_wbia import *  # NOQA
>>> import wbia
>>> infr = wbia.AnnotInference('PZ_MTEST', aids=list(range(1, 10)),
```

(continues on next page)

(continued from previous page)

```
>>>                               autoinit='annotmatch', verbose=4)
>>> pccs1 = list(infr.positive_components())
>>> print('pccs1 = %r' % (pccs1,))
>>> print('names = {}'.format(list(infr.gen_node_values('name_label',␣
↪infr.aids))))
>>> assert pccs1 == [{1, 2, 3, 4}, {5, 6, 7, 8}, {9}]
>>> # Split a PCC and then merge two other PCCs
>>> infr.add_feedback_from([(1, 2), (1, 3), (1, 4)], evidence_
↪decision=NEGTV)
>>> infr.add_feedback((6, 7), NEGTV)
>>> infr.add_feedback((5, 8), NEGTV)
>>> infr.add_feedback((4, 5), POSTV)
>>> infr.add_feedback((7, 8), POSTV)
>>> pccs2 = list(infr.positive_components())
>>> print('pccs2 = %r' % (pccs2,))
>>> pccs2 = sorted(pccs2)
>>> assert pccs2 == [{9}, {1}, {2, 3, 4, 5, 6}, {7, 8}]
>>> print(list(infr.gen_node_values('name_label', infr.aids)))
>>> name_delta_df = infr.get_wbia_name_delta()
>>> result = str(name_delta_df)
>>> print(result)
    old_name       new_name
aid
1      06_410  IBEIS_UNKNOWN_0042
5      07_061              06_410
6      07_061              06_410
```

Doctest:

```
>>> from wbia.algo.graph.mixin_wbia import *  # NOQA
>>> import wbia
>>> infr = wbia.AnnotInference('PZ_MTEST', aids=list(range(1, 10)),
>>>                            autoinit='annotmatch', verbose=4)
>>> infr.add_feedback_from([(1, 2), (1, 3), (1, 4)], evidence_
↪decision=NEGTV)
>>> infr.add_feedback((4, 5), POSTV)
>>> name_delta_df = infr.get_wbia_name_delta()
>>> result = str(name_delta_df)
>>> print(result)
    old_name new_name
aid
2      06_410   07_061
3      06_410   07_061
4      06_410   07_061
```

Doctest:

```
>>> from wbia.algo.graph.mixin_wbia import *  # NOQA
>>> import wbia
>>> infr = wbia.AnnotInference('PZ_MTEST', aids=list(range(1, 10)),
>>>                            autoinit='annotmatch', verbose=4)
>>> name_delta_df = infr.get_wbia_name_delta()
>>> result = str(name_delta_df)
>>> print(result)
Empty DataFrame
Columns: [old_name, new_name]
Index: []
```

match_state_delta(*old='annotmatch'*, *new='all'*)

Returns information about state change of annotmatches

By default this will return a pandas dataframe indicating which edges in the annotmatch table have changed and all new edges relative to the current infr.graph state.

Notes

valid values for *old* and *new* are {'annotmatch', 'staging', 'all', 'internal', or 'external'}.

The args old/new='all' resolves to the internal graph state, 'annotmatch' resolves to the on-disk annotmatch table, and 'staging' resolves to the on-disk staging table (you can further separate all by specifying 'internal' or 'external'). You any of these old/new combinations to check differences in the state. However, the default values are what you use to sync the graph state to annotmatch.

Parameters

- **old** (*str*) – indicates the old data (i.e. the place that will be written to)
- **new** (*str*) – indicates the new data (i.e. the data to write)

Returns

pd.DataFrame - edge_delta_df - indicates the old and new values of the changed edge attributes.

CommandLine: python -m wbia.algo.graph.core match_state_delta

Doctest:

```
>>> from wbia.algo.graph.mixin_wbia import *  # NOQA
>>> import wbia
>>> infr = wbia.AnnotInference('PZ_MTEST', aids=list(range(1, 10)),
>>>                            autoinit='annotmatch', verbose=4)
>>> # Split a PCC and then merge two other PCCs
>>> infr.add_feedback((1, 2), NEGTV)
>>> infr.add_feedback((6, 7), NEGTV)
>>> infr.add_feedback((5, 8), NEGTV)
>>> infr.add_feedback((4, 5), POSTV)
>>> infr.add_feedback((7, 8), POSTV)
>>> edge_delta_df = infr.match_state_delta()
>>> subset = edge_delta_df[['old_evidence_decision', 'new_evidence_
↪decision']]
>>> result = str(subset)
>>> # sort result by aid1
>>> result = '\n'.join(result.splitlines()[:2] + sorted(result.
↪splitlines()[2:]))
>>> print(result)
          old_evidence_decision new_evidence_decision
aid1 aid2
1    2                    match               nomatch
4    5                      NaN                 match
5    8               unreviewed               nomatch
6    7               unreviewed               nomatch
7    8                    match                 match
```

name_group_delta_stats(*old_ccs*, *new_ccs*, *verbose=False*)

name_group_stats(*verbose=None*)

name_label_group_delta_info()
If the name labeling delta is non-zero then you need to rectify names

infr.relabel_using_reviews(rectify=False)

read_wbia_annotmatch_feedback(*edges=None*)
Reads feedback from annotmatch table and returns the result. Internal state is not changed.

Parameters **only_existing_edges** (*bool*) – if True only reads info existing edges

CommandLine: python -m wbia.algo.graph.core read_wbia_annotmatch_feedback

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.core import *  # NOQA
>>> infr = testdata_infr('testdb1')
>>> feedback = infr.read_wbia_annotmatch_feedback()
>>> items = feedback[(2, 3)]
>>> result = ('feedback = %s' % (ut.repr2(feedback, nl=2),))
>>> print(result)
>>> assert len(feedback) >= 2, 'should contain at least 2 edges'
>>> assert len(items) == 1, '2-3 should have one review'
>>> assert items[0]['evidence_decision'] == POSTV, '2-3 must match'
```

read_wbia_staging_feedback(*edges=None*)
Reads feedback from review staging table.

Parameters **infr** –

Returns feedback

Return type

?

CommandLine: python -m wbia.algo.graph.mixin_wbia read_wbia_staging_feedback

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.graph.mixin_wbia import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('GZ_Master1')
>>> infr = wbia.AnnotInference(ibs=ibs, aids='all')
>>> feedback = infr.read_wbia_staging_feedback()
>>> result = ('feedback = %s' % (ut.repr2(feedback),))
>>> print(result)
```

reset_labels_to_wbia()
Sets to IBEIS de-facto labels if available

reset_staging_with_ensure()
Make sure staging has all info that annotmatch has.

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

wbia_delta_info(*edge_delta_df=None*, *name_delta_df=None*)

wbia_edge_delta_info(*edge_delta_df=None*)

wbia_name_group_delta_info(*verbose=None*)
infr.relabel_using_reviews(rectify=False)

write_wbia_annotmatch_feedback(*edge_delta_df=None*)
Commits the current state in external and internal into the annotmatch table. Annotmatch only stores the final review in the history of reviews.

By default this will sync the current graph state to the annotmatch table. It computes the edge_delta under the hood, so if you already made one then you can pass it in for a little extra speed.

Parameters **edge_delta_df** (*pd.DataFrame*) – precomputed using match_state_delta. if None it will be computed under the hood.

write_wbia_name_assignment(*name_delta_df=None*, ***kwargs*)
Write the name delta to the annotations table.

It computes the name delta under the hood, so if you already made one then you can pass it in for a little extra speed.

Note: This will call infr.relabel_using_reviews(rectify=True) if name_delta_df is not given directly.

Parameters **name_delta_df** (*pd.DataFrame*) – if None, the value is computed using *get_wbia_name_delta*. Note you should ensure this delta is made after nodes have been relabeled using reviews.

write_wbia_staging_feedback()
Commit all reviews in internal_feedback into the staging table. The edges are removed from interal_feedback and added to external feedback. The staging tables stores each review in the order it happened so history is fully reconstructable if staging is never deleted.

This write function is done using the implicit delta maintained by infr.internal_feedback. Therefore, it take no args. This is generally called automatically by *infr.accept*.

wbia.algo.graph.mixin_wbia.**fix_annotmatch_to_undirected_upper**(*ibs*)
Enforce that all items in annotmatch are undirected upper

import wbia # ibs = wbia.opendb('PZ_Master1') ibs = wbia.opendb('PZ_PB_RF_TRAIN')

wbia.algo.graph.mixin_wbia.**needs_conversion**(*infr*)

1.1.1.2.15 wbia.algo.graph.nx_dynamic_graph module

class wbia.algo.graph.nx_dynamic_graph.**DynConnGraph**(**args*, ***kwargs*)
Bases: networkx.classes.graph.Graph, *wbia.algo.graph.nx_dynamic_graph.GraphHelperMixin*

Dynamically connected graph.

Maintains a data structure parallel to a normal networkx graph that maintains dynamic connectivity for fast connected compoment queries.

Underlying Data Structures and limitations are

- UnionFind | lg(n) | n | No

- UnionFind2 | n* | n | 1
- EulerTourForest | lg^2(n) | lg^2(n) | lg(n) / lglg(n) - - Ammortized
- it seems to be very quick

References

https://courses.csail.mit.edu/6.851/spring14/lectures/L20.pdf https://courses.csail.mit.edu/6.851/spring14/lectures/L20.html http://cs.stackexchange.com/questions/33595/maintaining-connecte https://en.wikipedia.org/wiki/Dynamic_connectivity#Fully_dynamic_connectivity

CommandLine: python -m wbia.algo.graph.nx_dynamic_graph DynConnGraph

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.nx_dynamic_graph import *  # NOQA
>>> self = DynConnGraph()
>>> self.add_edges_from([(1, 2), (2, 3), (4, 5), (6, 7), (7, 4)])
>>> self.add_edges_from([(10, 20), (20, 30), (40, 50), (60, 70), (70, 40)])
>>> self._ccs
>>> u, v = 20, 1
>>> assert self.node_label(u) != self.node_label(v)
>>> assert self.connected_to(u) != self.connected_to(v)
>>> self.add_edge(u, v)
>>> assert self.node_label(u) == self.node_label(v)
>>> assert self.connected_to(u) == self.connected_to(v)
>>> self.remove_edge(u, v)
>>> assert self.node_label(u) != self.node_label(v)
>>> assert self.connected_to(u) != self.connected_to(v)
>>> ccs = list(self.connected_components())
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.qtensure()
>>> pt.show_nx(self)
```

todo: check if nodes exist when adding

add_edge(*u*, *v*, ***attr*)

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.nx_dynamic_graph import *  # NOQA
>>> self = DynConnGraph()
>>> self.add_edges_from([(1, 2), (2, 3), (4, 5), (6, 7), (7, 4)])
>>> assert self._ccs == {1: {1, 2, 3}, 4: {4, 5, 6, 7}}
>>> self.add_edge(1, 5)
>>> assert self._ccs == {1: {1, 2, 3, 4, 5, 6, 7}}
```

add_edges_from(*ebunch*, ***attr*)

Add all the edges in ebunch_to_add.

Parameters

- **ebunch_to_add** (*container of edges*) – Each edge given in the container will be added to the graph. The edges must be given as 2-tuples (u, v) or 3-tuples (u, v, d) where d is a dictionary containing edge data.
- **attr** (*keyword arguments, optional*) – Edge data (or labels or objects) can be assigned using keyword arguments.

See also:

add_edge() add a single edge

add_weighted_edges_from() convenient way to add weighted edges

Notes

Adding the same edge twice has no effect but any edge data will be updated when each duplicate edge is added.

Edge attributes specified in an ebunch take precedence over attributes specified via keyword arguments.

Examples

```
>>> G = nx.Graph()  # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_edges_from([(0, 1), (1, 2)])  # using a list of edge tuples
>>> e = zip(range(0, 3), range(1, 4))
>>> G.add_edges_from(e)  # Add the path graph 0-1-2-3
```

Associate data to edges

```
>>> G.add_edges_from([(1, 2), (2, 3)], weight=3)
>>> G.add_edges_from([(3, 4), (1, 4)], label="WN2898")
```

add_node(*n*, ***attr*)

Add a single node *node_for_adding* and update node attributes.

Parameters

- **node_for_adding** (*node*) – A node can be any hashable Python object except None.
- **attr** (*keyword arguments, optional*) – Set or change node attributes using key=value.

See also:

add_nodes_from()

Examples

```
>>> G = nx.Graph()  # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_node(1)
>>> G.add_node("Hello")
>>> K3 = nx.Graph([(0, 1), (1, 2), (2, 0)])
>>> G.add_node(K3)
>>> G.number_of_nodes()
3
```

Use keywords set/change node attributes:

```
>>> G.add_node(1, size=10)
>>> G.add_node(3, weight=0.4, UTM=("13S", 382871, 3972649))
```

Notes

A hashable object is one that can be used as a key in a Python dictionary. This includes strings, numbers, tuples of strings and numbers, etc.

On many platforms hashable items also include mutables such as NetworkX Graphs, though one should be careful that the hash doesn't change on mutables.

add_nodes_from(*nodes*, ***attr*)
Add multiple nodes.

Parameters

- **nodes_for_adding** (*iterable container*) – A container of nodes (list, dict, set, etc.). OR A container of (node, attribute dict) tuples. Node attributes are updated using the attribute dict.
- **attr** (*keyword arguments, optional (default= no attributes)*) – Update attributes for all nodes in nodes. Node attributes specified in nodes as a tuple take precedence over attributes specified via keyword arguments.

See also:

add_node()

Examples

```
>>> G = nx.Graph()  # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_nodes_from("Hello")
>>> K3 = nx.Graph([(0, 1), (1, 2), (2, 0)])
>>> G.add_nodes_from(K3)
>>> sorted(G.nodes(), key=str)
[0, 1, 2, 'H', 'e', 'l', 'o']
```

Use keywords to update specific node attributes for every node.

```
>>> G.add_nodes_from([1, 2], size=10)
>>> G.add_nodes_from([3, 4], weight=0.4)
```

Use (node, attrdict) tuples to update attributes for specific nodes.

```
>>> G.add_nodes_from([(1, dict(size=11)), (2, {"color": "blue"})])
>>> G.nodes[1]["size"]
11
>>> H = nx.Graph()
>>> H.add_nodes_from(G.nodes(data=True))
>>> H.nodes[1]["size"]
11
```

are_nodes_connected(*u*, *v*)

clear()
Remove all nodes and edges from the graph.

This also removes the name, and all graph, node, and edge attributes.

Examples

```
>>> G = nx.path_graph(4)  # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.clear()
>>> list(G.nodes)
[]
>>> list(G.edges)
[]
```

component(*label*)

component_labels()

component_nodes(*label*)

connected_components()

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.nx_dynamic_graph import *  # NOQA
>>> self = DynConnGraph()
>>> self.add_edges_from([(1, 2), (2, 3), (4, 5), (6, 7)])
>>> ccs = list(self.connected_components())
>>> result = 'ccs = {}'.format(ut.repr2(ccs, nl=0))
>>> print(result)
ccs = [{1, 2, 3}, {4, 5}, {6, 7}]
```

connected_to(*node*)

node_label(*node*)

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.nx_dynamic_graph import *  # NOQA
>>> self = DynConnGraph()
>>> self.add_edges_from([(1, 2), (2, 3), (4, 5), (6, 7)])
>>> assert self.node_label(2) == self.node_label(1)
>>> assert self.node_label(2) != self.node_label(4)
```

node_labels(**nodes*)

number_of_components()

remove_edge(*u*, *v*)

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.nx_dynamic_graph import *  # NOQA
>>> self = DynConnGraph()
>>> self.add_edges_from([(1, 2), (2, 3), (4, 5), (6, 7), (7, 4)])
>>> assert self._ccs == {1: {1, 2, 3}, 4: {4, 5, 6, 7}}
>>> self.add_edge(1, 5)
```

(continues on next page)

(continued from previous page)

```
>>> assert self._ccs == {1: {1, 2, 3, 4, 5, 6, 7}}
>>> self.remove_edge(1, 5)
>>> assert self._ccs == {1: {1, 2, 3}, 4: {4, 5, 6, 7}}
```

remove_edges_from(*ebunch*)

Remove all edges specified in ebunch.

Parameters **ebunch** (*list or container of edge tuples*) – Each edge given in the list or container will be removed from the graph. The edges can be:

- 2-tuples (u, v) edge between u and v.
- 3-tuples (u, v, k) where k is ignored.

See also:

remove_edge() remove a single edge

Notes

Will fail silently if an edge in ebunch is not in the graph.

Examples

```
>>> G = nx.path_graph(4)  # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> ebunch = [(1, 2), (2, 3)]
>>> G.remove_edges_from(ebunch)
```

remove_node(*n*)

CommandLine: python -m wbia.algo.graph.nx_dynamic_graph remove_node

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.nx_dynamic_graph import *  # NOQA
>>> self = DynConnGraph()
>>> self.add_edges_from([(1, 2), (2, 3), (4, 5), (5, 6), (6, 7), (7, 8), (8,␣
↪9)])
>>> assert self._ccs == {1: {1, 2, 3}, 4: {4, 5, 6, 7, 8, 9}}
>>> self.remove_node(2)
>>> assert self._ccs == {1: {1}, 3: {3}, 4: {4, 5, 6, 7, 8, 9}}
>>> self.remove_node(7)
>>> assert self._ccs == {1: {1}, 3: {3}, 4: {4, 5, 6}, 8: {8, 9}}
```

remove_nodes_from(*nodes*)

Remove multiple nodes.

Parameters **nodes** (*iterable container*) – A container of nodes (list, dict, set, etc.). If a node in the container is not in the graph it is silently ignored.

See also:

remove_node()

Examples

```
>>> G = nx.path_graph(3)  # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> e = list(G.nodes)
>>> e
[0, 1, 2]
>>> G.remove_nodes_from(e)
>>> list(G.nodes)
[]
```

subgraph (*nbunch*, *dynamic=False*)

Returns a SubGraph view of the subgraph induced on *nodes*.

The induced subgraph of the graph contains the nodes in *nodes* and the edges between those nodes.

Parameters **nodes** (*list, iterable*) – A container of nodes which will be iterated through once.

Returns **G** – A subgraph view of the graph. The graph structure cannot be changed but node/edge attributes can and are shared with the original graph.

Return type SubGraph View

Notes

The graph, edge and node attributes are shared with the original graph. Changes to the graph structure is ruled out by the view, but changes to attributes are reflected in the original graph.

To create a subgraph with its own copy of the edge/node attributes use: G.subgraph(nodes).copy()

For an inplace reduction of a graph to a subgraph you can remove nodes: G.remove_nodes_from([n for n in G if n not in set(nodes)])

Subgraph views are sometimes NOT what you want. In most cases where you want to do more than simply look at the induced edges, it makes more sense to just create the subgraph as its own graph with code like:

```
# Create a subgraph SG based on a (possibly multigraph) G
SG = G.__class__()
SG.add_nodes_from((n, G.nodes[n]) for n in largest_wcc)
if SG.is_multigraph():
    SG.add_edges_from((n, nbr, key, d)
        for n, nbrs in G.adj.items() if n in largest_wcc
        for nbr, keydict in nbrs.items() if nbr in largest_wcc
        for key, d in keydict.items())
else:
    SG.add_edges_from((n, nbr, d)
        for n, nbrs in G.adj.items() if n in largest_wcc
        for nbr, d in nbrs.items() if nbr in largest_wcc)
SG.graph.update(G.graph)
```

Examples

```
>>> G = nx.path_graph(4)  # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> H = G.subgraph([0, 1, 2])
>>> list(H.edges)
[(0, 1), (1, 2)]
```

class wbia.algo.graph.nx_dynamic_graph.**GraphHelperMixin**
Bases: utool.util_dev.NiceRepr

edges(*nbunch=None*, *data=False*, *default=None*)

has_edges(*edges*)

has_nodes(*nodes*)

class wbia.algo.graph.nx_dynamic_graph.**NiceGraph**(*incoming_graph_data=None*, ***attr*)
Bases: networkx.classes.graph.Graph, *wbia.algo.graph.nx_dynamic_graph.GraphHelperMixin*

class wbia.algo.graph.nx_dynamic_graph.**nx_UnionFind**(*elements=None*)
Bases: object

Based of nx code

add_element(*x*)

add_elements(*elements*)

clear()

rebalance(*elements=None*)

remove_entire_cc(*elements*)

to_sets()

union(**objects*)
Find the sets containing the objects and merge them all.

1.1.1.2.16 wbia.algo.graph.nx_edge_augmentation module

Algorithms for finding k-edge-augmentations

A k-edge-augmentation is a set of edges, that once added to a graph, ensures that the graph is k-edge-connected. Typically, the goal is to find the augmentation with minimum weight. In general, it is not gaurenteed that a k-edge-augmentation exists.

class wbia.algo.graph.nx_edge_augmentation.**MetaEdge**(*meta_uv*, *uv*, *w*)
Bases: tuple

meta_uv
Alias for field number 0

uv
Alias for field number 1

w
Alias for field number 2

wbia.algo.graph.nx_edge_augmentation.**bridge_augmentation**(*G*, *avail=None*, *weight=None*)
Finds the a set of edges that bridge connects G.

Adding these edges to G will make it 2-edge-connected. If no constraints are specified the returned set of edges is minimum an optimal, otherwise the solution is approximated.

Notes

If there are no constraints the solution can be computed in linear time using *unconstrained_bridge_augmentation()*. Otherwise, the problem becomes NP-hard and is the solution is approximated by *weighted_bridge_augmentation()*.

wbia.algo.graph.nx_edge_augmentation.**collapse**(*G*, *grouped_nodes*)
Collapses each group of nodes into a single node.

This is similar to condensation, but works on undirected graphs.

Parameters

- **G** (*NetworkX Graph*) – A directed graph.
- **grouped_nodes** (*list or generator*) – Grouping of nodes to collapse. The grouping must be disjoint. If grouped_nodes are strongly_connected_components then this is equivalent to condensation.

Returns **C** – The collapsed graph C of G with respect to the node grouping. The node labels are integers corresponding to the index of the component in the list of strongly connected components of G. C has a graph attribute named 'mapping' with a dictionary mapping the original nodes to the nodes in C to which they belong. Each node in C also has a node attribute 'members' with the set of original nodes in G that form the group that the node in C represents.

Return type NetworkX Graph

Examples

```
>>> # Collapses a graph using disjoint groups, but not necesarilly connected
>>> G = nx.Graph([(1, 0), (2, 3), (3, 1), (3, 4), (4, 5), (5, 6), (5, 7)])
>>> G.add_node('A')
>>> grouped_nodes = [{0, 1, 2, 3}, {5, 6, 7}]
>>> C = collapse(G, grouped_nodes)
>>> members = nx.get_node_attributes(C, 'members')
>>> sorted(members.keys())
[0, 1, 2, 3]
>>> member_values = set(map(frozenset, members.values()))
>>> assert {0, 1, 2, 3} in member_values
>>> assert {4} in member_values
>>> assert {5, 6, 7} in member_values
>>> assert {'A'} in member_values
```

wbia.algo.graph.nx_edge_augmentation.**compat_shuffle**(*rng*, *input*)

wbia.algo.graph.nx_edge_augmentation.**complement_edges**(*G*)
Returns only the edges in the complement of G

Example

```
>>> G = nx.path_graph((1, 2, 3, 4))
>>> sorted(complement_edges(G))
[(1, 3), (1, 4), (2, 4)]
>>> G = nx.path_graph((1, 2, 3, 4), nx.DiGraph())
>>> sorted(complement_edges(G))
[(1, 3), (1, 4), (2, 1), (2, 4), (3, 1), (3, 2), (4, 1), (4, 2), (4, 3)]
>>> G = nx.complete_graph(1000)
```

(continues on next page)

(continued from previous page)

```
>>> sorted(complement_edges(G))
[]
```

wbia.algo.graph.nx_edge_augmentation.**greedy_k_edge_augmentation**(*G*, *k*, *avail=None*, *weight=None*, *seed=None*)

Greedy algorithm for finding a k-edge-augmentation

Notes

The algorithm is simple. Edges are incrementally added between parts of the graph that are not yet locally k-edge-connected. Then edges are from the augmenting set are pruned as long as local-edge-connectivity is not broken.

This algorithm is greedy and does not provide optimiality gaurentees. It exists only to provide *k_edge_augmentation()* with the ability to generate a feasible solution for arbitrary k.

Example

```
>>> G = nx.path_graph((1, 2, 3, 4, 5, 6, 7))
>>> sorted(greedy_k_edge_augmentation(G, k=2))
[(1, 7)]
>>> sorted(greedy_k_edge_augmentation(G, k=1, avail=[]))
[]
>>> G = nx.path_graph((1, 2, 3, 4, 5, 6, 7))
>>> avail = {(u, v): 1 for (u, v) in complement_edges(G)}
>>> # randomized pruning process can produce different solutions
>>> sorted(greedy_k_edge_augmentation(G, k=4, avail=avail, seed=2))
[(1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (2, 4), (2, 6), (3, 7), (5, 7)]
>>> sorted(greedy_k_edge_augmentation(G, k=4, avail=avail, seed=3))
[(1, 3), (1, 5), (1, 6), (2, 4), (2, 6), (3, 7), (4, 7), (5, 7)]
```

wbia.algo.graph.nx_edge_augmentation.**is_k_edge_connected**(*G*, *k*)

Tests to see if a graph is k-edge-connected

See also:

is_locally_k_edge_connected()

Example

```
>>> G = nx.barbell_graph(10, 0)
>>> is_k_edge_connected(G, k=1)
True
>>> is_k_edge_connected(G, k=2)
False
```

wbia.algo.graph.nx_edge_augmentation.**is_locally_k_edge_connected**(*G*, *s*, *t*, *k*)

Tests to see if an edge in a graph is locally k-edge-connected

See also:

is_k_edge_connected()

Example

```
>>> G = nx.barbell_graph(10, 0)
>>> is_locally_k_edge_connected(G, 5, 15, k=1)
True
>>> is_locally_k_edge_connected(G, 5, 15, k=2)
False
>>> is_locally_k_edge_connected(G, 1, 5, k=2)
True
```

wbia.algo.graph.nx_edge_augmentation.**k_edge_augmentation**(*G*, *k*, *avail=None*, *weight=None*, *partial=False*)

Finds set of edges to k-edge-connect G.

This function uses the most efficient function available (depending on the value of k and if the problem is weighted or unweighted) to search for a minimum weight subset of available edges that k-edge-connects G. In general, finding a k-edge-augmentation is NP-hard, so solutions are not garuenteed to be minimal.

Parameters

- **G** (*NetworkX graph*) –
- **k** (*Integer*) – Desired edge connectivity
- **avail** (*dict or a set 2 or 3 tuples*) – The available edges that can be used in the augmentation.

  If unspecified, then all edges in the complement of G are available. Otherwise, each item is an available edge (with an optinal weight).

  In the unweighted case, each item is an edge `(u, v)`.

  In the weighted case, each item is a 3-tuple `(u, v, d)` or a dict with items `(u, v): d`. The third item, `d`, can be a dictionary or a real number. If `d` is a dictionary `d[weight]` correspondings to the weight.
- **weight** (*string*) – key to use to find weights if avail is a set of 3-tuples where the third item in each tuple is a dictionary.
- **partial** (*Boolean*) – If partial is True and no feasible k-edge-augmentation exists, then all available edges are returned.

Returns **aug_edges** – the G would become k-edge-connected. If partial is False, an error is raised if this is not possible. Otherwise, all available edges are generated.

Return type a generator of edges. If these edges are added to G, then

Raises

- NetworkXNotImplemented: – If the input graph is directed or a multigraph.
- ValueError: – If k is less than 1

Notes

When k=1 this returns an optimal solution.

When k=2 and avail is None, this returns an optimal solution. Otherwise when k=2, this returns a 2-approximation of the optimal solution.

For k>3, this problem is NP-hard and this uses a randomized algorithm that produces a feasible solution, but provides no gaurentees on the solution weight.

Example

```
>>> # Unweighted cases
>>> G = nx.path_graph((1, 2, 3, 4))
>>> G.add_node(5)
>>> sorted(k_edge_augmentation(G, k=1))
[(1, 5)]
>>> sorted(k_edge_augmentation(G, k=2))
[(1, 5), (5, 4)]
>>> sorted(k_edge_augmentation(G, k=3))
[(1, 4), (1, 5), (2, 5), (3, 5), (4, 5)]
>>> complement = list(k_edge_augmentation(G, k=5, partial=True))
>>> G.add_edges_from(complement)
>>> nx.edge_connectivity(G)
4
```

Example

```
>>> # Weighted cases
>>> G = nx.path_graph((1, 2, 3, 4))
>>> G.add_node(5)
>>> # avail can be a tuple with a dict
>>> avail = [(1, 5, {'weight': 11}), (2, 5, {'weight': 10})]
>>> sorted(k_edge_augmentation(G, k=1, avail=avail, weight='weight'))
[(2, 5)]
>>> # or avail can be a 3-tuple with a real number
>>> avail = [(1, 5, 11), (2, 5, 10), (4, 3, 1), (4, 5, 51)]
>>> sorted(k_edge_augmentation(G, k=2, avail=avail))
[(1, 5), (2, 5), (4, 5)]
>>> # or avail can be a dict
>>> avail = {(1, 5): 11, (2, 5): 10, (4, 3): 1, (4, 5): 51}
>>> sorted(k_edge_augmentation(G, k=2, avail=avail))
[(1, 5), (2, 5), (4, 5)]
>>> # If augmentation is infeasible, then all edges in avail are returned
>>> avail = {(1, 5): 11}
>>> sorted(k_edge_augmentation(G, k=2, avail=avail, partial=True))
[(1, 5)]
```

wbia.algo.graph.nx_edge_augmentation.**one_edge_augmentation**(*G*, *avail=None*, *weight=None*, *partial=False*)

Finds minimum weight set of edges to connect G.

Notes

Uses either *unconstrained_one_edge_augmentation()* or *weighted_one_edge_augmentation()* depending on whether `avail` is specified. Both algorithms are based on finding a minimum spanning tree. As such both algorithms find optimal solutions and run in linear time.

wbia.algo.graph.nx_edge_augmentation.**partial_k_edge_augmentation**(*G*, *k*, *avail*, *weight=None*)

Finds augmentation that k-edge-connects as much of the graph as possible

When a k-edge-augmentation is not possible, we can still try to find a small set of edges that partially k-edge-connects as much of the graph as possible.

Notes

Construct H that augments G with all edges in avail. Find the k-edge-subgraphs of H. For each k-edge-subgraph, if the number of nodes is more than k, then find the k-edge-augmentation of that graph and add it to the solution. Then add all edges in avail between k-edge subgraphs to the solution.

```
>>> G = nx.path_graph((1, 2, 3, 4, 5, 6, 7))
>>> G.add_node(8)
>>> avail = [(1, 3), (1, 4), (1, 5), (2, 4), (2, 5), (3, 5), (1, 8)]
>>> sorted(partial_k_edge_augmentation(G, k=2, avail=avail))
[(1, 5), (1, 8)]
```

wbia.algo.graph.nx_edge_augmentation.**unconstrained_bridge_augmentation**(*G*)

Finds an optimal 2-edge-augmentation of G using the fewest edges.

This is an implementation of the algorithm detailed in [1]_. The basic idea is to construct a meta-graph of bridge-ccs, connect leaf nodes of the trees to connect the entire graph, and finally connect the leafs of the tree in dfs-preorder to bridge connect the entire graph.

Notes

Input: a graph G. First find the bridge components of G and collapse each bridge-cc into a node of a metagraph graph C, which is gaurenteed to be a forest of trees.

C contains p "leafs" — nodes with exactly one incident edge. C contains q "isolated nodes" — nodes with no incident edges.

Theorem: If p + q > 1, then at least $ceil(p/2) + q$ **edges are** needed to bridge connect C. This algorithm achieves this min number.

The method first adds enough edges to make G into a tree and then pairs leafs in a simple fashion.

Let n be the number of trees in C. Let v(i) be an isolated vertex in the i-th tree if one exists, otherwise it is a pair of distinct leafs nodes in the i-th tree. Alternating edges from these sets (i.e. adding edges A1 = [(v(i)[0], v(i + 1)[1]), v(i + 1)[0], v(i + 2)[1]). . .]) connects C into a tree T. This tree has p' = p + 2q - 2(n -1) leafs and no isolated vertices. A1 has n - 1 edges. The next step finds ceil(p' / 2) edges to biconnect any tree with p' leafs.

Convert T into an arborescence T' by picking an arbitrary root node with degree >= 2 and directing all edges away from the root. Note the implementation implicitly constructs T'.

The leafs of T are the nodes with no existing edges in T'. Order the leafs of T' by DFS prorder. Then break this list in half and add the zipped pairs to A2.

The set A = A1 + A2 is the minimum augmentation in the metagraph.

To convert this to edges in the original graph

References

Example

```
>>> G = nx.path_graph((1, 2, 3, 4, 5, 6, 7))
>>> sorted(unconstrained_bridge_augmentation(G))
[(1, 7)]
>>> G = nx.path_graph((1, 2, 3, 2, 4, 5, 6, 7))
>>> sorted(unconstrained_bridge_augmentation(G))
[(1, 3), (3, 7)]
>>> G = nx.Graph([(0, 1), (0, 2), (1, 2)])
>>> G.add_node(4)
>>> sorted(unconstrained_bridge_augmentation(G))
[(1, 4), (4, 0)]
```

wbia.algo.graph.nx_edge_augmentation.**unconstrained_one_edge_augmentation**(*G*)

Finds the smallest set of edges to connect G.

This is a variant of the unweighted MST problem. If G is not empty, a feasible solution always exists.

Example

```
>>> G = nx.Graph([(1, 2), (2, 3), (4, 5)])
>>> G.add_nodes_from([6, 7, 8])
>>> sorted(unconstrained_one_edge_augmentation(G))
[(1, 4), (4, 6), (6, 7), (7, 8)]
```

wbia.algo.graph.nx_edge_augmentation.**weighted_bridge_augmentation**(*G*, *avail*, *weight=None*)

Finds an approximate min-weight 2-edge-augmentation of G.

This is an implementation of the approximation algorithm detailed in [1]_. It chooses a set of edges from avail to add to G that renders it 2-edge-connected if such a subset exists. This is done by finding a minimum spanning arborescence of a specially constructed metagraph.

Parameters

- **G** (*NetworkX graph*) –
- **avail** (*set of 2 or 3 tuples.*) – candidate edges (with optional weights) to choose from
- **weight** (*string*) – key to use to find weights if avail is a set of 3-tuples where the third item in each tuple is a dictionary.

Returns **aug_edges (set)**

Return type subset of avail chosen to augment G

Notes

Finding a weighted 2-edge-augmentation is NP-hard. Any edge not in `avail` is considered to have a weight of infinity. The approximation factor is 2 if `G` is connected and 3 if it is not. Runs in $O(m + nlog(n))$ time

References

Example

```
>>> G = nx.path_graph((1, 2, 3, 4))
>>> # When the weights are equal, (1, 4) is the best
>>> avail = [(1, 4, 1), (1, 3, 1), (2, 4, 1)]
>>> sorted(weighted_bridge_augmentation(G, avail))
[(1, 4)]
>>> # Giving (1, 4) a high weight makes the two edge solution the best.
>>> avail = [(1, 4, 1000), (1, 3, 1), (2, 4, 1)]
>>> sorted(weighted_bridge_augmentation(G, avail))
[(1, 3), (2, 4)]
>>> #------
>>> G = nx.path_graph((1, 2, 3, 4))
>>> G.add_node(5)
>>> avail = [(1, 5, 11), (2, 5, 10), (4, 3, 1), (4, 5, 1)]
>>> sorted(weighted_bridge_augmentation(G, avail=avail))
[(1, 5), (4, 5)]
>>> avail = [(1, 5, 11), (2, 5, 10), (4, 3, 1), (4, 5, 51)]
>>> sorted(weighted_bridge_augmentation(G, avail=avail))
[(1, 5), (2, 5), (4, 5)]
```

wbia.algo.graph.nx_edge_augmentation.**weighted_one_edge_augmentation**(*G*, *avail*, *weight=None*, *partial=False*)

Finds the minimum weight set of edges to connect G if one exists.

This is a variant of the weighted MST problem.

Example

```
>>> G = nx.Graph([(1, 2), (2, 3), (4, 5)])
>>> G.add_nodes_from([6, 7, 8])
>>> # any edge not in avail has an implicit weight of infinity
>>> avail = [(1, 3), (1, 5), (4, 7), (4, 8), (6, 1), (8, 1), (8, 2)]
>>> sorted(weighted_one_edge_augmentation(G, avail))
[(1, 5), (4, 7), (6, 1), (8, 1)]
>>> # find another solution by giving large weights to edges in the
>>> # previous solution (note some of the old edges must be used)
>>> avail = [(1, 3), (1, 5, 99), (4, 7, 9), (6, 1, 99), (8, 1, 99), (8, 2)]
>>> sorted(weighted_one_edge_augmentation(G, avail))
[(1, 5), (4, 7), (6, 1), (8, 2)]
```

1.1.1.2.17 wbia.algo.graph.nx_edge_kcomponents module

Algorithms for finding k-edge-connected components and subgraphs.

A k-edge-connected component (k-edge-cc) is a maximal set of nodes in G, such that all pairs of node have an edge-connectivity of at least k.

A k-edge-connected subgraph (k-edge-subgraph) is a maximal set of nodes in G, such that the subgraph of G defined by the nodes has an edge-connectivity at least k.

class wbia.algo.graph.nx_edge_kcomponents.**EdgeComponentAuxGraph**

Bases: object

A simple algorithm to find all k-edge-connected components in a graph.

Constructing the AuxillaryGraph (which may take some time) allows for the k-edge-ccs to be found in linear time for arbitrary k.

Notes

This implementation is based on [1]_. The idea is to construct an auxillary graph from which the k-edge-ccs can be extracted in linear time. The auxillary graph is constructed in O(|V|F) operations, where F is the complexity of max flow. Querying the components takes an additional O(|V|) operations. This algorithm can be slow for large graphs, but it handles an arbitrary k and works for both directed and undirected inputs.

The undirected case for k=1 is exactly connected components. The undirected case for k=2 is exactly bridge connected components. The directed case for k=1 is exactly strongly connected components.

References

Example

```
>>> from networkx.utils import pairwise
>>> # Build an interesting graph with multiple levels of k-edge-ccs
>>> paths = [
...     (1, 2, 3, 4, 1, 3, 4, 2),  # a 3-edge-cc (a 4 clique)
...     (5, 6, 7, 5),  # a 2-edge-cc (a 3 clique)
...     (1, 5),  # combine first two ccs into a 1-edge-cc
...     (0,),  # add an additional disconnected 1-edge-cc
... ]
>>> G = nx.Graph()
>>> G.add_nodes_from(it.chain(*paths))
>>> G.add_edges_from(it.chain(*[pairwise(path) for path in paths]))
>>> # Constructing the AuxGraph takes about O(n ** 4)
>>> aux_graph = EdgeComponentAuxGraph.construct(G)
>>> # Once constructed, querying takes O(n)
>>> sorted(map(sorted, aux_graph.k_edge_components(k=1)))
[[0], [1, 2, 3, 4, 5, 6, 7]]
>>> sorted(map(sorted, aux_graph.k_edge_components(k=2)))
[[0], [1, 2, 3, 4], [5, 6, 7]]
>>> sorted(map(sorted, aux_graph.k_edge_components(k=3)))
[[0], [1, 2, 3, 4], [5], [6], [7]]
>>> sorted(map(sorted, aux_graph.k_edge_components(k=4)))
[[0], [1], [2], [3], [4], [5], [6], [7]]
```

Example

```
>>> # The auxillary graph is primarilly used for k-edge-ccs but it
>>> # can also speed up the queries of k-edge-subgraphs by refining the
>>> # search space.
>>> from networkx.utils import pairwise
>>> paths = [
...     (1, 2, 4, 3, 1, 4),
... ]
>>> G = nx.Graph()
>>> G.add_nodes_from(it.chain(*paths))
>>> G.add_edges_from(it.chain(*[pairwise(path) for path in paths]))
>>> aux_graph = EdgeComponentAuxGraph.construct(G)
```

(continues on next page)

(continued from previous page)

```
>>> sorted(map(sorted, aux_graph.k_edge_subgraphs(k=3)))
[[1], [2], [3], [4]]
>>> sorted(map(sorted, aux_graph.k_edge_components(k=3)))
[[1, 4], [2], [3]]
```

classmethod construct(*G*)

Builds an auxillary graph encoding edge-connectivity between nodes.

Notes

Given G=(V, E), initialize an empty auxillary graph A. Choose an arbitrary source node s. Initialize a set N of available nodes (that can be used as the sink). The algorithm picks an arbitrary node t from N - {s}, and then computes the minimum st-cut (S, T) with value w. If G is directed the the minimum of the st-cut or the ts-cut is used instead. Then, the edge (s, t) is added to the auxillary graph with weight w. The algorithm is called recursively first using S as the available nodes and s as the source, and then using T and t. Recusion stops when the source is the only available node.

Parameters **G** (*NetworkX graph*) –

k_edge_components(*k*)

Queries the auxillary graph for k-edge-connected components.

Parameters **k** (*Integer*) – Desired edge connectivity

Returns **k_edge_components**

Return type a generator of k-edge-ccs

Notes

Given the auxillary graph, the k-edge-connected components can be determined in linear time by removing all edges with weights less than k from the auxillary graph. The resulting connected components are the k-edge-ccs in the original graph.

k_edge_subgraphs(*k*)

Queries the auxillary graph for k-edge-connected subgraphs.

Parameters **k** (*Integer*) – Desired edge connectivity

Returns **k_edge_subgraphs**

Return type a generator of k-edge-subgraphs

Notes

Refines the k-edge-ccs into k-edge-subgraphs. The running time is more than O(|V|).

For single values of k it is faster to use *nx.k_edge_subgraphs*. But for multiple values of k, it can be faster to build AuxGraph and then use this method.

wbia.algo.graph.nx_edge_kcomponents.**bridge_components**(*G*)

Finds all bridge-connected components G.

Parameters **G** (*NetworkX undirected graph*) –

Returns **bridge_components**

Return type a generator of 2-edge-connected components

See also:

k_edge_subgraphs() this function is a special case for an undirected graph where k=2.

biconnected_components() similar to this function, but is defined using 2-node-connectivity instead of 2-edge-connectivity.

Raises NetworkXNotImplemented: – If the input graph is directed or a multigraph.

Notes

Bridge-connected components are also known as 2-edge-connected components.

Example

```
>>> # The barbell graph with parameter zero has a single bridge
>>> G = nx.barbell_graph(5, 0)
>>> sorted(map(sorted, bridge_components(G)))
[[0, 1, 2, 3, 4], [5, 6, 7, 8, 9]]
```

wbia.algo.graph.nx_edge_kcomponents.**general_k_edge_subgraphs**(*G*, *k*)

General algorithm to find all maximal k-edge-connected subgraphs in G.

Returns **k_edge_subgraphs** – Each k-edge-subgraph is a maximal set of nodes that defines a subgraph of G that is k-edge-connected.

Return type a generator of nx.Graphs that are k-edge-subgraphs

Notes

Implementation of the basic algorithm from _[1]. The basic idea is to find a global minimum cut of the graph. If the cut value is at least k, then the graph is a k-edge-connected subgraph and can be added to the results. Otherwise, the cut is used to split the graph in two and the procedure is applied recursively. If the graph is just a single node, then it is also added to the results. At the end, each result is either gaurenteed to be a single node or a subgraph of G that is k-edge-connected.

This implementation contains optimizations for reducing the number of calls to max-flow, but there are other optimizations in _[1] that could be implemented.

References

Example

```
>>> from networkx.utils import pairwise
>>> paths = [
...      (11, 12, 13, 14, 11, 13, 14, 12),  # a 4-clique
...      (21, 22, 23, 24, 21, 23, 24, 22),  # another 4-clique
...      # connect the cliques with high degree but low connectivity
...      (50, 13),
...      (12, 50, 22),
...      (13, 102, 23),
...      (14, 101, 24),
... ]
```

(continues on next page)

(continued from previous page)

```
>>> G = nx.Graph(it.chain(*[pairwise(path) for path in paths]))
>>> sorted(map(len, k_edge_subgraphs(G, k=3)))
[1, 1, 1, 4, 4]
```

wbia.algo.graph.nx_edge_kcomponents.**k_edge_components**(*G*, *k*)

Generates nodes in each maximal k-edge-connected component in G.

Parameters

- **G** (*NetworkX graph*) –
- **k** (*Integer*) – Desired edge connectivity

Returns **k_edge_components** – will have k-edge-connectivity in the graph G.

Return type a generator of k-edge-ccs. Each set of returned nodes

See also:

`local_edge_connectivity()`

k_edge_subgraphs() similar to this function, but the subgraph defined by the nodes must also have k-edge-connectivity.

`k_components()` similar to this function, but uses node-connectivity instead of edge-connectivity

Raises

- NetworkXNotImplemented: – If the input graph is a multigraph.
- ValueError: – If k is less than 1

Notes

Attempts to use the most efficient implementation available based on k. If k=1, this is simply simply connected components for directed graphs and connected components for undirected graphs. If k=2 on an efficient bridge connected component algorithm from _[1] is run based on the chain decomposition. Otherwise, the algorithm from _[2] is used.

Example

```
>>> from networkx.utils import pairwise
>>> paths = [
...     (1, 2, 4, 3, 1, 4),
...     (5, 6, 7, 8, 5, 7, 8, 6),
... ]
>>> G = nx.Graph()
>>> G.add_nodes_from(it.chain(*paths))
>>> G.add_edges_from(it.chain(*[pairwise(path) for path in paths]))
>>> # note this returns {1, 4} unlike k_edge_subgraphs
>>> sorted(map(sorted, k_edge_components(G, k=3)))
[[1, 4], [2], [3], [5, 6, 7, 8]]
```

References

wbia.algo.graph.nx_edge_kcomponents.**k_edge_subgraphs**(*G*, *k*)

Generates nodes in each maximal k-edge-connected subgraph in G.

Parameters

- **G** (*NetworkX graph*) –
- **k** (*Integer*) – Desired edge connectivity

Returns **k_edge_subgraphs** – Each k-edge-subgraph is a maximal set of nodes that defines a subgraph of G that is k-edge-connected.

Return type a generator of k-edge-subgraphs

See also:

`edge_connectivity()`

k_edge_components() similar to this function, but nodes only need to have k-edge-connctivity within the graph G and the subgraphs might not be k-edge-connected.

Raises

- NetworkXNotImplemented: – If the input graph is a multigraph.
- ValueError: – If k is less than 1

Notes

Attempts to use the most efficient implementation available based on k. If k=1, or k=2 and the graph is undirected, then this simply calls *k_edge_components*. Otherwise the algorithm from _[1] is used.

Example

```
>>> from networkx.utils import pairwise
>>> paths = [
...     (1, 2, 4, 3, 1, 4),
...     (5, 6, 7, 8, 5, 7, 8, 6),
... ]
>>> G = nx.Graph()
>>> G.add_nodes_from(it.chain(*paths))
>>> G.add_edges_from(it.chain(*[pairwise(path) for path in paths]))
>>> # note this does not return {1, 4} unlike k_edge_components
>>> sorted(map(sorted, k_edge_subgraphs(G, k=3)))
[[1], [2], [3], [4], [5, 6, 7, 8]]
```

References

1.1.1.2.18 wbia.algo.graph.nx_utils module

TODO: the k-components will soon be implemented in networkx 2.0 use those instead

`wbia.algo.graph.nx_utils.`**`complement_edges`**`(`G`)`

`wbia.algo.graph.nx_utils.`**`connected_component_subgraphs`**`(`G`)`

`wbia.algo.graph.nx_utils.`**`demodata_bridge`**`()`

`wbia.algo.graph.nx_utils.`**`demodata_tarjan_bridge`**`()`

CommandLine: python -m wbia.algo.graph.nx_utils demodata_tarjan_bridge –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.nx_utils import *  # NOQA
>>> G = demodata_tarjan_bridge()
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.show_nx(G)
>>> ut.show_if_requested()
```

wbia.algo.graph.nx_utils.**diag_product**(*s1*, *s2*)
Does product, but iterates over the diagonal first

wbia.algo.graph.nx_utils.**e_**(*u*, *v*)

wbia.algo.graph.nx_utils.**edge_df**(*graph*, *edges*, *ignore=None*)

wbia.algo.graph.nx_utils.**edges_between**(*graph*, *nodes1*, *nodes2=None*, *assume_disjoint=False*, *assume_dense=True*)
Get edges between two components or within a single component

Parameters

- **graph** (*nx.Graph*) – the graph
- **nodes1** (*set*) – list of nodes
- **nodes2** (*set*) – if None it is equivlanet to nodes2=nodes1 (default=None)
- **assume_disjoint** (*bool*) – skips expensive check to ensure edges arnt returned twice (default=False)

CommandLine: python -m wbia.algo.graph.nx_utils –test-edges_between

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.nx_utils import *  # NOQA
>>> import utool as ut
>>> edges = [
>>>     (1, 2), (2, 3), (3, 4), (4, 1), (4, 3),  # cc 1234
>>>     (1, 5), (7, 2), (5, 1),  # cc 567 / 5678
>>>     (7, 5), (5, 6), (8, 7),
>>> ]
>>> digraph = nx.DiGraph(edges)
>>> graph = nx.Graph(edges)
>>> nodes1 = [1, 2, 3, 4]
>>> nodes2 = [5, 6, 7]
>>> n2 = sorted(edges_between(graph, nodes1, nodes2))
>>> n4 = sorted(edges_between(graph, nodes1))
>>> n5 = sorted(edges_between(graph, nodes1, nodes1))
>>> n1 = sorted(edges_between(digraph, nodes1, nodes2))
>>> n3 = sorted(edges_between(digraph, nodes1))
>>> print('n2 == %r' % (n2,))
>>> print('n4 == %r' % (n4,))
>>> print('n5 == %r' % (n5,))
>>> print('n1 == %r' % (n1,))
>>> print('n3 == %r' % (n3,))
```

(continues on next page)

(continued from previous page)

```
>>> assert n2 == ([(1, 5), (2, 7)]), '2'
>>> assert n4 == ([(1, 2), (1, 4), (2, 3), (3, 4)]), '4'
>>> assert n5 == ([(1, 2), (1, 4), (2, 3), (3, 4)]), '5'
>>> assert n1 == ([(1, 5), (5, 1), (7, 2)]), '1'
>>> assert n3 == ([(1, 2), (2, 3), (3, 4), (4, 1), (4, 3)]), '3'
>>> n6 = sorted(edges_between(digraph, nodes1 + [6], nodes2 + [1, 2], assume_
↪dense=False))
>>> print('n6 = %r' % (n6,))
>>> n6 = sorted(edges_between(digraph, nodes1 + [6], nodes2 + [1, 2], assume_
↪dense=True))
>>> print('n6 = %r' % (n6,))
>>> assert n6 == ([(1, 2), (1, 5), (2, 3), (4, 1), (5, 1), (5, 6), (7, 2)]), '6'
```

wbia.algo.graph.nx_utils.**edges_cross**(*graph*, *nodes1*, *nodes2*)

Finds edges between two sets of disjoint nodes. Running time is O(len(nodes1) * len(nodes2))

Parameters

- **graph** (*nx.Graph*) – an undirected graph
- **nodes1** (*set*) – set of nodes disjoint from *nodes2*
- **nodes2** (*set*) – set of nodes disjoint from *nodes1*.

wbia.algo.graph.nx_utils.**edges_inside**(*graph*, *nodes*)

Finds edges within a set of nodes Running time is O(len(nodes) ** 2)

Parameters

- **graph** (*nx.Graph*) – an undirected graph
- **nodes1** (*set*) – a set of nodes

wbia.algo.graph.nx_utils.**edges_outgoing**(*graph*, *nodes*)

Finds edges leaving a set of nodes. Average running time is O(len(nodes) * ave_degree(nodes)) Worst case running time is O(G.number_of_edges()).

Parameters

- **graph** (*nx.Graph*) – a graph
- **nodes** (*set*) – set of nodes

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.nx_utils import *  # NOQA
>>> import utool as ut
>>> G = demodata_bridge()
>>> nodes = {1, 2, 3, 4}
>>> outgoing = edges_outgoing(G, nodes)
>>> assert outgoing == {(3, 5), (4, 8)}
```

wbia.algo.graph.nx_utils.**ensure_multi_index**(*index*, *names*)

wbia.algo.graph.nx_utils.**group_name_edges**(*g*, *node_to_label*)

wbia.algo.graph.nx_utils.**is_complete**(*G*, *self_loops=False*)

wbia.algo.graph.nx_utils.**is_k_edge_connected**(*G*, *k*)

wbia.algo.graph.nx_utils.**k_edge_augmentation**(*G*, *k*, *avail=None*, *partial=False*)

wbia.algo.graph.nx_utils.**random_k_edge_connected_graph**(*size*, *k*, *p=0.1*, *rng=None*)

Super hacky way of getting a random k-connected graph

Example

```
>>> # ENABLE_DOCTEST
>>> import wbia.plottool as pt
>>> from wbia.algo.graph.nx_utils import *  # NOQA
>>> size, k, p = 25, 3, .1
>>> rng = ut.ensure_rng(0)
>>> gs = []
>>> for x in range(4):
>>>     G = random_k_edge_connected_graph(size, k, p, rng)
>>>     gs.append(G)
>>> ut.quit_if_noshow()
>>> pnum_ = pt.make_pnum_nextgen(nRows=2, nSubplots=len(gs))
>>> fnum = 1
>>> for g in gs:
>>>     pt.show_nx(g, fnum=fnum, pnum=pnum_())
```

1.1.1.2.19 wbia.algo.graph.refresh module

class wbia.algo.graph.refresh.**RefreshCriteria**(*window=20*, *patience=72*, *thresh=0.1*, *method='binomial'*)

Bases: object

Determine when to re-query for candidate edges.

Models an upper bound on the probability that any of the next *patience* reviews will be label-changing (meaningful). Once this probability is below a threshold the criterion triggers. The model is either binomial or poisson. They both work about the same. The binomial is a slightly better model.

Does this by maintaining an estimate of the probability any particular review will be label-chaging using an exponentially weighted moving average. This is the rate parameter / individual event probability.

add(*meaningful*, *user_id*, *decision=None*)

ave(*method='exp'*)

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.refresh import *  # NOQA
>>> from wbia.algo.graph import demo
>>> infr = demo.demodata_infr(num_pccs=40, size=4, size_std=2, ignore_
↪pair=True)
>>> edges = list(infr.dummy_verif.find_candidate_edges(K=100))
>>> scores = np.array(infr.dummy_verif.predict_edges(edges))
>>> #sortx = ut.shuffle(np.arange(len(edges)), rng=321)
>>> sortx = scores.argsort()[::-1]
>>> edges = ut.take(edges, sortx)
>>> scores = scores[sortx]
>>> ys = infr.match_state_df(edges)[POSTV].values
>>> y_remainsum = ys[::-1].cumsum()[::-1]
>>> refresh = RefreshCriteria(window=250)
```

(continues on next page)

(continued from previous page)

```
>>> ma1 = []
>>> ma2 = []
>>> reals = []
>>> xdata = []
>>> for count, (edge, y) in enumerate(zip(edges, ys)):
>>>     refresh.add(y, user_id='user:oracle')
>>>     ma1.append(refresh._ewma)
>>>     ma2.append(refresh.pos_frac)
>>>     n_real = y_remainsum[count] / (len(edges) - count)
>>>     reals.append(n_real)
>>>     xdata.append(count + 1)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.qtensure()
>>> pt.multi_plot(xdata, [ma1, ma2, reals], marker='',
>>>               label_list=['exp', 'win', 'real'], xlabel='review num',
>>>               ylabel='mu')
```

check()

clear()

pos_frac

pred_num_positives(*n_remain_edges*)

Uses poisson process to estimate remaining positive reviews.

Multipling mu * n_remain_edges gives a probabilistic upper bound on the number of errors remaning. This only provides a real estimate if reviewing in a random order

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.refresh import *  # NOQA
>>> from wbia.algo.graph import demo
>>> infr = demo.demodata_infr(num_pccs=50, size=4, size_std=2)
>>> edges = list(infr.dummy_verif.find_candidate_edges(K=100))
>>> #edges = ut.shuffle(sorted(edges), rng=321)
>>> scores = np.array(infr.dummy_verif.predict_edges(edges))
>>> sortx = scores.argsort()[::-1]
>>> edges = ut.take(edges, sortx)
>>> scores = scores[sortx]
>>> ys = infr.match_state_df(edges)[POSTV].values
>>> y_remainsum = ys[::-1].cumsum()[::-1]
>>> refresh = RefreshCriteria(window=250)
>>> n_pred_list = []
>>> n_real_list = []
>>> xdata = []
>>> for count, (edge, y) in enumerate(zip(edges, ys)):
>>>     refresh.add(y, user_id='user:oracle')
>>>     n_remain_edges = len(edges) - count
>>>     n_pred = refresh.pred_num_positives(n_remain_edges)
>>>     n_real = y_remainsum[count]
>>>     if count == 2000:
>>>         break
>>>     n_real_list.append(n_real)
```

(continues on next page)

(continued from previous page)

```
>>>     n_pred_list.append(n_pred)
>>>     xdata.append(count + 1)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.qtensure()
>>> n_pred_list = n_pred_list[10:]
>>> n_real_list = n_real_list[10:]
>>> xdata = xdata[10:]
>>> pt.multi_plot(xdata, [n_pred_list, n_real_list], marker='',
>>>               label_list=['pred', 'real'], xlabel='review num',
>>>               ylabel='pred remaining merges')
>>> stop_point = xdata[np.where(y_remainsum[10:] == 0)[0][0]]
>>> pt.gca().plot([stop_point, stop_point], [0, int(max(n_pred_list))], 'g-')
```

prob_any_remain(*n_remain_edges=None*)

wbia.algo.graph.refresh.**demo_refresh**()

CommandLine:

python -m wbia.algo.graph.refresh demo_refresh –num_pccs=40 –size=2 –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.graph.refresh import *  # NOQA
>>> demo_refresh()
>>> ut.show_if_requested()
```

1.1.1.2.20 wbia.algo.graph.state module

1.1.1.2.21 Module contents

wbia.algo.graph.**IMPORT_TUPLES = []**
cd /home/joncrall/code/wbia/wbia/algo/graph makeinit.py –modname=wbia.algo.graph

Type Regen Command

wbia.algo.graph.**reassign_submodule_attributes**(*verbose=1*)
Updates attributes in the __init__ modules with updated attributes in the submodules.

wbia.algo.graph.**reload_subs**(*verbose=1*)
Reloads wbia.algo.graph and submodules

wbia.algo.graph.**rrrr**(*verbose=1*)
Reloads wbia.algo.graph and submodules

1.1.1.3 wbia.algo.hots package

1.1.1.3.1 Submodules

1.1.1.3.2 wbia.algo.hots._pipeline_helpers module

wbia.algo.hots._pipeline_helpers.**testdata_post_sver**(*defaultdb='PZ_MTEST'*, *qaid_list=None*, *daid_list=None*, *codename='vsmany'*, *cfgdict=None*)

```
>>> from wbia.algo.hots._pipeline_helpers import *  # NOQA
```

wbia.algo.hots._pipeline_helpers.**testdata_pre**(*stopnode*, *defaultdb='testdb1'*, *p=['default']*, *a=['default:qindex=0:1,dindex=0:5']*, ***kwargs*)

New (1-1-2016) generic pipeline node testdata getter

Parameters

- **stopnode** (*str*) – name of pipeline function to be tested
- **defaultdb** (*str*) – (default = u'testdb1')
- **p** (*list*) – (default = [u'default:'])
- **a** (*list*) – (default = [u'default:qsize=1,dsize=4'])
- ****kwargs** – passed to **testdata_qreq_** qaid_override, daid_override

Returns (ibs, **qreq_**, args)

Return type tuple

CommandLine: python -m wbia.algo.hots._pipeline_helpers –exec-testdata_pre –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots._pipeline_helpers import *  # NOQA
>>> stopnode = 'build_chipmatches'
>>> defaultdb = 'testdb1'
>>> p = ['default:']
>>> a = ['default:qindex=0:1,dindex=0:5']
>>> qreq_, args = testdata_pre(stopnode, defaultdb, p, a)
```

wbia.algo.hots._pipeline_helpers.**testdata_pre_baselinefilter**(*defaultdb='testdb1'*, *qaid_list=None*, *daid_list=None*, *codename='vsmany'*)

wbia.algo.hots._pipeline_helpers.**testdata_pre_sver**(*defaultdb='PZ_MTEST'*, *qaid_list=None*, *daid_list=None*)

```
>>> from wbia.algo.hots._pipeline_helpers import *  # NOQA
```

wbia.algo.hots._pipeline_helpers.**testdata_sparse_matchinfo_nonagg**(*defaultdb='testdb1'*, *p=['default']*)

wbia.algo.hots._pipeline_helpers.**testrun_pipeline_upto**(*qreq_*, *stop_node='end'*, *verbose=True*)

Main tester function. Runs the pipeline by mirroring *request_wbia_query_L0*, but stops at a requested breakpoint and returns the local variables.

convinience: runs pipeline for tests this should mirror request_wbia_query_L0

Ignore:

```
>>> # TODO: autogenerate
>>> # The following is a stub that starts the autogeneration process
>>> import utool as ut
>>> from wbia.algo.hots import pipeline
>>> source = ut.get_func_sourcecode(pipeline.request_wbia_query_L0,
>>>                                 strip_docstr=True, stripdef=True,
>>>                                 strip_comments=True)
>>> import re
>>> source = re.sub(r'^\s*$\n', '', source, flags=re.MULTILINE)
>>> print(source)
>>> ut.replace_between_tags(source, '', sentinal)
```

1.1.1.3.3 wbia.algo.hots.chip_match module

python -m utool.util_inspect check_module_usage –pat=”chip_match.py”

class wbia.algo.hots.chip_match.**AnnotMatch**(**args*, ***kwargs*)

Bases: *wbia.algo.hots.chip_match.MatchBaseIO*, utool.util_dev.NiceRepr, wbia.algo.hots.chip_match._BaseVisualization, wbia.algo.hots.chip_match._AnnotMatchConvenienceGetter

This implements part the match between whole annotations and the other annotaions / names. This does not include algorithm specific feature matches.

evaluate_dnids(*qreq_=None*, *ibs=None*)

classmethod from_dict(*class_dict*, *ibs=None*)

Convert dict of arguments back to ChipMatch object

initialize(*qaid=None*, *daid_list=None*, *score_list=None*, *dnid_list=None*, *qnid=None*, *unique_nids=None*, *name_score_list=None*, *annot_score_list=None*, *autoinit=True*)

qaid and daid_list are not optional. fm_list and fsv_list are strongly encouraged and will probalby break things if they are not there.

set_cannonical_annot_score(*annot_score_list*)

set_cannonical_name_score(*annot_score_list*, *name_score_list*)

to_dict(*ibs=None*)

class wbia.algo.hots.chip_match.**ChipMatch**(**args*, ***kwargs*)

Bases: wbia.algo.hots.chip_match._ChipMatchVisualization, *wbia.algo.hots.chip_match.AnnotMatch*, wbia.algo.hots.chip_match._ChipMatchScorers, wbia.algo.hots.old_chip_match._OldStyleChipMatchSimulator, wbia.algo.hots.chip_match._ChipMatchConvenienceGetter, wbia.algo.hots.chip_match._ChipMatchDebugger

behaves as as the ChipMatchOldTup named tuple until we completely replace the old structure

append_featscore_column(*filtkey*, *filtweight_list*, *inplace=True*)

arraycast_self()
Ensures internal structure is in numpy array formats TODO: come up with better name Remove old initialize method and rename to initialize?

classmethod combine_cms(*cm_list*)

Example

```
>>> # FIXME failing-test (22-Jul-2020) This test is failing and it's not␣
↪clear how to fix it
>>> # xdoctest: +SKIP
>>> from wbia.core_annots import *  # NOQA
>>> ibs, depc, aid_list = testdata_core(size=4)
>>> request = depc.new_request('vsone', [1], [2, 3, 4], {'dim_size': 450})
>>> rawres_list2 = request.execute(postprocess=False)
>>> cm_list = ut.take_column(rawres_list2, 1)
>>> out = ChipMatch.combine_cms(cm_list)
>>> out.score_name_nsum(request)
>>> ut.quit_if_noshow()
>>> out.ishow_analysis(request)
>>> ut.show_if_requested()
```

compress_annots(*flags*, *inplace=False*, *keepscores=True*)

compress_results(*inplace=False*)

compress_top_feature_matches(*num=10*, *rng=<module 'numpy.random' from '/home/docs/checkouts/readthedocs.org/user_builds/wildbook-ia/envs/latest/lib/python3.7/site-packages/numpy/random/__init__.py'>*, *use_random=True*)
DO NOT USE

FIXME: Use boolean lists

Removes all but the best feature matches for testing purposes rng = np.random.RandomState(0)

extend_results(*qreq_*, *other_aids=None*)
Return a new ChipMatch containing empty data for an extended set of aids

Parameters

- **qreq** (*wbia.QueryRequest*) – query request object with hyper-parameters
- **other_aids** (*None*) – (default = None)

Returns out

Return type wbia.ChipMatch

CommandLine: python -m wbia.algo.hots.chip_match –exec-extend_results –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.chip_match import *  # NOQA
>>> import wbia
>>> import wbia
```

(continues on next page)

(continued from previous page)

```
>>> cm, qreq_ = wbia.testdata_cm('PZ_MTEST',
>>>                              a='default:dindex=0:10,qindex=0:1',
>>>                              t='best:SV=False')
>>> assert len(cm.daid_list) == 9
>>> cm.assert_self(qreq_)
>>> other_aids = qreq_.ibs.get_valid_aids()
>>> out = cm.extend_results(qreq_, other_aids)
>>> assert len(out.daid_list) == 118
>>> out.assert_self(qreq_)
```

classmethod from_dict(*class_dict*, *ibs=None*)
Convert dict of arguments back to ChipMatch object

classmethod from_json(*json_str*)
Convert json string back to ChipMatch object

CommandLine: # FIXME: util_test is broken with classmethods python -m wbia.algo.hots.chip_match –test-from_json –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.chip_match import *  # NOQA
>>> import wbia
>>> cm1, qreq_ = wbia.testdata_cm()
>>> json_str = cm1.to_json()
>>> cm = ChipMatch.from_json(json_str)
>>> ut.quit_if_noshow()
>>> cm.score_name_nsum(qreq_)
>>> cm.show_single_namematch(qreq_, 1)
>>> ut.show_if_requested()
```

get_fpath(*qreq_*)

initialize(*qaid=None*, *daid_list=None*, *fm_list=None*, *fsv_list=None*, *fk_list=None*, *score_list=None*, *H_list=None*, *fsv_col_lbls=None*, *dnid_list=None*, *qnid=None*, *unique_nids=None*, *name_score_list=None*, *annot_score_list=None*, *autoinit=True*, *filtnorm_aids=None*, *filtnorm_fxs=None*)
qaid and daid_list are not optional. fm_list and fsv_list are strongly encouraged and will probalby break things if they are not there.

classmethod load_from_fpath(*fpath*, *verbose=None*)

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

save(*qreq_*, *verbose=None*)

shortlist_subset(*top_aids*)
returns a new cmtup_old with only the requested daids TODO: rectify with take_feature_matches

sortself()
reorders the internal data using cm.score_list

take_annots(*idx_list*, *inplace=False*, *keepscores=True*)
Keeps results only for the selected annotation indices.

CommandLine: python -m wbia.algo.hots.chip_match take_annots

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.chip_match import *  # NOQA
>>> import wbia
>>> cm, qreq_ = wbia.testdata_cm('PZ_MTEST',
>>>                              a='default:dindex=0:10,qindex=0:1',
>>>                              t='best:sv=False')
>>> idx_list = list(range(cm.num_daids))
>>> inplace = False
>>> keepscores = True
>>> other = out = cm.take_annots(idx_list, inplace, keepscores)
>>> result = ('out = %s' % (ut.repr2(out, nl=1),))
>>> # Because the subset was all aids in order, the output
>>> # ChipMatch should be exactly the same.
>>> assert cm.inspect_difference(out), 'Should be exactly equal!'
>>> print(result)
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.chip_match import *  # NOQA
>>> import wbia
>>> cm, qreq_ = wbia.testdata_cm('PZ_MTEST',
>>>                              a='default:dindex=0:10,qindex=0:1',
>>>                              t='best:SV=False')
>>> idx_list = [0, 2]
>>> inplace = False
>>> keepscores = True
>>> other = out = cm.take_annots(idx_list, inplace, keepscores)
>>> result = ('out = %s' % (ut.repr2(out, nl=1),))
>>> print(result)
```

take_feature_matches(*indicies_list*, *inplace=False*, *keepscores=True*)

Removes outlier feature matches TODO: rectify with shortlist_subset

Parameters

- **indicies_list** (*list*) – list of lists of indicies to keep. if an item is None, the match to the corresponding daid is removed.
- **inplace** (*bool*) – (default = False)

Returns out

Return type wbia.ChipMatch

CommandLine: python -m wbia.algo.hots.chip_match –exec-take_feature_matches –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.chip_match import *  # NOQA
>>> import wbia
>>> cm, qreq_ = wbia.testdata_cm('PZ_MTEST', a='default:dindex=0:10,qindex=0:1
↪', t='best:SV=False')
```

(continues on next page)

(continued from previous page)

```
>>> indicies_list = [list(range(i + 1)) for i in range(cm.num_daids)]
>>> inplace = False
>>> keepscores = True
>>> out = cm.take_feature_matches(indicies_list, inplace, keepscores)
>>> assert not cm.inspect_difference(out, verbose=False), 'should be different
↪'
>>> result = ('out = %s' % (ut.repr2(out),))
>>> print(result)
```

to_json()

Serialize ChipMatch object as JSON string

CommandLine: python -m wbia.algo.hots.chip_match –test-ChipMatch.to_json:0 python -m wbia.algo.hots.chip_match –test-ChipMatch.to_json python -m wbia.algo.hots.chip_match –test-ChipMatch.to_json:1 –show

Example

```
>>> # ENABLE_DOCTEST
>>> # Simple doctest demonstrating the json format
>>> from wbia.algo.hots.chip_match import *  # NOQA
>>> import wbia
>>> cm, qreq_ = wbia.testdata_cm()
>>> cm.compress_top_feature_matches(num=4, rng=np.random.RandomState(0))
>>> # Serialize
>>> print('\n\nRaw ChipMatch JSON:\n')
>>> json_str = cm.to_json()
>>> print(json_str)
>>> print('\n\nPretty ChipMatch JSON:\n')
>>> # Pretty String Formatting
>>> dictrep = ut.from_json(json_str)
>>> dictrep = ut.delete_dict_keys(dictrep, [key for key, val in dictrep.
↪items() if val is None])
>>> result  = ut.repr2_json(dictrep, nl=2, precision=2, key_order_metric=
↪'strlen')
>>> print(result)
```

Example

```
>>> # ENABLE_DOCTEST
>>> # test to convert back and forth from json
>>> from wbia.algo.hots.chip_match import *  # NOQA
>>> import wbia
>>> cm, qreq_ = wbia.testdata_cm()
>>> cm1 = cm
>>> # Serialize
>>> json_str = cm.to_json()
>>> print(repr(json_str))
>>> # Unserialize
>>> cm = ChipMatch.from_json(json_str)
>>> # Show if it works
>>> ut.quit_if_noshow()
>>> cm.score_name_nsum(qreq_)
```

(continues on next page)

(continued from previous page)

```
>>> cm.show_single_namematch(qreq_, 1)
>>> ut.show_if_requested()
>>> # result = ('json_str = \n%s' % (str(json_str),))
>>> # print(result)
```

class wbia.algo.hots.chip_match.**MatchBaseIO**
Bases: object

copy()

classmethod load_from_fpath(*fpath*, *verbose=False*)

save_to_fpath(*fpath*, *verbose=False*)

CommandLine: python wbia –tf MatchBaseIO.save_to_fpath –verbtest –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.chip_match import *  # NOQA
>>> qaid = 18
>>> ibs, qreq_, cm_list = plh.testdata_pre_sver('PZ_MTEST', qaid_list=[qaid])
>>> cm = cm_list[0]
>>> cm.score_name_nsum(qreq_)
>>> dpath = ut.get_app_resource_dir('wbia')
>>> fpath = join(dpath, 'tmp_chipmatch.cPkl')
>>> ut.delete(fpath)
>>> cm.save_to_fpath(fpath)
>>> cm2 = ChipMatch.load_from_fpath(fpath)
>>> assert cm == cm2
>>> ut.quit_if_noshow()
>>> cm.ishow_analysis(qreq_)
>>> ut.show_if_requested()
```

exception wbia.algo.hots.chip_match.**NeedRecomputeError**
Bases: Exception

class wbia.algo.hots.chip_match.**TestLogger**(*verbose=True*)
Bases: object

context(*name*)

end_test()

log_failed(*msg*)

log_passed(*msg*)

log_skipped(*msg*)

skip_test()

start_test(*name*)

wbia.algo.hots.chip_match.**aslist**(*arr*)

wbia.algo.hots.chip_match.**check_arrs_eq**(*arr1*, *arr2*)

wbia.algo.hots.chip_match.**convert_numpy**(*arr*, *dtype*)

wbia.algo.hots.chip_match.**convert_numpy_lists**(*arr_list*, *dtype*, *dims=None*)

wbia.algo.hots.chip_match.**extend_nplists**(*x_list*, *num*, *shape*, *dtype*)

wbia.algo.hots.chip_match.**extend_nplists_**(*x_list*, *num*, *shape*, *dtype*)

wbia.algo.hots.chip_match.**extend_pylist**(*x_list*, *num*, *val*)

wbia.algo.hots.chip_match.**extend_pylist_**(*x_list*, *num*, *val*)

wbia.algo.hots.chip_match.**extend_scores**(*vals*, *num*)

wbia.algo.hots.chip_match.**filtnorm_op**(*filtnorm_*, *op_*, **args*, ***kwargs*)

wbia.algo.hots.chip_match.**get_chipmatch_fname**(*qaid*, *qreq_*, *qauuid=None*, *cfgstr=None*, *TRUNCATE_UUIDS=False*, *MAX_FNAME_LEN=200*)

CommandLine: python -m wbia.algo.hots.chip_match –test-get_chipmatch_fname

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.chip_match import *  # NOQA
>>> qreq_, args = plh.testdata_pre('spatial_verification',
>>>                                defaultdb='PZ_MTEST', qaid_override=[18],
>>>                                p='default:sqrd_dist_on=True')
>>> cm_list = args.cm_list_FILT
>>> cm = cm_list[0]
>>> fname = get_chipmatch_fname(cm.qaid, qreq_, qauuid=None,
>>>                             TRUNCATE_UUIDS=False, MAX_FNAME_LEN=200)
>>> result = fname
>>> print(result)
```

qaid=18_cm_cvgrsbnffsgifyom_quuid=a126d459-b730-573e-7a21-92894b016565.cPkl

wbia.algo.hots.chip_match.**prepare_dict_uuids**(*class_dict*, *ibs*)
Hacks to ensure proper uuid conversion

wbia.algo.hots.chip_match.**safe_check_lens_eq**(*arr1*, *arr2*, *msg=None*)
Check if it is safe to check if two arrays are equal

safe_check_lens_eq(None, 1) safe_check_lens_eq([3], [2, 4])

wbia.algo.hots.chip_match.**safe_check_nested_lens_eq**(*arr1*, *arr2*)
Check if it is safe to check if two arrays are equal (nested)

safe_check_nested_lens_eq(None, 1) safe_check_nested_lens_eq([[3, 4]], [[2, 4]]) safe_check_nested_lens_eq([[1, 2, 3], [1, 2]], [[1, 2, 3], [1, 2]]) safe_check_nested_lens_eq([[1, 2, 3], [1, 2]], [[1, 2, 3], [1]])

wbia.algo.hots.chip_match.**safecast_numpy_lists**(*arr_list*, *dtype=None*, *dims=None*)

wbia.algo.hots.chip_match.**safeop**(*op_*, *xs*, **args*, ***kwargs*)

wbia.algo.hots.chip_match.**testdata_cm**()

1.1.1.3.4 wbia.algo.hots.exceptions module

exception wbia.algo.hots.exceptions.**HotsCacheMissError**
Bases: Exception

exception wbia.algo.hots.exceptions.**HotsNeedsRecomputeError**
Bases: Exception

wbia.algo.hots.exceptions.**NoDescriptorsException**(*ibs*, *qaid*)

exception wbia.algo.hots.exceptions.**QueryException**(*msg*)
Bases: Exception

1.1.1.3.5 wbia.algo.hots.hstypes module

hstypes Todo: * SIFT: Root_SIFT -> L2 normalized -> Centering. # http://hal.archives-ouvertes.fr/docs/00/84/07/21/PDF/RR-8325.pdf The devil is in the deatails http://www.robots.ox.ac.uk/~vilem/bmvc2011.pdf This says dont clip, do rootsift instead # http://hal.archives-ouvertes.fr/docs/00/68/81/69/PDF/hal_v1.pdf * Quantization of residual vectors * Burstiness normalization for N-SMK * Implemented A-SMK * Incorporate Spatial Verification * Implement correct cfgstrs based on algorithm input for cached computations. * Color by word * Profile on hyrule * Train vocab on paris * Remove self matches. * New SIFT parameters for pyhesaff (root, powerlaw, meanwhatever, output_dtype)

Todo: This needs to be less constant when using non-sift descriptors

Issues: * 10GB are in use when performing query on Oxford 5K * errors when there is a word without any database vectors. currently a weight of zero is hacked in

class wbia.algo.hots.hstypes.**FiltKeys**
Bases: object

BARL2 = 'bar_l2'

DIST = 'dist'

DISTINCTIVENESS = 'distinctiveness'

FG = 'fg'

HOMOGERR = 'homogerr'

LNBNN = 'lnbnn'

RATIO = 'ratio'

wbia.algo.hots.hstypes.**PSEUDO_UINT8_MAX_SQRD = 262144.0**
vt.distance.understanding_pseudomax_props

Type SeeAlso

1.1.1.3.6 wbia.algo.hots.match_chips4 module

Runs functions in pipeline to get query reuslts and does some caching.

wbia.algo.hots.match_chips4.**execute_query2**(*qreq_*, *verbose*, *save_qcache*, *batch_size=None*, *use_supercache=False*)
Breaks up query request into several subrequests to process "more efficiently" and safer as well.

wbia.algo.hots.match_chips4.**execute_query_and_save_L1**(*qreq_*, *use_cache*, *save_qcache*, *verbose=True*, *batch_size=None*, *use_supercache=False*, *invalidate_supercache=False*)

Parameters

- **qreq** (*wbia.QueryRequest*) –
- **use_cache** (*bool*) –

Returns qaid2_cm

CommandLine: python -m wbia.algo.hots.match_chips4 execute_query_and_save_L1:0 python -m wbia.algo.hots.match_chips4 execute_query_and_save_L1:1 python -m wbia.algo.hots.match_chips4 execute_query_and_save_L1:2 python -m wbia.algo.hots.match_chips4 execute_query_and_save_L1:3

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +SKIP
>>> from wbia.algo.hots.match_chips4 import *  # NOQA
>>> cfgdict1 = dict(codename='vsmany', sv_on=True)
>>> p = 'default' + ut.get_cfg_lbl(cfgdict1)
>>> qreq_ = wbia.main_helpers.testdata_qreq_(p=p, qaid_override=[1, 2, 3, 4])
>>> use_cache, save_qcache, verbose = False, False, True
>>> qaid2_cm = execute_query_and_save_L1(qreq_, use_cache, save_qcache, verbose)
>>> print(qaid2_cm)
```

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +SKIP
>>> from wbia.algo.hots.match_chips4 import *  # NOQA
>>> cfgdict1 = dict(codename='vsone', sv_on=True)
>>> p = 'default' + ut.get_cfg_lbl(cfgdict1)
>>> qreq_ = wbia.main_helpers.testdata_qreq_(p=p, qaid_override=[1, 2, 3, 4])
>>> use_cache, save_qcache, verbose = False, False, True
>>> qaid2_cm = execute_query_and_save_L1(qreq_, use_cache, save_qcache, verbose)
>>> print(qaid2_cm)
```

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +SKIP
>>> # TEST SAVE
>>> from wbia.algo.hots.match_chips4 import *  # NOQA
>>> import wbia
>>> cfgdict1 = dict(codename='vsmany', sv_on=True)
>>> p = 'default' + ut.get_cfg_lbl(cfgdict1)
>>> qreq_ = wbia.main_helpers.testdata_qreq_(p=p, qaid_override=[1, 2, 3, 4])
>>> use_cache, save_qcache, verbose = False, True, True
>>> qaid2_cm = execute_query_and_save_L1(qreq_, use_cache, save_qcache, verbose)
>>> print(qaid2_cm)
```

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +SKIP
>>> # TEST LOAD
>>> from wbia.algo.hots.match_chips4 import *  # NOQA
>>> import wbia
>>> cfgdict1 = dict(codename='vsmany', sv_on=True)
>>> p = 'default' + ut.get_cfg_lbl(cfgdict1)
>>> qreq_ = wbia.main_helpers.testdata_qreq_(p=p, qaid_override=[1, 2, 3, 4])
>>> use_cache, save_qcache, verbose = True, True, True
>>> qaid2_cm = execute_query_and_save_L1(qreq_, use_cache, save_qcache, verbose)
>>> print(qaid2_cm)
```

Example

```
>>> # ENABLE_DOCTEST
>>> # TEST PARTIAL HIT
>>> from wbia.algo.hots.match_chips4 import *  # NOQA
>>> import wbia
>>> cfgdict1 = dict(codename='vsmany', sv_on=False, prescore_method='csum')
>>> p = 'default' + ut.get_cfg_lbl(cfgdict1)
>>> qreq_ = wbia.main_helpers.testdata_qreq_(p=p, qaid_override=[1, 2, 3,
>>>                                                              4, 5, 6,
>>>                                                              7, 8, 9])
>>> use_cache, save_qcache, verbose = False, True, False
>>> qaid2_cm = execute_query_and_save_L1(qreq_, use_cache,
>>>                                      save_qcache, verbose,
>>>                                      batch_size=3)
>>> cm = qaid2_cm[1]
>>> ut.delete(cm.get_fpath(qreq_))
>>> cm = qaid2_cm[4]
>>> ut.delete(cm.get_fpath(qreq_))
>>> cm = qaid2_cm[5]
>>> ut.delete(cm.get_fpath(qreq_))
>>> cm = qaid2_cm[6]
>>> ut.delete(cm.get_fpath(qreq_))
>>> print('Re-execute')
>>> qaid2_cm_ = execute_query_and_save_L1(qreq_, use_cache,
>>>                                       save_qcache, verbose,
>>>                                       batch_size=3)
>>> assert all([qaid2_cm_[qaid] == qaid2_cm[qaid] for qaid in qreq_.qaids])
>>> [ut.delete(fpath) for fpath in qreq_.get_chipmatch_fpaths(qreq_.qaids)]
```

Ignore: other = cm_ = qaid2_cm_[qaid] cm = qaid2_cm[qaid]

wbia.algo.hots.match_chips4.**submit_query_request**(*qreq_*, *use_cache=None*, *use_bigcache=None*, *verbose=None*, *save_qcache=None*, *use_supercache=None*, *invalidate_supercache=None*)

Called from qreq_.execute

Checks a big cache for qaid2_cm. If cache miss, tries to load each cm individually. On an individual cache miss, it preforms the query.

CommandLine: python -m wbia.algo.hots.match_chips4 –test-submit_query_request

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +SKIP
>>> from wbia.algo.hots.match_chips4 import *  # NOQA
>>> import wbia
>>> qaid_list = [1]
>>> daid_list = [1, 2, 3, 4, 5]
>>> use_bigcache = True
>>> use_cache = True
>>> ibs = wbia.opendb(db='testdb1')
>>> qreq_ = ibs.new_query_request(qaid_list, daid_list, verbose=True)
>>> cm_list = submit_query_request(qreq_=qreq_)
```

1.1.1.3.7 wbia.algo.hots.name_scoring module

class wbia.algo.hots.name_scoring.**NameScoreTup**(*sorted_nids*, *sorted_nscore*, *sorted_aids*, *sorted_scores*)

Bases: tuple

sorted_aids
Alias for field number 2

sorted_nids
Alias for field number 0

sorted_nscore
Alias for field number 1

sorted_scores
Alias for field number 3

wbia.algo.hots.name_scoring.**align_name_scores_with_annots**(*annot_score_list*, *annot_aid_list*, *daid2_idx*, *name_groupxs*, *name_score_list*)

takes name scores and gives them to the best annotation

Returns list of scores aligned with cm.daid_list and cm.dnid_list

Return type score_list

Parameters

- **annot_score_list** (*list*) – score associated with each annot
- **name_groupxs** (*list*) – groups annot_score lists into groups compatible with name_score_list
- **name_score_list** (*list*) – score assocated with name
- **nid2_nidx** (*dict*) – mapping from nids to index in name score list

CommandLine: python -m wbia.algo.hots.name_scoring –test-align_name_scores_with_annots python -m wbia.algo.hots.name_scoring –test-align_name_scores_with_annots –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.name_scoring import *  # NOQA
>>> ibs, qreq_, cm_list = plh.testdata_post_sver('PZ_MTEST', qaid_list=[18])
>>> cm = cm_list[0]
>>> cm.evaluate_csum_annot_score(qreq_)
>>> cm.evaluate_nsum_name_score(qreq_)
>>> # Annot aligned lists
>>> annot_score_list = cm.algo_annot_scores['csum']
>>> annot_aid_list   = cm.daid_list
>>> daid2_idx        = cm.daid2_idx
>>> # Name aligned lists
>>> name_score_list  = cm.algo_name_scores['nsum']
>>> name_groupxs     = cm.name_groupxs
>>> # Execute Function
>>> score_list = align_name_scores_with_annots(annot_score_list, annot_aid_list,␣
↪daid2_idx, name_groupxs, name_score_list)
>>> # Check that the correct name gets the highest score
>>> target = name_score_list[cm.nid2_nidx[cm.qnid]]
>>> test_index = np.where(score_list == target)[0][0]
>>> cm.score_list = score_list
>>> ut.assert_eq(ibs.get_annot_name_rowids(cm.daid_list[test_index]), cm.qnid)
>>> assert ut.isunique(cm.dnid_list[score_list > 0]), 'bad name score'
>>> top_idx = cm.algo_name_scores['nsum'].argmax()
>>> assert cm.get_top_nids()[0] == cm.unique_nids[top_idx], 'bug in alignment'
>>> ut.quit_if_noshow()
>>> cm.show_ranked_matches(qreq_)
>>> ut.show_if_requested()
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots.name_scoring import *  # NOQA
>>> annot_score_list = []
>>> annot_aid_list   = []
>>> daid2_idx        = {}
>>> # Name aligned lists
>>> name_score_list  = np.array([], dtype=np.float32)
>>> name_groupxs     = []
>>> # Execute Function
>>> score_list = align_name_scores_with_annots(annot_score_list, annot_aid_list,␣
↪daid2_idx, name_groupxs, name_score_list)
```

wbia.algo.hots.name_scoring.**compute_fmech_score**(*cm*, *qreq_=None*, *hack_single_ori=False*)

nsum. This is the fmech scoring mechanism.

Parameters **cm** (*wbia.ChipMatch*) –

Returns (unique_nids, nsum_score_list)

Return type tuple

CommandLine: python -m wbia.algo.hots.name_scoring –test-compute_fmech_score python -m wbia.algo.hots.name_scoring –test-compute_fmech_score:0 python -m wbia.algo.hots.name_scoring –test-compute_fmech_score:2 utprof.py -m wbia.algo.hots.name_scoring –test-compute_fmech_score:2

utprof.py -m wbia.algo.hots.pipeline –test-request_wbia_query_L0:0 –db PZ_Master1 -a timectrl:qindex=0:256

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.name_scoring import *  # NOQA
>>> cm = testdata_chipmatch()
>>> nsum_score_list = compute_fmech_score(cm)
>>> assert np.all(nsum_score_list == [ 4.,  7.,  5.])
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.name_scoring import *  # NOQA
>>> ibs, qreq_, cm_list = plh.testdata_post_sver('PZ_MTEST', qaid_list=[18])
>>> cm = cm_list[0]
>>> cm.evaluate_dnids(qreq_)
>>> cm._cast_scores()
>>> #cm.qnid = 1    # Hack for testdb1 names
>>> nsum_score_list = compute_fmech_score(cm, qreq_)
>>> #assert np.all(nsum_nid_list == cm.unique_nids), 'nids out of alignment'
>>> flags = (cm.unique_nids == cm.qnid)
>>> max_true = nsum_score_list[flags].max()
>>> max_false = nsum_score_list[~flags].max()
>>> assert max_true > max_false, 'is this truely a hard case?'
>>> assert max_true > 1.2, 'score=%r should be higher for aid=18' % (max_true,)
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.name_scoring import *  # NOQA
>>> ibs, qreq_, cm_list = plh.testdata_post_sver('PZ_MTEST', qaid_list=[18],␣
↪cfgdict=dict(query_rotation_heuristic=True))
>>> cm = cm_list[0]
>>> cm.score_name_nsum(qreq_)
>>> ut.quit_if_noshow()
>>> cm.show_ranked_matches(qreq_, ori=True)
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots.name_scoring import *  # NOQA
>>> #ibs, qreq_, cm_list = plh.testdata_pre_sver('testdb1', qaid_list=[1])
>>> ibs, qreq_, cm_list = plh.testdata_post_sver('testdb1', qaid_list=[1],␣
↪cfgdict=dict(query_rotation_heuristic=True))
>>> cm = cm_list[0]
>>> cm.score_name_nsum(qreq_)
>>> ut.quit_if_noshow()
>>> cm.show_ranked_matches(qreq_, ori=True)
```

wbia.algo.hots.name_scoring.**get_chipmatch_namescore_nonvoting_feature_flags**(*cm*, *qreq_=None*)

DEPRICATE

Computes flags to desribe which features can or can not vote

CommandLine: python -m wbia.algo.hots.name_scoring –exec-get_chipmatch_namescore_nonvoting_feature_flags

Example

```
>>> # ENABLE_DOCTEST
>>> # FIXME: breaks when fg_on=True
>>> from wbia.algo.hots.name_scoring import *  # NOQA
>>> from wbia.algo.hots import name_scoring
>>> # Test to make sure name score and chips score are equal when per_name=1
>>> qreq_, args = plh.testdata_pre('spatial_verification', defaultdb='PZ_MTEST',␣
↪a=['default:dpername=1,qsize=1,dsize=10'], p=['default:K=1,fg_on=True'])
>>> cm_list = args.cm_list_FILT
>>> ibs = qreq_.ibs
>>> cm = cm_list[0]
>>> cm.evaluate_dnids(qreq_)
>>> featflat_list = get_chipmatch_namescore_nonvoting_feature_flags(cm, qreq_)
>>> assert all(list(map(np.all, featflat_list))), 'all features should be able to␣
↪vote in K=1, per_name=1 case'
```

wbia.algo.hots.name_scoring.**get_namescore_nonvoting_feature_flags**(*fm_list*, *fs_list*, *dnid_list*, *name_groupxs*, *kpts1=None*)

DEPRICATE

fm_list = [fm[:min(len(fm), 10)] for fm in fm_list] fs_list = [fs[:min(len(fs), 10)] for fs in fs_list]

wbia.algo.hots.name_scoring.**testdata_chipmatch**()

1.1.1.3.8 wbia.algo.hots.neighbor_index module

Todo: Remove Bloat

multi_index.py as well

https://github.com/spotify/annoy

class wbia.algo.hots.neighbor_index.**NeighborIndex**(*flann_params*, *cfgstr*)

Bases: object

wrapper class around flann stores flann index and data it needs to index into

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index import *  # NOQA
>>> nnindexer, qreq_, ibs = testdata_nnindexer()
```

add_support(*new_daid_list*, *new_vecs_list*, *new_fgws_list*, *new_fxs_list*, *verbose=True*)
adds support data (aka data to be indexed)

Parameters

- **new_daid_list** (*list*) – list of annotation ids that are being added
- **new_vecs_list** (*list*) – list of descriptor vectors for each annotation
- **new_fgws_list** (*list*) – list of weights per vector for each annotation
- **verbose** (*bool*) – verbosity flag(default = True)

CommandLine: python -m wbia.algo.hots.neighbor_index –test-add_support

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index import *  # NOQA
>>> nnindexer, qreq_, ibs = testdata_nnindexer(use_memcache=False)
>>> new_daid_list = [2, 3, 4]
>>> K = 2
>>> qfx2_vec = ibs.get_annot_vecs(1, config2_=qreq_.get_internal_query_
↪config2())
>>> # get before data
>>> (qfx2_idx1, qfx2_dist1) = nnindexer.knn(qfx2_vec, K)
>>> new_vecs_list, new_fgws_list, new_fxs_list = get_support_data(qreq_, new_
↪daid_list)
>>> # execute test function
>>> nnindexer.add_support(new_daid_list, new_vecs_list, new_fgws_list, new_
↪fxs_list)
>>> # test before data vs after data
>>> (qfx2_idx2, qfx2_dist2) = nnindexer.knn(qfx2_vec, K)
>>> assert qfx2_idx2.max() > qfx2_idx1.max()
```

add_wbia_support(*qreq_*, *new_daid_list*, *verbose=True*)
TODO: ensure that the memcache changes appropriately

batch_knn(*vecs*, *K*, *chunksize=4096*, *label='batch knn'*)
Works like *indexer.knn* but the input is split into batches and progress is reported to give an esimated time remaining.

build_and_save(*cachedir*, *verbose=True*, *memtrack=None*)

debug_nnindexer()
Makes sure the indexer has valid SIFT descriptors

empty_neighbors(*nQfx*, *K*)

ensure_indexer(*cachedir*, *verbose=True*, *force_rebuild=False*, *memtrack=None*, *prog_hook=None*)
Ensures that you get a neighbor indexer. It either loads a chached indexer or rebuilds a new one.

ext = '.flann'

get_cfgstr(*noquery=False*)
returns string which uniquely identified configuration and support data

Parameters **noquery** (*bool*) – if True cfgstr is only relevant to building the index. No search params are returned (default = False)

Returns flann_cfgstr

Return type str

CommandLine: python -m wbia.algo.hots.neighbor_index –test-get_cfgstr

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index import *  # NOQA
>>> import wbia
>>> cfgdict = dict(fg_on=False)
>>> qreq_ = wbia.testdata_qreq_(defaultdb='testdb1', p='default:fg_on=False')
>>> qreq_.load_indexer()
>>> nnindexer = qreq_.indexer
>>> noquery = True
>>> flann_cfgstr = nnindexer.get_cfgstr(noquery)
>>> result = ('flann_cfgstr = %s' % (str(flann_cfgstr),))
>>> print(result)
flann_cfgstr = _FLANN((algo=kdtree,seed=42,t=8,))_VECS((11260,128)gj5nea@ni0
↪%f3aja)
```

get_dtype()

get_fname()

get_fpath(*cachedir*, *cfgstr=None*)

get_indexed_aids()

get_indexed_vecs()

get_nn_aids(*qfx2_nnidx*)

Parameters **qfx2_nnidx** – (N x K) qfx2_idx[n][k] is the index of the kth approximate nearest data vector

Returns

(N x K) qfx2_fx[n][k] is the annotation id index of the kth approximate nearest data vector

Return type qfx2_aid

CommandLine: python -m wbia.algo.hots.neighbor_index –exec-get_nn_aids

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index import *  # NOQA
>>> import wbia
>>> cfgdict = dict(fg_on=False)
>>> qreq_ = wbia.testdata_qreq_(defaultdb='testdb1', p='default:fg_on=False,
↪dim_size=450,resize_dim=area')
>>> qreq_.load_indexer()
>>> nnindexer = qreq_.indexer
>>> qfx2_vec = qreq_.ibs.get_annot_vecs(
>>>     qreq_.get_internal_qaids()[0],
>>>     config2_=qreq_.get_internal_query_config2())
>>> num_neighbors = 4
```

(continues on next page)

(continued from previous page)

```
>>> (qfx2_nnidx, qfx2_dist) = nnindexer.knn(qfx2_vec, num_neighbors)
>>> qfx2_aid = nnindexer.get_nn_aids(qfx2_nnidx)
>>> assert qfx2_aid.shape[1] == num_neighbors
>>> print('qfx2_aid.shape = %r' % (qfx2_aid.shape,))
>>> assert qfx2_aid.shape[1] == 4
>>> ut.assert_inbounds(qfx2_aid.shape[0], 1200, 1300)
```

get_nn_axs(*qfx2_nnidx*)
gets matching internal annotation indices

get_nn_featxs(*qfx2_nnidx*)

Parameters **qfx2_nnidx** – (N x K) qfx2_idx[n][k] is the index of the kth approximate nearest data vector

Returns

(N x K) qfx2_fx[n][k] is the feature index (w.r.t the source annotation) of the kth approximate nearest data vector

Return type qfx2_fx

get_nn_fgws(*qfx2_nnidx*)
Gets forground weights of neighbors

CommandLine: python -m wbia –tf NeighborIndex.get_nn_fgws

Parameters **qfx2_nnidx** – (N x K) qfx2_idx[n][k] is the index of the kth approximate nearest data vector

Returns

(N x K) qfx2_fgw[n][k] is the annotation id index of the kth forground weight

Return type qfx2_fgw

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index import *  # NOQA
>>> nnindexer, qreq_, ibs = testdata_nnindexer(dbname='testdb1')
>>> qfx2_nnidx = np.array([[0, 1, 2], [3, 4, 5]])
>>> qfx2_fgw = nnindexer.get_nn_fgws(qfx2_nnidx)
```

get_nn_nids(*qfx2_nnidx*, *qreq_*)
iccv hack, todo: make faster by direct lookup from idx

get_nn_vecs(*qfx2_nnidx*)
gets matching vectors

get_prefix()

get_removed_idxs()
__removed_ids = nnindexer.flann._FLANN__removed_ids invalid_idxs = nnindexer.get_removed_idxs() assert len(np.intersect1d(invalid_idxs, __removed_ids)) == len(__removed_ids)

init_support(*aid_list*, *vecs_list*, *fgws_list*, *fxs_list*, *verbose=True*)
prepares inverted indicies and FLANN data structure

flattens vecs_list and builds a reverse index from the flattened indices (idx) to the original aids and fxs

knn(*qfx2_vec*, *K*)

Returns the indices and squared distance to the nearest K neighbors. The distance is noramlized between zero and one using VEC_PSEUDO_MAX_DISTANCE = (np.sqrt(2) * VEC_PSEUDO_MAX)

Parameters

- **qfx2_vec** – (N x D) an array of N, D-dimensional query vectors
- **K** – number of approximate nearest neighbors to find

Returns: tuple of (qfx2_idx, qfx2_dist)

ndarray [qfx2_idx[n][k] (N x K) is the index of the kth] approximate nearest data vector w.r.t qfx2_vec[n]

ndarray [qfx2_dist[n][k] (N x K) is the distance to the kth] approximate nearest data vector w.r.t. qfx2_vec[n] distance is normalized squared euclidean distance.

CommandLine: python -m wbia –tf NeighborIndex.knn:0 –debug2 python -m wbia –tf NeighborIndex.knn:1

Example

```
>>> # FIXME failing-test (22-Jul-2020) This test is failing and it's not␣
↪clear how to fix it
>>> # xdoctest: +SKIP
>>> from wbia.algo.hots.neighbor_index import *  # NOQA
>>> indexer, qreq_, ibs = testdata_nnindexer()
>>> qfx2_vec = ibs.get_annot_vecs(1, config2_=qreq_.get_internal_query_
↪config2())
>>> K = 2
>>> indexer.debug_nnindexer()
>>> assert vt.check_sift_validity(qfx2_vec), 'bad SIFT properties'
>>> (qfx2_idx, qfx2_dist) = indexer.knn(qfx2_vec, K)
>>> result = str(qfx2_idx.shape) + ' ' + str(qfx2_dist.shape)
>>> print('qfx2_vec.dtype = %r' % (qfx2_vec.dtype,))
>>> print('indexer.max_distance_sqrd = %r' % (indexer.max_distance_sqrd,))
>>> assert np.all(qfx2_dist < 1.0), (
>>>     'distance should be less than 1. got %r' % (qfx2_dist,))
>>> # Ensure distance calculations are correct
>>> qfx2_dvec = indexer.idx2_vec[qfx2_idx.T]
>>> targetdist = vt.L2_sift(qfx2_vec, qfx2_dvec).T ** 2
>>> rawdist    = vt.L2_sqrd(qfx2_vec, qfx2_dvec).T
>>> assert np.all(qfx2_dist * indexer.max_distance_sqrd == rawdist), (
>>>     'inconsistant distance calculations')
>>> assert np.allclose(targetdist, qfx2_dist), (
>>>     'inconsistant distance calculations')
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index import *  # NOQA
>>> indexer, qreq_, ibs = testdata_nnindexer()
>>> qfx2_vec = np.empty((0, 128), dtype=indexer.get_dtype())
>>> K = 2
>>> (qfx2_idx, qfx2_dist) = indexer.knn(qfx2_vec, K)
```

(continues on next page)

(continued from previous page)

```
>>> result = str(qfx2_idx.shape) + ' ' + str(qfx2_dist.shape)
>>> print(result)
(0, 2) (0, 2)
```

load(*cachedir=None*, *fpath=None*, *verbose=True*)
Loads a cached flann neighbor indexer from disk (not the data)

num_indexed_annots()

num_indexed_vecs()

prefix1 = 'flann'

reindex(*verbose=True*, *memtrack=None*)
indexes all vectors with FLANN.

remove_support(*remove_daid_list*, *verbose=True*)

CommandLine: python -m wbia.algo.hots.neighbor_index –test-remove_support

SeeAlso: ~/code/flann/src/python/pyflann/index.py

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +SKIP
>>> # (IMPORTANT)
>>> from wbia.algo.hots.neighbor_index import *  # NOQA
>>> nnindexer, qreq_, ibs = testdata_nnindexer(use_memcache=False)
>>> remove_daid_list = [8, 9, 10, 11]
>>> K = 2
>>> qfx2_vec = ibs.get_annot_vecs(1, config2_=qreq_.get_internal_query_
↪config2())
>>> # get before data
>>> (qfx2_idx1, qfx2_dist1) = nnindexer.knn(qfx2_vec, K)
>>> # execute test function
>>> nnindexer.remove_support(remove_daid_list)
>>> # test before data vs after data
>>> (qfx2_idx2, qfx2_dist2) = nnindexer.knn(qfx2_vec, K)
>>> ax2_nvecs = ut.dict_take(ut.dict_hist(nnindexer.idx2_ax),␣
↪range(len(nnindexer.ax2_aid)))
>>> assert qfx2_idx2.max() < ax2_nvecs[0], 'should only get points from aid 7'
>>> assert qfx2_idx1.max() > ax2_nvecs[0], 'should get points from everyone'
```

remove_wbia_support(*qreq_*, *remove_daid_list*, *verbose=True*)
TODO: ensure that the memcache changes appropriately

requery_knn(*qfx2_vec*, *K*, *pad*, *impossible_aids*, *recover=True*)
hack for iccv - this is a highly coupled function

CommandLine: python -m wbia.algo.hots.neighbor_index requery_knn

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index import *  # NOQA
```

(continues on next page)

(continued from previous page)

```
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_(defaultdb='testdb1', a='default')
>>> qreq_.load_indexer()
>>> indexer = qreq_.indexer
>>> qannot = qreq_.internal_qannots[1]
>>> qfx2_vec = qannot.vecs
>>> K = 3
>>> pad = 1
>>> ibs = qreq_.ibs
>>> qaid = qannot.aid
>>> impossible_aids = ibs.get_annot_groundtruth(qaid, noself=False)
>>> impossible_aids = np.array([1, 2, 3, 4, 5])
>>> qfx2_idx, qfx2_dist = indexer.requery_knn(qfx2_vec, K, pad,
>>>                                           impossible_aids)
>>> #indexer.get_nn_axs(qfx2_idx)
>>> assert np.all(np.diff(qfx2_dist, axis=1) >= 0)
```

rrr(*verbose=True*, *reload_module=True*)
: special class reloading function This function is often injected as rrr of classes

save(*cachedir=None*, *fpath=None*, *verbose=True*)
: Caches a flann neighbor indexer to disk (not the data)

class wbia.algo.hots.neighbor_index.**NeighborIndex2**(*flann_params=None*, *cfgstr=None*)

Bases: *wbia.algo.hots.neighbor_index.NeighborIndex*, utool.util_dev.NiceRepr

static get_support(*depc*, *aid_list*, *config*)

on_load(*depc*)

on_save(*depc*, *fpath*)

rrr(*verbose=True*, *reload_module=True*)
: special class reloading function This function is often injected as rrr of classes

wbia.algo.hots.neighbor_index.**get_support_data**(*qreq_*, *daid_list*)

CommandLine: python -m wbia.algo.hots.neighbor_index get_support_data –show

Example

```
>>> # xdoctest: +REQUIRES(module:wbia_cnn)
>>> from wbia.algo.hots.neighbor_index import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_(defaultdb='PZ_MTEST', p=':fgw_thresh=.9,maxscale_
↪thresh=10', a=':size=2')
>>> daid_list = qreq_.daids
>>> tup  = get_support_data(qreq_, daid_list)
>>> vecs_list, fgws_list, fxs_list = tup
>>> assert all([np.all(fgws > .9) for fgws in fgws_list])
>>> result = ('depth_profile = %r' % (ut.depth_profile(tup),))
>>> print(result)
```

depth_profile = [[(128, 128), (174, 128)], [128, 174], [128, 174]]

I can't figure out why this tests isn't determenistic all the time and I can't get it to reproduce non-determenism.

This could be due to theano.

depth_profile = [[(39, 128), (22, 128)], [39, 22], [39, 22]] depth_profile = [[(35, 128), (24, 128)], [35, 24], [35, 24]] depth_profile = [[(34, 128), (31, 128)], [34, 31], [34, 31]] depth_profile = [[(83, 128), (129, 128)], [83, 129], [83, 129]] depth_profile = [[(13, 128), (104, 128)], [13, 104], [13, 104]]

wbia.algo.hots.neighbor_index.**in1d_shape**(*arr1*, *arr2*)

wbia.algo.hots.neighbor_index.**invert_index**(*vecs_list*, *fgws_list*, *ax_list*, *fxs_list*, *verbose=True*)

Aggregates descriptors of input annotations and returns inverted information

Parameters

- **vecs_list** (*list*) –
- **fgws_list** (*list*) –
- **ax_list** (*list*) –
- **fxs_list** (*list*) –
- **verbose** (*bool*) – verbosity flag(default = True)

Returns (idx2_vec, idx2_fgw, idx2_ax, idx2_fx)

Return type tuple

CommandLine: python -m wbia.algo.hots.neighbor_index invert_index

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index import *  # NOQA
>>> rng = np.random.RandomState(42)
>>> DIM_SIZE = 16
>>> nFeat_list = [3, 0, 4, 1]
>>> vecs_list = [rng.randn(nFeat, DIM_SIZE) for nFeat in nFeat_list]
>>> fgws_list = [rng.randn(nFeat) for nFeat in nFeat_list]
>>> fxs_list = [np.arange(nFeat) for nFeat in nFeat_list]
>>> ax_list = np.arange(len(vecs_list))
>>> fgws_list = None
>>> verbose = True
>>> tup = invert_index(vecs_list, fgws_list, ax_list, fxs_list)
>>> (idx2_vec, idx2_fgw, idx2_ax, idx2_fx) = tup
>>> result = 'output depth_profile = %s' % (ut.depth_profile(tup),)
>>> print(result)
output depth_profile = [(8, 16), 1, 8, 8]
```

Example

```
>>> # xdoctest: +REQUIRES(--slow)
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_(defaultdb='testdb1', a='default:species=zebra_
↪plains', p='default:fgw_thresh=.999')
>>> vecs_list, fgws_list, fxs_list = get_support_data(qreq_, qreq_.daids)
>>> ax_list = np.arange(len(vecs_list))
>>> input_ = vecs_list, fgws_list, ax_list, fxs_list
```

(continues on next page)

(continued from previous page)

```
>>> print('input depth_profile = %s' % (ut.depth_profile(input_),))
>>> tup = invert_index(*input_)
>>> (idx2_vec, idx2_fgw, idx2_ax, idx2_fx) = tup
>>> result = 'output depth_profile = %s' % (ut.depth_profile(tup),)
>>> print(result)
```

output depth_profile = [(1912, 128), 1912, 1912, 1912]

wbia.algo.hots.neighbor_index.**testdata_nnindexer**(**args*, ***kwargs*)

1.1.1.3.9 wbia.algo.hots.neighbor_index_cache module

NEEDS CLEANUP

class wbia.algo.hots.neighbor_index_cache.**UUIDMapHyrbridCache**

Bases: object

Class that lets multiple ways of writing to the uuid_map be swapped in and out interchangably

TODO: the global read / write should periodically sync itself to disk and it should be loaded from disk initially

dump(*cachedir*)

init(**args*, ***kwargs*)

load(*cachedir*)

Returns a cache UUIDMap

read_uuid_map_dict(*uuid_map_fpath*, *min_reindex_thresh*)

uses in memory dictionary instead of disk

write_uuid_map_dict(*uuid_map_fpath*, *visual_uuid_list*, *daids_hashid*)

uses in memory dictionary instead of disk

let the multi-indexer know about any big caches we've made multi-indexer. Also lets nnindexer know about other prebuilt indexers so it can attempt to just add points to them as to avoid a rebuild.

wbia.algo.hots.neighbor_index_cache.**background_flann_func**(*cachedir*, *daid_list*, *vecs_list*, *fgws_list*, *fxs_list*, *flann_params*, *cfgstr*, *uuid_map_fpath*, *daids_hashid*, *visual_uuid_list*, *min_reindex_thresh*)

FIXME: Duplicate code

wbia.algo.hots.neighbor_index_cache.**build_nnindex_cfgstr**(*qreq_*, *daid_list*)

builds a string that uniquely identified an indexer built with parameters from the input query requested and indexing descriptor from the input annotation ids

Parameters

- **qreq** (QueryRequest) – query request object with hyper-parameters
- **daid_list** (*list*) –

Returns nnindex_cfgstr

Return type str

CommandLine: python -m wbia.algo.hots.neighbor_index_cache –test-build_nnindex_cfgstr

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index_cache import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(db='testdb1')
>>> daid_list = ibs.get_valid_aids(species=wbia.const.TEST_SPECIES.ZEB_PLAIN)
>>> qreq_ = ibs.new_query_request(daid_list, daid_list, cfgdict=dict(fg_on=False))
>>> nnindex_cfgstr = build_nnindex_cfgstr(qreq_, daid_list)
>>> result = str(nnindex_cfgstr)
>>> print(result)
```

_VUUIDS((6)ylydksaqdigdecdd)_FLANN(8_kdtrees)_FeatureWeight(detector=cnn,sz256,thresh=20,ksz=20,enabled=False)_Fea

_VUUIDS((6)ylydksaqdigdecdd)_FLANN(8_kdtrees)_FEATWEIGHT(OFF)_FEAT(**hesaff+sift_**)_CHIP(sz450)

wbia.algo.hots.neighbor_index_cache.**can_request_background_nnindexer**()

wbia.algo.hots.neighbor_index_cache.**check_background_process**()
checks to see if the process has finished and then writes the uuid map to disk

wbia.algo.hots.neighbor_index_cache.**clear_memcache**()

wbia.algo.hots.neighbor_index_cache.**clear_uuid_cache**(*qreq_*)

CommandLine: python -m wbia.algo.hots.neighbor_index_cache –test-clear_uuid_cache

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index_cache import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_(defaultdb='testdb1', p='default:fg_on=True')
>>> fgws_list = clear_uuid_cache(qreq_)
>>> result = str(fgws_list)
>>> print(result)
```

wbia.algo.hots.neighbor_index_cache.**get_data_cfgstr**(*ibs*, *daid_list*)
part 2 data hash id

wbia.algo.hots.neighbor_index_cache.**get_nnindexer_uuid_map_fpath**(*qreq_*)

CommandLine: python -m wbia.algo.hots.neighbor_index_cache get_nnindexer_uuid_map_fpath

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index_cache import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_(defaultdb='testdb1', p='default:fgw_thresh=.3')
>>> uuid_map_fpath = get_nnindexer_uuid_map_fpath(qreq_)
>>> result = str(ut.path_ndir_split(uuid_map_fpath, 3))
>>> print(result)
```

.../_wbia_cache/flann/uuid_map_mzwwsbjisbkdxorl.cPkl .../_wbia_cache/flann/uuid_map_FLANN(8_kdtrees_fgwthrsh=0.3)_
.../_wbia_cache/flann/uuid_map_FLANN(8_kdtrees)_Feat(hesaff+sift)_Chip(sz700,width).cPkl
.../_wbia_cache/flann/uuid_map_FLANN(8_kdtrees)_FEAT(**hesaff+sift_**)_CHIP(sz450).cPkl

wbia.algo.hots.neighbor_index_cache.**group_daids_by_cached_nnindexer**(*qreq_*, *daid_list*, *min_reindex_thresh*, *max_covers=None*)

CommandLine: python -m wbia.algo.hots.neighbor_index_cache –test-group_daids_by_cached_nnindexer

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index_cache import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> ZEB_PLAIN = wbia.const.TEST_SPECIES.ZEB_PLAIN
>>> daid_list = ibs.get_valid_aids(species=ZEB_PLAIN)
>>> qreq_ = ibs.new_query_request(daid_list, daid_list)
>>> # Set the params a bit lower
>>> max_covers = None
>>> qreq_.qparams.min_reindex_thresh = 1
>>> min_reindex_thresh = qreq_.qparams.min_reindex_thresh
>>> # STEP 0: CLEAR THE CACHE
>>> clear_uuid_cache(qreq_)
>>> # STEP 1: ASSERT EMPTY INDEX
>>> daid_list = sorted(ibs.get_valid_aids(species=ZEB_PLAIN))[0:3]
>>> uncovered_aids, covered_aids_list = group_daids_by_cached_nnindexer(
...     qreq_, daid_list, min_reindex_thresh, max_covers)
>>> result1 = uncovered_aids, covered_aids_list
>>> ut.assert_eq(result1, ([1, 2, 3], []), 'pre request')
>>> # TEST 2: SHOULD MAKE 123 COVERED
>>> nnindexer = request_memcached_wbia_nnindexer(qreq_, daid_list)
>>> uncovered_aids, covered_aids_list = group_daids_by_cached_nnindexer(
...     qreq_, daid_list, min_reindex_thresh, max_covers)
>>> result2 = uncovered_aids, covered_aids_list
>>> ut.assert_eq(result2, ([1, 2, 3], []), 'post request')
```

wbia.algo.hots.neighbor_index_cache.**new_neighbor_index**(*daid_list*, *vecs_list*, *fgws_list*, *fxs_list*, *flann_params*, *cachedir*, *cfgstr*, *force_rebuild=False*, *verbose=True*, *memtrack=None*, *prog_hook=None*)

constructs neighbor index independent of wbia

Parameters

- **daid_list** (*list*) –
- **vecs_list** (*list*) –
- **fgws_list** (*list*) –
- **flann_params** (*dict*) –
- **flann_cachedir** (*None*) –
- **nnindex_cfgstr** (*str*) –
- **use_memcache** (*bool*) –

Returns nnindexer

CommandLine: python -m wbia.algo.hots.neighbor_index_cache –test-new_neighbor_index

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index_cache import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_(defaultdb='testdb1', a='default:species=zebra_
↪plains', p='default:fgw_thresh=.999')
>>> daid_list = qreq_.daids
>>> nnindex_cfgstr = build_nnindex_cfgstr(qreq_, daid_list)
>>> ut.exec_funckw(new_neighbor_index, globals())
>>> cfgstr = nnindex_cfgstr
>>> cachedir     = qreq_.ibs.get_flann_cachedir()
>>> flann_params = qreq_.qparams.flann_params
>>> # Get annot descriptors to index
>>> vecs_list, fgws_list, fxs_list = get_support_data(qreq_, daid_list)
>>> nnindexer = new_neighbor_index(daid_list, vecs_list, fgws_list, fxs_list,␣
↪flann_params, cachedir, cfgstr, verbose=True)
>>> result = ('nnindexer.ax2_aid = %s' % (str(nnindexer.ax2_aid),))
>>> print(result)
nnindexer.ax2_aid = [1 2 3 4 5 6]
```

wbia.algo.hots.neighbor_index_cache.**print_uuid_cache**(*qreq_*)

CommandLine: python -m wbia.algo.hots.neighbor_index_cache –test-print_uuid_cache

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index_cache import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_(defaultdb='PZ_Master0', p='default:fg_on=False')
>>> print_uuid_cache(qreq_)
>>> result = str(nnindexer)
>>> print(result)
```

wbia.algo.hots.neighbor_index_cache.**request_augmented_wbia_nnindexer**(*qreq_*, *daid_list*, *verbose=True*, *use_memcache=True*, *force_rebuild=False*, *memtrack=None*)

DO NOT USE. THIS FUNCTION CAN CURRENTLY CAUSE A SEGFAULT

tries to give you an indexer for the requested daids using the least amount of computation possible. By loading and adding to a partially build nnindex if possible and if that fails fallbs back to request_memcache.

Parameters

- **qreq** (QueryRequest) – query request object with hyper-parameters
- **daid_list** (*list*) –

Returns nnindex_cfgstr

Return type str

CommandLine: python -m wbia.algo.hots.neighbor_index_cache –test-request_augmented_wbia_nnindexer

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index_cache import *  # NOQA
>>> import wbia
>>> # build test data
>>> ZEB_PLAIN = wbia.const.TEST_SPECIES.ZEB_PLAIN
>>> ibs = wbia.opendb('testdb1')
>>> use_memcache, max_covers, verbose = True, None, True
>>> daid_list = sorted(ibs.get_valid_aids(species=ZEB_PLAIN))[0:6]
>>> qreq_ = ibs.new_query_request(daid_list, daid_list)
>>> qreq_.qparams.min_reindex_thresh = 1
>>> min_reindex_thresh = qreq_.qparams.min_reindex_thresh
>>> # CLEAR CACHE for clean test
>>> clear_uuid_cache(qreq_)
>>> # LOAD 3 AIDS INTO CACHE
>>> aid_list = sorted(ibs.get_valid_aids(species=ZEB_PLAIN))[0:3]
>>> # Should fallback
>>> nnindexer = request_augmented_wbia_nnindexer(qreq_, aid_list)
>>> # assert the fallback
>>> uncovered_aids, covered_aids_list = group_daids_by_cached_nnindexer(
...     qreq_, daid_list, min_reindex_thresh, max_covers)
>>> result2 = uncovered_aids, covered_aids_list
>>> ut.assert_eq(result2, ([4, 5, 6], [[1, 2, 3]]), 'pre augment')
>>> # Should augment
>>> nnindexer = request_augmented_wbia_nnindexer(qreq_, daid_list)
>>> uncovered_aids, covered_aids_list = group_daids_by_cached_nnindexer(
...     qreq_, daid_list, min_reindex_thresh, max_covers)
>>> result3 = uncovered_aids, covered_aids_list
>>> ut.assert_eq(result3, ([], [[1, 2, 3, 4, 5, 6]]), 'post augment')
>>> # Should fallback
>>> nnindexer2 = request_augmented_wbia_nnindexer(qreq_, daid_list)
>>> assert nnindexer is nnindexer2
```

wbia.algo.hots.neighbor_index_cache.**request_background_nnindexer**(*qreq_*, *daid_list*)

FIXME: Duplicate code

Parameters

- **qreq** (QueryRequest) – query request object with hyper-parameters
- **daid_list** (*list*) –

CommandLine: python -m wbia.algo.hots.neighbor_index_cache –test-request_background_nnindexer

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index_cache import *  # NOQA
```

(continues on next page)

(continued from previous page)

```
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> daid_list = ibs.get_valid_aids(species=wbia.const.TEST_SPECIES.ZEB_PLAIN)
>>> qreq_ = ibs.new_query_request(daid_list, daid_list)
>>> # execute function
>>> request_background_nnindexer(qreq_, daid_list)
>>> # verify results
>>> result = str(False)
>>> print(result)
```

wbia.algo.hots.neighbor_index_cache.**request_diskcached_wbia_nnindexer**(*qreq_*, *daid_list*, *nnindex_cfgstr=None*, *verbose=True*, *force_rebuild=False*, *memtrack=None*, *prog_hook=None*)

builds new NeighborIndexer which will try to use a disk cached flann if available

Parameters

- **qreq** (QueryRequest) – query request object with hyper-parameters
- **daid_list** (*list*) –
- **nnindex_cfgstr** –
- **verbose** (*bool*) –

Returns nnindexer

Return type NeighborIndexer

CommandLine: python -m wbia.algo.hots.neighbor_index_cache –test-request_diskcached_wbia_nnindexer

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index_cache import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> daid_list = ibs.get_valid_aids(species=wbia.const.TEST_SPECIES.ZEB_PLAIN)
>>> qreq_ = ibs.new_query_request(daid_list, daid_list)
>>> nnindex_cfgstr = build_nnindex_cfgstr(qreq_, daid_list)
>>> verbose = True
>>> # execute function
>>> nnindexer = request_diskcached_wbia_nnindexer(qreq_, daid_list, nnindex_
↪cfgstr, verbose)
>>> # verify results
>>> result = str(nnindexer)
>>> print(result)
```

wbia.algo.hots.neighbor_index_cache.**request_memcached_wbia_nnindexer**(*qreq_*, *daid_list*, *use_memcache=True*, *verbose=True*, *veryverbose=False*, *force_rebuild=False*, *memtrack=None*, *prog_hook=None*)

FOR INTERNAL USE ONLY takes custom daid list. might not be the same as what is in **qreq_**

CommandLine: python -m wbia.algo.hots.neighbor_index_cache –test-request_memcached_wbia_nnindexer

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index_cache import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> qreq_.qparams.min_reindex_thresh = 3
>>> ZEB_PLAIN = wbia.const.TEST_SPECIES.ZEB_PLAIN
>>> daid_list = ibs.get_valid_aids(species=ZEB_PLAIN)[0:3]
>>> qreq_ = ibs.new_query_request(daid_list, daid_list)
>>> verbose = True
>>> use_memcache = True
>>> # execute function
>>> nnindexer = request_memcached_wbia_nnindexer(qreq_, daid_list, use_memcache)
>>> # verify results
>>> result = str(nnindexer)
>>> print(result)
```

wbia.algo.hots.neighbor_index_cache.**request_wbia_nnindexer**(*qreq_*, *verbose=True*, ***kwargs*)

CALLED BY QUERYREQUST::LOAD_INDEXER IBEIS interface into neighbor_index_cache

Parameters **qreq** (QueryRequest) – hyper-parameters

Returns nnindexer

Return type NeighborIndexer

CommandLine: python -m wbia.algo.hots.neighbor_index_cache request_wbia_nnindexer

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index_cache import *  # NOQA
>>> nnindexer, qreq_, ibs = testdata_nnindexer(None)
>>> nnindexer = request_wbia_nnindexer(qreq_)
```

wbia.algo.hots.neighbor_index_cache.**testdata_nnindexer**(*dbname='testdb1'*, *with_indexer=True*, *use_memcache=True*)

Ignore:

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index_cache import *  # NOQA
>>> nnindexer, qreq_, ibs = testdata_nnindexer('PZ_Master1')
>>> S = np.cov(nnindexer.idx2_vec.T)
>>> import wbia.plottool as pt
>>> pt.ensureqt()
>>> pt.plt.imshow(S)
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index_cache import *  # NOQA
>>> nnindexer, qreq_, ibs = testdata_nnindexer()
```

1.1.1.3.10 wbia.algo.hots.nn_weights module

wbia.algo.hots.nn_weights.**all_normalized_weights_test**()

CommandLine: python -m wbia.algo.hots.nn_weights –exec-all_normalized_weights_test

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.nn_weights import *  # NOQA
>>> all_normalized_weights_test()
```

wbia.algo.hots.nn_weights.**apply_normweight**(*normweight_fn*, *neighb_normk*, *neighb_idx*, *neighb_dist*, *Knorm*)

helper applies the normalized weight function to one query annotation

Parameters

- **normweight_fn** (*func*) – chosen weight function e.g. lnbnn
- **qaid** (*int*) – query annotation id
- **neighb_idx** (*ndarray[int32_t, ndims=2]*) – mapping from query feature index to db neighbor index
- **neighb_dist** (*ndarray*) – mapping from query feature index to dist
- **Knorm** (*int*) –
- **qreq** (QueryRequest) – query request object with hyper-parameters

Returns neighb_normweight

Return type ndarray

CommandLine: python -m wbia.algo.hots.nn_weights –test-apply_normweight

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.nn_weights import *  # NOQA
>>> from wbia.algo.hots import nn_weights
>>> #cfgdict = {'K':10, 'Knorm': 10, 'normalizer_rule': 'name',
>>> #            'dim_size': 450, 'resize_dim': 'area'}
>>> #tup = plh.testdata_pre_weight_neighbors(cfgdict=cfgdict)
>>> qreq_, args = plh.testdata_pre('weight_neighbors', defaultdb='testdb1',
>>>                                p=['default:K=10,Knorm=10,normalizer_rule=name,
↪dim_size=450,resize_dim=area'])
>>> nns_list, nnvalid0_list = args
>>> qaid = qreq_.qaids[0]
>>> Knorm = qreq_.qparams.Knorm
>>> normweight_fn = lnbnn_fn
>>> normalizer_rule  = qreq_.qparams.normalizer_rule
>>> (neighb_idx, neighb_dist) = nns_list[0]
>>> neighb_normk = get_normk(qreq_, qaid, neighb_idx, Knorm, normalizer_rule)
>>> neighb_normweight = nn_weights.apply_normweight(
>>>   normweight_fn, neighb_normk, neighb_idx, neighb_dist, Knorm)
>>> ut.assert_inbounds(neighb_normweight.sum(), 600, 950)
```

wbia.algo.hots.nn_weights.**bar_l2_fn**(*vdist*, *ndist*)

The feature weight is (1 - the euclidian distance between the features). The normalizers are unused.

(not really a normaalized function)

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.nn_weights import *  # NOQA
>>> vdist, ndist = testdata_vn_dists()
>>> out = bar_l2_fn(vdist, ndist)
>>> result = ut.hz_str('barl2  = ', ut.repr2(out, precision=2))
>>> print(result)
barl2  = np.array([[1.  , 0.6 , 0.41],
                   [0.83, 0.7 , 0.49],
                   [0.87, 0.58, 0.27],
                   [0.88, 0.63, 0.46],
                   [0.82, 0.53, 0.5 ]])
```

wbia.algo.hots.nn_weights.**const_match_weighter**(*nns_list*, *nnvalid0_list*, *qreq_*)

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots.nn_weights import *  # NOQA
>>> #tup = plh.testdata_pre_weight_neighbors('PZ_MTEST')
>>> qreq_, args = plh.testdata_pre('weight_neighbors', defaultdb='PZ_MTEST')
>>> nns_list, nnvalid0_list = args
>>> ibs, qreq_, nns_list, nnvalid0_list = tup
>>> constvote_weight_list = const_match_weighter(nns_list, nnvalid0_list, qreq_)
>>> result = ('constvote_weight_list = %s' % (str(constvote_weight_list),))
>>> print(result)
```

wbia.algo.hots.nn_weights.**fg_match_weighter**(*nns_list*, *nnvalid0_list*, *qreq_*)
foreground feature match weighting

CommandLine: python -m wbia.algo.hots.nn_weights –exec-fg_match_weighter

Example

```
>>> # xdoctest: +REQUIRES(module:wbia_cnn)
>>> from wbia.algo.hots.nn_weights import *  # NOQA
>>> #tup = plh.testdata_pre_weight_neighbors('PZ_MTEST')
>>> #ibs, qreq_, nns_list, nnvalid0_list = tup
>>> qreq_, args = plh.testdata_pre('weight_neighbors', defaultdb='PZ_MTEST')
>>> nns_list, nnvalid0_list = args
>>> print(ut.repr2(qreq_.qparams.__dict__, sorted_=True))
>>> assert qreq_.qparams.fg_on == True, 'bug setting custom params fg_on'
>>> fgvotes_list = fg_match_weighter(nns_list, nnvalid0_list, qreq_)
>>> print('fgvotes_list = %r' % (fgvotes_list,))
```

wbia.algo.hots.nn_weights.**get_name_normalizers**(*qaid*, *qreq_*, *Knorm*, *neighb_idx*)
helper normalizers for ‘name’ normalizer_rule

Parameters

- **qaid** (*int*) – query annotation id
- **qreq** (*wbia.QueryRequest*) – hyper-parameters
- **Knorm** (*int*) –
- **neighb_idx** (*ndarray*) –

Returns neighb_normk

Return type ndarray

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.nn_weights import *  # NOQA
>>> from wbia.algo.hots import nn_weights
>>> #cfgdict = {'K':10, 'Knorm': 10, 'normalizer_rule': 'name'}
>>> #tup = plh.testdata_pre_weight_neighbors(cfgdict=cfgdict)
>>> qreq_, args = plh.testdata_pre('weight_neighbors', defaultdb='testdb1',
>>>                                p=['default:K=10,Knorm=10,normalizer_rule=name
↪'])
>>> nns_list, nnvalid0_list = args
>>> Knorm = qreq_.qparams.Knorm
>>> (neighb_idx, neighb_dist) = nns_list[0]
>>> qaid = qreq_.qaids[0]
>>> neighb_normk = get_name_normalizers(qaid, qreq_, Knorm, neighb_idx)
```

wbia.algo.hots.nn_weights.**get_normk**(*qreq_*, *qaid*, *neighb_idx*, *Knorm*, *normalizer_rule*)
Get positions of the LNBNN/ratio tests normalizers

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.nn_weights import *  # NOQA
>>> cfgdict = {'K':10, 'Knorm': 10, 'normalizer_rule': 'name',
>>>            'dim_size': 450, 'resize_dim': 'area'}
>>> #tup = plh.testdata_pre_weight_neighbors(cfgdict=cfgdict)
>>> qreq_, args = plh.testdata_pre('weight_neighbors', defaultdb='testdb1',
>>>                                p=['default:K=10,Knorm=10,normalizer_rule=name,
↪dim_size=450,resize_dim=area'])
>>> nns_list, nnvalid0_list = args
>>> (neighb_idx, neighb_dist) = nns_list[0]
>>> qaid = qreq_.qaids[0]
>>> K = qreq_.qparams.K
>>> Knorm = qreq_.qparams.Knorm
>>> neighb_normk1 = get_normk(qreq_, qaid, neighb_idx, Knorm, 'last')
>>> neighb_normk2 = get_normk(qreq_, qaid, neighb_idx, Knorm, 'name')
>>> assert np.all(neighb_normk1 == Knorm + K)
>>> assert np.all(neighb_normk2 <= Knorm + K) and np.all(neighb_normk2 > K)
```

wbia.algo.hots.nn_weights.**gravity_match_weighter**(*nns_list*, *nnvalid0_list*, *qreq_*)

wbia.algo.hots.nn_weights.**lnbnn_fn**(*vdist*, *ndist*)

Locale Naive Bayes Nearest Neighbor weighting

References

http://www.cs.ubc.ca/~lowe/papers/12mccannCVPR.pdf http://www.cs.ubc.ca/~sanchom/local-naive-bayes-nearest-neighbor

Sympy:

```
>>> import sympy
>>> #https://github.com/sympy/sympy/pull/10247
>>> from sympy import log
>>> from sympy.stats import P, E, variance, Die, Normal, FiniteRV
>>> C, Cbar = sympy.symbols('C Cbar')
>>> d_i = Die(sympy.symbols('di'), 6)
>>> log(P(di, C) / P(di, Cbar))
>>> #
>>> PdiC, PdiCbar = sympy.symbols('PdiC, PdiCbar')
>>> oddsC = log(PdiC / PdiCbar)
>>> sympy.simplify(oddsC)
>>> import vtool as vt
>>> vt.check_expr_eq(oddsC, log(PdiC) - log(PdiCbar))
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.nn_weights import *  # NOQA
>>> vdist, ndist = testdata_vn_dists()
>>> out = lnbnn_fn(vdist, ndist)
>>> result = ut.hz_str('lnbnn  = ', ut.repr2(out, precision=2))
>>> print(result)
lnbnn  = np.array([[0.62, 0.22, 0.03],
```

(continues on next page)

(continued from previous page)

```
                    [0.35, 0.22, 0.01],
                    [0.87, 0.58, 0.27],
                    [0.67, 0.42, 0.25],
                    [0.59, 0.3 , 0.27]])
```

wbia.algo.hots.nn_weights.**logger = <Logger wbia (INFO)>**

qfx2_ no longer applies due to fgw_thresh. Need to change names in this file

TODO: replace testdata_pre_weight_neighbors with

```
>>> qreq_, args = plh.testdata_pre('weight_neighbors', defaultdb='testdb1',
>>>                                a=['default:qindex=0:1,dindex=0:5,
↪hackerrors=False'],
>>>                                p=['default:codename=vsmany,bar_l2_on=True,fg_
↪on=False'], verbose=True)
```

Type FIXME

wbia.algo.hots.nn_weights.**loglnbnn_fn**(*vdist*, *ndist*)

Ignore: import vtool as vt vt.check_expr_eq('log(d) - log(n)', 'log(d / n)') # True vt.check_expr_eq('log(d) / log(n)', 'log(d - n)')

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.nn_weights import *  # NOQA
>>> vdist, ndist = testdata_vn_dists()
>>> out = loglnbnn_fn(vdist, ndist)
>>> result = ut.hz_str('loglnbnn  = ', ut.repr2(out, precision=2))
>>> print(result)
loglnbnn  = np.array([[0.48, 0.2 , 0.03],
                      [0.3 , 0.2 , 0.01],
                      [0.63, 0.46, 0.24],
                      [0.51, 0.35, 0.22],
                      [0.46, 0.26, 0.24]])
```

wbia.algo.hots.nn_weights.**logratio_fn**(*vdist*, *ndist*)

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.nn_weights import *  # NOQA
>>> vdist, ndist = testdata_vn_dists()
>>> out = normonly_fn(vdist, ndist)
>>> result = ut.repr2(out)
>>> print(result)
np.array([[0.62, 0.62, 0.62],
          [0.52, 0.52, 0.52],
          [1.  , 1.  , 1.  ],
          [0.79, 0.79, 0.79],
          [0.77, 0.77, 0.77]])
```

wbia.algo.hots.nn_weights.**mark_name_valid_normalizers**(*qnid*, *neighb_topnid*, *neighb_normnid*)

Helper func that allows matches only to the first result for a name

Each query feature finds its K matches and Kn normalizing matches. These are the candidates from which it can choose a set of matches and a single normalizer.

A normalizer is marked as invalid if it belongs to a name that was also in its feature's candidate matching set.

Parameters

- **neighb_topnid** (*ndarray*) – marks the names a feature matches
- **neighb_normnid** (*ndarray*) – marks the names of the feature normalizers
- **qnid** (*int*) – query name id

Returns neighb_selnorm - index of the selected normalizer for each query feature

CommandLine: python -m wbia.algo.hots.nn_weights –exec-mark_name_valid_normalizers

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.nn_weights import *  # NOQA
>>> qnid = 1
>>> neighb_topnid = np.array([[1, 1, 1, 1, 1],
...                           [1, 2, 1, 1, 1],
...                           [1, 2, 2, 3, 1],
...                           [5, 8, 9, 8, 8],
...                           [5, 8, 9, 8, 8],
...                           [6, 6, 9, 6, 8],
...                           [5, 8, 6, 6, 6],
...                           [1, 2, 8, 6, 6]], dtype=np.int32)
>>> neighb_normnid = np.array([[ 1, 1, 1],
...                            [ 2, 3, 1],
...                            [ 2, 3, 1],
...                            [ 6, 6, 6],
...                            [ 6, 6, 8],
...                            [ 2, 6, 6],
...                            [ 6, 6, 1],
...                            [ 4, 4, 9]], dtype=np.int32)
>>> neighb_selnorm = mark_name_valid_normalizers(qnid, neighb_topnid, neighb_
↪normnid)
>>> K = len(neighb_topnid.T)
>>> Knorm = len(neighb_normnid.T)
>>> neighb_normk_ = neighb_selnorm + (Knorm)  # convert form negative to pos␣
↪indexes
>>> result = str(neighb_normk_)
>>> print(result)
[2 1 2 0 0 0 2 0]
```

Ignore: logger.info(ut.doctest_repr(neighb_normnid, 'neighb_normnid', verbose=False)) logger.info(ut.doctest_repr(neighb_topnid, 'neighb_topnid', verbose=False))

wbia.algo.hots.nn_weights.**nn_normalized_weight**(*normweight_fn*, *nns_list*, *nnvalid0_list*, *qreq_*)

Generic function to weight nearest neighbors

ratio, lnbnn, and other nearest neighbor based functions use this

Parameters

- **normweight_fn** (*func*) – chosen weight function e.g. lnbnn
- **nns_list** (*dict*) – query descriptor nearest neighbors and distances.
- **nnvalid0_list** (*list*) – list of neighbors preflagged as valid
- **qreq** (QueryRequest) – hyper-parameters

Returns weights_list

Return type list

CommandLine: python -m wbia.algo.hots.nn_weights nn_normalized_weight –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.nn_weights import *  # NOQA
>>> #tup = plh.testdata_pre_weight_neighbors('PZ_MTEST')
>>> #ibs, qreq_, nns_list, nnvalid0_list = tup
>>> qreq_, args = plh.testdata_pre('weight_neighbors',
>>>                                defaultdb='PZ_MTEST')
>>> nns_list, nnvalid0_list = args
>>> normweight_fn = lnbnn_fn
>>> weights_list1, normk_list1 = nn_normalized_weight(
>>>     normweight_fn, nns_list, nnvalid0_list, qreq_)
>>> weights1 = weights_list1[0]
>>> nn_normonly_weight = NN_WEIGHT_FUNC_DICT['lnbnn']
>>> weights_list2, normk_list2 = nn_normonly_weight(nns_list, nnvalid0_list, qreq_
↪)
>>> weights2 = weights_list2[0]
>>> assert np.all(weights1 == weights2)
>>> ut.assert_inbounds(weights1.sum(), 100, 510)
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.nn_weights import *  # NOQA
>>> #tup = plh.testdata_pre_weight_neighbors('PZ_MTEST')
>>> qreq_, args = plh.testdata_pre('weight_neighbors',
>>>                                defaultdb='PZ_MTEST')
>>> nns_list, nnvalid0_list = args
>>> normweight_fn = ratio_fn
>>> weights_list1, normk_list1 = nn_normalized_weight(normweight_fn, nns_list,␣
↪nnvalid0_list, qreq_)
>>> weights1 = weights_list1[0]
>>> nn_normonly_weight = NN_WEIGHT_FUNC_DICT['ratio']
>>> weights_list2, normk_list2 = nn_normonly_weight(nns_list, nnvalid0_list, qreq_
↪)
>>> weights2 = weights_list2[0]
>>> assert np.all(weights1 == weights2)
>>> ut.assert_inbounds(weights1.sum(), 1500, 4500)
```

wbia.algo.hots.nn_weights.**normonly_fn**(*vdist*, *ndist*)

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.nn_weights import *  # NOQA
>>> vdist, ndist = testdata_vn_dists()
>>> out = normonly_fn(vdist, ndist)
>>> result = ut.repr2(out)
>>> print(result)
np.array([[0.62, 0.62, 0.62],
          [0.52, 0.52, 0.52],
          [1.  , 1.  , 1.  ],
          [0.79, 0.79, 0.79],
          [0.77, 0.77, 0.77]])
```

wbia.algo.hots.nn_weights.**ratio_fn**(*vdist*, *ndist*)

Parameters

- **vdist** (*ndarray*) – voting array
- **ndist** (*ndarray*) – normalizing array

Returns out

Return type ndarray

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.nn_weights import *  # NOQA
>>> vdist, ndist = testdata_vn_dists()
>>> out = ratio_fn(vdist, ndist)
>>> result = ut.hz_str('ratio = ', ut.repr2(out, precision=2))
>>> print(result)
ratio = np.array([[0.  , 0.65, 0.95],
                  [0.33, 0.58, 0.98],
                  [0.13, 0.42, 0.73],
                  [0.15, 0.47, 0.68],
                  [0.23, 0.61, 0.65]])
```

wbia.algo.hots.nn_weights.**testdata_vn_dists**(*nfeats=5*, *K=3*)

Test voting and normalizing distances

Returns (vdist, ndist) - test voting distances and normalizer distances

Return type tuple

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.nn_weights import *  # NOQA
>>> vdist, ndist = testdata_vn_dists()
>>> result = (ut.hz_str('vdist = ', ut.repr2(vdist))) + '\n'
>>> print(result + (ut.hz_str('ndist = ', ut.repr2(ndist))))
vdist = np.array([[0.  , 0.4 , 0.59],
                  [0.17, 0.3 , 0.51],
                  [0.13, 0.42, 0.73],
```

(continues on next page)

(continued from previous page)

```
                  [0.12, 0.37, 0.54],
                  [0.18, 0.47, 0.5 ]])
ndist = np.array([[0.62],
                  [0.52],
                  [1.  ],
                  [0.79],
                  [0.77]])
```

1.1.1.3.11 wbia.algo.hots.old_chip_match module

class wbia.algo.hots.old_chip_match.**AlignedListDictProxy**(*key2_idx*, *key_list*, *val_list*)

Bases: utool.util_dev.DictLike_old

simulates a dict when using parallel lists the point of this class is that when there are many instances of this class, then key2_idx can be shared between them. Ideally this class wont be used and will disappear when the parallel lists are being used properly.

DEPCIRATE AlignedListDictProxy's defaultdict behavior is weird

iteritems()

iterkeys()

itervalues()

pop(*key*)

1.1.1.3.12 wbia.algo.hots.pipeline module

Hotspotter pipeline module

Module Notation and Concepts: PREFIXES: qaid2_XXX - prefix mapping query chip index to qfx2_XXX - prefix mapping query chip feature index to

- nns - a (qfx2_idx, qfx2_dist) tuple
- idx - the index into the nnindexers descriptors
- qfx - query feature index wrt the query chip
- dfx - query feature index wrt the database chip
- dist - the distance to a corresponding feature
- fm - a list of feature match pairs / correspondences (qfx, dfx)
- fsv - a score vector of a corresponding feature
- valid - a valid bit for a corresponding feature

PIPELINE_VARS: nns_list - maping from query chip index to nns

- qfx2_idx - ranked list of query feature indexes to database feature indexes
- qfx2_dist - ranked list of query feature indexes to database feature indexes
- **qaid2_norm_weight - mapping from qaid to (qfx2_normweight, qfx2_selnorm)** = qaid2_nnfiltagg[qaid]

CommandLine: To see the ouput of a complete pipeline run use

Set to whichever database you like python main.py –db PZ_MTEST –setdb python main.py –db NAUT_test –setdb python main.py –db testdb1 –setdb

Then run whichever configuration you like python main.py –query 1 –yes –noqcache -t default:codename=vsmany python main.py –query 1 –yes –noqcache -t default:codename=vsmany_nsum

---

Todo:

- Don't preload the nn-indexer in case the nearest neighbors have already

been computed?

---

class wbia.algo.hots.pipeline.**Neighbors**(*qaid*, *idxs*, *dists*, *qfxs*)

Bases: utool.util_dev.NiceRepr

neighb_dists

neighb_idxs

num_query_feats

qaid

qfx_list

wbia.algo.hots.pipeline.**WeightRet_**

alias of wbia.algo.hots.pipeline.weight_ret

wbia.algo.hots.pipeline.**baseline_neighbor_filter**(*qreq_*, *nns_list*, *impossible_daids_list*, *verbose=False*)

Removes matches to self, the same image, or the same name.

CommandLine: python -m wbia.algo.hots.pipeline –test-baseline_neighbor_filter

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.pipeline import *   # NOQA
>>> qreq_, args = plh.testdata_pre(
>>>     'baseline_neighbor_filter', defaultdb='testdb1',
>>>     qaid_override=[1, 2, 3, 4],
>>>     daid_override=list(range(1, 11)),
>>>     p=['default:QRH=False,requery=False,can_match_samename=False'],
>>>     verbose=True)
>>> nns_list, impossible_daids_list = args
>>> nnvalid0_list = baseline_neighbor_filter(qreq_, nns_list,
>>>                                          impossible_daids_list)
>>> ut.assert_eq(len(nnvalid0_list), len(qreq_.qaids))
>>> assert not np.any(nnvalid0_list[0][:, 0]), (
...    'first col should be all invalid because of self match')
>>> assert not np.all(nnvalid0_list[0][:, 1]), (
...    'second col should have some good matches')
>>> ut.assert_inbounds(nnvalid0_list[0].sum(), 1000, 10000)
```

wbia.algo.hots.pipeline.**build_chipmatches**(*qreq_*, *nns_list*, *nnvalid0_list*, *filtkey_list*, *filtweights_list*, *filtvalids_list*, *filtnormks_list*, *verbose=False*)

pipeline step 4 - builds sparse chipmatches

Takes the dense feature matches from query feature to (what could be any) database features and builds sparse matching pairs for each annotation to annotation match.

CommandLine: python -m wbia build_chipmatches python -m wbia build_chipmatches:0 –show python -m wbia build_chipmatches:1 –show python -m wbia build_chipmatches:2 –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.pipeline import *  # NOQA
>>> qreq_, args = plh.testdata_pre(
>>>     'build_chipmatches', p=['default:codename=vsmany'])
>>> (nns_list, nnvalid0_list, filtkey_list, filtweights_list,
>>> filtvalids_list, filtnormks_list) = args
>>> verbose = True
>>> cm_list = build_chipmatches(qreq_, *args, verbose=verbose)
>>> # verify results
>>> [cm.assert_self(qreq_) for cm in cm_list]
>>> cm = cm_list[0]
>>> fm = cm.fm_list[cm.daid2_idx[2]]
>>> num_matches = len(fm)
>>> print('vsmany num_matches = %r' % num_matches)
>>> ut.assert_inbounds(num_matches, 500, 2000, 'vsmany nmatches out of bounds')
>>> ut.quit_if_noshow()
>>> cm.score_annot_csum(qreq_)
>>> cm_list[0].ishow_single_annotmatch(qreq_)
>>> ut.show_if_requested()
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots.pipeline import *  # NOQA
>>> # Test to make sure filtering by feature weights works
>>> qreq_, args = plh.testdata_pre(
>>>     'build_chipmatches',
>>>     p=['default:codename=vsmany,fgw_thresh=.9'])
>>> (nns_list, nnvalid0_list, filtkey_list, filtweights_list,
>>>  filtvalids_list, filtnormks_list) = args
>>> verbose = True
>>> cm_list = build_chipmatches(qreq_, *args, verbose=verbose)
>>> # verify results
>>> [cm.assert_self(qreq_) for cm in cm_list]
>>> cm = cm_list[0]
>>> fm = cm.fm_list[cm.daid2_idx[2]]
>>> num_matches = len(fm)
>>> print('num_matches = %r' % num_matches)
>>> ut.assert_inbounds(num_matches, 100, 410, 'vsmany nmatches out of bounds')
>>> ut.quit_if_noshow()
>>> cm.score_annot_csum(qreq_)
>>> cm_list[0].ishow_single_annotmatch(qreq_)
>>> ut.show_if_requested()
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.pipeline import *  # NOQA
>>> qreq_, args = plh.testdata_pre(
>>>     'build_chipmatches', p=['default:requery=True'], a='default')
>>> (nns_list, nnvalid0_list, filtkey_list, filtweights_list,
>>> filtvalids_list, filtnormks_list) = args
>>> verbose = True
>>> cm_list = build_chipmatches(qreq_, *args, verbose=verbose)
>>> # verify results
>>> [cm.assert_self(qreq_) for cm in cm_list]
>>> scoring.score_chipmatch_list(qreq_, cm_list, 'csum')
>>> cm = cm_list[0]
>>> for cm in cm_list:
>>>     # should be positive for LNBNN
>>>     assert np.all(cm.score_list[np.isfinite(cm.score_list)] >= 0)
```

wbia.algo.hots.pipeline.**build_impossible_daids_list**(*qreq_*, *verbose=False*)

Parameters **qreq** (QueryRequest) – query request object with hyper-parameters

CommandLine: python -m wbia.algo.hots.pipeline –test-build_impossible_daids_list

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.pipeline import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_(
>>>     defaultdb='testdb1',
>>>     a='default:species=zebra_plains,qhackerrors=True',
>>>     p='default:use_k_padding=True,can_match_sameimg=False,can_match_
↪samename=False')
>>> impossible_daids_list, Kpad_list = build_impossible_daids_list(qreq_)
>>> impossible_daids_list = [x.tolist() for x in impossible_daids_list]
>>> vals = ut.dict_subset(locals(), ['impossible_daids_list', 'Kpad_list'])
>>> result = ut.repr2(vals, nl=1, explicit=True, nobr=True, strvals=True)
>>> print(result)
>>> assert np.all(qreq_.qaids == [1, 4, 5, 6])
>>> assert np.all(qreq_.daids == [1, 2, 3, 4, 5, 6])
...
impossible_daids_list=[[1], [4], [5, 6], [5, 6]],
Kpad_list=[1, 1, 2, 2],
```

wbia.algo.hots.pipeline.**cachemiss_nn_compute_fn**(*flags_list*, *qreq_*, *Kpad_list*, *impossible_daids_list*, *K*, *Knorm*, *requery*, *verbose*)

Logic for computing neighbors if there is a cache miss

```
>>> flags_list = [True] * len(Kpad_list)
>>> flags_list = [True, False, True]
```

wbia.algo.hots.pipeline.**compute_matching_dlen_extent**(*qreq_*, *fm_list*, *kpts_list*)

helper for spatial verification, computes the squared diagonal length of matching chips

CommandLine: python -m wbia.algo.hots.pipeline –test-compute_matching_dlen_extent

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.pipeline import *  # NOQA
>>> ibs, qreq_, cm_list = plh.testdata_pre_sver('PZ_MTEST')
>>> verbose = True
>>> cm = cm_list[0]
>>> cm.set_cannonical_annot_score(cm.get_num_matches_list())
>>> cm.sortself()
>>> fm_list = cm.fm_list
>>> kpts_list = qreq_.get_qreq_dannot_kpts(cm.daid_list.tolist())
>>> topx2_dlen_sqrd = compute_matching_dlen_extent(qreq_, fm_list, kpts_list)
>>> ut.assert_inbounds(np.sqrt(topx2_dlen_sqrd)[0:5], 600, 1500)
```

wbia.algo.hots.pipeline.**get_sparse_matchinfo_nonagg**(*qreq_*, *nns*, *neighb_valid0*, *neighb_score_list*, *neighb_valid_list*, *neighb_normk_list*, *Knorm*, *fsv_col_lbls*)

builds sparse iterator that generates feature match pairs, scores, and ranks

Returns

vmt a tuple of corresponding lists. Each item in the list corresponds to a daid, dfx, scorevec, rank, norm_aid, norm_fx...

Return type ValidMatchTup_

CommandLine: python -m wbia.algo.hots.pipeline –test-get_sparse_matchinfo_nonagg –show python -m wbia.algo.hots.pipeline –test-get_sparse_matchinfo_nonagg:1 –show

utprof.py -m wbia.algo.hots.pipeline –test-get_sparse_matchinfo_nonagg

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.pipeline import *  # NOQA
>>> verbose = True
>>> qreq_, qaid, daid, args = plh.testdata_sparse_matchinfo_nonagg(
>>>     defaultdb='PZ_MTEST', p=['default:Knorm=3,normalizer_rule=name,const_
↪on=True,ratio_thresh=.2,sqrd_dist_on=True'])
>>> nns, neighb_valid0, neighb_score_list, neighb_valid_list, neighb_normk_list,␣
↪Knorm, fsv_col_lbls = args
>>> cm = get_sparse_matchinfo_nonagg(qreq_, *args)
>>> qannot = qreq_.ibs.annots([qaid], config=qreq_.qparams)
>>> dannot = qreq_.ibs.annots(cm.daid_list, config=qreq_.qparams)
>>> cm.assert_self(verbose=False)
>>> ut.quit_if_noshow()
>>> cm.score_annot_csum(qreq_)
>>> cm.show_single_annotmatch(qreq_)
>>> ut.show_if_requested()
```

wbia.algo.hots.pipeline.**nearest_neighbor_cacheid2**(*qreq_*, *Kpad_list*)

Returns a hacky cacheid for neighbor configs. DEPRICATE: This will be replaced by dtool caching

Parameters

- **qreq** (QueryRequest) – query request object with hyper-parameters

- **Kpad_list** (*list*) –

Returns (nn_mid_cacheid_list, nn_cachedir)

Return type tuple

CommandLine: python -m wbia.algo.hots.pipeline –exec-nearest_neighbor_cacheid2 python -m wbia.algo.hots.pipeline –exec-nearest_neighbor_cacheid2 –superstrict

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.pipeline import *  # NOQA
>>> import wbia
>>> verbose = True
>>> cfgdict = dict(K=4, Knorm=1, checks=800, use_k_padding=False)
>>> # test 1
>>> p = 'default' + ut.get_cfg_lbl(cfgdict)
>>> qreq_ = wbia.testdata_qreq_(
>>>     defaultdb='testdb1', p=[p], qaid_override=[1, 2],
>>>     daid_override=[1, 2, 3, 4, 5])
>>> locals_ = plh.testrun_pipeline_upto(qreq_, 'nearest_neighbors')
>>> Kpad_list, = ut.dict_take(locals_, ['Kpad_list'])
>>> tup = nearest_neighbor_cacheid2(qreq_, Kpad_list)
>>> (nn_cachedir, nn_mid_cacheid_list) = tup
>>> result1 = 'nn_mid_cacheid_list1 = ' + ut.repr2(nn_mid_cacheid_list, nl=1)
>>> # test 2
>>> cfgdict2 = dict(K=2, Knorm=3, use_k_padding=True)
>>> p2 = 'default' + ut.get_cfg_lbl(cfgdict)
>>> ibs = qreq_.ibs
>>> qreq_ = wbia.testdata_qreq_(defaultdb='testdb1', p=[p2], qaid_override=[1, 2],
 ↪ daid_override=[1, 2, 3, 4, 5])
>>> locals_ = plh.testrun_pipeline_upto(qreq_, 'nearest_neighbors')
>>> Kpad_list, = ut.dict_take(locals_, ['Kpad_list'])
>>> tup = nearest_neighbor_cacheid2(qreq_, Kpad_list)
>>> (nn_cachedir, nn_mid_cacheid_list) = tup
>>> result2 = 'nn_mid_cacheid_list2 = ' + ut.repr2(nn_mid_cacheid_list, nl=1)
>>> result = result1 + '\n' + result2
>>> print(result)
nn_mid_cacheid_list1 = [
    'nnobj_8687dcb6-1f1f-fdd3-8b72-8f36f9f41905_DVUUIDS((5)oavtblnlrtocnrpm)_
 ↪NN(single,cks800)_Chip(sz700,maxwh)_Feat(hesaff+sift)_FLANN(8_kdtrees)_truek6',
    'nnobj_a2aef668-20c1-1897-d8f3-09a47a73f26a_DVUUIDS((5)oavtblnlrtocnrpm)_
 ↪NN(single,cks800)_Chip(sz700,maxwh)_Feat(hesaff+sift)_FLANN(8_kdtrees)_truek6',
]
nn_mid_cacheid_list2 = [
    'nnobj_8687dcb6-1f1f-fdd3-8b72-8f36f9f41905_DVUUIDS((5)oavtblnlrtocnrpm)_
 ↪NN(single,cks800)_Chip(sz700,maxwh)_Feat(hesaff+sift)_FLANN(8_kdtrees)_truek6',
    'nnobj_a2aef668-20c1-1897-d8f3-09a47a73f26a_DVUUIDS((5)oavtblnlrtocnrpm)_
 ↪NN(single,cks800)_Chip(sz700,maxwh)_Feat(hesaff+sift)_FLANN(8_kdtrees)_truek6',
]
```

wbia.algo.hots.pipeline.**nearest_neighbors**(*qreq_*, *Kpad_list*, *impossible_daids_list=None*, *verbose=False*)

Plain Nearest Neighbors Tries to load nearest neighbors from a cache instead of recomputing them.

CommandLine: python -m wbia.algo.hots.pipeline –test-nearest_neighbors python -m wbia.algo.hots.pipeline

–test-nearest_neighbors –db PZ_MTEST –qaids=1:100 utprof.py -m wbia.algo.hots.pipeline –test-nearest_neighbors –db PZ_MTEST –qaids=1:100

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.pipeline import *  # NOQA
>>> import wbia
>>> verbose = True
>>> qreq_ = wbia.testdata_qreq_(defaultdb='testdb1', qaid_override=[1])
>>> locals_ = plh.testrun_pipeline_upto(qreq_, 'nearest_neighbors')
>>> Kpad_list, impossible_daids_list = ut.dict_take(
>>>     locals_, ['Kpad_list', 'impossible_daids_list'])
>>> nns_list = nearest_neighbors(qreq_, Kpad_list, impossible_daids_list,
>>>                              verbose=verbose)
>>> qaid = qreq_.internal_qaids[0]
>>> nn = nns_list[0]
>>> (qfx2_idx, qfx2_dist) = nn
>>> num_neighbors = Kpad_list[0] + qreq_.qparams.K + qreq_.qparams.Knorm
>>> # Assert nns tuple is valid
>>> ut.assert_eq(qfx2_idx.shape, qfx2_dist.shape)
>>> ut.assert_eq(qfx2_idx.shape[1], num_neighbors)
>>> ut.assert_inbounds(qfx2_idx.shape[0], 1000, 3000)
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.pipeline import *  # NOQA
>>> import wbia
>>> verbose = True
>>> qreq_ = wbia.testdata_qreq_(defaultdb='testdb1', qaid_override=[1])
>>> locals_ = plh.testrun_pipeline_upto(qreq_, 'nearest_neighbors')
>>> Kpad_list, impossible_daids_list = ut.dict_take(
>>>     locals_, ['Kpad_list', 'impossible_daids_list'])
>>> nns_list = nearest_neighbors(qreq_, Kpad_list, impossible_daids_list,
>>>                              verbose=verbose)
>>> qaid = qreq_.internal_qaids[0]
>>> nn = nns_list[0]
>>> (qfx2_idx, qfx2_dist) = nn
>>> num_neighbors = Kpad_list[0] + qreq_.qparams.K + qreq_.qparams.Knorm
>>> # Assert nns tuple is valid
>>> ut.assert_eq(qfx2_idx.shape, qfx2_dist.shape)
>>> ut.assert_eq(qfx2_idx.shape[1], num_neighbors)
>>> ut.assert_inbounds(qfx2_idx.shape[0], 1000, 3000)
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.pipeline import *  # NOQA
>>> import wbia
>>> verbose = True
>>> custom_nid_lookup = {a: a for a in range(14)}
>>> qreq1_ = wbia.testdata_qreq_(
```

(continues on next page)

(continued from previous page)

```
>>>     defaultdb='testdb1', t=['default:K=2,requery=True,can_match_samename=False
↪'],
>>>     daid_override=[2, 3, 4, 5, 6, 7, 8],
>>>     qaid_override=[2, 5, 1], custom_nid_lookup=custom_nid_lookup)
>>> locals_ = plh.testrun_pipeline_upto(qreq1_, 'nearest_neighbors')
>>> Kpad_list, impossible_daids_list = ut.dict_take(
>>>    locals_, ['Kpad_list', 'impossible_daids_list'])
>>> nns_list1 = nearest_neighbors(qreq1_, Kpad_list, impossible_daids_list,
>>>                               verbose=verbose)
>>> nn1 = nns_list1[0]
>>> nnvalid0_list1 = baseline_neighbor_filter(qreq1_, nns_list1,
>>>                                           impossible_daids_list)
>>> assert np.all(nnvalid0_list1[0]), (
>>>  'requery should never produce impossible results')
>>> # Compare versus not using requery
>>> qreq2_ = wbia.testdata_qreq_(
>>>     defaultdb='testdb1', t=['default:K=2,requery=False'],
>>>     daid_override=[1, 2, 3, 4, 5, 6, 7, 8],
>>>     qaid_override=[2, 5, 1])
>>> locals_ = plh.testrun_pipeline_upto(qreq2_, 'nearest_neighbors')
>>> Kpad_list, impossible_daids_list = ut.dict_take(
>>>    locals_, ['Kpad_list', 'impossible_daids_list'])
>>> nns_list2 = nearest_neighbors(qreq2_, Kpad_list, impossible_daids_list,
>>>                               verbose=verbose)
>>> nn2 = nns_list2[0]
>>> nn1.neighb_dists
>>> nn2.neighb_dists
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.pipeline import *  # NOQA
>>> import wbia
>>> verbose = True
>>> qreq1_ = wbia.testdata_qreq_(
>>>     defaultdb='testdb1', t=['default:K=5,requery=True,can_match_samename=False
↪'],
>>>     daid_override=[2, 3, 4, 5, 6, 7, 8],
>>>     qaid_override=[2, 5, 1])
>>> locals_ = plh.testrun_pipeline_upto(qreq1_, 'nearest_neighbors')
>>> Kpad_list, impossible_daids_list = ut.dict_take(
>>>    locals_, ['Kpad_list', 'impossible_daids_list'])
>>> nns_list1 = nearest_neighbors(qreq1_, Kpad_list, impossible_daids_list,
>>>                               verbose=verbose)
>>> nn1 = nns_list1[0]
>>> nnvalid0_list1 = baseline_neighbor_filter(qreq1_, nns_list1,
>>>                                           impossible_daids_list)
>>> assert np.all(nnvalid0_list1[0]), 'should always be valid'
```

wbia.algo.hots.pipeline.**request_wbia_query_L0**(*ibs*, *qreq_*, *verbose=False*)

Driver logic of query pipeline

Note: Make sure _pipeline_helpres.testrun_pipeline_upto reflects what happens in this function.

Parameters

- **ibs** (*wbia.IBEISController*) – IBEIS database object to be queried. technically this object already lives inside of **qreq_**.
- **qreq** (*wbia.QueryRequest*) – hyper-parameters. use ibs.new_query_request to create one

Returns cm_list containing wbia.ChipMatch objects

Return type list

CommandLine: python -m wbia.algo.hots.pipeline –test-request_wbia_query_L0:0 –show python -m wbia.algo.hots.pipeline –test-request_wbia_query_L0:1 –show

python -m wbia.algo.hots.pipeline –test-request_wbia_query_L0:0 –db testdb1 –qaid 325 python -m wbia.algo.hots.pipeline –test-request_wbia_query_L0:0 –db testdb3 –qaid 325 # background match python -m wbia.algo.hots.pipeline –test-request_wbia_query_L0:0 –db NNP_Master3 –qaid 12838

python -m wbia.algo.hots.pipeline –test-request_wbia_query_L0:0 python -m wbia.algo.hots.pipeline –test-request_wbia_query_L0:0 –db PZ_MTEST -a timectrl:qindex=0:256 python -m wbia.algo.hots.pipeline –test-request_wbia_query_L0:0 –db PZ_Master1 -a timectrl:qindex=0:256 utprof.py -m wbia.algo.hots.pipeline –test-request_wbia_query_L0:0 –db PZ_Master1 -a timectrl:qindex=0:256

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.pipeline import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.init.main_helpers.testdata_qreq_(a=['default:qindex=0:2,
↪dindex=0:10'])
>>> ibs = qreq_.ibs
>>> print(qreq_.qparams.query_cfgstr)
>>> verbose = True
>>> cm_list = request_wbia_query_L0(ibs, qreq_, verbose=verbose)
>>> cm = cm_list[0]
>>> ut.quit_if_noshow()
>>> cm.ishow_analysis(qreq_, fnum=0, make_figtitle=True)
>>> ut.show_if_requested()
```

wbia.algo.hots.pipeline.**spatial_verification**(*qreq_*, *cm_list_FILT*, *verbose=False*)
pipeline step 5 - spatially verify feature matches

Returns cm_listSVER - new list of spatially verified chipmatches

Return type list

CommandLine: python -m wbia.algo.hots.pipeline –test-spatial_verification –show python -m wbia.algo.hots.pipeline –test-spatial_verification –show –qaid 1 python -m wbia.algo.hots.pipeline –test-spatial_verification:0

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.pipeline import *  # NOQA
>>> ibs, qreq_, cm_list = plh.testdata_pre_sver('PZ_MTEST', qaid_list=[18])
>>> scoring.score_chipmatch_list(qreq_, cm_list, qreq_.qparams.prescore_method)
↪# HACK
>>> cm = cm_list[0]
>>> top_nids = cm.get_top_nids(6)
>>> verbose = True
>>> cm_list_SVER = spatial_verification(qreq_, cm_list)
>>> # Test Results
>>> cmSV = cm_list_SVER[0]
>>> scoring.score_chipmatch_list(qreq_, cm_list_SVER, qreq_.qparams.score_method)␣
↪ # HACK
>>> top_nids_SV = cmSV.get_top_nids(6)
>>> cm.print_csv(sort=True)
>>> cmSV.print_csv(sort=False)
>>> gt_daids  = np.intersect1d(cm.get_groundtruth_daids(), cmSV.get_groundtruth_
↪daids())
>>> fm_list   = cm.get_annot_fm(gt_daids)
>>> fmSV_list = cmSV.get_annot_fm(gt_daids)
>>> maplen = lambda list_: np.array(list(map(len, list_)))
>>> assert len(gt_daids) > 0, 'ground truth did not survive'
>>> ut.assert_lessthan(maplen(fmSV_list), maplen(fm_list)), 'feature matches were␣
↪not filtered'
>>> ut.quit_if_noshow()
>>> cmSV.show_daids_matches(qreq_, gt_daids)
>>> import wbia.plottool as pt
>>> #homog_tup = (refined_inliers, H)
>>> #aff_tup = (aff_inliers, Aff)
>>> #pt.draw_sv.show_sv(rchip1, rchip2, kpts1, kpts2, fm, aff_tup=aff_tup, homog_
↪tup=homog_tup, refine_method=refine_method)
>>> ut.show_if_requested()
```

wbia.algo.hots.pipeline.**sver_single_chipmatch**(*qreq_*, *cm*, *verbose=False*)

Spatially verifies a shortlist of a single chipmatch

TODO: move to chip match?

loops over a shortlist of results for a specific query annotation

Parameters

- **qreq** (QueryRequest) – query request object with hyper-parameters
- **cm** (ChipMatch) –

Returns cmSV

Return type wbia.ChipMatch

CommandLine:

python -m wbia draw_rank_cmc –db PZ_Master1 –show -t best:refine_method=[homog,affine,cv2-homog,cv2-ransac-homog,cv2-lmeds-homog] -a timectrlhard —acfginfo –veryverbtd

python -m wbia draw_rank_cmc –db PZ_Master1 –show -t best:refine_method=[homog,cv2-lmeds-homog],full_homog_checks=[True,False] -a timectrlhard —acfginfo –veryverbtd

python -m wbia sver_single_chipmatch –show -t default:full_homog_checks=True -a default –qaid 18

python -m wbia sver_single_chipmatch –show -t default:refine_method=affine -a default –qaid 18

python -m wbia sver_single_chipmatch –show -t default:refine_method=cv2-homog -a default –qaid 18

python -m wbia sver_single_chipmatch –show -t default:refine_method=cv2-homog,full_homog_checks=True -a default –qaid 18

python -m wbia sver_single_chipmatch –show -t default:refine_method=cv2-homog,full_homog_checks=False -a default –qaid 18

python -m wbia sver_single_chipmatch –show -t default:refine_method=cv2-lmeds-homog,full_homog_checks=False -a default –qaid 18

python -m wbia sver_single_chipmatch –show -t default:refine_method=cv2-ransac-homog,full_homog_checks=False -a default –qaid 18

python -m wbia sver_single_chipmatch –show -t default:full_homog_checks=False -a default –qaid 18

python -m wbia sver_single_chipmatch –show –qaid=18 –y=0 python -m wbia sver_single_chipmatch –show –qaid=18 –y=1

Example

```
>>> # DISABLE_DOCTEST
>>> # Visualization
>>> from wbia.algo.hots.pipeline import *  # NOQA
>>> qreq_, args = plh.testdata_pre('spatial_verification', defaultdb='PZ_MTEST')
↪#, qaid_list=[18])
>>> cm_list = args.cm_list_FILT
>>> ibs = qreq_.ibs
>>> cm = cm_list[0]
>>> scoring.score_chipmatch_list(qreq_, cm_list, qreq_.qparams.prescore_method)
↪# HACK
>>> #locals_ = ut.exec_func_src(sver_single_chipmatch, key_list=['svtup_list'],␣
↪sentinal='# <SENTINAL>')
>>> #svtup_list1, = locals_
>>> verbose = True
>>> source = ut.get_func_sourcecode(sver_single_chipmatch, stripdef=True, strip_
↪docstr=True)
>>> source = ut.replace_between_tags(source, '', '# <SENTINAL>', '# </SENTINAL>')
>>> globals_ = globals().copy()
>>> exec(source, globals_)
>>> svtup_list = globals_['svtup_list']
>>> gt_daids = cm.get_groundtruth_daids()
>>> x = ut.get_argval('--y', type_=int, default=0)
>>> #print('x = %r' % (x,))
>>> #daid = daids[x % len(daids)]
>>> notnone_list = ut.not_list(ut.flag_None_items(svtup_list))
>>> valid_idxs = np.where(notnone_list)
>>> valid_daids = cm.daid_list[valid_idxs]
>>> assert len(valid_daids) > 0, 'cannot spatially verify'
>>> valid_gt_daids = np.intersect1d(gt_daids, valid_daids)
>>> #assert len(valid_gt_daids) == 0, 'no sver groundtruth'
>>> daid = valid_gt_daids[x] if len(valid_gt_daids) > 0 else valid_daids[x]
>>> idx = cm.daid2_idx[daid]
>>> svtup = svtup_list[idx]
>>> assert svtup is not None, 'SV TUP IS NONE'
>>> refined_inliers, refined_errors, H = svtup[0:3]
>>> aff_inliers, aff_errors, Aff = svtup[3:6]
```

(continues on next page)

(continued from previous page)

```
>>> homog_tup = (refined_inliers, H)
>>> aff_tup = (aff_inliers, Aff)
>>> fm = cm.fm_list[idx]
>>> aid1 = cm.qaid
>>> aid2 = daid
>>> rchip1, = ibs.get_annot_chips([aid1], config2_=qreq_.extern_query_config2)
>>> kpts1,  = ibs.get_annot_kpts([aid1], config2_=qreq_.extern_query_config2)
>>> rchip2, = ibs.get_annot_chips([aid2], config2_=qreq_.extern_data_config2)
>>> kpts2, = ibs.get_annot_kpts([aid2], config2_=qreq_.extern_data_config2)
>>> import wbia.plottool as pt
>>> import matplotlib as mpl
>>> from wbia.scripts.thesis import TMP_RC
>>> mpl.rcParams.update(TMP_RC)
>>> show_aff = not ut.get_argflag('--noaff')
>>> refine_method = qreq_.qparams.refine_method if not ut.get_argflag('--
↪norefinelbl') else ''
>>> pt.draw_sv.show_sv(rchip1, rchip2, kpts1, kpts2, fm, aff_tup=aff_tup,
>>>                    homog_tup=homog_tup, show_aff=show_aff,
>>>                    refine_method=refine_method)
>>> ut.show_if_requested()
```

wbia.algo.hots.pipeline.**weight_neighbors**(*qreq_*, *nns_list*, *nnvalid0_list*, *verbose=False*)

pipeline step 3 - assigns weights to feature matches based on the active filter list

CommandLine: python -m wbia.algo.hots.pipeline –test-weight_neighbors python -m wbia.algo.hots.pipeline –test-weight_neighbors:0 –verbose –verbtd –ainfo –nocache –veryverbose python -m wbia.algo.hots.pipeline –test-weight_neighbors:0 –show python -m wbia.algo.hots.pipeline –test-weight_neighbors:1 –show

python -m wbia.algo.hots.pipeline –test-weight_neighbors:0 –show -t default:lnbnn_normer=lnbnn_fg_0.9__featscore,lnbnn_norm_thresh=.9

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.pipeline import *  # NOQA
>>> qreq_, args = plh.testdata_pre(
>>>     'weight_neighbors', defaultdb='testdb1',
>>>     a=['default:qindex=0:3,dindex=0:5,hackerrors=False'],
>>>     p=['default:codename=vsmany,bar_l2_on=True,fg_on=False'], verbose=True)
>>> nns_list, nnvalid0_list = args
>>> verbose = True
>>> weight_ret = weight_neighbors(qreq_, nns_list, nnvalid0_list, verbose)
>>> filtkey_list, filtweights_list, filtvalids_list, filtnormks_list = weight_ret
>>> import wbia.plottool as pt
>>> verbose = True
>>> cm_list = build_chipmatches(
>>>     qreq_, nns_list, nnvalid0_list, filtkey_list, filtweights_list,
>>>     filtvalids_list, filtnormks_list, verbose=verbose)
>>> ut.quit_if_noshow()
>>> cm = cm_list[0]
>>> cm.score_name_nsum(qreq_)
>>> cm.ishow_analysis(qreq_)
>>> ut.show_if_requested()
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.pipeline import *  # NOQA
>>> qreq_, args = plh.testdata_pre(
>>>     'weight_neighbors', defaultdb='testdb1',
>>>     a=['default:qindex=0:3,dindex=0:5,hackerrors=False'],
>>>     p=['default:codename=vsmany,bar_l2_on=True,fg_on=False'], verbose=True)
>>> nns_list, nnvalid0_list = args
>>> verbose = True
>>> weight_ret = weight_neighbors(qreq_, nns_list, nnvalid0_list, verbose)
>>> filtkey_list, filtweights_list, filtvalids_list, filtnormks_list = weight_ret
>>> nInternAids = len(qreq_.get_internal_qaids())
>>> nFiltKeys = len(filtkey_list)
>>> filtweight_depth = ut.depth_profile(filtweights_list)
>>> filtvalid_depth = ut.depth_profile(filtvalids_list)
>>> ut.assert_eq(nInternAids, len(filtweights_list))
>>> ut.assert_eq(nInternAids, len(filtvalids_list))
>>> ut.assert_eq(ut.get_list_column(filtweight_depth, 0), [nFiltKeys] *␣
↪nInternAids)
>>> ut.assert_eq(filtvalid_depth, (nInternAids, nFiltKeys))
>>> ut.assert_eq(filtvalids_list, [[None, None], [None, None], [None, None]])
>>> ut.assert_eq(filtkey_list, [hstypes.FiltKeys.LNBNN, hstypes.FiltKeys.BARL2])
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> verbose = True
>>> cm_list = build_chipmatches(
>>>     qreq_, nns_list, nnvalid0_list, filtkey_list, filtweights_list,
>>>     filtvalids_list, filtnormks_list, verbose=verbose)
>>> cm = cm_list[0]
>>> cm.score_name_nsum(qreq_)
>>> cm.ishow_analysis(qreq_)
>>> ut.show_if_requested()
```

1.1.1.3.13 wbia.algo.hots.query_params module

class wbia.algo.hots.query_params.**QueryParams**(*query_cfg=None*, *cfgdict=None*)

Bases: collections.abc.Mapping

copy()

get(*key*, **d*)

get a paramater value by string

get_postsver_filtkey_list()

HACK: gets columns of fsv post spatial verification. This will eventually be incorporated into cmtup_old instead and will not be dependant on specifically where you are in the pipeline

hack_lnbnn_config_trail()

1.1.1.3.14 wbia.algo.hots.query_request module

Todo: replace with dtool Rename to IdentifyRequest

python -m utool.util_inspect check_module_usage –pat=”query_request.py”

class wbia.algo.hots.query_request.**QueryRequest**
Bases: utool.util_dev.NiceRepr

Request object for pipline parameter run

daids
These are the users daids in vsone mode

dannots
external query annotation objects

dnids
save dnids in **qreq_** state

Type TODO

ensure_chips(*verbose=True*, *num_retries=1*)
ensure chips are computed (used in expt, not used in pipeline)

Parameters

- **verbose** (*bool*) – verbosity flag(default = True)
- **num_retries** (*int*) – (default = 0)

CommandLine: python -m wbia.algo.hots.query_request –test-ensure_chips

Example

```
>>> # ENABLE_DOCTEST
>>> # Delete chips (accidentally), then try to run a query
>>> from wbia.algo.hots.query_request import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> daids = ibs.get_valid_aids()[0:3]
>>> qaids = ibs.get_valid_aids()[0:6]
>>> qreq_ = ibs.new_query_request(qaids, daids)
>>> verbose = True
>>> num_retries = 1
>>> qchip_fpaths = ibs.get_annot_chip_fpath(qaids, config2_=qreq_.extern_
↪query_config2)
>>> dchip_fpaths = ibs.get_annot_chip_fpath(daids, config2_=qreq_.extern_data_
↪config2)
>>> ut.remove_file_list(qchip_fpaths)
>>> ut.remove_file_list(dchip_fpaths)
>>> result = qreq_.ensure_chips(verbose, num_retries)
>>> print(result)
```

ensure_features(*verbose=True*, *prog_hook=None*)
ensure features are computed :param verbose: verbosity flag(default = True) :type verbose: bool

CommandLine: python -m wbia.algo.hots.query_request –test-ensure_features

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.query_request import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> daids = ibs.get_valid_aids()[0:2]
>>> qaids = ibs.get_valid_aids()[0:3]
>>> qreq_ = ibs.new_query_request(qaids, daids)
>>> ibs.delete_annot_feats(qaids,  config2_=qreq_.extern_query_config2)  #␣
↪Remove the chips
>>> ut.remove_file_list(ibs.get_annot_chip_fpath(qaids, config2_=qreq_.extern_
↪query_config2))
>>> verbose = True
>>> result = qreq_.ensure_features(verbose)
>>> print(result)
```

ensure_featweights(*verbose=True*)
ensure feature weights are computed

execute(*qaids=None*, *prog_hook=None*, *use_cache=None*, *use_supercache=None*, *invalidate_supercache=None*)
Runs the hotspotter pipeline and returns chip match objects.

CommandLine: python -m wbia.algo.hots.query_request execute –show

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +SKIP
>>> from wbia.algo.hots.query_request import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_()
>>> cm_list = qreq_.execute()
>>> ut.quit_if_noshow()
>>> cm = cm_list[0]
>>> cm.ishow_analysis(qreq_)
>>> ut.show_if_requested()
```

extern_data_config2

extern_query_config2

get_big_cacher()

get_bigcache_info()

get_cfgstr(*with_input=False*, *with_data=True*, *with_pipe=True*, *hash_pipe=False*)
main cfgstring used to identify the ‘querytype’ FIXME: name params + data

Todo: rename query_cfgstr to pipe_cfgstr or pipeline_cfgstr EVERYWHERE

Parameters **with_input** (*bool*) – (default = False)

Returns cfgstr

Return type str

CommandLine: python -m wbia.algo.hots.query_request –exec-get_cfgstr

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.query_request import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_(defaultdb='testdb1',
>>>                             p='default:fgw_thresh=.3',
>>>                             a='default:species=zebra_plains')
>>> with_input = True
>>> cfgstr = qreq_.get_cfgstr(with_input)
>>> result = ('cfgstr = %s' % (str(cfgstr),))
>>> print(result)
```

get_chipmatch_fpaths(*qaid_list*, *super_qres_cache=False*)
Generates chipmatch paths for input query annotation rowids

get_data_hashid()

CommandLine: python -m wbia.algo.hots.query_request –exec-QueryRequest.get_query_hashid –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots.query_request import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_()
>>> data_hashid = qreq_.get_data_hashid()
>>> result = ('data_hashid = %s' % (ut.repr2(data_hashid),))
>>> print(result)
```

get_external_query_groundtruth(*qaids*)
gets groundtruth that are accessible via this query

get_full_cfgstr()
main cfgstring used to identify the ‘querytype’ FIXME: name params + data + query

get_infostr()

get_internal_daids()

get_internal_data_config2()

get_internal_qaids()

get_internal_query_config2()

get_pipe_cfgstr()
FIXME: name params only

get_pipe_hashid()

get_qreq_annot_nids(*aids*)

get_qreq_annot_visual_uuids(*aids*)

get_qreq_dannot_fgweights(*daids*)

get_qreq_dannot_kpts(*daids*)

get_qreq_pcc_hashes(*aids*)
aids = [1, 2, 3]

get_qreq_pcc_hashid(*aids*, *prefix=''*, *with_nids=False*)
Gets a combined hash of a group of aids. Each aid hash represents itself in the context of the query database.

only considers grouping of database names

CommandLine: python -m wbia.algo.hots.query_request –test-get_qreq_pcc_hashid:0

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.query_request import *  # NOQA
>>> import wbia
>>> p = ['default:K=2,nameknn=True']
>>> defaultdb = 'testdb1'
>>> # Test that UUIDS change when you change the name lookup
>>> new_ = ut.partial(wbia.testdata_qreq_, defaultdb=defaultdb, p=p,
>>>                   verbose=False)
>>> # All diff names
>>> qreq1 = new_(daid_override=[2, 3, 5, 6],
>>>              qaid_override=[1, 2, 4],
>>>              custom_nid_lookup={a: a for a in range(14)})
>>> # All same names
>>> qreq2 = new_(daid_override=[2, 3, 5, 6],
>>>              qaid_override=[1, 2, 4],
>>>              custom_nid_lookup={a: 1 for a in range(14)})
>>> # Change the PCC, removing a query (data should NOT change)
>>> # because the thing being queried against is the same
>>> qreq3 = new_(daid_override=[2, 3, 5, 6],
>>>              qaid_override=[1, 2],
>>>              custom_nid_lookup={a: 1 for a in range(14)})
>>> # Now remove a database object (query SHOULD change)
>>> # because the results are different depending on
>>> # nameing of database (maybe they shouldnt change...)
>>> qreq4 = new_(daid_override=[2, 3, 6],
>>>              qaid_override=[1, 2, 4],
>>>              custom_nid_lookup={a: 1 for a in range(14)})
>>> print(qreq1.get_cfgstr(with_input=True, with_pipe=False))
>>> print(qreq2.get_cfgstr(with_input=True, with_pipe=False))
>>> print(qreq3.get_cfgstr(with_input=True, with_pipe=False))
>>> print(qreq4.get_cfgstr(with_input=True, with_pipe=False))
>>> assert qreq3.get_data_hashid() == qreq2.get_data_hashid()
>>> assert qreq1.get_data_hashid() != qreq2.get_data_hashid()
```

get_qreq_pcc_uuids(*aids*)
TODO. dont use uuids anymore. they are slow

get_qreq_qannot_fgweights(*qaids*)

get_qreq_qannot_kpts(*qaids*)

get_qresdir()

get_query_hashid()

CommandLine: python -m wbia.algo.hots.query_request –exec-QueryRequest.get_query_hashid –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots.query_request import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_()
>>> query_hashid = qreq_.get_query_hashid()
>>> result = ('query_hashid = %s' % (ut.repr2(query_hashid),))
>>> print(result)
```

get_shortinfo_cfgstr()

get_shortinfo_parts()
Rename to get_nice_parts

get_unique_species()

internal_dannots

internal_qannots

lazy_load(*verbose=True*)
Performs preloading of all data needed for a batch of queries

lazy_preload(*prog_hook=None*, *verbose=True*)
feature weights and normalizers should be loaded before vsone queries are issued. They do not depened only on qparams

Load non-query specific normalizers / weights

load_indexer(*verbose=True*, *force=False*, *prog_hook=None*)

classmethod new_query_request(*qaid_list*, *daid_list*, *qparams*, *qresdir*, *ibs*, *query_config2_*, *data_config2_*, *_indexer_request_params*, *custom_nid_lookup=None*)
old way of calling new

Parameters

- **qaid_list** (*list*) –
- **daid_list** (*list*) –
- **qparams** (QueryParams) – query hyper-parameters
- **qresdir** (*str*) –
- **ibs** (*wbia.IBEISController*) – image analysis api
- **_indexer_request_params** (*dict*) –

Returns wbia.QueryRequest

qaids
These are the users qaids in vsone mode

qannots
internal query annotation objects

qnids
save qnids in **qreq_** state

Type TODO

rrr (*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

set_external_daids (*daid_list*)

set_external_qaid_mask (*masked_qaid_list*)

Parameters **qaid_list** (*list*) –

CommandLine: python -m wbia.algo.hots.query_request –test-set_external_qaid_mask

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.query_request import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(db='testdb1')
>>> qaid_list = [1, 2, 3, 4, 5]
>>> daid_list = [1, 2, 3, 4, 5]
>>> qreq_ = ibs.new_query_request(qaid_list, daid_list)
>>> masked_qaid_list = [2, 4, 5]
>>> qreq_.set_external_qaid_mask(masked_qaid_list)
>>> result = np.array_str(qreq_.qaids)
>>> print(result)
[1 3]
```

set_external_qaids (*qaid_list*)

set_internal_masked_daids (*masked_daid_list*)
used by the pipeline to execute a subset of the query request without modifying important state

set_internal_masked_qaids (*masked_qaid_list*)
used by the pipeline to execute a subset of the query request without modifying important state

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.query_request import *  # NOQA
>>> import utool as ut
>>> import wbia
>>> qaid_list = [1, 2, 3, 4]
>>> daid_list = [1, 2, 3, 4]
>>> qreq_ = wbia.testdata_qreq_(qaid_override=qaid_list, daid_override=daid_
↪list, p='default:sv_on=True')
>>> qaids = qreq_.get_internal_qaids()
>>> ut.assert_lists_eq(qaid_list, qaids)
>>> masked_qaid_list = [1, 2, 3,]
>>> qreq_.set_internal_masked_qaids(masked_qaid_list)
>>> new_internal_aids = qreq_.get_internal_qaids()
>>> ut.assert_lists_eq(new_internal_aids, [4])
```

shallowcopy (*qaids=None*)
Creates a copy of qreq with the same qparams object and a subset of the qx and dx objects. used to generate chunks of vsmany queries

CommandLine: python -m wbia.algo.hots.query_request QueryRequest.shallowcopy

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.query_request import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_(default_qaids=[1, 2])
>>> qreq2_ = qreq_.shallowcopy(qaids=1)
>>> assert qreq_.daids is qreq2_.daids, 'should be the same'
>>> assert len(qreq_.qaids) != len(qreq2_.qaids), 'should be diff'
>>> #assert qreq_.metadata is not qreq2_.metadata
```

wbia.algo.hots.query_request.**apply_species_with_detector_hack**(*ibs*, *cfgdict*, *qaids*, *daids*, *verbose=None*)

HACK turns of featweights if they cannot be applied

wbia.algo.hots.query_request.**cfg_deepcopy_test**()

TESTING FUNCTION

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.query_request import *  # NOQA
>>> result = cfg_deepcopy_test()
>>> print(result)
```

wbia.algo.hots.query_request.**new_wbia_query_request**(*ibs*, *qaid_list*, *daid_list*, *cfgdict=None*, *verbose=None*, *unique_species=None*, *use_memcache=True*, *query_cfg=None*, *custom_nid_lookup=None*)

wbia entry point to create a new query request object

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **qaid_list** (*list*) – query ids
- **daid_list** (*list*) – database ids
- **cfgdict** (*dict*) – pipeline dictionary config
- **query_cfg** (*dtool.Config*) – Pipeline Config Object
- **unique_species** (*None*) – (default = None)
- **use_memcache** (*bool*) – (default = True)
- **verbose** (*bool*) – verbosity flag(default = True)

Returns wbia.QueryRequest

CommandLine: python -m wbia.algo.hots.query_request –test-new_wbia_query_request:0 python -m wbia.algo.hots.query_request –test-new_wbia_query_request:1

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.query_request import *  # NOQA
>>> ibs, qaid_list, daid_list = testdata_newqreq('PZ_MTEST')
>>> unique_species = None
>>> verbose = ut.NOT_QUIET
>>> cfgdict = {'sv_on': False, 'fg_on': True}  # 'fw_detector': 'rf'}
>>> qreq_ = new_wbia_query_request(ibs, qaid_list, daid_list, cfgdict=cfgdict)
>>> print(qreq_.get_cfgstr())
>>> assert qreq_.qparams.sv_on is False, (
...     'qreq_.qparams.sv_on = %r ' % qreq_.qparams.sv_on)
>>> result = ibs.get_dbname() + qreq_.get_data_hashid()
>>> print(result)
PZ_MTEST_DPCC_UUIDS-a5-n2-vpkyggtpzbqbecuq
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.query_request import *  # NOQA
>>> ibs, qaid_list, daid_list = testdata_newqreq('NAUT_test')
>>> unique_species = None
>>> verbose = ut.NOT_QUIET
>>> cfgdict = {'sv_on': True, 'fg_on': True}
>>> qreq_ = new_wbia_query_request(ibs, qaid_list, daid_list, cfgdict=cfgdict)
>>> assert qreq_.query_config2_.featweight_enabled
>>> # Featweight should be off because there is no Naut detector
>>> print(qreq_.qparams.query_cfgstr)
>>> assert qreq_.qparams.sv_on is True, (
...     'qreq_.qparams.sv_on = %r ' % qreq_.qparams.sv_on)
>>> result = ibs.get_dbname() + qreq_.get_data_hashid()
>>> print(result)
NAUT_test_DPCC_UUIDS-a5-n3-rtuyggvzpczvmjcw
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.query_request import *  # NOQA
>>> ibs, qaid_list, daid_list = testdata_newqreq('PZ_MTEST')
>>> unique_species = None
>>> verbose = ut.NOT_QUIET
>>> cfgdict = {'sv_on': False, 'query_rotation_heuristic': True}
>>> qreq_ = new_wbia_query_request(ibs, qaid_list, daid_list, cfgdict=cfgdict)
>>> # Featweight should be off because there is no Naut detector
>>> print(qreq_.qparams.query_cfgstr)
>>> assert qreq_.qparams.sv_on is False, (
...     'qreq_.qparams.sv_on = %r ' % qreq_.qparams.sv_on)
>>> result = ibs.get_dbname() + qreq_.get_data_hashid()
>>> print(result)
PZ_MTEST_DPCC_UUIDS-a5-n2-vpkyggtpzbqbecuq
```

Ignore: # This is supposed to be the beginings of the code to transition the # pipeline configuration into the new minimal dict based structure that # supports different configs for query and database annotations. dcfg = qreq_.extern_data_config2 qcfg = qreq_.extern_query_config2 ut.dict_intersection(qcfg.__dict__,

dcfg.__dict__) from wbia.expt import cfghelpers cfg_list = [qcfg.__dict__, dcfg.__dict__] nonvaried_cfg, varied_cfg_list = ut.partition_varied_cfg_list(

cfg_list, recursive=True)

qvaried, dvaried = varied_cfg_list

wbia.algo.hots.query_request.**testdata_newqreq**(*defaultdb='testdb1'*)

Returns (wbia.IBEISController, list, list)

1.1.1.3.15 wbia.algo.hots.requery_knn module

class wbia.algo.hots.requery_knn.**FinalResults**(*shape*)
Bases: utool.util_dev.NiceRepr

assign(*index*, *idxs*, *dists*, *trueks*)

class wbia.algo.hots.requery_knn.**TempQuery**(*vecs*, *invalid_axs*, *get_neighbors*, *get_axs*)
Bases: utool.util_dev.NiceRepr

queries that are incomplete

compress_inplace(*flags*)

neighbors(*temp_K*)

class wbia.algo.hots.requery_knn.**TempResults**(*index*, *idxs*, *dists*, *validflags*)
Bases: utool.util_dev.NiceRepr

compress(*flags*)

done_flags(*num_neighbs*)

done_part(*num_neighbs*)

wbia.algo.hots.requery_knn.**in1d_shape**(*arr1*, *arr2*)

wbia.algo.hots.requery_knn.**requery_knn**(*get_neighbors*, *get_axs*, *qfx2_vec*, *num_neighbs*, *invalid_axs=[]*, *pad=2*, *limit=4*, *recover=True*)
Searches for *num_neighbs*, while ignoring certain matches. K is increassed until enough valid neighbors are found or a limit is reached.

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_(defaultdb='testdb1', a='default')
>>> qreq_.load_indexer()
>>> indexer = qreq_.indexer
>>> qannot = qreq_.internal_qannots[1]
>>> qfx2_vec = qannot.vecs
>>> ibs = qreq_.ibs
>>> qaid = qannot.aid
>>> impossible_aids = ibs.get_annot_groundtruth(qaid, noself=False)
>>> invalid_axs = np.array(ut.take(indexer.aid2_ax, impossible_aids))
>>> pad = 0
>>> limit = 1
>>> num_neighbs = 3
```

(continues on next page)

(continued from previous page)

```
>>> def get_neighbors(vecs, temp_K):
>>>     return indexer.flann.nn_index(vecs, temp_K, checks=indexer.checks,
>>>                                   cores=indexer.cores)
>>> get_axs = indexer.get_nn_axs
>>> res = requery_knn(
>>>     get_neighbors, get_axs, qfx2_vec, num_neighbs, invalid_axs, pad,
>>>     limit, recover=True)
>>> qfx2_idx, qfx2_dist = res
>>> assert np.all(np.diff(qfx2_dist, axis=1) >= 0)
```

Ignore:

```
>>> from wbia.algo.hots.neighbor_index import *  # NOQA
>>> from wbia.algo.hots.requery_knn import *  # NOQA
>>> max_k = 9
>>> n_pts = 5
>>> num_neighbs = 3
>>> temp_K = num_neighbs * 2
>>> #
>>> # Create dummy data
>>> rng = np.random.RandomState(0)
>>> tx2_idx_full = rng.randint(0, 10, size=(n_pts, max_k))
>>> tx2_idx_full[:, 0] = 0
>>> tx2_dist_full = np.meshgrid(np.arange(max_k), np.arange(n_pts))[0] / 10
>>> tx2_dist_full += (rng.rand(n_pts, max_k) * 10).astype(np.int) / 100
>>> qfx2_vec = np.arange(n_pts)[:, None]
>>> vecs = qfx2_vec
>>> #
>>> pad = 0
>>> limit = 1
>>> recover = True
>>> #
>>> invalid_axs = np.array([0, 1, 2, 5, 7, 9])
>>> get_axs = ut.identity
>>> #
>>> def get_neighbors(vecs, temp_K):
>>>     # simulates finding k nearest neighbors
>>>     idxs = tx2_idx_full[vecs.ravel(), 0:temp_K]
>>>     dists = tx2_dist_full[vecs.ravel(), 0:temp_K]
>>>     return idxs, dists
>>> #
>>> res = requery_knn(
>>>     get_neighbors, get_axs, qfx2_vec, num_neighbs, invalid_axs, pad,
>>>     limit, recover=True)
>>> qfx2_idx, qfx2_dist = res
```

1.1.1.3.16 wbia.algo.hots.scoring module

wbia.algo.hots.scoring.**get_name_shortlist_aids**(*daid_list*, *dnid_list*, *annot_score_list*, *name_score_list*, *nid2_nidx*, *nNameShortList*, *nAnnotPerName*)

CommandLine: python -m wbia.algo.hots.scoring –test-get_name_shortlist_aids

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.scoring import *  # NOQA
>>> daid_list        = np.array([11, 12, 13, 14, 15, 16, 17])
>>> dnid_list        = np.array([21, 21, 21, 22, 22, 23, 24])
>>> annot_score_list = np.array([ 6,  2,  3,  5,  6,  3,  2])
>>> name_score_list  = np.array([ 8,  9,  5,  4])
>>> nid2_nidx        = {21:0, 22:1, 23:2, 24:3}
>>> nNameShortList, nAnnotPerName = 3, 2
>>> args = (daid_list, dnid_list, annot_score_list, name_score_list,
...         nid2_nidx, nNameShortList, nAnnotPerName)
>>> top_daids = get_name_shortlist_aids(*args)
>>> result = str(top_daids)
>>> print(result)
[15, 14, 11, 13, 16]
```

wbia.algo.hots.scoring.**make_chipmatch_shortlists**(*qreq_*, *cm_list*, *nNameShortList*, *nAnnotPerName*, *score_method='nsum'*)

Makes shortlists for reranking

CommandLine: python -m wbia.algo.hots.scoring –test-make_chipmatch_shortlists –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.scoring import *  # NOQA
>>> ibs, qreq_, cm_list = plh.testdata_pre_sver('PZ_MTEST', qaid_list=[18])
>>> score_method    = 'nsum'
>>> nNameShortList  = 5
>>> nAnnotPerName   = 6
>>> # apply scores
>>> score_chipmatch_list(qreq_, cm_list, score_method)
>>> cm_input = cm_list[0]
>>> #assert cm_input.dnid_list.take(cm_input.argsort())[0] == cm_input.qnid
>>> cm_shortlist = make_chipmatch_shortlists(qreq_, cm_list, nNameShortList,␣
↪nAnnotPerName)
>>> cm_input.print_rawinfostr()
>>> cm = cm_shortlist[0]
>>> cm.print_rawinfostr()
>>> # should be sorted already from the shortlist take
>>> top_nid_list = cm.dnid_list
>>> top_aid_list = cm.daid_list
>>> qnid = cm.qnid
>>> print('top_aid_list = %r' % (top_aid_list,))
>>> print('top_nid_list = %r' % (top_nid_list,))
>>> print('qnid = %r' % (qnid,))
>>> rankx = top_nid_list.tolist().index(qnid)
>>> assert rankx == 0, 'qnid=%r should be first rank, not rankx=%r' % (qnid,␣
↪rankx)
>>> max_num_rerank = nNameShortList * nAnnotPerName
>>> min_num_rerank = nNameShortList
>>> ut.assert_inbounds(len(top_nid_list), min_num_rerank, max_num_rerank,
↪'incorrect number in shortlist', eq=True)
>>> ut.quit_if_noshow()
>>> cm.show_single_annotmatch(qreq_, daid=top_aid_list[0])
>>> ut.show_if_requested()
```

wbia.algo.hots.scoring.**score_chipmatch_list**(*qreq_*, *cm_list*, *score_method*, *progkw=None*)

CommandLine: python -m wbia.algo.hots.scoring –test-score_chipmatch_list python -m wbia.algo.hots.scoring –test-score_chipmatch_list:1 python -m wbia.algo.hots.scoring –test-score_chipmatch_list:0 –show

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +SKIP
>>> # (IMPORTANT)
>>> from wbia.algo.hots.scoring import *  # NOQA
>>> ibs, qreq_, cm_list = plh.testdata_pre_sver()
>>> score_method = qreq_.qparams.prescore_method
>>> score_chipmatch_list(qreq_, cm_list, score_method)
>>> cm = cm_list[0]
>>> assert cm.score_list.argmax() == 0
>>> ut.quit_if_noshow()
>>> cm.show_single_annotmatch(qreq_)
>>> ut.show_if_requested()
```

Example

```
>>> # SLOW_DOCTEST
>>> # (IMPORTANT)
>>> from wbia.algo.hots.scoring import *  # NOQA
>>> ibs, qreq_, cm_list = plh.testdata_post_sver()
>>> qaid = qreq_.qaids[0]
>>> cm = cm_list[0]
>>> score_method = qreq_.qparams.score_method
>>> score_chipmatch_list(qreq_, cm_list, score_method)
>>> assert cm.score_list.argmax() == 0
>>> ut.quit_if_noshow()
>>> cm.show_single_annotmatch(qreq_)
>>> ut.show_if_requested()
```

1.1.1.3.17 wbia.algo.hots.toy_nan_rf module

wbia.algo.hots.toy_nan_rf.**get_toydata**(*rng*)

wbia.algo.hots.toy_nan_rf.**main**()

SeeAlso: python -m sklearn.ensemble.tests.test_forest test_multioutput

CommandLine: python -m wbia toy_classify_nans python -m wbia toy_classify_nans –toy1 –save “rf_nan_toy1.jpg” –figsize=10,10 python -m wbia toy_classify_nans –toy2 –save “rf_nan_toy2.jpg” –figsize=10,10 python -m wbia toy_classify_nans –toy2 –save “rf_nan_toy3.jpg” –figsize=10,10 –extra python -m wbia toy_classify_nans –toy2 –save “rf_nan_toy4.jpg” –figsize=10,10 –extra –nanrate=0 python -m wbia toy_classify_nans –toy2 –save “rf_nan_toy5.jpg” –figsize=10,10 –nanrate=0

Example

```
>>> # DISABLE_DOCTEST
>>> result = toy_classify_nans()
```

wbia.algo.hots.toy_nan_rf.**show_nan_decision_function_2d**(*X*, *y*, *X_true*, *clf*)

wbia.algo.hots.toy_nan_rf.**toydata1**(*rng*)

Description of Plot

You'll notice that there are 4 plots. This is necessary to visualize a grid with nans. Each plot shows points in the 2-dimensional grid with corners at (0, 0) and (40, 40). The top left plot has these coordinates labeled. The other 3 plots correspond to the top left grid, but in these plots at least one of the dimensions has been "nanned". In the top right the x-dimension is "nanned". In the bottom left the y-dimension is "nanned", and in the bottom right both dimensions are "nanned". Even though all plots are drawn as a 2d-surface only the topleft plot is truly a surface with 2 degrees of freedom. The top right and bottom left plots are really lines with 1 degree of freedom, and the bottom right plot is actually just a single point with 0 degrees of freedom.

In this example I create 10 Gaussian blobs where the first 9 have their means laid out in a 3x3 grid and the last one has its mean in the center, but I gave it a high standard deviation. I'll refer to the high std cluster as 9, and label the other clusters at the grid means (to agree with the demo code) like this:

` 6 7 8 3 4 5 0 1 2 `

Looking at the top left plot you can see clusters 0, 1, 2, 4, 6, and 8. The reason the other cluster do not appear in this grid is because I've set at least one of their dimensions to be nan. Specifically, cluster 3 had its y dimension set to nan; cluster 5 and 7 had their x dimension set to nan; and cluster 9 had both x and y dimensions set to nan.

For clusters 3, 5, and 7, I plot "nanned" points as lines along the nanned dimension to show that only the non-nan dimensions can be used to distinguish these points. I also plot the original position before I "nanned" it for visualization purposes, but the learning algorithm never sees this. For cluster 9, I only plot the original positions because all of this data collapses to a single point [nan, nan].

Red points are of class 0, and blue points are of class 1. Points in each plot represent the training data. The colored background of each plot represents the classification surface.

wbia.algo.hots.toy_nan_rf.**toydata2**(*rng*)

1.1.1.3.18 Module contents

wbia.algo.hots.**IMPORT_TUPLES = [('_pipeline_helpers', None), ('chip_match', None), ('except**

cd /home/joncrall/code/wbia/wbia/algo/hots makeinit.py –modname=wbia.algo.hots

Type Regen Command

wbia.algo.hots.**reassign_submodule_attributes**(*verbose=True*)

why reloading all the modules doesnt do this I don't know

wbia.algo.hots.**reload_subs**(*verbose=True*)

Reloads wbia.algo.hots and submodules

wbia.algo.hots.**rrrr**(*verbose=True*)

Reloads wbia.algo.hots and submodules

1.1.1.4 wbia.algo.preproc package

1.1.1.4.1 Submodules

1.1.1.4.2 wbia.algo.preproc.occurrence_blackbox module

animal_walking_speeds

ZEBRA_SPEED_MAX = 64 # km/h ZEBRA_SPEED_RUN = 50 # km/h ZEBRA_SPEED_SLOW_RUN = 20 # km/h ZEBRA_SPEED_FAST_WALK = 10 # km/h ZEBRA_SPEED_WALK = 7 # km/h

km_per_sec = .02 km_per_sec = .001 mph = km_per_sec / ut.KM_PER_MILE * 60 * 60 print('mph = %r' % (mph,))

1 / km_per_sec

import datetime thresh_sec = datetime.timedelta(minutes=5).seconds thresh_km = thresh_sec * km_per_sec print('thresh_sec = %r' % (thresh_sec,)) print('thresh_km = %r' % (thresh_km,)) thresh_sec = thresh_km / km_per_sec print('thresh_sec = %r' % (thresh_sec,))

wbia.algo.preproc.occurrence_blackbox.**cluster_timespace_km**(*posixtimes*, *latlons*, *thresh_km*, *km_per_sec=0.002*)

Agglometerative clustering of time/space data

Parameters

- **X_data** (*ndarray*) – Nx3 array where columns are (seconds, lat, lon)
- **thresh_km** (*float*) – threshold in kilometers

References

http://docs.scipy.org/doc/scipy-0.14.0/reference/generated/ scipy.cluster.hierarchy.linkage.html scipy.cluster.hierarchy.fcluster.html

Notes

Visualize spots http://www.darrinward.com/lat-long/?id=2009879

CommandLine: python -m wbia.algo.preproc.occurrence_blackbox cluster_timespace_km

Doctest:

```
>>> from wbia.algo.preproc.occurrence_blackbox import *  # NOQA
>>> # Nx1 matrix denoting groundtruth locations (for testing)
>>> X_name = np.array([0, 1, 1, 1, 1, 1, 2, 2, 2])
>>> # Nx3 matrix where each columns are (time, lat, lon)
>>> X_data = np.array([
>>>     (0, 42.727985, -73.683994),  # MRC
>>>     (0, 42.657414, -73.774448),  # Park1
>>>     (0, 42.658333, -73.770993),  # Park2
>>>     (0, 42.654384, -73.768919),  # Park3
>>>     (0, 42.655039, -73.769048),  # Park4
>>>     (0, 42.657872, -73.764148),  # Park5
>>>     (0, 42.876974, -73.819311),  # CP1
>>>     (0, 42.862946, -73.804977),  # CP2
>>>     (0, 42.849809, -73.758486),  # CP3
>>> ])
```

(continues on next page)

(continued from previous page)

```
>>> thresh_km = 5.0  # kilometers
>>> posixtimes = X_data.T[0]
>>> latlons = X_data.T[1:3].T
>>> km_per_sec = KM_PER_SEC
>>> X_labels = cluster_timespace_km(posixtimes, latlons, thresh_km)
>>> result = 'X_labels = {}'.format(ut.repr2(X_labels))
>>> print(result)
X_labels = np.array([3, 2, 2, 2, 2, 2, 1, 1, 1])
```

wbia.algo.preproc.occurrence_blackbox.**cluster_timespace_sec**(*posixtimes*, *latlons*, *thresh_sec=5*, *km_per_sec=0.002*)

Parameters

- **X_data** (*ndarray*) – Nx3 array where columns are (seconds, lat, lon)
- **thresh_sec** (*float*) – threshold in seconds

Doctest:

```
>>> from wbia.algo.preproc.occurrence_blackbox import *  # NOQA
>>> # Nx1 matrix denoting groundtruth locations (for testing)
>>> X_name = np.array([0, 1, 1, 1, 1, 1, 2, 2, 2])
>>> # Nx3 matrix where each columns are (time, lat, lon)
>>> X_data = np.array([
>>>     (0, 42.727985, -73.683994),  # MRC
>>>     (0, 42.657414, -73.774448),  # Park1
>>>     (0, 42.658333, -73.770993),  # Park2
>>>     (0, 42.654384, -73.768919),  # Park3
>>>     (0, 42.655039, -73.769048),  # Park4
>>>     (0, 42.657872, -73.764148),  # Park5
>>>     (0, 42.876974, -73.819311),  # CP1
>>>     (0, 42.862946, -73.804977),  # CP2
>>>     (0, 42.849809, -73.758486),  # CP3
>>> ])
>>> posixtimes = X_data.T[0]
>>> latlons = X_data.T[1:3].T
>>> thresh_sec = 250  # seconds
>>> X_labels = cluster_timespace_sec(posixtimes, latlons, thresh_sec)
>>> result = ('X_labels = %r' % (X_labels,))
>>> print(result)
X_labels = array([6, 4, 4, 4, 4, 5, 1, 2, 3])
```

Doctest:

```
>>> from wbia.algo.preproc.occurrence_blackbox import *  # NOQA
>>> # Nx1 matrix denoting groundtruth locations (for testing)
>>> X_name = np.array([0, 1, 1, 1, 1, 1, 2, 2, 2])
>>> # Nx3 matrix where each columns are (time, lat, lon)
>>> X_data = np.array([
>>>     (np.nan, 42.657414, -73.774448),  # Park1
>>>     (0, 42.658333, -73.770993),  # Park2
>>>     (np.nan, np.nan, np.nan),  # Park3
>>>     (np.nan, np.nan, np.nan),  # Park3.5
>>>     (0, 42.655039, -73.769048),  # Park4
>>>     (0, 42.657872, -73.764148),  # Park5
```

(continues on next page)

(continued from previous page)

```
>>> ])
>>> posixtimes = X_data.T[0]
>>> latlons = X_data.T[1:3].T
>>> thresh_sec = 250  # seconds
>>> km_per_sec = KM_PER_SEC
>>> X_labels = cluster_timespace_sec(posixtimes, latlons, thresh_sec)
>>> result = 'X_labels = {}'.format(ut.repr2(X_labels))
>>> print(result)
X_labels = np.array([3, 4, 1, 2, 4, 5])
```

wbia.algo.preproc.occurrence_blackbox.**haversine**(*latlon1*, *latlon2*)

Calculate the great circle distance between two points on the earth (specified in decimal degrees)

Parameters

- **latlon1** (*tuple*) – (lat, lon)
- **latlon2** (*tuple*) – (lat, lon)

Returns distance in kilometers

Return type float

References

en.wikipedia.org/wiki/Haversine_formula gis.stackexchange.com/questions/81551/matching-gps-tracks stackoverflow.com/questions/4913349/haversine-distance-gps-points

Doctest:

```
>>> from wbia.algo.preproc.occurrence_blackbox import *  # NOQA
>>> import scipy.spatial.distance as spdist
>>> import functools
>>> latlon1 = [-80.21895315, -158.81099213]
>>> latlon2 = [  9.77816711,  -17.27471498]
>>> kilometers = haversine(latlon1, latlon2)
>>> result = ('kilometers = %0.08f' % (kilometers,))
>>> print(result)
kilometers = 11930.90936419
```

wbia.algo.preproc.occurrence_blackbox.**haversine_rad**(*lat1*, *lon1*, *lat2*, *lon2*)

wbia.algo.preproc.occurrence_blackbox.**main**()

CommandLine: ib cd ~/code/wbia/wbia/algo/preproc python occurrence_blackbox.py –lat 42.727985 42.657414 42.658333 42.654384 –lon -73.683994 -73.774448 -73.770993 -73.768919 –sec 0 0 0 0 # Should return X_labels = [2, 1, 1, 1]

wbia.algo.preproc.occurrence_blackbox.**prepare_data**(*posixtimes*, *latlons*, *km_per_sec=0.002*, *thresh_units='seconds'*)

Package datas and picks distance function

Parameters

- **posixtimes** (*ndarray*) –
- **latlons** (*ndarray*) –
- **km_per_sec** (*float*) – (default = 0.002)

- **thresh_units** (*str*) – (default = 'seconds')

Returns arr_ -

Return type ndarray

CommandLine: python -m wbia.algo.preproc.occurrence_blackbox prepare_data

Doctest:

```
>>> from wbia.algo.preproc.occurrence_blackbox import *  # NOQA
>>> posixtimes = np.array([10, 50, np.nan, np.nan, 5, 80, np.nan, np.nan])
>>> latlons = np.array([
>>>     (42.727985, -73.683994),
>>>     (np.nan, np.nan),
>>>     (np.nan, np.nan),
>>>     (42.658333, -73.770993),
>>>     (42.227985, -73.083994),
>>>     (np.nan, np.nan),
>>>     (np.nan, np.nan),
>>>     (42.258333, -73.470993),
>>> ])
>>> km_per_sec = 0.002
>>> thresh_units = 'seconds'
>>> X_data, dist_func, columns = prepare_data(posixtimes, latlons, km_per_sec,
↪ thresh_units)
>>> result = ('arr_ = %s' % (ut.repr2(X_data),))
>>> [dist_func(a, b) for a, b in ut.combinations(X_data, 2)]
>>> print(result)
```

wbia.algo.preproc.occurrence_blackbox.**space_distance_km**(*pt1*, *pt2*)

wbia.algo.preproc.occurrence_blackbox.**space_distance_sec**(*pt1*, *pt2*, *km_per_sec=0.002*)

wbia.algo.preproc.occurrence_blackbox.**time_dist_km**(*sec1*, *sec2*, *km_per_sec=0.002*)

wbia.algo.preproc.occurrence_blackbox.**time_dist_sec**(*sec1*, *sec2*)

wbia.algo.preproc.occurrence_blackbox.**timespace_distance_km**(*pt1*, *pt2*, *km_per_sec=0.002*)

Computes distance between two points in space and time. Time is converted into spatial units using km_per_sec

Parameters

- **pt1** (*tuple*) – (seconds, lat, lon)
- **pt2** (*tuple*) – (seconds, lat, lon)
- **km_per_sec** (*float*) – reasonable animal walking speed

Returns distance in kilometers

Return type float

Doctest:

```
>>> from wbia.algo.preproc.occurrence_blackbox import *  # NOQA
>>> import scipy.spatial.distance as spdist
>>> import functools
>>> km_per_sec = .02
>>> latlon1 = [40.779299,-73.9719498] # museum of natural history
```

(continues on next page)

(continued from previous page)

```
>>> latlon2 = [37.7336402,-122.5050342] # san fransisco zoo
>>> pt1 = [0.0] + latlon1
>>> pt2 = [0.0] + latlon2
>>> # google measures about 4138.88 kilometers
>>> dist_km1 = timespace_distance_km(pt1, pt2)
>>> print('dist_km1 = {!r}'.format(dist_km1))
>>> # Now add a time component
>>> pt1 = [360.0] + latlon1
>>> pt2 = [0.0] + latlon2
>>> dist_km2 = timespace_distance_km(pt1, pt2)
>>> print('dist_km2 = {!r}'.format(dist_km2))
>>> assert np.isclose(dist_km1, 4136.4568647922624)
>>> assert np.isclose(dist_km2, 4137.1768647922627)
```

wbia.algo.preproc.occurrence_blackbox.**timespace_distance_sec**(*pt1*, *pt2*, *km_per_sec=0.002*)

1.1.1.4.3 wbia.algo.preproc.preproc_annot module

helpers for controller manual_annot_funcs

wbia.algo.preproc.preproc_annot.**generate_annot_properties**(*ibs*, *gid_list*, *bbox_list=None*, *theta_list=None*, *species_list=None*, *nid_list=None*, *name_list=None*, *detect_confidence_list=None*, *notes_list=None*, *vert_list=None*, *annot_uuid_list=None*, *yaw_list=None*, *quiet_delete_thumbs=False*)

wbia.algo.preproc.preproc_annot.**make_annotation_uuids**(*image_uuid_list*, *bbox_list*, *theta_list*, *deterministic=True*)

wbia.algo.preproc.preproc_annot.**postget_annot_verts**(*vertstr_list*)

wbia.algo.preproc.preproc_annot.**testdata_preproc_annot**()

1.1.1.4.4 wbia.algo.preproc.preproc_image module

wbia.algo.preproc.preproc_image.**get_standard_ext**(*gpath*)
Returns standardized image extension

wbia.algo.preproc.preproc_image.**on_delete**(*ibs*, *featweight_rowid_list*, *qreq_=None*)

wbia.algo.preproc.preproc_image.**parse_exif**(*pil_img*)
Image EXIF helper

wbia.algo.preproc.preproc_image.**parse_imageinfo**(*gpath*, *cleanup=False*)
Worker function: gpath must be in UNIX-PATH format!

Parameters **gpath** (*str*) – image path

Returns

param_tup - if successful returns a tuple of image parameters which are values for SQL columns on else returns None

Return type tuple

CommandLine: python -m wbia.algo.preproc.preproc_image –exec-parse_imageinfo

Doctest:

```
>>> from wbia.algo.preproc.preproc_image import *  # NOQA
>>> gpath = ut.grab_test_imgpath('patsy.jpg')
>>> gpath_, param_tup = parse_imageinfo(gpath)
>>> result = ('param_tup = %s' % (str(param_tup),))
>>> print(result)
>>> uuid = param_tup[0]
>>> assert str(uuid) == '16008058-788c-2d48-cd50-f6029f726cbf'
```

1.1.1.4.5 wbia.algo.preproc.preproc_occurrence module

wbia.algo.preproc.preproc_occurrence.**agglomerative_cluster_occurrences**(*X_data*, *thresh_sec*)

Agglomerative occurrence clustering algorithm

Parameters

- **X_data** (*ndarray*) – Length N array of data to cluster
- **thresh_sec** (*float*) –

Returns (label_arr) - Length N array of cluster indexes

Return type ndarray

CommandLine: python -m wbia.algo.preproc.preproc_occurrence –exec-agglomerative_cluster_occurrences

References

https://docs.scipy.org/doc/scipy-0.9.0/reference/generated/scipy.cluster.hierarchy.fclusterdata.html#scipy.cluster.hierarchy.fclusterdata http://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.cluster.hierarchy.fcluster.html

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.preproc.preproc_occurrence import *  # NOQA
>>> X_data = '?'
>>> thresh_sec = '?'
>>> (occur_ids, occur_gids) = agglomerative_cluster_occurrences(X_data, thresh_
↪sec)
>>> result = ('(occur_ids, occur_gids) = %s' % (str((occur_ids, occur_gids)),))
>>> print(result)
```

wbia.algo.preproc.preproc_occurrence.**cluster_timespace**(*X_data*, *thresh*)

References

http://docs.scipy.org/doc/scipy-0.14.0/reference/generated/ scipy.cluster.hierarchy.linkage.html

CommandLine: python -m wbia.algo.preproc.preproc_occurrence cluster_timespace –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.preproc.preproc_occurrence import *  # NOQA
>>> X_data = testdata_gps()
>>> thresh = 10
>>> X_labels = cluster_timespace(X_data, thresh)
>>> fnum = pt.ensure_fnum(None)
>>> fig = pt.figure(fnum=fnum, doclf=True, docla=True)
>>> hier.dendrogram(linkage_mat, orientation='top')
>>> plot_annotaiton_gps(X_data)
>>> ut.show_if_requested()
```

wbia.algo.preproc.preproc_occurrence.**compute_occurrence_groups**(*ibs*, *gid_list*, *config={}*, *use_gps=False*, *verbose=None*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) –

Returns (None, None)

Return type tuple

CommandLine: python -m wbia compute_occurrence_groups

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.preproc.preproc_occurrence import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> verbose = True
>>> images = ibs.images()
>>> gid_list = images.gids
>>> config = {}  # wbia.algo.Config.OccurrenceConfig().asdict()
>>> tup = wbia_compute_occurrences(ibs, gid_list)
>>> (flat_imgsetids, flat_gids)
>>> aids_list = list(ut.group_items(aid_list_, flat_imgsetids).values())
>>> metric = list(map(len, aids_list))
>>> sortx = ut.list_argsort(metric)[::-1]
>>> index = sortx[1]
>>> aids = aids_list[index]
>>> gids = list(set(ibs.get_annot_gids(aids)))
```

wbia.algo.preproc.preproc_occurrence.**compute_occurrence_unixtime**(*ibs*, *occur_gids*)

wbia.algo.preproc.preproc_occurrence.**filter_and_relabel**(*labels*, *label_gids*, *min_imgs_per_occurence*, *occur_unixtimes=None*)

Removes clusters with too few members. Relabels clusters-labels such that label 0 has the most members

wbia.algo.preproc.preproc_occurrence.**group_images_by_label**(*label_arr*, *gid_arr*)

Input: Length N list of labels and ids Output: Length M list of unique labels, and lenth M list of lists of ids

wbia.algo.preproc.preproc_occurrence.**meanshift_cluster_occurrences**(*X_data*, *quantile*)

Meanshift occurrence clustering algorithm

Parameters

- **X_data** (*ndarray*) – Length N array of data to cluster
- **quantile** (*float*) – quantile should be between [0, 1]. eg: quantile=.5 represents the median of all pairwise distances

Returns Length N array of labels

Return type ndarray

CommandLine: python -m wbia.algo.preproc.preproc_occurrence –exec-meanshift_cluster_occurrences

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.preproc.preproc_occurrence import *  # NOQA
>>> X_data = '?'
>>> quantile = '?'
>>> result = meanshift_cluster_occurrences(X_data, quantile)
>>> print(result)
```

wbia.algo.preproc.preproc_occurrence.**plot_gps_html**(*gps_list*)

Plots gps coordinates on a map projection

InstallBasemap: sudo apt-get install libgeos-dev pip install git+https://github.com/matplotlib/basemap http://matplotlib.org/basemap/users/examples.html

pip install gmplot

sudo apt-get install netcdf-bin sudo apt-get install libnetcdf-dev pip install netCDF4

Ignore: pip install git+git://github.com/myuser/foo.git@v123

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.preproc.preproc_occurrence import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> images = ibs.images()
>>> # Setup GPS points to draw
>>> print('Setup GPS points')
>>> gps_list_ = np.array(images.gps2)
>>> unixtime_list_ = np.array(images.unixtime2)
>>> has_gps = np.all(np.logical_not(np.isnan(gps_list_)), axis=1)
```

(continues on next page)

(continued from previous page)

```
>>> has_unixtime = np.logical_not(np.isnan(unixtime_list_))
>>> isvalid = np.logical_and(has_gps, has_unixtime)
>>> gps_list = gps_list_.compress(isvalid, axis=0)
>>> unixtime_list = unixtime_list_.compress(isvalid)  # NOQA
>>> plot_image_gps(gps_list)
```

wbia.algo.preproc.preproc_occurrence.**prepare_X_data**(*ibs*, *gid_list*, *use_gps=True*)

Splits data into groups with/without gps and time

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.preproc.preproc_occurrence import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> images = ibs.images()
>>> # wbia.control.accessor_decors.DEBUG_GETTERS = True
>>> use_gps = True
>>> gid_list = images.gids
>>> datas = prepare_X_data(ibs, gid_list, use_gps)
>>> print(ut.repr2(datas, nl=2, precision=2))
>>> assert len(datas['both'][0]) == 12
>>> assert len(datas['neither'][0]) == 0
```

wbia.algo.preproc.preproc_occurrence.**testdata_gps**()

Simple data to test GPS algorithm.

Returns Nx1 matrix denoting groundtruth locations X_data (ndarray): Nx3 matrix where each columns are (time, lat, lon)

Return type X_name (ndarray)

wbia.algo.preproc.preproc_occurrence.**timespace_distance**(*pt1*, *pt2*)

wbia.algo.preproc.preproc_occurrence.**timespace_pdist**(*X_data*)

wbia.algo.preproc.preproc_occurrence.**wbia_compute_occurrences**(*ibs*, *gid_list*, *config=None*, *verbose=None*)

clusters occurrences togethers (by time, not yet space) An occurrence is a meeting, localized in time and space between a camera and a group of animals. Animals are identified within each occurrence.

Does not modify database state, just returns cluster ids

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) –

Returns (None, None)

Return type tuple

CommandLine: python -m wbia –tf wbia_compute_occurrences:0 –show TODO: FIXME: good example of autogen doctest return failure

1.1.1.4.6 wbia.algo.preproc.preproc_residual module

wbia.algo.preproc.preproc_residual.**add_residual_params_gen**(*ibs*, *fid_list*, *qreq_=None*)

wbia.algo.preproc.preproc_residual.**on_delete**(*ibs*, *featweight_rowid_list*)

1.1.1.4.7 wbia.algo.preproc.preproc_rvec module

wbia.algo.preproc.preproc_rvec.**add_rvecs_params_gen**(*ibs*, *nInput=None*)

wbia.algo.preproc.preproc_rvec.**generate_rvecs**(*vecs_list*, *words*)

1.1.1.4.8 Module contents

wbia.algo.preproc.**IMPORT_TUPLES = [('preproc_annot', None), ('preproc_image', None), ('prep**
cd /home/joncrall/code/wbia/wbia/algo/preproc makeinit.py –modname=wbia.algo.preproc –write

Type Regen Command

wbia.algo.preproc.**reassign_submodule_attributes**(*verbose=True*)
why reloading all the modules doesnt do this I don't know

wbia.algo.preproc.**reload_subs**(*verbose=True*)
Reloads wbia.algo.preproc and submodules

wbia.algo.preproc.**rrrr**(*verbose=True*)
Reloads wbia.algo.preproc and submodules

1.1.1.5 wbia.algo.smk package

1.1.1.5.1 Submodules

1.1.1.5.2 wbia.algo.smk.inverted_index module

class wbia.algo.smk.inverted_index.**InvertedAnnots**
Bases: *wbia.algo.smk.inverted_index.InvertedAnnotsExtras*

CommandLine: python -m wbia.algo.smk.inverted_index InvertedAnnots –show

Ignore:

```
>>> from wbia.algo.smk.inverted_index import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_(defaultdb='Oxford', a='oxford',
>>>                             p='default:proot=smk,nAssign=1,num_
↪words=64000')
>>> config = qreq_.qparams
>>> ibs = qreq_.ibs
>>> depc = qreq_.ibs.depc
>>> aids = qreq_.daids
>>> aids = qreq_.qaids
>>> input_tuple = (aids, [qreq_.daids])
>>> inva = ut.DynStruct()
>>> inva = InvertedAnnots(aids, qreq_)
```

Example

```
>>> # DISABLE_DOCTEST
>>> qreq_, inva = testdata_inva()
```

compute_gammas(*alpha*, *thresh*)

Example

```
>>> # xdoctest: +REQUIRES(--slow)
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.inverted_index import *  # NOQA
>>> qreq_, inva = testdata_inva()
>>> inva.wx_to_weight = inva.compute_word_weights('uniform')
>>> alpha = 3.0
>>> thresh = 0.0
>>> gamma_list = inva.compute_gammas(alpha, thresh)
```

compute_inverted_list()

compute_word_weights(*method='idf'*)
Compute a per-word weight like idf

Example

```
>>> # xdoctest: +REQUIRES(--slow)
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.inverted_index import *  # NOQA
>>> qreq_, inva = testdata_inva()
>>> wx_to_weight = inva.compute_word_weights()
>>> print('wx_to_weight = %r' % (wx_to_weight,))
```

classmethod from_depc(*depc*, *aids*, *vocab_aids*, *config*)

get_annot(*aid*)

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

wx_list

class wbia.algo.smk.inverted_index.**InvertedAnnotsExtras**
Bases: object

get_nbytes()

get_patches(*wx*, *ibs*, *verbose=True*)
Loads the patches assigned to a particular word in this stack

```
>>> inva.wx_to_aids = inva.compute_inverted_list()
>>> verbose=True
```

get_size_info()

get_word_patch(*wx*, *ibs*)

print_size_info()

render_inverted_vocab(*ibs*, *use_data=False*)

Renders the average patch of each word. This is a visualization of the entire vocabulary.

CommandLine: python -m wbia.algo.smk.inverted_index render_inverted_vocab –show python -m wbia.algo.smk.inverted_index render_inverted_vocab –show –use-data python -m wbia.algo.smk.inverted_index render_inverted_vocab –show –debug-depc

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.smk.inverted_index import *  # NOQA
>>> qreq_, inva = testdata_inva()
>>> ibs = qreq_.ibs
>>> all_words = inva.render_inverted_vocab(ibs)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.qt4ensure()
>>> pt.imshow(all_words)
>>> ut.show_if_requested()
```

render_inverted_vocab_word(*wx*, *ibs*, *fnum=None*)

Creates a visualization of a visual word. This includes the average patch, the SIFT-like representation of the centroid, and some of the patches that were assigned to it.

CommandLine: python -m wbia.algo.smk.inverted_index render_inverted_vocab_word –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.smk.inverted_index import *  # NOQA
>>> import wbia.plottool as pt
>>> qreq_, inva = testdata_inva()
>>> ibs = qreq_.ibs
>>> wx_list = list(inva.wx_to_aids.keys())
>>> wx = wx_list[0]
>>> ut.qtensure()
>>> fnum = 2
>>> fnum = pt.ensure_fnum(fnum)
>>> # Interactive visualization of many words
>>> for wx in ut.InteractiveIter(wx_list):
>>>     word_img = inva.render_inverted_vocab_word(wx, ibs, fnum)
>>>     pt.imshow(word_img, fnum=fnum, title='Word %r/%r' % (wx, '?'))
>>>     pt.update()
```

class wbia.algo.smk.inverted_index.**InvertedIndexConfig**(***kwargs*)

Bases: *wbia.dtool.base.Config*

class wbia.algo.smk.inverted_index.**SingleAnnot**

Bases: utool.util_dev.NiceRepr

Phis_flags(*idxs*)

get subset of aggregated residual vectors

classmethod from_inva(*inva*, *idx*)

fxs(*c*)

maws(*c*)

nbytes()

nbytes_info()

phis_flags_list(*idxs*)
get subset of non-aggregated residual vectors

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

to_dense(*inva=None*, *out=None*)

words

wbia.algo.smk.inverted_index.**compute_residual_assignments**(*depc*, *fid_list*, *vocab_id_list*, *config*)

CommandLine:

python -m wbia.control.IBEISControl show_depc_annot_table_input –show –tablename=residuals

Ignore: ibs.depc['vocab'].print_table()

Ignore: data = ibs.depc.get('inverted_agg_assign', ([1, 2473], qreq_.daids), config=qreq_.config) wxs1 = data[0][0] wxs2 = data[1][0]

Lev Example import wbia ibs = wbia.opendb('Oxford') depc = ibs.depc table = depc['inverted_agg_assign'] table.print_table() table.print_internal_info()

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.smk.inverted_index import *  # NOQA
>>> # Test depcache access
>>> import wbia
>>> ibs, aid_list = wbia.testdata_aids('testdb1')
>>> depc = ibs.depc_annot
>>> config = {'num_words': 1000, 'nAssign': 1}
>>> #input_tuple = (aid_list, [aid_list] * len(aid_list))
>>> daids = aid_list
>>> input_tuple = (daids, [daids])
>>> rowid_kw = {}
>>> tablename = 'inverted_agg_assign'
>>> target_tablename = tablename
>>> input_ids = depc.get_parent_rowids(tablename, input_tuple, config)
>>> fid_list = ut.take_column(input_ids, 0)
>>> vocab_id_list = ut.take_column(input_ids, 1)
>>> data = depc.get(tablename, input_tuple, config)
>>> tup = dat[1]
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.smk.inverted_index import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_(defaultdb='Oxford', a='oxford', p=
↪'default:proot=smk,nAssign=1,num_words=64000')
>>> config = {'num_words': 64000, 'nAssign': 1, 'int_rvec': True}
```

(continues on next page)

(continued from previous page)

```
>>> depc = qreq_.ibs.depc
>>> daids = qreq_.daids
>>> input_tuple = (daids, [daids])
>>> rowid_kw = {}
>>> tablename = 'inverted_agg_assign'
>>> target_tablename = tablename
>>> input_ids = depc.get_parent_rowids(tablename, input_tuple, config)
>>> fid_list = ut.take_column(input_ids, 0)
>>> vocab_id_list = ut.take_column(input_ids, 1)
```

wbia.algo.smk.inverted_index.**gen_residual_args**(*vocab*, *vecs_list*, *nAssign*, *int_rvec*)

wbia.algo.smk.inverted_index.**residual_args**(*vocab*, *vecs*, *nAssign*, *int_rvec*)

wbia.algo.smk.inverted_index.**residual_worker**(*argtup*)

wbia.algo.smk.inverted_index.**testdata_inva**()
from wbia.algo.smk.inverted_index import * # NOQA

1.1.1.5.3 wbia.algo.smk.match_chips5 module

TODO: semantic_uuids should be replaced with PCC-like hashes pertaining to annotation clusters if any form of name scoring is used.

class wbia.algo.smk.match_chips5.**EstimatorRequest**
Bases: utool.util_dev.NiceRepr

dnids
save dnids in **qreq_** state

Type TODO

ensure_nids()

execute(*qaids=None*, *prog_hook=None*, *use_cache=True*)

extern_data_config2

extern_query_config2

get_cfgstr(*with_input=False*, *with_data=True*, *with_pipe=True*, *hash_pipe=False*)

get_chipmatch_fpaths(*qaid_list*)
Efficient function to get a list of chipmatch paths

get_data_hashid()

get_nice_parts()

get_pipe_cfgstr()

get_pipe_hashid()

get_qreq_annot_gids(*aids*)

get_qreq_annot_nids(*aids*)

get_query_hashid()

qnids
save qnids in **qreq_** state

Type TODO

shallowcopy(*qaids=None*)
Creates a copy of qreq with the same qparams object and a subset of the qx and dx objects. used to generate chunks of vsone and vsmany queries

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.match_chips5 import *  # NOQA
>>> from wbia.algo.smk.smk_pipeline import testdata_smk
>>> import wbia
>>> wbia, smk, qreq_ = testdata_smk()
>>> qreq2_ = qreq_.shallowcopy(qaids=1)
>>> assert qreq_.daids is qreq2_.daids, 'should be the same'
>>> assert len(qreq_.qaids) != len(qreq2_.qaids), 'should be diff'
>>> #assert qreq_.metadata is not qreq2_.metadata
```

wbia.algo.smk.match_chips5.**execute_and_save**(*qreq_miss*)

wbia.algo.smk.match_chips5.**execute_bulk**(*qreq_*)

wbia.algo.smk.match_chips5.**execute_singles**(*qreq_*)

1.1.1.5.4 wbia.algo.smk.pickle_flann module

class wbia.algo.smk.pickle_flann.**PickleFLANN**(***kwargs*)
Bases: pyflann.index.FLANN

Adds the ability to pickle a flann class on a unix system. (Actually, pickle still wont work because we need the original point data. But we can do a custom dumps and a loads)

CommandLine: python -m wbia.algo.smk.pickle_flann PickleFLANN

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.pickle_flann import *  # NOQA
>>> import numpy as np
>>> rng = np.random.RandomState(42)
>>> data = rng.rand(10, 2)
>>> query = rng.rand(5, 2)
>>> flann = PickleFLANN()
>>> flann.build_index(data, random_seed=42)
>>> index_bytes = flann.dumps()
>>> flann2 = PickleFLANN()
>>> flann2.loads(index_bytes, data)
>>> assert flann2 is not flann
>>> assert flann2.dumps() == index_bytes
>>> idx1 = flann.nn_index(query)[0]
>>> idx2 = flann2.nn_index(query)[0]
>>> assert np.all(idx1 == idx2)
```

dumps()
Make a special wordflann pickle http://www.linuxscrew.com/2010/03/24/fastest-way-to-create-ramdisk-in-ubuntulinux/ sudo mkdir /tmp/ramdisk; chmod 777 /tmp/ramdisk sudo mount -t tmpfs -o size=256M tmpfs /tmp/ramdisk/ http://zeblog.co/?p=1588

loads(*index_bytes*, *pts*)

class wbia.algo.smk.pickle_flann.**Win32CompatTempFile**(*delete=True*, *verbose=False*)
Bases: object

mimics tempfile.NamedTemporaryFile but allows the file to be closed without being deleted. This lets a second process (like the FLANN) read/write to the file in a win32 system. The file is instead deleted after the Win32CompatTempFile object goes out of scope.

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.pickle_flann import *  # NOQA
>>> verbose = True
>>> temp = Win32CompatTempFile(verbose=verbose)
>>> data = '10010'
>>> data = data.encode()
>>> print('data = %r' % (data,))
>>> data1 = temp.read()
>>> print('data1 = %r' % (data1,))
>>> temp.write(data)
>>> data2 = temp.read()
>>> print('data2 = %r' % (data2,))
>>> temp.close()
>>> assert data != data1
>>> assert data == data2
>>> ut.assert_raises(ValueError, temp.close)
>>> assert not ut.checkpath(temp.fpath, verbose=verbose)
```

close()

name

read()

write(*data*)

1.1.1.5.5 wbia.algo.smk.script_smk module

Results so far without SV / fancyness Using standard descriptors / vocabulary

proot=bow,nWords=1E6 -> .594 proot=asmk,nWords=1E6 -> .529

Note:

- Results from SMK Oxford Paper (mAP)

ASMK nAssign=1, SV=False: .78 ASMK nAssign=5, SV=False: .82

Philbin with tf-idf ranking SV=False SIFT: .636, RootSIFT: .683 (+.05)

Philbin with tf-idf ranking SV=True SIFT: .672, RootSIFT: .720 (+.05)

- My Results (WITH BAD QUERY BBOXES)

smk:nAssign=1,SV=True,: .58 smk:nAssign=1,SV=False,: .38

Yesterday I got .22 when I fixed the bounding boxes And now I'm getting .08 and .32 (sv=[F,T]) after deleting and redoing everything (also removing junk images) After fix of normalization I get .38 and .44

Using oxford descriptors I get .51ish Then changing to root-sift I smk-bow = get=0.56294936807700813 Then using tfidf-bow2=0.56046968275748565 asmk-gets 0.54146

Going down to 8K words smk-BOW gets .153 Going down to 8K words tfidf-BOW gets .128 Going down to 8K words smk-asmk gets 0.374

Ok the 65K vocab smk-asmk gets mAP=0.461... Ok, after recomputing a new 65K vocab with centered and root-sifted

> descriptors, using float32 precision (in most places), asmk gets a new map score of: mAP=.5275... :(This is with permissive query kpts and oxford vocab. Next step: ensure everything is float32. Ensured float32 mAP=.5279, ... better but indiciative of real error

After that try again at Jegou's data. Ensure there are no smk algo bugs. There must be one.

FINALLY! Got Jegou's data working. With jegou percmopute oxford feats, words, and assignments And float32 version asmk = .78415 bow = .545

asmk got 0.78415 with float32 version bow got .545 bot2 got .551

vecs07, root_sift, approx assign, (either jegou or my words) mAP=.673

Weird: vecs07, root_sift, exact assign, Maybe jegou words or maybe my words. Can't quite tell. Might have messed with a config. mAP=0.68487357885738664

October 8 Still using the same descriptors, but my own vocab with approx assign mAP = 0.78032

my own vocab approx assign, no center map = .793

The problem was minibatch params. Need higher batch size and init size. Needed to modify sklearn to handle this requirement.

Using my own descriptors I got 0.7460. Seems good.

Now, back to the HS pipeline. Getting a 0.638, so there is an inconsistency. Should be getting .7460. Maybe I gotta root_sift it up?

Turned off root_sift in script got .769, so there is a problem in system script minibatch 29566/270340... rate=0.86 Hz, eta=0:00:00, total=9:44:35, wall=05:24 EST inertia: mean batch=53730.923812, ewa=53853.439903 now need to try turning off float32

Differences Between this and SMK:

- No RootSIFT
- No SIFT Centering
- No Independent Vocab
- Chip RESIZE

Differences between this and VLAD

- residual vectors are normalized
- larger default vocabulary size

1.1.1.5.5.1 Feat Info

1.1.1.5.5.2 name | num_vecs | n_annots |

Oxford13 | 12,534,635 | | Oxford07 | 16,334,970 | | mine1 | 8,997,955 | | mine2 | 13,516,721 | 5063 | mine3 | 8,371,196 | 4728 | mine4 | 8,482,137 | 4783 |

1.1.1.5.5.3 Cluster Algo Config

name | algo | init | init_size | batch size | ========== minibatch1 | minibatch kmeans | kmeans++ | num_words * 4 | 100 | minibatch2 | minibatch kmeans | kmeans++ | num_words * 4 | 1000 | given13 | Lloyd? | kmeans++? | num_words * 8? | nan? |

1.1.1.5.5.4 Assign Algo Config

1.1.1.5.5.5 name | algo | trees | checks |

approx | kdtree | 8 | 1024 | exact | linear | nan | nan | exact | linear | nan | nan |

1.1.1.5.5.6 SMK Results

tagid | mAP | train_feats | test_feats | center | rootSIFT | assign | num_words | cluster methods | int | only_xy |

0.38 | mine1 | mine1 | | | approx | 64000 | minibatch1 | | |
0.541 | oxford07 | oxford07 | | X | approx | 2 ** 16 | minibatch1 | | X |
0.673 | oxford13 | oxford13 | X | X | approx | 2 ** 16 | minibatch1 | | X |
0.684 | oxford13 | oxford13 | X | X | exact | 2 ** 16 | minibatch1 | | X |
mybest | 0.793 | oxford13 | oxford13 | | X | approx | 2 ** 16 | minibatch2 | | X |
0.780 | oxford13 | oxford13 | X | X | approx | 2 ** 16 | minibatch2 | | X |
0.788 | paras13 | oxford13 | X | X | approx | 2 ** 16 | given13 | | X |
allgiven | 0.784 | paras13 | oxford13 | X | X | given13 | 2 ** 16 | given13 | | X |
reported13 | 0.781 | paras13 | oxford13 | X | X | given13 | 2 ** 16 | given13 | | X |

inhouse1 | 0.746 | mine2 | mine2 | | X | approx | 2 ** 16 | minibatch2 | | X | inhouse2 | 0.769 | mine2 | mine2 | | | approx | 2 ** 16 | minibatch2 | | X | inhouse3 | 0.769 | mine2 | mine2 | | | approx | 2 ** 16 | minibatch2 | X | X | inhouse4 | 0.751 | mine2 | mine2 | | | approx | 2 ** 16 | minibatch2 | X | | |

sysharn1 | 0.638 | mine3 | mine3 | | | approx | 64000 | minibatch2 | X | | sysharn2 | 0.713 | mine3 | mine4 | | | approx | 64000 | minibatch2 | X | |

In the SMK paper they report 0.781 as shown in the table, but they also report a score of 0.820 when increasing the number of features to from 12.5M to 19.2M by lowering feature detection thresholds.

class wbia.algo.smk.script_smk.**SMK**(*wx_to_weight*, *method='asmk'*, ***kwargs*)
Bases: utool.util_dev.NiceRepr

gamma(*X*)
Compute gamma of X

gamma(X) = (M(X, X)) ** (-1/2)

kernel_bow_tfidf(*X*, *Y*)

kernel_smk(*X*, *Y*)

match_score_agg(*X*, *Y*)

match_score_bow(*X*, *Y*)

match_score_sep(*X*, *Y*)

word_isect(*X*, *Y*)

class wbia.algo.smk.script_smk.**SparseVector**(*_dict*)
Bases: utool.util_dev.NiceRepr

dot(*other*)

wbia.algo.smk.script_smk.**bow_vector**(*X*, *wx_to_weight*, *nwords*)

wbia.algo.smk.script_smk.**check_image_sizes**(*data_uri_order*, *all_kpts*, *offset_list*)
Check if any keypoints go out of bounds wrt their associated images

wbia.algo.smk.script_smk.**compare_data**(*Y_list_*)

wbia.algo.smk.script_smk.**ensure_tf**(*X*)

wbia.algo.smk.script_smk.**get_annots_imgid**(*_annots*)

wbia.algo.smk.script_smk.**hyrule_vocab_test**()

wbia.algo.smk.script_smk.**kpts_inside_bbox**(*kpts*, *bbox*, *only_xy=False*)

wbia.algo.smk.script_smk.**load_internal_data**()

wbia TestResult –db Oxford -p smk:nWords=[64000],nAssign=[1],SV=[False],can_match_sameimg=True,dim_size=None -a oxford –dev-mode

wbia TestResult –db GZ_Master1 -p smk:nWords=[64000],nAssign=[1],SV=[False],fg_on=False -a ctrl:qmingt=2 –dev-mode

wbia.algo.smk.script_smk.**load_ordered_annots**(*data_uri_order*, *query_uri_order*)

wbia.algo.smk.script_smk.**load_oxford_2007**()
Loads data from http://www.robots.ox.ac.uk:5000/~vgg/publications/2007/Philbin07/philbin07.pdf

```
>>> from wbia.algo.smk.script_smk import *  # NOQA
```

wbia.algo.smk.script_smk.**load_oxford_2013**()
Found this data in README of SMK publication https://hal.inria.fr/hal-00864684/document http://people.rennes.inria.fr/Herve.Jegou/publications.html with download script

CommandLine: # Download oxford13 data cd ~/work/Oxford mkdir -p smk_data_iccv_2013 cd smk_data_iccv_2013 wget -nH –cut-dirs=4 -r -Pdata/ ftp://ftp.irisa.fr/local/texmex/corpus/iccv2013/

This dataset has 5063 images wheras 07 has 5062 This dataset seems to contain an extra junk image:

ashmolean_000214

Remember that matlab is 1 indexed! # DONT FORGET TO CONVERT TO 0 INDEXING!

wbia.algo.smk.script_smk.**load_oxford_wbia**()

wbia.algo.smk.script_smk.**make_agg_vecs**(*X*, *words*, *fx_to_vecs*)

wbia.algo.smk.script_smk.**make_temporary_annot**(*aid*, *vocab*, *wx_to_weight*, *ibs*, *config*)

wbia.algo.smk.script_smk.**new_external_annot**(*aid*, *fx_to_wxs*, *fx_to_maws*, *int_rvec*)

wbia.algo.smk.script_smk.**oxford_conic_test**()

wbia.algo.smk.script_smk.**run_asmk_script**()

wbia.algo.smk.script_smk.**sanity_checks**(*offset_list*, *Y_list*, *query_annots*, *ibs*)

wbia.algo.smk.script_smk.**show_data_image**(*data_uri_order*, *i*, *offset_list*, *all_kpts*, *all_vecs*)
i = 12

wbia.algo.smk.script_smk.**verify_score**()
Recompute all SMK things for two annotations and compare scores.

```
>>> from wbia.algo.smk.script_smk import *  # NOQA
```

cm.print_inspect_str(**qreq_**) cm.show_single_annotmatch(**qreq_**, daid1) cm.show_single_annotmatch(**qreq_**, daid2)

1.1.1.5.6 wbia.algo.smk.smk_funcs module

References

Jegou's Source Code, Data, and Publications http://people.rennes.inria.fr/Herve.Jegou/publications.html

To aggregate or not to aggregate: selective match kernels for image search https://hal.inria.fr/hal-00864684/document

Image search with selective match kernels: aggregation across single and multiple images http://image.ntua.gr/iva/files/Tolias_ijcv15_iasmk.pdf

Negative evidences and co-occurrences in image retrieval: the benefit of PCA and whitening https://hal.inria.fr/file/index/docid/722626/filename/jegou_chum_eccv2012.pdf

Revisiting the VLAD image representation https://hal.inria.fr/file/index/docid/850249/filename/nextvlad_hal.pdf

Aggregating local descriptors into a compact image representation https://lear.inrialpes.fr/pubs/2010/JDSP10/jegou_compactimagerepresentation.pdf

Large-scale image retrieval with compressed Fisher vectors http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.401.9140&rep=rep1&type=pdf

Improving Bag of Features http://lear.inrialpes.fr/pubs/2010/JDS10a/jegou_improvingbof_preprint.pdf

Lost in Quantization http://www.robots.ox.ac.uk/~vgg/publications/papers/philbin08.ps.gz

A Context Dissimilarity Measure for Accurate and Efficient Image Search https://lear.inrialpes.fr/pubs/2007/JHS07/jegou_cdm.pdf

Video Google: A text retrieval approach to object matching in videos http://www.robots.ox.ac.uk/~vgg/publications/papers/sivic03.pdf

Hamming embedding and weak geometric consistency for large scale image search https://lear.inrialpes.fr/pubs/2008/JDS08/jegou_hewgc08.pdf

Three things everyone should know to improve object retrieval https://www.robots.ox.ac.uk/~vgg/publications/2012/Arandjelovic12/arandjelovic12.pdf

Object retrieval with large vocabularies and fast spatial matching http://www.robots.ox.ac.uk:5000/~vgg/publications/2007/Philbin07/philbin07.pdf

Aggregating Local Descriptors into Compact Codes https://hal.inria.fr/file/index/docid/633013/filename/jegou_aggregate.pdf

Local visual query expansion https://hal.inria.fr/hal-00840721/PDF/RR-8325.pdf

Root SIFT technique https://hal.inria.fr/hal-00688169/document

Fisher Kernel For Large Scale Classification https://www.robots.ox.ac.uk/~vgg/rg/papers/peronnin_etal_ECCV10.pdf

Orientation covariant aggregation of local descriptors with embeddings https://arxiv.org/pdf/1407.2170.pdf

wbia.algo.smk.smk_funcs.**aggregate_rvecs**(*rvecs*, *maws*, *error_flags*)
Compute aggregated residual vectors Phi(X_c)

CommandLine: python -m wbia.algo.smk.smk_funcs aggregate_rvecs –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.smk_funcs import *  # NOQA
>>> vecs, words = ut.take(testdata_rvecs(), ['vecs', 'words'])
>>> word = words[-1]
>>> rvecs, error_flags = compute_rvec(vecs, word)
>>> maws = [1.0] * len(rvecs)
>>> agg_rvec, agg_flag = aggregate_rvecs(rvecs, maws, error_flags)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.qt4ensure()
>>> pt.figure()
>>> # recenter residuals for visualization
>>> agg_cvec = agg_rvec + word
>>> cvecs = (rvecs + word[None, :])
>>> pt.plot(word[0], word[1], 'r*', markersize=12, label='word')
>>> pt.plot(agg_cvec[0], agg_cvec[1], 'ro', label='re-centered agg_rvec')
>>> pt.plot(vecs.T[0], vecs.T[1], 'go', label='original vecs')
>>> pt.plot(cvecs.T[0], cvecs.T[1], 'b.', label='re-centered rvec')
>>> pt.draw_line_segments2([word] * len(cvecs), cvecs, alpha=.5, color='black')
>>> pt.draw_line_segments2([word], [agg_cvec], alpha=.5, color='red')
>>> pt.gca().set_aspect('equal')
>>> pt.legend()
>>> ut.show_if_requested()
```

wbia.algo.smk.smk_funcs.**assign_to_words**(*vocab*, *idx_to_vec*, *nAssign*, *massign_alpha=1.2*, *massign_sigma=80.0*, *massign_equal_weights=False*, *verbose=None*)

Assigns descriptor-vectors to nearest word.

Notes

Maybe move out of this file? The usage of vocab is out of this file scope.

Parameters

- **wordflann** (*FLANN*) – nearest neighbor index over words
- **words** (*ndarray*) – vocabulary words
- **idx_to_vec** (*ndarray*) – descriptors to assign
- **nAssign** (*int*) – number of words to assign each descriptor to
- **massign_alpha** (*float*) – multiple-assignment ratio threshold
- **massign_sigma** (*float*) – multiple-assignment gaussian variance
- **massign_equal_weights** (*bool*) – assign equal weight to all multiassigned words

Returns inverted index, multi-assigned weights, and forward index formated as: * wx_to_idxs - word index -> vector indexes * wx_to_maws - word index -> multi-assignment weights * idx2_wxs - vector index -> assigned word indexes

Return type tuple

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +SKIP
>>> idx_to_vec = depc.d.get_feat_vecs(aid_list)[0][0::300]
>>> idx_to_vec = np.vstack((idx_to_vec, vocab.wx_to_word[0]))
>>> nAssign = 2
>>> massign_equal_weights = False
>>> massign_alpha = 1.2
>>> massign_sigma = 80.0
>>> nAssign = 2
>>> idx_to_wxs, idx_to_maws = assign_to_words(vocab, idx_to_vec, nAssign)
>>> print('idx_to_maws = %s' % (ut.repr2(idx_to_wxs, precision=2),))
>>> print('idx_to_wxs = %s' % (ut.repr2(idx_to_maws, precision=2),))
```

wbia.algo.smk.smk_funcs.**build_matches_agg**(*X_fxs*, *Y_fxs*, *X_maws*, *Y_maws*, *score_list*)

Builds explicit features matches. Break and distribute up each aggregate score amongst its contributing features.

Returns (fm, fs)

Return type tuple

CommandLine: python -m wbia.algo.smk.smk_funcs build_matches_agg –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.smk_funcs import *  # NOQA
>>> map_int = ut.partial(ut.lmap, ut.partial(np.array, dtype=np.int32))
>>> map_float = ut.partial(ut.lmap, ut.partial(np.array, dtype=np.float32))
>>> X_fxs = map_int([[0, 1], [2, 3, 4], [5]])
>>> Y_fxs = map_int([[8], [0, 4], [99]])
>>> X_maws = map_float([[1, 1], [1, 1, 1], [1]])
>>> Y_maws = map_float([[1], [1, 1], [1]])
>>> score_list = np.array([1, 2, 3], dtype=np.float32)
>>> (fm, fs) = build_matches_agg(X_fxs, Y_fxs, X_maws, Y_maws, score_list)
>>> print('fm = ' + ut.repr2(fm))
>>> print('fs = ' + ut.repr2(fs))
>>> assert len(fm) == len(fs)
>>> assert score_list.sum() == fs.sum()
```

wbia.algo.smk.smk_funcs.**build_matches_sep**(*X_fxs*, *Y_fxs*, *scores_list*)

Just build matches. Scores have already been broken up. No need to do that.

Returns (fm, fs)

Return type tuple

CommandLine: python -m wbia.algo.smk.smk_funcs build_matches_agg –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.smk_funcs import *  # NOQA
>>> map_int = ut.partial(ut.lmap, ut.partial(np.array, dtype=np.int32))
```

(continues on next page)

(continued from previous page)

```
>>> map_float = ut.partial(ut.lmap, ut.partial(np.array, dtype=np.float32))
>>> X_fxs = map_int([[0, 1], [2, 3, 4], [5]])
>>> Y_fxs = map_int([[8], [0, 4], [99]])
>>> scores_list = map_float([
>>>     [[.1], [.2],],
>>>     [[.3, .4], [.4, .6], [.5, .9],],
>>>     [[.4]],
>>> ])
>>> (fm, fs) = build_matches_sep(X_fxs, Y_fxs, scores_list)
>>> print('fm = ' + ut.repr2(fm))
>>> print('fs = ' + ut.repr2(fs))
>>> assert len(fm) == len(fs)
>>> assert np.isclose(np.sum(ut.total_flatten(scores_list)), fs.sum())
```

wbia.algo.smk.smk_funcs.**cast_residual_integer**(*rvecs*)

quantize residual vectors to 8-bits using the same trunctation hack as in SIFT. values will typically not reach the maximum, so we can multiply by a higher number for better fidelity.

Parameters **rvecs** (*ndarray[float64_t]*) –

Returns

Return type ndarray[uint8_t]

CommandLine: python -m wbia.algo.smk.smk_funcs cast_residual_integer –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.smk_funcs import *  # NOQA
>>> rvecs = testdata_rvecs(dim=128)['rvecs'][4:]
>>> rvecs_int8 = cast_residual_integer(rvecs)
>>> rvecs_float = uncast_residual_integer(rvecs_int8)
>>> # Casting from float to int8 will result in a max quantization error
>>> measured_error = np.abs(rvecs_float - rvecs)
>>> # But there are limits on what this error can be
>>> cutoff = 127  # np.iinfo(np.int8).max
>>> fidelity = 255.0
>>> theory_error_in = 1 / fidelity
>>> theory_error_out = (fidelity - cutoff) / fidelity
>>> # Determine if any component values exceed the cutoff
>>> is_inside = (np.abs(rvecs * fidelity) < cutoff)
>>> # Check theoretical maximum for values inside and outside cutoff
>>> error_stats_in = ut.get_stats(measured_error[is_inside])
>>> error_stats_out = ut.get_stats(measured_error[~is_inside])
>>> print('inside cutoff error stats: ' + ut.repr4(error_stats_in, precision=8))
>>> print('outside cutoff error stats: ' + ut.repr4(error_stats_out, precision=8))
>>> assert rvecs_int8.dtype == np.int8
>>> assert np.all(measured_error[is_inside] < theory_error_in)
>>> assert np.all(measured_error[~is_inside] < theory_error_out)
```

wbia.algo.smk.smk_funcs.**compute_rvec**(*vecs*, *word*)

Compute residual vectors phi(x_c)

Subtract each vector from its quantized word to get the resiudal, then normalize residuals to unit length.

CommandLine: python -m wbia.algo.smk.smk_funcs compute_rvec –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.smk_funcs import *  # NOQA
>>> vecs, words = ut.take(testdata_rvecs(), ['vecs', 'words'])
>>> word = words[-1]
>>> rvecs, error_flags = compute_rvec(vecs, word)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.figure()
>>> # recenter residuals for visualization
>>> cvecs = (rvecs + word[None, :])
>>> pt.plot(word[0], word[1], 'r*', markersize=12, label='word')
>>> pt.plot(vecs.T[0], vecs.T[1], 'go', label='original vecs')
>>> pt.plot(cvecs.T[0], cvecs.T[1], 'b.', label='re-centered rvec')
>>> pt.draw_line_segments2(cvecs, [word] * len(cvecs), alpha=.5, color='black')
>>> pt.gca().set_aspect('equal')
>>> pt.legend()
>>> ut.show_if_requested()
```

wbia.algo.smk.smk_funcs.**compute_stacked_agg_rvecs**(*words*, *flat_wxs_assign*, *flat_vecs*, *flat_offsets*)

More efficient version of agg on a stacked structure

Parameters

- **words** (*ndarray*) – entire vocabulary of words
- **flat_wxs_assign** (*ndarray*) – maps a stacked index to word index
- **flat_vecs** (*ndarray*) – stacked SIFT descriptors
- **flat_offsets** (*ndarray*) – offset positions per annotation

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.smk.smk_funcs import *  # NOQA
>>> data = testdata_rvecs(dim=2, nvecs=1000, nannots=10)
>>> words = data['words']
>>> flat_offsets = data['offset_list']
>>> flat_wxs_assign, flat_vecs = ut.take(data, ['idx_to_wx', 'vecs'])
>>> tup = compute_stacked_agg_rvecs(words, flat_wxs_assign, flat_vecs, flat_
↪offsets)
>>> all_agg_vecs, all_error_flags, agg_offset_list = tup
>>> agg_rvecs_list = [all_agg_vecs[l:r] for l, r in ut.itertwo(agg_offset_list)]
>>> agg_flags_list = [all_error_flags[l:r] for l, r in ut.itertwo(agg_offset_
↪list)]
>>> assert len(agg_flags_list) == len(flat_offsets) - 1
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.smk.smk_funcs import *  # NOQA
>>> data = testdata_rvecs(dim=2, nvecs=100, nannots=5)
>>> words = data['words']
```

(continues on next page)

(continued from previous page)

```
>>> flat_offsets = data['offset_list']
>>> flat_wxs_assign, flat_vecs = ut.take(data, ['idx_to_wx', 'vecs'])
>>> tup = compute_stacked_agg_rvecs(words, flat_wxs_assign, flat_vecs, flat_
↪offsets)
>>> all_agg_vecs, all_error_flags, agg_offset_list = tup
>>> agg_rvecs_list = [all_agg_vecs[l:r] for l, r in ut.itertwo(agg_offset_list)]
>>> agg_flags_list = [all_error_flags[l:r] for l, r in ut.itertwo(agg_offset_
↪list)]
>>> assert len(agg_flags_list) == len(flat_offsets) - 1
```

wbia.algo.smk.smk_funcs.**gamma_agg**(*phisX*, *flagsX*, *weight_list*, *alpha*, *thresh*)

Computes gamma (self consistency criterion) It is a scalar which ensures K(X, X) = 1

Returns sccw self-consistency-criterion weight

Return type float

Math: gamma(X) = (sum_{c in C} w_c M(X_c, X_c))^{-.5}

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.smk.smk_pipeline import *  # NOQA
>>> ibs, smk, qreq_= testdata_smk()
>>> X = qreq_.qinva.grouped_annots[0]
>>> wx_to_weight = qreq_.wx_to_weight
>>> print('X.gamma = %r' % (gamma(X),))
```

wbia.algo.smk.smk_funcs.**gamma_sep**(*phisX_list*, *flagsX_list*, *weight_list*, *alpha*, *thresh*)

wbia.algo.smk.smk_funcs.**inv_doc_freq**(*ndocs_total*, *ndocs_per_word*)

Parameters

- **ndocs_total** (*int*) – numer of unique documents
- **ndocs_per_word** (*ndarray*) – ndocs_per_word[i] should correspond to the number of unique documents containing word[i]

Returns idf_per_word

Return type ndarrary

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.smk_funcs import *  # NOQA
>>> ndocs_total = 21
>>> ndocs_per_word = [0, 21, 20, 2, 15, 8, 12, 1, 2]
>>> idf_per_word = inv_doc_freq(ndocs_total, ndocs_per_word)
>>> result = '%s' % (ut.repr2(idf_per_word, precision=2),)
>>> print(result)
np.array([0.  , 0.  , 0.05, 2.35, 0.34, 0.97, 0.56, 3.04, 2.35])
```

wbia.algo.smk.smk_funcs.**invert_assigns**(*idx_to_wxs*, *idx_to_maws*, *verbose=False*)

Inverts assignment of vectors->to->words into words->to->vectors. Invert mapping – Group by word indexes

This gives a HUGE speedup over the old invert_assigns

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.smk_funcs import *  # NOQA
>>> idx_to_wxs = np.ma.array([
>>>     (0, 4),
>>>     (2, -1),
>>>     (2, 0)], dtype=np.int32)
>>> idx_to_wxs[1, 1] = np.ma.masked
>>> idx_to_maws = np.ma.array(
>>>     [(.5, 1.), (1., np.nan), (.5, .5)], dtype=np.float32)
>>> idx_to_maws[1, 1] = np.ma.masked
>>> tup = invert_assigns(idx_to_wxs, idx_to_maws)
>>> wx_to_idxs, wx_to_maws = tup
>>> result = 'wx_to_idxs = %s' % (ut.repr4(wx_to_idxs, with_dtype=True),)
>>> result += '\nwx_to_maws = %s' % (ut.repr4(wx_to_maws, with_dtype=True),)
>>> print(result)
wx_to_idxs = {
    0: np.array([0, 2], dtype=np.int32),
    2: np.array([1, 2], dtype=np.int32),
    4: np.array([0], dtype=np.int32),
}
wx_to_maws = {
    0: np.array([0.5, 0.5], dtype=np.float32),
    2: np.array([1. , 0.5], dtype=np.float32),
    4: np.array([1.], dtype=np.float32),
}
```

wbia.algo.smk.smk_funcs.**invert_assigns_old**(*idx_to_wxs*, *idx_to_maws*, *verbose=False*)

Inverts assignment of vectors to words into words to vectors.

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.smk_funcs import *  # NOQA
>>> idx_to_wxs = [
>>>     np.array([0, 4], dtype=np.int32),
>>>     np.array([2], dtype=np.int32),
>>>     np.array([2, 0], dtype=np.int32),
>>> ]
>>> idx_to_maws = [
>>>     np.array([ 0.5,  0.5], dtype=np.float32),
>>>     np.array([ 1.], dtype=np.float32),
>>>     np.array([ 0.5,  0.5], dtype=np.float32),
>>> ]
>>> wx_to_idxs, wx_to_maws = invert_assigns_old(idx_to_wxs, idx_to_maws)
>>> result = 'wx_to_idxs = %s' % (ut.repr4(wx_to_idxs, with_dtype=True),)
>>> result += '\nwx_to_maws = %s' % (ut.repr4(wx_to_maws, with_dtype=True),)
>>> print(result)
wx_to_idxs = {
    0: np.array([0, 2], dtype=np.int32),
    2: np.array([1, 2], dtype=np.int32),
    4: np.array([0], dtype=np.int32),
```

(continues on next page)

(continued from previous page)

```
}
wx_to_maws = {
    0: np.array([0.5, 0.5], dtype=np.float32),
    2: np.array([1. , 0.5], dtype=np.float32),
    4: np.array([0.5], dtype=np.float32),
}
```

wbia.algo.smk.smk_funcs.**invert_lists**(*aids*, *wx_lists*, *all_wxs=None*)
takes corresponding lists of (aids, wxs) and maps wxs to aids

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.smk_funcs import *  # NOQA
>>> aids = [1, 2, 3]
>>> wx_lists = [[0, 1], [20, 0, 1], [3]]
>>> wx_to_aids = invert_lists(aids, wx_lists)
>>> result = ('wx_to_aids = %s' % (ut.repr2(wx_to_aids),))
>>> print(result)
wx_to_aids = {0: [1, 2], 1: [1, 2], 3: [3], 20: [2]}
```

wbia.algo.smk.smk_funcs.**match_scores_agg**(*PhisX*, *PhisY*, *flagsX*, *flagsY*, *alpha*, *thresh*)
Scores matches to multiple words using aggregate residual vectors

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.smk_funcs import *  # NOQA
>>> PhisX = np.array([[ 0.        ,  0.        ],
>>>                   [-1.        ,  0.        ],
>>>                   [ 0.85085751,  0.52539652],
>>>                   [-0.89795083, -0.4400958 ],
>>>                   [-0.99934547,  0.03617512]])
>>> PhisY = np.array([[ 0.88299408, -0.46938411],
>>>                   [-0.12096522, -0.99265675],
>>>                   [-0.99948266, -0.03216222],
>>>                   [-0.08394916, -0.99647004],
>>>                   [-0.96414952, -0.26535957]])
>>> flagsX = np.array([True, False, False, True, False])[:, None]
>>> flagsY = np.array([False, False, False, True, False])[:, None]
>>> alpha = 3.0
>>> thresh = 0.0
>>> score_list = match_scores_agg(PhisX, PhisY, flagsX, flagsY, alpha, thresh)
>>> result = 'score_list = ' + ut.repr2(score_list, precision=4)
>>> print(result)
score_list = np.array([1.    , 0.0018, 0.    , 1.    , 0.868 ])
```

wbia.algo.smk.smk_funcs.**match_scores_sep**(*phisX_list*, *phisY_list*, *flagsX_list*, *flagsY_list*, *alpha*, *thresh*)
Scores matches to multiple words using lists of separeated residual vectors

wbia.algo.smk.smk_funcs.**sccw_normalize**(*scores*, *weight_list*)

wbia.algo.smk.smk_funcs.**selective_match_score**(*phisX*, *phisY*, *flagsX*, *flagsY*, *alpha*, *thresh*)

computes the score of each feature match

wbia.algo.smk.smk_funcs.**selectivity**(*u*, *alpha=3.0*, *thresh=0.0*, *out=None*)

The selectivity function thresholds and applies a power law.

This downweights weak matches. The following is the exact definition from SMK paper. sigma_alpha(u) = (sign(u) * (u ** alpha) if u > thresh else 0)

Parameters

- **u** (*ndarray*) – input score between (-1, +1)
- **alpha** (*float*) – power law (default = 3.0)
- **thresh** (*float*) – number between 0 and 1 (default = 0.0)
- **out** (*None*) – inplace output (default = None)

Returns score

Return type float

CommandLine:

python -m wbia.plottool plot_func –show –range=-1,1 –setup="import wbia" –func wbia.algo.smk.smk_funcs.selectivity "lambda u: sign(u) * abs(u)**3.0 * greater_equal(u, 0)"

python -m wbia.algo.smk.smk_funcs selectivity –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.smk_funcs import *  # NOQA
>>> u = np.array([-1.0, -.5, -.1, 0, .1, .5, 1.0])
>>> alpha = 3.0
>>> thresh = 0
>>> score = selectivity(u, alpha, thresh)
>>> result = ut.repr2(score.tolist(), precision=4)
>>> print(result)
[0.0000, 0.0000, 0.0000, 0.0000, 0.0010, 0.1250, 1.0000]
```

wbia.algo.smk.smk_funcs.**testdata_rvecs**(*dim=2*, *nvecs=13*, *nwords=5*, *nannots=4*)

two dimensional test data

CommandLine: python -m wbia.algo.smk.smk_funcs testdata_rvecs –show

Ignore: dim = 2 nvecs = 13 nwords = 5 nannots = 5

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.smk.smk_funcs import *  # NOQA
>>> data = testdata_rvecs()
>>> ut.quit_if_noshow()
>>> exec(ut.execstr_dict(data))
>>> import wbia.plottool as pt
>>> from scipy.spatial import Voronoi, voronoi_plot_2d
```

(continues on next page)

(continued from previous page)

```
>>> pt.qt4ensure()
>>> fig = pt.figure()
>>> vor = Voronoi(words)
>>> pt.plot(words.T[0], words.T[1], 'r*', label='words')
>>> pt.plot(vecs.T[0], vecs.T[1], 'b.', label='vecs')
>>> # lines showing assignments (and residuals)
>>> pts1 = vecs
>>> pts2 = words[idx_to_wx.T[0]]
>>> pt.draw_line_segments2(pts1, pts2)
>>> pt.plot(vecs.T[0], vecs.T[1], 'g.', label='vecs')
>>> voronoi_plot_2d(vor, show_vertices=False, ax=pt.gca())
>>> extents = vt.get_pointset_extents(np.vstack((vecs, words)))
>>> extents = vt.scale_extents(extents, 1.1)
>>> ax = pt.gca()
>>> ax.set_aspect('equal')
>>> ax.set_xlim(*extents[0:2])
>>> ax.set_ylim(*extents[2:4])
>>> ut.show_if_requested()
```

wbia.algo.smk.smk_funcs.**uncast_residual_integer**(*rvecs*)

Parameters **rvecs** (*ndarray[uint8_t]*) –

Returns

Return type ndarray[float64_t]

wbia.algo.smk.smk_funcs.**weight_multi_assigns**(*_idx_to_wx*, *_idx_to_wdist*, *massign_alpha=1.2*, *massign_sigma=80.0*, *massign_equal_weights=False*)

Multi Assignment Weight Filtering from Improving Bag of Features

Parameters **()** (*massign_equal_weights*) – Turns off soft weighting. Gives all assigned vectors weight 1

Returns (idx_to_wxs, idx_to_maws)

Return type tuple

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.smk_funcs import *  # NOQA
>>> _idx_to_wx = np.array([[0, 1], [2, 3], [4, 5], [2, 0]])
>>> _idx_to_wdist = np.array([[.1, .11], [.2, .25], [.03, .25], [0, 1]])
>>> massign_alpha = 1.2
>>> massign_sigma = 80.0
>>> massign_equal_weights = False
>>> idx_to_wxs, idx_to_maws = weight_multi_assigns(
>>>     _idx_to_wx, _idx_to_wdist, massign_alpha, massign_sigma,
>>>     massign_equal_weights)
>>> result = 'idx_to_wxs = %s' % (ut.repr2(idx_to_wxs.astype(np.float64)),)
>>> result += '\nidx_to_maws = %s' % (ut.repr2(idx_to_maws, precision=2),)
>>> print(result)
idx_to_wxs = np.ma.MaskedArray([[0., 1.],
                  [2., inf],
                  [4., inf],
```

(continues on next page)

(continued from previous page)

```
                    [2., 0.]])
idx_to_maws = np.ma.MaskedArray([[0.5, 0.5],
                    [1. , inf],
                    [1. , inf],
                    [0.5, 0.5]])
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.smk.smk_funcs import *  # NOQA
>>> _idx_to_wx = np.array([[0, 1], [2, 3], [4, 5], [2, 0]])
>>> _idx_to_wdist = np.array([[.1, .11], [.2, .25], [.03, .25], [0, 1]])
>>> _idx_to_wx = _idx_to_wx.astype(np.int32)
>>> _idx_to_wdist = _idx_to_wdist.astype(np.float32)
>>> massign_alpha = 1.2
>>> massign_sigma = 80.0
>>> massign_equal_weights = True
>>> idx_to_wxs, idx_to_maws = weight_multi_assigns(
>>>     _idx_to_wx, _idx_to_wdist, massign_alpha, massign_sigma,
>>>     massign_equal_weights)
>>> result = 'idx_to_wxs = %s' % (ut.repr2(idx_to_wxs.astype(np.float64)),)
>>> result += '\nidx_to_maws = %s' % (ut.repr2(idx_to_maws, precision=2),)
>>> print(result)
idx_to_wxs = np.ma.MaskedArray([[0., 1.],
                    [2., inf],
                    [4., inf],
                    [2., 0.]])
idx_to_maws = np.ma.MaskedArray([[1., 1.],
                    [1., inf],
                    [1., inf],
                    [1., 1.]])
```

1.1.1.5.7 wbia.algo.smk.smk_pipeline module

Oxford Experiment: wbia TestResult –db Oxford -p smk:nWords=[64000],nAssign=[1],SV=[False],can_match_sameimg=True -a oxford

Zebra Experiment:

python -m wbia draw_rank_cmc –db GZ_Master1 –show

-p :proot=smk,num_words=[64000],fg_on=False,nAssign=[1],SV=[False]
:proot=vsmany,fg_on=False,SV=[False]

-a ctrl:qmingt=2

python -m wbia draw_rank_cmc –db PZ_Master1 –show

-p :proot=smk,num_words=[64000],fg_on=False,nAssign=[1],SV=[False]
:proot=vsmany,fg_on=False,SV=[False]

-a ctrl:qmingt=2

class wbia.algo.smk.smk_pipeline.**MatchHeuristicsConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*

class wbia.algo.smk.smk_pipeline.**SMK**
Bases: utool.util_dev.NiceRepr

Harness class that controls the execution of the SMK algorithm

K(X, Y) = gamma(X) * gamma(Y) * sum([Mc(Xc, Yc) for c in words])

match_single(*qaid*, *daids*, *qreq_*, *verbose=True*)

CommandLine: python -m wbia.algo.smk.smk_pipeline SMK.match_single –profile python -m wbia.algo.smk.smk_pipeline SMK.match_single –show

python -m wbia SMK.match_single -a ctrl:qmingt=2 –profile –db PZ_Master1 python -m wbia SMK.match_single -a ctrl –profile –db GZ_ALL

Example

```
>>> # xdoctest: +REQUIRES(--slow)
>>> # FUTURE_ENABLE
>>> from wbia.algo.smk.smk_pipeline import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_(defaultdb='PZ_MTEST')
>>> ibs = qreq_.ibs
>>> daids = qreq_.daids
>>> #ibs, daids = wbia.testdata_aids(defaultdb='PZ_MTEST', default_set='dcfg')
>>> qreq_ = SMKRequest(ibs, daids[0:1], daids, {'agg': True,
>>>                                             'num_words': 1000,
>>>                                             'sv_on': True})
>>> qreq_.ensure_data()
>>> qaid = qreq_.qaids[0]
>>> daids = qreq_.daids
>>> daid = daids[1]
>>> verbose = True
>>> cm = qreq_.smk.match_single(qaid, daids, qreq_)
>>> ut.quit_if_noshow()
>>> ut.qtensure()
>>> cm.ishow_analysis(qreq_)
>>> ut.show_if_requested()
```

predict_matches(*qreq_*, *verbose=True*)

```
>>> from wbia.algo.smk.smk_pipeline import *  # NOQA
>>> ibs, smk, qreq_ = testdata_smk()
>>> verbose = True
```

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

class wbia.algo.smk.smk_pipeline.**SMKRequest**(*ibs=None*, *qaids=None*, *daids=None*, *config=None*)
Bases: *wbia.algo.smk.match_chips5.EstimatorRequest*

qreq_-like object. Trying to work on becoming more scikit-ish

CommandLine: python -m wbia.algo.smk.smk_pipeline SMKRequest –profile python -m wbia.algo.smk.smk_pipeline SMKRequest –show

python -m wbia draw_rank_cmc –db GZ_ALL –show -p :proot=smk,num_words=[64000,4000],nAssign=[1,5],sv_on=
-a ctrl:qmingt=2

python -m wbia draw_rank_cmc –db PZ_MTEST –show

-p :proot=smk,num_words=[64000,8000,4000],nAssign=[1,2,4],sv_on=[True,False]
default:proot=vsmany,sv_on=[True,False]

-a default:qmingt=2

python -m wbia draw_rank_cmc –db PZ_MTEST –show

-p :proot=smk,num_words=[64000],nAssign=[1],sv_on=[True] default:proot=vsmany,sv_on=[True]

-a default:qmingt=2

python -m wbia draw_rank_cmc –db PZ_Master1 –show -p :proot=smk,num_words=[64000],nAssign=[1],sv_on=[Fal
-a ctrl:qmingt=2

python -m wbia draw_rank_cmc –db PZ_Master1 -p :proot=smk,num_words=[64000],nAssign=[1],sv_on=[True]
-a ctrl:qmingt=2,qindex=60:80 –profile

python -m wbia draw_rank_cmc –db GZ_ALL -p :proot=smk,num_words=[64000],nAssign=[1],sv_on=[True]
-a ctrl:qmingt=2,qindex=40:60 –profile

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.smk.smk_pipeline import *  # NOQA
>>> import wbia
>>> ibs, aid_list = wbia.testdata_aids(defaultdb='PZ_MTEST')
>>> qaids = aid_list[0:2]
>>> daids = aid_list[:]
>>> config = {'nAssign': 2, 'num_words': 64000, 'sv_on': True}
>>> qreq_ = SMKRequest(ibs, qaids, daids, config)
>>> qreq_.ensure_data()
>>> cm_list = qreq_.execute()
>>> ut.quit_if_noshow()
>>> ut.qtensure()
>>> cm_list[0].ishow_analysis(qreq_, fnum=1, viz_name_score=False)
>>> cm_list[1].ishow_analysis(qreq_, fnum=2, viz_name_score=False)
>>> ut.show_if_requested()
```

dump_vectors()

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.smk.smk_pipeline import *  # NOQA
>>> import wbia
>>> ibs, aid_list = wbia.testdata_aids(defaultdb='PZ_MTEST', a=
↪'default:mingt=2,pername=2')
>>> qaids = aid_list[0:2]
>>> daids = aid_list[:]
>>> config = {'nAssign': 1, 'num_words': 8000,
>>>           'sv_on': True}
>>> qreq_ = SMKRequest(ibs, qaids, daids, config)
>>> qreq_.ensure_data()
```

ensure_data()

```
>>> import wbia
qreq_ = wbia.testdata_qreq_(
    defaultdb='Oxford', a='oxford',
    p='default:proot=smk,nAssign=1,num_words=64000,SV=False,can_match_
↪sameimg=True,dim_size=None')
```

execute_pipeline()

```
>>> from wbia.algo.smk.smk_pipeline import *  # NOQA
>>> ibs, smk, qreq_ = testdata_smk()
>>> cm_list = qreq_.execute()
```

get_qreq_dannot_kpts(*daids*)

get_qreq_qannot_kpts(*qaids*)

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

class wbia.algo.smk.smk_pipeline.**SMKRequestConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*

Figure out how to do this

wbia.algo.smk.smk_pipeline.**check_can_match**(*qaid*, *hit_daids*, *qreq_*)

wbia.algo.smk.smk_pipeline.**match_kernel_agg**(*X*, *Y*, *wx_to_weight*, *alpha*, *thresh*)

wbia.algo.smk.smk_pipeline.**match_kernel_sep**(*X*, *Y*, *wx_to_weight*, *alpha*, *thresh*)

wbia.algo.smk.smk_pipeline.**testdata_smk**(**args*, ***kwargs*)

```
>>> from wbia.algo.smk.smk_pipeline import *  # NOQA
>>> kwargs = {}
```

wbia.algo.smk.smk_pipeline.**word_isect**(*X*, *Y*, *wx_to_weight*)

1.1.1.5.8 wbia.algo.smk.vocab_indexer module

class wbia.algo.smk.vocab_indexer.**VisualVocab**(*words=None*)
Bases: utool.util_dev.NiceRepr

Class that maintains a list of visual words (cluster centers) Also maintains a nearest neighbor index structure for finding words. This class is build using the depcache

build(*verbose=True*)

nn_index(*idx_to_vec*, *nAssign*, *checks=None*)

```
>>> idx_to_vec = depc.d.get_feat_vecs(aid_list)[0]
>>> vocab = vocab
>>> nAssign = 1
```

render_vocab()
Renders the average patch of each word. This is a quick visualization of the entire vocabulary.

CommandLine: python -m wbia.algo.smk.vocab_indexer render_vocab –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.smk.vocab_indexer import *  # NOQA
>>> vocab = testdata_vocab('PZ_MTEST', num_words=64)
>>> all_words = vocab.render_vocab()
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.qt4ensure()
>>> pt.imshow(all_words)
>>> ut.show_if_requested()
```

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

shape

class wbia.algo.smk.vocab_indexer.**VocabConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*

wbia.algo.smk.vocab_indexer.**compute_vocab**(*depc*, *fid_list*, *config*)
Depcache method for computing a new visual vocab

CommandLine: python -m wbia.core_annots –exec-compute_neighbor_index –show python -m wbia show_depc_annot_table_input –show –tablename=neighbor_index

python -m wbia.algo.smk.vocab_indexer –exec-compute_vocab:0 python -m wbia.algo.smk.vocab_indexer –exec-compute_vocab:1

FIXME make util_tests register python -m wbia.algo.smk.vocab_indexer compute_vocab:0

Ignore:

```
>>> # Lev Oxford Debug Example
>>> import wbia
>>> ibs = wbia.opendb('Oxford')
>>> depc = ibs.depc
>>> table = depc['vocab']
>>> # Check what currently exists in vocab table
>>> table.print_configs()
>>> table.print_table()
>>> table.print_internal_info()
>>> # Grab aids used to compute vocab
>>> from wbia.expt.experiment_helpers import get_annotcfg_list
>>> expanded_aids_list = get_annotcfg_list(ibs, ['oxford'])[1]
>>> qaids, daids = expanded_aids_list[0]
>>> vocab_aids = daids
>>> config = {'num_words': 64000}
>>> exists = depc.check_rowids('vocab', [vocab_aids], config=config)
>>> print('exists = %r' % (exists,))
>>> vocab_rowid = depc.get_rowids('vocab', [vocab_aids], config=config)[0]
>>> print('vocab_rowid = %r' % (vocab_rowid,))
>>> vocab = table.get_row_data([vocab_rowid], 'words')[0]
>>> print('vocab = %r' % (vocab,))
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.smk.vocab_indexer import *  # NOQA
>>> # Test depcache access
>>> import wbia
>>> ibs, aid_list = wbia.testdata_aids('testdb1')
>>> depc = ibs.depc_annot
>>> input_tuple = [aid_list]
>>> rowid_kw = {}
>>> tablename = 'vocab'
>>> vocabid_list = depc.get_rowids(tablename, input_tuple, **rowid_kw)
>>> vocab = depc.get(tablename, input_tuple, 'words')[0]
>>> assert vocab.wordflann is not None
>>> assert vocab.wordflann._FLANN__curindex_data is not None
>>> assert vocab.wordflann._FLANN__curindex_data is vocab.wx_to_word
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.smk.vocab_indexer import *  # NOQA
>>> import wbia
>>> ibs, aid_list = wbia.testdata_aids('testdb1')
>>> depc = ibs.depc_annot
>>> fid_list = depc.get_rowids('feat', aid_list)
>>> config = VocabConfig()
>>> vocab, train_vecs = ut.exec_func_src(compute_vocab, keys=['vocab', 'train_vecs
↪'])
>>> idx_to_vec = depc.d.get_feat_vecs(aid_list)[0]
>>> self = vocab
>>> ut.quit_if_noshow()
>>> data = train_vecs
>>> centroids = vocab.wx_to_word
>>> import wbia.plottool as pt
>>> vt.plot_centroids(data, centroids, num_pca_dims=2)
>>> ut.show_if_requested()
>>> #config = ibs.depc_annot['vocab'].configclass()
```

wbia.algo.smk.vocab_indexer.**testdata_vocab**(*defaultdb='testdb1'*, ***kwargs*)

```
>>> from wbia.algo.smk.vocab_indexer import *  # NOQA
>>> defaultdb='testdb1'
>>> kwargs = {'num_words': 1000}
```

1.1.1.5.9 Module contents

wbia.algo.smk.**IMPORT_TUPLES = [('match_chips5', None), ('smk_pipeline', None), ('vocab_inde**
cd /home/joncrall/code/wbia/wbia/algo/smk makeinit.py –modname=wbia.algo.smk

Type Regen Command

wbia.algo.smk.**reassign_submodule_attributes**(*verbose=True*)
why reloading all the modules doesnt do this I don't know

wbia.algo.smk.**reload_subs**(*verbose=True*)
Reloads wbia.algo.smk and submodules

wbia.algo.smk.**rrrr**(*verbose=True*)
Reloads wbia.algo.smk and submodules

1.1.1.6 wbia.algo.verif package

1.1.1.6.1 Subpackages

1.1.1.6.1.1 wbia.algo.verif.torch package

1.1.1.6.2 Submodules

1.1.1.6.1.3 wbia.algo.verif.torch.fit_harness module

class wbia.algo.verif.torch.fit_harness.**FitHarness**(*model*, *train_loader*, *vali_loader=None*, *test_loader=None*, *criterion='cross_entropy'*, *lr_scheduler='exp'*, *optimizer_cls='Adam'*, *class_weights=None*, *gpu_num=None*, *workdir=None*)

Bases: object

check_termination()

load_snapshot(*load_path*)

log(*msg*)

log_value(*key*, *value*, *n_iter*)

run()

save_snapshot()

train_batch(*input_batch*)
https://github.com/meetshah1995/pytorch-semseg/blob/master/train.py

train_epoch()

validation_batch(*input_batch*)

validation_epoch()

1.1.1.6.1.4 wbia.algo.verif.torch.gpu_util module

wbia.algo.verif.torch.gpu_util.**find_unused_gpu**(*min_memory=0*)
Finds GPU with the lowest memory usage by parsing output of nvidia-smi

python -c "from pysseg.util import gpu_util; print(gpu_util.find_unused_gpu())"

wbia.algo.verif.torch.gpu_util.**gpu_info**()
Parses nvidia-smi

wbia.algo.verif.torch.gpu_util.**have_gpu**(*min_memory=8000*)
Determine if we are on a machine with a good GPU

1.1.1.6.1.5 wbia.algo.verif.torch.lr_schedule module

class wbia.algo.verif.torch.lr_schedule.**Exponential**(*init_lr=0.001*, *decay_rate=0.01*, *lr_decay_epoch=100*)
Bases: object

Decay learning rate by a factor of *decay_rate* every *lr_decay_epoch* epochs.

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.torch.lr_schedule import *
>>> lr_scheduler = Exponential()
>>> rates = np.array([lr_scheduler(i) for i in range(6)])
>>> target = np.array([1E-3, 1E-3, 1E-5, 1E-5, 1E-7, 1E-7])
>>> assert all(list(np.isclose(target, rates)))
```

1.1.1.6.1.6 wbia.algo.verif.torch.models module

class wbia.algo.verif.torch.models.**Siamese**
Bases: torch.nn.modules.module.Module

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.siamese import *
>>> self = Siamese()
```

forward(*input1*, *input2*)
Compute a resnet50 vector for each input and look at the L2 distance between the vectors.

wbia.algo.verif.torch.models.**visualize**()

1.1.1.6.1.7 wbia.algo.verif.torch.netmath module

class wbia.algo.verif.torch.netmath.**ContrastiveLoss**(*margin=1.0*)
Bases: torch.nn.modules.module.Module

Contrastive loss function.

References

https://github.com/delijati/pytorch-siamese/blob/master/contrastive.py

LaTeX: $(y E)^2 + ((1 - y) max(m - E, 0)^2)$

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.siamese import *
>>> vecs1, vecs2, label = testdata_siam_desc()
>>> self = ContrastiveLoss()
>>> ut.exec_func_src(self.forward, globals())
>>> func = self.forward
>>> output = torch.nn.PairwiseDistance(p=2)(vecs1, vecs2)
>>> loss2x, dist_l2 = ut.exec_func_src(self.forward, globals(), globals(), keys=[
↪'loss2x', 'dist_l2'])
>>> ut.quit_if_noshow()
>>> loss2x, dist_l2, label = map(np.array, [loss, dist_l2, label])
>>> label = label.astype(np.bool)
>>> dist0_l2 = dist_l2[label]
>>> dist1_l2 = dist_l2[~label]
>>> loss0 = loss2x[label] / 2
>>> loss1 = loss2x[~label] / 2
>>> import wbia.plottool as pt
>>> pt.plot2(dist0_l2, loss0, 'x', color=pt.TRUE_BLUE, label='imposter_loss', y_
↪label='loss')
>>> pt.plot2(dist1_l2, loss1, 'x', color=pt.FALSE_RED, label='genuine_loss', y_
↪label='loss')
>>> pt.gca().set_xlabel('l2-dist')
>>> pt.legend()
>>> ut.show_if_requested()
```

forward(*output*, *label*, *weight=None*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class wbia.algo.verif.torch.netmath.**Criterions**

Bases: *wbia.algo.verif.torch.netmath.NetMathParams*

A collection of standard and custom loss criterion

class ContrastiveLoss(*margin=1.0*)

Bases: `torch.nn.modules.module.Module`

Contrastive loss function.

References

https://github.com/delijati/pytorch-siamese/blob/master/contrastive.py

LaTeX: $(y E)^2 + ((1 - y) max(m - E, 0)^2)$

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.siamese import *
>>> vecs1, vecs2, label = testdata_siam_desc()
>>> self = ContrastiveLoss()
>>> ut.exec_func_src(self.forward, globals())
>>> func = self.forward
>>> output = torch.nn.PairwiseDistance(p=2)(vecs1, vecs2)
>>> loss2x, dist_l2 = ut.exec_func_src(self.forward, globals(), globals(),␣
↪keys=['loss2x', 'dist_l2'])
>>> ut.quit_if_noshow()
>>> loss2x, dist_l2, label = map(np.array, [loss, dist_l2, label])
>>> label = label.astype(np.bool)
>>> dist0_l2 = dist_l2[label]
>>> dist1_l2 = dist_l2[~label]
>>> loss0 = loss2x[label] / 2
>>> loss1 = loss2x[~label] / 2
>>> import wbia.plottool as pt
>>> pt.plot2(dist0_l2, loss0, 'x', color=pt.TRUE_BLUE, label='imposter_loss',␣
↪y_label='loss')
>>> pt.plot2(dist1_l2, loss1, 'x', color=pt.FALSE_RED, label='genuine_loss',␣
↪y_label='loss')
>>> pt.gca().set_xlabel('l2-dist')
>>> pt.legend()
>>> ut.show_if_requested()
```

forward(*output*, *label*, *weight=None*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

static cross_entropy2d(*output*, *label*, *weight=None*, *size_average=True*)

https://github.com/ycszen/pytorch-seg/blob/master/loss.py

class wbia.algo.verif.torch.netmath.**LRSchedules**

Bases: *wbia.algo.verif.torch.netmath.NetMathParams*

A collection of standard and custom learning rate schedulers

static exp(*optimizer*, *epoch*, *init_lr=0.001*, *lr_decay_epoch=2*)

Decay learning rate by a factor of 0.1 every lr_decay_epoch epochs.

class wbia.algo.verif.torch.netmath.**Metrics**

Bases: *wbia.algo.verif.torch.netmath.NetMathParams*

static tpr(*output*, *label*)

true positive rate

class wbia.algo.verif.torch.netmath.**NetMathParams**

Bases: `object`

classmethod lookup(*key_or_scheduler*)

Accepts either a string that encodes a known scheduler or a custom callable that is returned as-is.

Parameters **key_or_scheduler** (*str or func*) – scheduler name or the func itself

class wbia.algo.verif.torch.netmath.**Optimizers**

Bases: *wbia.algo.verif.torch.netmath.NetMathParams*

class Adam(*params*, *lr=0.001*, *betas=(0.9, 0.999)*, *eps=1e-08*, *weight_decay=0*, *amsgrad=False*)

Bases: torch.optim.optimizer.Optimizer

Implements Adam algorithm.

---

input : γ (lr), β_1, β_2 (betas), θ_0 (params), $f(\theta)$ (objective)
λ (weight decay), $amsgrad$

initialize : $m_0 \leftarrow 0$ (first moment), $v_0 \leftarrow 0$ (second moment), $\widehat{v_0}^{max} \leftarrow 0$

---

for $t = 1$ **to** $\ldots$ **do**
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  if $\lambda \neq 0$
    $g_t \leftarrow g_t + \lambda\theta_{t-1}$
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
  $\widehat{m_t} \leftarrow m_t / \big(1 - \beta_1^t\big)$
  $\widehat{v_t} \leftarrow v_t / \big(1 - \beta_2^t\big)$
  if $amsgrad$
    $\widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t})$
    $\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / \big(\sqrt{\widehat{v_t}^{max}} + \epsilon\big)$
  else
    $\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / \big(\sqrt{\widehat{v_t}} + \epsilon\big)$

---

return θ_t

For further details regarding the algorithm we refer to Adam: A Method for Stochastic Optimization.

Parameters

- **params** (*iterable*) – iterable of parameters to optimize or dicts defining parameter groups
- **lr** (*float, optional*) – learning rate (default: 1e-3)
- **betas** (*Tuple[float, float], optional*) – coefficients used for computing running averages of gradient and its square (default: (0.9, 0.999))
- **eps** (*float, optional*) – term added to the denominator to improve numerical stability (default: 1e-8)
- **weight_decay** (*float, optional*) – weight decay (L2 penalty) (default: 0)
- **amsgrad** (*boolean, optional*) – whether to use the AMSGrad variant of this algorithm from the paper On the Convergence of Adam and Beyond (default: False)

step(*closure=None*)

Performs a single optimization step.

Parameters **closure** (*callable, optional*) – A closure that reevaluates the model and returns the loss.

class SGD (*params*, *lr=<required parameter>*, *momentum=0*, *dampening=0*, *weight_decay=0*, *nesterov=False*)

Bases: `torch.optim.optimizer.Optimizer`

Implements stochastic gradient descent (optionally with momentum).

---

input : γ (lr), θ_0 (params), $f(\theta)$ (objective), λ (weight decay),
μ (momentum), τ (dampening), $nesterov$

---

for $t = 1$ **to** $\ldots$ **do**
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  if $\lambda \neq 0$
    $g_t \leftarrow g_t + \lambda\theta_{t-1}$
  if $\mu \neq 0$
    if $t > 1$
      $\mathbf{b}_t \leftarrow \mu\mathbf{b}_{t-1} + (1-\tau)g_t$
    else
      $\mathbf{b}_t \leftarrow g_t$
    if $nesterov$
      $g_t \leftarrow g_{t-1} + \mu\mathbf{b}_t$
    else
      $g_t \leftarrow \mathbf{b}_t$
  $\theta_t \leftarrow \theta_{t-1} - \gamma g_t$

---

return θ_t

Nesterov momentum is based on the formula from On the importance of initialization and momentum in deep learning.

Parameters

- **params** (*iterable*) – iterable of parameters to optimize or dicts defining parameter groups
- **lr** (*float*) – learning rate
- **momentum** (*float, optional*) – momentum factor (default: 0)
- **weight_decay** (*float, optional*) – weight decay (L2 penalty) (default: 0)
- **dampening** (*float, optional*) – dampening for momentum (default: 0)
- **nesterov** (*bool, optional*) – enables Nesterov momentum (default: False)

Example

```
>>> optimizer = torch.optim.SGD(model.parameters(), lr=0.1, momentum=0.9)
>>> optimizer.zero_grad()
>>> loss_fn(model(input), target).backward()
>>> optimizer.step()
```

Note: The implementation of SGD with Momentum/Nesterov subtly differs from Sutskever et. al. and implementations in some other frameworks.

Considering the specific case of Momentum, the update can be written as

$$v_{t+1} = \mu * v_t + g_{t+1},$$
$$p_{t+1} = p_t - \text{lr} * v_{t+1},$$

where p, g, v and μ denote the parameters, gradient, velocity, and momentum respectively.

This is in contrast to Sutskever et. al. and other frameworks which employ an update of the form

$$v_{t+1} = \mu * v_t + \text{lr} * g_{t+1},$$
$$p_{t+1} = p_t - v_{t+1}.$$

The Nesterov version is analogously modified.

---

step(*closure=None*)
Performs a single optimization step.
Parameters **closure** (*callable, optional*) – A closure that reevaluates the model and returns the loss.

wbia.algo.verif.torch.netmath.**testdata_siam_desc**(*num_data=128*, *desc_dim=8*)

1.1.1.6.1.8 wbia.algo.verif.torch.old_harness module

1.1.1.6.1.9 wbia.algo.verif.torch.siamese module

1.1.1.6.1.10 wbia.algo.verif.torch.train_main module

class wbia.algo.verif.torch.train_main.**LRSchedule**
Bases: object

static exp(*optimizer*, *epoch*, *init_lr=0.001*, *lr_decay_epoch=2*)
Decay learning rate by a factor of 0.1 every lr_decay_epoch epochs.

class wbia.algo.verif.torch.train_main.**LabeledPairDataset**(*img1_fpaths*, *img2_fpaths*, *labels*, *transform='default'*)
Bases: torch.utils.data.dataset.Dataset

transform=transforms.Compose([

transforms.Scale(224), transforms.ToTensor(), torchvision.transforms.Normalize([0.5, 0.5, 0.5], [0.225, 0.225, 0.225])

]

Ignore:

```
>>> from wbia.algo.verif.torch.train_main import *
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> pblm = OneVsOneProblem.from_empty('PZ_MTEST')
>>> ibs = pblm.infr.ibs
>>> pblm.load_samples()
>>> samples = pblm.samples
>>> samples.print_info()
>>> xval_kw = pblm.xval_kw.asdict()
>>> skf_list = pblm.samples.stratified_kfold_indices(**xval_kw)
>>> train_idx, test_idx = skf_list[0]
```

(continues on next page)

(continued from previous page)

```
>>> aids1, aids2 = pblm.samples.aid_pairs[train_idx].T
>>> labels = pblm.samples['match_state'].y_enc[train_idx]
>>> labels = (labels == 1).astype(np.int64)
>>> chip_config = {'resize_dim': 'wh', 'dim_size': (224, 224)}
>>> img1_fpaths = ibs.depc_annot.get('chips', aids1, read_extern=False,␣
↪colnames='img', config=chip_config)
>>> img2_fpaths = ibs.depc_annot.get('chips', aids2, read_extern=False,␣
↪colnames='img', config=chip_config)
>>> self = LabeledPairDataset(img1_fpaths, img2_fpaths, labels)
>>> img1, img2, label = self[0]
```

class_weights()

wbia.algo.verif.torch.train_main.**siam_vsone_train**()

CommandLine: python -m wbia.algo.verif.torch.train_main siam_vsone_train

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.torch.train_main import *  # NOQA
>>> siam_vsone_train()
```

1.1.1.6.1.11 Module contents

1.1.1.6.2 Submodules

1.1.1.6.3 wbia.algo.verif.clf_helpers module

This module is a work in progress, as such concepts are subject to change.

MAIN IDEA: *MultiTaskSamples* serves as a structure to contain and manipulate a set of samples with potentially many different types of labels and features.

class wbia.algo.verif.clf_helpers.**ClfProblem**

Bases: utool.util_dev.NiceRepr

learn_deploy_classifiers(*task_keys=None*, *clf_key=None*, *data_key=None*)

Learns on data without any train/validation split

learn_evaluation_classifiers(*task_keys=None*, *clf_keys=None*, *data_keys=None*)

Evaluates by learning classifiers using cross validation. Do not use this to learn production classifiers.

python -m wbia.algo.verif.vsone evaluate_classifiers –db PZ_PB_RF_TRAIN –show

Example:

CommandLine: python -m clf_helpers learn_evaluation_classifiers

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.verif.clf_helpers import *  # NOQA
>>> pblm = IrisProblem()
>>> pblm.setup()
>>> pblm.verbose = True
>>> pblm.eval_clf_keys = ['Logit', 'RF']
>>> pblm.eval_task_keys = ['iris']
>>> pblm.eval_data_keys = ['learn(all)']
>>> result = pblm.learn_evaluation_classifiers()
>>> res = pblm.task_combo_res['iris']['Logit']['learn(all)']
>>> res.print_report()
>>> res = pblm.task_combo_res['iris']['RF']['learn(all)']
>>> res.print_report()
>>> print(result)
```

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

set_pandas_options()

set_pandas_options_low()

set_pandas_options_normal()

class wbia.algo.verif.clf_helpers.**ClfResult**
Bases: utool.util_dev.NiceRepr

Handles evaluation statistics for a multiclass classifier trained on a specific dataset with specific labels.

augment_if_needed()
Adds in dummy values for missing classes

classmethod combine_results(*res_list*, *labels=None*)
Combine results from cross validation runs into a single result representing the performance of the entire dataset

compress(*flags*)

confusions(*class_name*)

confusions_ovr()

extended_clf_report(*verbose=True*)

get_pos_threshes(*metric='fpr'*, *value=0.0001*, *maximize=False*, *warmup=200*, *priors=None*, *min_thresh=0.5*)
Finds a threshold that achieves the desired *value* for the desired metric, while maximizing or minimizing the threshold.

For positive classification you want to minimize the threshold. Priors can be passed in to augment probabilities depending on support. By default a class prior is 1 for threshold minimization and 0 for maximization.

get_thresholds(*metric='mcc'*, *value='maximize'*)
get_metric = 'thresholds' at_metric = metric = 'mcc' at_value = value = 'maximize'

a = [] b = [] for x in np.linspace(0, 1, 1000):

a += [cfms.get_metric_at_metric('thresholds', 'fpr', x, subindex=True)] b += [cfms.get_thresh_at_metric('fpr', x)]

a = np.array(a) b = np.array(b) d = (a - b) logger.info((d.min(), d.max()))

hardness_analysis(*samples*, *infr=None*, *method='argmax'*)
samples = pblm.samples

TODO MWE with sklearn data

ClfResult.make_single(ClfResult, clf, X_df, test_idx, labels, # data_key, feat_dims=None):

import sklearn.datasets iris = sklearn.datasets.load_iris()

TODO: make this setup simpler pblm = ClfProblem() task_key, clf_key, data_key = 'iris', 'RF', 'learn(all)' X_df = pd.DataFrame(iris.data, columns=iris.feature_names) samples = MultiTaskSamples(X_df.index) samples.apply_indicators({'iris': {name: iris.target == idx

for idx, name in enumerate(iris.target_names)}})

samples.X_dict = {'learn(all)': X_df}

pblm.samples = samples pblm.xval_kw['type'] = 'StratifiedKFold' clf_list, res_list = pblm._train_evaluation_clf(

task_key, data_key, clf_key)

labels = pblm.samples.subtasks[task_key] res = ClfResult.combine_results(res_list, labels)

res.get_thresholds('mcc', 'maximize')

predict_method = 'argmax'

index

ishow_roc()

classmethod make_single(*clf*, *X_df*, *test_idx*, *labels*, *data_key*, *feat_dims=None*)
Make a result for a single cross validiation subset

missing_classes()

print_report()

report_auto_thresholds(*threshes*, *verbose=True*)

report_thresholds(*warmup=200*)

roc_score()

roc_scores_ovr()

roc_scores_ovr_hat()

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

show_roc(*class_name*, ***kwargs*)

class wbia.algo.verif.clf_helpers.**IrisProblem**
Bases: *wbia.algo.verif.clf_helpers.ClfProblem*

Simple demo using the abstract clf problem to work on the iris dataset.

Example:

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.verif.clf_helpers import *  # NOQA
>>> pblm = IrisProblem()
>>> pblm.setup()
>>> pblm.samples
```

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

setup()

class wbia.algo.verif.clf_helpers.**MultiClassLabels**
Bases: utool.util_dev.NiceRepr

Used by samples to encode a single set of mutually exclusive labels. These can either be binary or multiclass.

import pandas as pd pd.options.display.max_rows = 10 # pd.options.display.max_rows = 20 pd.options.display.max_columns = 40 pd.options.display.width = 160

classmethod from_indicators(*indicator*, *index=None*, *task_name=None*)

gen_one_vs_rest_labels()

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.verif.clf_helpers import *  # NOQA
>>> indicator = ut.odict([
>>>         ('state1', [0, 0, 0, 1]),
>>>         ('state2', [0, 0, 1, 0]),
>>>         ('state3', [1, 1, 0, 0]),
>>>     ])
>>> labels = MultiClassLabels.from_indicators(indicator, task_name='task1')
>>> sublabels = list(labels.gen_one_vs_rest_labels())
>>> sublabel = sublabels[0]
```

has_support()

lookup_class_idx(*class_name*)

make_histogram()

one_vs_rest_task_names()

print_info()

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

target_type

y_bin

y_enc

class wbia.algo.verif.clf_helpers.**MultiTaskSamples**(*index*)
Bases: utool.util_dev.NiceRepr

Handles samples (i.e. feature-label pairs) with a combination of non-mutually exclusive subclassification labels

CommandLine: python -m wbia.algo.verif.clf_helpers MultiTaskSamples

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.verif.clf_helpers import *  # NOQA
>>> samples = MultiTaskSamples([0, 1, 2, 3])
>>> tasks_to_indicators = ut.odict([
>>>     ('task1', ut.odict([
```

(continues on next page)

(continued from previous page)

```
>>>          ('state1', [0, 0, 0, 1]),
>>>          ('state2', [0, 0, 1, 0]),
>>>          ('state3', [1, 1, 0, 0]),
>>>      ])),
>>>      ('task2', ut.odict([
>>>          ('state4', [0, 0, 0, 1]),
>>>          ('state5', [1, 1, 1, 0]),
>>>      ]))
>>> ])
>>> samples.apply_indicators(tasks_to_indicators)
```

apply_encoded_labels(*y_enc*, *class_names*, *task_name*)
Adds labels for a specific task. Alternative to *apply_indicators*

Parameters

- **y_enc** (*list*) – integer label indicating the class for each sample
- **class_names** (*list*) – list of strings indicating the class-domain
- **task_name** (*str*) – key for denoting this specific task

apply_indicators(*tasks_to_indicators*)
Adds labels for a specific task

Parameters **tasks_to_indicators** (*dict*) –

takes the form:

{

'**my_task_name1**' **{** 'class1': [list of bools indicating class membership] ... 'classN': [list of bools indicating class membership]

'my_task_nameN': ...

}

class_idx_basis_1d()
1d-index version of class_name_basis

class_idx_basis_2d()
2d-index version of class_name_basis

class_name_basis()
corresponds with indexes returned from encoded1d

encoded_1d()
Returns a unique label for each combination of samples

encoded_2d()

group_ids

items()

make_histogram()
label histogram

print_info()

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

stratified_kfold_indices(**xval_kw)
TODO: check xval label frequency

subsplit_indices(*subset_idx*, **xval_kw)
split an existing set

supported_tasks()

class wbia.algo.verif.clf_helpers.**XValConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

1.1.1.6.4 wbia.algo.verif.deploy module

class wbia.algo.verif.deploy.**Deployer**(*dpath='.'*, *pblm=None*)
Bases: object

Transforms a OneVsOne problem into a deployable model. Registers and loads published models.

deploy(*task_key=None*, *publish=False*)
Trains and saves a classifier for deployment

Notes

A deployment consists of the following information

- The classifier itself
- **Information needed to construct the input to the classifier**
  - TODO: can this be encoded as an sklearn pipeline?
- Metadata concerning what data the classifier was trained with
- PUBLISH TO /media/hdd/PUBLIC/models/pairclf

Example

```
>>> # xdoctest: +REQUIRES(module:wbia_cnn, --slow)
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> params = dict(sample_method='random')
>>> pblm = OneVsOneProblem.from_empty('PZ_MTEST', **params)
>>> pblm.setup(with_simple=False)
>>> task_key = pblm.primary_task_key
>>> self = Deployer(dpath='.', pblm=pblm)
>>> deploy_info = self.deploy()
```

Ignore: pblm.evaluate_classifiers(with_simple=False) res = pblm.task_combo_res[pblm.primary_task_key]['RF']['learn(s

ensure(*task_key*)

find_latest_local()

```
>>> self = Deployer()
>>> self.find_pretrained()
>>> self.find_latest_local()
```

find_latest_remote()
Used to update the published dict

CommandLine: python -m wbia.algo.verif.vsone find_latest_remote

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> self = Deployer()
>>> task_clf_names = self.find_latest_remote()
```

find_pretrained()

fname_fmtstr = 'vsone.{species}.{task_key}.{clf_key}.{n_dims}.{hashid}'

fname_parts = ['vsone', '{species}', '{task_key}', '{clf_key}', '{n_dims}', '{hashid}'

load_published(*ibs*, *species*)

meta_suffix = '.meta.json'

publish_info = {'path': '/data/public/models/pairclf', 'remote': 'cthulhu.dyn.wildme

published = {'giraffe_reticulated': {'match_state': 'vsone.giraffe_reticulated.match

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

1.1.1.6.5 wbia.algo.verif.oldvsone module

wbia.algo.verif.oldvsone.**demo_single_pairwise_feature_vector**()

CommandLine: python -m wbia.algo.verif.vsone demo_single_pairwise_feature_vector

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> match = demo_single_pairwise_feature_vector()
>>> print(match)
```

1.1.1.6.6 wbia.algo.verif.pairfeat module

class wbia.algo.verif.pairfeat.**MatchConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

class wbia.algo.verif.pairfeat.**PairFeatureConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

Config for building pairwise feature dimensions

I.E. Config to distil unordered feature correspondences into a fixed length vector.

class wbia.algo.verif.pairfeat.**PairwiseFeatureExtractor**(*ibs=None*, *config={}*, *use_cache=True*, *verbose=1*, *match_config=None*, *pairfeat_cfg=None*, *global_keys=None*, *need_lnbnn=None*, *feat_dims=None*)

Bases: object

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **match_config** (*dict*) – config for building feature correspondences
- **pairfeat_cfg** (*dict*) – config for making the pairwise feat vec
- **global_keys** (*list*) – global keys to use
- **need_lnbnn** (*bool*) – use LNBNN for enrichment
- **feat_dims** (*list*) – subset of feature dimensions (from pruning) if None, then all dimensions are used
- **use_cache** (*bool*) – turns on disk based caching (default = True)
- **verbose** (*int*) – verbosity flag (default = 1)

CommandLine: python -m wbia.algo.verif.pairfeat PairwiseFeatureExtractor

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.verif.pairfeat import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> extr = PairwiseFeatureExtractor(ibs)
>>> edges = [(1, 2), (2, 3)]
>>> X = extr.transform(edges)
>>> featinfo = vt.AnnotPairFeatInfo(X.columns)
>>> print(featinfo.get_infostr())
```

transform(*edges*)

Converts an annotation edge into their corresponding feature. By default this is a caching operation.

class wbia.algo.verif.pairfeat.**VsOneFeatConfig**(***kwargs*)

Bases: *wbia.dtool.base.Config*

keypoint params

class wbia.algo.verif.pairfeat.**VsOneMatchConfig**(***kwargs*)

Bases: *wbia.dtool.base.Config*

1.1.1.6.7 wbia.algo.verif.ranker module

TODO: rewrite the hotspotter lnbnn algo to be a generator

Wrapper around LNBNN hotspotter algorithm

class wbia.algo.verif.ranker.**Ranker**(*ibs=None*, *config={}*)
Bases: object

fit(*daids*, *dnids=None*)

predict(*qaids*, *qnids=None*, *prog_hook=None*)

1.1.1.6.8 wbia.algo.verif.sklearn_utils module

class wbia.algo.verif.sklearn_utils.**PrefitEstimatorEnsemble**(*clf_list*, *voting='soft'*, *weights=None*)
Bases: object

hacks around limitations of sklearn.ensemble.VotingClassifier

predict(*X*)
Predict class labels for X.

Parameters **X** ({*array-like, sparse matrix}, shape = [n_samples, n_features]*) – Training vectors, where n_samples is the number of samples and n_features is the number of features.

Returns **maj** – Predicted class labels.

Return type array-like, shape = [n_samples]

predict_proba(*X*)
Predict class probabilities for X in 'soft' voting

class wbia.algo.verif.sklearn_utils.**StratifiedGroupKFold**(*n_splits=3*, *shuffle=False*, *random_state=None*)
Bases: sklearn.model_selection._split._BaseKFold

Stratified K-Folds cross-validator with Grouping

Provides train/test indices to split data in train/test sets.

This cross-validation object is a variation of GroupKFold that returns stratified folds. The folds are made by preserving the percentage of samples for each class.

Parameters **n_splits** (*int, default=3*) – Number of folds. Must be at least 2.

split(*X*, *y*, *groups=None*)
Generate indices to split data into training and test set.

wbia.algo.verif.sklearn_utils.**classification_report2**(*y_true*, *y_pred*, *target_names=None*, *sample_weight=None*, *verbose=True*)

References

https://csem.flinders.edu.au/research/techreps/SIE07001.pdf https://www.mathworks.com/matlabcentral/fileexchange/5648-bm-cm-?requestedDomain=www.mathworks.com Jurman, Riccadonna, Furlanello, (2012). A Comparison of MCC and CEN

Error Measures in MultiClass Prediction

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.sklearn_utils import *  # NOQA
>>> y_true = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3]
>>> y_pred = [1, 2, 1, 3, 1, 2, 2, 3, 2, 2, 3, 3, 2, 3, 3, 3, 1, 3]
>>> target_names = None
>>> sample_weight = None
>>> verbose = True
>>> report = classification_report2(y_true, y_pred, verbose=verbose)
```

Ignore:

```
>>> size = 100
>>> rng = np.random.RandomState(0)
>>> p_classes = np.array([.90, .05, .05][0:2])
>>> p_classes = p_classes / p_classes.sum()
>>> p_wrong   = np.array([.03, .01, .02][0:2])
>>> y_true = testdata_ytrue(p_classes, p_wrong, size, rng)
>>> rs = []
>>> for x in range(17):
>>>     p_wrong += .05
>>>     y_pred = testdata_ypred(y_true, p_wrong, rng)
>>>     report = classification_report2(y_true, y_pred, verbose='hack')
>>>     rs.append(report)
>>> import wbia.plottool as pt
>>> pt.qtensure()
>>> df = pd.DataFrame(rs).drop(['raw'], axis=1)
>>> delta = df.subtract(df['target'], axis=0)
>>> sqrd_error = np.sqrt((delta ** 2).sum(axis=0))
>>> print('Error')
>>> print(sqrd_error.sort_values())
>>> ys = df.to_dict(orient='list')
>>> pt.multi_plot(ydata_list=ys)
```

wbia.algo.verif.sklearn_utils.**predict_from_probs**(*probs*, *method='argmax'*, *target_names=None*, ***kwargs*)

Predictions are returned as indices into columns or target_names

Doctest:

```
>>> from wbia.algo.verif.sklearn_utils import *
>>> rng = np.random.RandomState(0)
>>> probs = pd.DataFrame(rng.rand(10, 3), columns=['a', 'b', 'c'])
>>> pred1 = predict_from_probs(probs, 'argmax')
>>> pred2 = predict_from_probs(probs, 'argmax', target_names=probs.columns)
>>> threshes = probs.loc[0]
>>> pred3 = predict_from_probs(probs, threshes.values, force=True,
>>>                            target_names=probs.columns)
```

wbia.algo.verif.sklearn_utils.**predict_proba_df**(*clf*, *X_df*, *class_names=None*)

Calls sklearn classifier predict_proba but then puts results in a dataframe using the same index as X_df and incorporating all possible class_names given

wbia.algo.verif.sklearn_utils.**predict_with_thresh**(*probs*, *threshes*, *target_names=None*, *force=False*, *multi=True*, *return_flags=False*)

if force is true, everything will make a prediction, even if nothing passes the thresholds. In that case it will use argmax.

if more than one thing passes the thresold we take the highest one if multi=True, and return nan otherwise.

Doctest:

```
>>> from wbia.algo.verif.sklearn_utils import *
>>> probs = np.array([
>>>     [0.5, 0.5, 0.0],
>>>     [0.4, 0.5, 0.1],
>>>     [1.0, 0.0, 0.0],
>>>     [0.3, 0.3, 0.4],
>>>     [0.1, 0.3, 0.6],
>>>     [0.1, 0.6, 0.3],
>>>     [0.6, 0.1, 0.3],])
>>> threshes = [.5, .5, .5]
>>> pred_enc = predict_with_thresh(probs, threshes)
>>> a = predict_with_thresh(probs, [.5, .5, .5])
>>> b = predict_with_thresh(probs, [.5, .5, .5], force=True)
>>> assert np.isnan(a).sum() == 3
>>> assert np.isnan(b).sum() == 0
```

wbia.algo.verif.sklearn_utils.**temp**(*samples*)

wbia.algo.verif.sklearn_utils.**testdata_ypred**(*y_true*, *p_wrong*, *rng*)

wbia.algo.verif.sklearn_utils.**testdata_ytrue**(*p_classes*, *p_wrong*, *size*, *rng*)

wbia.algo.verif.sklearn_utils.**voting_ensemble**(*clf_list*, *voting='hard'*)
hack to construct a VotingClassifier from pretrained classifiers TODO: contribute similar functionality to sklearn

1.1.1.6.9 wbia.algo.verif.verifier module

class wbia.algo.verif.verifier.**BaseVerifier**
Bases: utool.util_dev.NiceRepr

easiness(*edges*, *real*)
Gets the probability of the class each edge is labeled as. Indicates how easy it is to classify this example.

fit(*edges*)
The vsone.OneVsOneProblem currently handles fitting a model based on edges. The actual fit call is in clf_helpers.py

predict(*edges*, *method='argmax'*, *encoded=False*)

predict_proba_df(*edges*)

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

class wbia.algo.verif.verifier.**IntraVerifier**(*pblm*, *task_key*, *clf_key*, *data_key*)
Bases: *wbia.algo.verif.verifier.BaseVerifier*

Predicts cross-validated intra-training sample probs.

Note: Requires the original OneVsOneProblem object. This classifier is for intra-dataset evaulation and is not meant to be pushlished for use on external datasets.

predict_proba_df(*want_edges*)

Predicts task probabilities in one of two ways:

(1) if the edge was in the training set then its cross-validated probability is returned.

(2) if the edge was not in the training set, then the average prediction over all cross validated classifiers are used.

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

class wbia.algo.verif.verifier.**Verifier**(*ibs=None*, *deploy_info=None*)
Bases: *wbia.algo.verif.verifier.BaseVerifier*

Notes

deploy_info should be a dict with the following keys: clf: sklearn classifier metadata: another dict with key:

class_names - classes that clf predicts task_key - str clf_key - str data_info - tuple of (feat_extract_config, feat_dims) # TODO: make feat dims part of feat_extract_config defaulted to None data_info - tuple of (feat_extract_config, feat_dims)

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('PZ_MTEST')
>>> speceis = 'zebra_plains'
>>> task_key = 'match_state'
>>> verif = Deployer()._load_published(ibs, species, task_key)
```

predict_proba_df(*edges*)

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

1.1.1.6.10 wbia.algo.verif.vsone module

CommandLine: # Test how well out-of-the-box vsone classifiers to: python -m wbia.algo.verif.vsone evaluate_classifiers –db DETECT_SEATURTLES

Train a classifier for deployment # Will output to the current working directory python -m wbia.algo.verif.vsone deploy –db GZ_Master1

class wbia.algo.verif.vsone.**AnnotPairSamples**(*ibs*, *aid_pairs*, *infr=None*, *apply=False*)
Bases: *wbia.algo.verif.clf_helpers.MultiTaskSamples*, ubelt.util_mixins.NiceRepr

Manages the different ways to assign samples (i.e. feat-label pairs) to 1-v-1 classification

CommandLine: python -m wbia.algo.verif.vsone AnnotPairSamples

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> pblm = OneVsOneProblem.from_empty()
>>> pblm.load_samples()
>>> samples = AnnotPairSamples(pblm.ibs, pblm.raw_simple_scores, {})
>>> print(samples)
>>> samples.print_info()
>>> print(samples.sample_hashid())
>>> encode_index = samples.subtasks['match_state'].encoded_df.index
>>> indica_index = samples.subtasks['match_state'].indicator_df.index
>>> assert np.all(samples.index == encode_index)
>>> assert np.all(samples.index == indica_index)
```

apply_multi_task_binary_label()

apply_multi_task_multi_label()

apply_single_task_multi_label()

compress(*flags*)

edge_set_hashid()
Faster than using ut.combine_uuids, because we condense and don't bother casting back to UUIDS, and we just directly hash.

group_ids
Prevents samples with the same group-id from appearing in the same cross validation fold. For us this means any pair within the same name or between the same names will have the same groupid.

is_comparable()

is_photobomb()

is_same()

print_featinfo()

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

sample_hashid()

set_feats(*X_dict*)

set_simple_scores(*simple_scores*)

task_label_hashid(*task_key*)

task_sample_hashid(*task_key*)

class wbia.algo.verif.vsone.**OneVsOneProblem**(*infr=None*, *verbose=None*, ***params*)
Bases: *wbia.algo.verif.clf_helpers.ClfProblem*

Keeps information about the one-vs-one pairwise classification problem

CommandLine: python -m wbia.algo.verif.vsone evaluate_classifiers python -m wbia.algo.verif.vsone evaluate_classifiers –db PZ_PB_RF_TRAIN python -m wbia.algo.verif.vsone evaluate_classifiers –db PZ_PB_RF_TRAIN –profile python -m wbia.algo.verif.vsone evaluate_classifiers –db PZ_MTEST –show python -m wbia.algo.verif.vsone evaluate_classifiers –db PZ_Master1 –show python -m wbia.algo.verif.vsone evaluate_classifiers –db GZ_Master1 –show python -m wbia.algo.verif.vsone evaluate_classifiers –db RotanTurtles –show

python -m wbia.algo.verif.vsone evaluate_classifiers –db testdb1 –show -a default

Example

```
>>> # xdoctest: +REQUIRES(module:wbia_cnn, --slow)
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> pblm = OneVsOneProblem.from_empty('PZ_MTEST')
>>> pblm.hyper_params['xval_kw']['n_splits'] = 10
>>> assert pblm.xval_kw.n_splits == 10
>>> pblm.xval_kw.n_splits = 5
>>> assert pblm.hyper_params['xval_kw']['n_splits'] == 5
>>> pblm.load_samples()
>>> pblm.load_features()
```

appname = 'vsone_rf_train'

auto_decisions_at_threshold(*primary_task*, *task_probs*, *task_thresh*, *task_keys*, *clf_key*, *data_key*)

build_feature_subsets()

Try to identify a useful subset of features to reduce problem dimensionality

CommandLine: python -m wbia.algo.verif.vsone build_feature_subsets –db GZ_Master1 python -m wbia.algo.verif.vsone build_feature_subsets –db PZ_PB_RF_TRAIN

python -m wbia Chap4._setup_pblm –db GZ_Master1 –eval python -m wbia Chap4._setup_pblm –db PZ_Master1 –eval

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> pblm = OneVsOneProblem.from_empty('PZ_MTEST')
>>> pblm.load_samples()
>>> pblm.load_features()
>>> pblm.build_feature_subsets()
>>> pblm.samples.print_featinfo()
```

deploy(*dpath='.'*, *task_key=None*, *publish=False*)

Trains and saves a classifier for deployment

Parameters

- **dpath** (*str*) – where to save the deployable model
- **task_key** (*str*) – task to train for (default match_state)
- **publish** (*bool*) – if True will try to rsync the model and metadata to the publication server.

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> pblm = OneVsOneProblem.from_empty(defaultdb='PZ_MTEST',
```

(continues on next page)

(continued from previous page)

```
>>>                                     sample_method='random')
>>> task_key = ut.get_argval('--task', default='match_state')
>>> publish = ut.get_argflag('--publish')
>>> pblm.deploy(task_key=task_key, publish=publish)
```

Notes

A deployment consists of the following information

- The classifier itself
- **Information needed to construct the input to the classifier**
  - TODO: can this be encoded as an sklearn pipeline?
- Metadata concerning what data the classifier was trained with
- PUBLISH TO /media/hdd/PUBLIC/models/pairclf

Ignore: pblm.evaluate_classifiers(with_simple=False) res = pblm.task_combo_res[pblm.primary_task_key]['RF']['learn(s

deploy_all(*dpath='.'*, *publish=False*)

ensure_deploy_classifiers(*dpath='.'*)

evaluate_classifiers(*with_simple=False*)

CommandLine: python -m wbia.algo.verif.vsone evaluate_classifiers python -m wbia.algo.verif.vsone evaluate_classifiers –db PZ_MTEST python -m wbia.algo.verif.vsone evaluate_classifiers –db GZ_Master1 python -m wbia.algo.verif.vsone evaluate_classifiers –db GIRM_Master1

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> pblm = OneVsOneProblem.from_empty(defaultdb='PZ_MTEST',
>>>                                   sample_method='random')
>>> #pblm.default_clf_key = 'Logit'
>>> pblm.default_clf_key = 'RF'
>>> pblm.evaluate_classifiers()
```

evaluate_simple_scores(*task_keys=None*)

```
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> pblm = OneVsOneProblem.from_empty()
>>> pblm.set_pandas_options()
>>> pblm.load_samples()
>>> pblm.load_features()
>>> pblm.evaluate_simple_scores()
```

extra_report(*task_probs*, *is_auto*, *want_samples*)

feature_importance(*task_key=None*, *clf_key=None*, *data_key=None*)

CommandLine: python -m wbia.algo.verif.vsone report_importance –show python -m wbia.algo.verif.vsone report_importance –show –db PZ_PB_RF_TRAIN

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> pblm = OneVsOneProblem.from_empty('GZ_Master1')
>>> data_key = pblm.default_data_key
>>> clf_key = pblm.default_clf_key
>>> task_key = pblm.primary_task_key
>>> pblm.setup_evaluation()
>>> featinfo = pblm.feature_info(task_key, clf_key, data_key)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> text = importances
>>> pt.wordcloud(featinfo.importances)
>>> ut.show_if_requested()
```

classmethod from_aids(*ibs*, *aids*, *verbose=None*, ***params*)
Constructs a OneVsOneProblem from a subset of aids. Use *pblm.load_samples* to sample a set of pairs

classmethod from_empty(*defaultdb=None*, ***params*)

```
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> defaultdb = 'GIRM_Master1'
>>> pblm = OneVsOneProblem.from_empty(defaultdb)
```

classmethod from_labeled_aidpairs(*ibs*, *labeled_aid_pairs*, *class_names*, *task_name*, ***params*)
Build a OneVsOneProblem directly from a set of aid pairs. It is not necessary to call *pblm.load_samples*.

Parameters

- **ibs** (IBEISController) –
- **labeled_aid_pairs** (*list*) – tuples of (aid1, aid2, int_label)
- **class_names** (*list*) – list of names corresponding to integer labels
- **task_name** (*str*) – identifier for the task (e.g. custom_match_state)

load_features(*use_cache=True*, *with_simple=False*)

CommandLine: python -m wbia.algo.verif.vsone load_features –profile

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> #pblm = OneVsOneProblem.from_empty('GZ_Master1')
>>> pblm = OneVsOneProblem.from_empty('PZ_PB_RF_TRAIN')
>>> pblm.load_samples()
>>> pblm.load_features(with_simple=False)
```

load_samples()

CommandLine: python -m wbia.algo.verif.vsone load_samples –profile

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> #pblm = OneVsOneProblem.from_empty('PZ_MTEST')
>>> #pblm = OneVsOneProblem.from_empty('PZ_PB_RF_TRAIN')
>>> pblm = OneVsOneProblem.from_empty('PZ_Master1')
>>> pblm.load_samples()
>>> samples = pblm.samples
>>> samples.print_info()
```

load_simple_scores()

make_graph_based_bootstrap_pairs()

Sampling method for when you want to bootstrap VAMP after several reviews.

Sample pairs for VAMP training using manually reviewed edges and mines other (random) pairs as needed.

We first sample a base set via:

(1) take all manually reviewed positive edges (not in an inconsistent PCC)

(2) take all manually reviewed negative edges (not touching an inconsistent PCC)

(3) take all manually reviewed incomparable edges. Note: it is important to ignore any PCC currently in an inconsistent state.

We can then generate additional positive samples by sampling automatically reviewed positive edges within PCCs.

We can do the same for negatives.

make_lnbnn_training_pairs()

make_randomized_training_pairs()

Randomized sample that does not require LNBNN

make_training_pairs()

CommandLine: python -m wbia.algo.verif.vsone make_training_pairs –db PZ_Master1

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> pblm = OneVsOneProblem.from_empty('PZ_MTEST')
>>> pblm.make_training_pairs()
```

prune_features()

References

http://blog.datadive.net/selecting-good-features-part-iii-random-forests/ http://alexperrier.github.io/jekyll/update/2015/08/27/feature-importance-random-forests-gini-accuracy.html https://arxiv.org/abs/1407.7502 https://github.com/glouppe/phd-thesis

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> pblm = OneVsOneProblem.from_empty(defaultdb='PZ_MTEST')
>>> pblm = OneVsOneProblem.from_empty(defaultdb='PZ_PB_RF_TRAIN')
>>> pblm = OneVsOneProblem.from_empty(defaultdb='PZ_Master1')
```

Ignore:

```
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> pblm = OneVsOneProblem.from_empty(defaultdb='GZ_Master1')
>>> pblm.setup_evaluation()
```

qt_review_hardcases()

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> pblm = OneVsOneProblem.from_empty('PZ_Master1')
>>> #pblm = OneVsOneProblem.from_empty('GIRM_Master1')
>>> #pblm = OneVsOneProblem.from_empty('PZ_PB_RF_TRAIN')
>>> pblm.evaluate_classifiers()
>>> win = pblm.qt_review_hardcases()
```

Ignore:

```
>>> from wbia.scripts.postdoc import *
>>> self = VerifierExpt('RotanTurtles')
>>> self = VerifierExpt('humpbacks_fb')
>>> import wbia
>>> self._precollect()
>>> ibs = self.ibs
>>> aids = self.aids_pool
>>> pblm = vsone.OneVsOneProblem.from_aids(ibs, aids)
>>> infr = pblm.infr
>>> infr.params['algo.hardcase'] = True
>>> infr.params['autoreview.enabled'] = False
>>> infr.params['redun.enabled'] = False
>>> infr.params['ranking.enabled'] = False
>>> win = infr.qt_review_loop()
```

```
>>> pblm.eval_data_keys = [pblm.default_data_key]
>>> pblm.eval_clf_keys = [pblm.default_clf_key]
>>> pblm.evaluate_classifiers()
```

Ignore:

```
>>> # TEST to ensure we can priorizite reviewed edges without inference
>>> import networkx as nx
>>> from wbia.algo.graph import demo
>>> kwargs = dict(num_pccs=6, p_incon=.4, size_std=2)
>>> infr = demo.demodata_infr(**kwargs)
```

(continues on next page)

(continued from previous page)

```
>>> infr.params['redun.pos'] = 1
>>> infr.params['redun.neg'] = 1
>>> infr.apply_nondynamic_update()
>>> edges = list(infr.edges())
>>> prob_match = ut.dzip(edges, infr.dummy_matcher.predict(edges))
>>> infr.set_edge_attrs('prob_match', prob_match)
>>> infr.params['redun.enabled'] = True
>>> infr.prioritize('prob_match', edges)
>>> order = []
>>> while True:
>>>     order.append(infr.pop())
>>> print(len(order))
```

report_classifier_importance2(*clf*, *data_key=None*)

report_evaluation()

CommandLine: python -m wbia.algo.verif.vsone report_evaluation –db PZ_MTEST

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.verif.vsone import *  # NOQA
>>> pblm = OneVsOneProblem.from_empty(defaultdb='PZ_MTEST',
>>>                                   sample_method='random')
>>> pblm.eval_clf_keys = ['MLP', 'Logit', 'RF']
>>> pblm.eval_data_keys = ['learn(sum,glob)']
>>> pblm.setup_evaluation(with_simple=False)
>>> pblm.report_evaluation()
```

report_importance(*task_key*, *clf_key*, *data_key*)

report_simple_scores(*task_key=None*)

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

setup(*with_simple=False*)

setup_evaluation(*with_simple=False*)

task_evaluation_report(*task_key*)
clf_keys = [pblm.default_clf_key]

class wbia.algo.verif.vsone.**PairSampleConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*

1.1.1.6.11 Module contents

wbia.algo.verif.**IMPORT_TUPLES = [('clf_helpers', None), ('sklearn_utils', None), ('vsone',**
cd /home/joncrall/code/wbia/wbia/algo/verif makeinit.py –modname=wbia.algo.verif

Type Regen Command

wbia.algo.verif.**reassign_submodule_attributes**(*verbose=1*)
Updates attributes in the __init__ modules with updated attributes in the submodules.

wbia.algo.verif.**reload_subs**(*verbose=1*)
Reloads wbia.algo.verif and submodules

wbia.algo.verif.**rrrr**(*verbose=1*)
Reloads wbia.algo.verif and submodules

1.1.2 Submodules

1.1.3 wbia.algo.Config module

DEPRICATE FOR CORE ANNOT AND CORE IMAGE DEFS

class wbia.algo.Config.**AggregateConfig**(***kwargs*)
Bases: utool.Preferences.Pref

Old Agg Cfg

get_cfgstr(***kwargs*)

get_cfgstr_list(***kwargs*)

get_config_name(***kwargs*)
the user might want to overwrite this function

initialize_params()
Initializes config class attributes based on params info list

keys(***kwargs*)

lookup_paraminfo(*key*)

meta_get_cfgstr_list(*ignore_keys=None*, ***kwargs*)
default get_cfgstr_list, can be overrided by a config object

parse_items(***kwargs*)

class wbia.algo.Config.**ChipConfig**(***kwargs*)
Bases: utool.Preferences.Pref

get_cfgstr(***kwargs*)

get_cfgstr_list(*ignore_keys=None*, ***kwargs*)
default get_cfgstr_list, can be overrided by a config object

get_config_name(***kwargs*)
the user might want to overwrite this function

get_param_info_list()

initialize_params()
Initializes config class attributes based on params info list

keys(***kwargs*)

lookup_paraminfo(*key*)

parse_items(***kwargs*)

class wbia.algo.Config.**ConfigMetaclass**
Bases: type

Defines extra methods for Configs

class wbia.algo.Config.**DetectionConfig**(***kwargs*)
Bases: utool.Preferences.Pref

CommandLine: python -m wbia.algo.Config –test-DetectionConfig

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.Config import *  # NOQA
>>> detect_cfg = DetectionConfig()
>>> result = (detect_cfg.get_cfgstr())
>>> print(result)
_DETECT(cnn,____,sz=800)
```

get_cfgstr(***kwargs*)

get_cfgstr_list()

get_config_name(***kwargs*)
the user might want to overwrite this function

initialize_params()
Initializes config class attributes based on params info list

keys(***kwargs*)

lookup_paraminfo(*key*)

meta_get_cfgstr_list(*ignore_keys=None*, ***kwargs*)
default get_cfgstr_list, can be overrided by a config object

parse_items(***kwargs*)

class wbia.algo.Config.**DisplayConfig**(***kwargs*)
Bases: utool.Preferences.Pref

get_cfgstr(***kwargs*)

get_cfgstr_list()

get_config_name(***kwargs*)
the user might want to overwrite this function

initialize_params()
Initializes config class attributes based on params info list

keys(***kwargs*)

lookup_paraminfo(*key*)

meta_get_cfgstr_list(*ignore_keys=None*, ***kwargs*)
default get_cfgstr_list, can be overrided by a config object

parse_items(***kwargs*)

class wbia.algo.Config.**FeatureConfig**(***kwargs*)
Bases: utool.Preferences.Pref

Feature configuration object.

TODO depcirate for core_annots.FeatConfig

CommandLine: python -m wbia.algo.Config –test-FeatureConfig

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo import Config  # NOQA
>>> from wbia.algo.Config import *  # NOQA
>>> feat_cfg = Config.FeatureConfig()
>>> result = (feat_cfg.get_cfgstr())
>>> print(result)
>>> #assert result.startswith('_FEAT(hesaff+sift_)_CHIP')
_Feat(hesaff+sift)
```

get_cfgstr(***kwargs*)

get_cfgstr_list(*ignore_keys=None*, ***kwargs*)
default get_cfgstr_list, can be overrided by a config object

get_config_name()

get_hesaff_params()

get_param_info_list()

initialize_params()
Initializes config class attributes based on params info list

keys(***kwargs*)

lookup_paraminfo(*key*)

meta_get_config_name(***kwargs*)
the user might want to overwrite this function

parse_items(***kwargs*)

class wbia.algo.Config.**FeatureWeightConfig**(***kwargs*)
Bases: utool.Preferences.Pref

CommandLine: python -m wbia.algo.Config –exec-FeatureWeightConfig

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.Config import *  # NOQA
>>> featweight_cfg = FeatureWeightConfig(fw_detector='rf',
>>>                                      featweight_enabled=True)
>>> result = featweight_cfg.get_cfgstr()
>>> print(result)
```

_FEATWEIGHT(ON,uselabel,rf)_FEAT(**hesaff+sift_**)_CHIP(sz450)

_FEATWEIGHT(OFF)_FEAT(**hesaff+sift_**)_CHIP(sz450)

get_cfgstr(***kwargs*)

get_cfgstr_list(*ignore_keys=None*, ***kwargs*)
default get_cfgstr_list, can be overrided by a config object

get_config_name(***kwargs*)
the user might want to overwrite this function

get_param_info_list()

initialize_params()
Initializes config class attributes based on params info list

keys(***kwargs*)

lookup_paraminfo(*key*)

parse_items(***kwargs*)

class wbia.algo.Config.**FlannConfig**(***kwargs*)
Bases: utool.Preferences.Pref

this flann is only for neareset neighbors in vsone/many TODO: this might not need to be here, should be part of neighbor config

References

http://www.cs.ubc.ca/research/flann/uploads/FLANN/flann_pami2014.pdf http://www.cs.ubc.ca/research/flann/uploads/FLANN/flann_manual-1.8.4.pdf http://docs.opencv.org/trunk/modules/flann/doc/flann_fast_approximate_nearest_neighbor_search.html

get_cfgstr(***kwargs*)

get_cfgstr_list(***kwargs*)

get_config_name(***kwargs*)
the user might want to overwrite this function

get_flann_params()

initialize_params()
Initializes config class attributes based on params info list

keys(***kwargs*)

lookup_paraminfo(*key*)

meta_get_cfgstr_list(*ignore_keys=None*, ***kwargs*)
default get_cfgstr_list, can be overrided by a config object

parse_items(***kwargs*)

class wbia.algo.Config.**GenericConfig**(**args*, ***kwargs*)
Bases: utool.Preferences.Pref

get_cfgstr(***kwargs*)

get_cfgstr_list(*ignore_keys=None*, ***kwargs*)
default get_cfgstr_list, can be overrided by a config object

get_config_name(***kwargs*)
the user might want to overwrite this function

initialize_params()
Initializes config class attributes based on params info list

keys(***kwargs*)

lookup_paraminfo(*key*)

parse_items(***kwargs*)

class wbia.algo.Config.**NNConfig**(***kwargs*)
Bases: utool.Preferences.Pref

CommandLine: python -m wbia.algo.Config –exec-NNConfig

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.Config import *  # NOQA
>>> nn_cfg = NNConfig()
>>> nn_cfg = NNConfig(requery=True)
>>> result = nn_cfg.get_cfgstr()
>>> print(result)
_NN(single,K=4,Kn=1,padk=False,cks800)
```

get_cfgstr(***kwargs*)

get_cfgstr_list(*ignore_keys=None*, ***kwargs*)
default get_cfgstr_list, can be overrided by a config object

get_config_name(***kwargs*)
the user might want to overwrite this function

get_param_info_list()

initialize_params()
Initializes config class attributes based on params info list

keys(***kwargs*)

lookup_paraminfo(*key*)

make_feasible()

parse_items(***kwargs*)

class wbia.algo.Config.**NNWeightConfig**(***kwargs*)
Bases: utool.Preferences.Pref

CommandLine: python -m wbia.algo.Config –test-NNWeightConfig

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.Config import *  # NOQA
>>> cfg_list = [
...     NNWeightConfig(),
...     NNWeightConfig(can_match_sameimg=True, can_match_samename=False),
...     NNWeightConfig(ratio_thresh=.625, lnbnn_on=False),
...     NNWeightConfig(ratio_thresh=.625, lnbnn_normer='foobarstr'),
... ]
>>> result = '\n'.join([cfg.get_cfgstr() for cfg in cfg_list])
>>> print(result)
_NNWeight(lnbnn,fg,last,nosqrd_dist)
_NNWeight(lnbnn,fg,last,sameimg,nosamename,nosqrd_dist)
_NNWeight(ratio_thresh=0.625,fg,last,nosqrd_dist)
_NNWeight(ratio_thresh=0.625,lnbnn,fg,last,lnbnn_normer=foobarstr,lnbnn_norm_
↪thresh=0.5,nosqrd_dist)
```

get_cfgstr(***kwargs*)

get_cfgstr_list(*ignore_keys=None*, ***kwargs*)
default get_cfgstr_list, can be overrided by a config object

get_config_name(***kwargs*)
the user might want to overwrite this function

get_param_info_list()

initialize_params()
Initializes config class attributes based on params info list

keys(***kwargs*)

lookup_paraminfo(*key*)

parse_items(***kwargs*)

class wbia.algo.Config.**OccurrenceConfig**(***kwargs*)
Bases: utool.Preferences.Pref

CommandLine: python -m wbia.algo.Config –exec-OccurrenceConfig –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.Config import *  # NOQA
>>> occur_cfg = OccurrenceConfig()
>>> print(occur_cfg.get_cfgstr())
```

get_cfgstr(***kwargs*)

get_cfgstr_list(*ignore_keys=None*, ***kwargs*)
default get_cfgstr_list, can be overrided by a config object

get_config_name(***kwargs*)
the user might want to overwrite this function

get_param_info_list()

initialize_params()
Initializes config class attributes based on params info list

keys(***kwargs*)

lookup_paraminfo(*key*)

parse_items(***kwargs*)

class wbia.algo.Config.**OtherConfig**(***kwargs*)
Bases: utool.Preferences.Pref

get_cfgstr(***kwargs*)

get_cfgstr_list(*ignore_keys=None*, ***kwargs*)
default get_cfgstr_list, can be overrided by a config object

get_config_name(***kwargs*)
the user might want to overwrite this function

initialize_params()
Initializes config class attributes based on params info list

keys(***kwargs*)

lookup_paraminfo(*key*)

parse_items(***kwargs*)

class wbia.algo.Config.**QueryConfig**(***kwargs*)
Bases: utool.Preferences.Pref

LNBNN ranking query configuration parameters

Example

```
>>> # ENABLE_DOCTEST
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> cfg = ibs.cfg.query_cfg
>>> cfgstr = ibs.cfg.query_cfg.get_cfgstr()
>>> print(cfgstr)
```

apply_codename(*codename=None*)
codenames denote mass changes to configurations it is a hacky solution to setting different parameter values all at once.

deepcopy(***kwargs*)

get_cfgstr(***kwargs*)

get_cfgstr_list(***kwargs*)

get_config_name(***kwargs*)
the user might want to overwrite this function

initialize_params()
Initializes config class attributes based on params info list

keys(***kwargs*)

lookup_paraminfo(*key*)

make_feasible()

make_feasible_()
removes invalid parameter settings over all cfgs (move to QueryConfig)

meta_get_cfgstr_list(*ignore_keys=None*, ***kwargs*)
default get_cfgstr_list, can be overrided by a config object

parse_items(***kwargs*)

update_query_cfg(***cfgdict*)

class wbia.algo.Config.**SpatialVerifyConfig**(***kwargs*)
Bases: utool.Preferences.Pref

Spatial verification

get_cfgstr(***kwargs*)

get_cfgstr_list(***kwargs*)

get_config_name(***kwargs*)
the user might want to overwrite this function

initialize_params()
Initializes config class attributes based on params info list

keys (***kwargs*)

lookup_paraminfo (*key*)

meta_get_cfgstr_list (*ignore_keys=None*, ***kwargs*)
default get_cfgstr_list, can be overrided by a config object

parse_items (***kwargs*)

wbia.algo.Config.**default_vsone_cfg** (*ibs*, ***kwargs*)

wbia.algo.Config.**load_named_config** (*cfgname*, *dpath*, *use_config_cache=False*, *verbose=False*)
hack 12-30-2014

Parameters

- **cfgname** (*str*) –
- **dpath** (*str*) –
- **use_config_cache** (*bool*) –

Returns cfg

Return type *Config*

CommandLine: python -m wbia.algo.Config –test-load_named_config

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.Config import *  # NOQA
>>> from wbia.algo.Config import _default_config  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('PZ_Master0')
>>> #ibs.cfg.save()
>>> # build test data
>>> cfgname = 'zebra_plains'
>>> dpath = ibs.get_dbdir()
>>> use_config_cache = True
>>> # execute function
>>> cfg = load_named_config(cfgname, dpath, use_config_cache)
>>> #
>>> keys1 = ut.take_column(cfg.parse_items(), 0)
>>> keys2 = ut.take_column(ibs.cfg.parse_items(), 0)
>>> symdiff = set(keys1) ^ set(keys2)
>>> # verify results
>>> result = str(cfg)
>>> print(result)
```

wbia.algo.Config.**make_config_metaclass** ()
Creates a metaclass for Config objects that automates some of the more tedious functions to write

Like: get_cfgstr and the comparison methods

Example

from wbia.algo.Config import * # NOQA @six.add_metaclass(ConfigMetaclass) class FooConfig(ConfigBase):

def __init__(cfg): super(FooConfig, cfg).__init__(name='FooConfig') cfg.initialize_params()

def get_param_info_list(cfg):

return [ut.ParamInfo('x', 'y'), ut.ParamInfo('z', 3),

]

cfg = FooConfig() logger.info(cfg.get_cfgstr(ignore_keys=['x'])) logger.info(cfg.get_cfgstr(ignore_keys=[]))

cfg = GenericConfig() cfg.x = 'y'

wbia.algo.Config.**parse_config_items**(*cfg*)
Recursively extracts key, val pairs from Config objects into a flat list. (there must not be name conflicts)

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.Config import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> cfg = ibs.cfg.query_cfg
>>> param_list = parse_config_items(cfg)
```

wbia.algo.Config.**set_query_cfg**(*cfg*, *query_cfg*)
hack 12-30-2014

wbia.algo.Config.**update_query_config**(*cfg*, ***kwargs*)
hack 12-30-2014

1.1.4 Module contents

wbia.algo.**IMPORT_TUPLES = [('Config', None, False), ('detect', None, True), ('hots', None,**
cd /home/joncrall/code/wbia/wbia/algo makeinit.py –modname=wbia.algo

Type Regen Command

wbia.algo.**reassign_submodule_attributes**(*verbose=True*)
why reloading all the modules doesnt do this I don't know

wbia.algo.**reload_subs**(*verbose=True*)
Reloads wbia.algo and submodules

wbia.algo.**rrrr**(*verbose=True*)
Reloads wbia.algo and submodules

1.2 wbia.control package

1.2.1 Submodules

1.2.2 wbia.control.DB_SCHEMA module

Module Licence and docstring

TODO: ideally the wbia.constants module would not be used here and each function would use its own constant variables that are suffixed with the last version number that they existed in

Todo: Add a table for original_image_path Add column for image exif orientation

CommandLine: python -m wbia.control.DB_SCHEMA –test-autogen_db_schema

wbia.control.DB_SCHEMA.**VALID_VERSIONS = {'0.0.0': (None, None, None), '1.0.0': (None, <fu**
When updating versions need to test and modify in IBEISController._init_sqldbcore

Type SeeAlso

wbia.control.DB_SCHEMA.**autogen_db_schema**()

CommandLine: python -m wbia.control.DB_SCHEMA –test-autogen_db_schema python -m wbia.control.DB_SCHEMA –test-autogen_db_schema –diff=1 python -m wbia.control.DB_SCHEMA –test-autogen_db_schema -n=-1 python -m wbia.control.DB_SCHEMA –test-autogen_db_schema -n=0 python -m wbia.control.DB_SCHEMA –test-autogen_db_schema -n=1 python -m wbia.control.DB_SCHEMA –force-incremental-db-update python -m wbia.control.DB_SCHEMA –test-autogen_db_schema –write python -m wbia.control.DB_SCHEMA –test-autogen_db_schema –force-incremental-db-update –dump-autogen-schema python -m wbia.control.DB_SCHEMA –test-autogen_db_schema –force-incremental-db-update

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.DB_SCHEMA import *  # NOQA
>>> autogen_db_schema()
```

wbia.control.DB_SCHEMA.**dump_schema_sql**()

CommandLine: python -m wbia.control.DB_SCHEMA dump_schema_sql

wbia.control.DB_SCHEMA.**post_1_0_0**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**post_1_2_0**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**post_1_2_1**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**post_1_3_4**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**post_1_4_7**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**post_1_4_9**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**post_1_5_2**(*db*, *ibs=None*, *verbose=False*)

wbia.control.DB_SCHEMA.**post_1_6_1**(*db*, *ibs=None*, *verbose=False*)

wbia.control.DB_SCHEMA.**post_1_6_4**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**post_1_7_0**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**pre_1_3_1**(*db*, *ibs=None*)
need to ensure that visual uuid columns are unique before we add that constaint to sql. This will remove any annotations that are not unique

wbia.control.DB_SCHEMA.**pre_1_4_8**(*db*, *ibs=None*)

Parameters **ibs** (*wbia.IBEISController*) –

wbia.control.DB_SCHEMA.**pre_1_4_9**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_0_0**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_0_1**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_0_2**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_1_0**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_1_1**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_2_0**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_2_1**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_3_0**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_3_1**(*db*, *ibs=None*)
update the visual_uuid to be a superkey by adding a constraint

wbia.control.DB_SCHEMA.**update_1_3_2**(*db*, *ibs=None*)
for SMART DATA

wbia.control.DB_SCHEMA.**update_1_3_3**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_3_4**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_3_5**(*db*, *ibs=None*)
expand datasets to use new quality measures

wbia.control.DB_SCHEMA.**update_1_3_6**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_3_7**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_3_8**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_3_9**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_4_0**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_4_1**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_4_2**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_4_3**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_4_4**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_4_5**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_4_6**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_4_7**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_4_8**(*db*, *ibs=None*)
change notes to tag_text_data add configuration that made the match add the score of the match add concept of: DEFINIATELY MATCHES, DOES NOT MATCH, CAN NOT DECIDE

Probably want a separate table for the config_rowid matching results because the primary key needs to be (config_rowid, aid1, aid2) OR just (config_rowid, annotmatch_rowid)

wbia.control.DB_SCHEMA.**update_1_4_9**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_5_0**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_5_1**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_5_2**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_5_3**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_5_4**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_5_5**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_6_0**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_6_1**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_6_2**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_6_3**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_6_4**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_6_5**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_6_6**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_6_7**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_6_8**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_6_9**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_7_0**(*db*, *ibs=None*)

Ignore: import wbia ibs = wbia.opendb('testdb1') ibs.annots().yaws ibs.annots().viewpoint_int codes = ibs.annots().viewpoint_code texts = ['unknown' if y is None else y for y in ibs.annots().yaw_texts] assert codes == texts

wbia.control.DB_SCHEMA.**update_1_7_1**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_8_0**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_8_1**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_8_2**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_1_8_3**(*db*, *ibs=None*)

wbia.control.DB_SCHEMA.**update_2_0_0**(*db*, *ibs=None*)

1.2.3 wbia.control.DB_SCHEMA_CURRENT module

AUTOGENERATED ON 10:14:54 2019/07/09 AutogenCommandLine:

python -m wbia.control.DB_SCHEMA –test-autogen_db_schema –force-incremental-db-update –write
python -m wbia.control.DB_SCHEMA –test-autogen_db_schema –force-incremental-db-update –diff=1
python -m wbia.control.DB_SCHEMA –test-autogen_db_schema –force-incremental-db-update

wbia.control.DB_SCHEMA_CURRENT.**update_current**(*db*, *ibs=None*)

1.2.4 wbia.control.IBEISControl module

This module contains the definition of IBEISController. This object allows access to a single database. Construction of this object should be done using wbia.opendb().

Todo: Module Licence and docstring

load plugin logic:

- known plugin list - plugin_register.txt / dirs/symlinks in plugin folder
- disabled flags
- try import && register

- except flag errored
- init db
- check versioning / update
- (determine plugin import ordering?)
- inject and initialize plugins

Note:

There are functions that are injected into the controller that are not defined in this module.

Functions in the IBEISController have been split up into several submodules.

look at the modules listed in autogenmodname_list to see the full list of functions that will be injected into an IBEISController object

Recently, these functions have been enumerated in wbia.control._autogen_explicit_controller.py, and explicitly added to the

controller using subclassing. This submodule only provides function headers, the source code still

resides in the injected modules.

---

class wbia.control.IBEISControl.**IBEISController**(*dbdir=None*, *ensure=True*, *wbaddr=None*, *verbose=True*, *request_dbversion=None*, *request_stagingversion=None*, *force_serial=None*)

Bases: object

IBEISController docstring

NameingConventions: chip - cropped region of interest in an image, maps to one animal cid - chip unique id gid - image unique id (could just be the relative file path) name - name unique id imgsetid - imageset unique id aid - region of interest unique id annot - an annotation i.e. region of interest for a chip theta - angle of rotation for a chip

backup_database()

base_uri
Base database URI without a specific database name

cleanup()
call on del?

clear_table_cache(*tablename=None*)

clone_handle(***kwargs*)

copy_database(*dest_dbdir*)

daily_backup_database()

disconnect_sqldatabase()

dump_database_csv()

ensure_directories()
Makes sure the core directores for the controller exist

get_big_cachedir()

Returns

database directory where aggregate results are stored

Return type bigcachedir (str)

get_cachedir()
database directory of all cached files

get_cachestats_str()
Returns info about the underlying SQL cache memory

get_chipdir()

get_current_log_text()

Example

```
>>> # xdoctest: +REQUIRES(--web-tests)
>>> import wbia
>>> with wbia.opendb_with_web('testdb1') as (ibs, client):
...     resp = client.get('/log/current/')
>>> resp.json
{'status': {'success': True, 'code': 200, 'message': '', 'cache': -1},
↪'response': None}
```

get_database_icon(*max_dsize=(None, 192)*, *aid=None*)

Parameters **max_dsize** (*tuple*) – (default = (None, 192))

Returns None

Return type None

CommandLine: python -m wbia.control.IBEISControl –exec-get_database_icon –show python -m wbia.control.IBEISControl –exec-get_database_icon –show –db Oxford

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.IBEISControl import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> icon = self.get_database_icon()
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.imshow(icon)
>>> ut.show_if_requested()
```

get_db_cache_path()

get_db_core_path()

get_db_init_uuid()

Returns The SQLDatabaseController's initialization UUID

Return type UUID

RESTful: Method: GET URL: /api/core/db/uuid/init/

get_db_name()
Alias for self.get_dbname().

get_db_numbers()

get_db_staging_path()

get_dbdir()
database dir with ibs internal directory

get_dbinfo()

get_dbname()

Returns database name

Return type list_ (list)

RESTful: Method: GET URL: /api/core/db/name/

get_detect_modeldir()

get_detectimg_cachedir()

Returns

database directory of image resized for detections

Return type detectimgdir (str)

get_fig_dir()
ibs internal directory

get_flann_cachedir()

Returns

database directory where the FLANN KD-Tree is stored

Return type flanndir (str)

get_ibsdir()
ibs internal directory

get_imgdir()
ibs internal directory

get_logdir_global(*local=False*)

get_logdir_local()

get_match_thumbdir()

get_neighbor_cachedir()

get_probchip_dir()

get_qres_cachedir()

Returns database directory where query results are stored

Return type qresdir (str)

get_shelves_path()

get_smart_patrol_dir(*ensure=True*)

Parameters **ensure** (*bool*) –

Returns str smart_patrol_dpath

CommandLine: python -m wbia.control.IBEISControl –test-get_smart_patrol_dir

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.IBEISControl import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> ensure = True
>>> # execute function
>>> smart_patrol_dpath = ibs.get_smart_patrol_dir(ensure)
>>> # verify results
>>> ut.assertpath(smart_patrol_dpath, verbose=True)
```

get_thumbdir()
database directory where thumbnails are cached

get_trashdir()

get_uploadsdir()
ibs internal directory

get_wbia_resource_dir()
returns the global resource dir in .config or AppData or whatever

get_web_port_via_scan(*url_base='127.0.0.1'*, *port_base=5000*, *scan_limit=100*, *verbose=True*)

get_workdir()
directory where databases are saved to

is_using_postgres_db
Indicates whether this controller is using postgres as the database

load_plugin_module(*module*)

make_cache_db_uri(*name*)
Given a name of the cache produce a database connection URI

notify_observers()

predict_ws_injury_interim_svm(*aids*)

print_cachestats_str()

register_controller()
registers controller with global list

register_observer(*observer*)

remove_observer(*observer*)

reset_table_cache()

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

show_depc_annot_graph(**args*, ***kwargs*)

CommandLine: python -m wbia.control.IBEISControl –test-show_depc_annot_graph –show python -m wbia.control.IBEISControl –test-show_depc_annot_graph –show –reduced

Example

```
>>> # SCRIPT
>>> from wbia.control.IBEISControl import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> reduced = ut.get_argflag('--reduced')
>>> ibs.show_depc_annot_graph(reduced=reduced)
>>> ut.show_if_requested()
```

show_depc_annot_table_input(*tablename*, **args*, ***kwargs*)

CommandLine: python -m wbia.control.IBEISControl –test-show_depc_annot_table_input –show –tablename=vsone python -m wbia.control.IBEISControl –test-show_depc_annot_table_input –show –tablename=neighbor_index python -m wbia.control.IBEISControl –test-show_depc_annot_table_input –show –tablename=feat_neighbs –testmode

Example

```
>>> # SCRIPT
>>> from wbia.control.IBEISControl import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> tablename = ut.get_argval('--tablename')
>>> ibs.show_depc_annot_table_input(tablename)
>>> ut.show_if_requested()
```

show_depc_graph(*depc*, *reduced=False*)

show_depc_image_graph(***kwargs*)

CommandLine: python -m wbia.control.IBEISControl –test-show_depc_image_graph –show python -m wbia.control.IBEISControl –test-show_depc_image_graph –show –reduced

Example

```
>>> # SCRIPT
>>> from wbia.control.IBEISControl import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> reduced = ut.get_argflag('--reduced')
>>> ibs.show_depc_image_graph(reduced=reduced)
>>> ut.show_if_requested()
```

unregister_controller()

wbia.control.IBEISControl.**request_IBEISController**(*dbdir=None*, *ensure=True*, *wbaddr=None*, *verbose=False*, *use_cache=True*, *request_dbversion=None*, *request_stagingversion=None*, *force_serial=False*, *asproxy=None*, *check_hsdb=True*)

Alternative to directory instantiating a new controller object. Might return a memory cached object

Parameters

- **dbdir** (*str*) – databse directory
- **ensure** (*bool*) –
- **wbaddr** (*None*) –
- **verbose** (*bool*) –
- **use_cache** (*bool*) – use the global wbia controller cache. Make sure this is false if calling from a Thread. (default=True)
- **request_dbversion** (*str*) – developer flag. Do not use.
- **request_stagingversion** (*str*) – developer flag. Do not use.

Returns ibs

Return type *IBEISController*

CommandLine: python -m wbia.control.IBEISControl –test-request_IBEISController

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.IBEISControl import *  # NOQA
>>> from wbia.init.sysres import get_workdir
>>> dbdir = '/'.join([get_workdir(), 'testdb1'])
>>> ensure = True
>>> wbaddr = None
>>> verbose = True
>>> use_cache = False
>>> ibs = request_IBEISController(dbdir, ensure, wbaddr, verbose,
>>>                               use_cache)
>>> result = str(ibs)
>>> print(result)
```

1.2.5 wbia.control.STAGING_SCHEMA module

Module Licence and docstring

TODO: ideally the wbia.constants module would not be used here and each function would use its own constant variables that are suffixed with the last version number that they existed in

CommandLine: python -m wbia.control.STAGING_SCHEMA –test-autogen_staging_schema

wbia.control.STAGING_SCHEMA.**VALID_VERSIONS = {'0.0.0': (None, None, None), '1.0.0': (None**

When updating versions need to test and modify in IBEISController._init_sqldbcore

Type SeeAlso

wbia.control.STAGING_SCHEMA.**autogen_staging_schema**()

CommandLine: python -m wbia.control.STAGING_SCHEMA –test-autogen_staging_schema python -m wbia.control.STAGING_SCHEMA –test-autogen_staging_schema –diff=1 python -m wbia.control.STAGING_SCHEMA –test-autogen_staging_schema -n=-1 python -m wbia.control.STAGING_SCHEMA –test-autogen_staging_schema -n=0 python -m wbia.control.STAGING_SCHEMA –test-autogen_staging_schema -n=1 python -m wbia.control.STAGING_SCHEMA –force-incremental-db-update python -m wbia.control.STAGING_SCHEMA –test-autogen_staging_schema –write python -m wbia.control.STAGING_SCHEMA –test-autogen_staging_schema –force-incremental-db-update –dump-autogen-schema python -m wbia.control.STAGING_SCHEMA –test-autogen_staging_schema –force-incremental-db-update

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.STAGING_SCHEMA import *  # NOQA
>>> autogen_staging_schema()
```

wbia.control.STAGING_SCHEMA.**post_1_0_2**(*db*, *ibs=None*)

wbia.control.STAGING_SCHEMA.**update_1_0_0**(*db*, *ibs=None*)

wbia.control.STAGING_SCHEMA.**update_1_0_1**(*db*, *ibs=None*)

wbia.control.STAGING_SCHEMA.**update_1_0_2**(*db*, *ibs=None*)

wbia.control.STAGING_SCHEMA.**update_1_0_3**(*db*, *ibs=None*)

wbia.control.STAGING_SCHEMA.**update_1_1_0**(*db*, *ibs=None*)

wbia.control.STAGING_SCHEMA.**update_1_1_1**(*db*, *ibs=None*)

wbia.control.STAGING_SCHEMA.**update_1_2_0**(*db*, *ibs=None*)

1.2.6 wbia.control.STAGING_SCHEMA_CURRENT module

AUTOGENERATED ON 10:10:32 2019/05/29 AutogenCommandLine:

python -m wbia.control.STAGING_SCHEMA –test-autogen_staging_schema –force-incremental-db-update –write python -m wbia.control.STAGING_SCHEMA –test-autogen_staging_schema –force-incremental-db-update –diff=1 python -m wbia.control.STAGING_SCHEMA –test-autogen_staging_schema –force-incremental-db-update

wbia.control.STAGING_SCHEMA_CURRENT.**update_current**(*db*, *ibs=None*)

1.2.7 wbia.control._autogen_party_funcs module

Autogenerated IBEISController functions

TemplateInfo: autogen_time = 15:14:53 2015/03/11 autogen_key = party

ToRegenerate: python -m wbia.templates.template_generator –key party –Tcfg with_api_cache=False with_web_api=False with_deleters=False –diff python -m wbia.templates.template_generator –key party –Tcfg with_api_cache=False with_web_api=False with_deleters=False –write

wbia.control._autogen_party_funcs.**add_party**(*ibs*, *party_tag_list*)

Returns returns party_rowid_list of added (or already existing partys)

TemplateInfo: Tadder_native tbl = party

RESTful: Method: POST URL: /api/autogen/

wbia.control._autogen_party_funcs.**get_party_rowid_from_superkey**(*ibs*, *party_tag_list*, *eager=True*, *nInput=None*)

party_rowid_list <- party[party_tag_list]

Parameters **lists** (*superkey*) – party_tag_list

Returns party_rowid_list

TemplateInfo: Tgetter_native_rowid_from_superkey tbl = party

RESTful: Method: GET URL: /api/autogen/party_rowid_from_superkey/

wbia.control._autogen_party_funcs.**get_party_tag**(*ibs*, *party_rowid_list*, *eager=True*, *nInput=None*)

party_tag_list <- party.party_tag[party_rowid_list]

gets data from the "native" column "party_tag" in the "party" table

Parameters **party_rowid_list** (*list*) –

Returns party_tag_list

Return type list

TemplateInfo: Tgetter_table_column col = party_tag tbl = party

RESTful: Method: GET URL: /api/autogen/party/tag/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control._autogen_party_funcs import *  # NOQA
>>> ibs, qreq_ = testdata_ibs()
>>> party_rowid_list = ibs._get_all_party_rowids()
>>> eager = True
>>> party_tag_list = ibs.get_party_tag(party_rowid_list, eager=eager)
>>> assert len(party_rowid_list) == len(party_tag_list)
```

wbia.control._autogen_party_funcs.**testdata_ibs**(*defaultdb='testdb1'*)

Auto-docstr for 'testdata_ibs'

1.2.8 wbia.control._sql_helpers module

wbia.control._sql_helpers.**autogenerate_nth_schema_version**(*schema_spec*, *n=-1*)

dumps, prints, or diffs autogen schema based on command line

Parameters **n** (*int*) –

CommandLine: python -m wbia.control._sql_helpers –test-autogenerate_nth_schema_version

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control._sql_helpers import *  # NOQA
>>> from wbia.control import DB_SCHEMA
>>> # build test data
>>> schema_spec = DB_SCHEMA
>>> n = 1
>>> # execute function
>>> tablename = autogenerate_nth_schema_version(schema_spec, n)
>>> # verify results
>>> result = str(tablename)
>>> print(result)
```

wbia.control._sql_helpers.**compare_string_versions**(*a*, *b*)

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control._sql_helpers import *  # NOQA
>>> a = '1.1.1'
>>> b = '1.0.0'
>>> result1 = compare_string_versions(a, b)
>>> result2 = compare_string_versions(b, a)
>>> result3 = compare_string_versions(a, a)
>>> result = ', '.join(map(str, [result1, result2, result3]))
>>> print(result)
1, -1, 0
```

wbia.control._sql_helpers.**copy_database**(*src_fpath*, *dst_fpath*)

wbia.control._sql_helpers.**database_backup**(*db_dir*, *db_fname*, *backup_dir*, *max_keep=2048*, *manual=True*)

```
>>> db_dir = ibs.get_ibsdir()
>>> db_fname = ibs.sqldb_fname
>>> backup_dir = ibs.backupdir
>>> max_keep = MAX_KEEP
>>> manual = False
```

wbia.control._sql_helpers.**ensure_correct_version**(*ibs*, *db*, *version_expected*, *schema_spec*, *dobackup=True*, *verbose=True*)

FIXME: AN SQL HELPER FUNCTION SHOULD BE AGNOSTIC TO CONTROLER OBJECTS

ensure_correct_version

Parameters

- **ibs** (IBEISController) –
- **db** (*SQLController*) –
- **version_expected** (*str*) – version you want to be at
- **schema_spec** (*module*) – schema module
- **dobackup** (*bool*) –

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control._sql_helpers import *  # NOQA
>>> ibs = '?'
>>> db = ibs.db
>>> version_expected = ibs.db_version_expected
>>> schema_spec = DB_SCHEMA
>>> dobackup = True
>>> result = ensure_correct_version(ibs, db, version_expected, schema_spec,␣
↪dobackup)
>>> print(result)
```

Parameters **schema_spec** (*module*) – module of schema specifications

wbia.control._sql_helpers.**ensure_daily_database_backup**(*db_dir*, *db_fname*, *backup_dir*, *max_keep=2048*)

wbia.control._sql_helpers.**fix_metadata_consistency**(*db*)

duct tape function

db.print_table_csv('metadata')

wbia.control._sql_helpers.**get_backup_fpaths**(*ibs*)

wbia.control._sql_helpers.**get_backupdir**(*db_dir*, *db_fname*)

CommandLine: python -m _sql_helpers get_backupdir –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control._sql_helpers import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> db_dir = ibs.get_ibsdir()
>>> db_fname = ibs.sqldb_fname
>>> backup_dir = ibs.backupdir
>>> result = get_backupdir(db_dir, db_fname)
```

wbia.control._sql_helpers.**get_nth_test_schema_version**(*schema_spec*, *n=-1*)

Gets a fresh and empty test version of a schema

Parameters

- **schema_spec** (*module*) – schema module to get nth version of
- **n** (*int*) – version index (-1 is the latest)

wbia.control._sql_helpers.**remove_old_backups**(*backup_dir*, *ext*, *max_keep*)

wbia.control._sql_helpers.**revert_to_backup**(*ibs*)

Parameters **db_dir** –

CommandLine: python -m wbia.control._sql_helpers –exec-revert_to_backup

Example

```
>>> # SCRIPT
>>> from wbia.control._sql_helpers import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='elephants')
>>> result = revert_to_backup(ibs)
>>> print(result)
```

wbia.control._sql_helpers.**update_schema_version**(*ibs*, *db*, *schema_spec*, *version*, *version_target*, *dobackup=True*, *clearbackup=False*)

version_target = version_expected clearbackup = False FIXME: AN SQL HELPER FUNCTION SHOULD BE AGNOSTIC TO CONTROLER OBJECTS

1.2.9 wbia.control.accessor_decors module

wbia.control.accessor_decors.**adder**(*func*)

wbia.control.accessor_decors.**cache_getter**(*tblname*, *colname=None*, *cfgkeys=None*, *force=False*, *debug=False*)

Creates a getter cacher the class must have a table_cache property varargs are currently unallowed

Parameters

- **tblname** (*str*) –
- **colname** (*str*) –

Returns closure_getter_cacher

Return type function

CommandLine: python -m wbia.control.accessor_decors –test-cache_getter

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.accessor_decors import *  # NOQA
>>> import wbia
>>> from wbia import constants as const
>>> ibs = wbia.opendb('testdb1')
>>> #ibs = wbia.opendb('PZ_MTEST')
>>> valid_nids = ibs.get_valid_nids()
>>> tblname = const.NAME_TABLE
>>> colname = 'annot_rowid'
>>> rowid_list = valid_nids
>>> rowid_list1 = rowid_list[::2]
>>> rowid_list2 = rowid_list[::3]
>>> rowid_list3 = rowid_list[1::2]
>>> kwargs = {}
>>> getter_func = ut.get_method_func(ibs.get_name_aids)
>>> wrp_getter_cacher = cache_getter(tblname, colname, force=True,␣
↪debug=False)(getter_func)
>>> ### Test Getter (caches)
>>> val_list1 = getter_func(ibs, rowid_list1)
```

(continues on next page)

(continued from previous page)

```
>>> val_list2 = wrp_getter_cacher(ibs, rowid_list1)
>>> print(ut.repr2(ibs.table_cache))
>>> val_list3 = wrp_getter_cacher(ibs, rowid_list1)
>>> val_list4 = wrp_getter_cacher(ibs, rowid_list2)
>>> print(ut.repr2(ibs.table_cache))
>>> val_list5 = wrp_getter_cacher(ibs, rowid_list3)
>>> val_list  = wrp_getter_cacher(ibs, rowid_list)
>>> ut.assert_eq(val_list1, val_list2, 'run1')
>>> ut.assert_eq(val_list1, val_list2, 'run2')
>>> print(ut.repr2(ibs.table_cache))
>>> ### Test Setter (invalidates)
>>> setter_func = ibs.set_name_texts
>>> wrp_cache_invalidator = cache_invalidator(tblname, force=True)(lambda *a:␣
↪None)
>>> wrp_cache_invalidator(ibs, rowid_list1)
>>> print(ut.repr2(ibs.table_cache))
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.accessor_decors import *  # NOQA
>>> import wbia
>>> from wbia import constants as const
>>> from wbia.control.manual_feat_funcs import FEAT_KPTS
>>> ibs = wbia.opendb('testdb1')
>>> tblname = const.FEATURE_TABLE,
>>> colname = FEAT_KPTS
>>> aid_list = ibs.get_valid_aids()[0:1]
>>> # Check that config2 actually gets you different vectors in the cache
>>> qreq_ = ibs.new_query_request(aid_list, aid_list, cfgdict={'affine_invariance
↪': False})
>>> config2_ = qreq_.extern_query_config2
>>> kpts_list1 = ibs.get_annot_kpts(aid_list, config2_=None)
>>> kpts_list2 = ibs.get_annot_kpts(aid_list, config2_=config2_)
>>> kp1 = kpts_list1[0][0:1]
>>> kp2 = kpts_list2[0][0:1]
>>> assert kp1.T[3] != 0
>>> assert kp2.T[3] == 0
>>> assert kp2.T[2] == kp2.T[4]
```

Ignore: %timeit getter_func(ibs, rowid_list) %timeit wrp_getter_cacher(ibs, rowid_list)

wbia.control.accessor_decors.**cache_invalidator**(*tblname*, *colnames=None*, *rowidx=None*, *force=False*)

cacher decorator

Parameters

- **tablename** (*str*) – the table that the owns the underlying cache
- **colnames** (*list*) – the list of cached column that this function will invalidate
- **rowidx** (*int*) – the position (not including self) of the invalidated table's native rowid in the writer function's argument signature. If this does not exist you should use None. (default=None)

wbia.control.accessor_decors.**deleter**(*func*)

wbia.control.accessor_decors.**dev_cache_getter**(*tblname*, *colname*, **args*, ***kwargs*)
cache getter for when the database is gaurenteed not to change

wbia.control.accessor_decors.**getter**(*func*)
Getter decorator for functions which takes as the first input a unique id list and returns a heterogeous list of values

wbia.control.accessor_decors.**getter_1to1**(*func*)
Getter decorator for functions which takes as the first input a unique id list and returns a heterogeous list of values

wbia.control.accessor_decors.**getter_1toM**(*func*)
Getter decorator for functions which takes as the first input a unique id list and returns a homogenous list of values

wbia.control.accessor_decors.**getter_numpy**(*func*)
Getter decorator for functions which takes as the first input a unique id list and returns a heterogeous list of values

wbia.control.accessor_decors.**getter_numpy_vector_output**(*func*)
Getter decorator for functions which takes as the first input a unique id list and returns a heterogeous list of values

wbia.control.accessor_decors.**getter_vector_output**(*func*)
Getter decorator for functions which takes as the first input a unique id list and returns a homogenous list of values

wbia.control.accessor_decors.**ider**(*func*)
This function takes returns ids subject to conditions

wbia.control.accessor_decors.**init_tablecache**()

Returns tablecache

Return type defaultdict

CommandLine: python -m wbia.control.accessor_decors –test-init_tablecache

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.accessor_decors import *  # NOQA
>>> result = init_tablecache()
>>> print(result)
```

wbia.control.accessor_decors.**setter**(*func*)

1.2.10 wbia.control.autowrap_api_decorators module

wbia.control.autowrap_api_decorators.**get_decorator**(*submodule*, *func*, *method*)

wbia.control.autowrap_api_decorators.**get_func**(*line*)

wbia.control.autowrap_api_decorators.**get_parts**(*line*, *sub*)

wbia.control.autowrap_api_decorators.**process_file**(*filename*, *sub*)

1.2.11 wbia.control.controller_inject module

Todo: Move flask registering into another file. Should also make the actual flask registration lazy. It should only be executed if a web instance is being started.

python -c "import wbia"

exception wbia.control.controller_inject.**WebDuplicateUUIDException**(*qdup_pos_map={}*, *ddup_pos_map={}*)

Bases: *wbia.control.controller_inject.WebException*

exception wbia.control.controller_inject.**WebException**(*message*, *rawreturn=None*, *code=400*)

Bases: utool.util_dev.NiceRepr, Exception

get_rawreturn(*debug_stack_trace=False*)

exception wbia.control.controller_inject.**WebInvalidInput**(*message*, *key=None*, *value=None*, *image=False*)

Bases: *wbia.control.controller_inject.WebException*

exception wbia.control.controller_inject.**WebInvalidMatchException**(*qaid_list*, *daid_list*)

Bases: *wbia.control.controller_inject.WebException*

exception wbia.control.controller_inject.**WebInvalidUUIDException**(*invalid_image_uuid_list=[]*, *invalid_annot_uuid_list=[]*)

Bases: *wbia.control.controller_inject.WebException*

exception wbia.control.controller_inject.**WebMatchThumbException**(*reference*, *qannot_uuid*, *dannot_uuid*, *version*, *message*)

Bases: *wbia.control.controller_inject.WebException*

exception wbia.control.controller_inject.**WebMissingInput**(*message*, *key=None*)

Bases: *wbia.control.controller_inject.WebException*

exception wbia.control.controller_inject.**WebMissingUUIDException**(*missing_image_uuid_list=[]*, *missing_annot_uuid_list=[]*)

Bases: *wbia.control.controller_inject.WebException*

exception wbia.control.controller_inject.**WebMultipleNamedDuplicateException**(*bad_dict*)

Bases: *wbia.control.controller_inject.WebException*

exception wbia.control.controller_inject.**WebReviewFinishedException**(*query_uuid*)

Bases: *wbia.control.controller_inject.WebException*

exception wbia.control.controller_inject.**WebReviewNotReadyException**(*query_uuid*)

Bases: *wbia.control.controller_inject.WebException*

exception wbia.control.controller_inject.**WebRuntimeException**(*message*)

Bases: *wbia.control.controller_inject.WebException*

exception wbia.control.controller_inject.**WebUnavailableUUIDException**(*unavailable_annot_uuid_list*, *query_uuid*)

Bases: *wbia.control.controller_inject.WebException*

exception wbia.control.controller_inject.**WebUnknownUUIDException**(*unknown_uuid_type_list*, *unknown_uuid_list*)

Bases: *wbia.control.controller_inject.WebException*

wbia.control.controller_inject.**api_remote_wbia**(*remote_wbia_url*, *remote_api_func*, *remote_wbia_port=5001*, ***kwargs*)

wbia.control.controller_inject.**authenticate**(*username*, ***kwargs*)

wbia.control.controller_inject.**authenticated**()

wbia.control.controller_inject.**authentication_challenge**()
Sends a 401 response that enables basic auth.

wbia.control.controller_inject.**authentication_either**(*func*)
authenticated by either hash or user

wbia.control.controller_inject.**authentication_hash_only**(*func*)

wbia.control.controller_inject.**authentication_hash_validate**()
This function is called to check if a username / password combination is valid.

wbia.control.controller_inject.**authentication_user_only**(*func*)

wbia.control.controller_inject.**authentication_user_validate**()
This function is called to check if a username / password combination is valid.

wbia.control.controller_inject.**create_key**()

wbia.control.controller_inject.**crossdomain**(*origin=None*, *methods=None*, *headers=None*, *max_age=21600*, *attach_to_all=True*, *automatic_options=True*)

wbia.control.controller_inject.**deauthenticate**()

wbia.control.controller_inject.**dev_autogen_explicit_imports**()

CommandLine: python -m wbia –tf dev_autogen_explicit_imports

Example

```
>>> # SCRIPT
>>> from wbia.control.controller_inject import *  # NOQA
>>> dev_autogen_explicit_imports()
```

wbia.control.controller_inject.**dev_autogen_explicit_injects**()

CommandLine: python -m wbia –tf dev_autogen_explicit_injects

Example

```
>>> # SCRIPT
>>> from wbia.control.controller_inject import *  # NOQA
>>> dev_autogen_explicit_injects()
```

wbia.control.controller_inject.**get_flask_app**(*templates_auto_reload=True*)

wbia.control.controller_inject.**get_signature**(*key*, *message*)

wbia.control.controller_inject.**get_url_authorization**(*url*)

wbia.control.controller_inject.**get_user**(*username=None*, *name=None*, *organization=None*)

wbia.control.controller_inject.**get_wbia_flask_api**(*__name__*, *DEBUG_PYTHON_STACK_TRACE_JSON_RESPONSE=False*)
For function calls that resolve to api calls and return json.

wbia.control.controller_inject.**get_wbia_flask_route**(*__name__*)
For function calls that resolve to webpages and return html.

wbia.control.controller_inject.**login_required_session**(*function*)

wbia.control.controller_inject.**make_ibs_register_decorator**(*modname*)
builds variables and functions that controller injectable modules need.

wbia.control.controller_inject.**remote_api_wrapper**(*func*)

wbia.control.controller_inject.**translate_wbia_webcall**(*func*, **args*, ***kwargs*)
Called from flask request context

Parameters **func** (*function*) – live python function

Returns (output, True, 200, None, jQuery_callback)

Return type tuple

Example

```
>>> # xdoctest: +REQUIRES(--web-tests)
>>> from wbia.control.controller_inject import *  # NOQA
>>> import wbia
>>> with wbia.opendb_with_web('testdb1') as (ibs, client):
...     aids = client.get('/api/annot/').json
...     failrsp = client.post('/api/annot/uuids/')
...     failrsp2 = client.get('/api/query/chips/simple_dict//', data={'qaid_list
↪': [0], 'daid_list': [0]})
...     log_text = client.get('/api/query/chips/simple_dict/', data={'qaid_list':␣
↪[0], 'daid_list': [0]})
>>> print('\n---\nfailrsp =\n%s' % (failrsp.data,))
>>> print('\n---\nfailrsp2 =\n%s' % (failrsp2.data,))
>>> print('Finished test')
Finished test
```

wbia.control.controller_inject.**translate_wbia_webreturn**(*rawreturn*, *success=True*, *code=None*, *message=None*, *jQuery_callback=None*, *cache=None*, *__skip_microsoft_validation__=False*)

1.2.12 wbia.control.docker_control module

wbia.control.docker_control.**docker_check_container**(*ibs*, *container_name*, *clone=None*, *retry_count=20*, *retry_timeout=15*)

wbia.control.docker_control.**docker_container_IP_port_options**(*ibs*, *container*)

wbia.control.docker_control.**docker_container_clone_name**(*container_name*, *clone=None*)

wbia.control.docker_control.**docker_container_status**(*ibs*, *container_name*, *clone=None*)

wbia.control.docker_control.**docker_container_status_dict**(*ibs*)

wbia.control.docker_control.**docker_container_urls**(*ibs*, *container*, *docker_get_config*)

wbia.control.docker_control.**docker_container_urls_from_name**(*ibs*, *container_name*, *clone=None*)

wbia.control.docker_control.**docker_ensure**(*ibs*, *container_name*, *check_container=True*, *clone=None*)

wbia.control.docker_control.**docker_ensure_image**(*ibs*, *image_name*)

wbia.control.docker_control.**docker_get_config**(*ibs*, *container_name*)

wbia.control.docker_control.**docker_get_container**(*ibs*, *container_name*, *clone=None*)

wbia.control.docker_control.**docker_get_image**(*ibs*, *image_name*)

wbia.control.docker_control.**docker_image_list**(*ibs*)

wbia.control.docker_control.**docker_image_run**(*ibs*, *port=6000*, *volumes=None*)

wbia.control.docker_control.**docker_login**(*ibs*)

wbia.control.docker_control.**docker_pull_image**(*ibs*, *image_name*)

The process of logging into the Azure Container Registry is arcane and complex. In the meantime we'll assume that any image we need in the ACR has been downloaded by a logged-in user.

Host: wildme.azurecr.io Username: example@example.com Password: asecurepassword

Login Script:

Install Azure CLI

https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest

az logout

Logout of any user you may already be using with az-cli

az login

Follow instructions to https://microsoft.com/devicelogin, input the code

Login as example@example.com with password above

az acr login –name wildme

Login to the Azure Container Registry (ACR) for Docker

Verify login with "cat ~/.docker/config.json | jq ".auths" and look for "wildme.azurecr.io"

docker pull wildme.azurecr.io/wbia/example-image:latest

Pull latest nightly image

wbia.control.docker_control.**docker_register_config**(*ibs*, *container_name*, *image_name*, *container_check_func=None*, *run_args={}*, *ensure_new=False*)

wbia.control.docker_control.**docker_run**(*ibs*, *image_name*, *container_name*, *override_run_args*, *clone=None*, *ensure_new=False*)

wbia.control.docker_control.**find_open_port**(*base=5000*, *blacklist=[]*)

wbia.control.docker_control.**is_local_port_open**(*port*)

Parameters **port** (*int*) –

Returns

Return type bool

References

http://stackoverflow.com/questions/7436801/identifying-listening-ports-using-python

CommandLine: python -m utool.util_web is_local_port_open –show

Example

```
>>> # DISABLE_DOCTEST
>>> from utool.util_web import *  # NOQA
>>> port = 32183
>>> assert is_local_port_open(80) is False, 'port 80 should always be closed'
>>> assert is_local_port_open(port) is True, 'maybe this port is actually used?'
```

1.2.13 wbia.control.manual_annot_funcs module

Autogen: python -c "import utool as ut; ut.write_modscript_alias('Tgen.sh', 'wbia.templates.template_generator')" # NOQA sh Tgen.sh –key annot –invert –Tcfg with_getters=True with_setters=True –modfname manual_annot_funcs –funcname-filter=age_m # NOQA sh Tgen.sh –key annot –invert –Tcfg with_getters=True with_setters=True –modfname manual_annot_funcs –funcname-filter=is_ # NOQA sh Tgen.sh –key annot –invert –Tcfg with_getters=True with_setters=True –modfname manual_annot_funcs –funcname-filter=is_ –diff # NOQA

wbia.control.manual_annot_funcs.**add_annots**(*ibs*, *gid_list*, *bbox_list=None*, *theta_list=None*, *species_list=None*, *nid_list=None*, *name_list=None*, *vert_list=None*, *annot_uuid_list=None*, *yaw_list=None*, *viewpoint_list=None*, *quality_list=None*, *multiple_list=None*, *interest_list=None*, *canonical_list=None*, *detect_confidence_list=None*, *notes_list=None*, *annot_visual_uuid_list=None*, *annot_semantic_uuid_list=None*, *species_rowid_list=None*, *staged_uuid_list=None*, *staged_user_id_list=None*, *quiet_delete_thumbs=False*, *prevent_visual_duplicates=True*, *skip_cleaning=False*, *delete_thumb=True*, ***kwargs*)

Adds an annotation to images

 # TODO: remove annot_visual_uuid_list and annot_semantic_uuid_list They are always inferred

Parameters

- **gid_list** (*list*) – image rowids to add annotation to

- **bbox_list** (*list*) – of [x, y, w, h] bounding boxes for each image (supply verts instead)
- **theta_list** (*list*) – orientations of annotations
- **species_list** (*list*) –
- **nid_list** (*list*) –
- **name_list** (*list*) –
- **detect_confidence_list** (*list*) –
- **notes_list** (*list*) –
- **vert_list** (*list*) – alternative to bounding box
- **annot_uuid_list** (*list*) –
- **yaw_list** (*list*) –
- **annot_visual_uuid_list** (*list*) –
- **annot_semantic_uuid_list** (*list*) –
- **quiet_delete_thumbs** (*bool*) –

Returns aid_list

Return type list

CommandLine: python -m wbia.control.manual_annot_funcs –test-add_annots python -m wbia.control.manual_annot_funcs –test-add_annots –verbose –print-caller

Ignore: theta_list = None species_list = None nid_list = None name_list = None detect_confidence_list = None notes_list = None vert_list = None annot_uuid_list = None yaw_list = None quiet_delete_thumbs = False prevent_visual_duplicates = False

RESTful: Method: POST URL: /api/annot/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.IBEISControl import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> prevalid = ibs.get_valid_aids()
>>> num_add = 2
>>> gid_list = ibs.get_valid_gids()[0:num_add]
>>> bbox_list = [(int(w * .1), int(h * .6), int(w * .5), int(h *  .3))
...              for (w, h) in ibs.get_image_sizes(gid_list)]
>>> # Add a test annotation
>>> print('Testing add_annots')
>>> aid_list = ibs.add_annots(gid_list, bbox_list=bbox_list)
>>> bbox_list2 = ibs.get_annot_bboxes(aid_list)
>>> vert_list2 = ibs.get_annot_verts(aid_list)
>>> theta_list2 = ibs.get_annot_thetas(aid_list)
>>> name_list2 = ibs.get_annot_names(aid_list)
>>> print('Ensure=False. Should get back None chip fpaths')
>>> chip_fpaths2 = ibs.get_annot_chip_fpath(aid_list, ensure=False)
>>> assert [fpath is None for fpath in chip_fpaths2], 'should not have fpaths'
>>> print('Ensure=True. Should get back None chip fpaths')
```

(continues on next page)

(continued from previous page)

```
>>> chip_fpaths = ibs.get_annot_chip_fpath(aid_list, ensure=True)
>>> assert all([ut.checkpath(fpath, verbose=True) for fpath in chip_fpaths]),
↪'paths should exist'
>>> ut.assert_eq(len(aid_list), num_add)
>>> ut.assert_eq(len(vert_list2[0]), 4)
>>> assert bbox_list2 == bbox_list, 'bboxes are unequal'
>>> # Be sure to remove test annotation
>>> # if this test fails a resetdbs might be nessary
>>> result = ''
>>> visual_uuid_list = ibs.get_annot_visual_uuids(aid_list)
>>> semantic_uuid_list = ibs.get_annot_semantic_uuids(aid_list)
>>> result += str(visual_uuid_list) + '\n'
>>> result += str(semantic_uuid_list) + '\n'
>>> print('Cleaning up. Removing added annotations')
>>> ibs.delete_annots(aid_list)
>>> assert not any([ut.checkpath(fpath, verbose=True) for fpath in chip_fpaths]),
↪'chip paths'
>>> postvalid = ibs.get_valid_aids()
>>> assert prevalid == postvalid, 'prevalid != postvalid'
>>> result += str(postvalid)
>>> print(result)
[UUID('30f7639b-5161-a561-2c4f-41aed64e5b65'), UUID('5ccbb26d-104f-e655-cf2b-
↪cf92e0ad2fd2')]
[UUID('58905a72-dd31-c42b-d5b5-2312adfc7cba'), UUID('dd58665a-2a8b-8e84-4919-
↪038c80bd9be0')]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
```

Example

```
>>> # Test with prevent_visual_duplicates on
>>> # ENABLE_DOCTEST
>>> from wbia.control.IBEISControl import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> prevalid = ibs.get_valid_aids()
>>> num_add = 1
>>> gid_list = ibs.get_valid_gids()[0:1] * num_add
>>> bbox_list = [(int(w * .1), int(h * .6), int(w * .5), int(h *  .3))
...              for (w, h) in ibs.get_image_sizes(gid_list)]
>>> bbox_list2 = [(int(w * .2), int(h * .6), int(w * .5), int(h *  .3))
...              for (w, h) in ibs.get_image_sizes(gid_list)]
>>> # Add a test annotation
>>> print('Testing add_annots')
>>> aid_list1 = ibs.add_annots(gid_list, bbox_list=bbox_list, prevent_visual_
↪duplicates=True)
>>> aid_list2 = ibs.add_annots(gid_list, bbox_list=bbox_list, prevent_visual_
↪duplicates=True)
>>> aid_list3 = ibs.add_annots(gid_list, bbox_list=bbox_list2, prevent_visual_
↪duplicates=True)
>>> assert aid_list1 == aid_list2, 'aid_list1 == aid_list2'
>>> assert aid_list1 != aid_list3, 'aid_list1 != aid_list3'
>>> aid_list_new = aid_list1 + aid_list3
>>> result = aid_list_new
>>> print('Cleaning up. Removing added annotations')
>>> ibs.delete_annots(aid_list_new)
```

wbia.control.manual_annot_funcs.**annotation_src_api**(*rowid=None*)
Returns the base64 encoded image of annotation <aid>

RESTful: Method: GET URL: /api/annot/<aid>/

wbia.control.manual_annot_funcs.**compute_annot_visual_semantic_uuids**(*ibs*, *gid_list*, *include_preprocess=False*, ***kwargs*)

wbia.control.manual_annot_funcs.**delete_annot_imgthumbs**(*ibs*, *aid_list*)

wbia.control.manual_annot_funcs.**delete_annot_nids**(*ibs*, *aid_list*)
Remove name assocation from the list of input aids. Does this by setting each annotations nid to the UNKNOWN name rowid

RESTful: Method: DELETE URL: /api/annot/name/rowid/

wbia.control.manual_annot_funcs.**delete_annot_speciesids**(*ibs*, *aid_list*)
Deletes nids of a list of annotations

RESTful: Method: DELETE URL: /api/annot/species/rowid/

wbia.control.manual_annot_funcs.**delete_annots**(*ibs*, *aid_list*)
deletes annotations from the database

RESTful: Method: DELETE URL: /api/annot/

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*int*) – list of annotation ids

CommandLine: python -m wbia.control.manual_annot_funcs –test-delete_annots python -m wbia.control.manual_annot_funcs –test-delete_annots –debug-api-cache python -m wbia.control.manual_annot_funcs –test-delete_annots

SeeAlso: back.delete_annot

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> from os.path import exists
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> ibs.delete_empty_nids()
>>> # Add some annotations to delete
>>> num_add = 2
>>> gid_list = ibs.get_valid_gids()[0:num_add]
>>> nid = ibs.make_next_nids(1)[0]
>>> nid_list = [nid] * num_add
>>> bbox_list = [(int(w * .1), int(h * .6), int(w * .5), int(h *  .3))
...              for (w, h) in ibs.get_image_sizes(gid_list)]
>>> new_aid_list = ibs.add_annots(gid_list, bbox_list=bbox_list,
>>>                               nid_list=nid_list)
>>> ibs.get_annot_nids(new_aid_list)
>>> ut.assert_lists_eq(ibs.get_annot_nids(new_aid_list), nid_list)
```

(continues on next page)

(continued from previous page)

```
>>> assert ibs.get_name_aids(nid) == new_aid_list, 'annots should all have same␣
↪name'
>>> assert new_aid_list == ibs.get_name_aids(nid), 'inverse name mapping should␣
↪work'
>>> #thumpaths = ibs.get_image_thumbpath(gid_list, ensure_paths=True, **{
↪'thumbsize': 221})
>>> #assert any(ut.lmap(exists, thumpaths)), 'thumbs should be there'
>>> before_aids = ibs.get_image_aids(gid_list)
>>> print('BEFORE gids: ' + str(before_aids))
>>> result = ibs.delete_annots(new_aid_list)
>>> assert ibs.get_name_aids(nid) == [], 'annots should be removed'
>>> after_aids = ibs.get_image_aids(gid_list)
>>> #thumpaths = ibs.get_image_thumbpath(gid_list, ensure_paths=False, **{
↪'thumbsize': 221})
>>> #assert not any(ut.lmap(exists, thumpaths)), 'thumbs should be gone'
>>> assert after_aids != before_aids, 'the invalidators must have bugs'
>>> print('AFTER gids: ' + str(after_aids))
>>> valid_aids = ibs.get_valid_aids()
>>> assert  [aid not in valid_aids for aid in new_aid_list], 'should no longer be␣
↪valid aids'
>>> print(result)
>>> ibs.delete_empty_nids()
```

wbia.control.manual_annot_funcs.**filter_annotation_set**(*ibs*, *aid_list*, *include_only_gid_list=None*, *yaw='no-filter'*, *is_exemplar=None*, *is_staged=False*, *species=None*, *is_known=None*, *hasgt=None*, *minqual=None*, *has_timestamp=None*, *sort=False*, *is_canonical=None*, *min_timedelta=None*)

wbia.control.manual_annot_funcs.**get_annot_age_months_est**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)

annot_age_months_est_list <- annot.annot_age_months_est[aid_list]

gets data from the annotation's native age in months

Parameters **aid_list** (*list*) –

Returns annot_age_months_est_list

Return type list

RESTful: Method: GET URL: /api/annot/age/months/

wbia.control.manual_annot_funcs.**get_annot_age_months_est_max**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)

annot_age_months_est_max_list <- annot.annot_age_months_est_max[aid_list]

gets data from the "native" column "annot_age_months_est_max" in the "annot" table

Parameters **aid_list** (*list*) –

Returns annot_age_months_est_max_list

Return type list

RESTful: Method: GET URL: /api/annot/age/months/max/

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> aid_list = ibs._get_all_aids()
>>> eager = True
>>> annot_age_months_est_max_list = ibs.get_annot_age_months_est_max(aid_list,␣
↪eager=eager)
>>> assert len(aid_list) == len(annot_age_months_est_max_list)
```

wbia.control.manual_annot_funcs.**get_annot_age_months_est_max_texts**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)

annot_age_months_est_max_texts_list <- annot.annot_age_months_est_max_texts[aid_list]

gets string versions of the annotation's native max age in months

Parameters **aid_list** (*list*) –

Returns annot_age_months_est_max_list

Return type list

RESTful: Method: GET URL: /api/annot/age/months/max/text/

wbia.control.manual_annot_funcs.**get_annot_age_months_est_min**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)

annot_age_months_est_min_list <- annot.annot_age_months_est_min[aid_list]

gets data from the "native" column "annot_age_months_est_min" in the "annot" table

Parameters **aid_list** (*list*) –

Returns annot_age_months_est_min_list

Return type list

RESTful: Method: GET URL: /api/annot/age/months/min/

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> aid_list = ibs._get_all_aids()
>>> eager = True
```

(continues on next page)

(continued from previous page)

```
>>> annot_age_months_est_min_list = ibs.get_annot_age_months_est_min(aid_list,␣
↪eager=eager)
>>> assert len(aid_list) == len(annot_age_months_est_min_list)
```

wbia.control.manual_annot_funcs.**get_annot_age_months_est_min_texts**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)

annot_age_months_est_min_texts_list <- annot.annot_age_months_est_min_texts[aid_list]

gets string versions of the annotation's native min age in months

Parameters **aid_list** (*list*) –

Returns annot_age_months_est_min_list

Return type list

RESTful: Method: GET URL: /api/annot/age/months/min/text/

wbia.control.manual_annot_funcs.**get_annot_age_months_est_texts**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)

annot_age_months_est_texts_list <- annot.annot_age_months_est_texts[aid_list]

gets string versions of the annotation's native combined age in months

Parameters **aid_list** (*list*) –

Returns annot_age_months_est_text_list

Return type list

RESTful: Method: GET URL: /api/annot/age/months/text/

wbia.control.manual_annot_funcs.**get_annot_aid**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)

self verifier .. rubric:: Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.IBEISControl import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids() + [None, -1, 10434320432]
>>> aid_list_ = ibs.get_annot_aid(aid_list)
>>> assert [r is None for r in aid_list_[-3:]]
>>> assert [r is not None for r in aid_list_[0:-3]]
```

wbia.control.manual_annot_funcs.**get_annot_aids_from_semantic_uuid**(*ibs*, *semantic_uuid_list*)

Parameters **semantic_uuid_list** (*list*) –

Returns annot rowids

Return type list

wbia.control.manual_annot_funcs.**get_annot_aids_from_uuid**(*ibs*, *uuid_list*)

Returns annot rowids

Return type list_ (list)

RESTful: Method: GET URL: /api/annot/rowid/uuid/

wbia.control.manual_annot_funcs.**get_annot_aids_from_visual_uuid**(*ibs*, *visual_uuid_list*)

Parameters **visual_uuid_list** (*list*) –

Returns annot rowids

Return type list

wbia.control.manual_annot_funcs.**get_annot_bboxes**(*ibs*, *aid_list*)

Returns annotation bounding boxes in image space

Return type bbox_list (list)

RESTful: Method: GET URL: /api/annot/bbox/

wbia.control.manual_annot_funcs.**get_annot_canonical**(*ibs*, *aid_list*, *default_none_to_false=True*)

RESTful: Method: GET URL: /api/annot/canonical/

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> flag_list = get_annot_canonical(ibs, aid_list)
>>> result = ('flag_list = %s' % (ut.repr2(flag_list),))
>>> print(result)
```

wbia.control.manual_annot_funcs.**get_annot_class_labels**(*ibs*, *aid_list*)
DEPRICATE?

Returns identifying animal name and view

Return type list of tuples

wbia.control.manual_annot_funcs.**get_annot_contact_aids**(*ibs*, *aid_list*, *daid_list=None*, *check_isect=False*, *assume_unique=False*)
Returns the other aids that appear in the same image that this annotation is from.

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*list*) –

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_contact_aids;1

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> contact_aids = ibs.get_annot_contact_aids(aid_list)
>>> contact_gids = ibs.unflat_map(ibs.get_annot_gids, contact_aids)
>>> gid_list = ibs.get_annot_gids(aid_list)
>>> for gids, gid, aids, aid in zip(contact_gids, gid_list, contact_aids, aid_
↪list):
...     assert ut.allsame(gids), 'annots should be from same image'
...     assert len(gids) == 0 or gids[0] == gid, 'and same image as parent annot'
...     assert aid not in aids, 'should not include self'
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb2')
>>> aid_list = ibs.get_valid_aids()
>>> contact_aids = ibs.get_annot_contact_aids(aid_list)
>>> contact_gids = ibs.unflat_map(ibs.get_annot_gids, contact_aids)
>>> gid_list = ibs.get_annot_gids(aid_list)
>>> print('contact_aids = %r' % (contact_aids,))
>>> for gids, gid, aids, aid in zip(contact_gids, gid_list, contact_aids, aid_
↪list):
...     assert ut.allsame(gids), 'annots should be from same image'
...     assert len(gids) == 0 or gids[0] == gid, 'and same image as parent annot'
...     assert aid not in aids, 'should not include self'
```

wbia.control.manual_annot_funcs.**get_annot_detect_confidence**(*ibs*, *aid_list*)

Returns a list confidences that the annotations is a valid detection

Return type list_ (list)

RESTful: Method: GET URL: /api/annot/detect/confidence/

wbia.control.manual_annot_funcs.**get_annot_exemplar_flags**(*ibs*, *aid_list*)
returns if an annotation is an exemplar

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*int*) – list of annotation ids

Returns annot_exemplar_flag_list - True if annotation is an exemplar

Return type list

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_exemplar_flags

RESTful: Method: GET URL: /api/annot/exemplar/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> gid_list = get_annot_exemplar_flags(ibs, aid_list)
>>> result = str(gid_list)
>>> print(result)
```

wbia.control.manual_annot_funcs.**get_annot_gar_rowids**(*ibs*, *aid_list*)
Auto-docstr for 'get_annot_gar_rowids'

wbia.control.manual_annot_funcs.**get_annot_gids**(*ibs*, *aid_list*, *assume_unique=False*)
Get parent image rowids of annotations

Parameters **aid_list** (*list*) –

Returns image rowids

Return type gid_list (list)

RESTful: Method: GET URL: /api/annot/image/rowid/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> result = get_annot_gids(ibs, aid_list)
>>> print(result)
```

wbia.control.manual_annot_funcs.**get_annot_groundfalse**(*ibs*, *aid_list*, *valid_aids=None*, *filter_unknowns=True*, *daid_list=None*)
gets all annotations with different names

Returns a list of aids which are known to be different for each

Return type groundfalse_list (list)

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> groundfalse_list = get_annot_groundfalse(ibs, aid_list)
>>> result = str(groundfalse_list)
>>> print(result)
```

wbia.control.manual_annot_funcs.**get_annot_groundtruth**(*ibs*, *aid_list*, *is_exemplar=None*, *noself=True*, *daid_list=None*)

gets all annotations with the same names

Parameters

- **aid_list** (*list*) – list of annotation rowids to get groundtruth of
- **is_exemplar** (*None*) –
- **noself** (*bool*) –
- **daid_list** (*list*) –

Returns a list of aids with the same name foreach aid in aid_list. a set of aids belonging to the same name is called a groundtruth. A list of these is called a groundtruth_list.

Return type groundtruth_list (list)

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_groundtruth:0 python -m wbia.control.manual_annot_funcs –test-get_annot_groundtruth:1 python -m wbia.control.manual_annot_funcs –test-get_annot_groundtruth:2 python -m –tf get_annot_groundtruth:0 –db=PZ_Master0 –aids=97 –exec-mode

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ut.get_argval('--aids', list, ibs.get_valid_aids())
>>> is_exemplar, noself, daid_list = None, True, None
>>> groundtruth_list = ibs.get_annot_groundtruth(aid_list, is_exemplar, noself,␣
↪daid_list)
>>> result = 'groundtruth_list = ' + str(groundtruth_list)
>>> print(result)
groundtruth_list = [[], [3], [2], [], [6], [5], [], [], [], [], [], [], []]
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> is_exemplar, noself, daid_list = True, True, None
>>> groundtruth_list = ibs.get_annot_groundtruth(aid_list, is_exemplar, noself,␣
↪daid_list)
>>> result = str(groundtruth_list)
>>> print(result)
[[], [3], [2], [], [6], [5], [], [], [], [], [], [], []]
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> is_exemplar, noself, daid_list = False, False, aid_list
>>> groundtruth_list = ibs.get_annot_groundtruth(aid_list, is_exemplar, noself,␣
↪daid_list)
>>> result = str(groundtruth_list)
>>> print(result)
[[1], [], [], [4], [], [], [], [], [9], [], [11], [], []]
```

wbia.control.manual_annot_funcs.**get_annot_has_groundtruth**(*ibs*, *aid_list*, *is_exemplar=None*, *noself=True*, *daid_list=None*)

Parameters

- **aid_list** (*list*) –
- **is_exemplar** (*None*) –
- **noself** (*bool*) –
- **daid_list** (*list*) –

Returns has_gt_list

Return type list

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_has_groundtruth

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> is_exemplar = None
>>> noself = True
>>> daid_list = None
>>> has_gt_list = get_annot_has_groundtruth(ibs, aid_list, is_exemplar, noself,␣
↪daid_list)
>>> result = str(has_gt_list)
>>> print(result)
```

wbia.control.manual_annot_funcs.**get_annot_hashid_semantic_uuid**(*ibs*, *aid_list*, *prefix=''*)

builds an aggregate semantic hash id for a list of aids

Parameters

- **ibs** (*wbia.IBEISController*) – wbia controller object
- **aid_list** (*list*) – list of annotation rowids

- **prefix** (*str*) – (default = '')
- **_new** (*bool*) – Eventually we will change the hashing scheme and all old data will be invalidated. (default=False)

Returns semantic_uuid_hashid

Return type str

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_hashid_semantic_uuid

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> annots = ibs.annots()
>>> prefix = ''
>>> semantic_uuid_hashid = get_annot_hashid_semantic_uuid(ibs, aid_list, prefix)
>>> result = ut.repr2(annots.semantic_uuids[0:2], nl=1) + '\n'
>>> result += ('semantic_uuid_hashid = ' + str(semantic_uuid_hashid))
>>> print(result)
[
    UUID('...'),
    UUID('...'),
]
semantic_uuid_hashid = SUUIDS-13-...
```

wbia.control.manual_annot_funcs.**get_annot_hashid_uuid**(*ibs*, *aid_list*, *prefix=''*)
builds an aggregate random hash id for a list of aids

RESTful: Method: GET URL: /api/annot/uuid/hashid/

wbia.control.manual_annot_funcs.**get_annot_hashid_visual_uuid**(*ibs*, *aid_list*, *prefix=''*, *pathsafe=False*)
builds an aggregate visual hash id for a list of aids

Parameters **_new** (*bool*) – Eventually we will change the hashing scheme and all old data will be invalidated. (default=False)

wbia.control.manual_annot_funcs.**get_annot_image_contributor_tag**(*ibs*, *aid_list*)
Auto-docstr for 'get_annot_image_contributor_tag'

wbia.control.manual_annot_funcs.**get_annot_image_datetime_str**(*ibs*, *aid_list*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*int*) – list of annotation ids

Returns datetime_list

Return type list

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_image_datetime_str

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> datetime_list = get_annot_image_datetime_str(ibs, aid_list)
>>> result = str(datetime_list)
>>> print(result)
```

wbia.control.manual_annot_funcs.**get_annot_image_gps**(*ibs*, *aid_list*)

Parameters **aid_list** (*list*) –

Returns unixtime_list

Return type list

RESTful: Method: GET URL: /api/annot/image/gps/

wbia.control.manual_annot_funcs.**get_annot_image_gps2**(*ibs*, *aid_list*)
fixes the (-1, -1) issue. returns nan instead.

wbia.control.manual_annot_funcs.**get_annot_image_names**(*ibs*, *aid_list*)

Parameters **aid_list** (*list*) –

Returns gname_list the image names of each annotation

Return type list of strs

RESTful: Method: GET URL: /api/annot/image/name/

wbia.control.manual_annot_funcs.**get_annot_image_paths**(*ibs*, *aid_list*)

Parameters **aid_list** (*list*) –

Returns gpath_list the image paths of each annotation

Return type list of strs

RESTful: Method: GET URL: /api/annot/image/file/path/

wbia.control.manual_annot_funcs.**get_annot_image_rowids**(*ibs*, *aid_list*)

wbia.control.manual_annot_funcs.**get_annot_image_set_texts**(*ibs*, *aid_list*)
Auto-docstr for 'get_annot_image_contributor_tag'

RESTful: Method: GET URL: /api/annot/imageset/text/

wbia.control.manual_annot_funcs.**get_annot_image_unixtimes**(*ibs*, *aid_list*, ***kwargs*)

Parameters **aid_list** (*list*) –

Returns unixtime_list

Return type list

RESTful: Method: GET URL: /api/annot/image/unixtime/

wbia.control.manual_annot_funcs.**get_annot_image_unixtimes_asfloat**(*ibs*, *aid_list*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*list*) – list of annotation rowids

Returns unixtime_list

Return type list

CommandLine: python -m wbia.control.manual_annot_funcs –exec-get_annot_image_unixtimes_asfloat –show –db PZ_MTEST

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> unixtime_list = get_annot_image_unixtimes_asfloat(ibs, aid_list)
>>> result = ('unixtime_list = %s' % (str(unixtime_list),))
>>> print(result)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

wbia.control.manual_annot_funcs.**get_annot_image_uuids**(*ibs*, *aid_list*)

Parameters **aid_list** (*list*) –

Returns image_uuid_list

Return type list

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_image_uuids –enableall

RESTful: Method: GET URL: /api/annot/image/uuid/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[0:1]
>>> result = get_annot_image_uuids(ibs, aid_list)
>>> print(result)
[UUID('66ec193a-1619-b3b6-216d-1784b4833b61')]
```

wbia.control.manual_annot_funcs.**get_annot_images**(*ibs*, *aid_list*)

Parameters **aid_list** (*list*) –

Returns the images of each annotation

Return type list of ndarrays

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_images

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[0:1]
>>> image_list = ibs.get_annot_images(aid_list)
>>> result = str(list(map(np.shape, image_list)))
>>> print(result)
[(715, 1047, 3)]
```

wbia.control.manual_annot_funcs.**get_annot_imgset_uuids**(*ibs*, *aid_list*)
Get parent image rowids of annotations

Parameters **aid_list** (*list*) –

Returns imageset uuids

Return type imgset_uuid_list (list)

RESTful: Method: GET URL: /api/annot/imageset/uuid/

wbia.control.manual_annot_funcs.**get_annot_imgsetids**(*ibs*, *aid_list*)
Get parent image rowids of annotations

Parameters **aid_list** (*list*) –

Returns imageset rowids

Return type imgsetid_list (list)

RESTful: Method: GET URL: /api/annot/imageset/rowid/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> result = get_annot_gids(ibs, aid_list)
>>> print(result)
```

wbia.control.manual_annot_funcs.**get_annot_interest**(*ibs*, *aid_list*)

RESTful: Method: GET URL: /api/annot/interest/

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> flag_list = get_annot_interest(ibs, aid_list)
```

(continues on next page)

(continued from previous page)

```
>>> result = ('flag_list = %s' % (ut.repr2(flag_list),))
>>> print(result)
```

wbia.control.manual_annot_funcs.**get_annot_isjunk**(*ibs*, *aid_list*)
Auto-docstr for 'get_annot_isjunk'

wbia.control.manual_annot_funcs.**get_annot_metadata**(*ibs*, *aid_list*, *return_raw=False*)

Returns annot metadata dictionary

Return type list_ (list)

RESTful: Method: GET URL: /api/annot/metadata/

wbia.control.manual_annot_funcs.**get_annot_missing_uuid**(*ibs*, *uuid_list*)

Returns a list of missing annot uuids

Return type list_ (list)

wbia.control.manual_annot_funcs.**get_annot_multiple**(*ibs*, *aid_list*)

RESTful: Method: GET URL: /api/annot/multiple/

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> flag_list = get_annot_multiple(ibs, aid_list)
>>> result = ('flag_list = %s' % (ut.repr2(flag_list),))
>>> print(result)
```

wbia.control.manual_annot_funcs.**get_annot_name_rowids**(*ibs*, *aid_list*, *distinguish_unknowns=True*, *assume_unique=False*)

Returns the name id of each annotation.

Return type list_ (list)

CommandLine: python -m wbia.control.manual_annot_funcs –exec-get_annot_name_rowids

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> from wbia import constants as const
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> distinguish_unknowns = True
>>> nid_arr1 = np.array(ibs.get_annot_name_rowids(aid_list, distinguish_
↪unknowns=distinguish_unknowns))
```

(continues on next page)

(continued from previous page)

```
>>> nid_arr2 = np.array(ibs.get_annot_name_rowids(aid_list, distinguish_
↪unknowns=False))
>>> nid_arr2 = np.array(ibs.get_annot_name_rowids(None, distinguish_
↪unknowns=True))
>>> assert const.UNKNOWN_LBLANNOT_ROWID == 0
>>> assert np.all(nid_arr1[np.where(const.UNKNOWN_LBLANNOT_ROWID == nid_arr2)[0]]
↪< 0)
```

wbia.control.manual_annot_funcs.**get_annot_name_texts**(*ibs*, *aid_list*, *distinguish_unknowns=False*)

Parameters **aid_list** (*list*) –

Returns

name_list. e.g: ['fred', 'sue', ...] for each annotation identifying the individual

Return type list or strs

RESTful: Method: GET URL: /api/annot/name/text/

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_name_texts

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[::2]
>>> result = ut.repr2(get_annot_name_texts(ibs, aid_list), nl=False)
>>> print(result)
['____', 'easy', 'hard', 'jeff', '____', '____', 'zebra']
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[::2]
>>> result = ut.repr2(get_annot_name_texts(ibs, aid_list, True), nl=False)
>>> print(result)
['____1', 'easy', 'hard', 'jeff', '____9', '____11', 'zebra']
```

wbia.control.manual_annot_funcs.**get_annot_name_uuids**(*ibs*, *aid_list*, ***kwargs*)

alias

RESTful: Method: GET URL: /api/annot/name/uuid/

wbia.control.manual_annot_funcs.**get_annot_names**(*ibs*, *aid_list*, *distinguish_unknowns=False*)

alias

wbia.control.manual_annot_funcs.**get_annot_nids**(*ibs*, *aid_list*, *distinguish_unknowns=True*)

alias

RESTful: Method: GET URL: /api/annot/name/rowid/

wbia.control.manual_annot_funcs.**get_annot_notes**(*ibs*, *aid_list*)

Returns a list of annotation notes

Return type annotation_notes_list (list)

RESTful: Method: GET URL: /api/annot/note/

wbia.control.manual_annot_funcs.**get_annot_num_contact_aids**(*ibs*, *aid_list*)
Auto-docstr for 'get_annot_num_contact_aids'

wbia.control.manual_annot_funcs.**get_annot_num_groundtruth**(*ibs*, *aid_list*, *is_exemplar=None*, *noself=True*, *daid_list=None*)

Returns number of other chips with the same name

Return type list_ (list)

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_num_groundtruth python -m wbia.control.manual_annot_funcs –test-get_annot_num_groundtruth:0 python -m wbia.control.manual_annot_funcs –test-get_annot_num_groundtruth:1

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> noself = True
>>> result = get_annot_num_groundtruth(ibs, aid_list, noself=noself)
>>> print(result)
[0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0]
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> noself = False
>>> result = get_annot_num_groundtruth(ibs, aid_list, noself=noself)
>>> print(result)
[1, 2, 2, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1]
```

wbia.control.manual_annot_funcs.**get_annot_num_verts**(*ibs*, *aid_list*)

Returns the number of vertices that form the polygon of each chip

Return type nVerts_list (list)

RESTful: Method: GET URL: /api/annot/num/vert/

wbia.control.manual_annot_funcs.**get_annot_otherimage_aids**(*ibs*, *aid_list*, *daid_list=None*, *assume_unique=False*)

Auto-docstr for 'get_annot_otherimage_aids'

wbia.control.manual_annot_funcs.**get_annot_parent_aid**(*ibs*, *aid_list*)

Returns a list of parent (in terms of parts) annotation rowids.

Return type list_ (list)

wbia.control.manual_annot_funcs.**get_annot_part_rowids**(*ibs*, *aid_list*, *is_staged=False*)

Returns a list of part rowids for each image by aid

Return type list_ (list)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*list*) –

Returns part_rowids_list

Return type list

RESTful: Method: GET URL: /api/annot/part/rowid/

wbia.control.manual_annot_funcs.**get_annot_probchip_fpath**(*ibs*, *aid_list*, *config2_=None*)

Returns paths to probability images.

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*list*) – list of annotation rowids
- **config2** (*dict*) – (default = None)

Returns probchip_fpath_list

Return type list

CommandLine: python -m wbia.control.manual_annot_funcs –exec-get_annot_probchip_fpath –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> aid_list = ibs.get_valid_aids()[0:10]
>>> config2_ = {'fw_detector': 'cnn'}
>>> probchip_fpath_list = get_annot_probchip_fpath(ibs, aid_list, config2_)
>>> result = ('probchip_fpath_list = %s' % (str(probchip_fpath_list),))
>>> print(result)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> iteract_obj = pt.interact_multi_image.MultiImageInteraction(probchip_fpath_
↪list, nPerPage=4)
```

(continues on next page)

(continued from previous page)

```
>>> iteract_obj.start()
>>> ut.show_if_requested()
```

wbia.control.manual_annot_funcs.**get_annot_qualities**(*ibs*, *aid_list*, *eager=True*)

annot_quality_list <- annot.annot_quality[aid_list]

gets data from the "native" column "annot_quality" in the "annot" table

Parameters **aid_list** (*list*) –

Returns annot_quality_list

Return type list

TemplateInfo: Tgetter_table_column col = annot_quality tbl = annot

SeeAlso: wbia.const.QUALITY_INT_TO_TEXT

RESTful: Method: GET URL: /api/annot/quality/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> ibs, qreq_ = testdata_ibs()
>>> aid_list = ibs._get_all_aids()
>>> eager = True
>>> annot_quality_list = ibs.get_annot_qualities(aid_list, eager=eager)
>>> print('annot_quality_list = %r' % (annot_quality_list,))
>>> assert len(aid_list) == len(annot_quality_list)
```

wbia.control.manual_annot_funcs.**get_annot_quality_int**(*ibs*, *aid_list*, *eager=True*)

new alias

wbia.control.manual_annot_funcs.**get_annot_quality_texts**(*ibs*, *aid_list*)

Auto-docstr for 'get_annot_quality_texts'

RESTful: Method: GET URL: /api/annot/quality/text/

wbia.control.manual_annot_funcs.**get_annot_reviewed**(*ibs*, *aid_list*)

Returns "All Instances Found" flag, true if all objects of interest

Return type list_ (list)

(animals) have an ANNOTATION in the annot

RESTful: Method: GET URL: /api/annot/reviewed/

wbia.control.manual_annot_funcs.**get_annot_rotated_verts**(*ibs*, *aid_list*)

Returns verticies after rotation by theta.

Return type rotated_vert_list (list)

RESTful: Method: GET URL: /api/annot/vert/rotated/

wbia.control.manual_annot_funcs.**get_annot_rowids_from_partial_vuuids**(*ibs*, *partial_vuuid_strs*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **partial_uuid_list** (*list*) –

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annots_from_partial_uuids

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[::2]
>>> vuuids = ibs.get_annot_visual_uuids(aid_list)
>>> partial_vuuid_strs = [u[0:4] for u in map(str, vuuids)]
>>> aids_list = get_annot_rowids_from_partial_vuuids(ibs, partial_uuid_list)
>>> print(result)
[[1], [3], [5], [7], [9], [11], [13]]
```

wbia.control.manual_annot_funcs.**get_annot_rowids_from_visual_uuid**(*ibs*, *visual_uuid_list*)

Parameters **visual_uuid_list** (*list*) –

Returns annot rowids

Return type list

wbia.control.manual_annot_funcs.**get_annot_rows**(*ibs*, *aid_list*)
Auto-docstr for 'get_annot_rows'

wbia.control.manual_annot_funcs.**get_annot_semantic_uuid_info**(*ibs*, *aid_list*, *_visual_infotup=None*)
Semenatic uuids are made up of visual and semantic information. Semantic information is name, species, yaw. Visual info is image uuid, verts, and theta

Parameters

- **aid_list** (*list*) –
- **_visual_infotup** (*tuple*) – internal use only

Returns semantic_infotup (image_uuid_list, verts_list, theta_list, yaw_list, name_list, species_list)

Return type tuple

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_semantic_uuid_info

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[0:2]
>>> semantic_infotup = ibs.get_annot_semantic_uuid_info(aid_list)
```

(continues on next page)

(continued from previous page)

```
>>> result = ut.repr2(list(zip(*semantic_infotup))[1])
>>> print(result)
(UUID('d8903434-942f-e0f5-d6c2-0dcbe3137bf7'), ((0, 0), (1035, 0), (1035, 576),␣
↪(0, 576)), 0.0, 'left', 'easy', 'zebra_plains')
```

wbia.control.manual_annot_funcs.**get_annot_semantic_uuids**(*ibs*, *aid_list*)
annot_semantic_uuid_list <- annot.annot_semantic_uuid[aid_list]

gets data from the "native" column "annot_semantic_uuid" in the "annot" table

Parameters **aid_list** (*list*) –

Returns annot_semantic_uuid_list

Return type list

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_semantic_uuids

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> ibs, qreq_ = testdata_ibs()
>>> aid_list = ibs._get_all_aids()[0:1]
>>> annot_semantic_uuid_list = ibs.get_annot_semantic_uuids(aid_list)
>>> assert len(aid_list) == len(annot_semantic_uuid_list)
>>> print(annot_semantic_uuid_list)
[UUID('9acc1a8e-b35f-11b5-f844-9e8fd5dd7ad9')]
```

wbia.control.manual_annot_funcs.**get_annot_sex**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)
Auto-docstr for 'get_annot_sex'

RESTful: Method: GET URL: /api/annot/sex/

wbia.control.manual_annot_funcs.**get_annot_sex_texts**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)
Auto-docstr for 'get_annot_sex_texts'

RESTful: Method: GET URL: /api/annot/sex/text/

wbia.control.manual_annot_funcs.**get_annot_species**(*ibs*, *aid_list*)
alias

RESTful: Method: GET URL: /api/annot/species/

wbia.control.manual_annot_funcs.**get_annot_species_rowids**(*ibs*, *aid_list*)
species_rowid_list <- annot.species_rowid[aid_list]

gets data from the "native" column "species_rowid" in the "annot" table

Parameters **aid_list** (*list*) –

Returns species_rowid_list

Return type list

RESTful: Method: GET URL: /api/annot/species/rowid/

wbia.control.manual_annot_funcs.**get_annot_species_texts**(*ibs*, *aid_list*)

Parameters **aid_list** (*list*) –

Returns species_list - a list of strings [‘plains_zebra’, ‘grevys_zebra’, ...] for each annotation identifying the species

Return type list

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_species_texts

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[1::3]
>>> result = ut.repr2(get_annot_species_texts(ibs, aid_list), nl=False)
>>> print(result)
['zebra_plains', 'zebra_plains', '____', 'bear_polar']
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('PZ_MTEST')
>>> aid_list = ibs.get_valid_aids()
>>> species_list = get_annot_species_texts(ibs, aid_list)
>>> result = ut.repr2(list(set(species_list)), nl=False)
>>> print(result)
['zebra_plains']
```

RESTful: Method: GET URL: /api/annot/species/text/

wbia.control.manual_annot_funcs.**get_annot_species_uuids**(*ibs*, *aid_list*)
species_rowid_list <- annot.species_rowid[aid_list]

Parameters **aid_list** (*list*) –

Returns species_uuid_list

Return type list

RESTful: Method: GET URL: /api/annot/species/uuid/

wbia.control.manual_annot_funcs.**get_annot_staged_flags**(*ibs*, *aid_list*)
returns if an annotation is staged

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*int*) – list of annotation ids

Returns annot_staged_flag_list - True if annotation is staged

Return type list

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_staged_flags

RESTful: Method: GET URL: /api/annot/staged/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> gid_list = get_annot_staged_flags(ibs, aid_list)
>>> result = str(gid_list)
>>> print(result)
```

wbia.control.manual_annot_funcs.**get_annot_staged_metadata**(*ibs*, *aid_list*, *return_raw=False*)

Returns annot metadata dictionary

Return type list_ (list)

RESTful: Method: GET URL: /api/annot/staged/metadata/

wbia.control.manual_annot_funcs.**get_annot_staged_user_ids**(*ibs*, *aid_list*)
returns if an annotation is staged

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*int*) – list of annotation ids

Returns annot_staged_user_id_list - True if annotation is staged

Return type list

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_staged_user_ids

RESTful: Method: GET URL: /api/annot/staged/user/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> gid_list = get_annot_staged_user_ids(ibs, aid_list)
>>> result = str(gid_list)
>>> print(result)
```

wbia.control.manual_annot_funcs.**get_annot_staged_uuids**(*ibs*, *aid_list*)

Returns annot_uuid_list a list of image uuids by aid

Return type list

RESTful: Method: GET URL: /api/annot/staged/uuid/

wbia.control.manual_annot_funcs.**get_annot_static_encounter**(*ibs*, *aids*)

wbia.control.manual_annot_funcs.**get_annot_tag_text**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)

annot_tags_list <- annot.annot_tags[aid_list]

gets data from the "native" column "annot_tags" in the "annot" table

Parameters **aid_list** (*list*) –

Returns annot_tags_list

Return type list

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> aid_list = ibs._get_all_aids()
>>> eager = True
>>> annot_tags_list = ibs.get_annot_tag_text(aid_list, eager=eager)
>>> assert len(aid_list) == len(annot_tags_list)
```

wbia.control.manual_annot_funcs.**get_annot_thetas**(*ibs*, *aid_list*)

Returns a list of floats describing the angles of each chip

Return type theta_list (list)

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_thetas

RESTful: Method: GET URL: /api/annot/theta/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('NAUT_test')
>>> aid_list = ibs.get_valid_aids()
>>> result = get_annot_thetas(ibs, aid_list)
>>> print(result)
[2.75742, 0.792917, 2.53605, 2.67795, 0.946773, 2.56729]
```

wbia.control.manual_annot_funcs.**get_annot_uuids**(*ibs*, *aid_list*)

Returns annot_uuid_list a list of image uuids by aid

Return type list

RESTful: Method: GET URL: /api/annot/uuid/

wbia.control.manual_annot_funcs.**get_annot_verts**(*ibs*, *aid_list*)

Returns the vertices that form the polygon of each chip

Return type vert_list (list)

RESTful: Method: GET URL: /api/annot/vert/

wbia.control.manual_annot_funcs.**get_annot_viewpoint_code**(*ibs*, *aids*)

Doctest:

```
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[::3]
>>> result = get_annot_viewpoint_code(ibs, aid_list)
>>> print(result)
['left', 'left', 'unknown', 'left', 'unknown']
```

wbia.control.manual_annot_funcs.**get_annot_viewpoint_int**(*ibs*, *aids*, *assume_unique=False*)

wbia.control.manual_annot_funcs.**get_annot_viewpoints**(*ibs*, *aid_list*, *assume_unique=False*)

Returns the viewpoint for the annotation

Return type viewpoint_text (list)

RESTful: Method: GET URL: /api/annot/viewpoint/

wbia.control.manual_annot_funcs.**get_annot_visual_uuid_info**(*ibs*, *aid_list*)

Returns information used to compute annotation UUID. The image uuid, annotation verticies, are theta is hashted together to

compute the visual uuid.

The visual uuid does not include name or species information.

get_annot_visual_uuid_info

Parameters **aid_list** (*list*) –

Returns visual_infotup (image_uuid_list, verts_list, theta_list)

Return type tuple

SeeAlso: get_annot_visual_uuids get_annot_semantic_uuid_info

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_visual_uuid_info

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[0:2]
>>> visual_infotup = ibs.get_annot_visual_uuid_info(aid_list)
>>> result = str(list(zip(*visual_infotup))[0])
>>> print(result)
(UUID('66ec193a-1619-b3b6-216d-1784b4833b61'), ((0, 0), (1047, 0), (1047, 715),␣
↪(0, 715)), 0.0)
```

wbia.control.manual_annot_funcs.**get_annot_visual_uuids**(*ibs*, *aid_list*)

The image uuid, annotation verticies, are theta is hashted together to compute the visual uuid. The visual uuid does not include name or species information.

annot_visual_uuid_list <- annot.annot_visual_uuid[aid_list]

gets data from the "native" column "annot_visual_uuid" in the "annot" table

Parameters **aid_list** (*list*) –

Returns annot_visual_uuid_list

Return type list

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_visual_uuids

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> ibs, qreq_ = testdata_ibs()
>>> aid_list = ibs._get_all_aids()[0:1]
>>> annot_visual_uuid_list = ibs.get_annot_visual_uuids(aid_list)
>>> assert len(aid_list) == len(annot_visual_uuid_list)
>>> print(annot_visual_uuid_list)
[UUID('8687dcb6-1f1f-fdd3-8b72-8f36f9f41905')]
```

[UUID('76de0416-7c92-e1b3-4a17-25df32e9c2b4')]

wbia.control.manual_annot_funcs.**get_annot_yaw_texts**(*ibs*, *aid_list*, *assume_unique=False*)

Auto-docstr for 'get_annot_yaw_texts'

DEPRICATE

RESTful: Method: GET URL: /api/annot/yaw/text/

wbia.control.manual_annot_funcs.**get_annot_yaws**(*ibs*, *aid_list*, *assume_unique=False*)

A yaw is the yaw of the annotation in radians yaw is inverted. Will be fixed soon.

DEPRICATE

The following views have these angles of yaw: left side - 0.50 tau radians front side - 0.25 tau radians right side - 0.00 tau radians back side - 0.75 tau radians

tau = 2 * pi

SeeAlso: wbia.const.VIEWTEXT_TO_YAW_RADIANS

Returns the yaw (in radians) for the annotation

Return type yaw_list (list)

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_annot_yaws

RESTful: Method: GET URL: /api/annot/yaw/

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[::3]
>>> result = get_annot_yaws(ibs, aid_list)
>>> print(result)
[3.141592653589793, 3.141592653589793, None, 3.141592653589793, None]
```

wbia.control.manual_annot_funcs.**get_annot_yaws_asfloat**(*ibs*, *aid_list*)
Ensures that Nones are returned as nans

DEPRICATE

wbia.control.manual_annot_funcs.**get_num_annotations**(*ibs*, ***kwargs*)
Number of valid annotations

wbia.control.manual_annot_funcs.**get_valid_aids**(*ibs*, *imgsetid=None*, *include_only_gid_list=None*, *yaw='no-filter'*, *is_exemplar=None*, *is_staged=False*, *species=None*, *is_known=None*, *hasgt=None*, *minqual=None*, *has_timestamp=None*, *min_timedelta=None*)
High level function for getting all annotation ids according a set of filters.

Note: The yaw value cannot be None as a default because None is used as a filtering value

Parameters

- **ibs** (*IBEISController*) – wbia controller object
- **imgsetid** (*int*) – imageset id (default = None)
- **include_only_gid_list** (*list*) – if specified filters annots not in these gids (default = None)
- **yaw** (*str*) – (default = 'no-filter')
- **is_exemplar** (*bool*) – if specified filters annots to either be or not be exemplars (default = None)
- **species** (*str*) – (default = None)
- **is_known** (*bool*) – (default = None)
- **min_timedelta** (*int*) – minimum timedelta between annots of known individuals
- **hasgt** (*bool*) – (default = None)

Returns aid_list - a list of valid ANNOTATION unique ids

Return type list

CommandLine: python -m wbia.control.manual_annot_funcs –test-get_valid_aids

Ignore: ibs.print_annotation_table()

RESTful: Method: GET URL: /api/annot/

Example

```
>>> # FIXME failing-test (22-Jul-2020) This test is failing and it's not clear␣
↪how to fix it
>>> # xdoctest: +SKIP
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> ut.exec_funckw(get_valid_aids, globals())
>>> imgsetid = 1
>>> yaw = 'no-filter'
>>> species = ibs.const.TEST_SPECIES.ZEB_PLAIN
>>> is_known = False
>>> ibs.delete_all_imagesets()
>>> ibs.compute_occurrences(config={'use_gps': False, 'seconds_thresh': 600})
>>> aid_list = get_valid_aids(ibs, imgsetid=imgsetid, species=species, is_
↪known=is_known)
>>> ut.assert_eq(ibs.get_annot_names(aid_list), [ibs.const.UNKNOWN] * 2, 'bad name
↪')
>>> ut.assert_eq(ibs.get_annot_species(aid_list), [species] * 2, 'bad species')
>>> ut.assert_eq(ibs.get_annot_exemplar_flags(aid_list), [False] * 2, 'bad␣
↪exemplar')
>>> result = str(aid_list)
>>> print(result)
```

[1, 4]

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list1 = get_valid_aids(ibs, is_exemplar=True)
>>> aid_list2 = get_valid_aids(ibs, is_exemplar=False)
>>> intersect_aids = set(aid_list1).intersection(aid_list2)
>>> ut.assert_eq(len(aid_list1), 9)
>>> ut.assert_eq(len(aid_list2), 4)
>>> ut.assert_eq(len(intersect_aids), 0)
```

Ignore: import utool as ut setup = ut.codeblock(

‘‘‘ import wbia ibs = wbia.opendb(‘PZ_Master1’) ‘‘‘

) stmt_list = [

ut.codeblock(‘‘‘ ibs.db.get_all_rowids_where(ibs.const.ANNOTATION_TABLE, wbia.control.DB_SCHEMA.ANNOT_PARENT_ROWID + ” IS NULL”, tuple()) ‘‘‘),

ut.codeblock(‘‘‘ ibs.db.get_all_rowids(ibs.const.ANNOTATION_TABLE) ‘‘‘),

] iterations = 100 verbose = True _ = ut.timeit_compare(stmt_list, setup=setup, iterations=iterations, verbose=verbose)

wbia.control.manual_annot_funcs.**get_valid_annot_uuids**(*ibs*)

Returns annot_uuid_list a list of image uuids for all valid aids

Return type list

wbia.control.manual_annot_funcs.**set_annot_age_months_est_max**(*ibs*, *aid_list*, *annot_age_months_est_max_list*, *duplicate_behavior='error'*)

annot_age_months_est_max_list -> annot.annot_age_months_est_max[aid_list]

Parameters

- **aid_list** –
- **annot_age_months_est_max_list** –

TemplateInfo: Tsetter_native_column tbl = annot col = annot_age_months_est_max

RESTful: Method: PUT URL: /api/annot/age/months/max/

wbia.control.manual_annot_funcs.**set_annot_age_months_est_min**(*ibs*, *aid_list*, *annot_age_months_est_min_list*, *duplicate_behavior='error'*)

annot_age_months_est_min_list -> annot.annot_age_months_est_min[aid_list]

Parameters

- **aid_list** –
- **annot_age_months_est_min_list** –

TemplateInfo: Tsetter_native_column tbl = annot col = annot_age_months_est_min

RESTful: Method: PUT URL: /api/annot/age/months/min/

wbia.control.manual_annot_funcs.**set_annot_bboxes**(*ibs*, *aid_list*, *bbox_list*, *delete_thumbs=True*, ***kwargs*)

Sets bboxes of a list of annotations by aid,

Parameters

- **aid_list** (*list of rowids*) – list of annotation rowids
- **bbox_list** (*list of (x, y, w, h)*) – new bounding boxes for each aid

---

Note: set_annot_bboxes is a proxy for set_annot_verts

RESTful: Method: PUT URL: /api/annot/bbox/

wbia.control.manual_annot_funcs.**set_annot_canonical**(*ibs*, *aid_list*, *flag_list*)

Sets the annot all instances found bit

RESTful: Method: PUT URL: /api/annot/canonical/

wbia.control.manual_annot_funcs.**set_annot_detect_confidence**(*ibs*, *aid_list*, *confidence_list*)

Sets annotation notes

RESTful: Method: PUT URL: /api/annot/detect/confidence/

wbia.control.manual_annot_funcs.**set_annot_exemplar_flags**(*ibs*, *aid_list*, *flag_list*)

Sets if an annotation is an exemplar

RESTful: Method: PUT URL: /api/annot/exemplar/

wbia.control.manual_annot_funcs.**set_annot_interest**(*ibs*, *aid_list*, *flag_list*, *quiet_delete_thumbs=False*, *delete_thumbs=True*)

Sets the annot all instances found bit

RESTful: Method: PUT URL: /api/annot/interest/

wbia.control.manual_annot_funcs.**set_annot_metadata**(*ibs*, *aid_list*, *metadata_dict_list*)

Sets the annot's metadata using a metadata dictionary

RESTful: Method: PUT URL: /api/annot/metadata/

CommandLine: python -m wbia.control.manual_annot_funcs –test-set_annot_metadata

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> import random
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[0:1]
>>> metadata_dict_list = [
>>>     {'test': random.uniform(0.0, 1.0)},
>>> ]
>>> print(ut.repr2(metadata_dict_list))
>>> ibs.set_annot_metadata(aid_list, metadata_dict_list)
>>> # verify results
>>> metadata_dict_list_ = ibs.get_annot_metadata(aid_list)
>>> print(ut.repr2(metadata_dict_list_))
>>> assert metadata_dict_list == metadata_dict_list_
>>> metadata_str_list = [ut.to_json(metadata_dict) for metadata_dict in metadata_
↪dict_list]
>>> print(ut.repr2(metadata_str_list))
>>> metadata_str_list_ = ibs.get_annot_metadata(aid_list, return_raw=True)
>>> print(ut.repr2(metadata_str_list_))
>>> assert metadata_str_list == metadata_str_list_
```

wbia.control.manual_annot_funcs.**set_annot_multiple**(*ibs*, *aid_list*, *flag_list*)

Sets the annot all instances found bit

RESTful: Method: PUT URL: /api/annot/multiple/

wbia.control.manual_annot_funcs.**set_annot_name_rowids**(*ibs*, *aid_list*, *name_rowid_list*, *notify_wildbook=True*, *assert_wildbook=False*)

name_rowid_list -> annot.name_rowid[aid_list]

Sets names/nids of a list of annotations.

Parameters

- **aid_list** (*list*) –
- **name_rowid_list** (*list*) –

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[0:2]
>>> # check clean state
>>> ut.assert_eq(ibs.get_annot_names(aid_list), ['____', 'easy'])
>>> ut.assert_eq(ibs.get_annot_exemplar_flags(aid_list), [0, 1])
>>> # run function
>>> name_list = ['easy', '____']
>>> name_rowid_list = ibs.get_name_rowids_from_text(name_list)
>>> ibs.set_annot_name_rowids(aid_list, name_rowid_list)
>>> # check results
>>> ut.assert_eq(ibs.get_annot_names(aid_list), ['easy', '____'])
>>> ut.assert_eq(ibs.get_annot_exemplar_flags(aid_list), [0, 0])
>>> # restore database state
>>> ibs.set_annot_names(aid_list, ['____', 'easy'])
>>> ibs.set_annot_exemplar_flags(aid_list, [0, 1])
>>> ut.assert_eq(ibs.get_annot_names(aid_list), ['____', 'easy'])
>>> ut.assert_eq(ibs.get_annot_exemplar_flags(aid_list), [0, 1])
```

wbia.control.manual_annot_funcs.**set_annot_name_texts**(*ibs*, *aid_list*, *name_list*)
alias

RESTful: Method: GET URL: /api/annot/name/

wbia.control.manual_annot_funcs.**set_annot_names**(*ibs*, *aid_list*, *name_list*, ***kwargs*)
Sets the attrlbl_value of type(INDIVIDUAL_KEY) Sets names/nids of a list of annotations.

CommandLine: python -m wbia.control.manual_annot_funcs –test-set_annot_names –enableall

RESTful: Method: PUT URL: /api/annot/name/

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> name_list1 = get_annot_names(ibs, aid_list)
>>> name_list2 = [name + '_TESTAUG' for name in name_list1]
>>> set_annot_names(ibs, aid_list, name_list2)
>>> name_list3 = get_annot_names(ibs, aid_list)
>>> set_annot_names(ibs, aid_list, name_list1)
>>> name_list4 = get_annot_names(ibs, aid_list)
>>> assert name_list2 == name_list3
>>> assert name_list4 == name_list1
>>> assert name_list4 != name_list2
>>> print(result)
```

wbia.control.manual_annot_funcs.**set_annot_notes**(*ibs*, *aid_list*, *notes_list*)
Sets annotation notes

RESTful: Method: PUT URL: /api/annot/note/

wbia.control.manual_annot_funcs.**set_annot_parent_rowid**(*ibs*, *aid_list*, *parent_aid_list*)
Sets the annotation's parent aid. TODO DEPRICATE IN FAVOR OF SEPARATE PARTS TABLE

RESTful: Method: PUT URL: /api/annot/parent/rowid/

wbia.control.manual_annot_funcs.**set_annot_qualities**(*ibs*, *aid_list*, *annot_quality_list*)
annot_quality_list -> annot.annot_quality[aid_list]

A quality is an integer representing the following types:

Parameters

- **aid_list** –
- **annot_quality_list** –

SeeAlso: wbia.const.QUALITY_INT_TO_TEXT

RESTful: Method: PUT URL: /api/annot/quality/

wbia.control.manual_annot_funcs.**set_annot_quality_texts**(*ibs*, *aid_list*, *quality_text_list*)
Auto-docstr for 'set_annot_quality_texts'

RESTful: Method: PUT URL: /api/annot/quality/text/

wbia.control.manual_annot_funcs.**set_annot_reviewed**(*ibs*, *aid_list*, *reviewed_list*)
Sets the annot all instances found bit

RESTful: Method: PUT URL: /api/annot/reviewed/

wbia.control.manual_annot_funcs.**set_annot_sex**(*ibs*, *aid_list*, *name_sex_list*, *eager=True*, *nInput=None*)
Auto-docstr for 'set_annot_sex'

RESTful: Method: PUT URL: /api/annot/sex/

wbia.control.manual_annot_funcs.**set_annot_sex_texts**(*ibs*, *aid_list*, *name_sex_text_list*, *eager=True*, *nInput=None*)
Auto-docstr for 'set_annot_sex_texts'

RESTful: Method: PUT URL: /api/annot/sex/text/

wbia.control.manual_annot_funcs.**set_annot_species**(*ibs*, *aid_list*, *species_text_list*, ***kwargs*)
Sets species/speciesids of a list of annotations. Convenience function for set_annot_lblannot_from_value

RESTful: Method: PUT URL: /api/annot/species/

wbia.control.manual_annot_funcs.**set_annot_species_and_notify**(*ibs*, **args*, ***kwargs*)

wbia.control.manual_annot_funcs.**set_annot_species_rowids**(*ibs*, *aid_list*, *species_rowid_list*)
species_rowid_list -> annot.species_rowid[aid_list]

Sets species/speciesids of a list of annotations.

Parameters

- **aid_list** –
- **species_rowid_list** –

RESTful: Method: PUT URL: /api/annot/species/rowid/

wbia.control.manual_annot_funcs.**set_annot_staged_metadata**(*ibs*, *aid_list*, *metadata_dict_list*)

Sets the annot's staged metadata using a metadata dictionary

RESTful: Method: PUT URL: /api/annot/staged/metadata/

CommandLine: python -m wbia.control.manual_annot_funcs –test-set_annot_metadata

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> import random
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[0:1]
>>> metadata_dict_list = [
>>>     {'test': random.uniform(0.0, 1.0)},
>>> ]
>>> print(ut.repr2(metadata_dict_list))
>>> ibs.set_annot_metadata(aid_list, metadata_dict_list)
>>> # verify results
>>> metadata_dict_list_ = ibs.get_annot_metadata(aid_list)
>>> print(ut.repr2(metadata_dict_list_))
>>> assert metadata_dict_list == metadata_dict_list_
>>> metadata_str_list = [ut.to_json(metadata_dict) for metadata_dict in metadata_
↪dict_list]
>>> print(ut.repr2(metadata_str_list))
>>> metadata_str_list_ = ibs.get_annot_metadata(aid_list, return_raw=True)
>>> print(ut.repr2(metadata_str_list_))
>>> assert metadata_str_list == metadata_str_list_
```

wbia.control.manual_annot_funcs.**set_annot_staged_user_ids**(*ibs*, *aid_list*, *user_id_list*)

Sets the staged annotation user id

RESTful: Method: PUT URL: /api/annot/staged/user/

wbia.control.manual_annot_funcs.**set_annot_staged_uuids**(*ibs*, *aid_list*, *annot_uuid_list*)

Returns all nids of known animals (does not include unknown names)

Return type list_ (list)

wbia.control.manual_annot_funcs.**set_annot_static_encounter**(*ibs*, *aids*, *vals*)

wbia.control.manual_annot_funcs.**set_annot_tag_text**(*ibs*, *aid_list*, *annot_tags_list*, *duplicate_behavior='error'*)

annot_tags_list -> annot.annot_tags[aid_list]

Parameters

- **aid_list** –
- **annot_tags_list** –

wbia.control.manual_annot_funcs.**set_annot_thetas**(*ibs*, *aid_list*, *theta_list*, *delete_thumbs=True*, *update_visual_uuids=True*, *notify_root=True*)

Sets thetas of a list of chips by aid

RESTful: Method: PUT URL: /api/annot/theta/

wbia.control.manual_annot_funcs.**set_annot_verts**(*ibs*, *aid_list*, *verts_list*, *theta_list=None*, *interest_list=None*, *canonical_list=None*, *delete_thumbs=True*, *update_visual_uuids=True*, *notify_root=True*)

Sets the vertices [(x, y), ...] of a list of chips by aid

RESTful: Method: PUT URL: /api/annot/vert/

wbia.control.manual_annot_funcs.**set_annot_viewpoint_code**(*ibs*, *aids*, *view_codes*, *_code_update=True*)

wbia.control.manual_annot_funcs.**set_annot_viewpoint_int**(*ibs*, *aids*, *view_ints*, *_code_update=True*)

wbia.control.manual_annot_funcs.**set_annot_viewpoints**(*ibs*, *aid_list*, *viewpoint_list*, *purge_cache=True*, *only_allow_known=True*, *_yaw_update=False*, *_code_update=True*)

Sets the viewpoint of the annotation

RESTful: Method: PUT URL: /api/annot/viewpoint/

wbia.control.manual_annot_funcs.**set_annot_yaw_texts**(*ibs*, *aid_list*, *yaw_text_list*)

Auto-docstr for 'set_annot_yaw_texts'

DEPRICATE

RESTful: Method: PUT URL: /api/annot/yaw/text/

wbia.control.manual_annot_funcs.**set_annot_yaws**(*ibs*, *aid_list*, *yaw_list*, *input_is_degrees=False*)

Sets the yaw of a list of chips by aid

DEPRICATE

A yaw is the yaw of the annotation in radians yaw is inverted. Will be fixed soon.

Note:

The following views have these angles of yaw: left side - 0.00 tau radians front side - 0.25 tau radians right side - 0.50 tau radians back side - 0.75 tau radians (tau = 2 * pi)

SeeAlso: wbia.const.VIEWTEXT_TO_YAW_RADIANS

References

http://upload.wikimedia.org/wikipedia/commons/7/7e/Rollpitchyawplain.png

RESTful: Method: PUT URL: /api/annot/yaw/

wbia.control.manual_annot_funcs.**testdata_ibs**()

Auto-docstr for 'testdata_ibs'

wbia.control.manual_annot_funcs.**update_annot_rotate_90**(*ibs*, *aid_list*, *direction*)

wbia.control.manual_annot_funcs.**update_annot_rotate_left_90**(*ibs*, *aid_list*)

wbia.control.manual_annot_funcs.**update_annot_rotate_right_90**(*ibs*, *aid_list*)

wbia.control.manual_annot_funcs.**update_annot_semantic_uuids**(*ibs*, *aid_list*, *_visual_infotup=None*)

Ensures that annots have the proper semantic uuids

wbia.control.manual_annot_funcs.**update_annot_visual_uuids**(*ibs*, *aid_list*)

Ensures that annots have the proper visual uuids

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*list*) – list of annotation rowids

CommandLine: python -m wbia.control.manual_annot_funcs update_annot_visual_uuids –db PZ_Master1 python -m wbia.control.manual_annot_funcs update_annot_visual_uuids python -m wbia update_annot_visual_uuids –db PZ_Master1 python -m wbia update_annot_visual_uuids –db PZ_Master0 python -m wbia update_annot_visual_uuids –db PZ_MTEST

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annot_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ibs._get_all_aids()[0:1]
>>> update_annot_visual_uuids(ibs, aid_list)
>>> result = ibs.get_annot_visual_uuids(aid_list)[0]
>>> print(result)
8687dcb6-1f1f-fdd3-8b72-8f36f9f41905
```

1.2.14 wbia.control.manual_annotgroup_funcs module

Autogenerated IBEISController functions

TemplateInfo: autogen_time = 13:31:28 2015/04/28 autogen_key = annotgroup

ToRegenerate: python -m wbia.templates.template_generator –key annotgroup –Tcfg with_web_api=True with_api_cache=False with_deleters=True no_extern_deleters=True –diff python -m wbia.templates.template_generator –key annotgroup –Tcfg with_web_api=True with_api_cache=False with_deleters=True no_extern_deleters=True –write

wbia.control.manual_annotgroup_funcs.**add_annotgroup**(*ibs*, *annotgroup_uuid_list*, *annotgroup_text_list*, *annotgroup_note_list*)

Returns returns annotgroup_rowid_list of added (or already existing annotgroups)

TemplateInfo: Tadder_native tbl = annotgroup

wbia.control.manual_annotgroup_funcs.**delete_annotgroup**(*ibs*, *annotgroup_rowid_list*, *config2_=None*)

annotgroup.delete(annotgroup_rowid_list)

delete annotgroup rows

Parameters **annotgroup_rowid_list** –

Returns num_deleted

Return type int

TemplateInfo: Tdeleter_native_tbl tbl = annotgroup

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_annotgroup_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> annotgroup_rowid_list = ibs._get_all_annotgroup_rowids()[:2]
>>> num_deleted = ibs.delete_annotgroup(annotgroup_rowid_list)
>>> print('num_deleted = %r' % (num_deleted,))
```

wbia.control.manual_annotgroup_funcs.**get_annotgroup_gar_rowids**(*ibs*, *annotgroup_rowid_list*, *eager=True*, *nInput=None*)

Auto-docstr for 'get_annotgroup_gar_rowids'

RESTful: Method: GET URL: /api/annotgroup/gar/rowids/

wbia.control.manual_annotgroup_funcs.**get_annotgroup_note**(*ibs*, *annotgroup_rowid_list*, *eager=True*, *nInput=None*)

annotgroup_note_list <- annotgroup.annotgroup_note[annotgroup_rowid_list]

gets data from the "native" column "annotgroup_note" in the "annotgroup" table

Parameters **annotgroup_rowid_list** (*list*) –

Returns annotgroup_note_list

Return type list

TemplateInfo: Tgetter_table_column col = annotgroup_note tbl = annotgroup

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annotgroup_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> annotgroup_rowid_list = ibs._get_all_annotgroup_rowids()
>>> eager = True
>>> annotgroup_note_list = ibs.get_annotgroup_note(annotgroup_rowid_list,␣
↪eager=eager)
>>> assert len(annotgroup_rowid_list) == len(annotgroup_note_list)
```

wbia.control.manual_annotgroup_funcs.**get_annotgroup_rowid_from_superkey**(*ibs*, *annotgroup_text_list*, *eager=True*, *nInput=None*)

annotgroup_rowid_list <- annotgroup[annotgroup_text_list]

Parameters **lists** (*superkey*) – annotgroup_text_list

Returns annotgroup_rowid_list

TemplateInfo: Tgetter_native_rowid_from_superkey tbl = annotgroup

wbia.control.manual_annotgroup_funcs.**get_annotgroup_text**(*ibs*, *annotgroup_rowid_list*, *eager=True*, *nInput=None*)

annotgroup_text_list <- annotgroup.annotgroup_text[annotgroup_rowid_list]

gets data from the "native" column "annotgroup_text" in the "annotgroup" table

Parameters **annotgroup_rowid_list** (*list*) –

Returns annotgroup_text_list

Return type list

TemplateInfo: Tgetter_table_column col = annotgroup_text tbl = annotgroup

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annotgroup_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> annotgroup_rowid_list = ibs._get_all_annotgroup_rowids()
>>> eager = True
>>> annotgroup_text_list = ibs.get_annotgroup_text(annotgroup_rowid_list,␣
↪eager=eager)
>>> assert len(annotgroup_rowid_list) == len(annotgroup_text_list)
```

wbia.control.manual_annotgroup_funcs.**get_annotgroup_uuid**(*ibs*, *annotgroup_rowid_list*, *eager=True*, *nInput=None*)

annotgroup_uuid_list <- annotgroup.annotgroup_uuid[annotgroup_rowid_list]

gets data from the "native" column "annotgroup_uuid" in the "annotgroup" table

Parameters **annotgroup_rowid_list** (*list*) –

Returns annotgroup_uuid_list

Return type list

TemplateInfo: Tgetter_table_column col = annotgroup_uuid tbl = annotgroup

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annotgroup_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> annotgroup_rowid_list = ibs._get_all_annotgroup_rowids()
>>> eager = True
>>> annotgroup_uuid_list = ibs.get_annotgroup_uuid(annotgroup_rowid_list,␣
↪eager=eager)
>>> assert len(annotgroup_rowid_list) == len(annotgroup_uuid_list)
```

wbia.control.manual_annotgroup_funcs.**set_annotgroup_note**(*ibs*, *annotgroup_rowid_list*, *annotgroup_note_list*, *duplicate_behavior='error'*)

annotgroup_note_list -> annotgroup.annotgroup_note[annotgroup_rowid_list]

Parameters

- **annotgroup_rowid_list** –
- **annotgroup_note_list** –

TemplateInfo: Tsetter_native_column tbl = annotgroup col = annotgroup_note

wbia.control.manual_annotgroup_funcs.**set_annotgroup_uuid**(*ibs*, *annotgroup_rowid_list*, *annotgroup_uuid_list*, *duplicate_behavior='error'*)

annotgroup_uuid_list -> annotgroup.annotgroup_uuid[annotgroup_rowid_list]

Parameters

- **annotgroup_rowid_list** –
- **annotgroup_uuid_list** –

TemplateInfo: Tsetter_native_column tbl = annotgroup col = annotgroup_uuid

wbia.control.manual_annotgroup_funcs.**testdata_ibs**(*defaultdb='testdb1'*)

1.2.15 wbia.control.manual_annotmatch_funcs module

Autogenerated IBEISController functions

TemplateInfo: autogen_time = 11:34:25 2016/01/05 autogen_key = annotmatch

ToRegenerate: python -m wbia.templates.template_generator –key annotmatch –Tcfg with_web_api=False with_api_cache=False with_deleters=True no_extern_deleters=True –diff python -m wbia.templates.template_generator –key annotmatch –Tcfg with_web_api=False with_api_cache=False with_deleters=True no_extern_deleters=True –write

wbia.control.manual_annotmatch_funcs.**add_annotmatch**(*ibs*, *aid1_list*, *aid2_list*, *annotmatch_evidence_decision_list=None*, *annotmatch_meta_decision_list=None*, *annotmatch_confidence_list=None*, *annotmatch_tag_text_list=None*, *annotmatch_reviewer_list=None*, *annotmatch_posixtime_modified_list=None*, *anotmatch_count_list=None*)

Returns returns annotmatch_rowid_list of added (or already existing annotmatchs)

TemplateInfo: Tadder_native tbl = annotmatch

wbia.control.manual_annotmatch_funcs.**delete_annotmatch**(*ibs*, *annotmatch_rowid_list*)

annotmatch.delete(annotmatch_rowid_list)

delete annotmatch rows

Parameters **annotmatch_rowid_list** –

Returns num_deleted

Return type int

TemplateInfo: Tdeleter_native_tbl tbl = annotmatch

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_annotmatch_funcs import *  # NOQA
>>> ibs, config2_ = testdata_annotmatch()
>>> annotmatch_rowid_list = ibs._get_all_annotmatch_rowids()[:2]
>>> num_deleted = ibs.delete_annotmatch(annotmatch_rowid_list)
>>> print('num_deleted = %r' % (num_deleted,))
```

wbia.control.manual_annotmatch_funcs.**get_annotmatch_aid1**(*ibs*, *annotmatch_rowid_list*, *eager=True*, *nInput=None*)

aid1_list <- annotmatch.aid1[annotmatch_rowid_list]

gets data from the "native" column "aid1" in the "annotmatch" table

Parameters **annotmatch_rowid_list** (*list*) –

Returns aid1_list

Return type list

TemplateInfo: Tgetter_table_column col = aid1 tbl = annotmatch

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annotmatch_funcs import *  # NOQA
>>> ibs, config2_ = testdata_annotmatch()
>>> annotmatch_rowid_list = ibs._get_all_annotmatch_rowids()
>>> eager = True
>>> aid1_list = ibs.get_annotmatch_aid1(annotmatch_rowid_list, eager=eager)
>>> assert len(annotmatch_rowid_list) == len(aid1_list)
```

wbia.control.manual_annotmatch_funcs.**get_annotmatch_aid2**(*ibs*, *annotmatch_rowid_list*, *eager=True*, *nInput=None*)

aid2_list <- annotmatch.aid2[annotmatch_rowid_list]

gets data from the "native" column "aid2" in the "annotmatch" table

Parameters **annotmatch_rowid_list** (*list*) –

Returns aid2_list

Return type list

TemplateInfo: Tgetter_table_column col = aid2 tbl = annotmatch

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annotmatch_funcs import *  # NOQA
>>> ibs, config2_ = testdata_annotmatch()
>>> annotmatch_rowid_list = ibs._get_all_annotmatch_rowids()
>>> eager = True
>>> aid2_list = ibs.get_annotmatch_aid2(annotmatch_rowid_list, eager=eager)
>>> assert len(annotmatch_rowid_list) == len(aid2_list)
```

wbia.control.manual_annotmatch_funcs.**get_annotmatch_confidence**(*ibs*, *annotmatch_rowid_list*, *eager=True*, *nInput=None*)

annotmatch_confidence_list <- annotmatch.annotmatch_confidence[annotmatch_rowid_list]

gets data from the "native" column "annotmatch_confidence" in the "annotmatch" table

Parameters **annotmatch_rowid_list** (*list*) –

Returns annotmatch_confidence_list

Return type list

TemplateInfo: Tgetter_table_column col = annotmatch_confidence tbl = annotmatch

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annotmatch_funcs import *  # NOQA
>>> ibs, config2_ = testdata_annotmatch()
>>> annotmatch_rowid_list = ibs._get_all_annotmatch_rowids()
>>> eager = True
>>> annotmatch_confidence_list = ibs.get_annotmatch_confidence(annotmatch_rowid_
↪list, eager=eager)
>>> assert len(annotmatch_rowid_list) == len(annotmatch_confidence_list)
```

wbia.control.manual_annotmatch_funcs.**get_annotmatch_count**(*ibs*, *annotmatch_rowid_list*, *eager=True*, *nInput=None*)

wbia.control.manual_annotmatch_funcs.**get_annotmatch_evidence_decision**(*ibs*, *annotmatch_rowid_list*, *eager=True*, *nInput=None*)

gets data from the "native" column "annotmatch_evidence_decision" in the "annotmatch" table

Parameters **annotmatch_rowid_list** (*list*) –

Returns annotmatch_evidence_decision_list

Return type list

TemplateInfo: Tgetter_table_column col = annotmatch_evidence_decision tbl = annotmatch

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annotmatch_funcs import *  # NOQA
>>> ibs, config2_ = testdata_annotmatch()
>>> annotmatch_rowid_list = ibs._get_all_annotmatch_rowids()
>>> eager = True
>>> decisions = ibs.get_annotmatch_evidence_decision(annotmatch_rowid_list,␣
↪eager=eager)
>>> assert len(annotmatch_rowid_list) == len(decisions)
```

wbia.control.manual_annotmatch_funcs.**get_annotmatch_meta_decision**(*ibs*, *annotmatch_rowid_list*, *eager=True*, *nInput=None*)

wbia.control.manual_annotmatch_funcs.**get_annotmatch_posixtime_modified**(*ibs*, *annotmatch_rowid_list*, *eager=True*, *nInput=None*)

annotmatch_posixtime_modified_list <- annotmatch.annotmatch_posixtime_modified[annotmatch_rowid_list]

gets data from the "native" column "annotmatch_posixtime_modified" in the "annotmatch" table

Parameters **annotmatch_rowid_list** (*list*) –

Returns annotmatch_posixtime_modified_list

Return type list

TemplateInfo: Tgetter_table_column col = annotmatch_posixtime_modified tbl = annotmatch

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annotmatch_funcs import *  # NOQA
>>> ibs, config2_ = testdata_annotmatch()
>>> annotmatch_rowid_list = ibs._get_all_annotmatch_rowids()
>>> eager = True
>>> annotmatch_posixtime_modified_list = ibs.get_annotmatch_posixtime_
↪modified(annotmatch_rowid_list, eager=eager)
>>> assert len(annotmatch_rowid_list) == len(annotmatch_posixtime_modified_list)
```

wbia.control.manual_annotmatch_funcs.**get_annotmatch_reviewer**(*ibs*, *annotmatch_rowid_list*, *eager=True*, *nInput=None*)

annotmatch_reviewer_list <- annotmatch.annotmatch_reviewer[annotmatch_rowid_list]

gets data from the "native" column "annotmatch_reviewer" in the "annotmatch" table

Parameters **annotmatch_rowid_list** (*list*) –

Returns annotmatch_reviewer_list

Return type list

TemplateInfo: Tgetter_table_column col = annotmatch_reviewer tbl = annotmatch

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annotmatch_funcs import *  # NOQA
>>> ibs, config2_ = testdata_annotmatch()
>>> annotmatch_rowid_list = ibs._get_all_annotmatch_rowids()
>>> eager = True
>>> annotmatch_reviewer_list = ibs.get_annotmatch_reviewer(annotmatch_rowid_list,␣
↪eager=eager)
>>> assert len(annotmatch_rowid_list) == len(annotmatch_reviewer_list)
```

wbia.control.manual_annotmatch_funcs.**get_annotmatch_rowid**(*ibs*, *annotmatch_rowid_list*, *eager=True*, *nInput=None*)

annotmatch_rowid_list <- annotmatch.annotmatch_rowid[annotmatch_rowid_list]

gets data from the "native" column "annotmatch_rowid" in the "annotmatch" table

Parameters **annotmatch_rowid_list** (*list*) –

Returns annotmatch_rowid_list

Return type list

TemplateInfo: Tgetter_table_column col = annotmatch_rowid tbl = annotmatch

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annotmatch_funcs import *  # NOQA
>>> ibs, config2_ = testdata_annotmatch()
>>> annotmatch_rowid_list = ibs._get_all_annotmatch_rowids()
>>> eager = True
>>> annotmatch_rowid_list = ibs.get_annotmatch_rowid(annotmatch_rowid_list,␣
↪eager=eager)
>>> assert len(annotmatch_rowid_list) == len(annotmatch_rowid_list)
```

wbia.control.manual_annotmatch_funcs.**get_annotmatch_rowid_from_superkey**(*ibs*, *aid1_list*, *aid2_list*, *eager=True*, *nInput=None*)

annotmatch_rowid_list <- annotmatch[aid1_list, aid2_list]

Parameters **lists** (*superkey*) – aid1_list, aid2_list

Returns annotmatch_rowid_list

TemplateInfo: Tgetter_native_rowid_from_superkey tbl = annotmatch

wbia.control.manual_annotmatch_funcs.**get_annotmatch_tag_text**(*ibs*, *annotmatch_rowid_list*, *eager=True*, *nInput=None*)

annotmatch_tag_text_list <- annotmatch.annotmatch_tag_text[annotmatch_rowid_list]

gets data from the "native" column "annotmatch_tag_text" in the "annotmatch" table

Parameters **annotmatch_rowid_list** (*list*) –

Returns annotmatch_tag_text_list

Return type list

TemplateInfo: Tgetter_table_column col = annotmatch_tag_text tbl = annotmatch

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_annotmatch_funcs import *  # NOQA
>>> ibs, config2_ = testdata_annotmatch()
>>> annotmatch_rowid_list = ibs._get_all_annotmatch_rowids()
>>> eager = True
>>> annotmatch_tag_text_list = ibs.get_annotmatch_tag_text(annotmatch_rowid_list,␣
↪eager=eager)
>>> assert len(annotmatch_rowid_list) == len(annotmatch_tag_text_list)
```

wbia.control.manual_annotmatch_funcs.**set_annotmatch_confidence**(*ibs*, *annotmatch_rowid_list*, *annotmatch_confidence_list*, *duplicate_behavior='error'*)

annotmatch_confidence_list -> annotmatch.annotmatch_confidence[annotmatch_rowid_list]

Parameters

- **annotmatch_rowid_list** –
- **annotmatch_confidence_list** –

TemplateInfo: Tsetter_native_column tbl = annotmatch col = annotmatch_confidence

wbia.control.manual_annotmatch_funcs.**set_annotmatch_count**(*ibs*, *annotmatch_rowid_list*, *annotmatch_count_list*, *duplicate_behavior='error'*)

wbia.control.manual_annotmatch_funcs.**set_annotmatch_evidence_decision**(*ibs*, *annotmatch_rowid_list*, *annotmatch_evidence_decision_list*, *duplicate_behavior='error'*)

wbia.control.manual_annotmatch_funcs.**set_annotmatch_meta_decision**(*ibs*, *annotmatch_rowid_list*, *annotmatch_meta_decision_list*, *duplicate_behavior='error'*)

wbia.control.manual_annotmatch_funcs.**set_annotmatch_posixtime_modified**(*ibs*, *annotmatch_rowid_list*, *annotmatch_posixtime_modified_li*, *duplicate_behavior='error'*)

annotmatch_posixtime_modified_list -> annotmatch.annotmatch_posixtime_modified[annotmatch_rowid_list]

Parameters

- **annotmatch_rowid_list** –
- **annotmatch_posixtime_modified_list** –

TemplateInfo: Tsetter_native_column tbl = annotmatch col = annotmatch_posixtime_modified

wbia.control.manual_annotmatch_funcs.**set_annotmatch_reviewer**(*ibs*, *annotmatch_rowid_list*, *annotmatch_reviewer_list*, *duplicate_behavior='error'*)

annotmatch_reviewer_list -> annotmatch.annotmatch_reviewer[annotmatch_rowid_list]

Parameters

- **annotmatch_rowid_list** –
- **annotmatch_reviewer_list** –

TemplateInfo: Tsetter_native_column tbl = annotmatch col = annotmatch_reviewer

wbia.control.manual_annotmatch_funcs.**set_annotmatch_tag_text**(*ibs*, *annotmatch_rowid_list*, *annotmatch_tag_text_list*, *duplicate_behavior='error'*)

annotmatch_tag_text_list -> annotmatch.annotmatch_tag_text[annotmatch_rowid_list]

Parameters

- **annotmatch_rowid_list** –
- **annotmatch_tag_text_list** –

TemplateInfo: Tsetter_native_column tbl = annotmatch col = annotmatch_tag_text

wbia.control.manual_annotmatch_funcs.**testdata_annotmatch**(*defaultdb='testdb1'*)

1.2.16 wbia.control.manual_chip_funcs module

wbia.control.manual_chip_funcs.**delete_annot_chips**(*ibs*, *aid_list*, *config2_=None*, *fallback=True*)

Clears annotation data (does not remove the annotation)

RESTful: Method: DELETE URL: /api/chip/

wbia.control.manual_chip_funcs.**delete_part_chips**(*ibs*, *part_rowid_list*, *config2_=None*)

Clears part data

RESTful: Method: DELETE URL: /api/pchip/

wbia.control.manual_chip_funcs.**get_annot_chip_dlensqrd**(*ibs*, *aid_list*, *config2_=None*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*list*) –

Returns topx2_dlen_sqrd

Return type list

CommandLine: python -m wbia.control.manual_chip_funcs get_annot_chip_dlensqrd

RESTful: Method: GET URL: /api/chip/dlensqrd/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_chip_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> config2_ = {'dim_size': 450, 'resize_dim': 'area'}
>>> topx2_dlen_sqrd = ibs.get_annot_chip_dlensqrd(aid_list, config2_=config2_)
>>> result = str(topx2_dlen_sqrd)
>>> print(result)
[435409, 476505, 422500, 422500, 422500, 437924, 405000, 405000, 447805, 420953,␣
↪405008, 406265, 512674]
```

wbia.control.manual_chip_funcs.**get_annot_chip_fpath**(*ibs*, *aid_list*, *ensure=True*, *config2_=None*, *check_external_storage=False*, *num_retries=1*)

Returns the cached chip uri based off of the current configuration.

Returns cfpaths defined by ANNOTATIONs

Return type chip_fpath_list (list)

RESTful: Method: GET URL: /api/chip/fpath/

wbia.control.manual_chip_funcs.**get_annot_chip_sizes**(*ibs*, *aid_list*, *ensure=True*, *config2_=None*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*int*) – list of annotation ids
- **ensure** (*bool*) – eager evaluation if True

Returns chipsz_list - the (width, height) of computed annotation chips.

Return type list

CommandLine: python -m wbia.control.manual_chip_funcs get_annot_chip_sizes

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_chip_funcs import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[0:3]
>>> ensure = True
>>> config2_ = {'dim_size': 450, 'resize_dim': 'area'}
>>> # execute function
>>> chipsz_list = get_annot_chip_sizes(ibs, aid_list, ensure, config2_=config2_)
>>> # verify results
>>> result = str(chipsz_list)
>>> print(result)
[(545, 372), (603, 336), (520, 390)]
```

wbia.control.manual_chip_funcs.**get_annot_chip_thumb_path2**(*ibs*, *aid_list*, *thumbsize=None*, *config=None*)

get chip thumb info The return type of this is interpreted and computed in ~/code/guitool/guitool/api_thumb_delegate.py

Parameters

- **aid_list** (*list*) –
- **thumbsize** (*int*) –

Returns thumbtup_list - [(thumb_path, img_path, imgsize, bboxes, thetas)]

Return type list

CommandLine: python -m wbia.control.manual_chip_funcs –test-get_annot_chip_thumbtup

RESTful: Method: GET URL: /api/chip/thumbtup/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_chip_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[1:2]
>>> thumbsize = 128
>>> result = get_annot_chip_thumbtup(ibs, aid_list, thumbsize)
>>> print(result)
```

wbia.control.manual_chip_funcs.**get_annot_chip_thumbpath**(*ibs*, *aid_list*, *thumbsize=None*, *config2_=None*)

just constructs the path. does not compute it. that is done by api_thumb_delegate

RESTful: Method: GET URL: /api/chip/thumbpath/

wbia.control.manual_chip_funcs.**get_annot_chip_thumbtup**(*ibs*, *aid_list*, *thumbsize=None*, *config2_=None*)

get chip thumb info The return type of this is interpreted and computed in ~/code/guitool/guitool/api_thumb_delegate.py

Parameters

- **aid_list** (*list*) –
- **thumbsize** (*int*) –

Returns thumbtup_list - [(thumb_path, img_path, imgsize, bboxes, thetas)]

Return type list

CommandLine: python -m wbia.control.manual_chip_funcs –test-get_annot_chip_thumbtup

RESTful: Method: GET URL: /api/chip/thumbtup/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_chip_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[1:2]
>>> thumbsize = 128
>>> result = get_annot_chip_thumbtup(ibs, aid_list, thumbsize)
>>> print(result)
```

wbia.control.manual_chip_funcs.**get_annot_chips**(*ibs*, *aid_list*, *config2_=None*, *ensure=True*, *verbose=False*, *eager=True*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*int*) – list of annotation ids
- **ensure** (*bool*) – eager evaluation if True
- **config2** (QueryRequest) – query request object with hyper-parameters

Returns chip_list

Return type list

CommandLine: python -m wbia.control.manual_chip_funcs get_annot_chips

RESTful: Method: GET URL: /api/chip/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_chip_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[0:5]
```

(continues on next page)

(continued from previous page)

```
>>> config2_ = {'dim_size': 450, 'resize_dim': 'area'}
>>> chip_list = get_annot_chips(ibs, aid_list, config2_)
>>> chip_sum_list = [chip.sum() for chip in chip_list]
>>> target = [96053684, 65140000, 67223205, 109367378, 73995663]
>>> ut.assert_almost_eq(chip_sum_list, target, 15000)
>>> print(chip_sum_list)
```

wbia.control.manual_chip_funcs.**get_part_chips**(*ibs*, *part_rowid_list*, *config2_=None*, *ensure=True*, *verbose=False*, *eager=True*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **part_rowid_list** (*int*) – list of part ids
- **ensure** (*bool*) – eager evaluation if True
- **config2** (QueryRequest) – query request object with hyper-parameters

Returns chip_list

Return type list

CommandLine: python -m wbia.control.manual_chip_funcs get_part_chips

RESTful: Method: GET URL: /api/pchip/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_chip_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> aid_list = aid_list[:10]
>>> bbox_list = ibs.get_annot_bboxes(aid_list)
>>> bbox_list = [
>>>     (xtl + 100, ytl + 100, w - 100, h - 100)
>>>     for xtl, ytl, w, h in bbox_list
>>> ]
>>> part_rowid_list = ibs.add_parts(aid_list, bbox_list=bbox_list)
>>> config2_ = {'dim_size': 450, 'resize_dim': 'area'}
>>> chip_list = get_part_chips(ibs, part_rowid_list, config2_)
>>> chip_sum_list = [chip.sum() for chip in chip_list]
>>> target = [86765003, 62005000, 61333186, 111424764, 63590900, 51397198,␣
↪139395045, 84100000, 41254190, 89657450]
>>> ut.assert_almost_eq(chip_sum_list, target, 50000)
>>> print(chip_sum_list)
```

wbia.control.manual_chip_funcs.**testdata_ibs**()

1.2.17 wbia.control.manual_feat_funcs module

python -c “import utool as ut; ut.write_modscript_alias(‘Tgen.sh’, ‘wbia.templates.template_generator’)” sh Tgen.sh –key feat –Tcfg with_setters=False with_getters=True with_adders=True –modfname manual_feat_funcs sh Tgen.sh –key feat –Tcfg with_deleters=True –autogen_modname manual_feat_funcs

wbia.control.manual_feat_funcs.**delete_annot_feats**(*ibs*, *aid_list*, *config2_=None*)
annot.feat.delete(aid_list)

Parameters **aid_list** –

TemplateInfo: Tdeleter_rl_depenant root = annot leaf = feat

CommandLine: python -m wbia.control.manual_feat_funcs –test-delete_annot_feats python -m wbia.control.manual_feat_funcs –test-delete_annot_feats –verb-control

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_feat_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> aid_list = ibs._get_all_aids()[:1]
>>> fids_list = ibs.get_annot_feat_rowids(aid_list, config2_=config2_,␣
↪ensure=True)
>>> num_deleted1 = ibs.delete_annot_feats(aid_list, config2_=config2_)
>>> ut.assert_eq(num_deleted1, len(fids_list))
>>> num_deleted2 = ibs.delete_annot_feats(aid_list, config2_=config2_)
>>> ut.assert_eq(num_deleted2, 0)
```

wbia.control.manual_feat_funcs.**get_annot_feat_rowids**(*ibs*, *aid_list*, *ensure=True*, *eager=True*, *nInput=None*, *config2_=None*, *num_retries=1*)

CommandLine: python -m wbia.control.manual_feat_funcs get_annot_feat_rowids –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.algo.hots.query_request import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aids = ibs.get_valid_aids()[0:3]
>>> config2_ = {}
>>> ibs.delete_annot_feats(aids, config2_=config2_)  # Remove the chips
>>> ut.remove_file_list(ibs.get_annot_chip_fpath(aids, config2_=config2_))
>>> qfids = ibs.get_annot_feat_rowids(aids, ensure=True, config2_=config2_)
```

wbia.control.manual_feat_funcs.**get_annot_kpts**(*ibs*, *aid_list*, *ensure=True*, *eager=True*, *nInput=None*, *config2_=None*)

Parameters

- **aid_list** (*int*) – list of annotation ids
- **ensure** (*bool*) – eager evaluation if True
- **eager** (*bool*) –
- **nInput** (*None*) –
- **config2** (QueryRequest) – query request object with hyper-parameters

Returns annotation descriptor keypoints

Return type kpts_list (list)

CommandLine: python -m wbia.control.manual_feat_funcs –test-get_annot_kpts –show python -m wbia.control.manual_feat_funcs –test-get_annot_kpts –show –darken .9 python -m wbia.control.manual_feat_funcs –test-get_annot_kpts –show –darken .9 –verbose python -m wbia.control.manual_feat_funcs –test-get_annot_kpts –show –darken .9 –verbose –no-affine-invariance python -m wbia.control.manual_feat_funcs –test-get_annot_kpts –show –darken .9 –verbose –no-affine-invariance –scale_max=20 python -m wbia.control.manual_feat_funcs –test-get_annot_kpts –show –feat_type=hesaff+siam128 ipython -i – –show –feat_type=hesaff+siam128

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +SKIP
>>> from wbia.control.manual_feat_funcs import *  # NOQA
>>> import vtool as vt
>>> import numpy as np
>>> import wbia
>>> import wbia.viz.interact
>>> # build test data
>>> qreq1_ = wbia.testdata_qreq_(defaultdb='testdb1', p=['default:RI=True'])
>>> qreq2_ = wbia.testdata_qreq_(defaultdb='testdb1', p=['default:RI=False'])
>>> ibs = qreq1_.ibs
>>> aid_list = qreq1_.get_external_qaids()
>>> with ut.Indenter('[TEST_GET_ANNOT_KPTS]'):
...     print('qreq1 params: ' + qreq1_.qparams.feat_cfgstr)
...     print('qreq2 params: ' + qreq2_.qparams.feat_cfgstr)
...     print('id(qreq1): ' + str(id(qreq1_)))
...     print('id(qreq2): ' + str(id(qreq2_)))
...     #print('feat_config_rowid1 = %r' % (ibs.get_feat_config_rowid(config2_
↪=qreq1_.extern_query_config2),))
...     #print('feat_config_rowid2 = %r' % (ibs.get_feat_config_rowid(config2_
↪=qreq2_.extern_query_config2),))
>>> # Force recomputation of features
>>> with ut.Indenter('[DELETE1]'):
...     ibs.delete_annot_feats(aid_list, config2_=qreq1_.extern_query_config2)
>>> with ut.Indenter('[DELETE2]'):
...     ibs.delete_annot_feats(aid_list, config2_=qreq2_.extern_query_config2)
>>> eager, ensure, nInput = True, True, None
>>> # execute function
>>> with ut.Indenter('[GET1]'):
...     kpts1_list = get_annot_kpts(ibs, aid_list, ensure, eager, nInput, qreq1_.
↪extern_query_config2)
>>> with ut.Indenter('[GET2]'):
...     kpts2_list = get_annot_kpts(ibs, aid_list, ensure, eager, nInput, qreq2_.
↪extern_query_config2)
>>> # verify results
>>> assert not np.all(vt.get_oris(kpts1_list[0]) == 0)
>>> assert np.all(vt.get_oris(kpts2_list[0]) == 0)
>>> ut.quit_if_noshow()
>>> #wbia.viz.viz_chip.show_chip(ibs, aid_list[0], config2_=qreq1_, ori=True)
>>> wbia.viz.interact.interact_chip.ishow_chip(ibs, aid_list[0], config2_=qreq1_.
↪extern_query_config2, ori=True, fnum=1)
>>> wbia.viz.interact.interact_chip.ishow_chip(ibs, aid_list[0], config2_=qreq2_.
↪extern_query_config2, ori=True, fnum=2)
>>> ut.show_if_requested()
```

wbia.control.manual_feat_funcs.**get_annot_num_feats**(*ibs*, *aid_list*, *ensure=True*, *eager=True*, *nInput=None*, *config2_=None*, *_debug=False*)

Parameters **aid_list** (*list*) –

Returns num descriptors per annotation

Return type nFeats_list (list)

CommandLine: python -m wbia.control.manual_feat_funcs –test-get_annot_num_feats

Example

```
>>> # ENABLE_DOCTEST
>>> # this test might fail on different machines due to
>>> # determenism bugs in hesaff maybe? or maybe jpeg...
>>> # in which case its hopeless
>>> from wbia.control.manual_feat_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[0:3]
>>> config2_ = {'dim_size': 450, 'resize_dim': 'area'}
>>> nFeats_list = get_annot_num_feats(ibs, aid_list, ensure=True, config2_
↪=config2_, _debug=True)
>>> print('nFeats_list = %r' % (nFeats_list,))
>>> assert len(nFeats_list) == 3
>>> ut.assert_inbounds(nFeats_list[0], 1200, 1263)
>>> ut.assert_inbounds(nFeats_list[1],  900,  923)
>>> ut.assert_inbounds(nFeats_list[2], 1300, 1344)
```

Ignore: depc = ibs.depc_annot tablename = ‘feat’ input_rowids = aid_list colnames = ‘num_feats’ config = config2_

wbia.control.manual_feat_funcs.**get_annot_vecs**(*ibs*, *aid_list*, *ensure=True*, *eager=True*, *nInput=None*, *config2_=None*)

Returns annotation descriptor vectors

Return type vecs_list (list)

wbia.control.manual_feat_funcs.**testdata_ibs**()

1.2.18 wbia.control.manual_featweight_funcs module

wbia.control.manual_featweight_funcs.**get_annot_fgweight_rowids**(*ibs*, *aid_list*, *config2_=None*, *ensure=True*)

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **aid_list** (*list*) – list of annotation rowids
- **config2** (*dict*) – (default = None)
- **ensure** (*bool*) – eager evaluation if True(default = True)

CommandLine: python -m wbia.control.manual_featweight_funcs get_annot_fgweight_rowids

wbia.control.manual_featweight_funcs.**get_annot_fgweights**(*ibs*, *aid_list*, *config2_=None*, *ensure=True*)

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **aid_list** (*list*) – list of annotation rowids
- **config2** (*dict*) – (default = None)
- **ensure** (*bool*) – eager evaluation if True(default = True)

CommandLine: python -m wbia.control.manual_featweight_funcs get_annot_fgweights

Example

```
>>> # xdoctest: +REQUIRES(module:wbia_cnn)
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_featweight_funcs import *  # NOQA
>>> import wbia
>>> import numpy as np
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> aid_list = [1, 2]
>>> config2_ = None
>>> ensure = True
>>> fgws_list = get_annot_fgweights(ibs, aid_list, config2_, ensure)
>>> depth = ut.depth_profile(fgws_list)
>>> assert np.all(np.array(depth) > [1200, 1400])
>>> percent_ = (fgws_list[0] > .5).sum() / len(fgws_list[0])
>>> print('Calculated percent = %0.04f' % (percent_, ))
>>> assert percent_ > .6 and percent_ < .8, 'should be around 0.7472'
```

1.2.19 wbia.control.manual_garelate_funcs module

Autogenerated IBEISController functions

TemplateInfo: autogen_time = 13:34:34 2015/04/28 autogen_key = gar

ToRegenerate: python -m wbia.templates.template_generator –key gar –Tcfg with_web_api=True with_api_cache=False with_deleters=True no_extern_deleters=True –diff python -m wbia.templates.template_generator –key gar –Tcfg with_web_api=True with_api_cache=False with_deleters=True no_extern_deleters=True –write

wbia.control.manual_garelate_funcs.**add_gar**(*ibs*, *annotgroup_rowid_list*, *aid_list*)

Returns returns gar_rowid_list of added (or already existing gars)

TemplateInfo: Tadder_native tbl = gar

wbia.control.manual_garelate_funcs.**delete_gar**(*ibs*, *gar_rowid_list*, *config2_=None*)
gar.delete(gar_rowid_list)

delete gar rows

Parameters **gar_rowid_list** –

Returns num_deleted

Return type int

TemplateInfo: Tdeleter_native_tbl tbl = gar

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_garelate_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> gar_rowid_list = ibs._get_all_gar_rowids()[:2]
>>> num_deleted = ibs.delete_gar(gar_rowid_list)
>>> print('num_deleted = %r' % (num_deleted,))
```

wbia.control.manual_garelate_funcs.**get_gar_aid**(*ibs*, *gar_rowid_list*, *eager=True*, *nInput=None*)

aid_list <- gar.aid[gar_rowid_list]

gets data from the "native" column "aid" in the "gar" table

Parameters **gar_rowid_list** (*list*) –

Returns aid_list

Return type list

TemplateInfo: Tgetter_table_column col = aid tbl = gar

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_garelate_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> gar_rowid_list = ibs._get_all_gar_rowids()
>>> eager = True
>>> aid_list = ibs.get_gar_aid(gar_rowid_list, eager=eager)
>>> assert len(gar_rowid_list) == len(aid_list)
```

wbia.control.manual_garelate_funcs.**get_gar_annotgroup_rowid**(*ibs*, *gar_rowid_list*, *eager=True*, *nInput=None*)

annotgroup_rowid_list <- gar.annotgroup_rowid[gar_rowid_list]

gets data from the "native" column "annotgroup_rowid" in the "gar" table

Parameters **gar_rowid_list** (*list*) –

Returns annotgroup_rowid_list

Return type list

TemplateInfo: Tgetter_table_column col = annotgroup_rowid tbl = gar

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_garelate_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> gar_rowid_list = ibs._get_all_gar_rowids()
>>> eager = True
>>> annotgroup_rowid_list = ibs.get_gar_annotgroup_rowid(gar_rowid_list,␣
↪eager=eager)
>>> assert len(gar_rowid_list) == len(annotgroup_rowid_list)
```

wbia.control.manual_garelate_funcs.**get_gar_rowid_from_superkey**(*ibs*, *annotgroup_rowid_list*, *aid_list*, *eager=True*, *nInput=None*)

gar_rowid_list <- gar[annotgroup_rowid_list, aid_list]

Parameters **lists** (*superkey*) – annotgroup_rowid_list, aid_list

Returns gar_rowid_list

TemplateInfo: Tgetter_native_rowid_from_superkey tbl = gar

wbia.control.manual_garelate_funcs.**testdata_ibs**(*defaultdb='testdb1'*)

1.2.20 wbia.control.manual_gsgrelate_funcs module

CommandLine: # Autogenerate ImageSet Functions # key should be the table name # the write flag makes a file, but dont use that python -m wbia.templates.template_generator –key imageset_image_relationship –onlyfn

wbia.control.manual_gsgrelate_funcs.**add_image_relationship**(*ibs*, *gid_list*, *imgsetid_list*)

Adds a relationship between an image and and imageset

wbia.control.manual_gsgrelate_funcs.**delete_empty_imgsetids**(*ibs*)

Removes imagesets without images

Parameters **ibs** (IBEISController) – wbia controller object

CommandLine: python -m wbia.control.manual_gsgrelate_funcs –test-delete_empty_imgsetids

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_gsgrelate_funcs import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> # execute function
>>> result = ibs.delete_empty_imgsetids()
>>> # verify results
>>> print(result)
```

wbia.control.manual_gsgrelate_funcs.**delete_gsgr_image_relations**(*ibs*, *gid_list*)

Removes relationship between input images and all imagesets

wbia.control.manual_gsgrelate_funcs.**delete_gsgr_imageset_relations**(*ibs*, *imgsetid_list*)

Removes relationship between input imagesets and all images

wbia.control.manual_gsgrelate_funcs.**get_gsgr_rowid_from_superkey**(*ibs*, *gid_list*, *imgsetid_list*)

Returns eg-relate-ids from info constrained to be unique (imgsetid, gid)

Return type gsgrid_list (list)

wbia.control.manual_gsgrelate_funcs.**get_image_gsgrids**(*ibs*, *gid_list*)

Returns a list of imageset-image-relationship rowids for each imageid

Return type list_ (list)

wbia.control.manual_gsgrelate_funcs.**unrelate_images_and_imagesets**(*ibs*, *gid_list*, *imgsetid_list*)

Seems to unrelate specific image imageset pairs

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) –
- **imgsetid_list** (*list*) –

Returns gids_list

Return type list

CommandLine: python -m wbia.control.manual_gsgrelate_funcs –test-unrelate_images_and_imagesets python -c "import utool; print(utool.auto_docstr('wbia.control.manual_gsgrelate_funcs', 'delete_gsgr_image_relations'))"

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_gsgrelate_funcs import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> # Reset and compute imagesets
>>> ibs.delete_all_imagesets()
>>> ibs.compute_occurrences(config={'use_gps': False, 'seconds_thresh': 600})
>>> imgsetid_list = ibs.get_valid_imgsetids()
>>> gids_list = ibs.get_imageset_gids(imgsetid_list)
>>> assert len(imgsetid_list) == 2, 'bad len %r' % (len(imgsetid_list),)
>>> assert len(gids_list) == 2, 'bad len %r' % (len(gids_list),)
>>> assert len(gids_list[0]) == 7, 'bad len %r' % (len(gids_list[0]),)
>>> assert len(gids_list[1]) == 6, 'bad len %r' % (len(gids_list[1]),)
>>> # Add imageset 2 gids to imageset 1 so an image belongs to multiple imagesets
>>> imgset2_gids = gids_list[1][0:1]
>>> imgset1_imgsetids = imgsetid_list[0:1]
>>> ibs.add_image_relationship(imgset2_gids, imgset1_imgsetids)
>>> # Now delete the image from the imageset 2
>>> imgset2_imgsetids = imgsetid_list[1:2]
>>> # execute function
>>> ibs.unrelate_images_and_imagesets(imgset2_gids, imgset2_imgsetids)
```

(continues on next page)

(continued from previous page)

```
>>> # verify results
>>> ibs.print_egpairs_table()
>>> imgsetid_list_ = ibs.get_valid_imgsetids()
>>> gids_list_ = ibs.get_imageset_gids(imgsetid_list_)
>>> result = str(gids_list_)
>>> print(result)
>>> # imgset2_gids should now only be in imageset1
>>> assert imgset2_gids[0] in gids_list_[0], 'imgset2_gids should now only be in␣
↪imageset1'
>>> assert imgset2_gids[0] not in gids_list_[1], 'imgset2_gids should now only be␣
↪in imageset1'
```

1.2.21 wbia.control.manual_image_funcs module

Functions for images and encoutners that will be injected into an IBEISController instance.

CommandLine: # Autogenerate ImageSet Functions # key should be the table name # the write flag makes a file, but dont use that python -m wbia.templates.template_generator –key image –onlyfn python -m wbia.templates.template_generator –key image –fnfilt timedelta_posix –modfname manual_image_funcs # NOQA python -m wbia.templates.template_generator –key image –fnfilt location –modfname manual_image_funcs # NOQA python -m wbia.templates.template_generator –key image –fnfilt set_.*time –modfname manual_image_funcs # NOQA

image_timedelta_posix

wbia.control.manual_image_funcs.**add_images**(*ibs*, *gpath_list*, *params_list=None*, *as_annots=False*, *auto_localize=None*, *location_for_names=None*, *ensure_unique=False*, *ensure_loadable=True*, *ensure_exif=True*, ***kwargs*)

Adds a list of image paths to the database.

Initially we set the image_uri to exactely the given gpath. Later we change the uri, but keeping it the same here lets us process images asychronously.

Parameters

- **gpath_list** (*list*) – list of image paths to add
- **params_list** (*list*) – metadata list for corresponding images that can either be specified outright or can be parsed from the image data directly if None
- **as_annots** (*bool*) – if True, an annotation is automatically added for the entire image
- **auto_localize** (*bool*) – if None uses the default specified in ibs.cfg
- **ensure** (*bool*) – check to see if the images exist on a *NIX system. Defaults to True

Returns gids are image rowids

Return type gid_list (list of rowids)

RESTful: Method: POST URL: /api/image/

CommandLine: python -m wbia.control.manual_image_funcs –test-add_images

Doctest:

```
>>> # Test returns None on fail to add
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> gpath_list = ['doesnotexist.jpg']
>>> assert not ut.checkpath(gpath_list[0])
>>> gid_list = ibs.add_images(gpath_list)
>>> assert len(gid_list) == len(gpath_list)
>>> assert gid_list[0] is None
```

Doctest:

```
>>> # FIXME failing-test (22-Jul-2020) This test is failing and it's not␣
↪clear how to fix it
>>> # xdoctest: +SKIP
>>> # test double add
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> new_gpath_list = [ut.grab_test_imgpath('carl.jpg')]
>>> new_gids1 = ibs.add_images(new_gpath_list, auto_localize=False)
>>> new_gids2 = ibs.add_images(new_gpath_list, auto_localize=False)
>>> #new_gids2 = ibs.add_images(new_gpath_list, auto_localize=True)
>>> assert new_gids1 == new_gids2, 'should be the same'
>>> new_gpath_list2 = ibs.get_image_paths(new_gids1)
>>> assert new_gpath_list == new_gpath_list2, 'should not move when␣
↪autolocalize is False'
>>> # Clean things up
>>> ibs.delete_images(new_gids1)
```

wbia.control.manual_image_funcs.**compute_image_uuids**(*ibs*, *gpath_list*, ***kwargs*)

wbia.control.manual_image_funcs.**delete_image_thumbs**(*ibs*, *gid_list*, ***config2_*)

Removes image thumbnails from disk

RESTful: Method: DELETE URL: /api/image/thumb/

Ignore:

```
>>> # UNPORTED_DOCTEST
>>> gpath_list = ut.get_test_gpaths(ndata=None)[0:4]
>>> gid_list = ibs.add_images(gpath_list)
>>> bbox_list = [(0, 0, 100, 100)] * len(gid_list)
>>> name_list = ['a', 'b', 'a', 'd']
>>> aid_list = ibs.add_annots(gid_list, bbox_list=bbox_list,
>>>                           name_list=name_list)
>>> assert len(aid_list) != 0, "No annotations added"
>>> thumbpath_list = ibs.get_image_thumbpath(gid_list)
>>> gpath_list = ibs.get_image_paths(gid_list)
>>> ibs.delete_image_thumbs(gid_list)
>>> assert utool.is_list(thumbpath_list), "thumbpath_list is not a list"
>>> assert utool.is_list(gpath_list), "gpath_list is not a list"
>>> for path in thumbpath_list:
>>>     assert not utool.checkpath(path), "Thumbnail not deleted"
>>> for path in gpath_list:
>>>     utool.assertpath(path)
```

wbia.control.manual_image_funcs.**delete_images**(*ibs*, *gid_list*, *trash_images=True*)

deletes images from the database that belong to gids

RESTful: Method: DELETE URL: /api/image/

Ignore:

```
>>> # UNPORTED_DOCTEST
>>> gpath_list = ut.get_test_gpaths(ndata=None)[0:4]
>>> gid_list = ibs.add_images(gpath_list)
>>> bbox_list = [(0, 0, 100, 100)] * len(gid_list)
>>> name_list = ['a', 'b', 'a', 'd']
>>> aid_list = ibs.add_annots(gid_list, bbox_list=bbox_list, name_list=name_
↪list)
>>> gid = gid_list[0]
>>> assert gid is not None, "gid is None"
>>> aid_list = ibs.get_image_aids(gid)
>>> assert len(aid_list) == 1, "Length of aid_list=%r" % (len(aid_list),)
>>> aid = aid_list[0]
>>> assert aid is not None, "aid is None"
>>> cid = ibs.get_annot_chip_rowids(aid, ensure=False)
>>> fid = ibs.get_annot_feat_rowids(aid, ensure=False)
>>> assert cid is None, "cid=%r should be None" % (cid,)
>>> assert fid is None, "fid=%r should be None" % (fid,)
>>> cid = ibs.get_annot_chip_rowids(aid, ensure=True)
>>> fid = ibs.get_annot_feat_rowids(aid, ensure=True)
>>> assert cid is not None, "cid should be computed"
>>> assert fid is not None, "fid should be computed"
>>> gthumbpath = ibs.get_image_thumbpath(gid)
>>> athumbpath = ibs.get_annot_chip_thumbpath(aid)
>>> ibs.delete_images(gid)
>>> all_gids = ibs.get_valid_gids()
>>> all_aids = ibs.get_valid_aids()
>>> all_cids = ibs.get_valid_cids()
>>> all_fids = ibs.get_valid_fids()
>>> assert gid not in all_gids, "gid still exists"
>>> assert aid not in all_aids, "rid %r still exists" % aid
>>> assert fid not in all_fids, "fid %r still exists" % fid
>>> assert cid not in all_cids, "cid %r still exists" % cid
>>> assert not utool.checkpath(gthumbpath), "Thumbnail still exists"
>>> assert not utool.checkpath(athumbpath), "ANNOTATION Thumbnail still␣
↪exists"
```

wbia.control.manual_image_funcs.**get_image_aids**(*ibs*, *gid_list*, *is_staged=False*)

Returns a list of aids for each image by gid

Return type list_ (list)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) –

Returns aids_list

Return type list

CommandLine: python -m wbia.control.manual_image_funcs –test-get_image_aids

RESTful: Method: GET URL: /api/image/annot/rowid/

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_annot_gids(ibs.get_valid_aids())
>>> gid_list = gid_list + gid_list[::5]
>>> # execute function
>>> aids_list = get_image_aids(ibs, gid_list)
>>> # verify results
>>> result = str(aids_list)
>>> print(result)
[[1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [1], [6],␣
↪[11]]
```

Ignore: logger.info('len(gid_list) = %r' % (len(gid_list),)) logger.info('len(input_list) = %r' % (len(input_list),)) logger.info('len(pair_list) = %r' % (len(pair_list),)) logger.info('len(aidscol) = %r' % (len(aidscol),)) logger.info('len(gidscol) = %r' % (len(gidscol),)) logger.info('len(unique_gids) = %r' % (len(unique_gids),))

wbia.control.manual_image_funcs.**get_image_aids_of_species**(*ibs*, *gid_list*, *species=None*)

Returns a list of aids for each image by gid filtered by species

Return type list_ (list)

RESTful: Method: GET URL: /api/image/annot/rowid/species/

wbia.control.manual_image_funcs.**get_image_annot_uuids**(*ibs*, *gid_list*)

wbia.control.manual_image_funcs.**get_image_annot_uuids_of_species**(*ibs*, *gid_list*, ***kwargs*)

wbia.control.manual_image_funcs.**get_image_cameratrap**(*ibs*, *gid_list*)

wbia.control.manual_image_funcs.**get_image_contributor_rowid**(*ibs*, *gid_list*, *eager=True*, *nInput=None*)

contributor_rowid_list <- image.contributor_rowid[gid_list]

gets data from the "native" column "contributor_rowid" in the "image" table

Parameters **gid_list** (*list*) –

Returns contributor_rowid_list - list of image contributor rowids by gid

Return type list

TemplateInfo: Tgetter_table_column col = contributor_rowid tbl = image

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> gid_list = ibs._get_all_image_rowids()
>>> eager = True
>>> contributor_rowid_list = ibs.get_image_contributor_rowid(gid_list,␣
↪eager=eager)
>>> assert len(gid_list) == len(contributor_rowid_list)
```

wbia.control.manual_image_funcs.**get_image_contributor_tag**(*ibs*, *gid_list*, *eager=True*, *nInput=None*)

contributor_tag_list <- image.contributor_tag[gid_list]

Parameters **gid_list** (*list*) –

Returns contributor_tag_list

Return type list

TemplateInfo: Tgetter_extern tbl = image externtbl = contributor externcol = contributor_tag

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> gid_list = ibs._get_all_image_rowids()
>>> eager = True
>>> contributor_tag_list = ibs.get_image_contributor_tag(gid_list, eager=eager)
>>> assert len(gid_list) == len(contributor_tag_list)
```

wbia.control.manual_image_funcs.**get_image_datetime**(*ibs*, *gid_list*, ***kwargs*)

wbia.control.manual_image_funcs.**get_image_datetime_str**(*ibs*, *gid_list*, ***kwargs*)

wbia.control.manual_image_funcs.**get_image_detect_confidence**(*ibs*, *gid_list*)

Returns image detection confidence as the max of ANNOTATION confidences

Return type list_ (list)

RESTful: Method: GET URL: /api/image/detect/confidence/

wbia.control.manual_image_funcs.**get_image_detectpaths**(*ibs*, *gid_list*)

Returns a list of image paths resized to a constant area for detection

Return type list_ (list)

wbia.control.manual_image_funcs.**get_image_enabled**(*ibs*, *gid_list*)

Returns “Image Enabled” flag, true if the image is enabled

Return type list_ (list)

wbia.control.manual_image_funcs.**get_image_exif_original**(*ibs*, *gid_list*)

wbia.control.manual_image_funcs.**get_image_exts**(*ibs*, *gid_list*)

Returns a list of image uuids by gid

Return type list_ (list)

wbia.control.manual_image_funcs.**get_image_gid**(*ibs*, *gid_list*, *eager=True*, *nInput=None*)

self verifier

CommandLine: python -m wbia.control.manual_image_funcs –exec-get_image_gid

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.IBEISControl import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids() + [None, -1, 10434320432]
>>> gid_list_ = ibs.get_image_gid(gid_list)
>>> assert [r is None for r in gid_list_[-3:]]
>>> assert [r is not None for r in gid_list_[0:-3]]
>>> print('gid_list_ = %r' % (gid_list_,))
```

wbia.control.manual_image_funcs.**get_image_gids_from_uuid**(*ibs*, *uuid_list*)

Returns a list of original image names

Return type list_ (list)

RESTful: Method: GET URL: /api/image/rowid/uuid/

wbia.control.manual_image_funcs.**get_image_gids_with_aids**(*ibs*, *gid_list=None*)

wbia.control.manual_image_funcs.**get_image_gnames**(*ibs*, *gid_list*)

Parameters **gid_list** (*list*) –

Returns gname_list - a list of original image names

Return type list

CommandLine: python -m wbia.control.manual_image_funcs –test-get_image_gnames

RESTful: Method: GET URL: /api/image/file/name/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids()
>>> # execute function
>>> gname_list = get_image_gnames(ibs, gid_list)
>>> # verify results
>>> result = ut.repr2(gname_list, nl=1)
>>> print(result)
[
    'easy1.JPG',
    'easy2.JPG',
    'easy3.JPG',
```

(continues on next page)

(continued from previous page)

```
    'hard1.JPG',
    'hard2.JPG',
    'hard3.JPG',
    'jeff.png',
    'lena.jpg',
    'occl1.JPG',
    'occl2.JPG',
    'polar1.jpg',
    'polar2.jpg',
    'zebra.jpg',
]
```

wbia.control.manual_image_funcs.**get_image_gps**(*ibs*, *gid_list*)

Returns -1 if no timedata exists for a given gid

Return type gps_list (list)

RESTful: Method: GET URL: /api/image/gps/

wbia.control.manual_image_funcs.**get_image_gps2**(*ibs*, *gid_list*)

Like get_image_gps, but fixes the SQL problem where -1 indicates a nan value.

Returns -1 if no timedata exists for a given gid

Return type gps_list (list)

RESTful: Method: GET URL: /api/image/gps/

wbia.control.manual_image_funcs.**get_image_hash**(*ibs*, *gid_list=None*, *algo='md5'*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) – a list of image absolute paths to img_dir

Returns hash_list

Return type list

CommandLine: python -m wbia.control.manual_image_funcs –test-get_image_hash

RESTful: Method: GET URL: /api/image/file/hash/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids()[:1]
>>> image_path = ibs.get_image_paths(gid_list)
>>> print('Hashing: %r' % (image_path, ))
>>> hash_list = ibs.get_image_hash(gid_list, algo='md5')
>>> assert hash_list[0] in ['56498e54b5ebbcbbcff60c91a135e8a3',
↪'ab31dc5e1355247a0ea5ec940802a468'], 'Found %r' % (hash_list, )
```

(continues on next page)

(continued from previous page)

```
>>> hash_list = ibs.get_image_hash(gid_list, algo='sha1')
>>> assert hash_list[0] in ['277e8dac1e5929c097f3fcbca2c77d92e1401d5f',
↪'66ec193a1619b3b6216d1784b4833b6194b13384'], 'Found %r' % (hash_list, )
>>> hash_list = ibs.get_image_hash(gid_list, algo='sha256')
>>> assert hash_list[0] in [
↪'ca03a0d7427c3d2f02e62e157e8d8ea5b7284be67ca67fc391a5747368d3ab0e',
↪'fd09d22ec18c32d9db2cd026a9511ab228aadf0e5f7271760413448ddd16d483'], 'Found %r'
↪% (hash_list, )
>>> hash_list = ibs.get_image_hash(gid_list, algo='sha512')
>>> assert hash_list[0] in [
↪'7b43dbc709a8cf903170b414f48a0bb7b569b703d9393c20a2cff95c42fd252ed2098bc56cba8eed393bcdf3388e5
↪',
↪'81d1d8ee4c8640b9aad26e4cc03536ed30a43b69e166748ec940a8f00e4776be93f4ac6367a06d92b772a9a60dc10
↪'], 'Found %r' % (hash_list, )
```

wbia.control.manual_image_funcs.**get_image_heights**(*ibs*, *gid_list*)

Returns a list of (width, height) tuples

Return type list_ (list)

RESTful: Method: GET URL: /api/image/height/

wbia.control.manual_image_funcs.**get_image_imagesettext**(*ibs*, *gid_list*)

Returns a list of imagesettexts for each image by gid

Return type list_ (list)

RESTful: Method: GET URL: /api/image/imageset/text/

wbia.control.manual_image_funcs.**get_image_imgdata**(*ibs*, *gid_list*, *ignore_orient=False*, ***kwargs*)

alias for get_images with standardized name

wbia.control.manual_image_funcs.**get_image_imgset_uuids**(*ibs*, *gid_list*)

wbia.control.manual_image_funcs.**get_image_imgsetids**(*ibs*, *gid_list*)

Returns a list of imageset ids for each image by gid

Return type list_ (list)

RESTful: Method: GET URL: /api/image/imageset/rowid/

wbia.control.manual_image_funcs.**get_image_lat**(*ibs*, *gid_list*)

RESTful: Method: GET URL: /api/image/lat/

wbia.control.manual_image_funcs.**get_image_location_codes**(*ibs*, *gid_list*, *eager=True*)

image_location_code_list <- image.image_location_code[gid_list]

gets data from the "native" column "image_location_code" in the "image" table

Parameters **gid_list** (*list*) –

Returns image_location_code_list

Return type list

TemplateInfo: Tgetter_table_column col = image_location_code tbl = image

RESTful: Method: GET URL: /api/image/location/code/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> gid_list = ibs._get_all_image_rowids()
>>> eager = True
>>> image_location_code_list = ibs.get_image_location_codes(gid_list, eager=eager)
>>> assert len(gid_list) == len(image_location_code_list)
```

wbia.control.manual_image_funcs.**get_image_lon**(*ibs*, *gid_list*)

RESTful: Method: GET URL: /api/image/lon/

wbia.control.manual_image_funcs.**get_image_metadata**(*ibs*, *gid_list*, *return_raw=False*)

Returns image metadata dictionary

Return type list_ (list)

RESTful: Method: GET URL: /api/image/metadata/

wbia.control.manual_image_funcs.**get_image_missing_uuid**(*ibs*, *uuid_list*)

Returns a list of missing image uuids

Return type list_ (list)

wbia.control.manual_image_funcs.**get_image_name_uuids**(*ibs*, *gid_list*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) –

Returns name_uuids_list - the name uuids associated with an image id

Return type list

CommandLine: python -m wbia.control.manual_image_funcs –test-get_image_nids

RESTful: Method: GET URL: /api/image/name/uuid/

wbia.control.manual_image_funcs.**get_image_nids**(*ibs*, *gid_list*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) –

Returns nids_list - the name ids associated with an image id

Return type list

CommandLine: python -m wbia.control.manual_image_funcs –test-get_image_nids

RESTful: Method: GET URL: /api/image/name/rowid/

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids()
>>> # execute function
>>> nids_list = ibs.get_image_nids(gid_list)
>>> # verify results
>>> result = str(nids_list)
>>> print(result)
```

wbia.control.manual_image_funcs.**get_image_notes**(*ibs*, *gid_list*)

Returns image notes

Return type list_ (list)

RESTful: Method: GET URL: /api/image/note/

wbia.control.manual_image_funcs.**get_image_num_annotations**(*ibs*, *gid_list*)

Returns the number of chips in each image

Return type list_ (list)

RESTful: Method: GET URL: /api/image/num/annot/

wbia.control.manual_image_funcs.**get_image_orientation**(*ibs*, *gid_list*)

RESTful: Method: GET URL: /api/image/orientation/

wbia.control.manual_image_funcs.**get_image_orientation_str**(*ibs*, *gid_list*)

RESTful: Method: GET URL: /api/image/orientation/str/

wbia.control.manual_image_funcs.**get_image_party_rowids**(*ibs*, *gid_list*, *eager=True*, *nInput=None*)

party_rowid_list <- image.party_rowid[gid_list]

gets data from the "native" column "party_rowid" in the "image" table

Parameters **gid_list** (*list*) –

Returns party_rowid_list

Return type list

TemplateInfo: Tgetter_table_column col = party_rowid tbl = image

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> gid_list = ibs._get_all_image_rowids()
>>> eager = True
```

(continues on next page)

(continued from previous page)

```
>>> party_rowid_list = ibs.get_image_party_rowids(gid_list, eager=eager)
>>> assert len(gid_list) == len(party_rowid_list)
```

wbia.control.manual_image_funcs.**get_image_party_tag**(*ibs*, *gid_list*, *eager=True*, *nInput=None*)

party_tag_list <- image.party_tag[gid_list]

Parameters **gid_list** (*list*) –

Returns party_tag_list

Return type list

TemplateInfo: Tgetter_extern tbl = image externtbl = party externcol = party_tag

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> gid_list = ibs._get_all_image_rowids()
>>> eager = True
>>> party_tag_list = ibs.get_image_party_tag(gid_list, eager=eager)
>>> assert len(gid_list) == len(party_tag_list)
```

wbia.control.manual_image_funcs.**get_image_paths**(*ibs*, *gid_list*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) – a list of image absolute paths to img_dir

Returns gpath_list

Return type list

CommandLine: python -m wbia.control.manual_image_funcs –test-get_image_paths

RESTful: Method: GET URL: /api/image/file/path/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> #gid_list = ibs.get_valid_gids()
>>> #gpath_list = get_image_paths(ibs, gid_list)
>>> new_gpath = ut.unixpath(ut.grab_test_imgpath('carl.jpg'))
>>> gid_list = ibs.add_images([new_gpath], auto_localize=False)
>>> new_gpath_list = get_image_paths(ibs, gid_list)
>>> ut.assert_eq(new_gpath, new_gpath_list[0])
>>> result = str(new_gpath_list)
>>> ibs.delete_images(gid_list)
>>> print(result)
```

wbia.control.manual_image_funcs.**get_image_reviewed**(*ibs*, *gid_list*)

Returns "All Instances Found" flag, true if all objects of interest

Return type list_ (list)

(animals) have an ANNOTATION in the image

RESTful: Method: GET URL: /api/image/reviewed/

wbia.control.manual_image_funcs.**get_image_sizes**(*ibs*, *gid_list*)

Returns a list of (width, height) tuples

Return type list_ (list)

RESTful: Method: GET URL: /api/image/size/

wbia.control.manual_image_funcs.**get_image_species_rowids**(*ibs*, *gid_list*)

Returns the name ids associated with an image id

Return type list_ (list)

RESTful: Method: GET URL: /api/image/species/rowid/

wbia.control.manual_image_funcs.**get_image_species_uuids**(*ibs*, *gid_list*)

Returns the name ids associated with an image id

Return type list_ (list)

RESTful: Method: GET URL: /api/image/species/uuid/

wbia.control.manual_image_funcs.**get_image_thumbnail**(*ibs*, *gid_list*, ***config*)

Returns the thumbnail path of each gid

Return type list_ (list)

wbia.control.manual_image_funcs.**get_image_thumbpath**(*ibs*, *gid_list*, *ensure_paths=False*, ***config*)

Returns the thumbnail path of each gid

Return type list_ (list)

wbia.control.manual_image_funcs.**get_image_thumbtup**(*ibs*, *gid_list*, ***kwargs*)

Returns thumbtup_list - [(thumb_path, img_path, imgsize, bboxes, thetas)]

Return type list

wbia.control.manual_image_funcs.**get_image_timedelta_posix**(*ibs*, *gid_list*, *eager=True*)

image_timedelta_posix_list <- image.image_timedelta_posix[gid_list]

TODO: INTEGRATE THIS FUNCTION. CURRENTLY OFFSETS ARE ENCODIED DIRECTLY IN UNIXTIME

gets data from the "native" column "image_timedelta_posix" in the "image" table

Parameters **gid_list** (*list*) –

Returns image_timedelta_posix_list

Return type list

TemplateInfo: Tgetter_table_column col = image_timedelta_posix tbl = image

RESTful: Method: GET URL: /api/image/timedelta/posix/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> gid_list = ibs._get_all_image_rowids()
>>> eager = True
>>> image_timedelta_posix_list = ibs.get_image_timedelta_posix(gid_list,␣
↪eager=eager)
>>> assert len(gid_list) == len(image_timedelta_posix_list)
```

wbia.control.manual_image_funcs.**get_image_unixtime**(*ibs*, *gid_list*, *timedelta_correction=True*)

Returns a list of times that the images were taken by gid.

Return type list_ (list)

Returns -1 if no timedata exists for a given gid

Return type list_ (list)

RESTful: Method: GET URL: /api/image/unixtime/

wbia.control.manual_image_funcs.**get_image_unixtime2**(*ibs*, *gid_list*, ***kwargs*)
alias for get_image_unixtime_asfloat

wbia.control.manual_image_funcs.**get_image_unixtime_asfloat**(*ibs*, *gid_list*, ***kwargs*)

Returns a list of times that the images were taken by gid.

Return type list_ (list)

Returns np.nan if no timedata exists for a given gid

Return type list_ (list)

wbia.control.manual_image_funcs.**get_image_uris**(*ibs*, *gid_list*)

Returns a list of image uris relative to the image dir by gid

Return type list_ (list)

RESTful: Method: GET URL: /api/image/uri/

wbia.control.manual_image_funcs.**get_image_uris_original**(*ibs*, *gid_list*)

Returns a list of (original) image uris relative to the image dir by gid

Return type list_ (list)

RESTful: Method: GET URL: /api/image/uri/original/

wbia.control.manual_image_funcs.**get_image_uuids**(*ibs*, *gid_list*)

Returns a list of image uuids by gid

Return type list_ (list)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) –

Returns image_uuid_list

Return type list

CommandLine: python -m wbia.control.manual_image_funcs –test-get_image_uuids

RESTful: Method: GET URL: /api/image/uuid/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids()
>>> # execute function
>>> image_uuid_list = ibs.get_image_uuids(gid_list)
>>> # verify results
>>> result = ut.repr2(image_uuid_list, nl=1)
>>> print(result)
[
    UUID('66ec193a-1619-b3b6-216d-1784b4833b61'),
    UUID('d8903434-942f-e0f5-d6c2-0dcbe3137bf7'),
    UUID('b73b72f4-4acb-c445-e72c-05ce02719d3d'),
    UUID('0cd05978-3d83-b2ee-2ac9-798dd571c3b3'),
    UUID('0a9bc03d-a75e-8d14-0153-e2949502aba7'),
    UUID('2deeff06-5546-c752-15dc-2bd0fdb1198a'),
    UUID('68ca272d-26f7-1dbb-76e9-08d192c1a4a7'),
    UUID('42fdad98-369a-2cbc-67b1-983d6d6a3a60'),
    UUID('c459d381-fd74-1d99-6215-e42e3f432ea9'),
    UUID('33fd9813-3a2b-774b-3fcc-4360d1ae151b'),
    UUID('97e8ea74-873f-2092-b372-f928a7be30fa'),
    UUID('588bc218-83a5-d400-21aa-d499832632b0'),
    UUID('163a890c-36f2-981e-3529-c552b6d668a3'),
]
```

wbia.control.manual_image_funcs.**get_image_widths**(*ibs*, *gid_list*)

Returns a list of (width, height) tuples

Return type list_ (list)

RESTful: Method: GET URL: /api/image/width/

wbia.control.manual_image_funcs.**get_images**(*ibs*, *gid_list*, *ignore_orient=False*, ***kwargs*)

Returns a list of images in numpy matrix form by gid

Return type list_ (list)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) –

Returns image_list

Return type list

CommandLine: python -m wbia.control.manual_image_funcs –test-get_images

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids()[0:1]
>>> # execute function
>>> image_list = get_images(ibs, gid_list)
>>> # verify results
>>> result = str(image_list[0].shape)
>>> print(result)
(715, 1047, 3)
```

wbia.control.manual_image_funcs.**get_num_images**(*ibs*, ***kwargs*)
Number of valid images

wbia.control.manual_image_funcs.**get_valid_gids**(*ibs*, *imgsetid=None*, *imgsetid_list=()*, *require_unixtime=False*, *require_gps=None*, *reviewed=None*, ***kwargs*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **imgsetid** (*None*) –
- **require_unixtime** (*bool*) –
- **reviewed** (*None*) –

Returns gid_list

Return type list

CommandLine: python -m wbia.control.manual_image_funcs –test-get_valid_gids

RESTful: Method: GET URL: /api/image/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> imgsetid = None
```

(continues on next page)

(continued from previous page)

```
>>> require_unixtime = False
>>> reviewed = None
>>> # execute function
>>> gid_list = get_valid_gids(ibs, imgsetid, require_unixtime, reviewed)
>>> # verify results
>>> result = str(gid_list)
>>> print(result)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
```

wbia.control.manual_image_funcs.**get_valid_image_rowids**(*ibs*, *imgsetid=None*, *require_unixtime=False*, *reviewed=None*)

alias

wbia.control.manual_image_funcs.**get_valid_image_uuids**(*ibs*)

Returns a list of image uuids for all valid gids

Return type list_ (list)

Parameters **ibs** (IBEISController) – wbia controller object

Returns image_uuid_list

Return type list

CommandLine: python -m wbia.control.manual_image_funcs –test-get_image_uuids

wbia.control.manual_image_funcs.**image_base64_api**(*rowid=None*, *thumbnail=False*, *fresh=False*, ***kwargs*)

Returns the base64 encoded image of image <rowid>

RESTful: Method: GET URL: /api/image/<rowid>/

wbia.control.manual_image_funcs.**localize_images**(*ibs*, *gid_list_=None*, *cache_uri_dict=None*, *cleanup=True*)

Moves the images into the wbia image cache. Images are renamed to img_uuid.ext

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) –

CommandLine: python -m wbia.control.manual_image_funcs –test-localize_images

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> import wbia
>>> import os
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> gpath_list  = [ut.unixpath(ut.grab_test_imgpath('carl.jpg'))]
>>> gid_list_   = ibs.add_images(gpath_list, auto_localize=False)
>>> gpath_list2 = ibs.get_image_paths(gid_list_)
>>> ut.assert_eq(gpath_list, gpath_list2, 'should not move when autolocalize is␣
↪False')
```

(continues on next page)

(continued from previous page)

```
>>> # execute function
>>> result = localize_images(ibs, gid_list_)
>>> gpath_list3 = ibs.get_image_paths(gid_list_)
>>> assert gpath_list3 != gpath_list2, 'should now be different gpath_list3=%r' %␣
↪(gpath_list3,)
>>> gpath3 = gpath_list3[0]
>>> rel_gpath3 = ut.relpath_unix(gpath3, ibs.get_workdir())
>>> result = rel_gpath3
>>> print(result)
>>> # Clean things up
>>> paths = ibs.get_image_paths(gid_list_)
>>> ibs.delete_images(gid_list_)
>>> for path in paths:
>>>     assert not os.path.exists(path)
```

Ignore: ibs.vd()

wbia.control.manual_image_funcs.**set_image_cameratrap**(*ibs*, *gid_list*, *cameratrap_list*)
Sets the image all instances found bit

wbia.control.manual_image_funcs.**set_image_contributor_rowid**(*ibs*, *gid_list*, *contributor_rowid_list*, ***kwargs*)
Sets the image contributor rowid

wbia.control.manual_image_funcs.**set_image_enabled**(*ibs*, *gid_list*, *enabled_list*)
Sets the image all instances found bit

wbia.control.manual_image_funcs.**set_image_gps**(*ibs*, *gid_list*, *gps_list=None*, *lat_list=None*, *lon_list=None*)

see get_image_gps for how the gps_list should look. lat and lon should be given in degrees

RESTful: Method: PUT URL: /api/image/gps/

wbia.control.manual_image_funcs.**set_image_gps_str**(*ibs*, *gid_list*, *gps_str_list*)

see get_image_gps for how the gps_list should look. lat and lon should be given in degrees

RESTful: Method: PUT URL: /api/image/gps/

wbia.control.manual_image_funcs.**set_image_imagesettext**(*ibs*, *gid_list*, *imagesettext_list*)
Sets the encoutertext of each image

RESTful: Method: PUT URL: /api/image/imageset/text/

wbia.control.manual_image_funcs.**set_image_imgsetids**(*ibs*, *gid_list*, *imgsetid_list*)
Sets the encoutertext of each image

RESTful: Method: PUT URL: /api/image/imageset/rowid/

wbia.control.manual_image_funcs.**set_image_location_codes**(*ibs*, *gid_list*, *image_location_code_list*, *duplicate_behavior='error'*)
image_location_code_list -> image.image_location_code[gid_list]

Parameters

- **gid_list** –

- **image_location_code_list** –

TemplateInfo: Tsetter_native_column tbl = image col = image_location_code

RESTful: Method: PUT URL: /api/image/location/code/

wbia.control.manual_image_funcs.**set_image_metadata**(*ibs*, *gid_list*, *metadata_dict_list*)

Sets the image's metadata using a metadata dictionary

RESTful: Method: PUT URL: /api/image/metadata/

CommandLine: python -m wbia.control.manual_image_funcs –test-set_image_metadata

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> import wbia
>>> import random
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids()[0:1]
>>> metadata_dict_list = [
>>>     {'test': random.uniform(0.0, 1.0)},
>>> ]
>>> print(ut.repr2(metadata_dict_list))
>>> ibs.set_image_metadata(gid_list, metadata_dict_list)
>>> # verify results
>>> metadata_dict_list_ = ibs.get_image_metadata(gid_list)
>>> print(ut.repr2(metadata_dict_list_))
>>> assert metadata_dict_list == metadata_dict_list_
>>> metadata_str_list = [ut.to_json(metadata_dict) for metadata_dict in metadata_
↪dict_list]
>>> print(ut.repr2(metadata_str_list))
>>> metadata_str_list_ = ibs.get_image_metadata(gid_list, return_raw=True)
>>> print(ut.repr2(metadata_str_list_))
>>> assert metadata_str_list == metadata_str_list_
```

wbia.control.manual_image_funcs.**set_image_notes**(*ibs*, *gid_list*, *notes_list*)

Sets the image all instances found bit

RESTful: Method: PUT URL: /api/image/note/

wbia.control.manual_image_funcs.**set_image_orientation**(*ibs*, *gid_list*, *orientation_list*)

RESTful: Method: PUT URL: /api/image/orientation/

wbia.control.manual_image_funcs.**set_image_party_rowids**(*ibs*, *gid_list*, *party_rowid_list*, *duplicate_behavior='error'*)

party_rowid_list -> image.party_rowid[gid_list]

Parameters

- **gid_list** –
- **party_rowid_list** –

TemplateInfo: Tsetter_native_column tbl = image col = party_rowid

wbia.control.manual_image_funcs.**set_image_reviewed**(*ibs*, *gid_list*, *reviewed_list*)
Sets the image all instances found bit

RESTful: Method: PUT URL: /api/image/reviewed/

wbia.control.manual_image_funcs.**set_image_time_posix**(*ibs*, *gid_list*, *image_time_posix_list*, *duplicate_behavior='error'*)
image_time_posix_list -> image.image_time_posix[gid_list]

SeeAlso: set_image_unixtime

Parameters

- **gid_list** –
- **image_time_posix_list** –

TemplateInfo: Tsetter_native_column tbl = image col = image_time_posix

RESTful: Method: PUT URL: /api/image/time/posix/

wbia.control.manual_image_funcs.**set_image_timedelta_posix**(*ibs*, *gid_list*, *image_timedelta_posix_list*, *duplicate_behavior='error'*)
image_timedelta_posix_list -> image.image_timedelta_posix[gid_list]

Parameters

- **gid_list** –
- **image_timedelta_posix_list** –

TemplateInfo: Tsetter_native_column tbl = image col = image_timedelta_posix

RESTful: Method: PUT URL: /api/image/timedelta/posix/

wbia.control.manual_image_funcs.**set_image_unixtime**(*ibs*, *gid_list*, *unixtime_list*, *duplicate_behavior='error'*)
Sets the image unixtime (does not modify exif yet) alias for set_image_time_posix

RESTful: Method: PUT URL: /api/image/unixtime/

CommandLine: python -m wbia.control.manual_image_funcs –test-set_image_unixtime

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> import wbia
>>> import random
>>> import time
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids()[0:5]
>>> unixtime_list = [
>>>     random.randint(1, int(time.time()))
>>>     for _ in gid_list
```

(continues on next page)

(continued from previous page)

```
>>> ]
>>> print(ut.repr2(unixtime_list))
>>> ibs.set_image_unixtime(gid_list, unixtime_list)
>>> # verify results
>>> unixtime_list_ = ibs.get_image_unixtime(gid_list)
>>> print(ut.repr2(unixtime_list_))
>>> assert unixtime_list == unixtime_list_
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> import wbia
>>> import random
>>> import time
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids()[0:5]
>>> gid_list = gid_list + gid_list
>>> unixtime_list = [
>>>     random.randint(1, int(time.time()))
>>>     for _ in gid_list
>>> ]
>>> try:
>>>     print(ut.repr2(unixtime_list))
>>>     ibs.set_image_unixtime(gid_list, unixtime_list)
>>> except AssertionError:
>>>     pass
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_image_funcs import *  # NOQA
>>> import wbia
>>> import random
>>> import time
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids()[0:5]
>>> unixtime_list = [
>>>     random.randint(1, int(time.time()))
>>>     for _ in gid_list
>>> ]
>>> gid_list = gid_list + gid_list
>>> unixtime_list = unixtime_list + unixtime_list
>>> print(ut.repr2(unixtime_list))
>>> ibs.set_image_unixtime(gid_list, unixtime_list)
>>> # verify results
>>> unixtime_list_ = ibs.get_image_unixtime(gid_list)
>>> print(ut.repr2(unixtime_list_))
>>> assert unixtime_list == unixtime_list_
```

wbia.control.manual_image_funcs.**set_image_uris**(*ibs*, *gid_list*, *new_gpath_list*)

Sets the image URIs to a new local path. This is used when localizing or unlocalizing images. An absolute path

can either be on this machine or on the cloud A relative path is relative to the wbia image cache on this machine.

RESTful: Method: PUT URL: /api/image/uri/

wbia.control.manual_image_funcs.**set_image_uris_original**(*ibs*, *gid_list*, *new_gpath_list*, *overwrite=False*)

Sets the (original) image URIs to a new local path.

Parameters **overwrite** (*bool*) – If overwrite, replace the information in the database. This ensures that original uris cannot be accidentally overwritten. Defaults to False.

RESTful: Method: PUT URL: /api/image/uri/original/

wbia.control.manual_image_funcs.**testdata_ibs**()

wbia.control.manual_image_funcs.**update_image_rotate_180**(*ibs*, *gid_list*)

wbia.control.manual_image_funcs.**update_image_rotate_90**(*ibs*, *gid_list*, *direction*)

wbia.control.manual_image_funcs.**update_image_rotate_left_90**(*ibs*, *gid_list*)

wbia.control.manual_image_funcs.**update_image_rotate_right_90**(*ibs*, *gid_list*)

1.2.22 wbia.control.manual_imageset_funcs module

wbia.control.manual_imageset_funcs.**add_imagesets**(*ibs*, *imagesettext_list*, *imageset_uuid_list=None*, *notes_list=None*, *occurence_flag_list=None*)

Adds a list of imagesets.

Parameters

- **imagesettext_list** (*list*) –
- **imageset_uuid_list** (*list*) –
- **notes_list** (*list*) –

Returns added imageset rowids

Return type imgsetid_list (list)

RESTful: Method: POST URL: /api/imageset/

wbia.control.manual_imageset_funcs.**delete_imagesets**(*ibs*, *imgsetid_list*)

Removes imagesets and thier relationships (images are not effected)

RESTful: Method: DELETE URL: /api/imageset/

wbia.control.manual_imageset_funcs.**get_imageset_aids**(*ibs*, *imgsetid_list*)

Returns a list of list of aids in each imageset

Return type aids_list (list)

RESTful: Method: GET URL: /api/imageset/annot/rowid/

Parameters

- **ibs** (IBEISController) – wbia controller object

- **imgsetid_list** (*list*) –

Returns aids_list

Return type list

CommandLine: python -m wbia.control.manual_imageset_funcs –test-get_imageset_aids

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> ibs.delete_imagesets(ibs.get_valid_imgsetids())
>>> ibs.compute_occurrences(config={'use_gps': False, 'seconds_thresh': 600})
>>> imgsetid_list = ibs.get_valid_imgsetids()
>>> aids_list = get_imageset_aids(ibs, imgsetid_list)
>>> result = ('aids_list = %s' % (str(aids_list),))
>>> print(result)
```

wbia.control.manual_imageset_funcs.**get_imageset_custom_filtered_aids**(*ibs*, *imgsetid_list*)

hacks to filter aids to only certain views and qualities

wbia.control.manual_imageset_funcs.**get_imageset_duration**(*ibs*, *imageset_rowid_list*)

gets the imageset's duration

Parameters **imageset_rowid_list** (*list*) –

Returns imageset_duration

Return type list

RESTful: Method: GET URL: /api/imageset/duration/

wbia.control.manual_imageset_funcs.**get_imageset_end_time_posix**(*ibs*, *imageset_rowid_list*)

imageset_end_time_posix_list <- imageset.imageset_end_time_posix[imageset_rowid_list]

gets data from the "native" column "imageset_end_time_posix" in the "imageset" table

Parameters **imageset_rowid_list** (*list*) –

Returns imageset_end_time_posix_list

Return type list

TemplateInfo: Tgetter_table_column col = imageset_end_time_posix tbl = imageset

RESTful: Method: GET URL: /api/imageset/time/posix/end/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
```

(continues on next page)

(continued from previous page)

```
>>> imageset_rowid_list = ibs._get_all_imageset_rowids()
>>> imageset_end_time_posix_list = ibs.get_imageset_end_time_posix(imageset_rowid_
↪list)
>>> assert len(imageset_rowid_list) == len(imageset_end_time_posix_list)
```

wbia.control.manual_imageset_funcs.**get_imageset_fraction_annotmatch_reviewed**(*ibs*, *imgsetid_list*)

wbia.control.manual_imageset_funcs.**get_imageset_fraction_imgs_reviewed**(*ibs*, *imgsetid_list*)

wbia.control.manual_imageset_funcs.**get_imageset_fraction_names_with_exemplar**(*ibs*, *imgsetid_list*)

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb2')
>>> imgsetid_list = ibs._get_all_imageset_rowids()
>>> fraction_exemplared_names_list = ibs.get_imageset_fraction_names_with_
↪exemplar(imgsetid_list)
```

wbia.control.manual_imageset_funcs.**get_imageset_gids**(*ibs*, *imgsetid_list*)

Returns a list of list of gids in each imageset

Return type gids_list (list)

RESTful: Method: GET URL: /api/imageset/image/rowid/

wbia.control.manual_imageset_funcs.**get_imageset_gps_lats**(*ibs*, *imageset_rowid_list*)

imageset_gps_lat_list <- imageset.imageset_gps_lat[imageset_rowid_list]

gets data from the "native" column "imageset_gps_lat" in the "imageset" table

Parameters **imageset_rowid_list** (*list*) –

Returns imageset_gps_lat_list

Return type list

TemplateInfo: Tgetter_table_column col = imageset_gps_lat tbl = imageset

RESTful: Method: GET URL: /api/imageset/gps/lat/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> imageset_rowid_list = ibs._get_all_imageset_rowids()
>>> imageset_gps_lat_list = ibs.get_imageset_gps_lats(imageset_rowid_list)
>>> assert len(imageset_rowid_list) == len(imageset_gps_lat_list)
```

wbia.control.manual_imageset_funcs.**get_imageset_gps_lons**(*ibs*, *imageset_rowid_list*)

imageset_gps_lon_list <- imageset.imageset_gps_lon[imageset_rowid_list]

gets data from the "native" column "imageset_gps_lon" in the "imageset" table

Parameters **imageset_rowid_list** (*list*) –

Returns imageset_gps_lon_list

Return type list

TemplateInfo: Tgetter_table_column col = imageset_gps_lon tbl = imageset

RESTful: Method: GET URL: /api/imageset/gps/lon/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> imageset_rowid_list = ibs._get_all_imageset_rowids()
>>> imageset_gps_lon_list = ibs.get_imageset_gps_lons(imageset_rowid_list)
>>> assert len(imageset_rowid_list) == len(imageset_gps_lon_list)
```

wbia.control.manual_imageset_funcs.**get_imageset_gsgrids**(*ibs*, *imgsetid_list=None*, *gid_list=None*)

Returns a list of imageset-image-relationship rowids for each encouterid

Return type **list_** (list)

wbia.control.manual_imageset_funcs.**get_imageset_image_uuids**(*ibs*, *imgsetid_list*)

Returns a list of list of gids in each imageset

Return type gids_list (list)

RESTful: Method: GET URL: /api/imageset/image/uuid/

wbia.control.manual_imageset_funcs.**get_imageset_imgsetids_from_text**(*ibs*, *imageset_text_list*, *ensure=True*)

Returns a list of imgsetids corresponding to each imageset imagesettext

Return type **list_** (list)

#FIXME: make new naming scheme for non-primary-key-getters get_imageset_imgsetids_from_text_from_text

RESTful: Method: GET URL: /api/imageset/rowid/text/

wbia.control.manual_imageset_funcs.**get_imageset_imgsetids_from_uuid**(*ibs*, *uuid_list*)

Returns a list of imgsetids corresponding to each imageset imagesettext

Return type **list_** (list)

#FIXME: make new naming scheme for non-primary-key-getters get_imageset_imgsetids_from_text_from_text

RESTful: Method: GET URL: /api/imageset/rowid/uuid/

wbia.control.manual_imageset_funcs.**get_imageset_metadata**(*ibs*, *imageset_rowid_list*, *return_raw=False*)

Returns imageset metadata dictionary

Return type list_ (list)

RESTful: Method: GET URL: /api/imageset/metadata/

wbia.control.manual_imageset_funcs.**get_imageset_name_uuids**(*ibs*, *imgsetid_list*)

Returns a list of list of known name uuids in each imageset

Return type name_uuid_list (list)

CommandLine: python -m wbia.control.manual_imageset_funcs –test-get_imageset_name_uuids

RESTful: Method: GET URL: /api/imageset/name/uuid/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> ibs.delete_imagesets(ibs.get_valid_imgsetids())
>>> ibs.compute_occurrences(config={'use_gps': False, 'seconds_thresh': 600})
>>> imgsetid_list = ibs.get_valid_imgsetids()
>>> nids_list = ibs.get_imageset_nids(imgsetid_list)
>>> result = nids_list
>>> print(result)
[[1, 2, 3], [4, 5, 6, 7]]
```

wbia.control.manual_imageset_funcs.**get_imageset_nids**(*ibs*, *imgsetid_list*)

Returns a list of list of known nids in each imageset

Return type list_ (list)

CommandLine: python -m wbia.control.manual_imageset_funcs –test-get_imageset_nids

RESTful: Method: GET URL: /api/imageset/name/rowid/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> ibs.delete_imagesets(ibs.get_valid_imgsetids())
>>> ibs.compute_occurrences(config={'use_gps': False, 'seconds_thresh': 600})
>>> imgsetid_list = ibs.get_valid_imgsetids()
>>> nids_list = ibs.get_imageset_nids(imgsetid_list)
>>> result = nids_list
>>> print(result)
[[1, 2, 3], [4, 5, 6, 7]]
```

wbia.control.manual_imageset_funcs.**get_imageset_note**(*ibs*, *imgsetid_list*)

Returns imageset_note of each imgsetid in imgsetid_list

Return type list_ (list)

RESTful: Method: GET URL: /api/imageset/note/

wbia.control.manual_imageset_funcs.**get_imageset_notes**(*ibs*, *imageset_rowid_list*)
imageset_note_list <- imageset.imageset_note[imageset_rowid_list]

gets data from the "native" column "imageset_note" in the "imageset" table

Parameters **imageset_rowid_list** (*list*) –

Returns imageset_note_list

Return type list

TemplateInfo: Tgetter_table_column col = imageset_note tbl = imageset

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> imageset_rowid_list = ibs._get_all_imageset_rowids()
>>> imageset_note_list = ibs.get_imageset_notes(imageset_rowid_list)
>>> assert len(imageset_rowid_list) == len(imageset_note_list)
```

wbia.control.manual_imageset_funcs.**get_imageset_num_aids**(*ibs*, *imgsetid_list*)

Returns number of images in each imageset

Return type nGids_list (list)

RESTful: Method: GET URL: /api/imageset/num/annot/

wbia.control.manual_imageset_funcs.**get_imageset_num_annotmatch_reviewed**(*ibs*, *imgsetid_list*)

RESTful: Method: GET URL: /api/imageset/num/annotmatch/reviewed/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> imgsetid_list = ibs._get_all_imageset_rowids()
>>> num_annots_reviewed_list = ibs.get_imageset_num_annotmatch_reviewed(imgsetid_
↪list)
```

wbia.control.manual_imageset_funcs.**get_imageset_num_annots_reviewed**(*ibs*, *imgsetid_list*)

RESTful: Method: GET URL: /api/imageset/num/annot/reviewed/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> # Reset and compute imagesets
>>> ibs.delete_all_imagesets()
>>> ibs.compute_occurrences(config={'use_gps': False, 'seconds_thresh': 600})
>>> imgsetid_list = ibs.get_valid_imgsetids()
>>> num_reviwed_list = ibs.get_imageset_num_imgs_reviewed(imgsetid_list)
>>> result = num_reviwed_list
>>> print(result)
[0, 0]
```

wbia.control.manual_imageset_funcs.**get_imageset_num_gids**(*ibs*, *imgsetid_list*)

Returns number of images in each imageset

Return type nGids_list (list)

RESTful: Method: GET URL: /api/imageset/num/image/

wbia.control.manual_imageset_funcs.**get_imageset_num_imgs_reviewed**(*ibs*, *imgsetid_list*)

RESTful: Method: GET URL: /api/imageset/num/image/reviewed/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> # Reset and compute imagesets
>>> ibs.delete_all_imagesets()
>>> ibs.compute_occurrences(config={'use_gps': False, 'seconds_thresh': 600})
>>> imgsetid_list = ibs.get_valid_imgsetids()
>>> num_reviwed_list = ibs.get_imageset_num_imgs_reviewed(imgsetid_list)
>>> result = num_reviwed_list
>>> print(result)
[0, 0]
```

wbia.control.manual_imageset_funcs.**get_imageset_num_names_with_exemplar**(*ibs*, *imgsetid_list*)

RESTful: Method: GET URL: /api/imageset/num/name/exemplar/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> imgsetid_list = ibs._get_all_imageset_rowids()
>>> num_annots_reviewed_list = ibs.get_imageset_num_annotmatch_reviewed(imgsetid_
↪list)
```

wbia.control.manual_imageset_funcs.**get_imageset_occurrence_flags**(*ibs*, *imageset_rowid_list*)

imageset_occurrence_flag_list <- imageset.imageset_occurrence_flag[imageset_rowid_list]

gets data from the "native" column "imageset_occurrence_flag" in the "imageset" table

Parameters **imageset_rowid_list** (*list*) –

Returns imageset_occurrence_flag_list

Return type list

TemplateInfo: Tgetter_table_column col = imageset_occurrence_flag tbl = imageset

RESTful: Method: GET URL: /api/imageset/occurrence/

CommandLine: python -m wbia.control.manual_imageset_funcs –test-get_imageset_occurrence_flags

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> imageset_rowid_list = ibs._get_all_imageset_rowids()
>>> imageset_occurrence_flag_list = ibs.get_imageset_occurrence_flags(imageset_
↪rowid_list)
>>> assert len(imageset_rowid_list) == len(imageset_occurrence_flag_list)
```

wbia.control.manual_imageset_funcs.**get_imageset_percent_annotmatch_reviewed_str**(*ibs*, *imgsetid_list*)

wbia.control.manual_imageset_funcs.**get_imageset_percent_imgs_reviewed_str**(*ibs*, *imgsetid_list*)

wbia.control.manual_imageset_funcs.**get_imageset_percent_names_with_exemplar_str**(*ibs*, *imgsetid_list*)

wbia.control.manual_imageset_funcs.**get_imageset_processed_flags**(*ibs*, *imageset_rowid_list*)

imageset_processed_flag_list <- imageset.imageset_processed_flag[imageset_rowid_list]

gets data from the "native" column "imageset_processed_flag" in the "imageset" table

Parameters **imageset_rowid_list** (*list*) –

Returns imageset_processed_flag_list

Return type list

TemplateInfo: Tgetter_table_column col = imageset_processed_flag tbl = imageset

RESTful: Method: GET URL: /api/imageset/processed/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> imageset_rowid_list = ibs._get_all_imageset_rowids()
```

(continues on next page)

(continued from previous page)

```
>>> imageset_processed_flag_list = ibs.get_imageset_processed_flags(imageset_
↪rowid_list)
>>> assert len(imageset_rowid_list) == len(imageset_processed_flag_list)
```

wbia.control.manual_imageset_funcs.**get_imageset_shipped_flags**(*ibs*, *imageset_rowid_list*)

imageset_shipped_flag_list <- imageset.imageset_shipped_flag[imageset_rowid_list]

gets data from the "native" column "imageset_shipped_flag" in the "imageset" table

Parameters **imageset_rowid_list** (*list*) –

Returns imageset_shipped_flag_list

Return type list

TemplateInfo: Tgetter_table_column col = imageset_shipped_flag tbl = imageset

RESTful: Method: GET URL: /api/imageset/shipped/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> imageset_rowid_list = ibs._get_all_imageset_rowids()
>>> imageset_shipped_flag_list = ibs.get_imageset_shipped_flags(imageset_rowid_
↪list)
>>> assert len(imageset_rowid_list) == len(imageset_shipped_flag_list)
```

wbia.control.manual_imageset_funcs.**get_imageset_smart_waypoint_ids**(*ibs*, *imageset_rowid_list*)

imageset_smart_waypoint_id_list <- imageset.imageset_smart_waypoint_id[imageset_rowid_list]

gets data from the "native" column "imageset_smart_waypoint_id" in the "imageset" table

Parameters **imageset_rowid_list** (*list*) –

Returns imageset_smart_waypoint_id_list

Return type list

TemplateInfo: Tgetter_table_column col = imageset_smart_waypoint_id tbl = imageset

RESTful: Method: GET URL: /api/imageset/smart/waypoint/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> imageset_rowid_list = ibs._get_all_imageset_rowids()
>>> imageset_smart_waypoint_id_list = ibs.get_imageset_smart_waypoint_
↪ids(imageset_rowid_list)
>>> assert len(imageset_rowid_list) == len(imageset_smart_waypoint_id_list)
```

wbia.control.manual_imageset_funcs.**get_imageset_smart_xml_contents**(*ibs*, *imageset_rowid_list*)

wbia.control.manual_imageset_funcs.**get_imageset_smart_xml_fnames**(*ibs*, *imageset_rowid_list*)

imageset_smart_xml_fname_list <- imageset.imageset_smart_xml_fname[imageset_rowid_list]

gets data from the "native" column "imageset_smart_xml_fname" in the "imageset" table

Parameters **imageset_rowid_list** (*list*) –

Returns imageset_smart_xml_fname_list

Return type list

TemplateInfo: Tgetter_table_column col = imageset_smart_xml_fname tbl = imageset

RESTful: Method: GET URL: /api/imageset/smart/xml/file/name/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> imageset_rowid_list = ibs._get_all_imageset_rowids()
>>> imageset_smart_xml_fname_list = ibs.get_imageset_smart_xml_fnames(imageset_
↪rowid_list)
>>> assert len(imageset_rowid_list) == len(imageset_smart_xml_fname_list)
```

wbia.control.manual_imageset_funcs.**get_imageset_start_time_posix**(*ibs*, *imageset_rowid_list*)

imageset_start_time_posix_list <- imageset.imageset_start_time_posix[imageset_rowid_list]

gets data from the "native" column "imageset_start_time_posix" in the "imageset" table

Parameters **imageset_rowid_list** (*list*) –

Returns imageset_start_time_posix_list

Return type list

TemplateInfo: Tgetter_table_column col = imageset_start_time_posix tbl = imageset

RESTful: Method: GET URL: /api/imageset/time/posix/start/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> imageset_rowid_list = ibs._get_all_imageset_rowids()
>>> imageset_start_time_posix_list = ibs.get_imageset_start_time_posix(imageset_
↪rowid_list)
>>> assert len(imageset_rowid_list) == len(imageset_start_time_posix_list)
```

wbia.control.manual_imageset_funcs.**get_imageset_text**(*ibs*, *imgsetid_list*)

Returns imageset_text of each imgsetid in imgsetid_list

Return type list_ (list)

RESTful: Method: GET URL: /api/imageset/text/

wbia.control.manual_imageset_funcs.**get_imageset_uuid**(*ibs*, *imgsetid_list*)

Returns imageset_uuid of each imgsetid in imgsetid_list

Return type list_ (list)

RESTful: Method: GET URL: /api/imageset/uuid/

wbia.control.manual_imageset_funcs.**get_imageset_uuids**(*ibs*, *imgsetid_list*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **imgsetid_list** (*list*) –

Returns annot_uuids_list

Return type list

RESTful: Method: GET URL: /api/imageset/annot/uuid/

CommandLine: python -m wbia.control.manual_imageset_funcs –test-get_imageset_aids

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_imageset_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> ibs.delete_imagesets(ibs.get_valid_imgsetids())
>>> ibs.compute_occurrences(config={'use_gps': False, 'seconds_thresh': 600})
>>> imgsetid_list = ibs.get_valid_imgsetids()
>>> aids_list = get_imageset_aids(ibs, imgsetid_list)
>>> result = ('aids_list = %s' % (str(aids_list),))
>>> print(result)
```

wbia.control.manual_imageset_funcs.**get_valid_imgsetids**(*ibs*, *min_num_gids=0*, *processed=None*, *shipped=None*, *is_occurrence=None*, *is_special=None*)

FIX NAME imgagesetids

Returns list of all imageset ids

Return type list_ (list)

RESTful: Method: GET URL: /api/imageset/

wbia.control.manual_imageset_funcs.**is_special_imageset**(*ibs*, *imgsetid_list*)

wbia.control.manual_imageset_funcs.**set_imageset_end_time_posix**(*ibs*, *imageset_rowid_list*, *imageset_end_time_posix_list*)

imageset_end_time_posix_list -> imageset.imageset_end_time_posix[imageset_rowid_list]

Parameters

- **imageset_rowid_list** –
- **imageset_end_time_posix_list** –

TemplateInfo: Tsetter_native_column tbl = imageset col = imageset_end_time_posix

RESTful: Method: PUT URL: /api/imageset/time/posix/end/

wbia.control.manual_imageset_funcs.**set_imageset_gps_lats**(*ibs*, *imageset_rowid_list*, *imageset_gps_lat_list*)

imageset_gps_lat_list -> imageset.imageset_gps_lat[imageset_rowid_list]

Parameters

- **imageset_rowid_list** –
- **imageset_gps_lat_list** –

TemplateInfo: Tsetter_native_column tbl = imageset col = imageset_gps_lat

RESTful: Method: PUT URL: /api/imageset/gps/lat/

wbia.control.manual_imageset_funcs.**set_imageset_gps_lons**(*ibs*, *imageset_rowid_list*, *imageset_gps_lon_list*)

imageset_gps_lon_list -> imageset.imageset_gps_lon[imageset_rowid_list]

Parameters

- **imageset_rowid_list** –
- **imageset_gps_lon_list** –

TemplateInfo: Tsetter_native_column tbl = imageset col = imageset_gps_lon

RESTful: Method: PUT URL: /api/imageset/gps/lon/

wbia.control.manual_imageset_funcs.**set_imageset_metadata**(*ibs*, *imageset_rowid_list*, *metadata_dict_list*)

Sets the imageset's metadata using a metadata dictionary

RESTful: Method: PUT URL: /api/imageset/metadata/

wbia.control.manual_imageset_funcs.**set_imageset_notes**(*ibs*, *imageset_rowid_list*, *imageset_note_list*)

imageset_note_list -> imageset.imageset_note[imageset_rowid_list]

Parameters

- **imageset_rowid_list** –
- **imageset_note_list** –

TemplateInfo: Tsetter_native_column tbl = imageset col = imageset_note

wbia.control.manual_imageset_funcs.**set_imageset_occurrence_flags**(*ibs*, *imageset_rowid_list*, *imageset_occurrence_flag_list*)

imageset_occurrence_flag_list -> imageset.imageset_occurrence_flag[imageset_rowid_list]

Parameters

- **imageset_rowid_list** –
- **imageset_occurrence_flag_list** –

TemplateInfo: Tsetter_native_column tbl = imageset col = imageset_occurrence_flag

RESTful: Method: PUT URL: /api/imageset/occurrence/

wbia.control.manual_imageset_funcs.**set_imageset_processed_flags**(*ibs*, *imageset_rowid_list*, *imageset_processed_flag_list*)

imageset_processed_flag_list -> imageset.imageset_processed_flag[imageset_rowid_list]

Parameters

- **imageset_rowid_list** –
- **imageset_processed_flag_list** –

TemplateInfo: Tsetter_native_column tbl = imageset col = imageset_processed_flag

RESTful: Method: PUT URL: /api/imageset/processed/

wbia.control.manual_imageset_funcs.**set_imageset_shipped_flags**(*ibs*, *imageset_rowid_list*, *imageset_shipped_flag_list*)

imageset_shipped_flag_list -> imageset.imageset_shipped_flag[imageset_rowid_list]

Parameters

- **imageset_rowid_list** –
- **imageset_shipped_flag_list** –

TemplateInfo: Tsetter_native_column tbl = imageset col = imageset_shipped_flag

RESTful: Method: PUT URL: /api/imageset/shipped/

wbia.control.manual_imageset_funcs.**set_imageset_smart_waypoint_ids**(*ibs*, *imageset_rowid_list*, *imageset_smart_waypoint_id_list*)

imageset_smart_waypoint_id_list -> imageset.imageset_smart_waypoint_id[imageset_rowid_list]

Parameters

- **imageset_rowid_list** –
- **imageset_smart_waypoint_id_list** –

TemplateInfo: Tsetter_native_column tbl = imageset col = imageset_smart_waypoint_id

RESTful: Method: PUT URL: /api/imageset/smart/waypoint/

wbia.control.manual_imageset_funcs.**set_imageset_smart_xml_fnames**(*ibs*, *imageset_rowid_list*, *imageset_smart_xml_fname_list*)

imageset_smart_xml_fname_list -> imageset.imageset_smart_xml_fname[imageset_rowid_list]

Parameters

- **imageset_rowid_list** –
- **imageset_smart_xml_fname_list** –

TemplateInfo: Tsetter_native_column tbl = imageset col = imageset_smart_xml_fname

RESTful: Method: PUT URL: /api/imageset/smart/xml/fname/

wbia.control.manual_imageset_funcs.**set_imageset_start_time_posix**(*ibs*, *imageset_rowid_list*, *imageset_start_time_posix_list*)

imageset_start_time_posix_list -> imageset.imageset_start_time_posix[imageset_rowid_list]

Parameters

- **imageset_rowid_list** –
- **imageset_start_time_posix_list** –

TemplateInfo: Tsetter_native_column tbl = imageset col = imageset_start_time_posix

RESTful: Method: PUT URL: /api/imageset/time/posix/start/

wbia.control.manual_imageset_funcs.**set_imageset_text**(*ibs*, *imgsetid_list*, *imageset_text_list*)

Sets names of imagesets (groups of animals)

RESTful: Method: PUT URL: /api/imageset/text/

wbia.control.manual_imageset_funcs.**testdata_ibs**()

wbia.control.manual_imageset_funcs.**update_imageset_info**(*ibs*, *imageset_rowid_list*, ***kwargs*)

sets start and end time for imagesets

FIXME: should not need to bulk update, should be handled as it goes

RESTful: Method: PUT URL: /api/imageset/info/

Example

```
>>> # DISABLE_DOCTEST
>>> imageset_rowid_list = ibs.get_valid_imgsetids()
```

1.2.23 wbia.control.manual_lblannot_funcs module

wbia.control.manual_lblannot_funcs.**add_annot_relationship**(*ibs*, *aid_list*, *lblannot_rowid_list*, *alr_confidence_list=None*)

Adds a relationship between annots and lblannots (annotations and labels of annotations)

wbia.control.manual_lblannot_funcs.**add_lblannots**(*ibs*, *lbltype_rowid_list*, *value_list*, *note_list=None*, *lblannot_uuid_list=None*)

Adds new lblannots (labels of annotations) creates a new uuid for any new pair(type, value) #TODO: reverse order of rowid_list value_list in input

wbia.control.manual_lblannot_funcs.**delete_annot_relations**(*ibs*, *aid_list*)

Deletes the relationship between an annotation and a label

wbia.control.manual_lblannot_funcs.**delete_annot_relations_oftype**(*ibs*, *aid_list*, *_lbltype*)

Deletes the relationship between an annotation and a label

wbia.control.manual_lblannot_funcs.**delete_lblannots**(*ibs*, *lblannot_rowid_list*)

deletes lblannots from the database

wbia.control.manual_lblannot_funcs.**get_alr_annot_rowids**(*ibs*, *alrid_list*)

Parameters **alrid_list** (*list of rowids*) – annot + label relationship rows

get the annot_rowid belonging to each relationship

wbia.control.manual_lblannot_funcs.**get_alr_annot_rowids_from_lblannot_rowid**(*ibs*, *lblannot_rowid_list*)

This is a 1toM getter

Get annotation rowids of labels. There may be more than one annotation per label.

Parameters **lblannot_rowid_list** (*list*) – of lblannot (labels of annotations) rowids

Returns of lists annotation rowids

Return type aids_list (list)

wbia.control.manual_lblannot_funcs.**get_alr_confidence**(*ibs*, *alrid_list*)

Parameters **alrid_list** (*list of rowids*) – annot + label relationship rows

Returns confidence in an annotation relationship

Return type alr_confidence_list (list of rowids)

wbia.control.manual_lblannot_funcs.**get_alr_lblannot_rowids**(*ibs*, *alrid_list*)

Parameters **alrid_list** (*list of rowids*) – annot + label relationship rows

Returns label rowids (of annotations)

Return type lblannot_rowids_list (list of rowids)

wbia.control.manual_lblannot_funcs.**get_alrid_from_superkey**(*ibs*, *aid_list*, *lblannot_rowid_list*)

Parameters

- **aid_list** (*list*) – list of annotation row-ids
- **lblannot_rowid_list** (*list*) – list of lblannot row-ids

Returns annot-label relationship id list

Return type alrid_list (list)

wbia.control.manual_lblannot_funcs.**get_annot_alrids**(*ibs*, *aid_list*)
FIXME: __name__ Get all the relationship ids belonging to the input annotations if lblannot lbltype is specified the relationship ids are filtered to be only of a specific lbltype/category/type

wbia.control.manual_lblannot_funcs.**get_annot_alrids_oftype**(*ibs*, *aid_list*, *lbltype_rowid*)
Get all the relationship ids belonging to the input annotations where the relationship ids are filtered to be only of a specific lbltype/category/type

wbia.control.manual_lblannot_funcs.**get_annot_lblannot_rowids**(*ibs*, *aid_list*)

Returns the name id of each annotation.

Return type list_ (list)

wbia.control.manual_lblannot_funcs.**get_annot_lblannot_rowids_oftype**(*ibs*, *aid_list*, *_lbltype=None*)

Returns the name id of each annotation.

Return type list_ (list)

wbia.control.manual_lblannot_funcs.**get_annot_lblannot_value_of_lbltype**(*ibs*, *aid_list*, *_lbltype*, *lblannot_value_getter*)

Returns a list of strings ['fred', 'sue', ...] for each chip identifying the animal

Return type lblannot_value_list (list)

wbia.control.manual_lblannot_funcs.**get_lblannot_lbltypes_rowids**(*ibs*, *lblannot_rowid_list*)

wbia.control.manual_lblannot_funcs.**get_lblannot_notes**(*ibs*, *lblannot_rowid_list*)

wbia.control.manual_lblannot_funcs.**get_lblannot_rowid_from_superkey**(*ibs*, *lbltype_rowid_list*, *value_list*)

Returns lblannot_rowid_list from the superkey (lbltype, value)

Return type list_ (list)

wbia.control.manual_lblannot_funcs.**get_lblannot_rowid_from_uuid**(*ibs*, *lblannot_uuid_list*)
UNSAFE

Returns lblannot_rowid_list from the superkey (lbltype, value)

wbia.control.manual_lblannot_funcs.**get_lblannot_uuids**(*ibs*, *lblannot_rowid_list*)

wbia.control.manual_lblannot_funcs.**get_lblannot_values**(*ibs*, *lblannot_rowid_list*, *_lbltype=None*)

Returns text lblannots

wbia.control.manual_lblannot_funcs.**set_alr_confidence**(*ibs*, *alrid_list*, *confidence_list*)
sets annotation-lblannot-relationship confidence

wbia.control.manual_lblannot_funcs.**set_alr_lblannot_rowids**(*ibs*, *alrid_list*, *lblannot_rowid_list*)

Associates whatever annotation is at row(alrid) with a new lblannot_rowid. (effectively changes the label value of the rowid)

wbia.control.manual_lblannot_funcs.**set_annot_lblannot_from_rowid**(*ibs*, *aid_list*, *lblannot_rowid_list*, *_lbltype*)

Sets items/lblannot_rowids of a list of annotations.

wbia.control.manual_lblannot_funcs.**set_annot_lblannot_from_value**(*ibs*, *aid_list*, *value_list*, *_lbltype*, *ensure=True*)

Associates the annot and lblannot of a specific type and value Adds the lblannot if it doesnt exist. Wrapper around convenience function for set_annot_from_lblannot_rowid

wbia.control.manual_lblannot_funcs.**set_lblannot_notes**(*ibs*, *lblannot_rowid_list*, *value_list*)

Updates the value for lblannots. Note this change applies to all annotations related to this lblannot_rowid

wbia.control.manual_lblannot_funcs.**set_lblannot_values**(*ibs*, *lblannot_rowid_list*, *value_list*)

Updates the value for lblannots. Note this change applies to all annotations related to this lblannot_rowid

1.2.24 wbia.control.manual_lblimage_funcs module

wbia.control.manual_lblimage_funcs.**add_image_relationship_one**(*ibs*, *gid_list*, *lblimage_rowid_list*, *glr_confidence_list=None*)

Adds a relationship between images and lblimages (imageations and labels of imageations)

wbia.control.manual_lblimage_funcs.**add_lblimages**(*ibs*, *lbltype_rowid_list*, *value_list*, *note_list=None*, *lblimage_uuid_list=None*)

Adds new lblimages (labels of imageations) creates a new uuid for any new pair(type, value) #TODO: reverse order of rowid_list value_list in input

wbia.control.manual_lblimage_funcs.**get_glr_confidence**(*ibs*, *glrid_list*)

Returns confidence in an image relationship

Return type list_ (list)

wbia.control.manual_lblimage_funcs.**get_glr_image_rowids**(*ibs*, *glrid_list*)

get the image_rowid belonging to each relationship

wbia.control.manual_lblimage_funcs.**get_glr_lblimage_rowids**(*ibs*, *glrid_list*)

get the lblimage_rowid belonging to each relationship

wbia.control.manual_lblimage_funcs.**get_glrid_from_superkey**(*ibs*, *gid_list*, *lblimage_rowid_list*)

Parameters

- **gid_list** (*list*) – list of image row-ids
- **lblimage_rowid_list** (*list*) – list of lblimage row-ids

Returns image-label relationship id list

Return type glrid_list (list)

wbia.control.manual_lblimage_funcs.**get_image_glrids**(*ibs*, *gid_list*)
FIXME: __name__ Get all the relationship ids belonging to the input images if lblimage lbltype is specified the relationship ids are filtered to be only of a specific lbltype/category/type

wbia.control.manual_lblimage_funcs.**get_lblimage_gids**(*ibs*, *lblimage_rowid_list*)

wbia.control.manual_lblimage_funcs.**get_lblimage_lbltypes_rowids**(*ibs*, *lblimage_rowid_list*)

wbia.control.manual_lblimage_funcs.**get_lblimage_notes**(*ibs*, *lblimage_rowid_list*)

wbia.control.manual_lblimage_funcs.**get_lblimage_rowid_from_superkey**(*ibs*, *lbltype_rowid_list*, *value_list*)

Returns lblimage_rowid_list from the superkey (lbltype, value)

Return type list_ (list)

wbia.control.manual_lblimage_funcs.**get_lblimage_rowid_from_uuid**(*ibs*, *lblimage_uuid_list*)

Returns lblimage_rowid_list from the superkey (lbltype, value)

Return type list_ (list)

wbia.control.manual_lblimage_funcs.**get_lblimage_uuids**(*ibs*, *lblimage_rowid_list*)

wbia.control.manual_lblimage_funcs.**get_lblimage_values**(*ibs*, *lblimage_rowid_list*, *_lbltype=None*)

Returns text lblimages

Return type list_ (list)

1.2.25 wbia.control.manual_lbltype_funcs module

wbia.control.manual_lbltype_funcs.**add_lbltype**(*ibs*, *text_list*, *default_list*)
Adds a label type and its default value Should only be called at the begining of the program.

wbia.control.manual_lbltype_funcs.**get_lbltype_default**(*ibs*, *lbltype_rowid_list*)

wbia.control.manual_lbltype_funcs.**get_lbltype_rowid_from_text**(*ibs*, *text_list*)

Returns lbltype_rowid where the lbltype_text is given

Return type lbltype_rowid (list)

wbia.control.manual_lbltype_funcs.**get_lbltype_text**(*ibs*, *lbltype_rowid_list*)

1.2.26 wbia.control.manual_meta_funcs module

controller functions for contributors, versions, configs, and other metadata

wbia.control.manual_meta_funcs.**add_contributors**(*ibs*, *tag_list*, *uuid_list=None*, *name_first_list=None*, *name_last_list=None*, *loc_city_list=None*, *loc_state_list=None*, *loc_country_list=None*, *loc_zip_list=None*, *notes_list=None*)

Adds a list of contributors.

Returns contributor rowids

Return type contributor_id_list (list)

RESTful: Method: POST URL: /api/contributor/

wbia.control.manual_meta_funcs.**add_metadata**(*ibs*, *metadata_key_list*, *metadata_value_list*, *db*)

Adds metadata

Returns metadata rowids

Return type metadata_rowid_list (list)

RESTful: Method: POST URL: /api/metadata/

wbia.control.manual_meta_funcs.**add_new_temp_contributor**(*ibs*, *user_prompt=False*, *offset=None*, *autolocate=False*)

RESTful: Method: POST URL: /api/contributor/new/temp/

wbia.control.manual_meta_funcs.**add_version**(*ibs*, *versiontext_list*)

Adds an algorithm / actor configuration as a string

wbia.control.manual_meta_funcs.**delete_contributors**(*ibs*, *contributor_rowid_list*)

deletes contributors from the database and all information associated

RESTful: Method: DELETE URL: /api/contributor/

wbia.control.manual_meta_funcs.**ensure_contributor_rowids**(*ibs*, *user_prompt=False*, *autolocate=False*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **user_prompt** (*bool*) –

Returns

Return type list

CommandLine: python -m wbia.control.manual_meta_funcs –test-ensure_contributor_rowids

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_meta_funcs import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb(db='testdb1')
>>> gid_list = ibs.get_valid_gids()
>>> ibs.delete_contributors(ibs.get_valid_contributor_rowids())
>>> contributor_rowid_list1 = ibs.get_image_contributor_rowid(gid_list)
>>> assert ut.allsame(contributor_rowid_list1)
>>> ut.assert_eq(contributor_rowid_list1[0], None)
>>> user_prompt = ut.get_argflag('--user-prompt')
>>> autolocate = ut.get_argflag('--user-prompt')
```

(continues on next page)

(continued from previous page)

```
>>> # execute function
>>> result = ensure_contributor_rowids(ibs, user_prompt, autolocate)
>>> # verify results
>>> ibs.print_contributor_table()
>>> print(result)
>>> contributor_rowid_list2 = ibs.get_image_contributor_rowid(gid_list)
>>> assert ut.allsame(contributor_rowid_list2)
>>> ut.assert_eq(contributor_rowid_list2[0], 1)
```

wbia.control.manual_meta_funcs.**get_all_uncontributed_configs**(*ibs*)

RESTful: Method: GET URL: /api/contributor/configs/uncontributed/

wbia.control.manual_meta_funcs.**get_all_uncontributed_images**(*ibs*)

RESTful: Method: GET URL: /api/contributor/gids/uncontributed/

wbia.control.manual_meta_funcs.**get_config_contributor_rowid**(*ibs*, *config_rowid_list*)

Returns contributor's rowid for algorithm configs

Return type cfgsuffix_list (list)

RESTful: Method: GET URL: /api/contributor/config/rowid/

wbia.control.manual_meta_funcs.**get_config_suffixes**(*ibs*, *config_rowid_list*)

Returns suffixes for algorithm configs

Return type cfgsuffix_list (list)

RESTful: Method: GET URL: /api/contributor/config/suffixes/

wbia.control.manual_meta_funcs.**get_contributor_city**(*ibs*, *contributor_rowid_list*)

Returns a contributor's location - city

Return type contributor_city_list (list)

RESTful: Method: GET URL: /api/contributor/location/city/

wbia.control.manual_meta_funcs.**get_contributor_country**(*ibs*, *contributor_rowid_list*)

Returns a contributor's location - country

Return type contributor_country_list (list)

RESTful: Method: GET URL: /api/contributor/location/country/

wbia.control.manual_meta_funcs.**get_contributor_first_name**(*ibs*, *contributor_rowid_list*)

Returns a contributor's first name

Return type contributor_name_first_list (list)

RESTful: Method: GET URL: /api/contributor/name/first/

wbia.control.manual_meta_funcs.**get_contributor_gids**(*ibs*, *contributor_rowid_list*)

TODO: Template 1_M reverse getter

Returns gids for a contributor

Return type gid_list (list)

RESTful: Method: GET URL: /api/contributor/gids/

wbia.control.manual_meta_funcs.**get_contributor_imgsetids**(*ibs*, *config_rowid_list*)

Returns imgsetids for a contributor

Return type imgsetid_list (list)

RESTful: Method: GET URL: /api/contributor/imageset/rowids/

wbia.control.manual_meta_funcs.**get_contributor_last_name**(*ibs*, *contributor_rowid_list*)

Returns a contributor's last name

Return type contributor_name_last_list (list)

RESTful: Method: GET URL: /api/contributor/name/last/

wbia.control.manual_meta_funcs.**get_contributor_location_string**(*ibs*, *contributor_rowid_list*)

Returns a contributor's location

Return type contributor_list (list)

RESTful: Method: GET URL: /api/contributor/location/

wbia.control.manual_meta_funcs.**get_contributor_name_string**(*ibs*, *contributor_rowid_list*, *include_tag=False*)

Returns a contributor's full name

Return type contributor_name_list (list)

RESTful: Method: GET URL: /api/contributor/name/

wbia.control.manual_meta_funcs.**get_contributor_note**(*ibs*, *contributor_rowid_list*)

Returns a contributor's note

Return type contributor_note_list (list)

RESTful: Method: GET URL: /api/contributor/note/

wbia.control.manual_meta_funcs.**get_contributor_rowid_from_tag**(*ibs*, *contributor_tag_list*)

Returns a contributor

Return type contributor_tag_list (list)

RESTful: Method: GET URL: /api/contributor/rowid/tag/

wbia.control.manual_meta_funcs.**get_contributor_rowid_from_uuid**(*ibs*, *contributor_uuid_list*)

Returns a contributor

Return type contributor_uuid_list (list)

RESTful: Method: GET URL: /api/contributor/rowid/uuid/

wbia.control.manual_meta_funcs.**get_contributor_state**(*ibs*, *contributor_rowid_list*)

Returns a contributor's location - state

Return type list_ (list)

RESTful: Method: GET URL: /api/contributor/location/state/

wbia.control.manual_meta_funcs.**get_contributor_tag**(*ibs*, *contributor_rowid_list*, *eager=True*, *nInput=None*)

contributor_tag_list <- contributor.contributor_tag[contributor_rowid_list]

gets data from the "native" column "contributor_tag" in the "contributor" table

Parameters **contributor_rowid_list** (*list*) –

Returns contributor_tag_list - a contributor's tag

Return type list

TemplateInfo: Tgetter_table_column col = contributor_tag tbl = contributor

CommandLine: python -m wbia.templates.template_generator –key contributor –Tcfg with_api_cache=False with_deleters=False

RESTful: Method: GET URL: /api/contributor/tag/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_meta_funcs import *  # NOQA
>>> ibs, qreq_ = testdata_ibs()
>>> contributor_rowid_list = ibs._get_all_contributor_rowids()
>>> eager = True
>>> contributor_tag_list = ibs.get_contributor_tag(contributor_rowid_list,␣
↪eager=eager)
>>> assert len(contributor_rowid_list) == len(contributor_tag_list)
```

wbia.control.manual_meta_funcs.**get_contributor_uuid**(*ibs*, *contributor_rowid_list*)

Returns a contributor's uuid

Return type contributor_uuid_list (list)

RESTful: Method: GET URL: /api/contributor/uuid/

wbia.control.manual_meta_funcs.**get_contributor_zip**(*ibs*, *contributor_rowid_list*)

Returns a contributor's location - zip

Return type contributor_zip_list (list)

RESTful: Method: GET URL: /api/contributor/location/zip/

wbia.control.manual_meta_funcs.**get_database_version**(*ibs*, *db=None*)
Gets the specified database version from the controller

RESTful: Method: GET URL: /api/core/dbversion/

wbia.control.manual_meta_funcs.**get_database_version_alias**(*ibs*, *db=None*)
Alias: *func:get_database_version*

RESTful: Method: GET URL: /api/core/version/

wbia.control.manual_meta_funcs.**get_metadata_rowid_from_metadata_key**(*ibs*, *metadata_key_list*, *db*)

RESTful: Method: GET URL: /api/metadata/rowid/key/

wbia.control.manual_meta_funcs.**get_metadata_value**(*ibs*, *metadata_key_list*, *db*)

RESTful: Method: GET URL: /api/metadata/value/

wbia.control.manual_meta_funcs.**get_valid_contributor_rowids**(*ibs*)

Returns list of all contributor ids

Return type list_ (list)

Returns contributor_rowids_list

Return type list

CommandLine: python -m wbia.control.manual_meta_funcs –test-get_valid_contributor_rowids

RESTful: Method: GET URL: /api/contributor/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_meta_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> contributor_rowids_list = get_valid_contributor_rowids(ibs)
>>> result = str(contributor_rowids_list)
>>> print(result)
```

wbia.control.manual_meta_funcs.**get_version**(*ibs*)
Returns the version of wbia

RESTful Method: GET URL: /api/version/

wbia.control.manual_meta_funcs.**set_config_contributor_rowid**(*ibs*, *config_rowid_list*, *contributor_rowid_list*)
Sets the config's contributor rowid

RESTful: Method: PUT URL: /api/config/contributor/rowid/

wbia.control.manual_meta_funcs.**set_config_contributor_unassigned**(*ibs*, *contributor_rowid*)

RESTful: Method: PUT URL: /api/config/contributor/unassigned/

wbia.control.manual_meta_funcs.**set_database_version**(*ibs*, *db*, *version*)
Sets the specified database's version from the controller

wbia.control.manual_meta_funcs.**set_metadata_value**(*ibs*, *metadata_key_list*, *metadata_value_list*, *db*)

Sets metadata key, value pairs

RESTful: Method: PUT URL: /api/metadata/value/

wbia.control.manual_meta_funcs.**testdata_ibs**()

wbia.control.manual_meta_funcs.**update_query_cfg**(*ibs*, ***kwargs*)

Updates query config only. Configs needs a restructure very badly DEPRICATE

RESTful: Method: PUT URL: /api/query/cfg/

1.2.27 wbia.control.manual_name_funcs module

python -c "import utool as ut; ut.write_modscript_alias('Tgen.sh', 'wbia.templates.template_generator')" # NOQA sh Tgen.sh –key name –invert –Tcfg with_getters=True with_setters=False –modfname manual_name_funcs # NOQA sh Tgen.sh –key name –invert –Tcfg with_getters=True with_setters=True –modfname manual_name_funcs –funcnamefilter=sex # NOQA

wbia.control.manual_name_funcs.**add_names**(*ibs*, *name_text_list*, *name_uuid_list=None*, *name_note_list=None*)

Adds a list of names.

Returns their nids

Return type name_rowid_list (list)

RESTful: Method: POST URL: /api/name/

wbia.control.manual_name_funcs.**delete_empty_nids**(*ibs*)

Removes names that have no Rois from the database

wbia.control.manual_name_funcs.**delete_names**(*ibs*, *name_rowid_list*, *safe=True*, *strict=False*, *verbose=False*)

deletes names from the database

CAREFUL. YOU PROBABLY DO NOT WANT TO USE THIS at least ensure that no annot is associated with any of these nids

RESTful: Method: DELETE URL: /api/name/

Ignore: # >>> # UNPORTED_DOCTEST # >>> gpath_list = grabdata.get_test_gpaths(ndata=None)[0:4] # >>> gid_list = ibs.add_images(gpath_list) # >>> bbox_list = [(0, 0, 100, 100)]*len(gid_list) # >>> name_list = ['a', 'b', 'a', 'd'] # >>> aid_list = ibs.add_annots(gid_list, bbox_list=bbox_list, name_list=name_list) # >>> assert len(aid_list) != 0, "No annotations added" # >>> nid_list = ibs.get_valid_nids() # >>> assert len(nid_list) != 0, "No names added" # >>> nid = nid_list[0] # >>> assert nid is not None, "nid is None" # >>> ibs.delete_names(nid) # >>> all_nids = ibs.get_valid_nids() # >>> assert nid not in all_nids, "NID not deleted"

wbia.control.manual_name_funcs.**get_empty_nids**(*ibs*, *_nid_list=None*)

get name rowids that do not have any annotations (not including UNKONWN)

Returns nid_list - all names without any animals (does not include unknown names) an nid is not invalid if it has a valid alias

Return type list

CommandLine: python -m wbia.control.manual_name_funcs –test-get_empty_nids

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> new_nid_list = ibs.make_next_nids(num=2)
>>> empty_nids = ibs.get_empty_nids()
>>> assert len(empty_nids) == 2, 'get_empty_nids fails1'
>>> assert new_nid_list == empty_nids, 'get_empty_nids fails2'
>>> ibs.delete_empty_nids()
>>> empty_nids2 = ibs.get_empty_nids()
>>> assert len(empty_nids2) == 0, 'get_empty_nids fails3'
>>> result = str(empty_nids2)
>>> print(result)
[]
```

wbia.control.manual_name_funcs.**get_name_age_months_est_max**(*ibs*, *name_rowid_list*)

RESTful: Method: GET URL: /api/name/age/months/max/

wbia.control.manual_name_funcs.**get_name_age_months_est_min**(*ibs*, *name_rowid_list*)

RESTful: Method: GET URL: /api/name/age/months/min/

wbia.control.manual_name_funcs.**get_name_aids**(*ibs*, *nid_list*, *enable_unknown_fix=True*, *is_staged=False*)

TODO: Rename to get_anot_rowids_from_name_rowid

Returns aids_list a list of list of aids in each name

Return type list

RESTful: Method: GET URL: /api/name/annot/rowid/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> # Map annotations to name ids
>>> aid_list = ibs.get_valid_aids()
>>> nid_list = ibs.get_annot_name_rowids(aid_list)
>>> # Get annotation ids for each name
>>> aids_list = ibs.get_name_aids(nid_list)
>>> # Run Assertion Test
>>> groupid2_items = ut.group_items(aids_list, nid_list)
>>> grouped_items = list(groupid2_items.values())
>>> passed_iter = map(ut.allsame, grouped_items)
>>> passed_list = list(passed_iter)
>>> assert all(passed_list), 'problem in get_name_aids'
>>> # Print gropued items
>>> print(ut.repr2(groupid2_items, newlines=False))
```

Ignore; from wbia.control.manual_name_funcs import * # NOQA import wbia #ibs = wbia.opendb('testdb1') #ibs = wbia.opendb('PZ_MTEST') ibs = wbia.opendb('PZ_Master0') #ibs = wbia.opendb('GZ_ALL')

nid_list = ibs.get_valid_nids() **nid_list_** = [const.UNKNOWN_NAME_ROWID if nid <= 0 else nid for nid in nid_list]

with ut.Timer('sql'): #aids_list1 = ibs.get_name_aids(nid_list, enable_unknown_fix=False) aids_list1 = ibs.db.get(const.ANNOTATION_TABLE, (ANNOT_ROWID,), **nid_list_**, id_colname=NAME_ROWID, unpack_scalars=False)

with ut.Timer('hackquery + group'): opstr = ''' SELECT annot_rowid, name_rowid FROM annotations WHERE name_rowid IN

(%s) ORDER BY name_rowid ASC, annot_rowid ASC

''' % (', '.join(map(str, nid_list))) pair_list = ibs.db.connection.execute(opstr).fetchall() aids = np.array(ut.get_list_column(pair_list, 0)) nids = np.array(ut.get_list_column(pair_list, 1)) unique_nids, groupx = vt.group_indices(nids) **grouped_aids_** = vt.apply_grouping(aids, groupx) aids_list5 = [sorted(arr.tolist()) for arr in **grouped_aids_**]

for aids1, aids5 in zip(aids_list1, aids_list5):

if (aids1) != (aids5): logger.info(aids1) logger.info(aids5) logger.info('——')

ut.assert_lists_eq(list(map(tuple, aids_list5)), list(map(tuple, aids_list1)))

with ut.Timer('numpy'): # alt method valid_aids = np.array(ibs.get_valid_aids()) valid_nids = np.array(ibs.get_annot_name_rowids(valid_aids, distinguish_unknowns=False)) aids_list2 = [valid_aids.take(np.flatnonzero(valid_nids == nid)).tolist() for nid in **nid_list_**]

with ut.Timer('numpy2'): # alt method valid_aids = np.array(ibs.get_valid_aids()) valid_nids = np.array(ibs.get_annot_name_rowids(valid_aids, distinguish_unknowns=False)) aids_list3 = [valid_aids.take(np.flatnonzero(np.equal(valid_nids, nid))).tolist() for nid in **nid_list_**]

with ut.Timer('numpy3'): # alt method valid_aids = np.array(ibs.get_valid_aids()) valid_nids = np.array(ibs.db.get_all_col_rows(const.ANNOTATION_TABLE, NAME_ROWID)) aids_list4 = [valid_aids.take(np.flatnonzero(np.equal(valid_nids, nid))).tolist() for nid in **nid_list_**]

assert aids_list2 == aids_list3 assert aids_list3 == aids_list4 assert aids_list1 == aids_list2

valid_aids = ibs.get_valid_aids() %timeit ibs.db.get_all_col_rows('annotations', 'rowid') %timeit ibs.db.get_all_col_rows('annotations', 'name_rowid') %timeit ibs.get_annot_name_rowids(valid_aids, distinguish_unknowns=False) %timeit ibs.get_valid_aids() %timeit ibs.get_annot_name_rowids(ibs.get_valid_aids(), distinguish_unknowns=False) valid_nids1 = ibs.get_annot_name_rowids(valid_aids, distinguish_unknowns=False) valid_nids2 = ibs.db.get_all_col_rows('annotations', 'name_rowid') assert valid_nids1 == valid_nids2

ibs.db.fname ibs.db.fpath

import sqlite3

con = sqlite3.connect(ibs.db.fpath)

opstr = ''' SELECT annot_rowid, name_rowid FROM annotations WHERE name_rowid IN

(SELECT name_rowid FROM name) ORDER BY name_rowid ASC, annot_rowid ASC

'''

annot_rowid_list = con.execute(opstr).fetchall() aid_list = ut.get_list_column(annot_rowid_list, 0) nid_list = ut.get_list_column(annot_rowid_list, 1)

HACKY HACKY HACK

with ut.Timer('hackquery + group'): #nid_list = ibs.get_valid_nids()[10:15] nid_list = ibs.get_valid_nids() opstr = ''' SELECT annot_rowid, name_rowid FROM annotations WHERE name_rowid IN

(%s) ORDER BY name_rowid ASC, annot_rowid ASC

‘’’ % (’, ‘.join(map(str, nid_list))) pair_list = ibs.db.connection.execute(opstr).fetchall() aids = np.array(ut.get_list_column(pair_list, 0)) nids = np.array(ut.get_list_column(pair_list, 1)) unique_nids, groupx = vt.group_indices(nids) **grouped_aids_** = vt.apply_grouping(aids, groupx) grouped_aids = [arr.tolist() for arr in **grouped_aids_**]

SELECT name_rowid, COUNT(annot_rowid) AS number, GROUP_CONCAT(annot_rowid) AS aid_list

FROM annotations WHERE name_rowid in (SELECT name_rowid FROM name)

GROUP BY name_rowid

ORDER BY name_rowid ASC

import vtool as vt vt vt.aid_list[0]

annot_rowid_list = con.execute(opstr).fetchall() opstr = ‘’’

SELECT annot_rowid FROM annotations WHERE name_rowid=? ‘’’

cur = ibs.db.connection.cursor()

cur = con.execute(‘BEGIN IMMEDIATE TRANSACTION’) cur = ibs.db.connection res = [cur.execute(opstr, (nid,)).fetchall() for nid in **nid_list_**] cur.execute(‘COMMIT TRANSACTION’)

res = [ibs.db.cur.execute(opstr, (nid,)).fetchall() for nid in **nid_list_**]

wbia.control.manual_name_funcs.**get_name_alias_texts**(*ibs*, *name_rowid_list*)

Returns name_alias_text_list

Return type **list_** (list)

CommandLine: python -m wbia.control.manual_name_funcs –test-get_name_texts

CommandLine: python -m wbia.control.manual_name_funcs –test-get_name_alias_texts

RESTful: Method: GET URL: /api/name/alias/text/

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> name_rowid_list = ibs.get_valid_nids()
>>> # execute function
>>> name_alias_text_list = get_name_alias_texts(ibs, name_rowid_list)
>>> # verify results
>>> result = str(name_alias_text_list)
>>> print(result)
[None, None, None, None, None, None, None]
```

wbia.control.manual_name_funcs.**get_name_annot_uuids**(*ibs*, *nid_list*, ***kwargs*)

wbia.control.manual_name_funcs.**get_name_exemplar_aids**(*ibs*, *nid_list*)

Returns a list of list of cids in each name

Return type **list_** (list)

CommandLine: python -m wbia.control.manual_name_funcs –test-get_name_exemplar_aids

RESTful: Method: GET URL: /api/name/annot/rowid/examplar/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> nid_list = ibs.get_annot_name_rowids(aid_list)
>>> exemplar_aids_list = ibs.get_name_exemplar_aids(nid_list)
>>> result = [sorted(i) for i in exemplar_aids_list]
>>> print(result)
[[], [2, 3], [2, 3], [], [5, 6], [5, 6], [7], [8], [], [10], [], [12], [13]]
```

wbia.control.manual_name_funcs.**get_name_exemplar_name_uuids**(*ibs*, *nid_list*, ***kwargs*)

wbia.control.manual_name_funcs.**get_name_gids**(*ibs*, *nid_list*)

Returns the image ids associated with name ids

Return type list_ (list)

RESTful: Method: GET URL: /api/name/image/rowid/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> nid_list = ibs._get_all_known_name_rowids()
>>> gids_list = ibs.get_name_gids(nid_list)
>>> result = [sorted(gids) for gids in gids_list]
>>> print(result)
[[2, 3], [5, 6], [7], [8], [10], [12], [13]]
```

wbia.control.manual_name_funcs.**get_name_gps_tracks**(*ibs*, *nid_list=None*, *aid_list=None*)

CommandLine: python -m wbia.other.ibsfuncs –test-get_name_gps_tracks

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> import wbia
>>> # build test data
>>> #ibs = wbia.opendb('PZ_Master0')
>>> ibs = wbia.opendb('testdb1')
>>> #nid_list = ibs.get_valid_nids()
>>> aid_list = ibs.get_valid_aids()
```

(continues on next page)

(continued from previous page)

```
>>> nid_list, gps_track_list, aid_track_list = ibs.get_name_gps_tracks(aid_
↪list=aid_list)
>>> nonempty_list = list(map(lambda x: len(x) > 0, gps_track_list))
>>> ut.compress(nid_list, nonempty_list)
>>> ut.compress(gps_track_list, nonempty_list)
>>> ut.compress(aid_track_list, nonempty_list)
>>> aid_track_list = list(map(sorted, aid_track_list))
>>> result = str(aid_track_list)
>>> print(result)
[[11], [], [4], [1], [2, 3], [5, 6], [7], [8], [10], [12], [13]]
```

wbia.control.manual_name_funcs.**get_name_has_split**(*ibs*, *nid_list*)

CommandLine: python -m wbia.other.ibsfuncs –test-get_name_speeds

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> nid_list = ibs._get_all_known_nids()
>>> splits_list = ibs.get_name_has_split(nid_list)
>>> result = str(splits_list)
>>> print(result)
```

wbia.control.manual_name_funcs.**get_name_hourdiffs**(*ibs*, *nid_list*)

CommandLine: python -m wbia.other.ibsfuncs –test-get_name_hourdiffs

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> nid_list = ibs._get_all_known_nids()
>>> hourdiffs_list = ibs.get_name_hourdiffs(nid_list)
>>> result = hourdiffs_list
>>> print(hourdiffs_list)
```

wbia.control.manual_name_funcs.**get_name_image_uuids**(*ibs*, *nid_list*)

DEPRICATE

Returns the image ids associated with name ids

Return type list_ (list)

RESTful: Method: GET URL: /api/name/image/uuid/

wbia.control.manual_name_funcs.**get_name_imgset_uuids**(*ibs*, *nid_list*)

RESTful: Method: GET URL: /api/name/imageset/uuid/

wbia.control.manual_name_funcs.**get_name_imgsetids**(*ibs*, *nid_list*)

RESTful: Method: GET URL: /api/name/imageset/rowid/

wbia.control.manual_name_funcs.**get_name_max_hourdiff**(*ibs*, *nid_list*)

wbia.control.manual_name_funcs.**get_name_max_speed**(*ibs*, *nid_list*)

CommandLine: python -m wbia.other.ibsfuncs –test-get_name_max_speed

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> nid_list = ibs._get_all_known_nids()
>>> maxspeed_list = ibs.get_name_max_speed(nid_list)
>>> result = maxspeed_list
>>> print(maxspeed_list)
```

wbia.control.manual_name_funcs.**get_name_metadata**(*ibs*, *name_rowid_list*, *return_raw=False*)

Returns name metadata dictionary

Return type list_ (list)

RESTful: Method: GET URL: /api/name/metadata/

wbia.control.manual_name_funcs.**get_name_nids_with_gids**(*ibs*, *nid_list=None*)

wbia.control.manual_name_funcs.**get_name_notes**(*ibs*, *name_rowid_list*)

Returns notes_list - name notes

Return type list_ (list)

RESTful: Method: GET URL: /api/name/note/

wbia.control.manual_name_funcs.**get_name_num_annotations**(*ibs*, *nid_list*)

Returns the number of annotations for each name

Return type list_ (list)

CommandLine: python -m wbia.control.manual_name_funcs –test-get_name_num_annotations

RESTful: Method: GET URL: /api/name/num/annot/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> nid_list = ibs._get_all_known_name_rowids()
>>> result = get_name_num_annotations(ibs, nid_list)
>>> print(result)
[2, 2, 1, 1, 1, 1, 1]
```

wbia.control.manual_name_funcs.**get_name_num_exemplar_annotations**(*ibs*, *nid_list*)

Returns the number of annotations, which are exemplars for each name

Return type list_ (list)

RESTful: Method: GET URL: /api/name/num/annot/exemplar/

wbia.control.manual_name_funcs.**get_name_rowids_from_text**(*ibs*, *name_text_list*, *ensure=True*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **name_text_list** (*list*) –
- **ensure** (*bool*) – adds as new names if non-existant (default = True)

Returns Creates one if it doesnt exist

Return type name_rowid_list (list)

CommandLine: python -m wbia.control.manual_name_funcs –test-get_name_rowids_from_text:0 python -m wbia.control.manual_name_funcs –test-get_name_rowids_from_text:1

Todo: should ensure be defaulted to False?

RESTful: Method: GET URL: /api/name/rowid/text/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> import wbia
>>> import utool as ut
>>> ibs = wbia.opendb('testdb1')
>>> name_text_list = [u'Fred', u'Sue', '____', u'zebra_grevys', 'TYPO', '____']
>>> ensure = False
>>> name_rowid_list = ibs.get_name_rowids_from_text(name_text_list, ensure)
>>> print(ut.repr2(list(zip(name_text_list, name_rowid_list))))
>>> ensure = True
>>> name_rowid_list = ibs.get_name_rowids_from_text(name_text_list, ensure)
>>> print(ut.repr2(list(zip(name_text_list, name_rowid_list))))
>>> ibs.print_name_table()
>>> result = str(name_rowid_list) + '\n'
>>> typo_rowids = ibs.get_name_rowids_from_text(['TYPO', 'Fred', 'Sue', 'zebra_
↪grevys'])
>>> ibs.delete_names(typo_rowids)
>>> result += str(ibs._get_all_known_name_rowids())
>>> print('----')
>>> ibs.print_name_table()
>>> assert result == f'{name_rowid_list}\n[1, 2, 3, 4, 5, 6, 7]'
>>> print(result)
```

wbia.control.manual_name_funcs.**get_name_rowids_from_text_**(*ibs*, *name_text_list*, *ensure=True*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **name_text_list** (*list*) –

Returns

Return type name_rowid_list (list)

CommandLine: python -m wbia.control.manual_name_funcs –**test-get_name_rowids_from_text_**

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> import wbia
>>> import utool as ut  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> name_text_list = [u'Fred', 'easy', u'Sue', '____', u'zebra_grevys', 'TYPO',
↪'jeff']
>>> name_rowid_list = ibs.get_name_rowids_from_text_(name_text_list)
>>> ibs.print_name_table()
>>> result = str(name_rowid_list)
>>> print(result)
[None, 1, None, 0, None, None, 3]
```

wbia.control.manual_name_funcs.**get_name_rowids_from_uuid**(*ibs*, *uuid_list*, *nid_hack=False*, *ensure=True*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **name_text_list** (*list*) –

Returns

Return type name_rowid_list (list)

wbia.control.manual_name_funcs.**get_name_sex**(*ibs*, *name_rowid_list*, *eager=True*, *nInput=None*)

name_sex_list <- name.name_sex[name_rowid_list]

gets data from the “native” column “name_sex” in the “name” table

Parameters **name_rowid_list** (*list*) –

Returns name_sex_list

Return type list

TemplateInfo: Tgetter_table_column col = name_sex tbl = name

RESTful: Method: GET URL: /api/name/sex/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> name_rowid_list = ibs._get_all_name_rowids()
>>> eager = True
>>> name_sex_list = ibs.get_name_sex(name_rowid_list, eager=eager)
>>> assert len(name_rowid_list) == len(name_sex_list)
```

wbia.control.manual_name_funcs.**get_name_sex_text**(*ibs*, *name_rowid_list*, *eager=True*, *nInput=None*)

RESTful: Method: GET URL: /api/name/sex/text/

wbia.control.manual_name_funcs.**get_name_speeds**(*ibs*, *nid_list*)

CommandLine: python -m wbia.other.ibsfuncs –test-get_name_speeds

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> nid_list = ibs._get_all_known_nids()
>>> speeds_list = get_name_speeds(ibs, nid_list)
>>> result = str(speeds_list)
>>> print(result)
```

wbia.control.manual_name_funcs.**get_name_temp_flag**(*ibs*, *name_rowid_list*, *eager=True*, *nInput=None*)

name_temp_flag_list <- name.name_temp_flag[name_rowid_list]

gets data from the “native” column “name_temp_flag” in the “name” table

Parameters **name_rowid_list** (*list*) –

Returns name_temp_flag_list

Return type list

TemplateInfo: Tgetter_table_column col = name_temp_flag tbl = name

CommandLine: python -m wbia.control.manual_name_funcs –test-get_name_temp_flag

RESTful: Method: GET URL: /api/name/temp/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> name_rowid_list = ibs._get_all_name_rowids()
>>> eager = True
>>> name_temp_flag_list = ibs.get_name_temp_flag(name_rowid_list, eager=eager)
>>> assert len(name_rowid_list) == len(name_temp_flag_list)
```

wbia.control.manual_name_funcs.**get_name_texts**(*ibs*, *name_rowid_list*, *apply_fix=True*)

Returns text names

Return type list_ (list)

CommandLine: python -m wbia.control.manual_name_funcs –test-get_name_texts

RESTful: Method: GET URL: /api/name/text/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> name_rowid_list = ibs._get_all_known_name_rowids()
>>> name_text_list = get_name_texts(ibs, name_rowid_list)
>>> result = ut.repr2(name_text_list)
>>> print(result)
['easy', 'hard', 'jeff', 'lena', 'occl', 'polar', 'zebra']
```

wbia.control.manual_name_funcs.**get_name_uuids**(*ibs*, *nid_list*)

Returns uuids_list - name uuids

Return type list_ (list)

RESTful: Method: GET URL: /api/name/uuid/

wbia.control.manual_name_funcs.**get_num_names**(*ibs*, ***kwargs*)

Number of valid names

CommandLine: python -m wbia.control.manual_name_funcs –test-get_num_names

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> result = get_num_names(ibs)
>>> print(result)
7
```

wbia.control.manual_name_funcs.**get_valid_nids**(*ibs*, *imgsetid=None*, *filter_empty=False*, *min_pername=None*)

Returns all valid names with at least one animal (does not include unknown names)

Return type list_ (list)

RESTful: Method: GET URL: /api/name/

wbia.control.manual_name_funcs.**sanitize_name_texts**(*ibs*, *name_text_list*)

RESTful: Method: PUT URL: /api/name/sanitize

wbia.control.manual_name_funcs.**set_name_alias_texts**(*ibs*, *name_rowid_list*, *name_alias_text_list*)

Returns name_alias_text_list

Return type list_ (list)

CommandLine: python -m wbia.control.manual_name_funcs –test-get_name_texts

RESTful: Method: PUT URL: /api/name/alias/text/

wbia.control.manual_name_funcs.**set_name_metadata**(*ibs*, *name_rowid_list*, *metadata_dict_list*)

Sets the name's metadata using a metadata dictionary

RESTful: Method: PUT URL: /api/name/metadata/

wbia.control.manual_name_funcs.**set_name_notes**(*ibs*, *name_rowid_list*, *notes_list*)

Sets a note for each name (multiple annotations)

RESTful: Method: PUT URL: /api/name/note/

wbia.control.manual_name_funcs.**set_name_sex**(*ibs*, *name_rowid_list*, *name_sex_list*, *duplicate_behavior='error'*)

name_sex_list -> name.name_sex[name_rowid_list]

Parameters

- **name_rowid_list** –
- **name_sex_list** –

TemplateInfo: Tsetter_native_column tbl = name col = name_sex

RESTful: Method: PUT URL: /api/name/sex/

wbia.control.manual_name_funcs.**set_name_sex_text**(*ibs*, *name_rowid_list*, *name_sex_text_list*)

RESTful: Method: PUT URL: /api/name/sex/text/

wbia.control.manual_name_funcs.**set_name_temp_flag**(*ibs*, *name_rowid_list*, *name_temp_flag_list*, *duplicate_behavior='error'*)

name_temp_flag_list -> name.name_temp_flag[name_rowid_list]

Parameters

- **name_rowid_list** –
- **name_temp_flag_list** –

TemplateInfo: Tsetter_native_column tbl = name col = name_temp_flag

RESTful: Method: PUT URL: /api/name/temp/

wbia.control.manual_name_funcs.**set_name_texts**(*ibs*, *name_rowid_list*, *name_text_list*, *verbose=False*, *notify_wildbook=False*, *assert_wildbook=False*, *update_json_log=True*)

Changes the name text. Does not affect the animals of this name. Effectively just changes the TEXT UUID

CommandLine: python -m wbia.control.manual_name_funcs –test-set_name_texts

RESTful: Method: PUT URL: /api/name/text/

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_name_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> nid_list = ibs.get_valid_nids()[0:2]
>>> name_list = ibs.get_name_texts(nid_list)
>>> result = set_name_texts(ibs, nid_list, name_list)
>>> print(result)
```

wbia.control.manual_name_funcs.**testdata_ibs**(*defaultdb='testdb1'*)

1.2.28 wbia.control.manual_part_funcs module

Autogen: python -c "import utool as ut; ut.write_modscript_alias('Tgen.sh', 'wbia.templates.template_generator')" # NOQA sh Tgen.sh –key part –invert –Tcfg with_getters=True with_setters=True –modfname manual_part_funcs –funcname-filter=age_m # NOQA sh Tgen.sh –key part –invert –Tcfg with_getters=True with_setters=True –modfname manual_part_funcs –funcname-filter=is_ # NOQA sh Tgen.sh –key part –invert –Tcfg with_getters=True with_setters=True –modfname manual_part_funcs –funcname-filter=is_ –diff # NOQA

wbia.control.manual_part_funcs.**add_parts**(*ibs*, *aid_list*, *bbox_list=None*, *theta_list=None*, *detect_confidence_list=None*, *notes_list=None*, *vert_list=None*, *part_uuid_list=None*, *viewpoint_list=None*, *quality_list=None*, *type_list=None*, *staged_uuid_list=None*, *staged_user_id_list=None*, ***kwargs*)

Adds an part to annotations

Parameters

- **aid_list** (*list*) – annotation rowids to add part to
- **bbox_list** (*list*) – of [x, y, w, h] bounding boxes for each annotation (supply verts instead)
- **theta_list** (*list*) – orientations of parts
- **vert_list** (*list*) – alternative to bounding box

Returns part_rowid_list

Return type list

Ignore: detect_confidence_list = None notes_list = None part_uuid_list = None viewpoint_list = None quality_list = None type_list = None

RESTful: Method: POST URL: /api/part/

wbia.control.manual_part_funcs.**delete_parts**(*ibs*, *part_rowid_list*)

deletes parts from the database

RESTful: Method: DELETE URL: /api/part/

Parameters

- **ibs** (IBEISController) – wbia controller object
- **part_rowid_list** (*int*) – list of part ids

wbia.control.manual_part_funcs.**filter_part_set**(*ibs*, *part_rowid_list*, *include_only_aid_list=None*, *is_staged=False*, *viewpoint='no-filter'*, *minqual=None*)

wbia.control.manual_part_funcs.**get_num_parts**(*ibs*, ***kwargs*)

Number of valid parts

wbia.control.manual_part_funcs.**get_part_aids**(*ibs*, *part_rowid_list*, *assume_unique=False*)

Get parent annotation rowids of parts

Parameters **part_rowid_list** (*list*) –

Returns annot rowids

Return type aid_list (list)

RESTful: Method: GET URL: /api/part/annot/rowid/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_part_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> part_rowid_list = ibs.get_valid_part_rowids()
>>> result = get_part_aids(ibs, part_rowid_list)
>>> print(result)
```

wbia.control.manual_part_funcs.**get_part_annot_rowids**(*ibs*, *part_rowid_list*)

wbia.control.manual_part_funcs.**get_part_annot_uuids**(*ibs*, *part_rowid_list*)

wbia.control.manual_part_funcs.**get_part_bboxes**(*ibs*, *part_rowid_list*)

Returns part bounding boxes in image space

Return type bbox_list (list)

RESTful: Method: GET URL: /api/part/bbox/

wbia.control.manual_part_funcs.**get_part_contour**(*ibs*, *part_rowid_list*, *return_raw=False*)

Returns part contour dictionary

Return type list_ (list)

RESTful: Method: GET URL: /api/part/contour/

wbia.control.manual_part_funcs.**get_part_detect_confidence**(*ibs*, *part_rowid_list*)

Returns a list confidences that the parts is a valid detection

Return type list_ (list)

RESTful: Method: GET URL: /api/part/detect/confidence/

wbia.control.manual_part_funcs.**get_part_gids**(*ibs*, *part_rowid_list*, *assume_unique=False*)

Get parent imageation rowids of parts

Parameters **part_rowid_list** (*list*) –

Returns image rowids

Return type gid_list (list)

RESTful: Method: GET URL: /api/part/image/rowid/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_part_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> part_rowid_list = ibs.get_valid_part_rowids()
>>> result = get_part_gids(ibs, part_rowid_list)
>>> print(result)
```

wbia.control.manual_part_funcs.**get_part_image_rowids**(*ibs*, *part_rowid_list*)

wbia.control.manual_part_funcs.**get_part_image_uuids**(*ibs*, *part_rowid_list*)

wbia.control.manual_part_funcs.**get_part_isjunk**(*ibs*, *part_rowid_list*)

Auto-docstr for 'get_part_isjunk'

wbia.control.manual_part_funcs.**get_part_metadata**(*ibs*, *part_rowid_list*, *return_raw=False*)

Returns part metadata dictionary

Return type list_ (list)

RESTful: Method: GET URL: /api/part/metadata/

wbia.control.manual_part_funcs.**get_part_missing_uuid**(*ibs*, *uuid_list*)

Returns a list of missing part uuids

Return type list_ (list)

wbia.control.manual_part_funcs.**get_part_notes**(*ibs*, *part_rowid_list*)

Returns a list of part notes

Return type part_notes_list (list)

RESTful: Method: GET URL: /api/part/note/

wbia.control.manual_part_funcs.**get_part_num_verts**(*ibs*, *part_rowid_list*)

Returns the number of vertices that form the polygon of each part

Return type nVerts_list (list)

RESTful: Method: GET URL: /api/part/num/vert/

wbia.control.manual_part_funcs.**get_part_qualities**(*ibs*, *part_rowid_list*, *eager=True*)
part_quality_list <- part.part_quality[part_rowid_list]

gets data from the "native" column "part_quality" in the "part" table

Parameters **part_rowid_list** (*list*) –

Returns part_quality_list

Return type list

TemplateInfo: Tgetter_table_column col = part_quality tbl = part

SeeAlso: wbia.const.QUALITY_INT_TO_TEXT

RESTful: Method: GET URL: /api/part/quality/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_part_funcs import *  # NOQA
>>> ibs, qreq_ = testdata_ibs()
>>> part_rowid_list = ibs._get_all_part_rowids()
>>> eager = True
>>> part_quality_list = ibs.get_part_qualities(part_rowid_list, eager=eager)
>>> print('part_quality_list = %r' % (part_quality_list,))
>>> assert len(part_rowid_list) == len(part_quality_list)
```

wbia.control.manual_part_funcs.**get_part_quality_texts**(*ibs*, *part_rowid_list*)
Auto-docstr for 'get_part_quality_texts'

RESTful: Method: GET URL: /api/part/quality/text/

wbia.control.manual_part_funcs.**get_part_reviewed**(*ibs*, *part_rowid_list*)

Returns "All Instances Found" flag, true if all objects of interest

Return type list_ (list)

(animals) have an PART in the part

RESTful: Method: GET URL: /api/part/reviewed/

wbia.control.manual_part_funcs.**get_part_rotated_verts**(*ibs*, *part_rowid_list*)

Returns verticies after rotation by theta.

Return type rotated_vert_list (list)

RESTful: Method: GET URL: /api/part/vert/rotated/

wbia.control.manual_part_funcs.**get_part_rowids_from_uuid**(*ibs*, *uuid_list*)

Returns part rowids

Return type list_ (list)

RESTful: Method: GET URL: /api/part/rowid/uuid/

wbia.control.manual_part_funcs.**get_part_rows**(*ibs*, *part_rowid_list*)
Auto-docstr for 'get_part_rows'

wbia.control.manual_part_funcs.**get_part_staged_flags**(*ibs*, *part_rowid_list*)
returns if an part is staged

Parameters

- **ibs** (IBEISController) – wbia controller object
- **part_rowid_list** (*int*) – list of part ids

Returns part_staged_flag_list - True if part is staged

Return type list

CommandLine: python -m wbia.control.manual_part_funcs –test-get_part_staged_flags

RESTful: Method: GET URL: /api/part/staged/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_part_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> part_rowid_list = ibs.get_valid_part_rowids()
>>> gid_list = get_part_staged_flags(ibs, part_rowid_list)
>>> result = str(gid_list)
>>> print(result)
```

wbia.control.manual_part_funcs.**get_part_staged_metadata**(*ibs*, *part_rowid_list*, *return_raw=False*)

Returns part metadata dictionary

Return type list_ (list)

RESTful: Method: GET URL: /api/part/staged/metadata/

wbia.control.manual_part_funcs.**get_part_staged_user_ids**(*ibs*, *part_rowid_list*)
returns if an part is staged

Parameters

- **ibs** (IBEISController) – wbia controller object
- **part_rowid_list** (*int*) – list of part ids

Returns part_staged_user_id_list - True if part is staged

Return type list

CommandLine: python -m wbia.control.manual_part_funcs –test-get_part_staged_user_ids

RESTful: Method: GET URL: /api/part/staged/user/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_part_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> part_rowid_list = ibs.get_valid_part_rowids()
>>> gid_list = get_part_staged_user_ids(ibs, part_rowid_list)
>>> result = str(gid_list)
>>> print(result)
```

wbia.control.manual_part_funcs.**get_part_staged_uuids**(*ibs*, *aid_list*)

Returns part_uuid_list a list of image uuids by aid

Return type list

RESTful: Method: GET URL: /api/part/staged/uuid/

wbia.control.manual_part_funcs.**get_part_tag_text**(*ibs*, *part_rowid_list*, ***kwargs*)

part_tags_list <- part.part_tags[part_rowid_list]

gets data from the "native" column "part_tags" in the "part" table

Parameters **part_rowid_list** (*list*) –

Returns part_tags_list

Return type list

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_part_funcs import *  # NOQA
>>> ibs, config2_ = testdata_ibs()
>>> part_rowid_list = ibs._get_all_part_rowids()
>>> eager = True
>>> part_tags_list = ibs.get_part_tag_text(part_rowid_list, eager=eager)
>>> assert len(part_rowid_list) == len(part_tags_list)
```

wbia.control.manual_part_funcs.**get_part_thetas**(*ibs*, *part_rowid_list*)

Returns a list of floats describing the angles of each part

Return type theta_list (list)

CommandLine: python -m wbia.control.manual_part_funcs –test-get_part_thetas

RESTful: Method: GET URL: /api/part/theta/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_part_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('NAUT_test')
>>> part_rowid_list = ibs.get_valid_part_rowids()
>>> result = get_part_thetas(ibs, part_rowid_list)
```

(continues on next page)

(continued from previous page)

```
>>> print(result)
[]
```

wbia.control.manual_part_funcs.**get_part_types**(*ibs*, *part_rowid_list*)

Returns a list of part notes

Return type part_notes_list (list)

RESTful: Method: GET URL: /api/part/note/

wbia.control.manual_part_funcs.**get_part_uuids**(*ibs*, *part_rowid_list*)

Returns part_uuid_list a list of part uuids by part_rowid

Return type list

RESTful: Method: GET URL: /api/part/uuid/

wbia.control.manual_part_funcs.**get_part_verts**(*ibs*, *part_rowid_list*)

Returns the vertices that form the polygon of each part

Return type vert_list (list)

RESTful: Method: GET URL: /api/part/vert/

wbia.control.manual_part_funcs.**get_part_viewpoints**(*ibs*, *part_rowid_list*)

Returns a list of part notes

Return type part_notes_list (list)

RESTful: Method: GET URL: /api/part/note/

wbia.control.manual_part_funcs.**get_valid_part_rowids**(*ibs*, *include_only_aid_list=None*, *is_staged=False*, *viewpoint='no-filter'*, *minqual=None*)

wbia.control.manual_part_funcs.**get_valid_part_uuids**(*ibs*)

Returns part_uuid_list a list of part uuids for all valid part_rowids

Return type list

wbia.control.manual_part_funcs.**part_src_api**(*rowid=None*)
Returns the base64 encoded image of part <rowid>

RESTful: Method: GET URL: /api/part/<rowid>/

wbia.control.manual_part_funcs.**set_part_bboxes**(*ibs*, *part_rowid_list*, *bbox_list*)
Sets bboxes of a list of parts by part_rowid,

Parameters

- **part_rowid_list** (*list of rowids*) – list of part rowids
- **bbox_list** (*list of (x, y, w, h)*) – new bounding boxes for each part_rowid

Note: set_part_bboxes is a proxy for set_part_verts

RESTful: Method: PUT URL: /api/part/bbox/

wbia.control.manual_part_funcs.**set_part_contour**(*ibs*, *part_rowid_list*, *contour_dict_list*)

Sets the part's contour using a contour dictionary

RESTful: Method: PUT URL: /api/part/contour/

CommandLine: python -m wbia.control.manual_part_funcs –test-set_part_contour

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_part_funcs import *  # NOQA
>>> import wbia
>>> import random
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[0:1]
>>> bbox_list = [[0, 0, 100, 100]] * len(aid_list)
>>> part_rowid_list = ibs.add_parts(aid_list, bbox_list=bbox_list)
>>> contour_dict_list = [
>>>     {'test': random.uniform(0.0, 1.0)},
>>> ]
>>> print(ut.repr2(contour_dict_list))
>>> ibs.set_part_contour(part_rowid_list, contour_dict_list)
>>> # verify results
>>> contour_dict_list_ = ibs.get_part_contour(part_rowid_list)
>>> print(ut.repr2(contour_dict_list_))
>>> assert contour_dict_list == contour_dict_list_
>>> contour_str_list = [ut.to_json(contour_dict) for contour_dict in contour_dict_
↪list]
>>> print(ut.repr2(contour_str_list))
>>> contour_str_list_ = ibs.get_part_contour(part_rowid_list, return_raw=True)
>>> print(ut.repr2(contour_str_list_))
>>> assert contour_str_list == contour_str_list_
>>> ibs.delete_parts(part_rowid_list)
```

wbia.control.manual_part_funcs.**set_part_detect_confidence**(*ibs*, *part_rowid_list*, *confidence_list*)

Sets part notes

RESTful: Method: PUT URL: /api/part/detect/confidence/

wbia.control.manual_part_funcs.**set_part_metadata**(*ibs*, *part_rowid_list*, *metadata_dict_list*)

Sets the part's metadata using a metadata dictionary

RESTful: Method: PUT URL: /api/part/metadata/

CommandLine: python -m wbia.control.manual_part_funcs –test-set_part_metadata

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_part_funcs import *  # NOQA
>>> import wbia
>>> import random
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[0:1]
>>> bbox_list = [[0, 0, 100, 100]] * len(aid_list)
>>> part_rowid_list = ibs.add_parts(aid_list, bbox_list=bbox_list)
>>> metadata_dict_list = [
>>>     {'test': random.uniform(0.0, 1.0)},
>>> ]
>>> print(ut.repr2(metadata_dict_list))
>>> ibs.set_part_metadata(part_rowid_list, metadata_dict_list)
>>> # verify results
>>> metadata_dict_list_ = ibs.get_part_metadata(part_rowid_list)
>>> print(ut.repr2(metadata_dict_list_))
>>> assert metadata_dict_list == metadata_dict_list_
>>> metadata_str_list = [ut.to_json(metadata_dict) for metadata_dict in metadata_
↪dict_list]
>>> print(ut.repr2(metadata_str_list))
>>> metadata_str_list_ = ibs.get_part_metadata(part_rowid_list, return_raw=True)
>>> print(ut.repr2(metadata_str_list_))
>>> assert metadata_str_list == metadata_str_list_
>>> ibs.delete_parts(part_rowid_list)
```

wbia.control.manual_part_funcs.**set_part_notes**(*ibs*, *part_rowid_list*, *notes_list*)

Sets part notes

RESTful: Method: PUT URL: /api/part/note/

wbia.control.manual_part_funcs.**set_part_qualities**(*ibs*, *part_rowid_list*, *part_quality_list*)

part_quality_list -> part.part_quality[part_rowid_list]

A quality is an integer representing the following types:

Parameters

- **part_rowid_list** –
- **part_quality_list** –

SeeAlso: wbia.const.QUALITY_INT_TO_TEXT

RESTful: Method: PUT URL: /api/part/quality/

wbia.control.manual_part_funcs.**set_part_quality_texts**(*ibs*, *part_rowid_list*, *quality_text_list*)

Auto-docstr for 'set_part_quality_texts'

RESTful: Method: PUT URL: /api/part/quality/text/

wbia.control.manual_part_funcs.**set_part_reviewed**(*ibs*, *part_rowid_list*, *reviewed_list*)

Sets the part all instances found bit

RESTful: Method: PUT URL: /api/part/reviewed/

wbia.control.manual_part_funcs.**set_part_staged_metadata**(*ibs*, *part_rowid_list*, *metadata_dict_list*)

Sets the part's staged metadata using a metadata dictionary

RESTful: Method: PUT URL: /api/part/staged/metadata/

CommandLine: python -m wbia.control.manual_part_funcs –test-set_part_staged_metadata

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_part_funcs import *  # NOQA
>>> import wbia
>>> import random
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[0:1]
>>> bbox_list = [[0, 0, 100, 100]] * len(aid_list)
>>> part_rowid_list = ibs.add_parts(aid_list, bbox_list=bbox_list)
>>> metadata_dict_list = [
>>>     {'test': random.uniform(0.0, 1.0)},
>>> ] * len(part_rowid_list)
>>> print(ut.repr2(metadata_dict_list))
>>> ibs.set_part_staged_metadata(part_rowid_list, metadata_dict_list)
>>> # verify results
>>> metadata_dict_list_ = ibs.get_part_staged_metadata(part_rowid_list)
>>> print(ut.repr2(metadata_dict_list_))
>>> assert metadata_dict_list == metadata_dict_list_
>>> metadata_str_list = [ut.to_json(metadata_dict) for metadata_dict in metadata_
↪dict_list]
>>> print(ut.repr2(metadata_str_list))
>>> metadata_str_list_ = ibs.get_part_staged_metadata(part_rowid_list, return_
↪raw=True)
>>> print(ut.repr2(metadata_str_list_))
>>> assert metadata_str_list == metadata_str_list_
>>> ibs.delete_parts(part_rowid_list)
```

wbia.control.manual_part_funcs.**set_part_staged_user_ids**(*ibs*, *part_rowid_list*, *user_id_list*)

Sets the staged part user id

RESTful: Method: PUT URL: /api/part/staged/user/

wbia.control.manual_part_funcs.**set_part_staged_uuids**(*ibs*, *aid_list*, *part_uuid_list*)

Returns all nids of known animals (does not include unknown names)

Return type list_ (list)

wbia.control.manual_part_funcs.**set_part_tag_text**(*ibs*, *part_rowid_list*, *part_tags_list*, *duplicate_behavior='error'*)

part_tags_list -> part.part_tags[part_rowid_list]

Parameters

- **part_rowid_list** –
- **part_tags_list** –

wbia.control.manual_part_funcs.**set_part_thetas**(*ibs*, *part_rowid_list*, *theta_list*)

Sets thetas of a list of part_rowid_list

RESTful: Method: PUT URL: /api/part/theta/

wbia.control.manual_part_funcs.**set_part_types**(*ibs*, *part_rowid_list*, *type_list*)
Sets part notes

RESTful: Method: PUT URL: /api/part/note/

wbia.control.manual_part_funcs.**set_part_verts**(*ibs*, *part_rowid_list*, *verts_list*, *delete_thumbs=True*, *notify_root=True*)
Sets the vertices [(x, y), ...] of a list of part_rowid_list

RESTful: Method: PUT URL: /api/part/vert/

wbia.control.manual_part_funcs.**set_part_viewpoints**(*ibs*, *part_rowid_list*, *viewpoint_list*)
Sets part notes

RESTful: Method: PUT URL: /api/part/note/

wbia.control.manual_part_funcs.**testdata_ibs**()
Auto-docstr for 'testdata_ibs'

wbia.control.manual_part_funcs.**update_part_rotate_90**(*ibs*, *part_rowid_list*, *direction*)

wbia.control.manual_part_funcs.**update_part_rotate_left_90**(*ibs*, *part_rowid_list*)

wbia.control.manual_part_funcs.**update_part_rotate_right_90**(*ibs*, *part_rowid_list*)

1.2.29 wbia.control.manual_review_funcs module

python -c "import utool as ut; ut.write_modscript_alias('Tgen.sh', 'wbia.templates.template_generator')" sh Tgen.sh –key review –invert –Tcfg with_getters=True with_setters=False –modfname manual_review_funcs

TODO: Fix this name it is too special case

wbia.control.manual_review_funcs.**add_review**(*ibs*, *aid_1_list*, *aid_2_list*, *evidence_decision_list*, *meta_decision_list=None*, *review_uuid_list=None*, *identity_list=None*, *user_confidence_list=None*, *tags_list=None*, *review_client_start_time_posix=None*, *review_client_end_time_posix=None*, *review_server_start_time_posix=None*, *review_server_end_time_posix=None*)
Adds a list of reviews.

Returns review_id_list - review rowids

Return type list

RESTful: Method: POST URL: /api/review/

CommandLine: python -m wbia.control.manual_review_funcs –test-add_review

Doctest:

```
>>> import wbia
>>> from wbia.control.manual_review_funcs import *
>>> ibs = wbia.opendb('testdb1')
>>> ibs.staging.get_table_as_pandas('reviews')
>>> # ensure it is empty
```

(continues on next page)

(continued from previous page)

```
>>> rowids = ibs.staging.get_all_rowids('reviews')
>>> ibs.staging.delete_rowids('reviews', rowids)
>>> ut.exec_funckw(ibs.add_review, globals())
>>> # Add some dummy reviews
>>> aid_1_list = [1, 2, 3, 2]
>>> aid_2_list = [2, 3, 4, 3]
>>> evidence_decision_list = [1, 0, 1, 2]
>>> new_rowids = ibs.add_review(aid_1_list, aid_2_list,
>>>                             evidence_decision_list)
>>> assert new_rowids == [1, 2, 3, 4]
>>> table = ibs.staging.get_table_as_pandas('reviews')
>>> print(table)
>>> # Then delete them
>>> ibs.staging.delete_rowids('reviews', new_rowids)
```

wbia.control.manual_review_funcs.**delete_review**(*ibs*, *review_rowid_list*)
deletes reviews from the database

RESTful: Method: DELETE URL: /api/review/

wbia.control.manual_review_funcs.**e_**(*u*, *v*)

wbia.control.manual_review_funcs.**get_review_aid_tuple**(*ibs*, *review_rowid_list*, *eager=True*, *nInput=None*)

wbia.control.manual_review_funcs.**get_review_count**(*ibs*, *review_rowid_list*)

wbia.control.manual_review_funcs.**get_review_counts_from_pairs**(*ibs*, *aid_pairs*, *eager=True*, *nInput=None*)

Returns review_counts_list - review counts

Return type list_ (list)

RESTful: Method: GET URL: /api/review/counts/tuple/

wbia.control.manual_review_funcs.**get_review_counts_from_tuple**(*ibs*, *aid_1_list*, *aid_2_list*, *eager=True*, *nInput=None*)

Returns review_counts_list - review counts

Return type list_ (list)

RESTful: Method: GET URL: /api/review/counts/tuple/

wbia.control.manual_review_funcs.**get_review_decision**(*ibs*, *review_rowid_list*)

wbia.control.manual_review_funcs.**get_review_decision_str**(*ibs*, *review_rowid_list*)

wbia.control.manual_review_funcs.**get_review_decisions_from_only**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)

Returns review_tuple_decisions_list - review decisions

Return type list_ (list)

RESTful: Method: GET URL: /api/review/decisions/only/

wbia.control.manual_review_funcs.**get_review_exists_from_edges**(*ibs*, *edges*, *eager=True*, *nInput=None*)

wbia.control.manual_review_funcs.**get_review_identities_from_tuple**(*ibs*, *aid_1_list*, *aid_2_list*, *eager=True*, *nInput=None*)

Returns review_identities_list - review identities

Return type list_ (list)

RESTful: Method: GET URL: /api/review/identities/tuple/

wbia.control.manual_review_funcs.**get_review_identity**(*ibs*, *review_rowid_list*)

wbia.control.manual_review_funcs.**get_review_metadata**(*ibs*, *review_rowid_list*, *return_raw=False*)

Returns review metadata dictionary

Return type list_ (list)

RESTful: Method: GET URL: /api/review/metadata/

wbia.control.manual_review_funcs.**get_review_posix_client_end_time**(*ibs*, *review_rowid_list*)

wbia.control.manual_review_funcs.**get_review_posix_client_start_time**(*ibs*, *review_rowid_list*)

wbia.control.manual_review_funcs.**get_review_posix_server_end_time**(*ibs*, *review_rowid_list*)

wbia.control.manual_review_funcs.**get_review_posix_server_start_time**(*ibs*, *review_rowid_list*)

wbia.control.manual_review_funcs.**get_review_posix_time**(*ibs*, *review_rowid_list*)

wbia.control.manual_review_funcs.**get_review_posix_times_from_tuple**(*ibs*, *aid_1_list*, *aid_2_list*, *eager=True*, *nInput=None*)

Returns identity_list - review posix times

Return type list_ (list)

RESTful: Method: GET URL: /api/review/time/posix/tuple/

wbia.control.manual_review_funcs.**get_review_rowid_from_superkey**(*ibs*, *aid_1_list*, *aid_2_list*, *count_list*, *eager=False*, *nInput=None*)

Returns review_rowid_list

Parameters **lists** (*superkey*) – review_rowid_list, aid_list

Returns review_rowid_list

wbia.control.manual_review_funcs.**get_review_rowids_between**(*ibs*, *aids1*, *aids2=None*, *method=1*)

Find staging rowids between sets of aids

Doctest:

```
>>> from wbia.control.manual_review_funcs import *
>>> import wbia
>>> ibs = wbia.opendb('PZ_MTEST')
>>> aids1 = aids2 = [1, 2, 3, 4, 5, 6]
>>> rowids_between = ibs.get_review_rowids_between
>>> ids1 = sorted(rowids_between(aids1, aids2, method=1))
>>> ids2 = sorted(rowids_between(aids1, aids2, method=2))
>>> assert len(ub.find_duplicates(ids1)) == 0
>>> assert len(ub.find_duplicates(ids2)) == 0
>>> assert ids1 == ids2
```

wbia.control.manual_review_funcs.**get_review_rowids_from_aid1**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)

wbia.control.manual_review_funcs.**get_review_rowids_from_aid2**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)

wbia.control.manual_review_funcs.**get_review_rowids_from_aid_tuple**(*ibs*, *aid_1_list*, *aid_2_list*, *eager=True*, *nInput=None*)

Aid pairs are undirected

Returns review_rowid_list - review rowid list of lists

Return type list_ (list)

RESTful: Method: GET URL: /api/review/rowid/tuple/

wbia.control.manual_review_funcs.**get_review_rowids_from_edges**(*ibs*, *edges*, *eager=True*, *nInput=None*, *directed=False*)

wbia.control.manual_review_funcs.**get_review_rowids_from_only**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)

Returns review_rowids

Return type list_ (list)

RESTful: Method: GET URL: /api/review/rowids/only/

wbia.control.manual_review_funcs.**get_review_rowids_from_single**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)

wbia.control.manual_review_funcs.**get_review_tags**(*ibs*, *review_rowid_list*)

wbia.control.manual_review_funcs.**get_review_tags_from_tuple**(*ibs*, *aid_1_list*, *aid_2_list*, *eager=True*, *nInput=None*)

Returns review_tags_list - review tags (list of strings)

Return type list_ (list)

RESTful: Method: GET URL: /api/review/tags/tuple/

wbia.control.manual_review_funcs.**get_review_user_confidence**(*ibs*, *review_rowid_list*)

wbia.control.manual_review_funcs.**get_review_uuid**(*ibs*, *review_rowid_list*)

wbia.control.manual_review_funcs.**hack_create_aidpair_index**(*ibs*)

wbia.control.manual_review_funcs.**set_review_metadata**(*ibs*, *review_rowid_list*, *metadata_dict_list*)

Sets the review's metadata using a metadata dictionary

RESTful: Method: PUT URL: /api/review/metadata/

CommandLine: python -m wbia.control.manual_review_funcs –test-set_review_metadata

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_review_funcs import *  # NOQA
>>> import wbia
>>> import random
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> review_rowid_list = ibs.add_review([1], [2], [0])
>>> metadata_dict_list = [
>>>     {'test': random.uniform(0.0, 1.0)},
>>> ]
>>> print(ut.repr2(metadata_dict_list))
>>> ibs.set_review_metadata(review_rowid_list, metadata_dict_list)
>>> # verify results
>>> metadata_dict_list_ = ibs.get_review_metadata(review_rowid_list)
>>> print(ut.repr2(metadata_dict_list_))
>>> assert metadata_dict_list == metadata_dict_list_
>>> metadata_str_list = [ut.to_json(metadata_dict) for metadata_dict in metadata_
↪dict_list]
>>> print(ut.repr2(metadata_str_list))
>>> metadata_str_list_ = ibs.get_review_metadata(review_rowid_list, return_
↪raw=True)
```

(continues on next page)

(continued from previous page)

```
>>> print(ut.repr2(metadata_str_list_))
>>> assert metadata_str_list == metadata_str_list_
>>> ibs.delete_review(review_rowid_list)
```

1.2.30 wbia.control.manual_species_funcs module

python -c "import utool as ut; ut.write_modscript_alias('Tgen.sh', 'wbia.templates.template_generator')" sh Tgen.sh –key species –invert –Tcfg with_getters=True with_setters=False –modfname manual_species_funcs

TODO: Fix this name it is too special case

wbia.control.manual_species_funcs.**add_species**(*ibs*, *species_nice_list*, *species_text_list=None*, *species_code_list=None*, *species_uuid_list=None*, *species_note_list=None*, *skip_cleaning=False*)

Adds a list of species.

Returns speciesid_list - species rowids

Return type list

RESTful: Method: POST URL: /api/species/

CommandLine: python -m wbia.control.manual_species_funcs –test-add_species

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_species_funcs import *  # NOQA
>>> import wbia
>>> import utool as ut
>>> ibs = wbia.opendb('testdb1')
>>> species_text_list = [
...     'jaguar', 'zebra_plains', 'zebra_plains', '____', 'TYPO',
...     '____', 'zebra_grevys', 'bear_polar+head']
>>> species_rowid_list = ibs.add_species(species_text_list)
>>> print(ut.repr2(list(zip(species_text_list, species_rowid_list))))
>>> ibs.print_species_table()
>>> species_text = ibs.get_species_texts(species_rowid_list)
>>> # Ensure we leave testdb1 in a clean state
>>> ibs.delete_species(ibs.get_species_rowids_from_text(['jaguar', 'TYPO']))
>>> all_species_rowids = ibs._get_all_species_rowids()
>>> result =  ut.repr2(species_text, nl=False) + '\n'
>>> result += ut.repr2(all_species_rowids, nl=False) + '\n'
>>> result += ut.repr2(ibs.get_species_texts(all_species_rowids), nl=False) + '\n'
>>> result += ut.repr2(ibs.get_species_codes(all_species_rowids), nl=False)
>>> print(result)
['jaguar', 'zebra_plains', 'zebra_plains', '____', 'typo', '____', 'zebra_grevys',
↪ 'bear_polar+head']
[1, 2, 3, 6]
['zebra_plains', 'zebra_grevys', 'bear_polar', 'bear_polar+head']
['PZ', 'GZ', 'PB', 'BP+H']
```

wbia.control.manual_species_funcs.**delete_empty_species**(*ibs*)
deletes empty species from the database

wbia.control.manual_species_funcs.**delete_species**(*ibs*, *species_rowid_list*)
deletes species from the database

CAREFUL. YOU PROBABLY DO NOT WANT TO USE THIS at least ensure that no annot is associated with any of these species rowids

RESTful: Method: DELETE URL: /api/species/

wbia.control.manual_species_funcs.**get_all_species_nice**(*ibs*)

Returns all nids of known animals (does not include unknown names)

Return type list_ (list)

wbia.control.manual_species_funcs.**get_all_species_texts**(*ibs*)

Returns all nids of known animals (does not include unknown names)

Return type list_ (list)

wbia.control.manual_species_funcs.**get_species_codes**(*ibs*, *species_rowid_list*)

Returns code_list - species codes

Return type list_ (list)

RESTful: Method: GET URL: /api/species/code/

wbia.control.manual_species_funcs.**get_species_enabled**(*ibs*, *species_rowid_list*)

Returns "Species Enabled" flag, true if the species is enabled

Return type list_ (list)

wbia.control.manual_species_funcs.**get_species_nice**(*ibs*, *species_rowid_list*)

Returns species_text_list nice names

Return type list

CommandLine: python -m wbia.control.manual_species_funcs –test-get_species_nice –enableall

RESTful: Method: GET URL: /api/species/nice/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_species_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> ibs._clean_species()
>>> species_rowid_list = ibs._get_all_species_rowids()
>>> result = get_species_nice(ibs, species_rowid_list)
>>> result = ut.repr2(result)
>>> print(result)
['Zebra (Plains)', "Zebra (Grevy's)", 'Polar Bear', 'bear_polar+head']
```

wbia.control.manual_species_funcs.**get_species_notes**(*ibs*, *species_rowid_list*)

Returns notes_list - species notes

Return type list_ (list)

RESTful: Method: GET URL: /api/species/note/

wbia.control.manual_species_funcs.**get_species_rowids_from_text**(*ibs*, *species_text_list*, *ensure=True*, ***kwargs*)

Returns Creates one if it doesnt exist

Return type species_rowid_list (list)

CommandLine: python -m wbia.control.manual_species_funcs –test-get_species_rowids_from_text:0 python -m wbia.control.manual_species_funcs –test-get_species_rowids_from_text:1

RESTful: Method: GET URL: /api/species/rowid/text/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_species_funcs import *  # NOQA
>>> import wbia
>>> import utool as ut
>>> ibs = wbia.opendb('testdb1')
>>> species_text_list = [
...     u'jaguar', u'zebra_plains', u'zebra_plains', '____', 'TYPO',
...     '____', u'zebra_grevys', u'bear_polar']
>>> ensure = False
>>> species_rowid_list = ibs.get_species_rowids_from_text(species_text_list,␣
↪ensure)
>>> # print(ut.repr2(list(zip(species_text_list, species_rowid_list))))
>>> ensure = True
>>> species_rowid_list = ibs.get_species_rowids_from_text(species_text_list,␣
↪ensure)
>>> # print(ut.repr2(list(zip(species_text_list, species_rowid_list))))
>>> ibs.print_species_table()
>>> species_text = ibs.get_species_texts(species_rowid_list)
>>> # Ensure we leave testdb1 in a clean state
>>> ibs.delete_species(ibs.get_species_rowids_from_text(['jaguar', 'TYPO']))
>>> all_species_rowids = ibs._get_all_species_rowids()
>>> assert ut.repr2(species_text, nl=False) == ['jaguar', 'zebra_plains', 'zebra_
↪plains', '____', 'typo', '____', 'zebra_grevys', 'bear_polar']
>>> assert ut.repr2(all_species_rowids, nl=False) == [1, 2, 3, 6]
>>> assert ut.repr2(ibs.get_species_texts(all_species_rowids), nl=False) == [
↪'zebra_plains', 'zebra_grevys', 'bear_polar', 'bear_polar+head']
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_species_funcs import *  # NOQA
>>> import wbia
>>> import utool as ut  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> species_text_list = [
```

(continues on next page)

(continued from previous page)

```
...     u'jaguar', u'zebra_plains', u'zebra_plains', '____', 'TYPO',
...     '____', u'zebra_grevys', u'bear_polar']
>>> ensure = False
>>> species_rowid_list = ibs.get_species_rowids_from_text(species_text_list,␣
↪ensure)
```

wbia.control.manual_species_funcs.**get_species_rowids_from_uuids**(*ibs*, *species_uuid_list*)

Returns Creates one if it doesnt exist

Return type species_rowid_list (list)

CommandLine: python -m wbia.control.manual_species_funcs –test-get_species_rowids_from_text:0 python -m wbia.control.manual_species_funcs –test-get_species_rowids_from_text:1

RESTful: Method: GET URL: /api/species/rowid/uuid/

wbia.control.manual_species_funcs.**get_species_texts**(*ibs*, *species_rowid_list*)

Returns species_text_list text names

Return type list

CommandLine: python -m wbia.control.manual_species_funcs –test-get_species_texts –enableall

RESTful: Method: GET URL: /api/species/text/

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_species_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> species_rowid_list = ibs._get_all_species_rowids()
>>> result = get_species_texts(ibs, species_rowid_list)
>>> result = ut.repr2(result)
>>> print(result)
['zebra_plains', 'zebra_grevys', 'bear_polar', 'bear_polar+head']
```

wbia.control.manual_species_funcs.**get_species_uuids**(*ibs*, *species_rowid_list*)

Returns uuids_list - species uuids

Return type list_ (list)

RESTful: Method: GET URL: /api/species/uuid/

wbia.control.manual_species_funcs.**sanitize_species_texts**(*ibs*, *species_text_list*)
changes unknown species to the unknown value

Parameters

- **ibs** (IBEISController) – wbia controller object
- **species_text_list** (*list*) –

Returns species_text_list_

Return type list

CommandLine: python -m wbia.control.manual_species_funcs –test-sanitize_species_texts

RESTful: Method: POST URL: /api/species/sanitize

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.manual_species_funcs import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> species_text_list = ['foo', 'bar', 'zebra_plains']
>>> # execute function
>>> species_text_list_ = sanitize_species_texts(ibs, species_text_list)
>>> # verify results
>>> result = ut.repr2(species_text_list_, nl=False)
>>> print(result)
['foo', 'bar', 'zebra_plains']
```

wbia.control.manual_species_funcs.**set_species_enabled**(*ibs*, *species_rowid_list*, *enabled_list*)

Sets the species all instances enabled bit

1.2.31 wbia.control.manual_test_funcs module

python -c “import utool as ut; ut.write_modscript_alias(‘Tgen.sh’, ‘wbia.templates.template_generator’)” sh Tgen.sh –key test –invert –Tcfg with_getters=True with_setters=False –modfname manual_test_funcs

TODO: Fix this name it is too special case

wbia.control.manual_test_funcs.**add_test**(*ibs*, *test_challenge_list*, *test_response_list*, *test_result_list=None*, *test_uuid_list=None*, *test_user_identity_list=None*)

wbia.control.manual_test_funcs.**delete_test**(*ibs*, *test_rowid_list*)

deletes tests from the database

RESTful: Method: DELETE URL: /api/test/

wbia.control.manual_test_funcs.**get_test_rowids_from_uuid**(*ibs*, *uuid_list*)

wbia.control.manual_test_funcs.**get_test_uuid**(*ibs*, *test_rowid_list*)

1.2.32 wbia.control.manual_wbiacontrol_funcs module

wbia.control.manual_wbiacontrol_funcs.**get_annot_kpts_distinctiveness**(*ibs*, *aid_list*, *config2_=None*, ***kwargs*)

very hacky, but cute way to cache keypoint distinctivness

Parameters

- **ibs** (IBEISController) – wbia controller object

- **aid_list** (*list*) –
- **dstncvs_normer** (*None*) –

Returns dstncvs_list

Return type list

CommandLine: python -m wbia.control.manual_wbiacontrol_funcs –test-get_annot_kpts_distinctiveness

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +SKIP
>>> from wbia.control.manual_wbiacontrol_funcs import *  # NOQA
>>> from wbia.algo.hots import distinctiveness_normalizer
>>> import wbia
>>> import numpy as np
>>> config2_ = None
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids(species=const.TEST_SPECIES.ZEB_PLAIN)
>>> # execute function
>>> aid_list1 = aid_list[::2]
>>> aid_list2 = aid_list[1::3]
>>> dstncvs_list1 = get_annot_kpts_distinctiveness(ibs, aid_list1)
>>> dstncvs_list2 = get_annot_kpts_distinctiveness(ibs, aid_list2)
>>> dstncvs_list = get_annot_kpts_distinctiveness(ibs, aid_list)
>>> print(ut.depth_profile(dstncvs_list1))
>>> stats_dict = ut.dict_stack([ut.get_stats(dstncvs) for dstncvs in dstncvs_
↪list])
>>> print(ut.repr2(stats_dict))
>>> assert np.all(np.array(stats_dict['min']) >= 0), 'distinctiveness was out of␣
↪bounds'
>>> assert np.all(np.array(stats_dict['max']) <= 1), 'distinctiveness was out of␣
↪bounds'
```

wbia.control.manual_wbiacontrol_funcs.**get_feat_kpts_distinctiveness**(*ibs*, *fid_list*, *dstncvs_normer=None*, *species_rowid=None*, ***kwargs*)

wbia.control.manual_wbiacontrol_funcs.**new_query_request**(*ibs*, *qaid_list*, *daid_list*, *cfgdict=None*, *verbose=True*, ***kwargs*)

alias for wbia.algo.hots.query_request.new_wbia_query_request

Parameters

- **qaid_list** (*list*) –
- **daid_list** (*list*) –
- **cfgdict** (*None*) –
- **verbose** (*bool*) –

Returns qreq_ - hyper-parameters

Return type wbia.QueryRequest

wbia.control.manual_wbiacontrol_funcs.**show_annot**(*ibs*, *aid*, **args*, ***kwargs*)
viz helper see wbia.viz.viz_chip.show_chip

wbia.control.manual_wbiacontrol_funcs.**show_annot_image**(*ibs*, *aid*, **args*, ***kwargs*)
viz helper see wbia.viz.viz_chip.show_chip

1.2.33 wbia.control.manual_wildbook_funcs module

CommandLine; # Reset IBEIS database (can skip if done) python -m wbia.tests.reset_testdbs –reset_mtest python -m wbia –tf reset_mtest

Notes

Moving components: java, tomcat, wildbook.war.

python -m utool.util_inspect check_module_usage –pat=”manual_wildbook_funcs.py”

CommandLine; # Start IA server python -m wbia –web –db PZ_MTEST

Reset Wildbook database python -m wbia purge_local_wildbook

Install Wildbook python -m wbia install_wildbook

Startup Wildbook python -m wbia startup_wildbook_server –show

Poll wildbook info python -m wbia get_wildbook_ia_url

Login to wildbook (can skip) python -m wbia test_wildbook_login

Ship ImageSets to wildbook python -m wbia wildbook_signal_imgsetid_list

Change annotations names to a single name python -m wbia wildbook_signal_annot_name_changes:1

Change annotations names back to normal python -m wbia wildbook_signal_annot_name_changes:2

wbia.control.manual_wildbook_funcs.**assert_ia_available_for_wb**(*ibs*, *wb_target=None*)

wbia.control.manual_wildbook_funcs.**delete_wildbook_orphaned_annot_uuids**(*ibs*, *auto_delete=True*)

wbia.control.manual_wildbook_funcs.**delete_wildbook_orphaned_image_uuids**(*ibs*, *auto_delete=True*)

wbia.control.manual_wildbook_funcs.**get_wildbook_annot_uuids**(*ibs*, *filter_match_against_on=True*)

wbia.control.manual_wildbook_funcs.**get_wildbook_base_url**(*ibs*, *wb_target=None*)

wbia.control.manual_wildbook_funcs.**get_wildbook_ia_url**(*ibs*, *wb_target=None*)
Where does wildbook expect us to be?

CommandLine: python -m wbia get_wildbook_ia_url

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_wildbook_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> ia_url = ibs.get_wildbook_ia_url()
>>> print('ia_url = %r' % (ia_url,))
```

wbia.control.manual_wildbook_funcs.**get_wildbook_image_uuids**(*ibs*)

wbia.control.manual_wildbook_funcs.**wildbook_get_existing_names**(*ibs*, *wb_target=None*)

wbia.control.manual_wildbook_funcs.**wildbook_signal_annot_name_changes**(*ibs*, *aid_list=None*, *wb_target=None*, *dryrun=False*)

Parameters

- **aid_list** (*int*) – list of annotation ids(default = None)
- **tomcat_dpath** (*None*) – (default = None)
- **wb_target** (*None*) – (default = None)
- **dryrun** (*bool*) – (default = False)

CommandLine: python -m wbia wildbook_signal_annot_name_changes:0 –dryrun python -m wbia wildbook_signal_annot_name_changes:1 –dryrun python -m wbia wildbook_signal_annot_name_changes:1 python -m wbia wildbook_signal_annot_name_changes:2

Setup:

```
>>> wb_target = None
>>> dryrun = ut.get_argflag('--dryrun')
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_wildbook_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> #gid_list = ibs.get_valid_gids()[0:10]
>>> gid_list = ibs.get_valid_gids()[3:5]
>>> aid_list = ut.flatten(ibs.get_image_aids(gid_list))
>>> # Test case where some names change, some do not. There are no new names.
>>> old_nid_list = ibs.get_annot_name_rowids(aid_list)
>>> new_nid_list = ut.list_roll(old_nid_list, 1)
>>> ibs.set_annot_name_rowids(aid_list, new_nid_list)
>>> result = ibs.wildbook_signal_annot_name_changes(aid_list, wb_target, dryrun)
>>> ibs.set_annot_name_rowids(aid_list, old_nid_list)
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_wildbook_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> #gid_list = ibs.get_valid_gids()[0:10]
>>> gid_list = ibs.get_valid_gids()[3:5]
>>> aid_list = ut.flatten(ibs.get_image_aids(gid_list))
>>> # Test case where all names change to one known name
>>> #old_nid_list = ibs.get_annot_name_rowids(aid_list)
>>> #new_nid_list = [old_nid_list[0]] * len(old_nid_list)
>>> old_nid_list = [1, 2]
>>> new_nid_list = [1, 1]
>>> print('old_nid_list = %r' % (old_nid_list,))
>>> print('new_nid_list = %r' % (new_nid_list,))
>>> ibs.set_annot_name_rowids(aid_list, new_nid_list)
>>> result = ibs.wildbook_signal_annot_name_changes(aid_list, wb_target, dryrun)
>>> # Undo changes here (not undone in wildbook)
>>> #ibs.set_annot_name_rowids(aid_list, old_nid_list)
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_wildbook_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> gid_list = ibs.get_valid_gids()[3:5]
>>> aid_list = ut.flatten(ibs.get_image_aids(gid_list))
>>> old_nid_list = [1, 2]
>>> ibs.set_annot_name_rowids(aid_list, old_nid_list)
>>> # Signal what currently exists (should put them back to normal)
>>> result = ibs.wildbook_signal_annot_name_changes(aid_list, wb_target, dryrun)
```

wbia.control.manual_wildbook_funcs.**wildbook_signal_imgsetid_list**(*ibs*, *imgsetid_list=None*, *set_shipped_flag=True*, *open_url_on_complete=True*, *wb_target=None*, *dryrun=False*)

Exports specified imagesets to wildbook. This is a synchronous call.

Parameters

- **imgsetid_list** (*list*) – (default = None)
- **set_shipped_flag** (*bool*) – (default = True)
- **open_url_on_complete** (*bool*) – (default = True)

RESTful: Method: PUT URL: /api/wildbook/signal/imageset/

Ignore: cd $CODE_DIR/Wildbook/tmp

Ensure IA server is up python -m wbia –web –db PZ_MTEST

Reset IBEIS database python -m wbia.tests.reset_testdbs –reset_mtest python -m wbia reset_mtest

Completely remove Wildbook database python -m wbia purge_local_wildbook

Install Wildbook python -m wbia install_wildbook

Startup Wildbook python -m wbia startup_wildbook_server

Login to wildbook python -m wbia test_wildbook_login

Ship ImageSets to wildbook python -m wbia wildbook_signal_imgsetid_list

Change annotations names to a single name python -m wbia wildbook_signal_annot_name_changes:1

Change annotations names back to normal python -m wbia wildbook_signal_annot_name_changes:2

CommandLine: python -m wbia wildbook_signal_imgsetid_list python -m wbia wildbook_signal_imgsetid_list –dryrun python -m wbia wildbook_signal_imgsetid_list –break

SeeAlso: ~/local/build_scripts/init_wildbook.sh

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.manual_wildbook_funcs import *  # NOQA
>>> dryrun = ut.get_argflag('--dryrun')
>>> wb_target = None
>>> import wbia
>>> # Need to start a web server for wildbook to hook into
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> #gid_list = ibs.get_valid_gids()[0:10]
>>> gid_list = ibs.get_valid_gids()[3:6]
>>> new_imgsetid = ibs.create_new_imageset_from_images(gid_list)  # NOQA
>>> imgsetid = new_imgsetid
>>> print('new imageset uuid = %r' % (ibs.get_imageset_uuid(new_imgsetid),))
>>> print('new imageset text = %r' % (ibs.get_imageset_text(new_imgsetid),))
>>> imgsetid_list = [new_imgsetid]
>>> ibs.set_imageset_processed_flags([new_imgsetid], [1])
>>> gid_list = ibs.get_imageset_gids(new_imgsetid)
>>> ibs.set_image_reviewed(gid_list, [1] * len(gid_list))
>>> set_shipped_flag = True
>>> open_url_on_complete = True
>>> if ut.get_argflag('--bg'):
>>>     with wbia.opendb_bg_web(defaultdb, managed=True) as web_ibs:
...         result = web_ibs.wildbook_signal_imgsetid_list(imgsetid_list, set_
↪shipped_flag, open_url_on_complete, wb_target, dryrun)
>>> else:
...     result = ibs.wildbook_signal_imgsetid_list(imgsetid_list, set_shipped_
↪flag, open_url_on_complete, wb_target, dryrun)
>>> # cleanup
>>> #ibs.delete_imagesets(new_imgsetid)
>>> print(result)
```

wbia.control.manual_wildbook_funcs.**wildbook_signal_name_changes**(*ibs*, *nid_list*, *new_name_list*, *wb_target=None*, *dryrun=False*)

Parameters

- **nid_list** (*int*) – list of name ids
- **new_name_list** (*str*) – list of corresponding names
- **wb_target** (*None*) – (default = None)

- **dryrun** (*bool*) – (default = False)

CommandLine: python -m wbia wildbook_signal_name_changes:0 –dryrun python -m wbia wildbook_signal_name_changes:1 –dryrun python -m wbia wildbook_signal_name_changes:1 python -m wbia wildbook_signal_name_changes:2

Setup:

```
>>> wb_target = None
>>> dryrun = ut.get_argflag('--dryrun')
```

wbia.control.manual_wildbook_funcs.**wildbook_sync**(*ibs*, ***kwargs*)

1.2.34 wbia.control.wildbook_manager module

Manages local wildbook installations.

CommandLine: python -m utool.util_inspect check_module_usage –pat=”wildbook_manager.py”

Utils: # TODO go to http://localhost:8080/wbia/createAssetStore.jsp tail -f ~/.config/wbia/tomcat/logs/catalina.out cat ~/.config/wbia/tomcat/logs/catalina.out python -m wbia shutdown_wildbook_server python -m wbia update_wildbook_install_config

wbia.control.wildbook_manager.**download_tomcat**()
Put tomcat into a directory controlled by wbia

CommandLine: # Reset python -c “import utool as ut; ut.delete(ut.unixjoin(ut.get_app_resource_dir(‘wbia’), ‘tomcat’))”

wbia.control.wildbook_manager.**ensure_local_war**(*verbose=True*)
Ensures tomcat has been unpacked and the war is localized

CommandLine: wbia ensure_local_war

Example

```
>>> # SCRIPT
>>> from wbia.control.wildbook_manager import *  # NOQA
>>> result = ensure_local_war()
>>> print(result)
```

wbia.control.wildbook_manager.**ensure_wb_mysql**()

CommandLine: python -m wbia ensure_wb_mysql

Example

```
>>> # SCRIPT
>>> from wbia.control.wildbook_manager import *  # NOQA
>>> result = ensure_wb_mysql()
```

wbia.control.wildbook_manager.**find_installed_tomcat**(*check_unpacked=True*, *strict=True*)
Asserts that tomcat was properly installed

Parameters **check_unpacked** (*bool*) – (default = True)

Returns tomcat_dpath

Return type str

CommandLine: python -m wbia find_installed_tomcat

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.wildbook_manager import *  # NOQA
>>> check_unpacked = False
>>> strict = False
>>> tomcat_dpath = find_installed_tomcat(check_unpacked, strict)
>>> result = ('tomcat_dpath = %s' % (str(tomcat_dpath),))
>>> print(result)
```

wbia.control.wildbook_manager.**find_java_jvm**()

CommandLine: python -m wbia find_java_jvm

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.wildbook_manager import *  # NOQA
>>> jvm_fpath = find_java_jvm()
>>> result = ('jvm_fpath = %r' % (jvm_fpath,))
>>> print(result)
```

wbia.control.wildbook_manager.**find_or_download_tomcat**()

Returns tomcat_dpath

Return type str

CommandLine: # Reset python -m purge_local_wildbook

python -m wbia –tf purge_local_wildbook python -m wbia –tf find_or_download_tomcat

Example

```
>>> # SCRIPT
>>> from wbia.control.wildbook_manager import *  # NOQA
>>> tomcat_dpath = find_or_download_tomcat()
>>> result = ('tomcat_dpath = %s' % (str(tomcat_dpath),))
>>> print(result)
```

wbia.control.wildbook_manager.**find_or_download_wilbook_warfile**(*ensure=True*, *redownload=False*)

scp jonc@pachy.cs.uic.edu:/var/lib/tomcat/webapps/wbia.war ~/Downloads/pachy_wbia.war wget

http://dev.wildme.org/wbia_data_dir/wbia.war

wbia.control.wildbook_manager.**find_tomcat**(*verbose=True*)

Searches likely places for tomcat to be installed

Returns tomcat_dpath

Return type str

Ignore: locate –regex "tomcat/webapps$"

CommandLine: python -m wbia find_tomcat

Example

```
>>> # SCRIPT
>>> from wbia.control.wildbook_manager import *  # NOQA
>>> tomcat_dpath = find_tomcat()
>>> result = ('tomcat_dpath = %s' % (str(tomcat_dpath),))
>>> print(result)
```

wbia.control.wildbook_manager.**get_tomcat_startup_tmpdir**()

wbia.control.wildbook_manager.**get_wildbook_tomcat_path**(*ibs*, *tomcat_dpath=None*, *wb_target=None*)

wbia.control.wildbook_manager.**install_wildbook**(*verbose=True*)

Script to setup wildbook on a unix based system (hopefully eventually this will generalize to win32)

CommandLine: # Reset wbia purge_local_wildbook wbia ensure_wb_mysql wbia ensure_local_war # Setup wbia install_wildbook # wbia install_wildbook –nomysql # Startup wbia startup_wildbook_server –show

Alternates: wbia install_wildbook –redownload-war wbia install_wildbook –assets wbia startup_wildbook_server –show

Example

```
>>> # SCRIPT
>>> from wbia.control.wildbook_manager import *  # NOQA
>>> verbose = True
>>> result = install_wildbook()
>>> print(result)
```

wbia.control.wildbook_manager.**monitor_wildbook_logs**(*verbose=True*)

Parameters **verbose** (*bool*) – verbosity flag(default = True)

CommandLine: python -m wbia monitor_wildbook_logs –show

Example

```
>>> # SCRIPT
>>> from wbia.control.wildbook_manager import *  # NOQA
>>> monitor_wildbook_logs()
```

wbia.control.wildbook_manager.**purge_local_wildbook**()

Shuts down the server and then purges the server on disk

CommandLine: python -m wbia purge_local_wildbook python -m wbia purge_local_wildbook –purge-war

Example

```
>>> # SCRIPT
>>> from wbia.control.wildbook_manager import *  # NOQA
>>> purge_local_wildbook()
```

wbia.control.wildbook_manager.**shutdown_wildbook_server**(*verbose=True*)

Parameters **verbose** (*bool*) – verbosity flag(default = True)

Ignore: tail -f ~/.config/wbia/tomcat/logs/catalina.out

CommandLine: python -m wbia shutdown_wildbook_server

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.wildbook_manager import *  # NOQA
>>> verbose = True
>>> wb_url = shutdown_wildbook_server()
>>> ut.quit_if_noshow()
>>> ut.get_prefered_browser(PREFERED_BROWSER).open_new_tab(wb_url)
```

wbia.control.wildbook_manager.**startup_wildbook_server**(*verbose=True*)

Parameters **verbose** (*bool*) – verbosity flag(default = True)

CommandLine: python -m wbia startup_wildbook_server python -m wbia startup_wildbook_server –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.wildbook_manager import *  # NOQA
>>> verbose = True
>>> wb_url = startup_wildbook_server()
>>> ut.quit_if_noshow()
>>> ut.get_prefered_browser(PREFERED_BROWSER).open_new_tab(wb_url)
```

wbia.control.wildbook_manager.**tryout_wildbook_login**()

Helper function to test wildbook login automagically

Returns (wb_target, tomcat_dpath)

Return type tuple

CommandLine: python -m wbia tryout_wildbook_login

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.control.wildbook_manager import *  # NOQA
>>> tryout_wildbook_login()
```

wbia.control.wildbook_manager.**update_wildbook_ia_config**(*ibs*, *wildbook_tomcat_path*, *dryrun=False*)

#if use_config_file and wildbook_tomcat_path: # # Update the Wildbook configuration to see *THIS* wbia database # with lockfile.LockFile(lock_fpath): # update_wildbook_ia_config(ibs, wildbook_tomcat_path, dryrun)

wbia.control.wildbook_manager.**update_wildbook_install_config**(*webapps_dpath*, *unpacked_war_dpath*)

CommandLine: python -m wbia ensure_local_war python -m wbia update_wildbook_install_config python -m wbia update_wildbook_install_config –show

Example

```
>>> # xdoctest: +REQUIRES(--tomcat)
>>> from wbia.control.wildbook_manager import *  # NOQA
>>> import wbia
>>> tomcat_dpath = find_installed_tomcat()
>>> webapps_dpath = join(tomcat_dpath, 'webapps')
>>> wb_target = wbia.const.WILDBOOK_TARGET
>>> unpacked_war_dpath = join(webapps_dpath, wb_target)
>>> locals_ = ut.exec_func_src(update_wildbook_install_config, globals())
>>> #update_wildbook_install_config(webapps_dpath, unpacked_war_dpath)
>>> ut.quit_if_noshow()
>>> ut.vd(unpacked_war_dpath)
>>> ut.editfile(locals_['permission_fpath'])
>>> ut.editfile(locals_['jdoconfig_fpath'])
>>> ut.editfile(locals_['asset_store_fpath'])
```

1.2.35 Module contents

wbia.control.**IMPORT_TUPLES = [('DB_SCHEMA', None, False), ('IBEISControl', None, False), (**

cd /home/joncrall/code/wbia/wbia/control makeinit.py -x DBCACHE_SCHEMA_CURRENT DB_SCHEMA_CURRENT _grave_template manual_wbiacontrol_funcs template_definitions templates _autogen_wbiacontrol_funcs

Type Regen Command

wbia.control.**reload_subs**(*verbose=True*)

Reloads wbia.control and submodules

wbia.control.**rrrr**(*verbose=True*)

Reloads wbia.control and submodules

1.3 wbia.dbio package

1.3.1 Submodules

1.3.2 wbia.dbio.export_hsdb module

Converts an IBEIS database to a hotspotter db

wbia.dbio.export_hsdb.**dump_hots_flat_table**(*ibs*)

wbia.dbio.export_hsdb.**dump_hots_tables**(*ibs*)
Dumps hotspotter like tables to disk

wbia.dbio.export_hsdb.**export_wbia_to_hotspotter**(*ibs*)

wbia.dbio.export_hsdb.**get_hots_flat_table**(*ibs*)
Dumps hotspotter flat tables

Parameters **ibs** (IBEISController) – wbia controller object

Returns flat_table_str

Return type str

CommandLine: python -m wbia.dbio.export_hsdb –exec-get_hots_flat_table

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dbio.export_hsdb import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> flat_table_str = get_hots_flat_table(ibs)
>>> result = ('flat_table_str = %s' % (str(flat_table_str),))
>>> print(result)
```

wbia.dbio.export_hsdb.**get_hots_table_strings**(*ibs*)

Parameters **ibs** (IBEISController) – wbia controller object

CommandLine: python -m wbia.dbio.export_hsdb –test-get_hots_table_strings

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dbio.export_hsdb import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> ibs.delete_empty_nids()
>>> # execute function
>>> csvtup = get_hots_table_strings(ibs)
>>> # hack so hashtag is at the end of each line
>>> result = '\n'.join(csvtup).replace('\n', '#\n') + '#'
>>> # verify results
>>> print(result)
# image table#
# num_rows=13#
#   gid,                                                  gname,   aif#
      1,   ../_ibsdb/images/66ec193a-1619-b3b6-216d-1784b4833b61.jpg,    0#
      2,   ../_ibsdb/images/d8903434-942f-e0f5-d6c2-0dcbe3137bf7.jpg,    0#
      3,   ../_ibsdb/images/b73b72f4-4acb-c445-e72c-05ce02719d3d.jpg,    0#
      4,   ../_ibsdb/images/0cd05978-3d83-b2ee-2ac9-798dd571c3b3.jpg,    0#
      5,   ../_ibsdb/images/0a9bc03d-a75e-8d14-0153-e2949502aba7.jpg,    0#
      6,   ../_ibsdb/images/2deeff06-5546-c752-15dc-2bd0fdb1198a.jpg,    0#
```

(continues on next page)

(continued from previous page)

```
      7,  ../_ibsdb/images/68ca272d-26f7-1dbb-76e9-08d192c1a4a7.png,     0#
      8,  ../_ibsdb/images/42fdad98-369a-2cbc-67b1-983d6d6a3a60.jpg,     0#
      9,  ../_ibsdb/images/c459d381-fd74-1d99-6215-e42e3f432ea9.jpg,     0#
     10,  ../_ibsdb/images/33fd9813-3a2b-774b-3fcc-4360d1ae151b.jpg,     0#
     11,  ../_ibsdb/images/97e8ea74-873f-2092-b372-f928a7be30fa.jpg,     0#
     12,  ../_ibsdb/images/588bc218-83a5-d400-21aa-d499832632b0.jpg,     0#
     13,  ../_ibsdb/images/163a890c-36f2-981e-3529-c552b6d668a3.jpg,     0#
# name table#
# num_rows=7#
#   nid,   name#
      1,   easy#
      2,   hard#
      3,   jeff#
      4,   lena#
      5,   occl#
      6,  polar#
      7,  zebra#
# chip table#
# num_rows=13#
#   cid,  gid,  nid,      [tlx tly w h],  theta,                                 ␣
↪      notes#
      1,    1,   -1,  [0  0  1047  715],   0.00,              aid 1 and 2 are␣
↪correct matches#
      2,    2,    1,  [0  0  1035  576],   0.00,                                 ␣
↪           #
      3,    3,    1,  [0  0  1072  804],   0.00,                                 ␣
↪           #
      4,    4,   -4,  [0  0  1072  804],   0.00,                                 ␣
↪           #
      5,    5,    2,  [0  0  1072  804],   0.00,                                 ␣
↪           #
      6,    6,    2,   [0  0  450  301],   0.00,                                 ␣
↪           #
      7,    7,    3,   [0  0  400  400],   0.00,  very simple image to debug␣
↪feature detector#
      8,    8,    4,   [0  0  220  220],   0.00,                           ␣
↪standard test image#
      9,    9,   -9,   [0  0  450  284],   0.00,              this is actually a␣
↪plains zebra#
     10,   10,    5,   [0  0  450  341],   0.00,              this is actually a␣
↪plains zebra#
     11,   11,  -11,   [0  0  741  734],   0.00,                                 ␣
↪           #
     12,   12,    6,   [0  0  673  634],   0.00,                                 ␣
↪           #
     13,   13,    7,  [0  0  1114  545],   0.00,                                 ␣
↪           #
```

wbia.dbio.export_hsdb.**get_hsdb_image_gpaths**(*ibs*, *gid_list*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) –

Returns gpath_list

Return type list

CommandLine: python -m wbia.dbio.export_hsdb –test-get_hsdb_image_gpaths

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dbio.export_hsdb import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids()[0:2]
>>> # execute function
>>> gpath_list = get_hsdb_image_gpaths(ibs, gid_list)
>>> # verify results
>>> result = ut.repr2(gpath_list, nl=1)
>>> print(result)
[
    '../_ibsdb/images/66ec193a-1619-b3b6-216d-1784b4833b61.jpg',
    '../_ibsdb/images/d8903434-942f-e0f5-d6c2-0dcbe3137bf7.jpg',
]
```

1.3.3 wbia.dbio.export_subset module

Exports subset of an IBEIS database to a new IBEIS database

wbia.dbio.export_subset.**check_database_overlap**(*ibs1*, *ibs2*)

CommandLine: python -m wbia.other.dbinfo –test-get_dbinfo:1 –db PZ_MTEST dev.py -t listdbs python -m wbia.dbio.export_subset check_database_overlap –db PZ_MTEST –db2 PZ_MOTHERS

CommandLine: python -m wbia.dbio.export_subset check_database_overlap

python -m wbia.dbio.export_subset check_database_overlap –db1=PZ_MTEST –db2=PZ_Master0 # NOQA python -m wbia.dbio.export_subset check_database_overlap –db1=NNP_Master3 –db2=PZ_Master0 # NOQA

python -m wbia.dbio.export_subset check_database_overlap –db1=GZ_Master0 –db2=GZ_ALL python -m wbia.dbio.export_subset check_database_overlap –db1=GZ_ALL –db2=lewa_grevys

python -m wbia.dbio.export_subset check_database_overlap –db1=PZ_FlankHack –db2=PZ_Master1 python -m wbia.dbio.export_subset check_database_overlap –db1=PZ_PB_RF_TRAIN –db2=PZ_Master1

Example

```
>>> # SCRIPT
>>> from wbia.dbio.export_subset import *  # NOQA
>>> import wbia
>>> import utool as ut
>>> #ibs1 = wbia.opendb(db='PZ_Master0')
>>> #ibs2 = wbia.opendb(dbdir='/raid/work2/PZ_Master')
>>> db1 = ut.get_argval('--db1', str, default='PZ_MTEST')
>>> db2 = ut.get_argval('--db2', str, default='testdb1')
>>> dbdir1 = ut.get_argval('--dbdir1', str, default=None)
>>> dbdir2 = ut.get_argval('--dbdir2', str, default=None)
>>> ibs1 = wbia.opendb(db=db1, dbdir=dbdir1)
```

(continues on next page)

(continued from previous page)

```
>>> ibs2 = wbia.opendb(db=db2, dbdir=dbdir2)
>>> check_database_overlap(ibs1, ibs2)
```

wbia.dbio.export_subset.**check_merge**(*ibs_src*, *ibs_dst*)

wbia.dbio.export_subset.**export_annots**(*ibs*, *aid_list*, *new_dbpath=None*)
exports a subset of annotations and other required info

Todo: PZ_Master1 needs to backproject information back on to NNP_Master3 and PZ_Master0

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*list*) – list of annotation rowids
- **new_dbpath** (*None*) – (default = None)

Returns new_dbpath

Return type str

CommandLine: python -m wbia.dbio.export_subset export_annots python -m wbia.dbio.export_subset export_annots –db NNP_Master3

-a viewpoint_compare –nocache-aid –verbtd –new_dbpath=PZ_ViewPoints

python -m wbia.expt.experiment_helpers get_annotcfg_list:0 –db NNP_Master3 -a viewpoint_compare –nocache-aid –verbtd

python -m wbia.expt.experiment_helpers get_annotcfg_list:0 –db NNP_Master3 -a viewpoint_compare –nocache-aid –verbtd

python -m wbia.expt.experiment_helpers get_annotcfg_list:0 –db NNP_Master3 -a default:aids=all,is_known=True,view_pername=#primary>0&#primary1>0,per_name=4,size=200

python -m wbia.expt.experiment_helpers get_annotcfg_list:0 –db NNP_Master3 -a default:aids=all,is_known=True,view_pername='#primary>0&#primary1>0',per_name=4,size=200 –acfginfo

python -m wbia.expt.experiment_helpers get_annotcfg_list:0 –db PZ_Master1 -a default:has_any=photobomb –acfginfo

Example

```
>>> # SCRIPT
>>> from wbia.dbio.export_subset import *  # NOQA
>>> import wbia
>>> from wbia.expt import experiment_helpers
>>> ibs = wbia.opendb(defaultdb='NNP_Master3')
>>> acfg_name_list = ut.get_argval(('--aidcfg', '--acfg', '-a'), type_=list,␣
↪default=[''])
>>> acfg_list, expanded_aids_list = experiment_helpers.get_annotcfg_list(ibs,␣
↪acfg_name_list)
>>> aid_list = expanded_aids_list[0][0]
>>> ibs.print_annot_stats(aid_list, viewcode_isect=True, per_image=True)
```

(continues on next page)

(continued from previous page)

```
>>> # Expand to get all annots in each chosen image
>>> gid_list = ut.unique_ordered(ibs.get_annot_gids(aid_list))
>>> aid_list = ut.flatten(ibs.get_image_aids(gid_list))
>>> ibs.print_annot_stats(aid_list, viewcode_isect=True, per_image=True)
>>> new_dbpath = ut.get_argval('--new-dbpath', default='PZ_ViewPoints')
>>> new_dbpath = export_annots(ibs, aid_list, new_dbpath)
>>> result = ('new_dbpath = %s' % (str(new_dbpath),))
>>> print(result)
```

wbia.dbio.export_subset.**export_data**(*ibs*, *gid_list*, *aid_list*, *nid_list*, *new_dbpath=None*)
exports a subset of data and other required info

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) – list of image rowids
- **aid_list** (*list*) – list of annotation rowids
- **nid_list** (*list*) – list of name rowids
- **imgsetid_list** (*list*) – list of imageset rowids
- **gsgrid_list** (*list*) – list of imageset-image pairs rowids
- **new_dbpath** (*None*) – (default = None)

Returns new_dbpath

Return type str

wbia.dbio.export_subset.**export_images**(*ibs*, *gid_list*, *new_dbpath=None*)
exports a subset of images and other required info

Todo: PZ_Master1 needs to backproject information back on to NNP_Master3 and PZ_Master0

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) – list of annotation rowids
- **new_dbpath** (*None*) – (default = None)

Returns new_dbpath

Return type str

wbia.dbio.export_subset.**export_names**(*ibs*, *nid_list*, *new_dbpath=None*)
exports a subset of names and other required info

Parameters

- **ibs** (IBEISController) – wbia controller object
- **nid_list** (*list*) –

CommandLine: python -m wbia.dbio.export_subset –test-export_names

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.dbio.export_subset import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb2')
>>> ibs.delete_empty_nids()
>>> nid_list = ibs._get_all_known_nids()[0:2]
>>> # execute function
>>> result = export_names(ibs, nid_list)
>>> # verify results
>>> print(result)
```

wbia.dbio.export_subset.**find_gid_list**(*ibs*, *min_count=500*, *ensure_annots=False*)

wbia.dbio.export_subset.**find_overlap_annots**(*ibs1*, *ibs2*, *method='annots'*)
Finds the aids of annotations in ibs1 that are also in ibs2

ibs1 = wbia.opendb('PZ_Master1') ibs2 = wbia.opendb('PZ_MTEST')

wbia.dbio.export_subset.**fix_annotmatch_pzmaster1**()
PZ_Master1 had annotmatch rowids that did not agree with the current name labeling. Looking at the inconsistencies in the graph interface was too cumbersome, because over 3000 annots were incorrectly grouped together.

This function deletes any annotmatch rowid that is not consistent with the current labeling so we can go forward with using the new AnnotInference object

wbia.dbio.export_subset.**fix_bidirectional_annotmatch**(*ibs*)

wbia.dbio.export_subset.**make_new_dbpath**(*ibs*, *id_label*, *id_list*)
Creates a new database path unique to the exported subset of ids.

wbia.dbio.export_subset.**merge_databases**(*ibs_src*, *ibs_dst*, *rowid_subsets=None*, *localize_images=True*)
New way of merging using the non-hacky sql table merge. However, its only workings due to major hacks.

FIXME: annotmatch table

CommandLine: python -m wbia –test-merge_databases

python -m wbia merge_databases:0 –db1 LF_OPTIMIZADAS_NI_V_E –db2 LF_ALL python -m wbia merge_databases:0 –db1 LF_WEST_POINT_OPTIMIZADAS –db2 LF_ALL

python -m wbia merge_databases:0 –db1 PZ_Master0 –db2 PZ_Master1 python -m wbia merge_databases:0 –db1 NNP_Master3 –db2 PZ_Master1

python -m wbia merge_databases:0 –db1 GZ_ALL –db2 GZ_Master1 python -m wbia merge_databases:0 –db1 lewa_grevys –db2 GZ_Master1

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dbio.export_subset import *  # NOQA
>>> from wbia.init.sysres import get_workdir
>>> import wbia
>>> db1 = ut.get_argval('--db1', str, default=None)
>>> db2 = ut.get_argval('--db2', str, default=None)
>>> dbdir1 = ut.get_argval('--dbdir1', str, default=None)
```

(continues on next page)

(continued from previous page)

```
>>> dbdir2 = ut.get_argval('--dbdir2', str, default=None)
>>> delete_ibsdir = False
>>> # Check for test mode instead of script mode
>>> if db1 is None and db2 is None and dbdir1 is None and dbdir2 is None:
...     dbdir1 = '/'.join([get_workdir(), 'testdb1'])
...     dbdir2 = '/'.join([get_workdir(), 'testdb_dst'])
...     delete_ibsdir = True
>>> # Open the source and destination database
>>> assert db1 is not None or dbdir1 is not None
>>> assert db2 is not None or dbdir2 is not None
>>> ibs_src = wbia.opendb(db=db1, dbdir=dbdir1)
>>> ibs_dst = wbia.opendb(db=db2, dbdir=dbdir2, allow_newdir=True,
...                       delete_ibsdir=delete_ibsdir)
>>> merge_databases(ibs_src, ibs_dst)
>>> check_merge(ibs_src, ibs_dst)
>>> # ibs_dst.print_dbinfo()
```

wbia.dbio.export_subset.**remerge_subset**()

Assumes ibs1 is an updated subset of ibs2. Re-merges ibs1 back into ibs2.

TODO: annotmatch table must have non-directional edges for this to work. I.e. u < v

Ignore:

Ensure annotmatch and names are up to date with staging

Load graph import ibei ibs = wbia.opendb('PZ_PB_RF_TRAIN') infr = wbia.AnnotInference(aids='all', ibs=ibs, verbose=3) infr.reset_feedback('staging', apply=True) infr.relabel_using_reviews()

Check deltas infr.wbia_name_group_delta_info() infr.wbia_delta_info()

Write if it looks good infr.write_wbia_annotmatch_feedback() infr.write_wbia_name_assignment()

Ignore: import wbia ibs = wbia.opendb('PZ_Master1') infr = wbia.AnnotInference(ibs, 'all') infr.reset_feedback('annotmatch', apply=True)

CommandLine: python -m wbia.dbio.export_subset remerge_subset

wbia.dbio.export_subset.**slow_merge_test**()

CommandLine: python -m wbia.dbio.export_subset –test-slow_merge_test

Example

```
>>> # SLOW_DOCTEST
>>> from wbia.dbio.export_subset import *  # NOQA
>>> result = slow_merge_test()
>>> print(result)
```

1.3.4 wbia.dbio.ingest_database module

This module lists known raw databases and how to ingest them.

Specify arguments and run the following command to ingest a database

python -m wbia –tf ingest_rawdata –db seaturtles –imgdir "~/turtles/Turtles from Jill" –ingest-type=named_folders –species=turtles python -m wbia –tf ingest_rawdata –db PZ_OlPej2016 –imgdir /raid/raw/OlPejPZ_June_2016 –ingest-type=named_folders –species=zebra_plains

— GET DATA — rsync -avhzP <user>@<host>:<remotedir> <path-to-raw-imgs> # — RUN INGEST SCRIPT — python -m wbia –tf ingest_rawdata –db <new-wbia-db-name> –imgdir <path-to-raw-imgs> –ingest-type=named_folders –species=<optional> –fmtkey=<optional>

Example

```
>>> # xdoctest: +SKIP
>>> # The scripts in this file essentiall do this:
>>> dbdir = '<your new database directory>'
>>> gpath_list = '<path to your images>'
>>> ibs = wbia.opendb(dbdir=dbdir, allow_newdir=True)
>>> gid_list_ = ibs.add_images(gpath_list, auto_localize=False)  # NOQA
>>> # use whole images as annotations
>>> aid_list = ibs.use_images_as_annotations(gid_list_, adjust_percent=0)
>>> # Extra stuff
>>> name_list = '<names that correspond to your annots>'
>>> ibs.set_annot_names(aid_list, name_list)
>>> occur_text_list = '<occurrence that images belongs to>'
>>> ibs.set_image_imagesettext(gid_list_, occur_text_list)
>>> ibs.append_annot_case_tags(aid_list, '<annotation tags>')
```

class wbia.dbio.ingest_database.**FMT_KEYS**

Bases: object

elephant_fmt = '{prefix?}{name}_{view}_{id?}.{ext}'

giraffe1_fmt = '{name:*}_{id:d}.{ext}'

name_fmt = '{name:*}[id:d].{ext}'

seal2_fmt = '{name:Phsd*}{id:[A-Z]}.{ext}'

snails_fmt = '{name:*dd}{id:dd}.{ext}'

class wbia.dbio.ingest_database.**Ingestable**(*dbname*, *img_dir=None*, *ingest_type=None*, *fmtkey=None*, *adjust_percent=0.0*, *postingest_func=None*, *zipfile=None*, *species=None*, *images_as_annots=False*)

Bases: object

Temporary structure representing how to ingest a databases

ensure_feasibility()

class wbia.dbio.ingest_database.**Ingestable2**(*dbdir*, *imgpath_list=None*, *imgdir_list=None*, *zipfile_list=None*, *postingest_func=None*, *ingest_config={}*, ***kwargs*)

Bases: object

execute(*ibs=None*)

wbia.dbio.ingest_database.**get_name_texts_from_gnames**(*gpath_list*, *img_dir*, *fmtkey='{name:*}[aid:d].{ext}'*)

Parameters

- **gpath_list** (*list*) – list of image paths

- **img_dir** (*str*) – path to image directory
- **fmtkey** (*str*) – pattern string to parse names from (default = '{name:*}[aid:d].{ext}')

Returns name_list - based on the parent folder of each image

Return type list

CommandLine: python -m wbia.dbio.ingest_database –test-get_name_texts_from_gnames

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.dbio.ingest_database import *  # NOQA
>>> gpath_list = ['e_f0273_f.jpg', 'f0001_f.jpg', 'f0259_l_3.jpg', 'f0259_f_1.jpg
↪', 'f0259_f (1).jpg', 'f0058_u16_f.jpg']
>>> img_dir = ''
>>> fmtkey = FMT_KEYS.elephant_fmt
>>> result = get_name_texts_from_gnames(gpath_list, img_dir, fmtkey)
>>> print(result)
```

wbia.dbio.ingest_database.**get_name_texts_from_parent_folder**(*gpath_list*, *img_dir*, *fmtkey=None*)

Input: gpath_list Output: names based on the parent folder of each image

wbia.dbio.ingest_database.**get_standard_ingestable**(*dbname*)

wbia.dbio.ingest_database.**ingest_Elephants_drop1**(*dbname*)

wbia.dbio.ingest_database.**ingest_Giraffes1**(*dbname*)

wbia.dbio.ingest_database.**ingest_JAG_Kieryn**(*dbname*)

wbia.dbio.ingest_database.**ingest_coco_style_db**(*dbdir*, *dryrun=False*)

Ingest a PASCAL VOC formatted datbase

Parameters **dbdir** (*str*) –

CommandLine: python -m wbia.dbio.ingest_database –exec-ingest_coco_style_db –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.dbio.ingest_database import *  # NOQA
>>> dbdir = '/Datasets/coco'
>>> dryrun = True
>>> ingest_coco_style_db(dbdir)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

wbia.dbio.ingest_database.**ingest_humpbacks**(*dbname*)

wbia.dbio.ingest_database.**ingest_lynx**(*dbname*)

CommandLine: python -m wbia.dbio.ingest_database –exec-injest_main –db lynx

wbia.dbio.ingest_database.**ingest_oxford_style_db**(*dbdir*, *dryrun=False*)

Ingest either oxford or paris

Parameters **dbdir** (*str*) –

CommandLine: python -m wbia.dbio.ingest_database –exec-ingest_oxford_style_db –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.dbio.ingest_database import *  # NOQA
>>> dbdir = '/raid/work/Oxford'
>>> dryrun = True
>>> ingest_oxford_style_db(dbdir)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

Ignore:

```
>>> from wbia.dbio.ingest_database import *  # NOQA
>>> import wbia
>>> dbdir = '/raid/work/Oxford'
>>> dbdir = '/raid/work/Paris'
>>>
#>>> wbia.dbio.convert_db.ingest_oxford_style_db(dbdir)
```

wbia.dbio.ingest_database.**ingest_polar_bears**(*dbname*)

wbia.dbio.ingest_database.**ingest_rawdata**(*ibs*, *ingestable*, *localize=False*)

Ingests rawdata into an wbia database.

Parameters

- **ibs** (*wbia.IBEISController*) – wbia controller object
- **ingestable** (Ingestable) –
- **localize** (*bool*) – (default = False)

Returns aid_list - list of annotation rowids

Return type list

Notes

if ingest_type == 'named_folders': Converts folder structure where folders = name, to ibsdb

if ingest_type == 'named_images': Converts imgname structure where imgnames = name_id.ext, to ibsdb

CommandLine: python wbia/dbio/ingest_database.py –db seals_drop2 python -m wbia.dbio.ingest_database –exec-ingest_rawdata python -m wbia.dbio.ingest_database –exec-ingest_rawdata –db snow-leopards –imgdir /raid/raw_rsync/snow-leopards

python -m wbia –tf ingest_rawdata –db wd_peter2 –imgdir /raid/raw_rsync/african-dogs –ingest-type=named_folders –species=wild_dog –fmtkey='African Wild Dog: {name}' –force-delete python -m wbia –tf ingest_rawdata –db <newdbname> –imgdir <path-to-images> –ingest-type=named_folders –species=humpback

Example

```
>>> # SCRIPT
>>> # General ingest script
>>> from wbia.dbio.ingest_database import *  # NOQA
>>> import wbia
>>> dbname = ut.get_argval('--db', str, None)  # 'snow-leopards')
>>> force_delete = ut.get_argflag(('--force_delete', '--force-delete'))
>>> img_dir = ut.get_argval('--imgdir', type_=str, default=None)
>>> ingest_type = ut.get_argval('--ingest-type', type_=str, default='unknown')
>>> fmtkey = ut.get_argval('--fmtkey', type_=str, default=None)
>>> species = ut.get_argval('--species', type_=str, default=None)
>>> images_as_annots = ut.get_argval('--images-as-annots', type_=bool,␣
↪default=None)
>>> if images_as_annots is None:
>>>     images_as_annots = ingest_type != 'unknown'
>>> assert img_dir is not None, 'specify img dir'
>>> assert dbname is not None, 'specify dbname'
>>> ingestable = Ingestable(
>>>     dbname, img_dir=img_dir, ingest_type=ingest_type,
>>>     fmtkey=fmtkey, species=species, images_as_annots=images_as_annots,
>>>     adjust_percent=0.00)
>>> from wbia.control import IBEISControl
>>> dbdir = wbia.sysres.db_to_dbdir(dbname, allow_newdir=True)
>>> ut.ensuredir(dbdir, verbose=True)
>>> if force_delete:
>>>     ibsfuncs.delete_wbia_database(dbdir)
>>> ibs = IBEISControl.request_IBEISController(dbdir)
>>> localize = False
>>> gid_list = ingest_rawdata(ibs, ingestable, localize)
>>> result = ('gid_list = %s' % (str(gid_list),))
>>> print(result)
```

wbia.dbio.ingest_database.**ingest_seals_drop2**(*dbname*)

wbia.dbio.ingest_database.**ingest_serengeti_mamal_cameratrap**(*species*)

Downloads data from Serengeti dryad server

References

http://datadryad.org/resource/doi:10.5061/dryad.5pt92 Swanson AB, Kosmala M, Lintott CJ, Simpson RJ, Smith A, Packer C (2015) Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna. Scientific Data 2: 150026. http://dx.doi.org/10.1038/sdata.2015.26 Swanson AB, Kosmala M, Lintott CJ, Simpson RJ, Smith A, Packer C (2015) Data from: Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna. Dryad Digital Repository. http://dx.doi.org/10.5061/dryad.5pt92

Parameters **species** –

CommandLine: python -m wbia.dbio.ingest_database –test-ingest_serengeti_mamal_cameratrap –species zebra_plains python -m wbia.dbio.ingest_database –test-ingest_serengeti_mamal_cameratrap –species cheetah

Example

```
>>> # SCRIPT
>>> from wbia.dbio.ingest_database import *  # NOQA
>>> import wbia
>>> species = ut.get_argval('--species', type_=str, default=wbia.const.TEST_
↪SPECIES.ZEB_PLAIN)
>>> # species = ut.get_argval('--species', type_=str, default='cheetah')
>>> result = ingest_serengeti_mamal_cameratrap(species)
>>> print(result)
```

wbia.dbio.ingest_database.**ingest_snails_drop1**(*dbname*)

wbia.dbio.ingest_database.**ingest_standard_database**(*dbname*, *force_delete=False*)

Parameters

- **dbname** (*str*) – database name
- **force_delete** (*bool*) –

Ignore:

```
>>> from wbia.dbio.ingest_database import *  # NOQA
>>> dbname = 'testdb1'
>>> force_delete = False
>>> result = ingest_standard_database(dbname, force_delete)
>>> print(result)
```

wbia.dbio.ingest_database.**ingest_testdb1**(*dbname*)

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.dbio.ingest_database import *  # NOQA
>>> import utool as ut
>>> from wbia import demodata
>>> import wbia
>>> demodata.ensure_testdata()
>>> # DELETE TESTDB1
>>> TESTDB1 = ut.unixjoin(wbia.sysres.get_workdir(), 'testdb1')
>>> ut.delete(TESTDB1, ignore_errors=False)
>>> result = ingest_testdb1(dbname)
```

wbia.dbio.ingest_database.**ingest_whale_sharks**(*dbname*)

CommandLine: python -m wbia.dbio.ingest_database –exec-injest_main –db WS_ALL

wbia.dbio.ingest_database.**ingest_wilddog_peter**(*dbname*)

CommandLine: python -m wbia.dbio.ingest_database –exec-injest_main –db wd_peter_blinston

wbia.dbio.ingest_database.**injest_main**()

CommandLine: python -m wbia.dbio.ingest_database –test-injest_main python -m wbia.dbio.ingest_database –test-injest_main –db snow-leopards

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.dbio.ingest_database import *  # NOQA
>>> injest_main()
```

wbia.dbio.ingest_database.**logger = <Logger wbia (INFO)>**
New Lynx

python -m wbia –tf ingest_rawdata –db lynx2 –imgdir "/media/raid/raw/WildME-WWF-lynx-Sept-2016/CARPETAS CATALOGO INDIVIDUOS" –ingest-type=named_folders –species=lynx –dry

wbia.dbio.ingest_database.**normalize_name**(*name*)
Maps unknonwn names to the standard ____

wbia.dbio.ingest_database.**resolve_name_conflicts**(*gid_list*, *name_list*)

1.3.5 wbia.dbio.ingest_ggr module

Converts a GGR-style raw data to IBEIS database.

wbia.dbio.ingest_ggr.**convert_ggr2018_to_wbia**(*ggr_path*, *dbdir=None*, *purge=True*, *dry_run=False*, *apply_updates=True*, ***kwargs*)
Convert the raw GGR2 (2018) data to an wbia database.

Args ggr_path (str): Directory to folder *containing* raw GGR 2018 data dbdir (str): Output directory

CommandLine: python -m wbia convert_ggr2018_to_wbia

Example

```
>>> # SCRIPT
>>> from wbia.dbio.ingest_ggr import *  # NOQA
>>> default_ggr_path = join('/', 'data', 'wbia', 'GGR2', 'GGR2018data')
>>> default_dbdir = join('/', 'data', 'wbia', 'GGR2-IBEIS')
>>> dbdir = ut.get_argval('--dbdir', type_=str, default=default_dbdir)
>>> ggr_path = ut.get_argval('--ggr', type_=str, default=default_ggr_path)
>>> result = convert_ggr2018_to_wbia(ggr_path, dbdir=dbdir, purge=False, dry_
↪run=True, apply_updates=False)
>>> print(result)
```

1.3.6 wbia.dbio.ingest_hsdb module

Converts a hotspostter database to IBEIS

wbia.dbio.ingest_hsdb.**check_unconverted_hsdb**(*dbdir*)
Returns if a directory is an unconverted hotspotter database

wbia.dbio.ingest_hsdb.**convert_hsdb_to_wbia**(*hsdir*, *dbdir=None*, ***kwargs*)

Args hsdir (str): Directory to folder *containing* _hsdb dbdir (str): Output directory (defaults to same as hsdb)

CommandLine: python -m wbia convert_hsdb_to_wbia –dbdir ~/work/Frogs python -m wbia convert_hsdb_to_wbia –hsdir "/raid/raw/RotanTurtles/Roatan HotSpotter Nov_21_2016"

Ignore: from wbia.dbio.ingest_hsdb import * # NOQA hsdir = "/raid/raw/RotanTurtles/Roatan HotSpotter Nov_21_2016" dbdir = "~/work/RotanTurtles"

Example

```
>>> # SCRIPT
>>> from wbia.dbio.ingest_hsdb import *  # NOQA
>>> dbdir = ut.get_argval('--dbdir', type_=str, default=None)
>>> hsdir = ut.get_argval('--hsdir', type_=str, default=dbdir)
>>> result = convert_hsdb_to_wbia(hsdir)
>>> print(result)
```

wbia.dbio.ingest_hsdb.**get_hsinternal**(*hsdb_dir*)

wbia.dbio.ingest_hsdb.**get_unconverted_hsdbs**(*workdir=None*)

Parameters **workdir** (*None*) – (default = None)

CommandLine: python -m wbia.dbio.ingest_hsdb –test-get_unconverted_hsdbs

Example

```
>>> # SCRIPT
>>> from wbia.dbio.ingest_hsdb import *  # NOQA
>>> workdir = None
>>> result = get_unconverted_hsdbs(workdir)
>>> print(result)
```

wbia.dbio.ingest_hsdb.**ingest_unconverted_hsdbs_in_workdir**()

wbia.dbio.ingest_hsdb.**is_hsdb**(*dbdir*)

wbia.dbio.ingest_hsdb.**is_hsdbv3**(*dbdir*)

wbia.dbio.ingest_hsdb.**is_hsdbv4**(*dbdir*)

wbia.dbio.ingest_hsdb.**is_hsinternal**(*dbdir*)

wbia.dbio.ingest_hsdb.**is_succesful_convert**(*dbdir*)
the sucess flag is only written if the _ibsdb was properly generated

wbia.dbio.ingest_hsdb.**testdata_ensure_unconverted_hsdb**()
Makes an unconverted test datapath

CommandLine: python -m wbia.dbio.ingest_hsdb –test-testdata_ensure_unconverted_hsdb

Example

```
>>> # SCRIPT
>>> from wbia.dbio.ingest_hsdb import *  # NOQA
>>> result = testdata_ensure_unconverted_hsdb()
>>> print(result)
```

1.3.7 wbia.dbio.ingest_mdb module

1.3.8 wbia.dbio.ingest_my_hotspotter_dbs module

1.3.9 Module contents

this module handles importing and exporting. the best word i can think of is io. maybe marshall?

1.4 wbia.detecttools package

1.4.1 Subpackages

1.4.1.1 wbia.detecttools.ctypes_interface package

1.4.1.1.1 Module contents

wbia.detecttools.ctypes_interface.**find_lib_fpath**(*libname*, *root_dir*, *recurse_down=True*, *verbose=False*)
Search for the library

wbia.detecttools.ctypes_interface.**get_lib_dpath_list**(*root_dir*)
returns possible lib locations

Parameters **root_dir** (*str*) – deepest directory to look for a library (dll, so, dylib)

Returns plausible directories to look for libraries

Return type list

wbia.detecttools.ctypes_interface.**get_lib_fname_list**(*libname*)

Parameters **libname** (*str*) – library name (e.g. 'hesaff', not 'libhesaff')

Returns list of plausible library file names

Return type list

wbia.detecttools.ctypes_interface.**load_clib**(*libname*, *root_dir*)
Does the work.

Parameters

- **libname** (*str*) – library name (e.g. 'hesaff', not 'libhesaff')
- **root_dir** (*str*) – the deepest directory searched for the library file (dll, dylib, or so).

Returns clib a ctypes object used to interface with the library

Return type ctypes.cdll

1.4.1.2 wbia.detecttools.directory package

1.4.1.2.1 Module contents

class wbia.detecttools.directory.**Directory**(*directory_path*, ***kwargs*)
Bases: object

base()

directories(***kwargs*)

files(***kwargs*)

num_directories(***kwargs*)

num_files(***kwargs*)

1.4.1.3 wbia.detecttools.pascaldata package

1.4.1.3.1 Submodules

1.4.1.3.2 wbia.detecttools.pascaldata.common module

wbia.detecttools.pascaldata.common.**get**(*et*, *category*, *text=True*, *singularize=True*)

wbia.detecttools.pascaldata.common.**histogram**(*_list*)

wbia.detecttools.pascaldata.common.**openImage**(*filename*, *color=False*, *alpha=False*)

wbia.detecttools.pascaldata.common.**randColor**()

wbia.detecttools.pascaldata.common.**randInt**(*lower*, *upper*)

1.4.1.3.3 wbia.detecttools.pascaldata.pascal_image module

class wbia.detecttools.pascaldata.pascal_image.**PASCAL_Image**(*filename_xml*, *absolute_dataset_path*, ***kwargs*)

Bases: object

accuracy(*prediction_list*, *category*, *alpha=0.5*)

bounding_boxes(*parts=False*)

categories(*unique=True*, *patches=False*)

image_path()

show(*objects=True*, *parts=True*, *display=True*, *prediction_list=None*, *category=None*, *alpha=0.5*)

1.4.1.3.4 wbia.detecttools.pascaldata.pascal_object module

class wbia.detecttools.pascaldata.pascal_object.**PASCAL_Object**(*_xml*, *width*, *height*, *name=None*, ***kwargs*)

Bases: object

bounding_box(*parts=False*)

1.4.1.3.5 wbia.detecttools.pascaldata.pascal_part module

class wbia.detecttools.pascaldata.pascal_part.**PASCAL_Part**(*_xml*, ***kwargs*)

Bases: object

bounding_box()

1.4.1.3.6 Module contents

class wbia.detecttools.pascaldata.**PASCAL_Data**(*dataset_path*, ***kwargs*)
Bases: object

dataset(*positive_category*, *neg_exclude_categories=[]*, *max_rois_pos=None*, *max_rois_neg=None*)

print_distribution()

1.4.1.4 wbia.detecttools.pypascalmarkup package

1.4.1.4.1 Module contents

class wbia.detecttools.pypascalmarkup.**PascalVOC_Markup_Annotation**(*fullpath*, *folder*, *filename*, ***kwargs*)
Bases: object

add_object(*name*, *bounding_box*, ***kwargs*)

add_part(*object_index*, *name*, *bounding_box*)

xml()

yml()

class wbia.detecttools.pypascalmarkup.**PascalVOC_Markup_Object**(*name*, *bbox_XxYy*, ***kwargs*)
Bases: object

add_part(*name*, *bounding_box*)

xml()

yml()

class wbia.detecttools.pypascalmarkup.**PascalVOC_Markup_Part**(*name*, *bbox*)
Bases: object

xml()

yml()

1.4.1.5 wbia.detecttools.wbiadata package

1.4.1.5.1 Submodules

1.4.1.5.2 wbia.detecttools.wbiadata.common module

wbia.detecttools.wbiadata.common.**get**(*et*, *category*, *text=True*, *singularize=True*)

wbia.detecttools.wbiadata.common.**histogram**(*_list*)

wbia.detecttools.wbiadata.common.**openImage**(*filename*, *color=False*, *alpha=False*)

wbia.detecttools.wbiadata.common.**randColor**()

wbia.detecttools.wbiadata.common.**randInt**(*lower*, *upper*)

1.4.1.5.3 wbia.detecttools.wbiadata.wbia_image module

class wbia.detecttools.wbiadata.wbia_image.**IBEIS_Image**(*filename_xml*, *absolute_dataset_path*, ***kwargs*)

Bases: object

accuracy(*prediction_list*, *category*, *alpha=0.5*)

bounding_boxes(***kwargs*)

categories(*unique=True*, *sorted_=True*, *patches=False*)

image_path()

show(*objects=True*, *parts=True*, *display=True*, *prediction_list=None*, *category=None*, *alpha=0.5*, *label=True*)

1.4.1.5.4 wbia.detecttools.wbiadata.wbia_object module

class wbia.detecttools.wbiadata.wbia_object.**IBEIS_Object**(*_xml*, *width*, *height*, *name=None*, ***kwargs*)

Bases: object

bounding_box(*parts=False*)

1.4.1.5.5 wbia.detecttools.wbiadata.wbia_part module

class wbia.detecttools.wbiadata.wbia_part.**IBEIS_Part**(*_xml*, ***kwargs*)

Bases: object

bounding_box()

1.4.1.5.6 Module contents

class wbia.detecttools.wbiadata.**IBEIS_Data**(*dataset_path*, ***kwargs*)

Bases: object

dataset(*positive_category*, *neg_exclude_categories=[]*, *max_rois_pos=None*, *max_rois_neg=None*)

parse_dataset(*category*, *_type*)

print_distribution()

1.4.2 Module contents

1.5 wbia.dtool package

1.5.1 Submodules

1.5.2 wbia.dtool.base module

class wbia.dtool.base.**AlgoResult**

Bases: object

Base class for algo result objects

copy()

classmethod load_from_fpath(*fpath*, *verbose=False*)

save_to_fpath(*fpath*, *verbose=False*)

class wbia.dtool.base.**AnnotSimiliarity**
Bases: object

get_data_hashid()

get_query_hashid()

class wbia.dtool.base.**BaseRequest**
Bases: *wbia.dtool.base.IBEISRequestHacks*, utool.util_dev.NiceRepr

Class that maintains both an algorithm, inputs, and a config.

ensure_dependencies()

CommandLine: python -m dtool.base –exec-BaseRequest.ensure_dependencies

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.base import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> request = depc.new_request('vsmany', [1, 2], [2, 3, 4])
>>> request.ensure_dependencies()
```

execute(*parent_rowids=None*, *use_cache=None*, *postprocess=True*)

get_cfgstr(*with_input=False*, *with_pipe=True*, ***kwargs*)
main cfgstring used to identify the 'querytype'

get_input_hashid()

get_pipe_cfgstr()

get_pipe_hashid()

classmethod new(*depc*, *parent_rowids*, *cfgdict=None*, *tablename=None*)

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

static static_new(*cls*, *depc*, *parent_rowids*, *cfgdict=None*, *tablename=None*)
hack for autoreload

class wbia.dtool.base.**ClassVsClassSimilarityRequest**
Bases: *wbia.dtool.base.BaseRequest*

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

class wbia.dtool.base.**Config**(***kwargs*)
Bases: utool.util_dev.NiceRepr, utool.util_dict.DictLike

Base class for heirarchical config need to overwrite get_param_info_list

CommandLine: python -m dtool.base Config

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.base import *  # NOQA
>>> cfg1 = Config.from_dict({'a': 1, 'b': 2})
>>> cfg2 = Config.from_dict({'a': 2, 'b': 2})
>>> # Must be hashable and orderable
>>> hash(cfg1)
>>> cfg1 > cfg2
```

assert_self_types(*verbose=True*)

classmethod class_from_dict(*dict_*, *tablename=None*)

deepcopy()

classmethod from_argv_cfgs()
handy command line tool

classmethod from_argv_dict(***kwargs*)
handy command line tool ut.parse_argv_cfg

classmethod from_dict(*dict_*, *tablename=None*)

Parameters

- **dict** (dict_) – a dictionary
- **tablename** (*None*) – (default = None)

Returns param_info_list

Return type list

CommandLine: python -m dtool.base Config.from_dict –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.dtool.base import *  # NOQA
>>> cls = Config
>>> dict_ = {'K': 1, 'Knorm': 5, 'min_pername': 1, 'max_pername': 1,}
>>> tablename = None
>>> config = cls.from_dict(dict_, tablename)
>>> print(config)
>>> # xdoctest: +REQUIRES(--show)
>>> ut.quit_if_noshow()
>>> dlg = config.make_qt_dialog(
>>>     title='Confirm Merge Query',
>>>     msg='Confirm')
>>> dlg.resize(700, 500)
>>> dlg.show()
>>> import wbia.plottool as pt
>>> self = dlg.widget
>>> guitool.qtapp_loop(qwin=dlg)
>>> updated_config = self.config  # NOQA
>>> print('updated_config = %r' % (updated_config,))
```

get(*key*, **d*)
get a paramater value by string

get_cfgstr(***kwargs*)

get_cfgstr_list(*ignore_keys=None*, *with_name=True*, ***kwargs*)
default get_cfgstr_list, can be overrided by a config object

get_config_name(***kwargs*)
the user might want to overwrite this function

get_hashid()

get_param_info_dict()

get_param_info_list()

get_sub_config_list()

get_varnames()

getinfo(*key*)

getitem(*key*)
Required for DictLike interface

getstate_todict_recursive()

initialize_params(***kwargs*)
Initializes config class attributes based on params info list

keys()
Required for DictLike interface

make_qt_dialog(*parent=None*, *title='Edit Config'*, *msg='Confim'*)

native_items()

nested_items()

parse_items()

Returns param_list

Return type list

CommandLine: python -m dtool.base –exec-parse_items

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.base import *  # NOQA
>>> from wbia.dtool.example_depcache import DummyVsManyConfig
>>> cfg = DummyVsManyConfig()
>>> param_list = cfg.parse_items()
>>> result = ('param_list = %s' % (ut.repr2(param_list, nl=1),))
>>> print(result)
```

parse_namespace_config_items()
Recursively extracts key, val pairs from Config objects into a flat list. (there must not be name conflicts)

pop_update(*other*)
Updates based on other, while popping off used arguments. (useful for testing if a parameter was unused or misspelled)

Doctest:

```
>>> from wbia.dtool.base import *  # NOQA
>>> from wbia import dtool as dt
>>> cfg = dt.Config.from_dict({'a': 1, 'b': 2, 'c': 3})
>>> other = {'a': 5, 'e': 2}
>>> cfg.pop_update(other)
>>> assert cfg['a'] == 5
>>> assert len(other) == 1 and 'a' not in other
```

setitem(*key*, *value*)
Required for DictLike interface

update(***kwargs*)
Overwrites default DictLike update for only keys that exist. Non-existing key are ignored.

Note: prefixed keys in the form <classname>_<key> will be just be interpreted as <key>

CommandLine: python -m dtool.base update –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.base import *  # NOQA
>>> from wbia.dtool.example_depcache import DummyVsManyConfig
>>> cfg = DummyVsManyConfig()
>>> cfg.update(DummyAlgo_version=4)
>>> print(cfg)
```

update2(**args*, ***kwargs*)
Overwrites default DictLike update for only keys that exist. Non-existing key are ignored. Also updates nested configs.

Note: prefixed keys in the form <classname>_<key> will be just be interpreted as <key>

CommandLine: python -m dtool.base update –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.base import *  # NOQA
>>> from wbia import dtool as dt
>>> cfg = dt.Config.from_dict({
>>>     'a': 1,
>>>     'b': 2,
>>>     'c': 3,
>>>     'sub1': dt.Config.from_dict({
>>>         'x': 'x',
>>>         'y': {'z', 'x'},
>>>         'c': 33,
>>>     }),
>>>     'sub2': dt.Config.from_dict({
```

(continues on next page)

(continued from previous page)

```
>>>         's': [1, 2, 3],
>>>         't': (1, 2, 3),
>>>         'c': 42,
>>>         'sub3': dt.Config.from_dict({
>>>             'b': 99,
>>>             'c': 88,
>>>         }),
>>>     }),
>>> })
>>> kwargs = {'c': 10}
>>> cfg.update2(c=10, y={1,2})
>>> assert cfg.c == 10
>>> assert cfg.sub1.c == 10
>>> assert cfg.sub2.c == 10
>>> assert cfg.sub2.sub3.c == 10
>>> assert cfg.sub1.y == {1, 2}
```

class wbia.dtool.base.**IBEISRequestHacks**

Bases: object

dannots

extern_data_config2

extern_query_config2

get_qreq_annot_nids(*aids*)

ibs

HACK specific to wbia

qannots

class wbia.dtool.base.**MatchResult**(*qaid=None*, *daids=None*, *qnid=None*, *dnid_list=None*, *annot_score_list=None*, *unique_nids=None*, *name_score_list=None*)

Bases: *wbia.dtool.base.AlgoResult*, utool.util_dev.NiceRepr

daids

num_daids

qaids

class wbia.dtool.base.**StackedConfig**(*config_list*)

Bases: utool.util_dict.DictLike, utool.util_class.HashComparable

Manages a list of configurations

get_cfgstr()

getitem(*key*)

keys()

class wbia.dtool.base.**VsManySimilarityRequest**

Bases: *wbia.dtool.base.BaseRequest*, *wbia.dtool.base.AnnotSimiliarity*

Request for one-vs-many simlarity

CommandLine: python -m dtool.base –exec-VsManySimilarityRequest

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.base import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> qaid_list = [1, 2]
>>> daid_list = [2, 3, 4]
>>> depc = testdata_depc()
>>> request = depc.new_request('vsmany', qaid_list, daid_list)
>>> request.ensure_dependencies()
>>> results = request.execute()
>>> # Test dependence on data
>>> request2 = depc.new_request('vsmany', qaid_list + [3], daid_list + [5])
>>> results2 = request2.execute()
>>> print('results = %r' % (results,))
>>> print('results2 = %r' % (results2,))
>>> assert len(results) == 2, 'incorrect num output'
>>> assert len(results2) == 3, 'incorrect num output'
```

get_cfgstr(*with_input=False*, *with_data=True*, *with_pipe=True*, *hash_pipe=False*)
Override default get_cfgstr to show reliance on data

get_input_hashid()

classmethod new(*depc*, *qaid_list*, *daid_list*, *cfgdict=None*, *tablename=None*)

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

class wbia.dtool.base.**VsOneSimilarityRequest**
Bases: *wbia.dtool.base.BaseRequest*, *wbia.dtool.base.AnnotSimiliarity*

Similarity request for pairwise scores

References

https://thingspython.wordpress.com/2010/09/27/ another-super-wrinkle-raising-typeerror/

CommandLine: python -m dtool.base –exec-VsOneSimilarityRequest

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.base import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> qaid_list = [1, 2, 3, 5]
>>> daid_list = [2, 3, 4]
>>> depc = testdata_depc()
>>> request = depc.new_request('vsone', qaid_list, daid_list)
>>> results = request.execute()
>>> # Test that adding a query / data id only recomputes necessary items
>>> request2 = depc.new_request('vsone', qaid_list + [4], daid_list + [5])
>>> results2 = request2.execute()
>>> print('results = %r' % (results,))
>>> print('results2 = %r' % (results2,))
>>> ut.assert_eq(len(results), 10, 'incorrect num output')
>>> ut.assert_eq(len(results2), 16, 'incorrect num output')
```

execute(*parent_rowids=None*, *use_cache=None*, *postprocess=True*, ***kwargs*)
HACKY REIMPLEMENTATION

get_input_hashid()

static make_parent_rowids(*qaid_list*, *daid_list*)

classmethod new(*depc*, *qaid_list*, *daid_list*, *cfgdict=None*, *tablename=None*)

parent_rowids_T

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

wbia.dtool.base.**config_graph_subattrs**(*cfg*, *depc*)

wbia.dtool.base.**from_param_info_list**(*param_info_list*, *tablename='Unnamed'*)

wbia.dtool.base.**make_configclass**(*dict_*, *tablename*)
Creates a custom config class from a dict

wbia.dtool.base.**safeop**(*op_*, *xs*, **args*, ***kwargs*)

1.5.3 wbia.dtool.depcache_control module

implicit version of dependency cache from wbia/templates/template_generator

class wbia.dtool.depcache_control.**DependencyCache**(*controller*, *name*, *get_root_uuid*, *table_name=None*, *root_getters=None*, *use_globals=True*)

Bases: object

check_rowids(*tablename*, *input_tuple*, *config={}*)
Returns a list of flags where True means the row has been computed and False means that it needs to be computed.

clear_all()

close()
Close all managed SQL databases

delete_property(*tablename*, *root_rowids*, *config=None*, *_debug=False*)
Deletes the rowids of *tablename* that correspond to *root_rowids* using *config*.

FIXME: make this work for all configs

delete_property_all(*tablename*, *root_rowids*, *_debug=False*)
Deletes the rowids of *tablename* that correspond to *root_rowids* using *config*.

FIXME: make this work for all configs

delete_root(*root_rowids*, *delete_extern=None*, *_debug=False*, *table_config_filter=None*, *prop=None*)
Deletes all properties of a root object regardless of config

Parameters **root_rowids** (*list*) –

CommandLine: python -m dtool.depcache_control delete_root –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_control import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> exec(ut.execstr_funckw(depc.delete_root), globals())
>>> root_rowids = [1]
>>> depc.delete_root(root_rowids)
>>> depc.get('fgweight', [1])
>>> depc.delete_root(root_rowids)
```

explicit_graph

get(*tablename*, *root_rowids*, *colnames=None*, *config=None*, *ensure=True*, *_debug=None*, *recompute=False*, *recompute_all=False*, *eager=True*, *nInput=None*, *read_extern=True*, *onthefly=False*, *num_retries=3*, *retry_delay_min=1*, *retry_delay_max=3*, *hack_paths=False*)
Access dependant properties the primary objects using primary ids.

Gets the data in *colnames* of *tablename* that correspond to *root_rowids* using *config*. if colnames is None, all columns are returned.

Parameters

- **tablename** (*str*) – table name containing desired property
- **root_rowids** (*List[int]*) – ids of the root object
- **colnames** (*None*) – desired property (default = None)
- **config** (*None*) – (default = None)
- **read_extern** – if False then only returns extern URI
- **hack_paths** – if False then does not compute extern info just returns path that it will be located at

Returns prop_list

Return type list

CommandLine: python -m dtool.depcache_control –exec-get

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_control import *  # NOQA
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> depc = testdata_depc3(True)
>>> exec(ut.execstr_funckw(depc.get), globals())
>>> aids = [1, 2, 3]
>>> tablename = 'labeler'
>>> root_rowids = aids
>>> prop_list = depc.get(
>>>     tablename, root_rowids, colnames)
>>> result = ('prop_list = %s' % (ut.repr2(prop_list),))
>>> print(result)
prop_list = [('labeler([root(1)]:42)',), ('labeler([root(2)]:42)',), (
↪'labeler([root(3)]:42)',)]
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_control import *  # NOQA
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> depc = testdata_depc3(True)
>>> exec(ut.execstr_funckw(depc.get), globals())
>>> aids = [1, 2, 3]
>>> tablename = 'smk_match'
>>> tablename = 'vocab'
>>> table = depc[tablename]
>>> root_rowids = [aids]
>>> prop_list = depc.get(
>>>     tablename, root_rowids, colnames, config)
>>> result = ('prop_list = %s' % (ut.repr2(prop_list),))
>>> print(result)
prop_list = [('vocab([root(1;2;3)]:42)',)]
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_control import *  # NOQA
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> depc = testdata_depc3(True)
>>> exec(ut.execstr_funckw(depc.get), globals())
>>> aids = [1, 2, 3]
>>> depc = testdata_depc()
>>> tablename = 'chip'
>>> table = depc[tablename]
>>> root_rowids = aids
>>> # Ensure chips are computed
>>> prop_list1 = depc.get(tablename, root_rowids)
>>> # Get file paths and delete them
>>> prop_list2 = depc.get(tablename, root_rowids, read_extern=False)
>>> n = ut.remove_file_list(ut.take_column(prop_list2, 1))
>>> assert n == len(prop_list2), 'files were not computed'
>>> prop_list3 = depc.get(tablename, root_rowids)
>>> assert np.all(prop_list1[0][1] == prop_list3[0][1]), 'computed same info'
```

get_allconfig_descendant_rowids(*root_rowids*, *table_config_filter=None*)

get_ancestor_rowids(*tablename*, *native_rowids*, *ancestor_tablename=None*)
ancestor_tablename = depc.root; native_rowids = cid_list; tablename = const.CHIP_TABLE

get_config_history(*tablename*, *root_rowids*, *config=None*)

get_config_trail(*tablename*, *config*)

get_config_trail_str(*tablename*, *config*)

get_db_by_name(*name*)
Get the database (i.e. SQLController) for the given database name

get_dependencies(*tablename*)
gets level dependences from root to tablename

CommandLine: python -m dtool.depcache_control –exec-get_dependencies

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_control import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> tablename = 'fgweight'
>>> result = ut.repr3(depc.get_dependencies(tablename), nl=1)
>>> print(result)
[
    ['dummy_annot'],
    ['chip', 'probchip'],
    ['keypoint'],
    ['fgweight'],
]
```

get_edges(*data=False*)
edges for networkx structure

get_implicit_edges(*data=False*)
Edges defined by subconfigurations

get_native(*tablename*, *tbl_rowids*, *colnames=None*, *_debug=None*, *read_extern=True*)
Gets data using internal ids, which is faster if you have them.

CommandLine: python -m dtool.depcache_control get_native:0 python -m dtool.depcache_control get_native:1

Example

```
>>> # ENABLE_DOCTEST
>>> # Simple test of get native
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> config = {}
>>> depc = testdata_depc()
>>> tablename = 'keypoint'
>>> aids = [1,]
>>> tbl_rowids = depc.get_rowids(tablename, aids, config=config)
>>> data = depc.get_native(tablename, tbl_rowids)
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> depc = testdata_depc()
>>> config = {}
>>> tablename = 'chip'
>>> colnames = extern_colname = 'chip'
>>> aids = [1, 2]
>>> depc.delete_property(tablename, aids, config=config)
>>> # Ensure chip rowids exist then delete external data without
>>> # notifying the depcache. This forces the depcache to recover
>>> tbl_rowids = chip_rowids = depc.get_rowids(tablename, aids,␣
↪config=config)
>>> data_fpaths = depc.get(tablename, aids, extern_colname, config=config,␣
↪read_extern=False)
```

(continues on next page)

(continued from previous page)

```
>>> ut.remove_file_list(data_fpaths)
>>> chips = depc.get_native(tablename, tbl_rowids, extern_colname)
>>> print('chips = %r' % (chips,))
```

get_native_property(*tablename*, *tbl_rowids*, *colnames=None*, *_debug=None*, *read_extern=True*)

Gets data using internal ids, which is faster if you have them.

CommandLine: python -m dtool.depcache_control get_native:0 python -m dtool.depcache_control get_native:1

Example

```
>>> # ENABLE_DOCTEST
>>> # Simple test of get native
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> config = {}
>>> depc = testdata_depc()
>>> tablename = 'keypoint'
>>> aids = [1,]
>>> tbl_rowids = depc.get_rowids(tablename, aids, config=config)
>>> data = depc.get_native(tablename, tbl_rowids)
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> depc = testdata_depc()
>>> config = {}
>>> tablename = 'chip'
>>> colnames = extern_colname = 'chip'
>>> aids = [1, 2]
>>> depc.delete_property(tablename, aids, config=config)
>>> # Ensure chip rowids exist then delete external data without
>>> # notifying the depcache. This forces the depcache to recover
>>> tbl_rowids = chip_rowids = depc.get_rowids(tablename, aids,␣
↪config=config)
>>> data_fpaths = depc.get(tablename, aids, extern_colname, config=config,␣
↪read_extern=False)
>>> ut.remove_file_list(data_fpaths)
>>> chips = depc.get_native(tablename, tbl_rowids, extern_colname)
>>> print('chips = %r' % (chips,))
```

get_parent_rowids(*target_tablename*, *input_tuple*, *config=None*, ***kwargs*)

Returns the parent rowids needed to get / compute a property of tablename

Parameters **input_tuple** – to be explicit send in as a tuple of lists. Each list corresponds to parent information needed by expanded rmis (root most input).

Each item in the tuple correponds a root most node, and should be specified as a list of inputs. For single items this is a scalar, for multi-items it is a list.

For example if you have a property like a chip that depends on only one parent, then to get the chips for the first N annotations your list input tuple is:

input_tuple = ([1, 2, 3, . . . , N],)

For a single multi inputs: If you want to get two vocabs for even and odd annots then you have:

([[0, 2, 4, . . .], [1, 3, 5, . . .]],)

For a single comparasion version multi inputs: If you want to query the first N annotats against two vocabs then you have:

([1, 2, 3, . . . , N], [[0, 2, 4, . . .], [1, 3, 5, . . .]],)

(Note this only works if broadcasting is on)

get_property(*tablename*, *root_rowids*, *colnames=None*, *config=None*, *ensure=True*, *_debug=None*, *recompute=False*, *recompute_all=False*, *eager=True*, *nInput=None*, *read_extern=True*, *onthefly=False*, *num_retries=3*, *retry_delay_min=1*, *retry_delay_max=3*, *hack_paths=False*)

Access dependant properties the primary objects using primary ids.

Gets the data in *colnames* of *tablename* that correspond to *root_rowids* using *config*. if colnames is None, all columns are returned.

Parameters

- **tablename** (*str*) – table name containing desired property
- **root_rowids** (*List[int]*) – ids of the root object
- **colnames** (*None*) – desired property (default = None)
- **config** (*None*) – (default = None)
- **read_extern** – if False then only returns extern URI
- **hack_paths** – if False then does not compute extern info just returns path that it will be located at

Returns prop_list

Return type list

CommandLine: python -m dtool.depcache_control –exec-get

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_control import *  # NOQA
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> depc = testdata_depc3(True)
>>> exec(ut.execstr_funckw(depc.get), globals())
>>> aids = [1, 2, 3]
>>> tablename = 'labeler'
>>> root_rowids = aids
>>> prop_list = depc.get(
>>>     tablename, root_rowids, colnames)
>>> result = ('prop_list = %s' % (ut.repr2(prop_list),))
>>> print(result)
prop_list = [('labeler([root(1)]:42)',), ('labeler([root(2)]:42)',), (
↪'labeler([root(3)]:42)',)]
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_control import *  # NOQA
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> depc = testdata_depc3(True)
>>> exec(ut.execstr_funckw(depc.get), globals())
>>> aids = [1, 2, 3]
>>> tablename = 'smk_match'
>>> tablename = 'vocab'
>>> table = depc[tablename]
>>> root_rowids = [aids]
>>> prop_list = depc.get(
>>>     tablename, root_rowids, colnames, config)
>>> result = ('prop_list = %s' % (ut.repr2(prop_list),))
>>> print(result)
prop_list = [('vocab([root(1;2;3)]:42)',)]
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_control import *  # NOQA
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> depc = testdata_depc3(True)
>>> exec(ut.execstr_funckw(depc.get), globals())
>>> aids = [1, 2, 3]
>>> depc = testdata_depc()
>>> tablename = 'chip'
>>> table = depc[tablename]
>>> root_rowids = aids
>>> # Ensure chips are computed
>>> prop_list1 = depc.get(tablename, root_rowids)
>>> # Get file paths and delete them
>>> prop_list2 = depc.get(tablename, root_rowids, read_extern=False)
>>> n = ut.remove_file_list(ut.take_column(prop_list2, 1))
>>> assert n == len(prop_list2), 'files were not computed'
>>> prop_list3 = depc.get(tablename, root_rowids)
>>> assert np.all(prop_list1[0][1] == prop_list3[0][1]), 'computed same info'
```

get_root_rowids(*tablename*, *native_rowids*)

Parameters

- **tablename** (*str*) –
- **native_rowids** (*list*) –

Returns

Return type list

CommandLine: python -m dtool.depcache_control get_root_rowids –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> depc = testdata_depc()
>>> config1 = {'adapt_shape': False}
>>> config2 = {'adapt_shape': True}
>>> root_rowids = [2, 3, 5, 7]
>>> native_rowids1 = depc.get_rowids('keypoint', root_rowids, config=config1)
>>> native_rowids2 = depc.get_rowids('keypoint', root_rowids, config=config2)
>>> ancestor_rowids1 = list(depc.get_root_rowids('keypoint', native_rowids1))
>>> ancestor_rowids2 = list(depc.get_root_rowids('keypoint', native_rowids2))
>>> assert native_rowids1 != native_rowids2, 'should have different native␣
↪rowids'
>>> assert ancestor_rowids1 == root_rowids, 'should have same root'
>>> assert ancestor_rowids2 == root_rowids, 'should have same root'
```

get_rowids (*tablename*, *input_tuple*, ***rowid_kw*)

Used to get tablename rowids. Ensures rows exist unless ensure=False. rowids uniquely specify parent inputs and a configuration.

CommandLine: python -m dtool.depcache_control get_rowids –show python -m dtool.depcache_control get_rowids:1

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_control import *  # NOQA
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> depc = testdata_depc3(True)
>>> exec(ut.execstr_funckw(depc.get), globals())
>>> kwargs = {}
>>> root_rowids = [1, 2, 3]
>>> root_rowids2 = [(4, 5, 6, 7)]
>>> root_rowids3 = root_rowids2
>>> tablename = 'smk_match'
>>> input_tuple = (root_rowids, root_rowids2, root_rowids3)
>>> target_table = depc[tablename]
>>> inputs = target_table.rootmost_inputs.total_expand()
>>> depc.get_rowids(tablename, input_tuple)
>>> depc.print_all_tables()
```

Example

```
>>> # ENABLE_DOCTEST
>>> # Test external / ensure getters
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> config = {}
>>> depc = testdata_depc()
>>> aids = [1,]
>>> depc.delete_property('keypoint', aids, config=config)
>>> chip_fpaths = depc.get('chip', aids, 'chip', config=config, read_
↪extern=False)
>>> ut.remove_file_list(chip_fpaths)
```

(continues on next page)

(continued from previous page)

```
>>> rowids = depc.get_rowids('keypoint', aids, ensure=True, config=config)
>>> print('rowids = %r' % (rowids,))
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> depc = testdata_depc()
>>> depc.clear_all()
>>> root_rowids = [1, 2]
>>> config = {}
>>> # Recompute the first few, make sure the rowids do not change
>>> _ = depc.get_rowids('chip', root_rowids + [3], config=config)
>>> assert _ == [1, 2, 3]
>>> initial_rowids = depc.get_rowids('chip', root_rowids, config=config)
>>> recomp_rowids = depc.get_rowids('chip', root_rowids, config=config,␣
↪recompute=True)
>>> assert recomp_rowids == initial_rowids, 'rowids should not change due to␣
↪recompute'
```

get_tablenames()

get_uuids(*tablename*, *root_rowids*, *config=None*)
TODO: Make uuids for dependant object based on root uuid and path of # construction.

graph

initialize(*_debug=None*)
Creates all registered tables

make_graph(***kwargs*)
Constructs a networkx representation of the dependency graph

CommandLine: python -m dtool –tf DependencyCache.make_graph –show –reduced

python -m wbia.control.IBEISControl show_depc_annot_graph –show –reduced

python -m wbia.control.IBEISControl show_depc_annot_graph –show –reduced –testmode python -m wbia.control.IBEISControl show_depc_annot_graph –show –testmode

python -m wbia.control.IBEISControl –test-show_depc_image_graph –show –reduced python -m wbia.control.IBEISControl –test-show_depc_image_graph –show

python -m wbia.scripts.specialdraw double_depcache_graph –show –testmode

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_control import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> import utool as ut
>>> depc = testdata_depc()
>>> graph = depc.make_graph(reduced=ut.get_argflag('--reduced'))
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.ensureqt()
```

(continues on next page)

(continued from previous page)

```
>>> import networkx as nx
>>> #pt.show_nx(nx.dag.transitive_closure(graph))
>>> #pt.show_nx(ut.nx_transitive_reduction(graph))
>>> pt.show_nx(graph)
>>> pt.show_nx(graph, layout='agraph')
>>> ut.show_if_requested()
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_control import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> import utool as ut
>>> depc = testdata_depc()
>>> graph = depc.make_graph(reduced=True)
>>> # xdoctest: +REQUIRES(--show)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.ensureqt()
>>> import networkx as nx
>>> #pt.show_nx(nx.dag.transitive_closure(graph))
>>> #pt.show_nx(ut.nx_transitive_reduction(graph))
>>> pt.show_nx(graph)
>>> pt.show_nx(graph, layout='agraph')
>>> ut.show_if_requested()
```

make_root_info_uuid(*root_rowids*, *info_props*)

Creates a uuid that depends on certain properties of the root object. This is used for implicit cache invalidation because, if those properties change then this uuid also changes.

The depcache needs to know about stateful properties of dynamic root objects in order to correctly compute their hashes.

```
>>> #ibs = wbia.opendb(defaultdb='testdb1')
>>> root_rowids = ibs._get_all_aids()
>>> depc = ibs.depc_annot
>>> info_props = ['image_uuid', 'verts', 'theta']
>>> info_props = ['image_uuid', 'verts', 'theta', 'name', 'species', 'yaw']
```

new_request(*tablename*, *qaids*, *daids*, *cfgdict=None*)

creates a request for data that can be executed later

notify_root_changed(*root_rowids*, *prop*, *force_delete=False*)

this is where we are notified that a "registered" root property has changed.

print_all_tables()

print_config_tables()

print_schemas()

print_table(*tablename*)

rectify_input_tuple(*exi_inputs*, *input_tuple*)

Standardizes inputs allowed for convinience into the expected input for get_parent_rowids.

reduced_graph

register_delete_table_exclusion(*tablename*, *prop*)

register_preproc(**args*, ***kwargs*)

Decorator for registration of cachables

Parameters

- **tablename** (*str*) – name of the node (corrsponds to SQL table)
- **parents** (*list*) – tables this node depends on
- **colnames** (*list*) – data returned by this table
- **coltypes** (*list*) – types of data returned by this table
- **chunksize** (*int*) – (default = None)
- **configclass** (*dtool.TableConfig*) – derivative of dtool.TableConfig. if None, a default class will be constructed for you. (default = None)
- **docstr** (*str*) – (default = None)
- **fname** (*str*) – file name(default = None)
- **asobject** (*bool*) – hacky dont use (default = False)

SeeAlso: depcache_table.DependencyCacheTable

root

show_graph(*reduced=False*, ***kwargs*)
Helper "fluff" function

stacked_config(*source*, *dest*, *config*)

CommandLine: python -m dtool.depcache_control stacked_config –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_control import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> source = depc.root
>>> dest = 'fgweight'
>>> config = {}
>>> stacked_config = depc.stacked_config(source, dest, config)
>>> cfgstr = stacked_config.get_cfgstr()
>>> result = ('cfgstr = %s' % (ut.repr2(cfgstr),))
>>> print(result)
```

tablenames

tables

wbia.dtool.depcache_control.**check_register**(*args*, *kwargs*)

wbia.dtool.depcache_control.**make_depcache_decors**(*root_tablename*)
Makes global decorators to register functions for a tablename.

A preproc function is meant to belong only to a single parent An algo function belongs to the root node, and may depend on a set of root nodes rather than just a single one.

1.5.4 wbia.dtool.depcache_table module

Module contining DependencyCacheTable

python -m dtool.depcache_control –exec-make_graph –show python -m dtool.depcache_control –exec-make_graph –show –reduce

FIXME:

RECTIFY: ismulti / ismodel need to be rectified. This indicate that this table recieves multiple inputs from at least one parent table.

RECTIFY: Need to standardize parent rowids -vs- parent args. in one-to-one cases they are the same. In multi cases the rowids indicate a uuid and the args are the saved set of rowids that exist in the manifest.

RECTIFY: is rowid_list row-major or column-major? I think currently rowid_list is row-major and rowid_listT is column-major but this may not be consistent.

class wbia.dtool.depcache_table.**DependencyCacheTable**(*depc=None*, *parent_tablenames=None*, *tablename=None*, *data_colnames=None*, *data_coltypes=None*, *preproc_func=None*, *docstr='no docstr'*, *fname=None*, *asobject=False*, *chunksize=None*, *isinteractive=False*, *default_to_unpack=False*, *default_onthefly=False*, *rm_extern_on_delete=False*, *vectorized=True*, *taggable=False*)

Bases: wbia.dtool.depcache_table._TableGeneralHelper, wbia.dtool.depcache_table._TableInternalSetup, wbia.dtool.depcache_table._TableDebugHelper, wbia.dtool.depcache_table._TableComputeHelper, wbia.dtool.depcache_table._TableConfigHelper

An individual node in the dependency graph.

All SQL column information is stored in: internal_col_attrs - keeps track of internal info

Additional metadata about specific columns is stored in parent_col_attrs - keeps track of parent info data_col_attrs - keeps track of computed data

db
pointer to underlying database

Type dtool.SQLDatabaseController

depc
pointer to parent cache

Type dtool.DependencyCache

tablename
name of the table

Type str

docstr
documentation for table

Type str

parent_tablenames
parent tables in depcache

Type str

data_colnames
columns produced by preproc_func

Type List[str]

data_coltypes
column SQL types produced by preproc_func

Type List[str]

preproc_func
worker function

Type func

vectorized
by defaults it is assumed registered functions can process multiple inputs at once.

Type bool

taggable
specifies if a computed object can be disconected from its ancestors and accessed via a tag.

Type bool

CommandLine: python -m dtool.depcache_table –exec-DependencyCacheTable

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_table import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> print(depc['vsmany'])
>>> print(depc['spam'])
>>> print(depc['vsone'])
>>> print(depc['nnindexer'])
```

clear_table()
Deletes all data in this table

delete_rows(*rowid_list*, *delete_extern=None*, *dry=False*, *verbose=None*)

CommandLine: python -m dtool.depcache_table –exec-delete_rows

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_table import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> #table = depc['keypoint']
>>> table = depc['chip']
>>> exec(ut.execstr_funckw(table.delete_rows), globals())
```

(continues on next page)

(continued from previous page)

```
>>> tablename = table.tablename
>>> graph = depc.explicit_graph
>>> config1 = None
>>> config2 = table.configclass(version=-1)
>>> config3 = table.configclass(version=-1, ext='.jpg')
>>> config4 = table.configclass(ext='.jpg')
>>> # Create several configs of rowid
>>> aids = [1, 2, 3]
>>> depc.get_rowids('spam', aids, config=config1)
>>> depc.get_rowids('spam', aids, config=config2)
>>> depc.get_rowids('spam', aids, config=config3)
>>> depc.get_rowids('spam', aids, config=config4)
>>> # Delete the png configs
>>> rowid_list1 = depc.get_rowids(table.tablename, aids,
>>>                               config=config2)
>>> rowid_list2 = depc.get_rowids(table.tablename, aids,
>>>                               config=config1)
>>> rowid_list = rowid_list1 + rowid_list2
>>> assert len(ut.setintersect_ordered(rowid_list1, rowid_list2)) == 0
>>> table.delete_rows(rowid_list)
```

ensure_rows(*parent_ids_*, *preproc_args*, *config=None*, *verbose=True*, *_debug=None*, *retry=3*, *retry_delay_min=1*, *retry_delay_max=10*)

Lazy addition

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_table import *  # NOQA
>>> from wbia.dtool.example_depcache2 import testdata_depc3
>>> depc = testdata_depc3()
>>> table = depc['vsone']
>>> exec(ut.execstr_funckw(table.get_rowid), globals())
>>> config = table.configclass()
>>> verbose = True
>>> # test duplicate inputs are detected and accounted for
>>> parent_rowids = [(i, i) for i in list(range(100))] * 100
>>> rectify_tup = table._rectify_ids(parent_rowids)
>>> (parent_ids_, preproc_args, idxs1, idxs2) = rectify_tup
>>> rowids = table.ensure_rows(parent_ids_, preproc_args, config=config)
>>> result = ('rowids = %r' % (rowids,))
>>> print(result)
```

export_rows(*rowid*, *target*)

The goal of this is to export taggable data that can be used independantly of its dependant features.

TODO List:

- **Gather information about columns**
  - **Native and (localized) external data**
    - <table>_rowid - non-transferable
    - Parent UUIDS - non-transferable
    - config rowid - non-transferable

        * model_uuid -
        * augment_bit - transferable - trivial
        * words_extern_uri - copy to destination
        * feat_setsize - transferable - trivial
        * model_tag
      - **Should also gather info from manifest:**
        * feat_setuuid_primary_ids - non-transferable
        * feat_setuuid_model_input - non-transferable
      - Should gather exhaustive config history
    - Save to disk
    - Add function to reload data in exported format
    - Getters should be able to specify a tag inplace of the root input

    for the tagged. Additionally native root-ids should also be allowed.

  rowid = 1

fname
  Backwards compatible name of the database this Table belongs to

classmethod from_name(*db_name*, *table_name*, *depcache_controller*, *parent_tablenames=None*, *data_colnames=None*, *data_coltypes=None*, *preproc_func=None*, *docstr='no docstr'*, *asobject=False*, *chunksize=None*, *default_to_unpack=False*, *rm_extern_on_delete=False*, *vectorized=True*, *taggable=False*)
  Build the instance based on a database and table name.

get_internal_columns(*tbl_rowids*, *colnames=None*, *eager=True*, *nInput=None*, *unpack_scalars=True*, *keepwrap=False*, *showprog=False*)
  Access data in this table using the table PRIMARY KEY rowids (not depc PRIMARY ids)

get_row_data(*tbl_rowids*, *colnames=None*, *_debug=None*, *read_extern=True*, *num_retries=1*, *eager=True*, *nInput=None*, *ensure=True*, *delete_on_fail=True*, *showprog=False*, *unpack_columns=None*)
  FIXME: unpacking is confusing with sql controller TODO: Clean up and allow for eager=False

  colnames = ('mask', 'size')

  CommandLine: python -m dtool.depcache_table –test-get_row_data:0 python -m dtool.depcache_table –test-get_row_data:1

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_table import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> table = depc['chip']
>>> exec(ut.execstr_funckw(table.get_row_data), globals())
>>> tbl_rowids = depc.get_rowids('chip', [1, 2, 3], _debug=True,␣
↪recompute=True)
>>> colnames = ('size_1', 'size', 'chip' + EXTERN_SUFFIX, 'chip')
```

(continues on next page)

(continued from previous page)

```
>>> kwargs = dict(read_extern=True, num_retries=1, _debug=True)
>>> prop_list = table.get_row_data(tbl_rowids, colnames, **kwargs)
>>> prop_list0 = ut.take_column(prop_list, [0, 1, 2]) # data subset
>>> result = (ut.repr2(prop_list0, nl=1))
>>> print(result)
>>> #_debug, num_retries, read_extern = True, 1, True
>>> prop_gen = table.get_row_data(tbl_rowids, colnames, eager=False)
>>> prop_list2 = list(prop_gen)
>>> assert len(prop_list2) == len(prop_list), 'inconsistent lens'
>>> assert all([ut.lists_eq(prop_list2[1], prop_list[1]) for x in␣
↪range(len(prop_list))]), 'inconsistent vals'
>>> chips = table.get_row_data(tbl_rowids, 'chip', eager=False)
```

[[2453, (1707, 2453), 'chip_chip_id=1_pyrappzicqoskdjq.png'], [250, (300, 250), 'chip_chip_id=2_pyrappzicqoskdjq.png'], [372, (545, 372), 'chip_chip_id=3_pyrappzicqoskdjq.png'],

]

Example

```
>>> # ENABLE_DOCTEST
>>> # Test external / ensure getters
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> depc = testdata_depc()
>>> table = depc['chip']
>>> exec(ut.execstr_funckw(table.get_row_data), globals())
>>> depc.clear_all()
>>> config = {}
>>> aids = [1,]
>>> read_extern = False
>>> tbl_rowids = depc.get_rowids('chip', aids, config=config)
>>> data_fpaths = depc.get('chip', aids, 'chip', config=config, read_
↪extern=False)
>>> # Ensure data is recomputed if an external file is missing
>>> ut.remove_fpaths(data_fpaths)
>>> data = table.get_row_data(tbl_rowids, 'chip', read_extern=False,␣
↪ensure=False)
>>> data = table.get_row_data(tbl_rowids, 'chip', read_extern=False,␣
↪ensure=True)
```

get_rowid(*parent_rowids*, *config=None*, *ensure=True*, *eager=True*, *nInput=None*, *recompute=False*, *_debug=None*, *num_retries=1*)

Returns the rowids of derived properties. If they do not exist it computes them.

Parameters

- **parent_rowids** (*list*) – list of tuples with the parent rowids as the value of each tuple
- **config** (*None*) – (default = None)
- **ensure** (*bool*) – eager evaluation if True (default = True)
- **eager** (*bool*) – (default = True)
- **nInput** (*int*) – (default = None)

- **recompute** (*bool*) – (default = False)
- **_debug** (*None*) – (default = None) deprecated; no-op

Returns rowid_list

Return type list

CommandLine: python -m dtool.depcache_table –exec-get_rowid

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.depcache_table import *  # NOQA
>>> from wbia.dtool.example_depcache2 import testdata_depc3
>>> depc = testdata_depc3()
>>> table = depc['labeler']
>>> exec(ut.execstr_funckw(table.get_rowid), globals())
>>> config = table.configclass()
>>> parent_rowids = list(zip([1, None, None, 2]))
>>> rowids = table.get_rowid(parent_rowids, config=config)
>>> result = ('rowids = %r' % (rowids,))
>>> print(result)
rowids = [1, None, None, 2]
```

initialize(*_debug=None*)
Ensures the SQL schema for this cache table

number_of_rows

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

class wbia.dtool.depcache_table.**ExternType**(*read_func*, *write_func*, *extern_ext=None*, *extkey=None*)
Bases: ubelt.util_mixins.NiceRepr

Type to denote an external resource not saved in an SQL table

exception wbia.dtool.depcache_table.**ExternalStorageException**(**args*, ***kwargs*)
Bases: Exception

Indicates a missing external file

exception wbia.dtool.depcache_table.**TableOutOfSyncError**(*db*, *tablename*, *extended_msg*)
Bases: Exception

Raised when the code's table definition doesn't match the defition in the database

wbia.dtool.depcache_table.**ensure_config_table**(*db*)
SQL definition of configuration table.

wbia.dtool.depcache_table.**make_extern_io_funcs**(*table*, *cls*)
Hack in read/write defaults for pickleable classes

wbia.dtool.depcache_table.**predrop_grace_period**(*tablename*, *seconds=None*)
Hack that gives the user some time to abort deleting everything

1.5.5 wbia.dtool.example_depcache module

CommandLine: python -m dtool.example_depcache –exec-dummy_example_depcacahe –show python -m dtool.depcache_control –exec-make_graph –show

class wbia.dtool.example_depcache.**DummyAnnotMatch**(*qaid=None*, *daids=None*, *qnid=None*, *dnid_list=None*, *annot_score_list=None*, *unique_nids=None*, *name_score_list=None*)

Bases: *wbia.dtool.base.MatchResult*

class wbia.dtool.example_depcache.**DummyChipConfig**(***kwargs*)

Bases: *wbia.dtool.base.Config*

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> cfg = DummyChipConfig()
>>> cfg.dim_size = 700
>>> cfg.histeq = True
>>> print(cfg)
>>> cfg.histeq = False
>>> print(cfg)
```

class wbia.dtool.example_depcache.**DummyController**(*cache_dpath*)

Bases: object

Just enough (IBEIS) controller to make the dependency cache examples work

get_cachedir()

make_cache_db_uri(*name*)

class wbia.dtool.example_depcache.**DummyIndexerConfig**(***kwargs*)

Bases: *wbia.dtool.base.Config*

class wbia.dtool.example_depcache.**DummyKptsConfig**(***kwargs*)

Bases: *wbia.dtool.base.Config*

get_param_info_list()

class wbia.dtool.example_depcache.**DummyNNConfig**(***kwargs*)

Bases: *wbia.dtool.base.Config*

get_param_info_list()

class wbia.dtool.example_depcache.**DummySVERConfig**(***kwargs*)

Bases: *wbia.dtool.base.Config*

class wbia.dtool.example_depcache.**DummyVsManyConfig**(***kwargs*)

Bases: *wbia.dtool.base.Config*

class wbia.dtool.example_depcache.**DummyVsManyRequest**

Bases: *wbia.dtool.base.VsManySimilarityRequest*

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> algo_config = DummyVsManyConfig()
>>> print(algo_config)
```

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

class wbia.dtool.example_depcache.**DummyVsOneConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*

get_param_info_list()

get_sub_config_list()

class wbia.dtool.example_depcache.**DummyVsOneMatch**
Bases: *wbia.dtool.base.AlgoResult*, utool.util_dev.NiceRepr

class wbia.dtool.example_depcache.**DummyVsOneRequest**
Bases: *wbia.dtool.base.VsOneSimilarityRequest*

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

class wbia.dtool.example_depcache.**ProbchipConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*
CommandLine: python -m dtool.example_depcache –exec-ProbchipConfig –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.dtool.depcache_control import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> table = depc['probchip']
>>> exec(ut.execstr_funckw(table.get_rowid), globals())
>>> config = table.configclass(testerror=True)
>>> root_rowids = [1, 2, 3]
>>> parent_rowids = list(zip(root_rowids))
>>> proptup_gen = list(table.preproc_func(depc, root_rowids, config))
>>> pc_rowids = depc.get_rowids('probchip', root_rowids, config)
>>> prop_list2 = depc.get('probchip', root_rowids, config=config, read_
↪extern=False)
>>> print(prop_list2)
>>> #depc.new_request('probchip', [1, 2, 3])
>>> fg_rowids = depc.get_rowids('fgweight', root_rowids, config)
>>> fg = depc.get('fgweight', root_rowids, config=config)
>>> ############
>>> config = table.configclass(testerror=False)
>>> root_rowids = [1, 2, 3]
>>> parent_rowids = list(zip(root_rowids))
>>> proptup_gen = list(table.preproc_func(depc, root_rowids, config))
>>> pc_rowids2 = depc.get_rowids('probchip', root_rowids, config)
>>> prop_list2 = depc.get('probchip', root_rowids, config=config, read_
↪extern=False)
>>> print(prop_list2)
```

(continues on next page)

(continued from previous page)

```
>>> #depc.new_request('probchip', [1, 2, 3])
>>> fg_rowids2 = depc.get_rowids('fgweight', root_rowids, config)
```

wbia.dtool.example_depcache.**dummy_example_depcacahe**()
CommandLine: python -m dtool.example_depcache –exec-dummy_example_depcacahe

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.example_depcache import *  # NOQA
>>> depc = dummy_example_depcacahe()
```

wbia.dtool.example_depcache.**example_getter_methods**(*depc*, *tablename*, *root_rowids*)
example of different ways to get data

wbia.dtool.example_depcache.**test_getters**(*depc*)

wbia.dtool.example_depcache.**testdata_depc**(*fname=None*)
Example of local registration

1.5.6 wbia.dtool.example_depcache2 module

wbia.dtool.example_depcache2.**depc_34_helper**(*depc*)

wbia.dtool.example_depcache2.**testdata_custom_annot_depc**(*dummy_dependencies*, *in_memory=True*)

wbia.dtool.example_depcache2.**testdata_depc3**(*in_memory=True*)
Example of local registration
CommandLine: python -m dtool.example_depcache2 testdata_depc3 –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> depc = testdata_depc3()
>>> data = depc.get('labeler', [1, 2, 3], 'data', _debug=True)
>>> data = depc.get('indexer', [[1, 2, 3]], 'data', _debug=True)
>>> depc.print_all_tables()
>>> # xdoctest: +REQUIRES(--show)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> depc.show_graph()
>>> from wbia.plottool.interactions import ExpandableInteraction
>>> inter = ExpandableInteraction(nCols=2)
>>> depc['smk_match'].show_input_graph(inter)
>>> depc['vsone'].show_input_graph(inter)
>>> #depc['vocab'].show_input_graph(inter)
>>> depc['neighbs'].show_input_graph(inter)
>>> inter.start()
>>> #depc['viewpoint_classification'].show_input_graph()
>>> ut.show_if_requested()
```

wbia.dtool.example_depcache2.**testdata_depc4**(*in_memory=True*)
Example of local registration
CommandLine: python -m dtool.example_depcache2 testdata_depc4 –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> depc = testdata_depc4()
>>> #data = depc.get('labeler', [1, 2, 3], 'data', _debug=True)
>>> #data = depc.get('indexer', [[1, 2, 3]], 'data', _debug=True)
>>> depc.print_all_tables()
>>> # xdoctest: +REQUIRES(--show)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> depc.show_graph()
>>> from wbia.plottool.interactions import ExpandableInteraction
>>> inter = ExpandableInteraction(nCols=2)
>>> depc['smk_match'].show_input_graph(inter)
>>> depc['vsone'].show_input_graph(inter)
>>> depc['vocab'].show_input_graph(inter)
>>> depc['neighbs'].show_input_graph(inter)
>>> inter.start()
>>> #depc['viewpoint_classification'].show_input_graph()
>>> ut.show_if_requested()
```

1.5.7 wbia.dtool.input_helpers module

class wbia.dtool.input_helpers.**BranchId**(*accum_ids*, *k*, *parent_colx*)
Bases: utool.util_class.HashComparable

class wbia.dtool.input_helpers.**ExiNode**(*node_id*, *branch_id*)
Bases: utool.util_class.HashComparable

Expanded Input Node

helps distinguish nodes and branch_ids

branch_id

node_id

class wbia.dtool.input_helpers.**RootMostInput**(*node*, *sink*, *exi_graph*)
Bases: utool.util_class.HashComparable

compute_order()
Returns order of computation from this input node to the sink

ismulti

parent_level()
Returns rootmost inputs above this node

Example

```
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> depc = testdata_depc4()
>>> inputs = depc['smk_match'].rootmost_inputs
>>> rmi = inputs.rmi_list[1]
>>> assert len(rmi.parent_level()) == 2
```

class wbia.dtool.input_helpers.**TableInput**(*rmi_list*, *exi_graph*, *table*, *reorder=False*)

Bases: utool.util_dev.NiceRepr

Specifies a set of inputs that can validly compute the output of a table in the dependency graph

exi_nodes()

expand_input(*index*, *inplace=False*)

Pushes the rootmost inputs all the way up to the sources of the graph

CommandLine: python -m dtool.input_helpers expand_input

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.input_helpers import *  # NOQA
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> depc = testdata_depc4()
>>> inputs = depc['smk_match'].rootmost_inputs
>>> inputs = depc['neighbs'].rootmost_inputs
>>> print('(pre-expand)  inputs  = %r' % (inputs,))
>>> index = 'indexer'
>>> inputs2 = inputs.expand_input(index)
>>> print('(post-expand) inputs2 = %r' % (inputs2,))
>>> assert 'indexer' in str(inputs), 'missing indexer1'
>>> assert 'indexer' not in str(inputs2), (
>>>     '(2) unexpected indexer in %s' % (inputs2,))
```

expected_input_depth()

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.dtool.input_helpers import *  # NOQA
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> depc = testdata_depc4()
>>> inputs = depc['neighbs'].rootmost_inputs
>>> index = 'indexer'
>>> inputs = inputs.expand_input(index)
>>> size = inputs.expected_input_depth()
>>> print('size = %r' % (size,))
>>> inputs = depc['feat'].rootmost_inputs
>>> size = inputs.expected_input_depth()
>>> print('size = %r' % (size,))
```

flat_compute_order()

This is basically the scheduler

Todo: We need to verify the correctness of this logic. It seems to not be deterministic between versions of python.

CommandLine: python -m dtool.input_helpers flat_compute_order

Example

```
>>> # xdoctest: +REQUIRES(--fixme)
>>> from wbia.dtool.input_helpers import *  # NOQA
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> depc = testdata_depc4()
>>> inputs = depc['feat'].rootmost_inputs.total_expand()
>>> flat_compute_order = inputs.flat_compute_order()
>>> result = ut.repr2(flat_compute_order)
...
>>> print(result)
[chip[t, t:1, 1:1], probchip[t, t:1, 1:1], feat[t, t:1]]
```

flat_compute_rmi_edges()

Defines order of computation that maps input_ids to target_ids.

CommandLine: python -m dtool.input_helpers flat_compute_rmi_edges

Returns

compute_edges

Each item is a tuple of input/output RootMostInputs ([parent_1, ..., parent_n], node_i)

All parents should be known before you reach the i-th item in the list. Results of the the i-th item may be used in subsequent item computations.

Return type list

Example

```
>>> from wbia.dtool.input_helpers import *  # NOQA
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> depc =testdata_custom_annot_depc([
...    dict(tablename='chips', parents=['annot']),
...    dict(tablename='Notch_Tips', parents=['annot']),
...    dict(tablename='Cropped_Chips', parents=['chips', 'Notch_Tips']),
... ])
>>> table = depc['Cropped_Chips']
>>> inputs = exi_inputs = table.rootmost_inputs.total_expand()
>>> compute_rmi_edges = exi_inputs.flat_compute_rmi_edges()
>>> input_rmis = compute_rmi_edges[-1][0]
>>> result = ut.repr2(input_rmis)
>>> print(result)
[chips[t, t:1, 1:1], Notch_Tips[t, t:1, 1:1]]
```

is_single_inputs()

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

show_exi_graph(*inter=None*)

CommandLine: python -m dtool.input_helpers TableInput.show_exi_graph –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.dtool.input_helpers import *  # NOQA
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> depc = testdata_depc3()
>>> # table = depc['smk_match']
>>> table = depc['neighbs']
>>> inputs = table.rootmost_inputs
>>> print('inputs = %r' % (inputs,))
>>> import wbia.plottool as pt
>>> from wbia.plottool.interactions import ExpandableInteraction
>>> inter = ExpandableInteraction(nCols=1)
>>> inputs.show_exi_graph(inter=inter)
>>> # FIXME; Expanding inputs can overspecify inputs
>>> #inputs = inputs.expand_input(2)
>>> #print('inputs = %r' % (inputs,))
>>> #inputs.show_exi_graph(inter=inter)
>>> #inputs = inputs.expand_input(1)
>>> #inputs = inputs.expand_input(3)
>>> #inputs = inputs.expand_input(2)
>>> #inputs = inputs.expand_input(2)
>>> #inputs = inputs.expand_input(1)
>>> #print('inputs = %r' % (inputs,))
>>> #inputs.show_exi_graph(inter=inter)
>>> inter.start()
>>> ut.show_if_requested()
```

total_expand()

wbia.dtool.input_helpers.**get_rootmost_inputs**(*exi_graph*, *table*)

CommandLine: python -m dtool.input_helpers get_rootmost_inputs –show

Parameters

- **exi_graph** (*nx.Graph*) – made from make_expanded_input_graph(graph, target)
- **table** (*dtool.Table*) –

CommandLine: python -m dtool.input_helpers get_rootmost_inputs

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.input_helpers import *  # NOQA
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> depc = testdata_depc3()
>>> tablename = 'smk_match'
>>> table = depc[tablename]
>>> exi_graph = table.expanded_input_graph
```

(continues on next page)

(continued from previous page)

```
>>> inputs_ = get_rootmost_inputs(exi_graph, table)
>>> print('inputs_ = %r' % (inputs_,))
>>> inputs = inputs_.expand_input(1)
>>> rmi = inputs.rmi_list[0]
>>> result = ('inputs = %s' % (inputs,)) + '\n'
>>> result += ('compute_edges = %s' % (ut.repr2(inputs.flat_compute_rmi_edges(),␣
↪nl=1)))
>>> print(result)
```

wbia.dtool.input_helpers.**make_expanded_input_graph**(*graph*, *target*)

Starting from the *target* property we trace all possible paths in the *graph* back to all sources.

Parameters

- **graph** (*nx.DiMultiGraph*) – the dependency graph with a single source.
- **target** (*str*) – a single target node in graph

Notes

Each edge in the graph must have a *local_input_id* that defines the type of edge it is: (eg one-to-many, one-to-one, nwise/multi).

Step 1: Extracting the Relevant Subgraph We start by searching for all sources of the graph (we assume there is only one). Then we extract the subgraph defined by all edges between the sources and the target. We augment this graph with a dummy super source *s* and super sink *t*. This allows us to associate an edge with the real source and sink.

Step 2: Trace all paths from *s* to *t*. Create a set of all paths from the source to the sink and accumulate the *local_input_id* of each edge along the path. This will uniquely identify each path. We use a hack to condense the accumualated ids in order to display them nicely.

Step 3: Create the new *exi_graph* Using the traced paths with ids we construct a new graph representing expanded inputs. The nodes in the original graph will be copied for each unique path that passes through the node. We identify these nodes using the accumulated ids built along the edges in our path set. For each path starting from the target we add each node augmented with the accumulated ids on its output(?) edge. We also add the edges along these paths which results in the final *exi_graph*.

Step 4: Identify valid inputs candidates The purpose of this graph is to identify which inputs are needed to compute dependant properties. One valid set of inputs is all sources of the graph. However, sometimes it is preferable to specify a model that may have been trained from many inputs. Therefore any node with a one-to-many input edge may also be specified as an input.

Step 5: Identify root-most inputs The user will only specify one possible set of the inputs. We refer to this set as the "root-most" inputs. This is a set of candiate nodes such that all paths from the sink to the super source are blocked. We default to the set of inputs which results in the fewest dependency computations. However this is arbitary.

The last step that is not represented here is to compute the order that the branches must be specified in when given to the depcache for a computation.

Returns exi_graph: the expanded input graph

Return type nx.DiGraph

Notes

All * nodes are defined to be distinct. TODO: To make a * node non-distinct it must be suffixed with an identifier.
CommandLine: python -m dtool.input_helpers make_expanded_input_graph –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.input_helpers import *  # NOQA
>>> from wbia.dtool.example_depcache2 import * # NOQA
>>> depc = testdata_depc3()
>>> table = depc['smk_match']
>>> table = depc['vsone']
>>> graph = table.depc.explicit_graph.copy()
>>> target = table.tablename
>>> exi_graph = make_expanded_input_graph(graph, target)
>>> x = list(exi_graph.nodes())[0]
>>> print('x = %r' % (x,))
>>> # xdoctest: +REQUIRES(--show)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.show_nx(graph, fnum=1, pnum=(1, 2, 1))
>>> pt.show_nx(exi_graph, fnum=1, pnum=(1, 2, 2))
>>> ut.show_if_requested()
```

wbia.dtool.input_helpers.**recolor_exi_graph**(*exi_graph*, *rootmost_nodes*)

wbia.dtool.input_helpers.**sort_rmi_list**(*rmi_list*)

CommandLine: python -m dtool.input_helpers sort_rmi_list

Example

```
>>> from wbia.dtool.input_helpers import *  # NOQA
>>> from wbia.dtool.example_depcache2 import *  # NOQA
>>> depc =testdata_custom_annot_depc([
...    dict(tablename='Notch_Tips', parents=['annot']),
...    dict(tablename='chips', parents=['annot']),
...    dict(tablename='Cropped_Chips', parents=['chips', 'Notch_Tips']),
... ])
>>> table = depc['Cropped_Chips']
>>> inputs = exi_inputs = table.rootmost_inputs
>>> compute_rmi_edges = exi_inputs.flat_compute_rmi_edges()
>>> input_rmis = compute_rmi_edges[-1][0]
>>> rmi_list = input_rmis[::-1]
>>> rmi_list = sort_rmi_list(rmi_list)
>>> assert rmi_list[0].node[0] == 'chips'
```

1.5.8 wbia.dtool.sql_control module

Interface into SQL for the IBEIS Controller

TODO; need to use some sort of sticky bit so sql files are created with reasonable permissions.

class wbia.dtool.sql_control.**SQLColumnRichInfo**(*column_id*, *name*, *type_*, *notnull*, *dflt_value*, *pk*)

Bases: tuple

column_id

Alias for field number 0

dflt_value

Alias for field number 4

name
Alias for field number 1

notnull
Alias for field number 3

pk
Alias for field number 5

type_
Alias for field number 2

class wbia.dtool.sql_control.**SQLDatabaseController**(*uri*, *name*, *readonly=False*, *timeout=600*)

Bases: object

Interface to an SQL database

class Metadata(*ctrlr*)
Bases: collections.abc.Mapping

Metadata is an attribute of the SQLDatabaseController that facilitates easy usages by internal and exteral users. Each metadata attributes represents a table (i.e. an instance of TableMetadata). Each TableMetadata instance has metadata names as attributes. The TableMetadata can also be adapated to a dictionary for compatability.

The the database attribute is a special case that results in a DatabaseMetadata instance rather than TableMetadata. This primarily give access to the version and initial UUID, respectively as database.version and database.init_uuid.

Parameters **ctrlr** (SQLDatabaseController) – parent controller object

class DatabaseMetadata(*ctrlr*)
Bases: collections.abc.MutableMapping

Special metadata for database information

init_uuid

version

class TableMetadata(*ctrlr*, *table_name*)
Bases: collections.abc.MutableMapping

Metadata on a particular SQL table

update(***kwargs*)
Update or insert the value into the metadata table with the given keyword arguments of metadata field names

add_cleanly(*tblname*, *colnames*, *params_iter*, *get_rowid_from_superkey*, *superkey_paramx=(0,)*, ***kwargs*)
ADDER Extra input: the first item of params_iter must be a superkey (like a uuid),

Does not add None values. Does not add duplicate values. For each None input returns None ouptut. For each duplicate input returns existing rowid

Parameters

- **tblname** (*str*) – table name to add into
- **colnames** (*tuple of strs*) – columns whos values are specified in params_iter

- **params_iter** (*iterable*) – an iterable of tuples where each tuple corresonds to a row
- **get_rowid_from_superkey** (*func*) – function that tests if a row needs to be added. It should return None for any new rows to be inserted. It should return the existing rowid if one exists
- **superkey_paramx** (*tuple of ints*) – indices of tuples in params_iter which correspond to superkeys. defaults to (0,)

Returns rowid_list_ – list of newly added or previously added rowids

Return type iterable

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.dtool.sql_control import *  # NOQA
>>> db = SQLDatabaseController('sqlite:///', 'testing')
>>> db.add_table('dummy_table', (
>>>     ('rowid',               'INTEGER PRIMARY KEY'),
>>>     ('key',                 'TEXT'),
>>>     ('superkey1',           'TEXT'),
>>>     ('superkey2',           'TEXT'),
>>>     ('val',                 'TEXT'),
>>> ),
>>>     superkeys=[('key',), ('superkey1', 'superkey2')],
>>>     docstr='')
>>> db.print_schema()
>>> tblname = 'dummy_table'
>>> colnames = ('key', 'val')
>>> params_iter = [('spam', 'eggs'), ('foo', 'bar')]
>>> # Find a useable superkey
>>> superkey_colnames = db.get_table_superkey_colnames(tblname)
>>> superkey_paramx = None
>>> for superkey in superkey_colnames:
>>>    if all(k in colnames for k in superkey):
>>>        superkey_paramx = [colnames.index(k) for k in superkey]
>>>        superkey_colnames = ut.take(colnames, superkey_paramx)
>>>        break
>>> def get_rowid_from_superkey(superkeys_list):
>>>     return db.get_where_eq(tblname, ('rowid',), zip(superkeys_list),␣
↪superkey_colnames)
>>> rowid_list_ = db.add_cleanly(
>>>     tblname, colnames, params_iter, get_rowid_from_superkey, superkey_
↪paramx)
>>> print(rowid_list_)
```

add_column(*tablename*, *colname*, *coltype*)

add_table(*tablename=None*, *coldef_list=None*, ***metadata_keyval*)

Parameters

- **tablename** (*str*) –
- **coldef_list** (*list*) –
- **constraint** (*list or None*) –
- **docstr** (*str*) –

- **superkeys** (*list or None*) – list of tuples of column names which uniquely identifies a rowid

backup(*backup_filepath*)
backup_filepath = dst_fpath

check_rowid_exists(*tablename*, *rowid_iter*, *eager=True*, ***kwargs*)
Check for the existence of rows (`rowid_iter`) in a table (`tablename`). Returns as sequence of rowids that exist in the given sequence.

The 'rowid' term is an alias for the primary key. When calling this method, you should know that the primary key may be more than one column.

connect()
Create a connection instance to wrap a SQL execution block as a context manager

delete(*tblname*, *id_list*, *id_colname='rowid'*, ***kwargs*)
Deletes rows from a SQL table (`tblname`) by ID, given a sequence of IDs (`id_list`). Optionally a different ID column can be specified via `id_colname`.

delete_rowids(*tblname*, *rowid_list*, ***kwargs*)
deletes the the rows in rowid_list

drop_all_tables()
DELETES ALL INFO IN TABLE

drop_table(*tablename*, *invalidate_cache=True*)

dump_schema()
Convenience: Dumps all csv database files to disk NOTE: This function is semi-obsolete because of the auto-generated current schema file. Use dump_schema_current_autogeneration instead for all purposes except for parsing out the database schema or for consice visual representation.

dump_tables_to_csv(*dump_dir=None*)
Convenience: Dumps all csv database files to disk

ensure_postgresql_types(*conn*)
Create a connection instance to wrap a SQL execution block as a context manager

executemany(*operation*, *params_iter*, *unpack_scalars=True*, *keepwrap=False*, ***kwargs*)
Executes the given `operation` once for each item in `params_iter`

Parameters

- **operation** (*str*) – SQL operation
- **params_iter** (*sequence*) – a sequence of sequences containing parameters in the sql operation
- **unpack_scalars** (*bool*) – [deprecated] use to unpack a single result from each query only use with operations that return a single result for each query (default: True)

executeone(*operation*, *params=()*, *eager=True*, *verbose=False*, *use_fetchone_behavior=False*, *keepwrap=False*)
Executes the given `operation` once with the given set of `params`

Parameters

- **operation** (*str | TextClause*) – SQL statement
- **params** (*sequence | dict*) – parameters to pass in with SQL execution
- **eager** – [deprecated] no-op

- **verbose** – [deprecated] no-op
- **use_fetchone_behavior** (*bool*) – Use DBAPI `fetchone` behavior when outputing no rows (i.e. None)

exists_where_eq(*tblname*, *params_iter*, *where_colnames*, *op='AND'*, *unpack_scalars=True*, *eager=True*, ***kwargs*)
hacked in function for nicer templates

get(*tblname*, *colnames*, *id_iter=None*, *id_colname='rowid'*, *eager=True*, *assume_unique=False*, *batch_size=10000*, ***kwargs*)
Get rows of data by ID

Parameters

- **tblname** (*str*) – table name to get from
- **colnames** (*tuple of str*) – column names to grab from
- **id_iter** (*iterable*) – iterable of search keys
- **id_colname** (*str*) – column to be used as the search key (default: rowid)
- **eager** (*bool*) – use eager evaluation
- **assume_unique** (*bool*) – default False. Experimental feature that could result in a 10x speedup
- **unpack_scalars** (*bool*) – default True

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> depc.clear_all()
>>> rowids = depc.get_rowids('notch', [1, 2, 3])
>>> table = depc['notch']
>>> db = table.db
>>> table.print_csv()
>>> # Break things to test set
>>> colnames = ('dummy_annot_rowid',)
>>> got_data = db.get('notch', colnames, id_iter=rowids)
>>> assert got_data == [1, 2, 3]
```

get_all_col_rows(*tblname*, *colname*)
returns a list of all rowids from a table in ascending order

get_all_rowids(*tblname*, ***kwargs*)
returns a list of all rowids from a table in ascending order

get_all_rowids_where(*tblname*, *where_clause*, *params*, ***kwargs*)
returns a list of rowids from a table in ascending order satisfying a condition

get_coldef_list(*tablename*)

Returns each tuple is (col_name, col_type)

Return type list of (str, str)

get_column(*tablename*, *name*)
Get all the values for the specified column (`name`) of the table (`tablename`)

get_column_names(*tablename*)
Conveinience: Returns the sql tablename columns

get_columns(*tablename*)

Parameters **tablename** (*str*) – table name

Returns

list of tuples with format:

(column_id : id of the column name : the name of the column type_ : the type of the column notnull : 0 or 1 if the column can contains null values dflt_value : the default value pk : 0 or 1 if the column partecipate to the primary key

)

Return type column_list

References

http://stackoverflow.com/questions/17717829/how-to-get-column-names-from-a-table-in-sqlite-via-pragma-net-c
http://stackoverflow.com/questions/1601151/how-do-i-check-in-sqlite-whether-a-table-exists

CommandLine: python -m dtool.sql_control –exec-get_columns python -m dtool.sql_control –exec-get_columns –tablename=contributors python -m dtool.sql_control –exec-get_columns –tablename=nonexist

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.sql_control import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> tablename = 'keypoint'
>>> db = depc[tablename].db
>>> colrichinfo_list = db.get_columns(tablename)
>>> result = ('colrichinfo_list = %s' % (ut.repr2(colrichinfo_list, nl=1),))
>>> print(result)
colrichinfo_list = [
    (0, 'keypoint_rowid', 'INTEGER', 0, None, 1),
    (1, 'chip_rowid', 'INTEGER', 1, None, 0),
    (2, 'config_rowid', 'INTEGER', 0, '0', 0),
    (3, 'kpts', 'NDARRAY', 0, None, 0),
    (4, 'num', 'INTEGER', 0, None, 0),
]
```

get_db_init_uuid(*ensure=True*)
Get the database initialization (creation) UUID

CommandLine: python -m dtool.sql_control get_db_init_uuid

Example

```
>>> # ENABLE_DOCTEST
>>> import uuid
>>> import os
>>> from wbia.dtool.sql_control import *  # NOQA
>>> # Check random database gets new UUID on init
>>> db = SQLDatabaseController('sqlite:///', 'testing')
>>> uuid_ = db.get_db_init_uuid()
>>> print('New Database: %r is valid' % (uuid_, ))
>>> assert isinstance(uuid_, uuid.UUID)
>>> # Check existing database keeps UUID
>>> sqldb_dpath = ut.ensure_app_resource_dir('dtool')
>>> sqldb_fname = u'test_database.sqlite3'
>>> path = os.path.join(sqldb_dpath, sqldb_fname)
>>> db_uri = 'sqlite:///{}'.format(os.path.realpath(path))
>>> db1 = SQLDatabaseController(db_uri, 'db1')
>>> uuid_1 = db1.get_db_init_uuid()
>>> db2 = SQLDatabaseController(db_uri, 'db2')
>>> uuid_2 = db2.get_db_init_uuid()
>>> print('Existing Database: %r == %r' % (uuid_1, uuid_2, ))
>>> assert uuid_1 == uuid_2
```

get_db_version(*ensure=True*)

get_metadata_items()

Returns metadata_items

Return type list

CommandLine: python -m dtool.sql_control –exec-get_metadata_items

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.example_depcache import testdata_depc
>>> from wbia.dtool.sql_control import *  # NOQA
>>> db = testdata_depc()['notch'].db
>>> metadata_items = db.get_metadata_items()
>>> result = ('metadata_items = %s' % (ut.repr2(sorted(metadata_items)),))
>>> print(result)
```

get_metadata_val(*key*, *eval_=False*, *default=None*)

val is the repr string unless **eval_** is true

get_row_count(*tblname*)

get_rowid_from_superkey(*tblname*, *params_iter=None*, *superkey_colnames=None*, ***kwargs*)

getter which uses the constrained superkeys instead of rowids

get_schema_current_autogeneration_str(*autogen_cmd=''*)

Convenience: Autogenerates the most up-to-date database schema

CommandLine: python -m dtool.sql_control –exec-get_schema_current_autogeneration_str

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.sql_control import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> tablename = 'keypoint'
>>> db = depc[tablename].db
>>> result = db.get_schema_current_autogeneration_str('')
>>> print(result)
```

get_sql_version()
Conveinience

get_table_as_pandas(*tablename*, *rowids=None*, *columns=None*, *exclude_columns=[]*)
aid = 30 db = ibs.staging rowids = ut.flatten(ibs.get_review_rowids_from_single([aid])) tablename = 'reviews' exclude_columns = 'review_user_confidence review_user_identity'.split(' ') logger.info(db.get_table_as_pandas(tablename, rowids, exclude_columns=exclude_columns))

db = ibs.db rowids = ut.flatten(ibs.get_annotmatch_rowids_from_aid([aid])) tablename = 'annotmatch' exclude_columns = 'annotmatch_confidence annotmatch_posixtime_modified annotmatch_reviewer'.split(' ') logger.info(db.get_table_as_pandas(tablename, rowids, exclude_columns=exclude_columns))

get_table_autogen_dict(*tablename*)

Parameters **tablename** (*str*) –

Returns autogen_dict

Return type dict

CommandLine: python -m dtool.sql_control get_table_autogen_dict

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.sql_control import *  # NOQA
>>> db = SQLDatabaseController('sqlite:///', 'testing')
>>> tablename = 'dummy_table'
>>> db.add_table(tablename, (
>>>     ('rowid', 'INTEGER PRIMARY KEY'),
>>>     ('value1', 'TEXT'),
>>>     ('value2', 'TEXT NOT NULL'),
>>>     ('value3', 'TEXT DEFAULT 1'),
>>>     ('time_added', "INTEGER DEFAULT (CAST(STRFTIME('%s', 'NOW', 'UTC')␣
↪AS INTEGER))")
>>> ))
>>> autogen_dict = db.get_table_autogen_dict(tablename)
>>> result = ut.repr2(autogen_dict, nl=2)
>>> print(result)
```

get_table_autogen_str(*tablename*)

Parameters **tablename** (*str*) –

Returns quoted_docstr

Return type str

CommandLine: python -m dtool.sql_control get_table_autogen_str

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.sql_control import *  # NOQA
>>> db = SQLDatabaseController('sqlite:///', 'testing')
>>> tablename = 'dummy_table'
>>> db.add_table(tablename, (
>>>     ('rowid', 'INTEGER PRIMARY KEY'),
>>>     ('value', 'TEXT'),
>>>     ('time_added', "INTEGER DEFAULT (CAST(STRFTIME('%s', 'NOW', 'UTC')␣
↪AS INTEGER))")
>>> ))
>>> result = '\n'.join(db.get_table_autogen_str(tablename))
>>> print(result)
```

get_table_column_data(*tablename*, *columns=None*, *exclude_columns=[]*, *rowids=None*)
Grabs a table of information

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.sql_control import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> tablename = 'keypoint'
>>> db = depc[tablename].db
>>> column_list, column_names = db.get_table_column_data(tablename)
>>> column_list
[[], [], [], [], []]
>>> column_names
['keypoint_rowid', 'chip_rowid', 'config_rowid', 'kpts', 'num']
```

get_table_constraints(*tablename*)
TODO: use coldef_list with table_autogen_dict instead

get_table_csv(*tablename*, *exclude_columns=[]*, *rowids=None*, *truncate=False*)
Converts a tablename to csv format

Parameters

- **tablename** (*str*) –
- **exclude_columns** (*list*) –

Returns csv_table

Return type str

CommandLine: python -m dtool.sql_control –test-get_table_csv python -m dtool.sql_control –exec-get_table_csv –tablename=contributors

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.sql_control import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> depc.clear_all()
>>> rowids = depc.get_rowids('notch', [1, 2, 3])
>>> table = depc['notch']
>>> db = table.db
>>> ut.exec_funckw(db.get_table_csv, globals())
>>> tablename = 'notch'
>>> csv_table = db.get_table_csv(tablename, exclude_columns, truncate=True)
>>> print(csv_table)
```

get_table_csv_header(*tablename*)

get_table_docstr(*tablename*)

CommandLine: python -m dtool.sql_control –exec-get_table_docstr

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.sql_control import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> tablename = 'keypoint'
>>> db = depc[tablename].db
>>> result = db.get_table_docstr(tablename)
>>> print(result)
Used to store individual chip features (ellipses)
```

get_table_names(*lazy=False*)

Conveinience:

get_table_new_transferdata(*tablename*, *exclude_columns=[]*)

CommandLine: python -m dtool.sql_control –test-get_table_column_data python -m dtool.sql_control –test-get_table_new_transferdata python -m dtool.sql_control –test-get_table_new_transferdata:1

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.sql_control import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> tablename = 'keypoint'
>>> db = depc[tablename].db
>>> tablename_list = db.get_table_names()
>>> colrichinfo_list = db.get_columns(tablename)
>>> for tablename in tablename_list:
...     new_transferdata = db.get_table_new_transferdata(tablename)
...     column_list, column_names, extern_colx_list, extern_superkey_colname_
↪list, extern_superkey_colval_list, extern_tablename_list, extern_
↪primarycolnames_list = new_transferdata
```

(continues on next page)

(continued from previous page)

```
...     print('tablename = %r' % (tablename,))
...     print('colnames = ' + ut.repr2(column_names))
...     print('extern_colx_list = ' + ut.repr2(extern_colx_list))
...     print('extern_superkey_colname_list = ' + ut.repr2(extern_superkey_
↪colname_list))
...     print('L___')
```

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +REQUIRES(module:wbia)
>>> from wbia.dtool.sql_control import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> db = ibs.db
>>> exclude_columns = []
>>> tablename_list = ibs.db.get_table_names()
>>> for tablename in tablename_list:
...     new_transferdata = db.get_table_new_transferdata(tablename)
...     column_list, column_names, extern_colx_list, extern_superkey_colname_
↪list, extern_superkey_colval_list, extern_tablename_list, extern_
↪primarycolnames_list = new_transferdata
...     print('tablename = %r' % (tablename,))
...     print('colnames = ' + ut.repr2(column_names))
...     print('extern_colx_list = ' + ut.repr2(extern_colx_list))
...     print('extern_superkey_colname_list = ' + ut.repr2(extern_superkey_
↪colname_list))
...     print('L___')
```

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +REQUIRES(module:wbia)
>>> from wbia.dtool.sql_control import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> db = ibs.db
>>> exclude_columns = []
>>> tablename = ibs.const.IMAGE_TABLE
>>> new_transferdata = db.get_table_new_transferdata(tablename)
>>> column_list, column_names, extern_colx_list, extern_superkey_colname_
↪list, extern_superkey_colval_list, extern_tablename_list, extern_
↪primarycolnames_list = new_transferdata
>>> dependsmap = db.metadata[tablename].dependsmap
>>> print('tablename = %r' % (tablename,))
>>> print('colnames = ' + ut.repr2(column_names))
>>> print('extern_colx_list = ' + ut.repr2(extern_colx_list))
>>> print('extern_superkey_colname_list = ' + ut.repr2(extern_superkey_
↪colname_list))
>>> print('dependsmap = %s' % (ut.repr2(dependsmap, nl=True),))
>>> print('L___')
>>> tablename = ibs.const.ANNOTATION_TABLE
>>> new_transferdata = db.get_table_new_transferdata(tablename)
```

(continues on next page)

(continued from previous page)

```
>>> column_list, column_names, extern_colx_list, extern_superkey_colname_
↪list, extern_superkey_colval_list, extern_tablename_list, extern_
↪primarycolnames_list = new_transferdata
>>> dependsmap = db.metadata[tablename].dependsmap
>>> print('tablename = %r' % (tablename,))
>>> print('colnames = ' + ut.repr2(column_names))
>>> print('extern_colx_list = ' + ut.repr2(extern_colx_list))
>>> print('extern_superkey_colname_list = ' + ut.repr2(extern_superkey_
↪colname_list))
>>> print('dependsmap = %s' % (ut.repr2(dependsmap, nl=True),))
>>> print('L___')
```

get_table_primarykey_colnames(*tablename*)

get_table_superkey_colnames(*tablename*)

Actually resturns a list of tuples. need to change the name to get_table_superkey_colnames_list

Parameters **tablename** (*str*) –

Returns superkeys

Return type list

CommandLine: python -m dtool.sql_control –test-get_table_superkey_colnames python -m wbia –tf get_table_superkey_colnames –tablename=contributors python -m wbia –tf get_table_superkey_colnames –db PZ_Master0 –tablename=annotations python -m wbia –tf get_table_superkey_colnames –db PZ_Master0 –tablename=contributors # NOQA

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.sql_control import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> db = depc['chip'].db
>>> superkeys = db.get_table_superkey_colnames('chip')
>>> result = ut.repr2(superkeys, nl=False)
>>> print(result)
[('dummy_annot_rowid', 'config_rowid')]
```

get_where(*tblname*, *colnames*, *params_iter*, *where_clause*, *unpack_scalars=True*, *eager=True*, ***kwargs*)

Interface to do a SQL select with a where clause

Parameters

- **tblname** (*str*) – table name
- **colnames** (*tuple[str]*) – sequence of column names
- **params_iter** (*list[dict]*) – a sequence of dicts with parameters, where each item in the sequence is used in a SQL execution
- **where_clause** (*str|Operation*) – conditional statement used in the where clause
- **unpack_scalars** (*bool*) – [deprecated] use to unpack a single result from each query only use with operations that return a single result for each query (default: True)

get_where_eq(*tblname*, *colnames*, *params_iter*, *where_colnames*, *unpack_scalars=True*, *op='AND'*, *batch_size=10000*, ***kwargs*)
Executes a SQL select where the given parameters match/equal the specified where columns.

Parameters

- **tblname** (*str*) – table name
- **colnames** (*tuple[str]*) – sequence of column names
- **params_iter** (*list[list]*) – a sequence of a sequence with parameters, where each item in the sequence is used in a SQL execution
- **where_colnames** (*list[str]*) – column names to match for equality against the same index of the param_iter values
- **op** (*str*) – SQL boolean operator (e.g. AND, OR)
- **unpack_scalars** (*bool*) – [deprecated] use to unpack a single result from each query only use with operations that return a single result for each query (default: True)

get_where_eq_set(*tblname*, *colnames*, *params_iter*, *where_colnames*, *unpack_scalars=True*, *eager=True*, *op='AND'*, ***kwargs*)

has_table(*tablename*, *colnames=None*, *lazy=True*)
checks if a table exists

integrity()

invalidate_tables_cache()
Invalidates the controller's cache of table names and objects Resets the caches and/or repopulates them.

is_using_postgres

is_using_sqlite

make_json_table_definition(*tablename*)
VERY HACKY FUNC RIGHT NOW. NEED TO FIX LATER

Parameters **tablename** –

Returns new_transferdata

Return type

?

CommandLine: python -m wbia –tf sql_control.make_json_table_definition

CommandLine: python -m utool –tf iter_module_doctestable –modname=dtool.sql_control –include_inherited=True python -m dtool.sql_control –exec-make_json_table_definition

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.sql_control import *  # NOQA
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> tablename = 'keypoint'
>>> db = depc[tablename].db
>>> table_def = db.make_json_table_definition(tablename)
>>> result = ('table_def = %s' % (ut.repr2(table_def, nl=True),))
```

(continues on next page)

(continued from previous page)

```
>>> print(result)
table_def = {
    'keypoint_rowid': 'INTEGER',
    'chip_rowid': 'INTEGER',
    'config_rowid': 'INTEGER',
    'kpts': 'NDARRAY',
    'num': 'INTEGER',
}
```

merge_databases_new(*db_src*, *ignore_tables=None*, *rowid_subsets=None*)

Copies over all non-rowid properties into another sql table. handles annotated dependenceis. Does not handle external files Could handle dependency tree order, but not yet implemented.

FINISHME

Parameters **db_src** (*SQLController*) – merge data from db_src into db

CommandLine: python -m dtool.sql_control –test-merge_databases_new:0 python -m dtool.sql_control –test-merge_databases_new:2

Example

```
>>> # DISABLE_DOCTEST
>>> # xdoctest: +REQUIRES(module:wbia)
>>> from wbia.dtool.sql_control import *  # NOQA
>>> import wbia
>>> #ibs_dst = wbia.opendb(dbdir='testdb_dst')
>>> ibs_src = wbia.opendb(db='testdb1')
>>> # OPEN A CLEAN DATABASE
>>> ibs_dst = wbia.opendb(dbdir='test_sql_merge_dst1', allow_newdir=True,␣
↪delete_ibsdir=True)
>>> ibs_src.ensure_contributor_rowids()
>>> # build test data
>>> db = ibs_dst.db
>>> db_src = ibs_src.db
>>> rowid_subsets = None
>>> # execute function
>>> db.merge_databases_new(db_src)
```

Example

```
>>> # DISABLE_DOCTEST
>>> # xdoctest: +REQUIRES(module:wbia)
>>> from wbia.dtool.sql_control import *  # NOQA
>>> import wbia
>>> ibs_src = wbia.opendb(db='testdb2')
>>> # OPEN A CLEAN DATABASE
>>> ibs_dst = wbia.opendb(dbdir='test_sql_merge_dst2', allow_newdir=True,␣
↪delete_ibsdir=True)
>>> ibs_src.ensure_contributor_rowids()
>>> # build test data
>>> db = ibs_dst.db
>>> db_src = ibs_src.db
```

(continues on next page)

(continued from previous page)

```
>>> ignore_tables = ['lblannot', 'lblimage', 'image_lblimage_relationship',
↪'annotation_lblannot_relationship', 'keys']
>>> rowid_subsets = None
>>> # execute function
>>> db.merge_databases_new(db_src, ignore_tables=ignore_tables)
```

Example

```
>>> # DISABLE_DOCTEST
>>> # xdoctest: +REQUIRES(module:wbia)
>>> from wbia.dtool.sql_control import *  # NOQA
>>> import wbia
>>> ibs_src = wbia.opendb(db='testdb2')
>>> # OPEN A CLEAN DATABASE
>>> ibs_src.fix_invalid_annotmatches()
>>> ibs_dst = wbia.opendb(dbdir='test_sql_subexport_dst2', allow_newdir=True,
↪ delete_ibsdir=True)
>>> ibs_src.ensure_contributor_rowids()
>>> # build test data
>>> db = ibs_dst.db
>>> db_src = ibs_src.db
>>> ignore_tables = ['lblannot', 'lblimage', 'image_lblimage_relationship',
↪'annotation_lblannot_relationship', 'keys']
>>> # execute function
>>> aid_subset = [1, 2, 3]
>>> rowid_subsets = {ANNOTATION_TABLE: aid_subset,
...                  NAME_TABLE: ibs_src.get_annot_nids(aid_subset),
...                  IMAGE_TABLE: ibs_src.get_annot_gids(aid_subset),
...                  ANNOTMATCH_TABLE: [],
...                  GSG_RELATION_TABLE: [],
...                  }
>>> db.merge_databases_new(db_src, ignore_tables=ignore_tables, rowid_
↪subsets=rowid_subsets)
```

modify_table(*tablename=None*, *colmap_list=None*, *tablename_new=None*, *drop_columns=[]*, *add_columns=[]*, *rename_columns=[]*, ***metadata_keyval*)

function to modify the schema - only columns that are being added, removed or changed need to be enumerated

Parameters

- **tablename** (*str*) – tablename
- **colmap_list** (*list*) – of tuples (orig_colname, new_colname, new_coltype, convert_func) orig_colname - the original name of the column, None to append, int for index new_colname - the new column name (” for same, None to delete) new_coltype - New Column Type. None to use data unmodified convert_func - Function to convert data from old to new
- **constraint** (*str*) –
- **superkeys** (*list*) –
- **docstr** (*str*) –
- **tablename_new** –

Example

```
>>> # DISABLE_DOCTEST
>>> def loc_zip_map(x):
...     return x
>>> db.modify_table(const.CONTRIBUTOR_TABLE, (
>>>         # orig_colname,             new_colname,      new_coltype,␣
↪convert_func
>>>         # a non-needed, but correct mapping (identity function)
>>>         ('contrib_rowid',      '',                       '',              ␣
↪None),
>>>         # for new columns, function is ignored (TYPE CANNOT BE EMPTY IF␣
↪ADDING)
>>>         (None,                 'contrib_loc_address', 'TEXT',             ␣
↪None),
>>>         # adding a new column at index 4 (if index is invalid, None is␣
↪used)
>>>         (4,                    'contrib_loc_address', 'TEXT',             ␣
↪None),
>>>         # for deleted columns, type and function are ignored
>>>         ('contrib_loc_city',    None,                    '',              ␣
↪None),
>>>         # for renamed columns, type and function are ignored
>>>         ('contrib_loc_city',   'contrib_loc_town',    '',       None),
>>>         ('contrib_loc_zip',    'contrib_loc_zip',     'TEXT',   loc_zip_
↪map),
>>>         # type not changing, only NOT NULL provision
>>>         ('contrib_loc_country', '',                   'TEXT NOT NULL', ␣
↪None),
>>>     ),
>>>     superkeys=[('contributor_rowid',)],
>>>     constraint=[],
>>>     docstr='Used to store the contributors to the project'
>>> )
```

optimize()

print_dbg_schema()

print_schema()

print_table_csv(*tablename*, *exclude_columns=[]*, *truncate=False*)

reboot()

rename_table(*tablename_old*, *tablename_new*, *invalidate_cache=True*)

rows_exist(*tblname*, *rowids*)
Checks if rowids exist. Yields True if they do

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

schema_name
The name of the namespace schema (using with Postgres).

set(*tblname*, *colnames*, *val_iter*, *id_iter*, *id_colname='rowid'*, *duplicate_behavior='error'*, *duplcate_auto_resolve=True*, ***kwargs*)
setter

CommandLine: python -m dtool.sql_control set

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.dtool.example_depcache import testdata_depc
>>> depc = testdata_depc()
>>> depc.clear_all()
>>> rowids = depc.get_rowids('notch', [1, 2, 3])
>>> table = depc['notch']
>>> db = table.db
>>> table.print_csv()
>>> # Break things to test set
>>> colnames = ('dummy_annot_rowid',)
>>> val_iter = [(9003,), (9001,), (9002,)]
>>> orig_data = db.get('notch', colnames, id_iter=rowids)
>>> db.set('notch', colnames, val_iter, id_iter=rowids)
>>> new_data = db.get('notch', colnames, id_iter=rowids)
>>> assert new_data == [x[0] for x in val_iter]
>>> assert new_data != orig_data
>>> table.print_csv()
>>> depc.clear_all()
```

set_db_version(*version*)

set_metadata_val(*key*, *val*)
key must be given as a repr-ed string

shrink_memory()

squeeze()

tablenames

vacuum()

view_db_in_external_reader()

class wbia.dtool.sql_control.**SQLTable**(*db*, *name*)
Bases: utool.util_dev.NiceRepr

convinience object for dealing with a specific table

table = db table = SQLTable(db, 'annotmatch')

as_pandas(*rowids=None*, *columns=None*)

clear()

delete(*rowids*)

get(*colnames*, *id_iter*, *id_colname='rowid'*, *eager=True*)

number_of_rows()

rowids()

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

wbia.dtool.sql_control.**compare_coldef_lists**(*coldef_list1*, *coldef_list2*)

wbia.dtool.sql_control.**create_engine**(*uri*, *POSTGRESQL_POOL_SIZE=20*, *ENGINES={}*, *timeout=600*)

wbia.dtool.sql_control.**sanitize_sql**(*db*, *tablename_*, *columns=None*)
Sanatizes an sql tablename and column. Use sparingly

wbia.dtool.sql_control.**tuplize**(*list_*)
Converts each scalar item in a list to a dimension-1 tuple

1.5.9 Module contents

1.6 wbia.expt package

1.6.1 Submodules

1.6.2 wbia.expt.annotation_configs module

Definitions for common aid configurations

Rename to annot_cfgdef

wbia.expt.annotation_configs.**apply_qualcontrol**(*acfg*)

wbia.expt.annotation_configs.**apply_timecontrol**(*acfg*, *min_timedelta='6h'*, *require_timestamp=True*)

wbia.expt.annotation_configs.**compress_acfg_list_for_printing**(*acfg_list*)
CommandLine: python -m wbia –tf compress_acfg_list_for_printing
Ignore:

```
>>> from wbia.expt.annotation_configs import *  # NOQA
>>> qcfg_list = [{'f': 1, 'b': 1}, {'f': 2, 'b': 1}, {'f': 3, 'b': 1, 'z': 4}
↪]
>>> acfg_list = [{'qcfg': qcfg} for qcfg in qcfg_list]
>>> nonvaried_dict, varied_dicts = compress_acfg_list_for_printing(acfg_list)
>>> result = ('varied_dicts = %s\n' % (ut.repr2(varied_dicts),))
>>> result += ('nonvaried_dict = %s' % (ut.repr2(nonvaried_dict),))
>>> print(result)
```

wbia.expt.annotation_configs.**compress_aidcfg**(*acfg*, *filter_nones=False*, *filter_empty=False*, *force_noncommon=[]*)
Idea is to add a third subconfig named *common* that is the intersection of *qcfg* and *dcfg*.

Parameters **acfg** (*dict*) –

Returns acfg

Return type dict

CommandLine: # python -m wbia –tf compress_aidcfg python -m wbia.expt.annotation_configs –exec-compress_aidcfg –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.annotation_configs import *  # NOQA
>>> acfg = default
>>> acfg = compress_aidcfg(acfg)
>>> result = ('acfg = %s' % (ut.repr2(acfg),))
>>> print(default)
>>> print(result)
```

wbia.expt.annotation_configs.**ctrl = {'dcfg': {'any_endswith': None, 'any_match': None,**
wbia -e print_acfg –db PZ_Master1 -a timectrl

wbia.expt.annotation_configs.**default2 = {'dcfg': {'any_endswith': None, 'any_match': Nor**
wbia -e print_acfg –db PZ_Master1 -a unctrl

wbia.expt.annotation_configs.**flatten_acfg_list**(*acfg_list*)
Returns a new config where subconfig params are prefixed by subconfig keys

wbia.expt.annotation_configs.**get_varied_acfg_labels**(*acfg_list*, *mainkey='_cfgname'*, *checkname=False*)

```
>>> from wbia.expt.annotation_configs import *  # NOQA
```

wbia.expt.annotation_configs.**partition_acfg_list**(*acfg_list*)

wbia.expt.annotation_configs.**print_acfg**(*acfg*, *expanded_aids=None*, *ibs=None*, ***kwargs*)

wbia.expt.annotation_configs.**print_acfg_list**(*acfg_list*, *expanded_aids_list=None*, *ibs=None*, *combined=False*, *only_summary=False*, ***kwargs*)

Parameters

- **acfg_list** (*list*) –
- **expanded_aids_list** (*list*) – (default = None)
- **ibs** (IBEISController) – wbia controller object(default = None)
- **combined** (*bool*) – (default = False)

CommandLine: python -m wbia.expt.annotation_configs –exec-print_acfg_list

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.annotation_configs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> a = ['default']
>>> acfg_list, expanded_aids_list = wbia.expt.experiment_helpers.get_annotcfg_
↪list(
>>>     ibs, acfg_name_list=a, verbose=0)
>>> combined = False
>>> result = print_acfg_list(acfg_list, expanded_aids_list, ibs, combined)
>>> print(result)
```

wbia.expt.annotation_configs.**shorten_to_alias_labels**(*key*)

wbia.expt.annotation_configs.**timectrlL = {'dcfg': {'any_endswith': None, 'any_match': No**
wbia -e print_acfg –db PZ_Master1 -a timequalctrl

wbia.expt.annotation_configs.**timectrlhard = {'dcfg': {'any_endswith': None, 'any_match':**
wbia -e print_acfg -a viewdiff –db PZ_Master1 –verbtd –nocache –per_vp=True wbia -e print_acfg -a viewdiff_td –db PZ_Master1 –verbtd –nocache –per_vp=True

wbia.expt.annotation_configs.**unctrl_comp = {'dcfg': {'any_endswith': None, 'any_match':**
wbia -e print_acfg –db PZ_Master1 -a ctrl wbia -e print_acfg –db PZ_Master1 -a unctrl ctrl::unctrl:qpername=1,qview_ext=0 wbia -e print_acfg –db PZ_Master1 -a unctrl ctrl::unctrl_comp

wbia.expt.annotation_configs.**unflatten_acfgdict**(*flat_dict*, *prefix_list=['dcfg', 'qcfg']*)

wbia.expt.annotation_configs.**varynannots_td1h = {'dcfg': {'any_endswith': None, 'any_matc**
wbia -e print_acfg -a viewpoint_compare –db PZ_Master1 –verbtd –nocache python -m wbia –tf parse_acfg_combo_list -a viewpoint_compare python -m wbia –tf get_annotcfg_list –db PZ_Master1 -a viewpoint_compare –verbtd # Check composition of names per viewpoint python -m wbia –tf group_annots_by_multi_prop –db PZ_Master1 –props=yaw_texts,name_rowids –keys1 frontleft python -m wbia –tf get_annot_stats_dict –db PZ_Master1 –per_name_vpedge=True

TODO: Need to explicitly setup the common config I think? wbia -e print_acfg -a viewdiff:min_timedelta=1h –db PZ_Master1 –verbtd –nocache-aid wbia –tf get_annotcfg_list -a viewdiff:min_timedelta=1h –db PZ_Master1 –verbtd –nocache-aid

wbia.expt.annotation_configs.**varypername2_td = {'dcfg': {'any_endswith': None, 'any_matcl**
wbia -e print_acfg –db PZ_Master1 -a ctrl2 wbia -e print_acfg –db PZ_Master1 -a timectrl2 wbia -e rank_cmc –db PZ_Master1 -a timectrl2 -t invarbest

wbia.expt.annotation_configs.**varypername_td1h = {'dcfg': {'any_endswith': None, 'any_matc**
wbia -e print_acfg –db PZ_Master1 -a varypername_tdqual

wbia.expt.annotation_configs.**varypername_tdqual = {'dcfg': {'any_endswith': None, 'any_ma**
python -m wbia –tf get_num_annots_per_name –db PZ_Master1 wbia -e print_acfg -a varysize2 –db PZ_Master1 –verbtd –nocache wbia -e print_acfg -a varysize2 –db NNP_MasterGIRM_core –verbtd –nocache

wbia.expt.annotation_configs.**varysize = {'dcfg': {'any_endswith': None, 'any_match': Nor**
wbia -e print_acfg -a varysize2_td –db PZ_Master1 –verbtd –nocache

wbia.expt.annotation_configs.**viewdiff_td1h = {'dcfg': {'any_endswith': None, 'any_match'**
qhas_any=(query,),dpername=2,exclude_reference=True –acfginfo –verbtd –veryverbtd wbia get_annotcfg_list –db Oxford -a oxford –acfginfo (‘_QSUUIDS((55)qxlgljvomqpdvlny)’, ‘_DSUUIDS((4240)vhtqsdkrwetbftis)’),

wbia draw_rank_cmc –db Oxford –save oxfordccm.png -p :proot=smk,num_words=[64000],nAssign=[1],sv_on=[False] -a oxford

Type wbia get_annotcfg_list –db Oxford -a default

1.6.3 wbia.expt.cfghelpers module

Helper module that helps expand parameters for grid search

DEPRICATE: Most of this can likely be replaced by util_gridsearch TODO: rectify with versions in util_gridsearch

It turns out a lot of the commandlines made possible here can be generatd by using bash brace expansion. http://www.linuxjournal.com/content/bash-brace-expansion

wbia.expt.cfghelpers.**customize_base_cfg**(*cfgname*, *cfgopt_strs*, *base_cfg*, *cfgtype*, *alias_keys=None*, *valid_keys=None*, *offset=0*, *strict=True*)

DEPRICATE

wbia.expt.cfghelpers.**parse_argv_cfg**(*argname, default=[”], named_defaults_dict=None, valid_keys=None*)

simple configs

Parameters

- **argname** –
- **default** (*list*) – (default = [])
- **named_defaults_dict** (*dict*) – (default = None)
- **valid_keys** (*None*) – (default = None)

Returns cfg_list

Return type list

CommandLine: python -m wbia.expt.cfghelpers –exec-parse_argv_cfg –filt :foo=bar python -m wbia.expt.cfghelpers –test-parse_argv_cfg

Example

```
>>> # ENABLE_DOCTET
>>> from wbia.expt.cfghelpers import *  # NOQA
>>> argname = '--filt'
>>> cfg_list = parse_argv_cfg(argname)
>>> result = ('cfg_list = %s' % (str(cfg_list),))
>>> print(result)
```

wbia.expt.cfghelpers.**parse_cfgstr_list2**(*cfgstr_list*, *named_defaults_dict=None*, *cfgtype=None*, *alias_keys=None*, *valid_keys=None*, *expand_nested=True*, *strict=True*, *special_join_dict=None*, *is_nestedcfgtype=False*, *metadata=None*)

Parses config strings. By looking up name in a dict of configs

DEPRICATE

Parameters

- **cfgstr_list** (*list*) –
- **named_defaults_dict** (*dict*) – (default = None)
- **cfgtype** (*None*) – (default = None)
- **alias_keys** (*None*) – (default = None)
- **valid_keys** (*None*) – (default = None)
- **expand_nested** (*bool*) – (default = True)
- **strict** (*bool*) – (default = True)
- **– used for annot configs so special joins arent geometrically combined** (*is_nestedcfgtype*) –

Note:
Normal Case: –flag name
Custom Arugment Cases: –flag name:custom_key1=custom_val1,custom_key2=custom_val2
Multiple Config Case: –flag name1:custom_args1 name2:custom_args2
Multiple Config (special join) Case: (here name2 and name3 have some special interaction) –flag name1:custom_args1 name2:custom_args2::name3:custom_args3
Varied Argument Case: –flag name:key1=[val1,val2]

Returns cfg_combos_list

Return type list

CommandLine: python -m wbia.expt.cfghelpers –exec-parse_cfgstr_list2 python -m wbia.expt.cfghelpers –test-parse_cfgstr_list2

Example

```
>>> # ENABLE_DOCTET
>>> from wbia.expt.cfghelpers import *  # NOQA
>>> cfgstr_list = ['name', 'name:f=1', 'name:b=[1,2]', 'name1:f=1::name2:f=1,b=2']
>>> #cfgstr_list = ['name', 'name1:f=1::name2:f=1,b=2']
>>> named_defaults_dict = None
>>> cfgtype = None
>>> alias_keys = None
>>> valid_keys = None
>>> expand_nested = True
>>> strict = False
>>> special_join_dict = {'joined': True}
>>> cfg_combos_list = parse_cfgstr_list2(cfgstr_list, named_defaults_dict,
>>>                                      cfgtype, alias_keys, valid_keys,
>>>                                      expand_nested, strict,
>>>                                      special_join_dict)
>>> print('cfg_combos_list = %s' % (ut.repr2(cfg_combos_list, nl=2),))
>>> print(ut.depth_profile(cfg_combos_list))
>>> cfg_list = ut.flatten(cfg_combos_list)
>>> cfg_list = ut.flatten([cfg if isinstance(cfg, list) else [cfg] for cfg in cfg_
↪list])
>>> result = ut.repr2(ut.get_varied_cfg_lbls(cfg_list))
>>> print(result)
['name:', 'name:f=1', 'name:b=1', 'name:b=2', 'name1:f=1,joined=True', 'name2:b=2,
↪f=1,joined=True']
```

wbia.expt.cfghelpers.**remove_prefix_hack**(*cfg*, *cfgtype*, *cfg_options*, *alias_keys*)

1.6.4 wbia.expt.draw_helpers module

class wbia.expt.draw_helpers.**IndividualResultsCopyTaskQueue**

Bases: object

append_copy_task(*fpath_orig*, *dstdir=None*)

helper which copies a summary figure to root dir

flush_copy_tasks()

wbia.expt.draw_helpers.**make_individual_latex_figures**(*ibs*, *fpaths_list*, *flat_case_labels*, *cfgx2_shortlbl*, *case_figdir*, *analysis_fpath_list*)

1.6.5 wbia.expt.experiment_configs module

In this file dicts specify all possible combinations of the varied parameters and lists specify the union of parameters

Rename to pipe_cfgdef

wbia.expt.experiment_configs.**apply_CircQRH**(*cfg*)

wbia.expt.experiment_configs.**apply_Ell**(*cfg*)

wbia.expt.experiment_configs.**apply_EllQRH**(*cfg*)

wbia.expt.experiment_configs.**apply_k**(*cfg*)

wbia.expt.experiment_configs.**apply_knorm**(*cfg*)

wbia.expt.experiment_configs.**apply_param**(*cfg*, ***kwargs*)

wbia.expt.experiment_configs.**augbase**(*basedict*, *updatedict*)

wbia.expt.experiment_configs.**best**(*metadata*)
Infer the best pipeline config based on the metadata

wbia.expt.experiment_configs.**get_candidacy_dbnames**()

1.6.6 wbia.expt.experiment_drawing module

./dev.py -t custom:affine_invariance=False,adapteq=True,fg_on=False –db Elephants_drop1_ears –allgt –index=0:10 –guiview # NOQA

wbia.expt.experiment_drawing.**draw_annot_scoresep**(*ibs*, *testres*, *f=None*, *verbose=None*)
Draws the separation between true positive and true negative name scores.

Todo: plot the difference between the top true score and the next best false score?

CommandLine: ib python -m wbia draw_annot_scoresep –show python -m wbia draw_annot_scoresep –db PZ_MTEST –allgt -w –show –serial python -m wbia draw_annot_scoresep -t scores –db PZ_MTEST –allgt –show python -m wbia draw_annot_scoresep -t scores –db PZ_Master0 –allgt –show python -m wbia draw_annot_scoresep –db PZ_Master1 -a timectrl -t best –show python -m wbia draw_annot_scoresep –db PZ_Master1 -a timectrl -t best –show -f :without_tag=photobomb

Paper: python -m wbia draw_annot_scoresep –dbdir lev/media/hdd/golden/GGR-IBEIS -a timectrl –save gz_scoresep.png python -m wbia draw_annot_scoresep –dbdir lev/media/hdd/golden/GZGC -a timectrl:species=zebra_plains –save pz_scoresep.png python -m wbia draw_annot_scoresep –dbdir lev/media/hdd/golden/GZGC -a timectrl1h:species=giraffe_masai –save girm_scoresep.png

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.experiment_drawing import *  # NOQA
>>> from wbia.init import main_helpers
>>> defaultdb = 'PZ_MTEST'
>>> ibs, testres = main_helpers.testdata_expts(defaultdb, a=['timectrl'], t=['best
↪'])
>>> f = ut.get_argval(('--filt', '-f'), type_=list, default=[''])
>>> draw_annot_scoresep(ibs, testres, f=f, verbose=ut.VERBOSE)
>>> ut.show_if_requested()
```

Ignore: import IPython IPython.get_ipython().magic('pylab qt4')

wbia.expt.experiment_drawing.**draw_case_timedeltas**(*ibs*, *testres*, *falsepos=None*, *truepos=None*, *verbose=False*)

CommandLine: python -m wbia.dev -e draw_case_timedeltas –show python -m wbia.dev -e draw_case_timedeltas –show -t default

-a unctrl:num_names=1,name_offset=[1,2]

python -m wbia.dev -e draw_case_timedeltas –show -t default -a unctrl:num_names=1,name_offset=[1,2],joinme=1

python -m wbia.dev -e draw_case_timedeltas –show -t default

-a unctrl:num_names=1,name_offset=[1,2] unctrl:num_names=1,name_offset=[3,0]

python -m wbia.dev -e timedelta_hist –show -t baseline -a unctrl ctrl:force_const_size=True unctrl:force_const_size=True –consistent –db PZ_MTEST

Testing python -m wbia.dev -e timedelta_hist –show -t baseline

-a unctrl ctrl:force_const_size=True unctrl:force_const_size=True –consistent –db PZ_Master1

python -m wbia.dev -e timedelta_hist –show -t baseline -a unctrl ctrl:sample_rule_ref=max_timedelta –db PZ_Master1 –aidcfginfo

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.experiment_drawing import *  # NOQA
>>> from wbia.init import main_helpers
>>> ibs, testres = main_helpers.testdata_expts('PZ_MTEST')
>>> draw_case_timedeltas(ibs, testres)
>>> ut.show_if_requested()
```

wbia.expt.experiment_drawing.**draw_casetag_hist**(*ibs*, *testres*, *f=None*, *with_wordcloud=True*)

Parameters

- **ibs** (*wbia.IBEISController*) – wbia controller object
- **testres** (TestResult) – test result object

CommandLine: wbia –tf -draw_casetag_hist –show

Experiments I tagged wbia –tf -draw_casetag_hist -a timectrl -t invarbest –db PZ_Master1 –show

wbia -e taghist -a timectrl -t best –db PZ_Master1 –show

wbia -e taghist -a timequalctrl -t invarbest –db PZ_Master1 –show wbia -e taghist -a timequalctrl:minqual=good -t invarbest –db PZ_Master1 –show wbia -e taghist -a timequalctrl:minqual=good -t invarbest –db PZ_Master1 –show –filt :fail=True

Do more tagging wbia -e cases -a timequalctrl:minqual=good -t invarbest –db PZ_Master1

–filt :orderby=gfscore,reverse=1,min_gtrank=1,max_gf_tags=0 –show

wbia -e print -a timequalctrl:minqual=good -t invarbest –db PZ_Master1 –show wbia -e cases -a timequalctrl -t invarbest –db PZ_Master1

–filt :orderby=gfscore,reverse=1,max_gf_tags=0,:fail=True,min_gf_timedelta=12h –show

wbia -e cases -a timequalctrl -t invarbest –db PZ_Master1 –filt :orderby=gfscore,reverse=1,max_gf_tags=0,:fail=True,min_gf_timedelta=12h –show

python -m wbia -e taghist –db PZ_Master1 -a timectrl -t best –filt :fail=True –no-wordcloud –hargv=tags –prefix “Failure Case ” –label PZTags –figsize=10,3 –left=.2

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.experiment_drawing import *  # NOQA
>>> from wbia.init import main_helpers
>>> ibs, testres = main_helpers.testdata_expts('PZ_Master1', a=[
↪'timequalcontrolled'])
>>> f = ut.get_argval(('--filt', '-f'), type_=list, default=[''])
>>> draw_casetag_hist(ibs, testres, f=f)
>>> ut.show_if_requested()
```

wbia.expt.experiment_drawing.**draw_match_cases**(*ibs*, *testres*, *metadata=None*, *f=None*, *show_in_notebook=False*, *annot_modes=None*, *figsize=None*, *case_pos_list=None*, *verbose=None*, *interact=None*, *figdir=None*, ***kwargs*)

Parameters

- **ibs** (*wbia.IBEISController*) – wbia controller object
- **testres** (TestResult) – test result object
- **metadata** (*None*) – (default = None)

CommandLine: python -m wbia –tf draw_match_cases python -m wbia.dev -e draw_match_cases –figdir=figure python -m wbia.dev -e draw_match_cases –db PZ_Master1 -a ctrl

-t default –filt :fail=True,min_gtrank=5,gtrank_lt=20 –render

Shows the best results python -m wbia.dev -e cases –db PZ_Master1

-a timectrl -t invarbest –filt :sortasc=gtscore,success=True,index=200:201 –show

Shows failures sorted by gt score python -m wbia.dev -e cases –db PZ_Master1

-a timectrl -t invarbest –filt :sortdsc=gfscore,min_gtrank=1 –show

Find the untagged photobomb and scenery cases python -m wbia.dev -e cases –db PZ_Master1 -a timectrl

-t invarbest –show –filt :orderby=gfscore,reverse=1,min_gtrank=1,max_gf_td=24h,max_gf_tags=0

Find untagged failures python -m wbia.dev -e cases –db PZ_Master1 -a timectrl

-t invarbest –filt :orderby=gfscore,reverse=1,min_gtrank=1,max_gf_tags=0 –show

Show disagreement cases wbia –tf draw_match_cases –db PZ_MTEST -a default:size=20

-t default:K=[1,4] –filt :disagree=True,index=0:4 –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.experiment_drawing import *  # NOQA
>>> from wbia.init import main_helpers
>>> ibs, testres = main_helpers.testdata_expts('PZ_MTEST')
>>> filt_cfg = main_helpers.testdata_filtcfg()
>>> metadata = None
>>> figdir = ut.get_argval(('--figdir', '--dpath'), type_=str, default=None)
>>> analysis_fpath_list = draw_match_cases(ibs, testres, metadata,
>>>                                        f=filt_cfg, figdir=figdir)
>>> ut.show_if_requested()
```

wbia.expt.experiment_drawing.**draw_rank_cmc**(*ibs*, *testres*, *verbose=False*, *test_cfgx_slice=None*, *group_queries=False*, *draw_icon=True*, *numranks=5*, *kind='cmc'*, *cdfzoom=True*, ***kwargs*)

Parameters

- **ibs** (*wbia.IBEISController*) – wbia controller object
- **testres** (TestResult) –

Todo: # Cross-validated results with timectrl python -m wbia draw_rank_cmc –db PZ_MTEST –show -a timectrl:xval=True -t invar –kind=cmc

---

CommandLine: python -m wbia draw_rank_cmc python -m wbia draw_rank_cmc –db PZ_MTEST –show -a timectrl -t default –kind=cmc

python -m wbia draw_rank_cmc –db PZ_MTEST –show -a :proot=smk,num_words=64000 python -m wbia draw_rank_cmc –db PZ_MTEST –show -a ctrl -t best:prescore_method=csum python -m wbia draw_rank_cmc –db PZ_MTEST –show -a timectrl -t invar –kind=cmc –cdfzoom python -m wbia draw_rank_cmc –db PZ_MTEST –show -a varypername_td -t CircQRH_ScoreMech:K=3 #wbia -e rank_cmc –db lynx -a default:qsame_imageset=True,been_adjusted=True,excluderef=True -t default:K=1 –show

python -m wbia.dev -e draw_rank_cmc –db lynx -a default:qsame_imageset=True,been_adjusted=True,excluderef=True -t default:K=1 –show

python -m wbia –tf draw_rank_cmc -t best -a timectrl –db PZ_Master1 –show

python -m wbia –tf draw_rank_cmc –db PZ_Master1 –show -t best -a timectrl:qhas_any=(needswork,correctable,mildviewpoint),qhas_none=(viewpoint,photobomb,error:viewpoint,quality) –acfginfo –veryverbtd

wbia –tf draw_match_cases –db GZ_ALL -a ctrl -t default:K=1,resize_dim=[width],dim_size=[700,750] -f :sortdsc=gfscore,without_tag=scenerymatch,disagree=True –show

wbia –tf autogen_ipynb –db GZ_ALL –ipynb -a ctrl

-t default:K=1,resize_dim=[width],dim_size=[600,700,750] default:K=1,resize_dim=[area],dim_size=[450,550,6

wbia draw_rank_cmc –db GZ_ALL -a ctrl -t default –show wbia draw_match_cases –db GZ_ALL -a ctrl -t default -f :fail=True –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.experiment_drawing import *  # NOQA
>>> from wbia.init import main_helpers
>>> #ibs, testres = main_helpers.testdata_expts(
>>> #    'seaturtles', a='default2:qhas_any=(left),sample_occur=True,occur_
↪offset=[0,1,2,3,4,5,6,7,8],num_names=None')
>>> ibs, testres = main_helpers.testdata_expts('PZ_MTEST')
>>> kwargs = ut.argparse_funckw(draw_rank_cmc)
>>> result = draw_rank_cmc(ibs, testres, **kwargs)
>>> ut.show_if_requested()
>>> print(result)
```

wbia.expt.experiment_drawing.**draw_rank_surface**(*ibs*, *testres*, *verbose=None*, *fnum=None*)

Draws n dimensional data + a score / rank The rank is always on the y axis.

The first dimension is on the x axis. The second dimension is split over multiple plots. The third dimension becomes multiple lines. May need to clean this scheme up a bit.

Parameters

- **ibs** (*wbia.IBEISController*) – wbia controller object
- **testres** (TestResult) – test result object

CommandLine: wbia –tf draw_rank_surface –db PZ_Master1 -a varysize_td -t CircQRH_K –show

wbia –tf draw_rank_surface –show -t best -a varysize –db PZ_Master1 –show

wbia –tf draw_rank_surface –show -t CircQRH_K -a varysize_td –db PZ_Master1 –show wbia –tf draw_rank_surface –show -t CircQRH_K -a varysize_td –db PZ_Master1 –show

wbia –tf draw_rank_surface –show -t candidacy_k -a varysize –db PZ_Master1 –show –param-keys=K,dcfg_sample_per_name,dcfg_sample_size wbia –tf draw_rank_surface –show -t best

-a varynannots_td varynannots_td:qmin_pername=3,dpername=2 –db PZ_Master1 –show –param-keys=dcfg_sample_per_name,dcfg_sample_size

wbia –tf draw_rank_surface –show -t best -a varynannots_td –db PZ_Master1 –show –param-keys=dcfg_sample_size

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.experiment_drawing import *  # NOQA
>>> from wbia.init import main_helpers
>>> ibs, testres = main_helpers.testdata_expts('PZ_MTEST')
>>> result = draw_rank_surface(ibs, testres)
>>> ut.show_if_requested()
>>> print(result)
```

wbia.expt.experiment_drawing.**scorediff**(*ibs*, *testres*, *f=None*, *verbose=None*)

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **testres** (*wbia.TestResult*) – test result object
- **f** (*None*) – (default = None)
- **verbose** (*bool*) – verbosity flag(default = None)

CommandLine: python -m wbia.expt.experiment_drawing scorediff –db PZ_Master1 -a timectrl -t best –show python -m wbia.expt.experiment_drawing scorediff –db PZ_MTEST -a default -t best –show

python -m wbia.expt.experiment_drawing scorediff –db humpbacks_fb -a default:has_any=hasnotch,mingt=2 -t default:proot=BC_DTW,decision=max,crop_dim_size=500,crop_enabled=True,u –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.experiment_drawing import *  # NOQA
>>> from wbia.init import main_helpers
>>> defaultdb = 'PZ_MTEST'
>>> ibs, testres = main_helpers.testdata_expts(defaultdb, a=['timectrl'], t=['best
↪'])
```

(continues on next page)

(continued from previous page)

```
>>> f = ut.get_argval(('--filt', '-f'), type_=list, default=[''])
>>> scorediff(ibs, testres, f=f, verbose=ut.VERBOSE)
>>> ut.show_if_requested()
```

wbia.expt.experiment_drawing.**temp_multidb_cmc**()
Plots multiple database CMC curves in the same plot for the AI for social good paper

wbia.expt.experiment_drawing.**temp_num_exmaples_cmc**()

1.6.7 wbia.expt.experiment_helpers module

Helper module that helps expand parameters for grid search TODO: move into custom pipe_cfg and annot_cfg modules

wbia.expt.experiment_helpers.**filter_duplicate_acfgs**(*expanded_aids_list*, *acfg_list*, *acfg_name_list*, *verbose=None*)
Removes configs with the same expanded aids list
CommandLine: # The following will trigger this function: wbia -m wbia get_annotcfg_list:0 -a timectrl timectrl:view=left –db PZ_MTEST

wbia.expt.experiment_helpers.**get_annotcfg_list**(*ibs*, *acfg_name_list*, *filter_dups=True*, *qaid_override=None*, *daid_override=None*, *initial_aids=None*, *use_cache=None*, *verbose=None*)
For now can only specify one acfg name list

TODO: move to filter_annots
Parameters **annot_cfg_name_list** (*list*) –

CommandLine: python -m wbia get_annotcfg_list:0 python -m wbia get_annotcfg_list:1 python -m wbia get_annotcfg_list:2

wbia get_annotcfg_list:0 –ainfo wbia get_annotcfg_list:0 –db NNP_Master3 -a viewpoint_compare –nocache-aid –verbtd wbia get_annotcfg_list:0 –db PZ_ViewPoints -a viewpoint_compare –nocache-aid –verbtd wbia get_annotcfg_list:0 –db PZ_MTEST -a unctrl ctrl::unctrl –ainfo –nocache-aid wbia get_annotcfg_list:0 –db testdb1 -a : –ainfo –nocache-aid wbia get_annotcfg_list:0 –db Oxford -a :qhas_any=query –ainfo –nocache-aid wbia get_annotcfg_list:0 –db Oxford -a :qhas_any=query,dhas_any=distractor –ainfo –nocache-aid

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.experiment_helpers import *  # NOQA
>>> import wbia
>>> from wbia.expt import annotation_configs
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> filter_dups = not ut.get_argflag('--nofilter-dups')
>>> acfg_name_list = testdata_acfg_names()
>>> _tup = get_annotcfg_list(ibs, acfg_name_list, filter_dups)
>>> acfg_list, expanded_aids_list = _tup
>>> print('\n PRINTING TEST RESULTS')
>>> result = ut.repr2(acfg_list, nl=3)
>>> print('\n')
>>> #statskw = ut.parse_func_kwarg_keys(ibs.get_annot_stats_dict, with_vals=False)
```

(continues on next page)

(continued from previous page)

```
>>> printkw = dict(combined=True, per_name_vpedge=None,
>>>                per_qual=False, per_vp=False, case_tag_hist=False)
>>> annotation_configs.print_acfg_list(
>>>     acfg_list, expanded_aids_list, ibs, **printkw)
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.expt.experiment_helpers import *  # NOQA
>>> import wbia
>>> from wbia.init import main_helpers
>>> from wbia.expt import annotation_configs
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> aids = ibs.get_valid_aids()
>>> main_helpers.monkeypatch_encounters(ibs, aids, days=50)
>>> a = ['default:crossval_enc=True,require_timestamp=True']
>>> acfg_name_list = testdata_acfg_names(a)
>>> acfg_list, expanded_aids_list = get_annotcfg_list(ibs, acfg_name_list)
>>> annotation_configs.print_acfg_list(acfg_list, expanded_aids_list)
>>> # Restore state
>>> main_helpers.unmonkeypatch_encounters(ibs)
```

wbia.expt.experiment_helpers.**get_pipecfg_list**(*test_cfg_name_list*, *ibs=None*, *verbose=None*)

Builds a list of varied query configurations. Only custom configs depend on an ibs object. The order of the output is not gaurenteed to aggree with input order.

FIXME: This breaks if you proot=BC_DTW and ibs is None

Parameters

- **test_cfg_name_list** (*list*) – list of strs
- **ibs** (*wbia.IBEISController*) – wbia controller object (optional)

Returns

(cfg_list, cfgx2_lbl) - cfg_list (list): list of config objects cfgx2_lbl (list): denotes which parameters are being varied.

If there is just one config then nothing is varied

Return type tuple

CommandLine: python -m wbia get_pipecfg_list:0 python -m wbia get_pipecfg_list:1 –db humpbacks python -m wbia get_pipecfg_list:2

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.expt.experiment_helpers import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> #test_cfg_name_list = ['best', 'custom', 'custom:sv_on=False']
>>> #test_cfg_name_list = ['default', 'default:sv_on=False', 'best']
>>> test_cfg_name_list = ['default', 'default:sv_on=False', 'best']
>>> # execute function
```

(continues on next page)

(continued from previous page)

```
>>> (pcfgdict_list, pipecfg_list) = get_pipecfg_list(test_cfg_name_list, ibs)
>>> # verify results
>>> assert pipecfg_list[0].sv_cfg.sv_on is True
>>> assert pipecfg_list[1].sv_cfg.sv_on is False
>>> pipecfg_lbls = get_varied_pipecfg_lbls(pcfgdict_list)
>>> result = ('pipecfg_lbls = '+ ut.repr2(pipecfg_lbls))
>>> print(result)
pipecfg_lbls = ['default:', 'default:sv_on=False']
```

Example

```
>>> # DISABLE_DOCTEST
>>> import wbia_flukematch.plugin
>>> from wbia.expt.experiment_helpers import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='humpbacks')
>>> test_cfg_name_list = ['default:pipeline_root=BC_DTW,decision=average,crop_dim_
↪size=[960,500]', 'default:K=[1,4]']
>>> (pcfgdict_list, pipecfg_list) = get_pipecfg_list(test_cfg_name_list, ibs)
>>> pipecfg_lbls = get_varied_pipecfg_lbls(pcfgdict_list)
>>> result = ('pipecfg_lbls = '+ ut.repr2(pipecfg_lbls))
>>> print(result)
>>> print_pipe_configs(pcfgdict_list, pipecfg_list)
```

wbia.expt.experiment_helpers.**get_varied_pipecfg_lbls**(*cfgdict_list*, *pipecfg_list=None*)

wbia.expt.experiment_helpers.**parse_acfg_combo_list**(*acfg_name_list*)

Parses the name list into a list of config dicts

Parameters **acfg_name_list** (*list*) – a list of annotation config strings

Returns acfg_combo_list

Return type list

CommandLine: python -m wbia parse_acfg_combo_list:0 python -m wbia parse_acfg_combo_list:1 python -m wbia parse_acfg_combo_list:2

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.expt.experiment_helpers import *  # NOQA
>>> import wbia
>>> from wbia.expt import annotation_configs
>>> acfg_name_list = testdata_acfg_names(['default', 'uncontrolled'])
>>> acfg_combo_list = parse_acfg_combo_list(acfg_name_list)
>>> acfg_list = ut.flatten(acfg_combo_list)
>>> printkw = dict()
>>> annotation_configs.print_acfg_list(acfg_list, **printkw)
>>> result = ut.repr2(sorted(acfg_list[0].keys()))
>>> print(result)
['dcfg', 'qcfg']
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.expt.experiment_helpers import *  # NOQA
>>> import wbia
>>> from wbia.expt import annotation_configs
>>> # double colon :: means expand consistently and force const size
>>> acfg_name_list = testdata_acfg_names(['unctrl', 'ctrl::unctrl'])
>>> acfg_name_list = testdata_acfg_names(['unctrl', 'varysize', 'ctrl::unctrl'])
>>> acfg_name_list = testdata_acfg_names(['unctrl', 'varysize', 'ctrl::varysize',
↪'ctrl::unctrl'])
>>> acfg_combo_list = parse_acfg_combo_list(acfg_name_list)
>>> acfg_list = ut.flatten(acfg_combo_list)
>>> printkw = dict()
>>> annotation_configs.print_acfg_list(acfg_list, **printkw)
```

wbia.expt.experiment_helpers.**print_pipe_configs**(*cfgdict_list*, *pipecfg_list*)

wbia.expt.experiment_helpers.**testdata_acfg_names**(*default_acfg_name_list=['default']*)

1.6.8 wbia.expt.experiment_printres module

displays results from harness

TODO: save a testres variable so reloading and regenration becomes easier.

wbia.expt.experiment_printres.**get_diffmat_str**(*rank_mat*, *qaids*, *nConfig*)

wbia.expt.experiment_printres.**get_diffranks**(*rank_mat*, *qaids*)
Find rows which scored differently over the various configs FIXME: duplicated

wbia.expt.experiment_printres.**print_latexsum**(*ibs*, *testres*, *verbose=True*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **testres** –

CommandLine: python -m wbia.expt.experiment_printres –exec-print_latexsum python -m wbia.scripts.gen_cand_expts –exec-gen_script

python -m wbia –tf print_latexsum -t candidacy –db PZ_Master0 -a controlled –rank-lt-list=1,5,10,100
python -m wbia –tf print_latexsum -t candidacy –db PZ_MTEST -a controlled –rank-lt-list=1,5,10,100

Example

```
>>> # SCRIPT
>>> from wbia.expt.experiment_printres import *  # NOQA
>>> from wbia.init import main_helpers
>>> ibs, testres = main_helpers.testdata_expts()
>>> tabular_str2 = print_latexsum(ibs, testres)
```

wbia.expt.experiment_printres.**print_results**(*ibs*, *testres*, ***kwargs*)
Prints results from an experiment harness run. Rows store different qaids (query annotation ids) Cols store different configurations (algorithm parameters)

TODO: join acfgs

Parameters

- **ibs** (IBEISController) – wbia controller object
- **testres** (test_result.TestResult) –

CommandLine:

python dev.py -e print –db PZ_MTEST -a default:dpername=1,qpername=[1,2] -t default:fg_on=False

python dev.py -e print -t best –db seals2 –allgt –vz python dev.py -e print –db PZ_MTEST –allgt -t custom –print-confusion-stats

python dev.py -e print –db PZ_MTEST –allgt –noqcache –index 0:10:2 -t custom:rrvsone_on=True –print-confusion-stats

python dev.py -e print –db PZ_MTEST –allgt –noqcache –qaid4 -t custom:rrvsone_on=True –print-confusion-stats

python -m wbia print_results -t default –db PZ_MTEST -a ctrl python -m wbia print_results -t default –db PZ_MTEST -a ctrl python -m wbia print_results –db PZ_MTEST -a default -t default:lnbnn_on=True default:lnbnn_on=False,bar_l2_on=True default:lnbnn_on=False,normonly_on=True

CommandLine: python -m wbia.expt.experiment_printres –test-print_results utprof.py -m wbia.expt.experiment_printres –test-print_results

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.experiment_printres import *  # NOQA
>>> from wbia.init import main_helpers
>>> ibs, testres = main_helpers.testdata_expts(
>>>     'pz_mtest', a='default:dpername=1,qpername=[1,2]',
>>>     t='default:fg_on=false')
>>> result = print_results(ibs, testres)
>>> print(result)
```

wbia.expt.experiment_printres.**rankscore_str**(*thresh*, *nLess*, *total*, *withlbl=True*)

1.6.9 wbia.expt.harness module

Runs many queries and keeps track of some results

wbia.expt.harness.**make_single_testres**(*ibs*, *qaids*, *daids*, *pipecfg_list*, *cfgx2_lbl*, *cfgdict_list*, *lbl*, *testnameid*, *use_cache=None*, *subindexer_partial=<class 'utool.util_progress.ProgIter'>*)

CommandLine: python -m wbia run_expt

wbia.expt.harness.**run_expt**(*ibs*, *acfg_name_list*, *test_cfg_name_list*, *use_cache=None*, *qaid_override=None*, *daid_override=None*, *initial_aids=None*)

Loops over annot configs.

Try and use this function as a starting point to clean up this module. The code is getting too untenable.

CommandLine: python -m wbia.expt.harness run_expt –acfginfo python -m wbia.expt.harness run_expt –pcfginfo python -m wbia.expt.harness run_expt

Ignore: test_cfg_name_list = [p]

Example

```
>>> # SLOW_DOCTEST
>>> from wbia.expt.harness import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> default_acfgstrs = ['ctrl:qsize=20,dpername=1,dsize=10',
>>>                     'ctrl:qsize=20,dpername=10,dsize=20']
>>> acfg_name_list = default_acfgstrs
>>> test_cfg_name_list = ['default:proot=smk', 'default']
>>> #test_cfg_name_list = ['custom', 'custom:fg_on=False']
>>> use_cache = False
>>> testres_list = run_expt(ibs, acfg_name_list, test_cfg_name_list, use_cache)
```

1.6.10 wbia.expt.old_storage module

1.6.11 wbia.expt.test_result module

class wbia.expt.test_result.**TestResult**(*cfg_list*, *cfgx2_lbl*, *cfgx2_cmsinfo*, *cfgx2_qreq_*)

Bases: utool.util_dev.NiceRepr

CommandLine: export SMK_PIPE="smk:nwords=[64000],sv=[False]" wbia TestResult –db PZ_MTEST -a ctrl -p $SMK_PIPE wbia TestResult –db Oxford -a oxford -p $SMK_PIPE

Example

```
>>> # Script
>>> from wbia.init import main_helpers
>>> import utool as ut
>>> ibs, testres = main_helpers.testdata_expts()
>>> testres.help()
>>> actions = testres.get_actions()
>>> testres.map_score()
>>> ut.qtensure()
>>> prompt = ut.InteractivePrompt(actions)
>>> prompt.loop()
```

case_sample2(*filt_cfg*, *qaids=None*, *return_mask=False*, *verbose=None*)

Filters individual test result cases based on how they performed, what tags they had, and various other things.

Parameters **filt_cfg** (*dict*) –

Returns case_pos_list (list of (qx, cfgx)) or isvalid mask

Return type list

CommandLine: python -m wbia TestResult.case_sample2 python -m wbia TestResult.case_sample2:0 python -m wbia TestResult.case_sample2:1 –db GZ_ALL –filt :min_tags=1 python -m wbia TestResult.case_sample2:1 –db PZ_Master1 –filt :min_gf_tags=1

python -m wbia TestResult.case_sample2:2 –db PZ_Master1

Example

```
>>> # DISABLE_DOCTEST
>>> # The same results is achievable with different filter config settings
>>> from wbia.expt.test_result import *  # NOQA
>>> from wbia.init import main_helpers
>>> verbose = True
>>> ibs, testres = main_helpers.testdata_expts('PZ_MTEST', a=['ctrl'])
>>> filt_cfg1 = {'fail': True}
>>> case_pos_list1 = testres.case_sample2(filt_cfg1)
>>> filt_cfg2 = {'min_gtrank': 1}
>>> case_pos_list2 = testres.case_sample2(filt_cfg2)
>>> filt_cfg3 = {'min_gtrank': 0}
>>> case_pos_list3 = testres.case_sample2(filt_cfg3)
>>> filt_cfg4 = {}
>>> case_pos_list4 = testres.case_sample2(filt_cfg4)
>>> assert np.all(case_pos_list1 == case_pos_list2), 'should be equiv configs
↪'
>>> assert np.any(case_pos_list2 != case_pos_list3), 'should be diff configs'
>>> assert np.all(case_pos_list3 == case_pos_list4), 'should be equiv configs
↪'
>>> ibs, testres = main_helpers.testdata_expts('PZ_MTEST', a=['ctrl'], t=[
↪'default:sv_on=[True,False]'])
>>> filt_cfg5 = filt_cfg1.copy()
>>> mask5 = testres.case_sample2(filt_cfg5, return_mask=True)
>>> case_pos_list5 = testres.case_sample2(filt_cfg5, return_mask=False)
>>> assert len(mask5.shape) == 2
>>> assert np.all(mask5.T[0] == mask5.T[1])
>>> filt_cfg6 = {'fail': True, 'allcfg': True}
>>> mask6 = testres.case_sample2(filt_cfg6, return_mask=True)
>>> assert np.all(mask6.T[0] == mask6.T[1])
>>> print(mask5)
>>> print(case_pos_list5)
>>> filt_cfg = filt_cfg7 = {'disagree': True}
>>> case_pos_list7 = testres.case_sample2(filt_cfg7, verbose=verbose)
>>> print(case_pos_list7)
```

Example

```
>>> # SCRIPT
>>> from wbia.expt.test_result import *  # NOQA
>>> from wbia.init import main_helpers
>>> ibs, testres = main_helpers.testdata_expts('PZ_MTEST', a=['ctrl'])
>>> filt_cfg = main_helpers.testdata_filtcfg()
>>> case_pos_list = testres.case_sample2(filt_cfg)
>>> result = ('case_pos_list = %s' % (str(case_pos_list),))
>>> print(result)
>>> # Extra stuff
>>> all_tags = testres.get_all_tags()
>>> selcted_tags = ut.take(all_tags, case_pos_list.T[0])
>>> print('selcted_tags = %r' % (selcted_tags,))
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> from wbia.init import main_helpers
>>> ibs, testres = main_helpers.testdata_expts('PZ_MTEST', a=['ctrl'], t=[
↪'default:K=[1,2,3]'])
>>> ut.exec_funckw(testres.case_sample2, globals())
>>> filt_cfg = {'fail': True, 'min_gtrank': 1, 'max_gtrank': None, 'min_gf_
↪timedelta': '24h'}
>>> ibs, testres = main_helpers.testdata_expts('humpbacks_fb', a=[
↪'default:has_any=hasnotch,mingt=2,qindex=0:300,dindex=0:300'], t=[
↪'default:proot=BC_DTW,decision=max,crop_dim_size=500,crop_enabled=True,
↪manual_extract=False,use_te_scorer=True,ignore_notch=True,te_net=annot_
↪simple', 'default:proot=vsmany'], qaid_override=[12])
>>> filt_cfg = ':disagree=True,index=0:8,min_gtscore=.00001,require_all_
↪cfg=True'
>>> #filt_cfg = cfghelpers.parse_argv_cfg('--filt')[0]
>>> case_pos_list = testres.case_sample2(filt_cfg, verbose=True)
>>> result = ('case_pos_list = %s' % (str(case_pos_list),))
>>> print(result)
>>> # Extra stuff
>>> all_tags = testres.get_all_tags()
>>> selcted_tags = ut.take(all_tags, case_pos_list.T[0])
>>> print('selcted_tags = %r' % (selcted_tags,))
```

logger.info(‘qaid = %r’ % (qaid,)) logger.info(‘qx = %r’ % (qx,)) logger.info(‘cfgxs = %r’ % (cfgxs,)) # print testres info about this item take_cfgs = ut.partial(ut.take, index_list=cfgxs) take_qx = ut.partial(ut.take, index_list=qx) truth_cfgs = ut.hmap_vals(take_qx, truth2_prop) truth_item = ut.hmap_vals(take_cfgs, truth_cfgs, max_depth=1) prop_cfgs = ut.hmap_vals(take_qx, prop2_mat) prop_item = ut.hmap_vals(take_cfgs, prop_cfgs, max_depth=0) logger.info(‘truth2_prop[item] = ‘ + ut.repr3(truth_item, nl=2)) logger.info(‘prop2_mat[item] = ‘ + ut.repr3(prop_item, nl=1))

cfgx2_daids

cfgx2_qaids

draw_failure_cases(***kwargs*)

```
>>> from wbia.other.dbinfo import *  # NOQA
>>> import wbia
>>> ibs, testres = wbia.testdata_expts(defaultdb='PZ_MTEST', a=
↪'timectrl:qsize=2', t='invar:AI=[False],RI=False', use_cache=False)
```

draw_match_cases(***kwargs*)
Wrapper

draw_rank_cmc()
Wrapper

draw_score_diff_disti()

CommandLine: python -m wbia –tf TestResult.draw_score_diff_disti –show -a varynannots_td -t best –db PZ_Master1 python -m wbia –tf TestResult.draw_score_diff_disti –show -a varynannots_td -t best –db GZ_Master1 python -m wbia –tf TestResult.draw_score_diff_disti –show -a varynannots_td1h -t best –db GIRM_Master1

python -m wbia –tf TestResult.draw_score_diff_disti –show -a varynannots_td:qmin_pername=3,dpername=2 -t best –db PZ_Master1

python -m wbia –tf get_annotcfg_list -a varynannots_td -t best –db PZ_Master1 13502 python -m wbia –tf draw_match_cases –db PZ_Master1 -a varynannots_td:dsample_size=.01 -t best –show –qaid 13502 python -m wbia –tf draw_match_cases –db PZ_Master1 -a varynannots_td -t best –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> import wbia
>>> ibs, testres = wbia.testdata_expts('PZ_Master1', a=['varynannots_td'],␣
↪t=['best'])
>>> result = testres.draw_score_diff_disti()
>>> print(result)
>>> ut.show_if_requested()
```

embed_testres()

CommandLine: python -m wbia TestResults.embed_testres

Example

```
>>> # SCRIPT
>>> from wbia.expt.test_result import *  # NOQA
>>> from wbia.init import main_helpers
>>> ibs, testres = main_helpers.testdata_expts(defaultdb='PZ_MTEST')
>>> embed_testres(testres)
```

find_score_thresh_cutoff()
FIXME DUPLICATE CODE rectify with experiment_drawing

classmethod from_cms(*cm_list*, *qreq_*)

get_X_LIST()
DEPRICATE or refactor

get_actions()

get_all_qaids()

get_all_tags()

CommandLine: python -m wbia –tf TestResult.get_all_tags –db PZ_Master1 –show –filt : python -m wbia –tf TestResult.get_all_tags –db PZ_Master1 –show –filt :min_gf_timedelta=24h python -m wbia –tf TestResult.get_all_tags –db PZ_Master1 –show –filt :min_gf_timedelta=24h,max_gt_rank=5

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> from wbia.init import main_helpers
>>> ibs, testres = main_helpers.testdata_expts('PZ_Master1', a=['timectrl'])
>>> filt_cfg = main_helpers.testdata_filtcfg()
>>> case_pos_list = testres.case_sample2(filt_cfg)
```

(continues on next page)

(continued from previous page)

```
>>> all_tags = testres.get_all_tags()
>>> selected_tags = ut.take(all_tags, case_pos_list.T[0])
>>> flat_tags = list(map(str, ut.flatten(ut.flatten(selected_tags))))
>>> print(ut.repr2(ut.dict_hist(flat_tags), key_order_metric='val'))
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.word_histogram2(flat_tags, fnum=1, pnum=(1, 2, 1))
>>> pt.wordcloud(' '.join(flat_tags), fnum=1, pnum=(1, 2, 2))
>>> pt.set_figtitle(ut.get_cfg_lbl(filt_cfg))
>>> ut.show_if_requested()
```

get_all_varied_params()

Returns the parameters that were varied between different configurations in this test

Returns varied_params

Return type list

CommandLine: python -m wbia TestResult.get_all_varied_params

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> import wbia
>>> testres = wbia.testdata_expts(
>>>     'PZ_MTEST', t='default:K=[1,2]')[1]
>>> varied_params = sorted(testres.get_all_varied_params())
>>> result = ('varied_params = %s' % (ut.repr2(varied_params),))
>>> print(result)
varied_params = ['K', '_cfgindex']
```

get_annotcfg_args()

CommandLine: # TODO: More robust fix # To reproduce the error wbia -e rank_cmc –db humpbacks_fb -a default:mingt=2,qsize=10,dsize=100 default:qmingt=2,qsize=10,dsize=100 -t default:proot=BC_DTW,decision=max,crop_dim_size=500,crop_enabled=True,manual_extract=False,use_te_scorer=T –show

get_cfgstr(*cfgx*)

just dannots and config_str

get_cfgx_groupxs()

Returns the group indices of configurations specified to be joined.

Ignore: a = [‘default:minqual=good,require_timestamp=True,view=left,crossval_enc=True,joinme=1’, ‘default:minqual=good,require_timestamp=True,view=right,crossval_enc=True,joinme=1’, ‘default:minqual=ok,require_timestamp=True,view=left,crossval_enc=True,joinme=2’, ‘default:minqual=ok,require_timestamp=True,view=right,crossval_enc=True,joinme=2’,] >>> a = [>>> ‘default:minqual=good,require_timestamp=True,view=left,crossval_enc=True,joinme=1’, >>> ‘default:minqual=good,require_timestamp=True,view=right,crossval_enc=True,joinme=1’, >>> ‘default:minqual=ok,require_timestamp=True,view=left,crossval_enc=True,joinme=2’, >>> ‘default:minqual=ok,require_timestamp=True,view=right,crossval_enc=True,joinme=2’, >>>] >>> from wbia.init import main_helpers >>> #a = ‘default:minqual=good,require_timestamp=True,crossval_enc=True,view=[right,left]’ >>> t =

'default:K=[1]' >>> ibs, testres = main_helpers.testdata_expts('WWF_Lynx_Copy', a=a, t=t) >>> testres.get_cfgx_groupxs()

ut.lmap(sum, ut.apply_grouping([len(ut.unique(ibs.annots(aids).nids)) for aids in testres.cfgx2_qaids], testres.get_cfgx_groupxs())) ut.lmap(sum, ut.apply_grouping([len(ut.unique(ibs.annots(aids))) for aids in testres.cfgx2_qaids], testres.get_cfgx_groupxs()))

Example

```
>>> # xdoctest: +REQUIRES(--slow)
>>> # ENABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> from wbia.init import main_helpers
>>> ibs, testres = main_helpers.testdata_expts(
>>>    'PZ_MTEST',
>>>     a=['default:qnum_names=1,qname_offset=[0,1],joinme=1,dpername=1',
>>>        'default:qsize=1,dpername=[1,2]'],
>>>     t=['default:K=[1,2]'])
>>> groupxs = testres.get_cfgx_groupxs()
>>> result = groupxs
>>> print(result)
[[6], [4], [0, 2], [7], [5], [1, 3]]
```

get_cfgx_with_param(*key*, *val*)
Gets configs where the given parameter is held constant

get_common_qaids()

get_fname_aug(***kwargs*)

get_full_cfgstr(*cfgx*)
both qannots and dannots included

get_gf_tags()

Returns case_pos_list

Return type list

CommandLine: python -m wbia –tf TestResult.get_gf_tags –db PZ_Master1 –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> from wbia.init import main_helpers
>>> ibs, testres = main_helpers.testdata_expts('PZ_Master1', a=['timectrl'])
>>> filt_cfg = main_helpers.testdata_filtcfg()
>>> case_pos_list = testres.case_sample2(filt_cfg)
>>> gf_tags = testres.get_gf_tags()
```

get_gt_annot_tags()

get_gt_tags()

get_gtquery_annot_tags()

get_infoprop_list(*key*, *qaids=None*)
key = 'qx2_gt_rank' key = 'qx2_gt_rank' qaids = testres.get_test_qaids()

get_infoprop_mat(*key*, *qaids=None*)
key = 'qx2_gf_raw_score' key = 'qx2_gt_raw_score'

get_nLessX_dict()
Build a (histogram) dictionary mapping X (as in #ranks < X) to a list of cfg scores

get_options()

get_param_basis(*key*)
Returns what a param was varied between over all tests key = 'K' key = 'dcfg_sample_size'

get_param_val_from_cfgx(*cfgx*, *key*)

get_pipecfg_args()

get_query_annot_tags()

get_rank_histogram_bins()
easy to see histogram bins

get_rank_histograms(*bins=None*, *key=None*, *join_acfgs=False*)

Ignore: testres.get_infoprop_mat('qnx2_gt_name_rank') testres.get_infoprop_mat('qnx2_gf_name_rank') testres.get_infoprop_mat('qnx2_qnid')

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> from wbia.init import main_helpers
>>> ibs, testres = main_helpers.testdata_expts('testdb1', a=['default'])
>>> bins = 'dense'
>>> key = 'qnx2_gt_name_rank'
>>> config_hists = testres.get_rank_histograms(bins, key=key)
```

get_rank_mat(*qaids=None*)

get_rank_percentage_cumhist(*bins='dense'*, *key=None*, *join_acfgs=False*)

Parameters

- **bins** (*unicode*) – (default = u'dense')
- **key** (*None*) – (default = None)
- **join_acfgs** (*bool*) – (default = False)

Returns (config_cdfs, edges)

Return type tuple

CommandLine: python -m wbia –tf TestResult.get_rank_percentage_cumhist python -m wbia –tf TestResult.get_rank_percentage_cumhist

-t baseline -a unctrl ctrl

python -m wbia –tf TestResult.get_rank_percentage_cumhist –db lynx -a default:qsame_imageset=True,been_adjusted=True,excluderef=True -t default:K=1 –show –cmd

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> from wbia.init import main_helpers
>>> ibs, testres = main_helpers.testdata_expts(
>>>     'testdb1', a=['default:num_names=1,name_offset=[0,1]'])
>>> bins = u'dense'
>>> key = None
>>> (config_cdfs, edges) = testres.get_rank_percentage_cumhist(bins)
>>> result = ('(config_cdfs, edges) = %s' % (str((config_cdfs, edges)),))
>>> print(result)
```

get_short_cfglbls(*join_acfgs=False*)

Labels for published tables

cfg_lbls = [‘baseline:nRR=200+default:’, ‘baseline:+default:’]

CommandLine: python -m wbia –tf TestResult.get_short_cfglbls

Example

```
>>> # SLOW_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> import wbia
>>> ibs, testres = wbia.testdata_expts('PZ_MTEST', a=['ctrl:size=10'],
>>>                                    t=['default:dim_size=[450,550]'])
>>> cfg_lbls = testres.get_short_cfglbls()
>>> result = ('cfg_lbls = %s' % (ut.repr2(cfg_lbls),))
>>> print(result)
cfg_lbls = [
    'default:dim_size=450+ctrl',
    'default:dim_size=550+ctrl',
]
```

get_sorted_config_labels()

helper

get_test_qaids()

get_title_aug(*with_size=True*, *with_db=True*, *with_cfg=True*, *friendly=False*)

Parameters **with_size** (*bool*) – (default = True)

Returns title_aug

Return type str

CommandLine: python -m wbia –tf TestResult.get_title_aug –db PZ_Master1 -a timequalctrl::timectrl

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> import wbia
>>> ibs, testres = wbia.testdata_expts('PZ_MTEST')
```

(continues on next page)

(continued from previous page)

```
>>> with_size = True
>>> title_aug = testres.get_title_aug(with_size)
>>> res = u'title_aug = %s' % (title_aug,)
>>> print(res)
```

get_total_num_varied_params()

get_truth2_prop(*qaids=None*, *join_acfg=False*)

Returns (truth2_prop, prop2_mat)

Return type tuple

CommandLine: python -m wbia.expt.test_result –exec-get_truth2_prop –show

Example

```
>>> # xdoctest: +REQUIRES(--slow)
>>> # ENABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> import wbia
>>> ibs, testres = wbia.testdata_expts('PZ_MTEST', a=['ctrl'])
>>> (truth2_prop, prop2_mat) = testres.get_truth2_prop()
>>> result = '(truth2_prop, prop2_mat) = %s' % str((truth2_prop, prop2_mat))
>>> print(result)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

get_varied_labels(*shorten=False*, *join_acfgs=False*, *sep=''*)

Returns labels indicating only the parameters that have been varied between different annot/pipeline configurations.

Helper for consistent figure titles

CommandLine: python -m wbia –tf TestResult.make_figtitle –prefix “Seperability ” –db GIRM_Master1 -a timectrl -t Ell:K=2 –hargv=scores python -m wbia –tf TestResult.make_figtitle python -m wbia TestResult.get_varied_labels

Example

```
>>> # SLOW_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> import wbia
>>> ibs, testres = wbia.testdata_expts(
>>>     'PZ_MTEST', t='default:K=[1,2]',
>>>     #a=['timectrl:qsize=[1,2],dsize=[3,4]']
>>>     a=[
>>>        'default:qsize=[1,2],dsize=2,joinme=1,view=left',
>>>        'default:qsize=2,dsize=3,joinme=1,view=primary',
>>>        'default:qsize=[3,2],dsize=4,joinme=2,view=left',
>>>        'default:qsize=4,dsize=5,joinme=2,view=primary',
>>>       ]
>>> )
>>> # >>> ibs, testres = wbia.testdata_expts(
```

(continues on next page)

(continued from previous page)

```
>>> # >>>     'WWF_Lynx_Copy', t='default:K=1',
>>> # >>>     a=[
>>> # >>>         'default:minqual=good,require_timestamp=True,view=left,
↪dcrossval_enc=1,joinme=1',
>>> # >>>         'default:minqual=good,require_timestamp=True,view=left,
↪dcrossval_enc=2,joinme=2',
>>> # >>>         #'default:minqual=good,require_timestamp=True,view=left,
↪dcrossval_enc=3,joinme=3',
>>> # >>>         'default:minqual=good,require_timestamp=True,view=right,
↪dcrossval_enc=1,joinme=1',
>>> # >>>         'default:minqual=good,require_timestamp=True,view=right,
↪dcrossval_enc=2,joinme=2',
>>> # >>>         #'default:minqual=good,require_timestamp=True,view=right,
↪dcrossval_enc=3,joinme=3',
>>> # >>>       ]
>>> # >>> )
>>> varied_lbls = testres.get_varied_labels(shorten=False, join_acfgs=True)
>>> result = ('varied_lbls = %s' % (ut.repr2(varied_lbls, strvals=True,␣
↪nl=2),))
>>> print(result)
```

varied_lbls = [u'K=1+qsize=1', u'K=2+qsize=1', u'K=1+qsize=2', u'K=2+qsize=2']

get_worst_possible_rank()

has_constant_daids()

has_constant_length_daids()

has_constant_length_qaids()

has_constant_qaids()

help()

ibs

interact_individual_result(*qaid*, *cfgx=0*)

make_figtitle(*plotname=''*, *filt_cfg=None*)

Helper for consistent figure titles

CommandLine: python -m wbia –tf TestResult.make_figtitle –prefix "Seperability " –db GIRM_Master1 -a timectrl -t Ell:K=2 –hargv=scores python -m wbia –tf TestResult.make_figtitle

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> import wbia
>>> ibs, testres = wbia.testdata_expts('PZ_MTEST')
>>> plotname = ''
>>> figtitle = testres.make_figtitle(plotname)
>>> result = ('figtitle = %r' % (figtitle,))
>>> print(result)
```

map_score()

For each query compute a precision recall curve. Then, for each query compute the average precision. Then take the mean of all average precisions to obtain the mAP.

Script:

```
>>> #ibs = wbia.opendb('Oxford')
>>> #ibs, testres = wbia.testdata_expts('Oxford', a='oxford', p=
↪'smk:nWords=[64000],nAssign=[1],SV=[False,True]')
>>> import wbia
>>> ibs, testres = wbia.testdata_expts('Oxford', a='oxford', p=
↪'smk:nWords=[64000],nAssign=[1],SV=[False,True],can_match_sameimg=True
↪')
>>> import wbia
>>> ibs, testres = wbia.testdata_expts('Oxford', a='oxford', p=
↪'smk:nWords=[64000],nAssign=[1],SV=[False],can_match_sameimg=True')
```

nConfig

nQuery

print_acfg_info(***kwargs*)

Prints verbose information about the annotations used in each test configuration

CommandLine: python -m wbia –tf TestResult.print_acfg_info

Kwargs: see ibs.get_annot_stats_dict hashid, per_name, per_qual, per_vp, per_name_vpedge, per_image, min_name_hourdist

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> import wbia
>>> ibs, testres = wbia.testdata_expts('PZ_MTEST',
>>>                                    a=['ctrl::unctrl_comp'],
>>>                                    t=['candk:K=[1,2]'])
>>> ibs = None
>>> result = testres.print_acfg_info()
>>> print(result)
```

print_config_overlap(*with_plot=True*)

print_pcfg_info()

Prints verbose information about each pipeline configuration

```
>>> from wbia.expt.test_result import *  # NOQA
```

print_percent_identification_success()

Prints names identified (at rank 1) / names queried. This combines results over multiple queries of a particular name using max

OLD, MAYBE DEPRIATE

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
```

print_results(***kwargs*)

CommandLine: python -m wbia –tf TestResult.print_results

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> from wbia.expt import harness
>>> ibs, testres = harness.testdata_expts('PZ_MTEST')
>>> result = testres.print_results()
>>> print(result)
```

print_unique_annot_config_stats(*ibs=None*)

Parameters **ibs** (IBEISController) – wbia controller object(default = None)

CommandLine: python -m wbia TestResult.print_unique_annot_config_stats

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> import wbia
>>> testres = wbia.testdata_expts('PZ_MTEST', a=['ctrl::unctrl_comp'])
>>> ibs = None
>>> result = testres.print_unique_annot_config_stats(ibs)
>>> print(result)
```

qaids

rank_mat

reconstruct_test_flags()

report()

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

unique_pcfgs

wbia.expt.test_result.**build_cmsinfo**(*cm_list*, *qreq_*)
Helper function to report results over multiple queries (chip matches). Basically given a group of queries of the same name, we only care if one of them is correct. This emulates encounters.

Runs queries of a specific configuration returns the best rank of each query.

Parameters

- **cm_list** (*list*) – list of chip matches
- **qreq** (QueryRequest) – request that computed the chip matches.

Returns cmsinfo - info about multiple chip matches cm_list

Return type dict

CommandLine: python -m wbia get_query_result_info python -m wbia get_query_result_info:0 –db lynx -a :qsame_imageset=True,been_adjusted=True,excluderef=True -t :K=1

python -m wbia get_query_result_info:0 –db lynx -a :qsame_imageset=True,been_adjusted=True,excluderef=True -t :K=1 –cmd

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.main_helpers.testdata_qreq_(a=[':qindex=0:3,dindex=0:5'])
>>> cm_list = qreq_.execute()
>>> cmsinfo = build_cmsinfo(cm_list, qreq_)
>>> print(ut.repr2(cmsinfo))
```

Ignore:

wbia -e rank_cmc –db humpbacks -a :has_any=hasnotch,mingt=2 -t :proot=BC_DTW –show –nocache-big

wbia -e rank_cmc –db humpbacks -a :is_known=True,mingt=2 -t :pipeline_root=BC_DTW

wbia -e rank_cmc –db humpbacks -a :is_known=True -t :pipeline_root=BC_DTW –qaid=1,9,15,16,18 –daid-override=1,9,15,16,18,21,22 –show –debug-depc

–clear-all-depcache

wbia.expt.test_result.**combine_testres_list**(*ibs*, *testres_list*)

combine test results over multiple annot configs

The combination of pipeline and annotation config is indexed by cfgx. A cfgx corresponds to a unique query request

CommandLine: python -m wbia –tf combine_testres_list

python -m wbia –tf -draw_rank_cmc –db PZ_MTEST –show python -m wbia –tf -draw_rank_cmc –db PZ_Master1 –show python -m wbia –tf -draw_rank_cmc –db PZ_MTEST –show -a varysize -t default python -m wbia –tf -draw_rank_cmc –db PZ_MTEST –show -a varysize -t default

```
>>> # DISABLE_DOCTEST
>>> from wbia.expt.test_result import *  # NOQA
>>> from wbia.expt import harness
>>> ibs, testres = harness.testdata_expts('PZ_MTEST', ['varysize'])
```

1.6.12 Module contents

wbia.expt.**IMPORT_TUPLES = [('experiment_configs', None), ('harness', None), ('experiment_he**

cd /home/joncrall/code/wbia/wbia.expt makeinit.py

Type Regen Command

wbia.expt.**reassign_submodule_attributes**(*verbose=True*)

why reloading all the modules doesnt do this I don't know

wbia.expt.**reload_subs**(*verbose=True*)

Reloads wbia.expt and submodules

wbia.expt.**rrrr**(*verbose=True*)

Reloads wbia.expt and submodules

1.7 wbia.gui package

1.7.1 Submodules

1.7.2 wbia.gui.clock_offset_gui module

1.7.3 wbia.gui.guiback module

1.7.4 wbia.gui.guiexcept module

exception wbia.gui.guiexcept.**InvalidRequest**(**args*)
Bases: Exception

exception wbia.gui.guiexcept.**NeedsUserInput**(**args*)
Bases: Exception

exception wbia.gui.guiexcept.**UserCancel**(**args*)
Bases: Exception

1.7.5 wbia.gui.guiexceptions module

1.7.6 wbia.gui.guiheaders module

This model provides the declarative interface to all of the api_*_models in guitool. Each different type of model/view has to register its iders, getters, and potentially setters (hopefully if guitool ever gets off the ground the delters as well)

Different columns can be hidden / shown by modifying this file

TODO: need to cache the total number of annotations or something about imagesets on disk to help startuptime.

wbia.gui.guiheaders.**make_table_declarations**(*ibs*)
these used to be global variables, hopefully we can make them a little more configurable

wbia.gui.guiheaders.**make_wbia_headers_dict**(*ibs*)

wbia.gui.guiheaders.**partial_imap_1to1**(*func*, *si_func*)

1.7.7 wbia.gui.guimenus module

1.7.8 wbia.gui.id_review_api module

1.7.9 wbia.gui.inspect_gui module

1.7.10 wbia.gui.models_and_views module

1.7.11 wbia.gui.newgui module

1.7.12 Module contents

1.8 wbia.guitool package

1.8.1 Subpackages

1.8.1.1 wbia.guitool.__PYQT__ package

1.8.1.1.1 Submodules

1.8.1.1.2 wbia.guitool.__PYQT__.QtCore module

1.8.1.1.3 wbia.guitool.__PYQT__.QtGui module

1.8.1.1.4 wbia.guitool.__PYQT__.QtTest module

1.8.1.1.5 wbia.guitool.__PYQT__.QtWidgets module

1.8.1.1.6 wbia.guitool.__PYQT__._internal module

Move to PyQt5? pip install git+git://github.com/pyqt/python-qt5.git

Ignore:

```
>>> import sys
>>> from PyQt5 import QtWidgets
>>> app = QtWidgets.QApplication(sys.argv)
>>> button = QtWidgets.QPushButton("Hello")
>>> button.setFixedSize(400, 400)
>>> button.show()
>>> app.exec_()
```

1.8.1.1.7 Module contents

wbia.guitool.__PYQT__.**QVariantHack**(**args*)
Hack when sip.setapi('QVariant') is 2

1.8.1.2 wbia.guitool.tests package

1.8.1.2.1 Submodules

1.8.1.2.2 wbia.guitool.tests.test_treenode module

1.8.1.2.3 Module contents

1.8.2 Submodules

1.8.3 wbia.guitool.PrefWidget2 module

1.8.4 wbia.guitool.PreferenceWidget module

1.8.5 wbia.guitool.api_button_delegate module

1.8.6 wbia.guitool.api_item_model module

1.8.7 wbia.guitool.api_item_view module

1.8.8 wbia.guitool.api_item_widget module

1.8.9 wbia.guitool.api_table_view module

1.8.10 wbia.guitool.api_thumb_delegate module

1.8.11 wbia.guitool.api_timestamp_delegate module

1.8.12 wbia.guitool.api_tree_node module

1.8.13 wbia.guitool.api_tree_view module

1.8.14 wbia.guitool.filter_proxy_model module

1.8.15 wbia.guitool.guitool_components module

1.8.16 wbia.guitool.guitool_decorators module

1.8.17 wbia.guitool.guitool_delegates module

1.8.18 wbia.guitool.guitool_dialogs module

1.8.19 wbia.guitool.guitool_main module

1.8.20 wbia.guitool.guitool_misc module

1.8.21 wbia.guitool.guitool_tables module

1.8.22 wbia.guitool.mpl_embed module

1.8.23 wbia.guitool.mpl_widget module

1.8.24 wbia.guitool.qt_enums module

Todo:

- cross validation
- encounter vs database (time filtering)

wbia.init.filter_annots.**annot_crossval**(*ibs*, *aid_list*, *n_qaids_per_name=1*, *n_daids_per_name=1*, *rng=None*, *debug=True*, *n_splits=None*, *confusors=True*)

Stratified sampling per name size

Parameters **n_splits** (*int*) – number of query/database splits to create. note, some names may not be big enough to split this many times.

CommandLine: python -m wbia.init.filter_annots annot_crossval

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.init.filter_annots import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> aid_list = ibs.get_valid_aids()
>>> n_qaids_per_name = 2
>>> n_daids_per_name = 3
>>> rng = 0
>>> debug = True
>>> n_splits = None
>>> expanded_aids_list = annot_crossval(
>>>     ibs, aid_list, n_qaids_per_name, n_daids_per_name, rng, debug,
>>>     n_splits, confusors=False)
>>> result = ('expanded_aids_list = %s' % (ut.repr2(expanded_aids_list, nl=2),))
>>> print(result)
```

wbia.init.filter_annots.**crossval_helper**(*nid_to_sample_pool*, *perquery*, *perdatab*, *n_need*, *n_splits=None*, *rng=None*, *rebalance=True*)

does sampling based on some grouping (or no grouping) of annots

perquery = 2 perdatab = 2
nid_to_sample_pool = { 1: [1, 2, 3, 4], 2: [6, 7, 8, 9],
}

wbia.init.filter_annots.**encounter_crossval**(*ibs*, *aids*, *qenc_per_name=1*, *denc_per_name=1*, *enc_labels=None*, *confusors=True*, *rng=None*, *annots_per_enc=None*, *rebalance=True*, *n_splits=None*, *early=False*)

Constructs a list of [(qaids, daids)] where there are *qenc_per_name* and *denc_per_name* for each individual in the datasets respectively. *enc_labels* specifies custom encounter labels.
CommandLine: python -m wbia.init.filter_annots encounter_crossval

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.init.filter_annots import *  # NOQA
```

(continues on next page)

(continued from previous page)

```
>>> from wbia.init import main_helpers
>>> import wbia
>>> #ibs, aids = wbia.testdata_aids(
>>> #    defaultdb='WWF_Lynx_Copy',
>>> #    a='default:minqual=good,require_timestamp=True,view=left')
>>> ibs, aids = wbia.testdata_aids(defaultdb='PZ_MTEST',
>>>                                a='default:require_timestamp=True')
>>> main_helpers.monkeypatch_encounters(ibs, aids, days=50)
>>> qenc_per_name = 2
>>> denc_per_name = 2
>>> confusors = False
>>> print('denc_per_name = %r' % (denc_per_name,))
>>> print('qenc_per_name = %r' % (qenc_per_name,))
>>> rng = 0
>>> n_splits = 5
>>> expanded_aids = encounter_crossval(ibs, aids, n_splits=n_splits,
>>>                                    qenc_per_name=qenc_per_name,
>>>                                    denc_per_name=denc_per_name,
>>>                                    confusors=confusors, rng=rng)
>>> # ensure stats agree
>>> cfgargs = dict(per_vp=False, per_multiple=False, combo_dists=False,
>>>                per_name=False, per_enc=True, use_hist=False)
>>> for qaids, daids in expanded_aids:
>>>     stats = ibs.get_annotconfig_stats(qaids, daids, **cfgargs)
>>>     del stats['confusor_daid_stats']
>>>     print(ut.repr2(stats, strvals=True, strkeys=True, nl=2))
>>>     denc_stats = stats['matchable_daid_stats']['denc_per_name']
>>>     qenc_stats = stats['qaid_stats']['qenc_per_name']
>>>     assert denc_stats['min'] == denc_stats['max']
>>>     assert denc_stats['min'] == denc_per_name
>>>     assert qenc_stats['min'] == qenc_stats['max']
>>>     assert qenc_stats['min'] == qenc_per_name
>>> # Restore state
>>> main_helpers.unmonkeypatch_encounters(ibs)
>>> #qaids, daids = expanded_aids[0]
>>> #stats = ibs.get_annotconfig_stats(qaids, daids, use_hist=True)
>>> #print(ut.repr2(stats, strvals=True, strkeys=True, nl=2))
```

wbia.init.filter_annots.**ensure_flatiterable**(*input_*)

wbia.init.filter_annots.**ensure_flatlistlike**(*input_*)

wbia.init.filter_annots.**expand_acfgs**(*ibs*, *aidcfg*, *verbose=None*, *use_cache=None*, *hack_exclude_keys=None*, *initial_aids=None*, *save_cache=True*)

Main multi-expansion function. Expands an annot config dict into qaids and daids. New version of this function based on a configuration dictionary built from command line argumetns

Parameters

- **ibs** (*IBEISController*) – wbia controller object
- **aidcfg** (*dict*) – configuration of the annotation filter
- **verbose** (*bool*) – verbosity flag(default = False)
- **use_cache** (*bool*) – turns on disk based caching(default = None)
- **hack_exclude_keys** (*None*) – (default = None)
- **initial_aids** (*None*) – (default = None)

Returns

expanded_aids=(qaid_list, daid_list) - expanded list of aids that meet the criteria of the aidcfg filter

Return type tuple

Todo: The database should be created first in most circumstances, then the queries should be filtered to meet the database restrictions? I'm not sure Sometimes you need to set the query aids constant, but sometimes you need to set the data aids constant. Seems to depend.

This function very much needs the idea of filter chains
OkNewIdea:

3 filters:

- Common sampling - takes care of things like min time delta,
- species, quality viewpoint etc.
- query sampling
- database sampling

Basic idea is

- Sample large pool
- Partition pool into query and database

Requires:

- base sampling params
- partition1 params
- partition2 params
- inter partition params?

CommandLine: python -m wbia.dev -e print_acfg -a timectrl:qsize=10,dsize=10 –db PZ_MTEST –veryverbtd –nocache-aid python -m wbia.dev -e print_acfg -a timectrl:qminqual=good,qsize=10,dsize=10 –db PZ_MTEST –veryverbtd –nocache-aid

python -m wbia.dev -e print_acfg -a timectrl –db PZ_MTEST –verbtd –nocache-aid python -m wbia.dev -e print_acfg -a timectrl –db PZ_Master1 –verbtd –nocache-aid python -m wbia.dev -e print_acfg -a timequalctrl –db PZ_Master1 –verbtd –nocache-aid

python -m wbia.dev -e rank_cmc -a controlled:qsize=10,dsize=10,dper_name=2 -t default –db PZ_MTEST python -m wbia.dev -e rank_cmc -a controlled:qsize=10,dsize=20,dper_name=2 -t default –db PZ_MTEST python -m wbia.dev -e print -a controlled:qsize=10,dsize=10 -t default –db PZ_MTEST –verbtd –nocache-aid

python -m wbia.dev -e latexsum -t candinvar -a viewpoint_compare –db NNP_Master3 –acfginfo utprof.py -m wbia.dev -e print -t candk -a varysize –db PZ_MTEST –acfginfo utprof.py -m wbia.dev -e latexsum -t candk -a controlled –db PZ_Master0 –acfginfo

python -m wbia –tf get_annotcfg_list:0 –db NNP_Master3 -a viewpoint_compare –nocache-aid –verbtd

python -m wbia –tf get_annotcfg_list –db PZ_Master1 -a timectrl:qhas_any=(needswork,correctable,mildviewpoint),ql –acfginfo –veryverbtd –veryverbtd

python -m wbia –tf draw_rank_cmc –db PZ_Master1 –show -t best -a timectrl:qhas_any=(needswork,correctable,mildviewpoint),qhas_none=(viewpoint,photobomb,error:viewpoint,quality) –acfginfo –veryverbtd

python -m wbia –tf get_annotcfg_list –db Oxford -a default:qhas_any=(query,),dpername=2,exclude_reference=True –acfginfo –verbtd –veryverbtd –nocache-aid

CommandLine: python -m wbia.init.filter_annots –exec-expand_acfgs –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.init.filter_annots import *  # NOQA
>>> import wbia
>>> from wbia.expt import annotation_configs
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aidcfg = copy.deepcopy(annotation_configs.default)
>>> aidcfg['qcfg']['species'] = 'primary'
>>> initial_aids = None
>>> expanded_aids = expand_acfgs(ibs, aidcfg, initial_aids=initial_aids)
>>> result = ut.repr3(expanded_aids, nl=1, nobr=True)
>>> print(result)
[1, 2, 3, 4, 5, 6],
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13],
```

wbia.init.filter_annots.**expand_acfgs_consistently**(*ibs*, *acfg_combo*, *initial_aids=None*, *use_cache=None*, *verbose=None*, *base=0*)

Expands a set of configurations such that they are comparable

CommandLine:

python -m wbia –tf parse_acfg_combo_list -a varysize

wbia –tf get_annotcfg_list –db PZ_Master1 -a varysize #wbia –tf get_annotcfg_list –db lynx -a default:hack_imageset=True wbia –tf get_annotcfg_list –db PZ_Master1 -a varysize:qsize=None wbia –tf get_annotcfg_list –db PZ_Master0 –nofilter-dups -a varysize wbia –tf get_annotcfg_list –db PZ_MTEST -a varysize –nofilter-dups wbia –tf get_annotcfg_list –db PZ_Master0 –verbtd

–nofilter-dups -a varysize

wbia –tf get_annotcfg_list –db PZ_Master1 -a viewpoint_compare –verbtd –nofilter-dups

wbia –tf get_annotcfg_list -a timectrl –db GZ_Master1 –verbtd –nofilter-dups

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.init.filter_annots import *  # NOQA
>>> from wbia.init import main_helpers
>>> from wbia.expt import annotation_configs
>>> from wbia.expt.experiment_helpers import parse_acfg_combo_list
>>> import wbia
>>> ibs = wbia.opendb('PZ_MTEST')
>>> #acfg_name_list = ['timectrl:dpername=[1,2]']
>>> acfg_name_list = ['default:crossval_enc=True,require_timestamp=True']
>>> aids = ibs.get_valid_aids()
>>> main_helpers.monkeypatch_encounters(ibs, aids, days=50)
```

(continues on next page)

(continued from previous page)

```
>>> acfg_combo_list = parse_acfg_combo_list(acfg_name_list)
>>> acfg_combo = acfg_combo_list[0]
>>> initial_aids = None
>>> use_cache = False
>>> verbose = False
>>> expanded_aids_combo_list = expand_acfgs_consistently(
>>>     ibs, acfg_combo, initial_aids=initial_aids, use_cache=use_cache,
>>>     verbose=verbose)
>>> # Restore state
>>> main_helpers.unmonkeypatch_encounters(ibs)
>>> ut.assert_eq(len(expanded_aids_combo_list), 5)
```

wbia.init.filter_annots.**expand_single_acfg**(*ibs*, *aidcfg*, *verbose=None*)
for main_helpers

wbia.init.filter_annots.**expand_species**(*ibs*, *species*, *avail_aids=None*)

wbia.init.filter_annots.**filter_annots_general**(*ibs*, *aid_list=None*, *filter_kw={}*, *verbose=False*, ***kwargs*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*list*) – list of annotation rowids
- **filter_kw** –

Kwargs: has_none_annotmatch, any_match_annotmatch, has_all, is_known, any_match_annot, logic_annot, none_match_annotmatch, max_num_annotmatch, any_startswith_annot, has_any, require_quality, species, any_match, view_ext, has_any_annotmatch, view_pername, max_num_annot, min_timedelta, any_startswith, max_numfeat, any_startswith_annotmatch, been_adjusted, any_endswith_annot, require_viewpoint, logic, has_any_annot, min_num_annotmatch, min_num, min_num_annot, has_all_annot, has_none, min_pername, any_endswith_annotmatch, any_endswith, require_timestamp, none_match, contributor_contains, has_all_annotmatch, logic_annotmatch, min_numfeat, none_match_annot, view_ext1, view_ext2, max_num, has_none_annot, minqual, view

CommandLine: python -m wbia –tf filter_annots_general python -m wbia –tf filter_annots_general –db PZ_Master1

–has_any=[needswork,correctable,mildviewpoint] –has_none=[viewpoint,photobomb,error:viewpoint,quality] –show

python -m wbia –tf filter_annots_general –db=GZ_Master1 –max-numfeat=300 –show –min-qual=junk –species=None

python -m wbia –tf filter_annots_general –db=lynx –been_adjusted=True

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.init.filter_annots import *  # NOQA
>>> import wbia
>>> filter_kw = ut.argparse_dict(get_default_annot_filter_form(),
>>>                              type_hint=ut.ddict(list, has_any=list,
>>>                                                 has_none=list,
>>>                                                 logic=str))
>>> print('filter_kw = %s' % (ut.repr2(filter_kw),))
>>> ibs = wbia.opendb(defaultdb='testdb1')
```

(continues on next page)

(continued from previous page)

```
>>> aid_list = ibs.get_valid_aids()
>>> #filter_kw = dict(is_known=True, min_num=1, has_any='viewpoint')
>>> #filter_kw = dict(is_known=True, min_num=1, any_match='.*error.*')
>>> aid_list_ = filter_annots_general(ibs, aid_list, filter_kw)
>>> print('len(aid_list_) = %r' % (len(aid_list_),))
>>> all_tags = ut.flatten(ibs.get_annot_all_tags(aid_list_))
>>> filtered_tag_hist = ut.dict_hist(all_tags)
>>> ut.print_dict(filtered_tag_hist, key_order_metric='val')
>>> ut.print_dict(ibs.get_annot_stats_dict(aid_list_), 'annot_stats')
>>> ut.quit_if_noshow()
>>> import wbia.viz.interact
>>> wbia.viz.interact.interact_chip.interact_multichips(ibs, aid_list_)
>>> ut.show_if_requested()
```

wbia.init.filter_annots.**filter_annots_independent**(*ibs*, *avail_aids*, *aidcfg*, *prefix=''*, *verbose=False*, *withpre=False*)

Filtering that doesn't have to do with a reference set of aids

TODO make filterflags version

Parameters

- **ibs** (IBEISController) – wbia controller object
- **avail_aids** (*list*) –
- **aidcfg** (*dict*) –
- **prefix** (*str*) – (default = '')
- **verbose** (*bool*) – verbosity flag(default = False)

Returns avail_aids

Return type list

CommandLine: python -m wbia –tf filter_annots_independent –veryverbtd

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.init.filter_annots import *  # NOQA
>>> import wbia
>>> from wbia.expt import annotation_configs
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> avail_aids = input_aids = ibs.get_valid_aids()
>>> aidcfg = annotation_configs.default['dcfg']
>>> aidcfg['require_timestamp'] = True
>>> aidcfg['require_quality'] = False
>>> aidcfg['is_known'] = True
>>> prefix = ''
>>> verbose = True
>>> avail_aids = filter_annots_independent(ibs, avail_aids, aidcfg,
>>>                                        prefix, verbose)
>>> result = ('avail_aids = %s' % (str(avail_aids),))
>>> print(result)
```

Ignore: # Testing tag features python -m wbia –tf draw_rank_cmc –db PZ_Master1 –show -t best

-a timectrl:qhas_any=(needswork,correctable,mildviewpoint),qhas_none=(viewpoint,photobomb,error:viewpoint,qua
—acfginfo –veryverbtd

wbia.init.filter_annots.**filter_annots_intragroup**(*ibs*, *avail_aids*, *aidcfg*, *prefix=''*, *verbose=False*, *withpre=False*)

This filters annots using information about the relationships between the annotations in the `avail_aids` group. This function is not independent and a second consecutive call may yield new results. Thus, the order in which this filter is applied matters.

CommandLine:

wbia –tf get_annotcfg_list -a default:qsame_imageset=True,been_adjusted=True,excluderef=True –db lynx –veryverbtd –nocache-aid

Ignore:

```
>>> aidcfg['min_timedelta'] = 60 * 60 * 24
>>> aidcfg['min_pername'] = 3
```

wbia.init.filter_annots.**filterannots_by_tags**(*ibs*, *aid_list*, *filter_kw*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*list*) – list of annotation rowids

CommandLine: python -m wbia –tf filterannots_by_tags utprof.py -m wbia –tf filterannots_by_tags
SeeAlso: filter_annotmatch_by_tags

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.init.filter_annots import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_Master1')
>>> aid_list = ibs.get_valid_aids()
>>> has_any = ut.get_argval('--tags', type_=list,
>>>                         default=['SceneryMatch', 'Photobomb'])
>>> min_num = ut.get_argval('--min_num', type_=int, default=1)
>>> filter_kw = dict(has_any=has_any, min_num=1)
>>> aid_list_ = filterannots_by_tags(ibs, aid_list, filter_kw)
>>> print('aid_list_ = %r' % (aid_list_,))
>>> ut.quit_if_noshow()
>>> pass
>>> # TODO: show special annot group in GUI
```

wbia.init.filter_annots.**get_acfg_cacheinfo**(*ibs*, *aidcfg*)

Returns location and name of the ~~annot~~ data cache

wbia.init.filter_annots.**get_annot_tag_filterflags**(*ibs*, *aid_list*, *filter_kw*, *request_defaultkw=False*)

Filters annotations by tags including those that is belongs to in a pair

wbia.init.filter_annots.**get_default_annot_filter_form**()

Returns dictionary containing defaults for all valid filter parameters

CommandLine: python -m wbia –tf get_default_annot_filter_form

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.init.filter_annots import *  # NOQA
>>> filter_kw = get_default_annot_filter_form()
>>> print(ut.repr2(filter_kw, align=True))
>>> print(', '.join(filter_kw.keys()))
```

wbia.init.filter_annots.**get_reference_preference_order**(*ibs*, *gt_ref_grouped_aids*, *gt_avl_grouped_aids*, *prop_getter*, *cmp_func*, *aggfn*, *rng*, *verbose=False*)

Orders preference for sampling based on some metric

wbia.init.filter_annots.**hack_extra**(*ibs*, *expanded_aids*)

wbia.init.filter_annots.**hack_remove_label_errors**(*ibs*, *expanded_aids*, *verbose=None*)

wbia.init.filter_annots.**multi_sampled_seaturtle_queries**()

wbia.init.filter_annots.**sample_annots**(*ibs*, *avail_aids*, *aidcfg*, *prefix=''*, *verbose=False*)

Sampling preserves input sample structure and thust does not always return exact values
CommandLine: python -m wbia –tf sample_annots –veryverbtd

python -m wbia –tf get_annotcfg_list –db seaturtles -a default:qhas_any=(left,right),sample_occur=True,exclude_refer
–acfginfo

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.init.filter_annots import *  # NOQA
>>> import wbia
>>> from wbia.expt import annotation_configs
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> avail_aids = input_aids = ibs.get_valid_aids()
>>> aidcfg = copy.deepcopy(annotation_configs.default['dcfg'])
>>> aidcfg['sample_per_name'] = 3
>>> aidcfg['sample_size'] = 10
>>> aidcfg['min_pername'] = 2
>>> prefix = ''
>>> verbose = True
>>> avail_aids = filter_annots_independent(ibs, avail_aids, aidcfg,
>>>                                        prefix, verbose)
>>> avail_aids = sample_annots(ibs, avail_aids, aidcfg,
>>>                            prefix, avail_aids)
>>> result = ('avail_aids = %s' % (str(avail_aids),))
>>> print(result)
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.init.filter_annots import *  # NOQA
>>> import wbia
>>> from wbia.expt import annotation_configs
>>> db = 'seaturtles'  # 'testdb1'
```

(continues on next page)

(continued from previous page)

```
>>> ibs = wbia.opendb(defaultdb=db)
>>> aidcfg = copy.deepcopy(annotation_configs.default)['qcfg']
>>> aidcfg['sample_occur'] = True
>>> initial_aids = ibs.get_valid_aids()
>>> withpre, verbose, prefix = True, 2, ''
>>> avail_aids = filter_annots_independent(
>>>     ibs, initial_aids, {'has_any': ['left', 'right']}, prefix, verbose)
>>> qaids = sample_annots(ibs, avail_aids, aidcfg, prefix, verbose)
>>> avail_aids = initial_aids
>>> ref_aids = qaids
>>> dcfg = dict(exclude_reference=True, sample_occur=True)
>>> daids = sample_annots_wrt_ref(ibs, initial_aids, dcfg, qaids, prefix, verbose)
>>> ibs.print_annotconfig_stats(qaids, daids, enc_per_name=True, per_enc=True)
```

wbia.init.filter_annots.**sample_annots_general**(*ibs*, *aid_list=None*, *filter_kw={}*, *verbose=False*, ***kwargs*)

filter + sampling

wbia.init.filter_annots.**sample_annots_wrt_ref**(*ibs*, *avail_aids*, *aidcfg*, *ref_aids*, *prefix=''*, *verbose=False*)

Sampling when a reference set is given

wbia.init.filter_annots.**subindex_annots**(*ibs*, *avail_aids*, *aidcfg*, *ref_aids=None*, *prefix=''*, *verbose=False*)

Returns exact subindex of annotations

wbia.init.filter_annots.**time_filter_annots**()

python -m wbia.init.filter_annots time_filter_annots –db PZ_Master1 -a ctrl:qmingt=2 –profile

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.init.filter_annots import *  # NOQA
>>> result = time_filter_annots()
```

wbia.init.filter_annots.**verb_context**(*filtertype*, *aidcfg*, *verbose*)

closure helper

1.9.3 wbia.init.main_commands module

TODO: Rename to wbia/init/commands.py

TODO; remove params module

wbia.init.main_commands.**postload_commands**(*ibs*, *back*)

Postload commands deal with a specific wbia database

wbia –db PZ_MTEST –occur "*All Images" –query 1 wbia –db PZ_MTEST –occur "*All Images" –query-intra

wbia.init.main_commands.**preload_commands**(*dbdir*, ***kwargs*)

Preload commands work with command line arguments and global caches

wbia.init.main_commands.**vdd**(*ibs*)

view data dir

wbia.init.main_commands.**vdq**(*dbdir*)

view directory and quit

wbia.init.main_commands.**vwd**()
view work dir

1.9.4 wbia.init.main_helpers module

This module defines helper functions to access common input needed to test many functions. These functions give a rich command line interface to specifically select subsets of annotations, pipeline configurations, and other filters.

TODO: standardize function signatures

wbia.init.main_helpers.**monkeypatch_encounters**(*ibs*, *aids*, *cache=None*, ***kwargs*)
Hacks in a temporary custom definition of encounters for this controller

50 days for PZ_MTEST kwargs = dict(days=50)
if False: name_mindeltas = [] for name in annots.group_items(annots.nids).values():

times = name.image_unixtimes_asfloat deltas = [ut.unixtime_to_timedelta(np.abs(t1 - t2))

for t1, t2 in ut.combinations(times, 2)]

if deltas: name_mindeltas.append(min(deltas))

logger.info(ut.repr3(ut.lmap(ut.get_timedelta_str, sorted(name_mindeltas))))

wbia.init.main_helpers.**testdata_aids**(*defaultdb=None*, *a=None*, *adefault='default'*, *ibs=None*, *return_acfg=False*, *verbose=None*, *default_aids=None*, *default_set='qcfg'*)
Grabs default testdata for functions, but is command line overrideable
CommandLine: python -m wbia testdata_aids –verbtd –db PZ_ViewPoints python -m wbia testdata_aids –verbtd –db NNP_Master3 -a is_known=True,view_pername='#primary>0&#primary1>=1' python -m wbia testdata_aids –verbtd –db PZ_Master1 -a default:is_known=True,view_pername='#primary>0&#primary1>=1' python -m wbia testdata_aids –verbtd –db PZ_Master1 -a default:species=primary,minqual=ok –verbtd python -m wbia.other.dbinfo –test-latex_dbstats –dblist python -m wbia testdata_aids –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.init.main_helpers import *  # NOQA
>>> from wbia.expt import annotation_configs
>>> import wbia
>>> #ibs = wbia.opendb(defaultdb='PZ_ViewPoints')
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> a = None
>>> adefault = 'default:is_known=True'
>>> aids, aidcfg = testdata_aids(ibs=ibs, a=a, adefault=adefault, return_
↪acfg=True)
>>> print('\n RESULT:')
>>> annotation_configs.print_acfg(aidcfg, aids, ibs, per_name_vpedge=None)
```

wbia.init.main_helpers.**testdata_cm**(*defaultdb=None*, *default_qaids=None*, *default_daids=None*, *t=None*, *p=None*, *a=None*)
CommandLine: python -m wbia.init.main_helpers –test-testdata_cm python -m wbia.init.main_helpers –test-testdata_cm –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.init.main_helpers import *  # NOQA
>>> cm, qreq_ = testdata_cm()
>>> cm.print_csv(ibs=qreq_.ibs)
>>> ut.quit_if_noshow()
>>> cm.show_single_annotmatch(qreq_, 2)
>>> ut.show_if_requested()
```

wbia.init.main_helpers.**testdata_cmlist**(*defaultdb=None*, *default_qaids=None*, *default_daids=None*, *t=None*, *p=None*, *a=None*, *verbose=None*)

Returns cm_list, qreq_

Return type list, wbia.QueryRequest

wbia.init.main_helpers.**testdata_expanded_aids**(*defaultdb=None*, *a=None*, *ibs=None*, *default_qaids=None*, *default_daids=None*, *qaid_override=None*, *daid_override=None*, *return_annot_info=False*, *verbose=None*, *use_cache=None*)

Parameters

- **default_qaids** (*list*) – (default = [1])
- **default_daids** (*str*) – (default = 'all')
- **defaultdb** (*str*) – (default = 'testdb1')
- **ibs** (IBEISController) – wbia controller object(default = None)
- **verbose** (*bool*) – verbosity flag(default = False)
- **return_annot_info** (*bool*) – (default = False)

Returns

Return type ibs, qaid_list, daid_list, annot_info

CommandLine: python -m wbia.init.main_helpers testdata_expanded_aids python -m wbia.init.main_helpers testdata_expanded_aids –db PZ_MTEST –acfg default:index=0:25 –verbose-testdata python -m wbia.init.main_helpers testdata_expanded_aids –db PZ_MTEST –qaid 3 python -m wbia.init.main_helpers testdata_expanded_aids –db GZ_ALL –acfg ctrl –verbose-testdata

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.init.main_helpers import *  # NOQA
>>> import wbia
>>> from wbia.expt import annotation_configs
>>> ibs, qaid_list, daid_list, aidcfg = testdata_expanded_aids(return_annot_
↪info=True)
>>> print('Printing annot config')
>>> annotation_configs.print_acfg(aidcfg)
>>> print('Printing annotconfig stats')
>>> ibs.print_annotconfig_stats(qaid_list, daid_list)
>>> print('Combined annotconfig stats')
```

(continues on next page)

(continued from previous page)

```
>>> ibs.print_annot_stats(qaid_list + daid_list, viewcode_isect=True)
>>> print('qaid_list = %r' % (qaid_list,))
```

wbia.init.main_helpers.**testdata_expts**(*defaultdb='testdb1', default_acfgstr_name_list=['default:qindex=0:10:4,dindex=0:20'], default_test_cfg_name_list=['default'], a=None, t=None, p=None, qaid_override=None, daid_override=None, initial_aids=None, use_cache=None, dbdir=None, ibs=None*)

Use this if you want data from an experiment. Command line interface to quickly get testdata for test_results.

Command line flags can be used to specify db, aidcfg, pipecfg, qaid override, daid override (and maybe initial aids).

CommandLine: python -m wbia.init.main_helpers testdata_expts

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.dbinfo import *  # NOQA
>>> import wbia
>>> ibs, testres = wbia.testdata_expts(defaultdb='pz_mtest',
>>>                                    a='timectrl:qsize=2',
>>>                                    t='invar:ai=[false],ri=false',
>>>                                    use_cache=false)
>>> print('testres = %r' % (testres,))
```

wbia.init.main_helpers.**testdata_filtcfg**(*default=None*)

wbia.init.main_helpers.**testdata_pipecfg**(*p=None*, *t=None*, *ibs=None*, *verbose=None*)

Returns pcfgdict

Return type dict

CommandLine: python -m wbia testdata_pipecfg python -m wbia testdata_pipecfg -t default:AI=False

Ignore: from jedi.evaluate import docstrings script = jedi.Script(ut.readfrom(main_helpers.__file__)) mod = script._get_module() func = mod.names_dict['testdata_pipecfg'][0].parent docstrings.find_return_types(script._evaluator, func)

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.init.main_helpers import *  # NOQA
>>> pcfgdict = testdata_pipecfg()
>>> result = ('pcfgdict = %s' % (ut.repr2(pcfgdict),))
>>> print(result)
```

wbia.init.main_helpers.**testdata_qreq_**(*p=None*, *a=None*, *t=None*, *default_qaids=None*, *default_daids=None*, *custom_nid_lookup=None*, *verbose=None*, ***kwargs*)

Parameters

- **p** (*None*) – (default = None)
- **a** (*None*) – (default = None)
- **t** (*None*) – (default = None)

- **default_qaids** (*None*) – (default = None)
- **default_daids** (*None*) – (default = None)

Kwargs: defaultdb, ibs, qaid_override, daid_override, return_annot_info, verbose, use_cache

Returns qreq_ - query request object with hyper-parameters

Return type wbia.QueryRequest

CommandLine: python -m wbia testdata_qreq_ –show –qaid 3

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.init.main_helpers import *  # NOQA
>>> kwargs = {}
>>> p = None
>>> a = None
>>> qreq_ = testdata_qreq_(p)
>>> result = ('qreq_ = %s' % (str(qreq_),))
```

wbia.init.main_helpers.**unmonkeypatch_encounters**(*ibs*)

1.9.5 wbia.init.sysres module

sysres.py == system_resources Module for dealing with system resoureces in the context of IBEIS but without the need for an actual IBEIS Controller

wbia.init.sysres.**copy_wbiadb**(*source_dbdir*, *dest_dbdir*)

wbia.init.sysres.**db_to_dbdir**(*db*, *allow_newdir=False*, *extra_workdirs=[]*)
Implicitly gets dbdir. Searches for db inside of workdir

wbia.init.sysres.**delete_dbdir**(*dbname*)

wbia.init.sysres.**ensure_db_from_url**(*zipped_db_url*)
SeeAlso wbia.init.sysres

wbia.init.sysres.**ensure_nauts**()
Ensures that you have the NAUT_test dataset

wbia.init.sysres.**ensure_pz_mtest**()
Ensures that you have the PZ_MTEST dataset
CommandLine: python -m wbia.init.sysres –exec-ensure_pz_mtest python -m wbia –tf ensure_pz_mtest
Ignore: from wbia.sysres import delete_dbdir delete_dbdir('PZ_MTEST')

Example

```
>>> # SCRIPT
>>> from wbia.init.sysres import *  # NOQA
>>> ensure_pz_mtest()
```

wbia.init.sysres.**ensure_pz_mtest_batchworkflow_test**()
CommandLine: python -m wbia.init.sysres –test-ensure_pz_mtest_batchworkflow_test python -m wbia.init.sysres –test-ensure_pz_mtest_batchworkflow_test –reset python -m wbia.init.sysres –test-ensure_pz_mtest_batchworkflow_test –reset

Example

```
>>> # SCRIPT
>>> from wbia.init.sysres import *  # NOQA
>>> ensure_pz_mtest_batchworkflow_test()
```

wbia.init.sysres.**ensure_pz_mtest_mergesplit_test**()
Make a test database for MERGE and SPLIT cases
CommandLine: python -m wbia.init.sysres –test-ensure_pz_mtest_mergesplit_test

Example

```
>>> # SCRIPT
>>> from wbia.init.sysres import *  # NOQA
>>> ensure_pz_mtest_mergesplit_test()
```

wbia.init.sysres.**ensure_testdb2**()

wbia.init.sysres.**ensure_testdb_assigner**

wbia.init.sysres.**ensure_testdb_curvrank**()

wbia.init.sysres.**ensure_testdb_identification_example**()

wbia.init.sysres.**ensure_testdb_kaggle7**()

wbia.init.sysres.**ensure_testdb_orientation**()

wbia.init.sysres.**ensure_wd_peter2**()

wbia.init.sysres.**ensure_wilddogs**()
Ensures that you have the NAUT_test dataset

wbia.init.sysres.**get_args_dbdir**(*defaultdb=None*, *allow_newdir=False*, *db=None*, *dbdir=None*)
Machinery for finding a database directory using the following priorities. The function first defaults to the specified function arguments. If those are not specified, then command line arguments are used. In all other circumstances the defaultdb is used. If defaultdb='cache' then the most recently used database directory is returned.

Parameters

- **defaultdb** (*None*) – database return if none other is specified
- **allow_newdir** (*bool*) – raises error if True and directory not found
- **db** (*None*) – specification using workdir priority
- **dbdir** (*None*) – specification using normal directory priority
- **cache_priority** (*bool*) – (default = False)

Returns dbdir

Return type str

CommandLine: python -m wbia.init.sysres get_args_dbdir

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.init.sysres import *  # NOQA
>>> dir1 = get_args_dbdir(None, False, 'testdb1', None)
>>> print('dir1 = %r' % (dir1,))
>>> dir2 = get_args_dbdir(None, False, dir1, None)
>>> print('dir2 = %r' % (dir2,))
>>> ut.assert_raises(ValueError, get_args_dbdir)
>>> print('dir3 = %r' % (dir2,))
```

wbia.init.sysres.**get_available_databases**(*workdir=None*)

Lists the available valid wbia databases inside of a work directory

Parameters **workdir** (*None*) –

Returns ibsdb_list - wbia controller object

Return type *IBEISController*

CommandLine: python -m wbia.init.sysres –test-get_ibsdb_list

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.init.sysres import *  # NOQA
>>> workdir = None
>>> ibsdb_list = get_ibsdb_list(workdir)
>>> result = str('\n'.join(ibsdb_list))
>>> print(result)
```

wbia.init.sysres.**get_dbalias_dict**()

wbia.init.sysres.**get_default_dbdir**()

wbia.init.sysres.**get_global_distinctiveness_modeldir**(*ensure=True*)

wbia.init.sysres.**get_ibsdb_list**(*workdir=None*)

Lists the available valid wbia databases inside of a work directory

Parameters **workdir** (*None*) –

Returns ibsdb_list - wbia controller object

Return type *IBEISController*

CommandLine: python -m wbia.init.sysres –test-get_ibsdb_list

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.init.sysres import *  # NOQA
>>> workdir = None
>>> ibsdb_list = get_ibsdb_list(workdir)
>>> result = str('\n'.join(ibsdb_list))
>>> print(result)
```

wbia.init.sysres.**get_logdir_global**()

wbia.init.sysres.**get_rawdir**()
Returns the standard raw data directory

wbia.init.sysres.**get_wbia_db_uri**(*db_dir: str = None*)
Central location to acquire the database URI value.

Parameters **db_dir** (*str*) – colloquial "dbdir" (default: None)

The db_dir argument is only to be used in testing. This function is monkeypatched by the testing environment (see wbia.conftest for that code). The monkeypatching is done because two or more instances of a controller (i.e. IBEISController) could be running in the same test. In that scenario more than one URI may need to be defined, which is not the case in production and why the body of this function is kept fairly simple. We ask the caller to supply the db_dir value in order to match up the corresponding URI.

wbia.init.sysres.**get_wbia_resource_dir**()

wbia.init.sysres.**get_workdir**(*allow_gui=True*)
Returns the work directory set for this computer. If allow_gui is true, a dialog will ask a user to specify the workdir if it does not exist.

python -c "import wbia; print(wbia.get_workdir())"

Parameters **allow_gui** (*bool*) – (default = True)

Returns work_dir

Return type str

CommandLine: python -m wbia.init.sysres get_workdir

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.init.sysres import *  # NOQA
>>> allow_gui = True
>>> work_dir = get_workdir(allow_gui)
>>> result = ('work_dir = %s' % (str(work_dir),))
>>> print(result)
```

wbia.init.sysres.**guiselect_workdir**()
Prompts the user to specify a work directory

wbia.init.sysres.**is_wbiadb**(*path*)
Checks to see if path contains the IBEIS internal dir

wbia.init.sysres.**list_dbs**(*workdir=None*)
Lists the available valid wbia databases inside of a work directory

Parameters **workdir** (*None*) –

Returns ibsdb_list - wbia controller object

Return type *IBEISController*

CommandLine: python -m wbia.init.sysres –test-get_ibsdb_list

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.init.sysres import *  # NOQA
>>> workdir = None
>>> ibsdb_list = get_ibsdb_list(workdir)
```

(continues on next page)

(continued from previous page)

```
>>> result = str('\n'.join(ibsdb_list))
>>> print(result)
```

wbia.init.sysres.**lookup_dbdir**(*db*, *allow_newdir=False*, *extra_workdirs=[]*)
Implicitly gets dbdir. Searches for db inside of workdir

wbia.init.sysres.**reset_mtest_graph**()
Resets the annotmatch and stating table
CommandLine: python -m wbia reset_mtest_graph

Example

```
>>> # SCRIPT
>>> from wbia.init.sysres import *  # NOQA
>>> reset_mtest_graph()
```

wbia.init.sysres.**set_default_dbdir**(*dbdir*)

wbia.init.sysres.**set_logdir**(*log_dir*)

wbia.init.sysres.**set_workdir**(*work_dir=None*, *allow_gui=False*)
Sets the workdirectory for this computer

Parameters

- **work_dir** (*None*) – (default = None)
- **allow_gui** (*bool*) – (default = True)

CommandLine: python -c "import wbia; wbia.sysres.set_workdir('/raid/work2')" python -c "import wbia; wbia.sysres.set_workdir('/raid/work')"

python -m wbia.init.sysres set_workdir

Example

```
>>> # SCRIPT
>>> from wbia.init.sysres import *  # NOQA
>>> print('current_work_dir = %s' % (str(get_workdir(False)),))
>>> work_dir = ut.get_argval('--workdir', type_=str, default=None)
>>> allow_gui = True
>>> result = set_workdir(work_dir, allow_gui)
```

1.9.6 Module contents

1.10 wbia.other package

1.10.1 Submodules

1.10.2 wbia.other.dbinfo module

get_dbinfo is probably the only usefull funciton in here # This is not the cleanest module

wbia.other.dbinfo.**cache_memory_stats**(*ibs*, *cid_list*, *fnum=None*)

wbia.other.dbinfo.**get_dbinfo**(*ibs*, *verbose=True*, *with_imgsize=True*, *with_bytes=True*, *with_contrib=True*, *with_agesex=True*, *with_header=True*, *with_reviews=True*, *with_ggr=False*, *with_ca=False*, *with_map=False*, *short=False*, *tag='dbinfo'*, *aid_list=None*, *aids=None*, *gmt_offset=3.0*)

Returns dictionary of digestable database information Infostr is a string summary of all the stats. Prints infostr in addition to returning locals

Parameters

- **ibs** (IBEISController) –
- **verbose** (*bool*) –
- **with_imgsize** (*bool*) –
- **with_bytes** (*bool*) –

Returns

Return type dict

SeeAlso: python -m wbia.other.ibsfuncs –exec-get_annot_stats_dict –db PZ_PB_RF_TRAIN –use-hist=True –old=False –per_name_vpedge=False python -m wbia.other.ibsfuncs –exec-get_annot_stats_dict –db PZ_PB_RF_TRAIN –all

CommandLine: python -m wbia.other.dbinfo –exec-get_dbinfo:0 python -m wbia.other.dbinfo –test-get_dbinfo:1 python -m wbia.other.dbinfo –test-get_dbinfo:0 –db NNP_Master3 python -m wbia.other.dbinfo –test-get_dbinfo:0 –db PZ_Master1 python -m wbia.other.dbinfo –test-get_dbinfo:0 –db GZ_ALL python -m wbia.other.dbinfo –exec-get_dbinfo:0 –db PZ_ViewPoints python -m wbia.other.dbinfo –exec-get_dbinfo:0 –db GZ_Master1

python -m wbia.other.dbinfo –exec-get_dbinfo:0 –db LF_Bajo_bonito -a default python -m wbia.other.dbinfo –exec-get_dbinfo:0 –db DETECT_SEATURTLES -a default –readonly

python -m wbia.other.dbinfo –exec-get_dbinfo:0 -a ctrl python -m wbia.other.dbinfo –exec-get_dbinfo:0 -a default:minqual=ok,require_timestamp=True –dbdir ~/lev/media/danger/LEWA python -m wbia.other.dbinfo –exec-get_dbinfo:0 -a default:minqual=ok,require_timestamp=True –dbdir ~/lev/media/danger/LEWA –loadbackup=0

python -m wbia.other.dbinfo –exec-get_dbinfo:0 -a default: –dbdir ~/lev/media/danger/LEWA python -m wbia.other.dbinfo –exec-get_dbinfo:0 -a default: –dbdir ~/lev/media/danger/LEWA –loadbackup=0

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.dbinfo import *  # NOQA
>>> import wbia
>>> defaultdb = 'testdb1'
>>> ibs, aid_list = wbia.testdata_aids(defaultdb, a='default:minqual=ok,
↪view=primary,view_ext1=1')
>>> kwargs = ut.get_kwdefaults(get_dbinfo)
>>> kwargs['verbose'] = False
>>> kwargs['aid_list'] = aid_list
>>> kwargs = ut.parse_dict_from_argv(kwargs)
>>> output = get_dbinfo(ibs, **kwargs)
>>> result = (output['info_str'])
>>> print(result)
>>> #ibs = wbia.opendb(defaultdb='testdb1')
>>> # <HACK FOR FILTERING>
>>> #from wbia.expt import cfghelpers
```

(continues on next page)

(continued from previous page)

```
>>> #from wbia.expt import annotation_configs
>>> #from wbia.init import filter_annots
>>> #named_defaults_dict = ut.dict_take(annotation_configs.__dict__,
>>> #                                   annotation_configs.TEST_NAMES)
>>> #named_qcfg_defaults = dict(zip(annotation_configs.TEST_NAMES,
>>> #                               ut.get_list_column(named_defaults_dict, 'qcfg
↪')))
>>> #acfg = cfghelpers.parse_argv_cfg(('--annot-filter', '-a'), named_defaults_
↪dict=named_qcfg_defaults, default=None)[0]
>>> #aid_list = ibs.get_valid_aids()
>>> # </HACK FOR FILTERING>
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.dbinfo import *  # NOQA
>>> import wbia
>>> verbose = True
>>> short = True
>>> #ibs = wbia.opendb(db='GZ_ALL')
>>> #ibs = wbia.opendb(db='PZ_Master0')
>>> ibs = wbia.opendb('testdb1')
>>> assert ibs.get_dbname() == 'testdb1', 'DO NOT DELETE CONTRIBUTORS OF OTHER DBS
↪'
>>> ibs.delete_contributors(ibs.get_valid_contributor_rowids())
>>> ibs.delete_empty_nids()
>>> #ibs = wbia.opendb(db='PZ_MTEST')
>>> output = get_dbinfo(ibs, with_contrib=False, verbose=False, short=True)
>>> result = (output['info_str'])
>>> print(result)
+============================
DB Info:  testdb1
DB Notes: None
DB NumContrib: 0
----------
# Names                      = 7
# Names (unassociated)       = 0
# Names (singleton)          = 5
# Names (multiton)           = 2
----------
# Annots                     = 13
# Annots (unknown)           = 4
# Annots (singleton)         = 5
# Annots (multiton)          = 4
----------
# Img                        = 13
L============================
```

wbia.other.dbinfo.**get_short_infostr**(*ibs*)

Returns printable database information

Parameters **ibs** (IBEISController) – wbia controller object

Returns infostr

Return type str

CommandLine: python -m wbia.other.dbinfo –test-get_short_infostr

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.dbinfo import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> infostr = get_short_infostr(ibs)
>>> result = str(infostr)
>>> print(result)
dbname = 'testdb1'
num_images = 13
num_annotations = 13
num_names = 7
```

wbia.other.dbinfo.**hackshow_names**(*ibs*, *aid_list*, *fnum=None*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*list*) –

CommandLine: python -m wbia.other.dbinfo –exec-hackshow_names –show python -m wbia.other.dbinfo –exec-hackshow_names –show –db PZ_Master1

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.dbinfo import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> aid_list = ibs.get_valid_aids()
>>> result = hackshow_names(ibs, aid_list)
>>> print(result)
>>> ut.show_if_requested()
```

wbia.other.dbinfo.**latex_dbstats**(*ibs_list*, ***kwargs*)

Parameters **ibs** (IBEISController) – wbia controller object

CommandLine: python -m wbia.other.dbinfo –exec-latex_dbstats –dblist testdb1 python -m wbia.other.dbinfo –exec-latex_dbstats –dblist testdb1 –show python -m wbia.other.dbinfo –exec-latex_dbstats –dblist PZ_Master0 testdb1 –show python -m wbia.other.dbinfo –exec-latex_dbstats –dblist PZ_Master0 PZ_MTEST GZ_ALL –show python -m wbia.other.dbinfo –test-latex_dbstats –dblist GZ_ALL NNP_MasterGIRM_core –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.dbinfo import *  # NOQA
>>> import wbia
>>> db_list = ut.get_argval('--dblist', type_=list, default=['testdb1'])
>>> ibs_list = [wbia.opendb(db=db) for db in db_list]
>>> tabular_str = latex_dbstats(ibs_list)
```

(continues on next page)

(continued from previous page)

```
>>> tabular_cmd = ut.latex_newcommand(ut.latex_sanitize_command_name('DatabaseInfo
↪'), tabular_str)
>>> ut.copy_text_to_clipboard(tabular_cmd)
>>> write_fpath = ut.get_argval('--write', type_=str, default=None)
>>> if write_fpath is not None:
>>>     fpath = ut.truepath(write_fpath)
>>>     text = ut.readfrom(fpath)
>>>     new_text = ut.replace_between_tags(text, tabular_cmd, '% <DBINFO>', '% </
↪DBINFO>')
>>>     ut.writeto(fpath, new_text)
>>> ut.print_code(tabular_cmd, 'latex')
>>> ut.quit_if_noshow()
>>> ut.render_latex_text('\\noindent \n' + tabular_str)
```

wbia.other.dbinfo.**print_qd_info**(*ibs*, *qaid_list*, *daid_list*, *verbose=False*)

SeeAlso: ibs.print_annotconfig_stats(qaid_list, daid_list)

information for a query/database aid configuration

wbia.other.dbinfo.**show_image_time_distributions**(*ibs*, *gid_list*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) –

CommandLine: python -m wbia.other.dbinfo show_image_time_distributions –show python -m wbia.other.dbinfo show_image_time_distributions –show –db lynx

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.dbinfo import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aids = wbia.testdata_aids(ibs=ibs)
>>> gid_list = ut.unique_unordered(ibs.get_annot_gids(aids))
>>> result = show_image_time_distributions(ibs, gid_list)
>>> print(result)
>>> ut.show_if_requested()
```

wbia.other.dbinfo.**show_time_distributions**(*ibs*, *unixtime_list*)

wbia.other.dbinfo.**sight_resight_count**(*nvisit1*, *nvisit2*, *resight*)

Lincoln Petersen Index

The Lincoln-Peterson index is a method used to estimate the total number of individuals in a population given two independent sets observations. The likelihood of a population size is a hypergeometric distribution given by assuming a uniform sampling distribution.

Parameters

- **nvisit1** (*int*) – the number of individuals seen on visit 1.
- **nvisit2** (*int*) – be the number of individuals seen on visit 2.
- **resight** (*int*) – the number of (matched) individuals seen on both visits.

Returns (pl_index, pl_error)

Return type tuple

LaTeX:

begin{equation}label{eqn:lpifull} L(poptotal given nvisit_1, nvisit_2, resight) = frac{

binom{nvisit_1}{resight} binom{poptotal - nvisit_1}{nvisit_2 - resight}

}{ binom{poptotal}{nvisit_2}

}

end{equation} Assuming that T has a uniform prior distribution, the maximum

likelihood estimation of population size given two visits to a location is:

begin{equation}label{eqn:lpi} poptotal approx frac{nvisit_1 nvisit_2}{resight} pm 1.96 sqrt{frac{{(nvisit_1)}^2 (nvisit_2) (nvisit_2 - resight)}{resight^3}}

end{equation}

References

https://en.wikipedia.org/wiki/Mark_and_recapture https://en.wikipedia.org/wiki/Talk:Mark_and_recapture#Statistical_treatment https://mail.google.com/mail/u/0/#search/lincoln+peterse+n/14c6b50227f5209f https://probabilityandstats.wordpress.com/tag/maximum-likelihood-estimate/ http://math.arizona.edu/~jwatkins/o-mle.pdf

CommandLine: python -m wbia.other.dbinfo sight_resight_count –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.dbinfo import *  # NOQA
>>> nvisit1 = 100
>>> nvisit2 = 20
>>> resight = 10
>>> (pl_index, pl_error) = sight_resight_count(nvisit1, nvisit2, resight)
>>> result = '(pl_index, pl_error) = %s' % ut.repr2((pl_index, pl_error))
>>> pl_low = max(pl_index - pl_error, 1)
>>> pl_high = pl_index + pl_error
>>> print('pl_low = %r' % (pl_low,))
>>> print('pl_high = %r' % (pl_high,))
>>> print(result)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> import scipy, scipy.stats
>>> x = pl_index  # np.array([10, 11, 12])
>>> k, N, K, n = resight, x, nvisit1, nvisit2
>>> #k, M, n, N = k, N, k, n  # Wiki to SciPy notation
>>> #prob = scipy.stats.hypergeom.cdf(k, N, K, n)
>>> fig = pt.figure(1)
>>> fig.clf()
>>> N_range = np.arange(1, pl_high * 2)
>>> # Something seems to be off
>>> probs = sight_resight_prob(N_range, nvisit1, nvisit2, resight)
>>> pl_prob = sight_resight_prob([pl_index], nvisit1, nvisit2, resight)[0]
>>> pt.plot(N_range, probs, 'b-', label='probability of population size')
>>> pt.plt.title('nvisit1=%r, nvisit2=%r, resight=%r' % (
```

(continues on next page)

(continued from previous page)

```
>>>     nvisit1, nvisit2, resight))
>>> pt.plot(pl_index, pl_prob, 'rx', label='Lincoln Peterson Estimate')
>>> pt.plot([pl_low, pl_high], [pl_prob, pl_prob], 'gx-',
>>>         label='Lincoln Peterson Error Bar')
>>> pt.legend()
>>> ut.show_if_requested()
```

1.10.3 wbia.other.detectcore module

Developer convenience functions for ibs (detections).

TODO: need to split up into sub modules: consistency_checks feasibility_fixes move the export stuff to dbio

then there are also convineience functions that need to be ordered at least within this file

wbia.other.detectcore.**classifier_visualize_training_localizations**(*ibs*, *classifier_weight_filepath*, *species_list=['zebra']*, *scheme=2*, *output_path=None*, *values=None*, ***kwargs*)

wbia.other.detectcore.**export_to_coco**(*ibs*, *species_list*, *species_mapping={}*, *viewpoint_mapping={}*, *target_size=2400*, *use_maximum_linear_dimension=True*, *use_existing_train_test=True*, *include_parts=False*, *gid_list=None*, *include_reviews=False*, *require_image_reviewed=False*, *require_named=False*, *output_images=True*, *use_global_train_set=False*, ***kwargs*)

Create training COCO dataset for training models.

wbia.other.detectcore.**export_to_pascal**(*ibs*, **args*, ***kwargs*)

Alias for export_to_xml

wbia.other.detectcore.**export_to_xml**(*ibs*, *species_list*, *species_mapping=None*, *offset='auto'*, *enforce_viewpoint=False*, *target_size=900*, *purge=False*, *use_maximum_linear_dimension=True*, *use_existing_train_test=True*, *include_parts=False*, *gid_list=None*, *output_path=None*, *allow_empty_images=False*, *min_annot_size=5*, ***kwargs*)

Create training XML for training models.

wbia.other.detectcore.**imageset_train_test_split**(*ibs*, *train_split=0.8*, *is_tile=False*, *gid_list=None*, ***kwargs*)

wbia.other.detectcore.**localizer_distributions**(*ibs*, *threshold=10*, *dataset=None*)

wbia.other.detectcore.**nms**(*dets*, *scores*, *thresh*, *use_cpu=True*)

wbia.other.detectcore.**redownload_detection_models**(*ibs*)

Re-download detection models.

Parameters **ibs** (IBEISController) –

CommandLine: python -c "from wbia.algo.detect import grabmodels; grabmodels.redownload_models()" python -c "import utool, wbia.algo; utool.view_directory(wbia.algo.detect.grabmodels._expand_modeldir())"

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.detectcore import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> result = redownload_detection_models(ibs)
>>> print(result)
```

wbia.other.detectcore.**view_model_dir**(*ibs*)

wbia.other.detectcore.**visualize_bounding_boxes**(*ibs*, *config*, *version*, *gid_list=None*, *randomize=False*, *num_images=10*, *t_width=500*, *output_path=None*)

wbia.other.detectcore.**visualize_distributions**(*distro_dict*, *threshold=10*)

wbia.other.detectcore.**visualize_ground_truth**(*ibs*, *config*, ***kwargs*)

wbia.other.detectcore.**visualize_pascal_voc_dataset**(*ibs*, *dataset_path*, *num_examples=30*, *randomize=False*, *write=True*, *write_path=None*)

Visualize the PASCAL VOC dataset.

Parameters

- **ibs** (IBEISController) –
- **dataset_path** (*str*) – the dataset path in the PASCAL VOC format
- **num_examples** (*int, optional*) – the number of examples to draw
- **randomize** (*bool, optional*) – if to randomize the visualization
- **write** (*bool, optional*) – if to display or write the files

CommandLine: python -m wbia.other.detectcore –test-visualize_pascal_voc_dataset

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.detectcore import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> dataset_path = '/Users/jason.parham/Downloads/wilddog_data/data/VOCdevkit/
↪VOC2020/'
>>> # dataset_path = '/Users/jason.parham/Downloads/LearningData/'
>>> # dataset_path = '/Users/jason.parham/Downloads/VOCdevkit/VOC2018/'
>>> ibs.visualize_pascal_voc_dataset(dataset_path, randomize=True)
```

wbia.other.detectcore.**visualize_predictions**(*ibs*, *config*, ***kwargs*)

1.10.4 wbia.other.detectexport module

wbia.other.detectexport.**get_cnn_classifier_cameratrap_binary_training_images_pytorch**(*ibs*, *pos-i-tive_ima neg-a-tive_ima dest_pa valid_ra im-age_size purge=T skip_rat skip_rat skip_rat*

wbia.other.detectexport.**get_cnn_classifier_canonical_training_images_pytorch**(*ibs*, *species*, *dest_path=None*, *valid_rate=0.2*, *im-age_size=224*, *purge=True*, *skip_rate=0.0*, *skip_rate_pos=0.0*, *skip_rate_neg=0.0*)

wbia.other.detectexport.**get_cnn_classifier_multiclass_training_images_pytorch**(*ibs*, *gid_list*, *la-bel_list*, *dest_path=None*, *valid_rate=0.2*, *im-age_size=224*, *purge=True*, *skip_rate=0.0*)

wbia.other.detectexport.**get_cnn_labeler_training_images_pytorch**(*ibs*, *dest_path=None*, *image_size=224*, *category_list=None*, *min_examples=10*, *category_mapping=None*, *viewpoint_mapping=None*, *flip_mapping=None*, *purge=True*, *strict=True*, *skip_rate=0.0*, *valid_rate=0.2*, *use_axis_aligned_chips=False*, *train_gid_set=None*)

wbia.other.detectexport.**get_cnn_localizer_canonical_training_images_pytorch**(*ibs*, *species*, *dest_path=None*, *valid_rate=0.2*, *image_size=224*, *purge=True*, *skip_rate=0.0*)

1.10.5 wbia.other.detectfuncs module

Developer convenience functions for ibs (detections).

TODO: need to split up into sub modules: consistency_checks feasibility_fixes move the export stuff to dbio

then there are also convineience functions that need to be ordered at least within this file

wbia.other.detectfuncs.**aoi2_confusion_matrix_algo_plot**(*ibs*, *label*, *color*, *conf*, *output_cases=False*, *category_list=None*, *test_gid_set_=None*, ***kwargs*)

wbia.other.detectfuncs.**aoi2_precision_recall_algo**(*ibs*, *category_list=None*, *test_gid_set_=None*, ***kwargs*)

wbia.other.detectfuncs.**aoi2_precision_recall_algo_display**(*ibs*, *test_gid_list=None*, *output_cases=False*, *figsize=(20, 20)*)

wbia.other.detectfuncs.**aoi2_precision_recall_algo_plot**(*ibs*, ***kwargs*)

wbia.other.detectfuncs.**aoi2_roc_algo_plot**(*ibs*, ***kwargs*)

wbia.other.detectfuncs.**background_accuracy_display**(*ibs*, *category_list*, *test_gid_set=None*, *output_path=None*)

wbia.other.detectfuncs.**canonical_confusion_matrix_algo_plot**(*ibs*, *label*, *color*, *conf*, *species*, *output_cases=False*, ***kwargs*)

wbia.other.detectfuncs.**canonical_localization_deviation_plot**(*ibs*, *attribute*, *color*, *index*, *label=None*, *species=None*, *marker='o'*, ***kwargs*)

wbia.other.detectfuncs.**canonical_localization_iou_plot**(*ibs*, *color*, *index*, *label=None*, *species=None*, *marker='o'*, *threshold=0.75*, ***kwargs*)

wbia.other.detectfuncs.**canonical_localization_iou_visualize**(*ibs*, *index*, *test_aid_set*, *test_bbox_set*, *prediction_list*, *overlap_list*, *color_list*, *label=None*, *species=None*, ***kwargs*)

wbia.other.detectfuncs.**canonical_localization_precision_recall_algo_display**(*ibs*, *figsize=(20, 40)*)

wbia.other.detectfuncs.**canonical_precision_recall_algo**(*ibs*, *species*, ***kwargs*)

wbia.other.detectfuncs.**canonical_precision_recall_algo_display**(*ibs*, *figsize=(20, 20)*)

wbia.other.detectfuncs.**canonical_precision_recall_algo_plot**(*ibs*, ***kwargs*)

wbia.other.detectfuncs.**canonical_roc_algo_plot**(*ibs*, ***kwargs*)

wbia.other.detectfuncs.**classifier2_precision_recall_algo**(*ibs*, *category*, *species_mapping={}*, *output_path=None*, *test_gid_list=None*, *test_label_list=None*, ***kwargs*)

wbia.other.detectfuncs.**classifier2_precision_recall_algo_display**(*ibs*, *species_list=None*, *species_mapping={}*, *nice_mapping={}*, *test_gid_list=None*, *test_label_list=None*, *figsize=(20, 9)*, ***kwargs*)

wbia.other.detectfuncs.**classifier2_precision_recall_algo_plot**(*ibs*, ***kwargs*)

wbia.other.detectfuncs.**classifier2_roc_algo_plot**(*ibs*, ***kwargs*)

wbia.other.detectfuncs.**classifier_cameratrap_confusion_matrix_algo_plot**(*ibs*, *label*, *color*, *conf*, *positive_imageset_id*, *negative_imageset_id*, *output_cases=False*, ***kwargs*)

wbia.other.detectfuncs.**classifier_cameratrap_precision_recall_algo**(*ibs*, *positive_imageset_id*, *negative_imageset_id*, ***kwargs*)

wbia.other.detectfuncs.**classifier_cameratrap_precision_recall_algo_display**(*ibs*, *positive_imageset_id*, *negative_imageset_id*, *config_list=None*, *figsize=(20, 20)*)

wbia.other.detectfuncs.**classifier_cameratrap_precision_recall_algo_plot**(*ibs*, ***kwargs*)

wbia.other.detectfuncs.**classifier_cameratrap_roc_algo_plot**(*ibs*, ***kwargs*)

wbia.other.detectfuncs.**detector_parse_gt**(*ibs*, *test_gid_list=None*, ***kwargs*)

wbia.other.detectfuncs.**general_area_best_conf**(*conf_list*, *x_list*, *y_list*, *label='Unknown'*, *color='b'*, *marker='o'*, *plot_point=True*, *interpolate=True*, *target=(1.0, 1.0)*, *target_recall=None*, ***kwargs*)

wbia.other.detectfuncs.**general_confusion_matrix_algo**(*label_correct_list*, *label_predict_list*, *category_list*, *category_mapping*, *fig_*, *axes_*, *fuzzy_dict=None*, *conf=None*, *conf_list=None*, *size=10*, ***kwargs*)

wbia.other.detectfuncs.**general_get_imageset_gids**(*ibs*, *imageset_text*, *unique=True*, ***kwargs*)

wbia.other.detectfuncs.**general_identify_operating_point**(*conf_list*, *x_list*, *y_list*, *target=(1.0, 1.0)*)

wbia.other.detectfuncs.**general_interpolate_precision_recall**(*conf_list*, *re_list*, *pr_list*)

wbia.other.detectfuncs.**general_intersection_over_union**(*bbox1*, *bbox2*)

wbia.other.detectfuncs.**general_overlap**(*gt_list*, *pred_list*)

wbia.other.detectfuncs.**general_parse_gt**(*ibs*, *test_gid_list=None*, ***kwargs*)

wbia.other.detectfuncs.**general_parse_gt_annots**(*ibs*, *aid_list*, *include_parts=True*, *species_mapping={}*, *gt_species_mapping={}*, ***kwargs*)

wbia.other.detectfuncs.**general_precision_recall_algo**(*ibs*, *label_list*, *confidence_list*, *category='positive'*, *samples=1000*, ***kwargs*)

wbia.other.detectfuncs.**general_tp_fp_fn**(*gt_list*, *pred_list*, *min_overlap*, ***kwargs*)

wbia.other.detectfuncs.**get_species_nice_mapping**(*ibs*, *species*)

wbia.other.detectfuncs.**labeler_confusion_matrix_algo_plot**(*ibs*, *category_list*, *species_mapping={}*, *viewpoint_mapping={}*, *category_mapping=None*, *test_gid_set=None*, ***kwargs*)

wbia.other.detectfuncs.**labeler_precision_recall_algo**(*ibs*, *category_list*, *label_dict*, ***kwargs*)

wbia.other.detectfuncs.**labeler_precision_recall_algo_display**(*ibs*, *category_list=None*, *species_mapping={}*, *viewpoint_mapping={}*, *category_mapping=None*, *fuzzy_dict=None*, *figsize=(30, 9)*, *test_gid_set=None*, *use_axis_aligned_chips=False*, *labeler_weight_filepath=None*, *config_list=None*, ***kwargs*)

wbia.other.detectfuncs.**labeler_precision_recall_algo_plot**(*ibs*, ***kwargs*)

wbia.other.detectfuncs.**labeler_roc_algo_plot**(*ibs*, ***kwargs*)

wbia.other.detectfuncs.**labeler_tp_tn_fp_fn**(*ibs*, *category_list*, *species_mapping={}*, *viewpoint_mapping={}*, *samples=1000*, *test_gid_set=None*, ***kwargs*)

wbia.other.detectfuncs.**localizer_assign**(*gt_list*, *pred*, *min_overlap*)

wbia.other.detectfuncs.**localizer_assignments**(*pred_list*, *gt_list*, *gt_list_=[]*, *min_overlap=0.5*)

wbia.other.detectfuncs.**localizer_confusion_matrix_algo_plot**(*ibs*, *label=None*, *target_conf=None*, *test_gid_list=None*, ***kwargs*)

wbia.other.detectfuncs.**localizer_iou_recall_algo**(*ibs*, *samples=100*, *test_gid_list=None*, *ignore_filter_func=None*, ***kwargs*)

wbia.other.detectfuncs.**localizer_iou_recall_algo_plot**(*ibs*, ***kwargs*)

wbia.other.detectfuncs.**localizer_parse_pred**(*ibs*, *test_gid_list=None*, *species_mapping={}*, *pred_species_mapping={}*, ***kwargs*)

wbia.other.detectfuncs.**localizer_parse_pred_dirty**(*ibs*, *test_gid_list*, *species_mapping_*, ***kwargs*)

wbia.other.detectfuncs.**localizer_precision_recall**(*ibs*, *config_dict=None*, *output_path=None*, *test_gid_list=None*, ***kwargs*)

wbia.other.detectfuncs.**localizer_precision_recall_algo**(*ibs*, *samples=1000*, *test_gid_list=None*, ***kwargs*)

wbia.other.detectfuncs.**localizer_precision_recall_algo_display**(*ibs*, *config_list*, *config_tag=''*, *min_overlap=0.5*, *figsize=(40, 9)*, *target_recall=0.8*, *BEST_INDEX=None*, *offset_color=0*, *write_images=False*, *plot_point=True*, *output_path=None*, *plot_iou_recall=True*, ***kwargs*)

wbia.other.detectfuncs.**localizer_precision_recall_algo_display_animate**(*ibs*, *config_list*, ***kwargs*)

wbia.other.detectfuncs.**localizer_precision_recall_algo_plot**(*ibs*, ***kwargs*)

wbia.other.detectfuncs.**localizer_tp_fp**(*uuid_list*, *gt_dict*, *pred_dict*, *min_overlap=0.5*, ***kwargs*)

wbia.other.detectfuncs.**simple_code**(*label*)

1.10.6 wbia.other.detectgrave module

Developer convenience functions for ibs (detections).

TODO: need to split up into sub modules: consistency_checks feasibility_fixes move the export stuff to dbio

then there are also convineience functions that need to be ordered at least within this file

wbia.other.detectgrave.**bootstrap**(*ibs, species_list=['zebra'], N=10, rounds=20, scheme=2, ensemble=9, output_path=None, precompute=True, precompute_test=True, recompute=False, visualize=True, C=1.0, kernel='rbf', **kwargs*)

wbia.other.detectgrave.**bootstrap2**(*ibs, species_list=['zebra'], alpha=10, gamma=16, epsilon=0.3, rounds=20, ensemble=3, dims=64, pca_limit=1000000, nms_thresh_pos=0.5, nms_thresh_neg=0.9, C=1.0, kernel='rbf', theta=1.0, output_path=None, precompute=True, precompute_test=True, recompute=False, recompute_classifications=True, overlap_thresh_cat_1=0.75, overlap_thresh_cat_2=0.25, overlap_thresh_cat_3=0.0, **kwargs*)

wbia.other.detectgrave.**bootstrap_pca_test**(*ibs, dims=64, pca_limit=500000, ann_batch=50, model_path=None, output_path=None, neighbors=1000, nms_thresh=0.5, min_confidence=0.3, **kwargs*)

wbia.other.detectgrave.**bootstrap_pca_train**(*ibs, dims=64, pca_limit=500000, ann_batch=50, output_path=None, **kwargs*)

wbia.other.detectgrave.**classifier2_train_image_rf**(*ibs, species_list, output_path=None, dryrun=False, n_estimators=100*)

wbia.other.detectgrave.**classifier2_train_image_rf_sweep**(*ibs, species_list, precompute=True, **kwargs*)

wbia.other.detectgrave.**classifier_train_image_svm**(*ibs, species_list, output_path=None, dryrun=False, C=1.0, kernel='rbf'*)

wbia.other.detectgrave.**classifier_train_image_svm_sweep**(*ibs, species_list, precompute=True, **kwargs*)

wbia.other.detectgrave.**get_classifier2_rf_data_labels**(*ibs, dataset_tag, category_list*)

wbia.other.detectgrave.**get_classifier_svm_data_labels**(*ibs, dataset_tag, species_list*)

wbia.other.detectgrave.**remove_rfdetect**(*ibs*)

wbia.other.detectgrave.**set_reviewed_from_target_species_count**(*ibs, species_set=None, target=1000*)

1.10.7 wbia.other.detecttrain module

Developer convenience functions for ibs (detections).

TODO: need to split up into sub modules: consistency_checks feasibility_fixes move the export stuff to dbio

then there are also convineience functions that need to be ordered at least within this file

wbia.other.detecttrain.**aoi2_train**(*ibs, species_list=None, train_gid_list=None, purge=True, cache=False*)

wbia.other.detecttrain.**aoi_train**(*ibs, species_list=None*)

wbia.other.detecttrain.**background_train**(*ibs, species, train_gid_set=None, global_limit=500000, **kwargs*)

Example: >>> values = output_path, X_file, y_file >>> print(values) >>> output_path, X_file, y_file = values >>> from wbia_cnn.models.background import train_background >>> values = (>>> '/data/ibeis/IMS_Master/_ibsdb/_ibeis_cache/training/background', >>> '/data/ibeis/IMS_Master/_ibsdb/_ibeis_cache/extracted/background/raw/X.npy', >>>

'/data/ibeis/IMS_Master/_ibsdb/_ibeis_cache/extracted/background/labels/y.npy' >>>) >>> output_path, X_file, y_file = values

wbia.other.detecttrain.**canonical_classifier_train**(*ibs*, *species*, *ensembles=3*, *extracted_path=None*, ***kwargs*)

wbia.other.detecttrain.**canonical_localizer_train**(*ibs*, *species*, *ensembles=3*, ***kwargs*)

wbia.other.detecttrain.**classifier2_train**(*ibs*, *species_list=None*, *species_mapping={}*, *train_gid_set=None*, ***kwargs*)

wbia.other.detecttrain.**classifier_binary_train**(*ibs*, *species_list*, ***kwargs*)

wbia.other.detecttrain.**classifier_cameratrap_densenet_train**(*ibs*, *positive_imageset_id*, *negative_imageset_id*, *ensembles=3*, ***kwargs*)

wbia.other.detecttrain.**classifier_cameratrap_train**(*ibs*, *positive_imageset_id*, *negative_imageset_id*, ***kwargs*)

wbia.other.detecttrain.**classifier_multiclass_densenet_train**(*ibs*, *gid_list*, *label_list*, *ensembles=3*, ***kwargs*)

```
>>> import uuid
>>> manifest_filepath = join(ibs.dbdir, 'flukebook_groundtruth.csv')
>>> with open(manifest_filepath, 'r') as manifest_file:
>>>     line_list = manifest_file.readlines()
>>>
>>> label_dict = {
>>>     'Left Dorsal Fin'  : 'left_dorsal_fin',
>>>     'Right Dorsal Fin' : 'right_dorsal_fin',
>>>     'Tail Fluke'       : 'tail_fluke',
>>> }
>>>
>>> uuid_list = []
>>> label_list = []
>>> for line in line_list:
>>>     line = line.strip().split(',')
>>>     assert len(line) == 2
>>>     uuid_, label_ = line
>>>     uuid_ = uuid.UUID(uuid_)
>>>     label_ = label_.strip()
>>>     print(uuid_, label_)
>>>     uuid_list.append(uuid_)
>>>     label_ = label_dict.get(label_, None)
>>>     assert label_ is not None
>>>     label_list.append(label_)
>>>
>>> gid_list = ibs.get_image_gids_from_uuid(uuid_list)
>>> assert None not in gid_list
>>> # archive_path = ibs.classifier_multiclass_densenet_train(gid_list, label_
↪list)
>>> ibs.classifier2_precision_recall_algo_display(test_gid_list=gid_list, test_
↪label_list=label_list)
```

wbia.other.detecttrain.**classifier_train**(*ibs*, ***kwargs*)

wbia.other.detecttrain.**detector_train**(*ibs*)

wbia.other.detecttrain.**labeler_train**(*ibs*, *species_list=None*, *species_mapping=None*, *viewpoint_mapping=None*, *ensembles=3*, ***kwargs*)

wbia.other.detecttrain.**labeler_train_wbia_cnn**(*ibs*, *species_list=None*, *species_mapping=None*, *viewpoint_mapping=None*, ***kwargs*)

wbia.other.detecttrain.**localizer_lightnet_train**(*ibs*, *species_list*, *cuda_device='0'*, *batches=60000*, *validate_with_accuracy=True*, *deploy_tag=None*, *cleanup=True*, *cleanup_all=True*, *deploy=True*, *cache_species_str=None*, ***kwargs*)

wbia.other.detecttrain.**localizer_yolo_train**(*ibs*, *species_list=None*, ***kwargs*)

wbia.other.detecttrain.**validate_model**(*cuda_str*, *python_exe*, *test_py_path*, *config_py_path*, *results_path*, *backup_path*, *validate_with_accuracy*, *deploy_path*, *deploy*, *deploy_tag*, *cleanup*, *cleanup_all*, *bin_path*, *cfg_path*, *data_path*, *weights_path*, *cache_species_str*)

1.10.8 wbia.other.duct_tape module

wbia.other.duct_tape.**enforce_unkonwn_name_is_explicit**(*ibs*)

wbia.other.duct_tape.**fix_compname_configs**(*ibs*)
duct tape to keep version in check

wbia.other.duct_tape.**fix_nulled_yaws**(*ibs*)

wbia.other.duct_tape.**remove_database_slag**(*ibs*, *delete_empty_names=False*, *delete_empty_imagesets=False*, *delete_annotations_for_missing_images=False*, *delete_image_labels_for_missing_types=False*, *delete_annot_labels_for_missing_types=False*, *delete_chips_for_missing_annotations=False*, *delete_features_for_missing_annotations=False*, *delete_invalid_eg_relations=False*, *delete_invalid_gl_relations=False*, *delete_invalid_al_relations=True*)

1.10.9 wbia.other.ibsfuncs module

developer convenience functions for ibs

TODO: need to split up into sub modules: consistency_checks feasibility_fixes move the export stuff to dbio

python -m utool.util_inspect check_module_usage –pat="ibsfuncs.py"

then there are also convineience functions that need to be ordered at least within this file

wbia.other.ibsfuncs.**add_next_imageset**(*ibs*)
Adds a new imageset to the database

wbia.other.ibsfuncs.**add_trivial_annotations**(*ibs*, **args*, ***kwargs*)

wbia.other.ibsfuncs.**aidstr**(*aid*, *ibs=None*, *notes=False*)
Helper to make a string from an aid

wbia.other.ibsfuncs.**alias_common_coco_species**(*ibs*, ***kwargs*)

wbia.other.ibsfuncs.**annotstr**(*ibs*, *aid*)

wbia.other.ibsfuncs.**assert_images_are_unique**(*ibs*, *gid_list=None*, *verbose=True*)

wbia.other.ibsfuncs.**assert_images_exist**(*ibs*, *gid_list=None*, *verbose=True*)

wbia.other.ibsfuncs.**assert_lblannot_rowids_are_type**(*ibs*, *lblannot_rowid_list*, *valid_lbltype_rowid*)

wbia.other.ibsfuncs.**assert_singleton_relationship**(*ibs*, *alrids_list*)

wbia.other.ibsfuncs.**assert_valid_aids**(*ibs*, *aid_list*, *verbose=False*, *veryverbose=False*, *msg=''*, *auuid_list=None*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*int*) – list of annotation ids
- **verbose** (*bool*) – verbosity flag(default = False)
- **veryverbose** (*bool*) – (default = False)

CommandLine: python -m wbia.other.ibsfuncs –test-assert_valid_aids

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> verbose = False
>>> veryverbose = False
>>> print('Asserting multiple')
>>> result = assert_valid_aids(ibs, aid_list, verbose, veryverbose)
>>> print('Asserting single')
>>> result = assert_valid_aids(ibs, aid_list[0:1], verbose, veryverbose)
>>> print('Asserting multiple incorrect')
>>> auuid_list = ibs.get_annot_uuids(aid_list) + [None]
>>> try:
>>>     result = assert_valid_aids(ibs, aid_list + [0], verbose, veryverbose,␣
↪auuid_list=auuid_list)
>>> except AssertionError:
>>>     print('Correctly got assertion')
>>> else:
>>>     assert False, 'should have failed'
>>> print('Asserting single incorrect')
>>> try:
>>>     result = assert_valid_aids(ibs, [0], verbose, veryverbose)
>>> except AssertionError:
>>>     print('Correctly got assertion')
>>> else:
>>>     assert False, 'should have failed'
>>> print(result)
>>> print(result)
```

wbia.other.ibsfuncs.**assert_valid_gids**(*ibs*, *gid_list*, *verbose=False*, *veryverbose=False*)

wbia.other.ibsfuncs.**assert_valid_names**(*name_list*)
Asserts that user specified names do not conflict with the standard unknown name

wbia.other.ibsfuncs.**assert_valid_species_texts**(*ibs*, *species_list*, *iswarning=True*)

wbia.other.ibsfuncs.**batch_rename_consecutive_via_species**(*ibs*, *imgsetid=None*, *location_text=None*, *notify_wildbook=True*, *assert_wildbook=True*)

wbia.other.ibsfuncs.**bytes2human**(*n*, *format='%(value).02f%(symbol)s'*, *symbols='customary'*)
(c) http://code.activestate.com/recipes/578019/
Convert n bytes into a human readable string based on format. symbols can be either "customary", "customary_ext", "iec" or "iec_ext", see: https://en.wikipedia.org/wiki/Binary_prefix#Specific_units_of_IEC_60027-2_A.2_and_ISO.2FIEC_80000

```
>>> bytes2human(0)
'0.0 B'
>>> bytes2human(0.9)
'0.0 B'
>>> bytes2human(1)
'1.0 B'
>>> bytes2human(1.9)
'1.0 B'
>>> bytes2human(1024)
'1.0 K'
>>> bytes2human(1048576)
'1.0 M'
>>> bytes2human(109951162777612739812378912 1)
'909.5 Y'
```

```
>>> bytes2human(9856, symbols="customary")
'9.6 K'
>>> bytes2human(9856, symbols="customary_ext")
'9.6 kilo'
>>> bytes2human(9856, symbols="iec")
'9.6 Ki'
>>> bytes2human(9856, symbols="iec_ext")
'9.6 kibi'
```

```
>>> bytes2human(10000, "%(value).1f %(symbol)s/sec")
'9.8 K/sec'
```

```
>>> # precision can be adjusted by playing with %f operator
>>> bytes2human(10000, format="%(value).5f %(symbol)s")
'9.76562 K'
```

wbia.other.ibsfuncs.**check_annot_consistency**(*ibs*, *aid_list=None*)

Parameters

- **ibs** (IBEISController) –
- **aid_list** (*list*) –

CommandLine: python -m wbia.other.ibsfuncs –test-check_annot_consistency

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> result = check_annot_consistency(ibs, aid_list)
>>> print(result)
```

wbia.other.ibsfuncs.**check_annot_corrupt_uuids**(*ibs*, *aid_list=None*)

```
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('PZ_MTEST')
>>> aid_list = ibs.get_valid_aids()
>>> check_annot_corrupt_uuids(ibs, aid_list)
```

wbia.other.ibsfuncs.**check_annot_overlap**(*ibs*, *gid_list=None*, *PIXELS=100.0*, *IOU=0.1*)

wbia.other.ibsfuncs.**check_annot_size**(*ibs*)

wbia.other.ibsfuncs.**check_annotmatch_consistency**(*ibs*)

wbia.other.ibsfuncs.**check_cache_purge**(*ibs*, *ttl_days=90*, *dryrun=True*, *squeeze=True*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) – (default = None)

CommandLine: python -m wbia.other.ibsfuncs –exec-check_cache_purge

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> result = check_cache_purge(ibs)
>>> print(result)
```

wbia.other.ibsfuncs.**check_cache_purge_delete_worker**(*args*)

wbia.other.ibsfuncs.**check_cache_purge_exists_worker**(*args*)

wbia.other.ibsfuncs.**check_cache_purge_parallel_wrapper**(*func*, *arguments_list*)

wbia.other.ibsfuncs.**check_cache_purge_time_worker**(*args*)

wbia.other.ibsfuncs.**check_chip_existence**(*ibs*, *aid_list=None*)

wbia.other.ibsfuncs.**check_exif_data**(*ibs*, *gid_list*)

TODO CALL SCRIPT

wbia.other.ibsfuncs.**check_for_unregistered_images**(*ibs*)

wbia.other.ibsfuncs.**check_ggr_valid_aids**(*ibs*, *aid_list*, *species='zebra_grevys'*, *threshold=0.75*, *enable_grid=True*, *verbose=True*)

wbia.other.ibsfuncs.**check_image_bit_depth**(*ibs*, *gid_list=None*)

wbia.other.ibsfuncs.**check_image_bit_depth_worker**(*gpath*)

wbia.other.ibsfuncs.**check_image_consistency**(*ibs*, *gid_list=None*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) – (default = None)

CommandLine: python -m wbia.other.ibsfuncs –exec-check_image_consistency –db=GZ_Master1

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> gid_list = None
>>> result = check_image_consistency(ibs, gid_list)
>>> print(result)
```

wbia.other.ibsfuncs.**check_image_duplcates**(*ibs*, *gid_list=None*)

wbia.other.ibsfuncs.**check_image_loadable**(*ibs*, *gid_list=None*)

wbia.other.ibsfuncs.**check_image_loadable_worker**(*gpath*, *orient*)

wbia.other.ibsfuncs.**check_image_uuid_consistency**(*ibs*, *gid_list=None*)

Checks to make sure image uuids are computed detemenistically by recomputing all guuids and checking that they are equal to what is already there.

VERY SLOW

CommandLine: python -m wbia.other.ibsfuncs –test-check_image_uuid_consistency –db=PZ_Master0 python -m wbia.other.ibsfuncs –test-check_image_uuid_consistency –db=GZ_Master1 python -m wbia.other.ibsfuncs –test-check_image_uuid_consistency python -m wbia.other.ibsfuncs –test-check_image_uuid_consistency –db lynx

Example

```
>>> # SCRIPT
>>> import wbia
>>> import utool as ut
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> images = ibs.images()
>>> # Check only very the largest files
>>> #bytes_list_ = [
>>> #    ut.get_file_nBytes(path)
>>> #    for path in ut.ProgIter(images.paths, lbl='reading nbytes')]
>>> #sortx = ut.list_argsort(bytes_list_, reverse=True)[0:10]
>>> #images = images.take(sortx)
>>> gid_list = list(images)
>>> wbia.other.ibsfuncs.check_image_uuid_consistency(ibs, gid_list)
```

wbia.other.ibsfuncs.**check_name_consistency**(*ibs*, *nid_list*)

Parameters

- **ibs** (IBEISController) – wbia controller object

- **nid_list** (*list*) –

CommandLine: python -m wbia.other.ibsfuncs –test-check_name_consistency

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> nid_list = ibs._get_all_known_nids()
>>> result = check_name_consistency(ibs, nid_list)
>>> print(result)
```

wbia.other.ibsfuncs.**check_name_mapping_consistency**(*ibs*, *nx2_aids*)
checks that all the aids grouped in a name ahave the same name

wbia.other.ibsfuncs.**commit_ggr_fix_gps**(*ibs*, ***kwargs*)

wbia.other.ibsfuncs.**compare_nested_props**(*ibs*, *aids1_list*, *aids2_list*, *getter_func*, *cmp_func*)
Compares properties of query vs database annotations

grouped_qaids = aids1_list grouped_groundtruth_list = aids2_list

getter_func = ibs.get_annot_yaws cmp_func = vt.ori_distance

getter_func = ibs.get_annot_image_unixtimes_asfloat cmp_func = ut.unixtime_hourdiff
ExpandNestedComparisions: import itertools list(map(list, itertools.starmap(ut.iprod, zip(aids1_list, aids2_list))))

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aids1_list** (*list*) –
- **aids2_list** (*list*) –
- **getter_func** –
- **cmp_func** –

Returns

Return type list of ndarrays

CommandLine: python -m wbia.other.ibsfuncs –exec-compare_nested_props –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> aids1_list = [ibs.get_valid_aids()[8:11]]
>>> aids2_list = [ibs.get_valid_aids()[8:11]]
>>> getter_func = ibs.get_annot_image_unixtimes_asfloat
>>> cmp_func = ut.unixtime_hourdiff
>>> result = compare_nested_props(ibs, aids1_list, aids2_list, getter_func, cmp_
↪func)
```

(continues on next page)

(continued from previous page)

```
>>> print(result)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

wbia.other.ibsfuncs.**compute_all_chips**(*ibs*, *aid_list=None*, ***kwargs*)
Executes lazy evaluation of all chips

wbia.other.ibsfuncs.**compute_ggr_fix_gps_contributors_aids**(*ibs*, *min_diff=600*, *individual=False*)

wbia.other.ibsfuncs.**compute_ggr_fix_gps_contributors_gids**(*ibs*, *min_diff=600*, *individual=False*)

wbia.other.ibsfuncs.**compute_ggr_fix_gps_names**(*ibs*, *min_diff=1800*)

wbia.other.ibsfuncs.**compute_ggr_imagesets**(*ibs*, *gid_list=None*, *min_diff=86400*, *individual=False*, *purge_all_old=False*)

wbia.other.ibsfuncs.**compute_ggr_path_dict**(*ibs*)

wbia.other.ibsfuncs.**compute_occurrences**(*ibs*, *config=None*)
Clusters ungrouped images into imagesets representing occurrences
CommandLine: python -m wbia.control.IBEISControl –test-compute_occurrences

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.control.IBEISControl import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> ibs.compute_occurrences(config={'use_gps': False, 'seconds_thresh': 600})
>>> ibs.update_special_imagesets()
>>> # Remove some images from a non-special imageset
>>> nonspecial_imgsetids = [i for i in ibs.get_valid_imgsetids() if i not in ibs.
↪get_special_imgsetids()]
>>> images_to_remove = ibs.get_imageset_gids(nonspecial_imgsetids[0:1])[0][0:1]
>>> ibs.unrelate_images_and_imagesets(images_to_remove,nonspecial_imgsetids[0:1]␣
↪* len(images_to_remove))
>>> ibs.update_special_imagesets()
>>> ungr_imgsetid = ibs.get_imageset_imgsetids_from_text(const.UNGROUPED_IMAGES_
↪IMAGESETTEXT)
>>> ungr_gids = ibs.get_imageset_gids([ungr_imgsetid])[0]
>>> #Now let's make sure that when we recompute imagesets, our non-special␣
↪imgsetid remains the same
>>> print('PRE COMPUTE: ImageSets are %r' % ibs.get_valid_imgsetids())
>>> print('Containing: %r' % ibs.get_imageset_gids(ibs.get_valid_imgsetids()))
>>> ibs.compute_occurrences(config={'use_gps': False, 'seconds_thresh': 600})
>>> print('COMPUTE: New imagesets are %r' % ibs.get_valid_imgsetids())
>>> print('Containing: %r' % ibs.get_imageset_gids(ibs.get_valid_imgsetids()))
>>> ibs.update_special_imagesets()
>>> print('UPDATE SPECIAL: New imagesets are %r' % ibs.get_valid_imgsetids())
>>> print('Containing: %r' % ibs.get_imageset_gids(ibs.get_valid_imgsetids()))
>>> assert(images_to_remove[0] not in ibs.get_imageset_gids(nonspecial_
↪imgsetids[0:1])[0])
```

wbia.other.ibsfuncs.**compute_occurrences_smart**(*ibs*, *gid_list*, *smart_xml_fpath*)
Function to load and process a SMART patrol XML file

wbia.other.ibsfuncs.**convert_empty_images_to_annotations**(*ibs*)
images without chips are given an ANNOTATION over the entire image

wbia.other.ibsfuncs.**copy_imagesets**(*ibs*, *imgsetid_list*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **imgsetid_list** (*list*) –

Returns new_imgsetid_list

Return type list

CommandLine: python -m wbia.other.ibsfuncs –test-copy_imagesets

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> ibs.delete_all_imagesets()
>>> ibs.compute_occurrences(config={'use_gps': False, 'seconds_thresh': 600})
>>> imgsetid_list = ibs.get_valid_imgsetids()
>>> new_imgsetid_list = copy_imagesets(ibs, imgsetid_list)
>>> result = str(ibs.get_imageset_text(new_imgsetid_list))
>>> assert [2] == list(set(map(len, ibs.get_image_imgsetids(ibs.get_valid_
↪gids()))))
>>> print(result)
>>> ibs.delete_all_imagesets()
>>> ibs.compute_occurrences(config={'use_gps': False, 'seconds_thresh': 600})
```

wbia.other.ibsfuncs.**create_ggr_match_leaves_recursive**(*ibs*, *tag*, *imageset_rowid_list*, *k*, *level=0*, *index=0*)

wbia.other.ibsfuncs.**create_ggr_match_trees**(*ibs*)

CommandLine: python -m wbia.other.ibsfuncs create_ggr_match_trees

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> from os.path import expanduser
>>> import wbia  # NOQA
>>> default_dbdir = join('/', 'data', 'wbia', 'GGR2-IBEIS')
>>> # default_dbdir = expanduser(join('~', 'data', 'GGR2-IBEIS'))
>>> dbdir = ut.get_argval('--dbdir', type_=str, default=default_dbdir)
>>> ibs = wbia.opendb(dbdir=dbdir)
>>> imageset_rowid_list = ibs.create_ggr_match_trees()
```

wbia.other.ibsfuncs.**create_new_imageset_from_images**(*ibs*, *gid_list*, *new_imgsetid=None*)

Parameters **gid_list** (*list*) –

CommandLine: python -m wbia.other.ibsfuncs –test-create_new_imageset_from_images

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids()[::2]
>>> new_imgsetid = create_new_imageset_from_images(ibs, gid_list)
>>> result = new_imgsetid
>>> print(result)
```

wbia.other.ibsfuncs.**create_new_imageset_from_names**(*ibs*, *nid_list*)

Parameters **nid_list** (*list*) –

CommandLine: python -m wbia.other.ibsfuncs –test-create_new_imageset_from_names

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> nid_list = ibs._get_all_known_nids()[0:2]
>>> new_imgsetid = ibs.create_new_imageset_from_names(nid_list)
>>> # clean up
>>> ibs.delete_imagesets(new_imgsetid)
>>> result = new_imgsetid
>>> print(result)
```

wbia.other.ibsfuncs.**dans_lists**(*ibs*, *positives=10*, *negatives=10*, *verbose=False*)

wbia.other.ibsfuncs.**delete_all_annotations**(*ibs*)

Carefull with this function. Annotations are not recomputable

wbia.other.ibsfuncs.**delete_all_chips**(*ibs*)

wbia.other.ibsfuncs.**delete_all_features**(*ibs*)

wbia.other.ibsfuncs.**delete_all_imagesets**(*ibs*)

wbia.other.ibsfuncs.**delete_all_recomputable_data**(*ibs*)

Delete all cached data including chips and imagesets

wbia.other.ibsfuncs.**delete_cache**(*ibs*, *delete_imagesets=False*)

Deletes the cache directory in the database directory. Can specify to delete encoutners as well.

CommandLine: python -m wbia delete_cache –db testdb1

Example

```
>>> # SCRIPT
>>> import wbia
>>> ibs = wbia.opendb()
>>> result = ibs.delete_cache()
```

wbia.other.ibsfuncs.**delete_cachedir**(*ibs*)

Deletes the cache directory in the database directory.

CommandLine: python -m wbia.other.ibsfuncs delete_cachedir python -m wbia delete_cachedir –db testdb1

Example

```
>>> # SCRIPT
>>> import wbia
>>> ibs = wbia.opendb()
>>> result = ibs.delete_cachedir()
```

wbia.other.ibsfuncs.**delete_flann_cachedir**(*ibs*)

wbia.other.ibsfuncs.**delete_neighbor_cache**(*ibs*)

wbia.other.ibsfuncs.**delete_qres_cache**(*ibs*)

Parameters **ibs** (IBEISController) – wbia controller object

CommandLine: python -m wbia –tf delete_qres_cache python -m wbia –tf delete_qres_cache –db PZ_MTEST python -m wbia –tf delete_qres_cache –db PZ_Master1

Example

```
>>> # SCRIPT
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> result = delete_qres_cache(ibs)
>>> print(result)
```

wbia.other.ibsfuncs.**delete_thumbnails**(*ibs*)

wbia.other.ibsfuncs.**delete_unregistered_images**(*ibs*, *verbose=True*)

wbia.other.ibsfuncs.**delete_wbia_database**(*dbdir*)

wbia.other.ibsfuncs.**ensure_annotation_data**(*ibs*, *aid_list*, *chips=True*, *feats=True*, *featweights=False*)

wbia.other.ibsfuncs.**ensure_unix_gpaths**(*gpath_list*)

Asserts that all paths are given with forward slashes. If not it fixes them

wbia.other.ibsfuncs.**export_ggr_folders**(*ibs*, *output_path=None*)

wbia.other.ibsfuncs.**export_to_hotspotter**(*ibs*)

wbia.other.ibsfuncs.**filter_aids_count**(*ibs*, *aid_list=None*, *pre_unixtime_sort=True*)

wbia.other.ibsfuncs.**filter_aids_to_quality**(*ibs*, *aid_list*, *minqual*, *unknown_ok=True*, *speedhack=True*)

DEPRICATE

```
>>> import wbia
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> ibs = wbia.opendb(defaultdb='PZ_Master1')
>>> aid_list = ibs.get_valid_aids()
>>> minqual = 'good'
>>> x1 = filter_aids_to_quality(ibs, aid_list, 'good', True, speedhack=True)
>>> x2 = filter_aids_to_quality(ibs, aid_list, 'good', True, speedhack=False)
```

wbia.other.ibsfuncs.**filter_aids_to_species**(*ibs*, *aid_list*, *species*, *speedhack=True*)

Parameters

- **ibs** (IBEISController) – wbia controller object

- **aid_list** (*int*) – list of annotation ids
- **species** –

Returns aid_list_

Return type list

CommandLine: python -m wbia.other.ibsfuncs –exec-filter_aids_to_species

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> species = wbia.const.TEST_SPECIES.ZEB_GREVY
>>> aid_list_ = filter_aids_to_species(ibs, aid_list, species)
>>> result = 'aid_list_ = %r' % (aid_list_,)
>>> print(result)
aid_list_ = [9, 10]
```

wbia.other.ibsfuncs.**filter_aids_to_viewpoint**(*ibs*, *aid_list*, *valid_yaws*, *unknown_ok=True*)

Removes aids that do not have a valid yaw

TODO: rename to valid_viewpoint because this func uses category labels

valid_yaws = [‘primary’, ‘primary1’, ‘primary-1’]

wbia.other.ibsfuncs.**filter_aids_without_name**(*ibs*, *aid_list*, *invert=False*, *speedhack=True*)

Remove aids without names

Example

```
>>> # ENABLE_DOCTEST
>>> import wbia
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> annots = ibs.annots(aid_list)
>>> aid_list1_ = ibs.filter_aids_without_name(aid_list)
>>> aid_list2_ = ibs.filter_aids_without_name(aid_list, invert=True)
>>> annots1_ = ibs.annots(aid_list1_)
>>> annots2_ = ibs.annots(aid_list2_)
>>> assert len(annots1_) + len(annots2_) == len(annots)
>>> assert np.all(np.array(annots1_.nids) > 0)
>>> assert len(annots1_) == 9
>>> assert np.all(np.array(annots2_.nids) < 0)
>>> assert len(annots2_) == 4
```

wbia.other.ibsfuncs.**filter_aids_without_timestamps**(*ibs*, *aid_list*, *invert=False*)

Removes aids without timestamps aid_list = ibs.get_valid_aids()

wbia.other.ibsfuncs.**filter_annots_using_minimum_timedelta**(*ibs*, *aid_list*, *min_timedelta*)

Uses a dynamic program to find the maximum number of annotations that are above the minimum timedelta requirement.

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** –
- **min_timedelta** –

CommandLine: python -m wbia.other.ibsfuncs –exec-filter_annots_using_minimum_timedelta python -m wbia.other.ibsfuncs –exec-filter_annots_using_minimum_timedelta –db PZ_Master1

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> aid_list = ibs.get_valid_aids()
>>> aid_list = ibs.filter_aids_without_timestamps(aid_list)
>>> print('Before')
>>> ibs.print_annot_stats(aid_list, min_name_hourdist=True)
>>> min_timedelta = 60 * 60 * 24
>>> filtered_aids = filter_annots_using_minimum_timedelta(ibs, aid_list, min_
↪timedelta)
>>> print('After')
>>> ibs.print_annot_stats(filtered_aids, min_name_hourdist=True)
>>> ut.quit_if_noshow()
>>> wbia.other.dbinfo.hackshow_names(ibs, aid_list)
>>> wbia.other.dbinfo.hackshow_names(ibs, filtered_aids)
>>> ut.show_if_requested()
```

wbia.other.ibsfuncs.**filter_junk_annotations**(*ibs*, *aid_list*)

remove junk annotations from a list

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*int*) – list of annotation ids

Returns filtered_aid_list

Return type list

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> filtered_aid_list = filter_junk_annotations(ibs, aid_list)
>>> result = str(filtered_aid_list)
>>> print(result)
```

wbia.other.ibsfuncs.**find_unlabeled_name_members**(*ibs*, ***kwargs*)
Find annots where some members of a name have information but others do not.

Parameters **ibs** (IBEISController) – wbia controller object

CommandLine: python -m wbia.other.ibsfuncs –exec-find_unlabeled_name_members –qual

Example

```
>>> # SCRIPT
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_Master1')
>>> defaultdict = dict(ut.parse_func_kwarg_keys(find_unlabeled_name_members, with_
↪vals=True))
>>> kwargs = ut.argparse_dict(defaultdict)
>>> result = find_unlabeled_name_members(ibs, **kwargs)
>>> print(result)
```

wbia.other.ibsfuncs.**fix_and_clean_database**(*ibs*)
Function to run all database cleanup scripts

Rename to run_cleanup_scripts
Break into two funcs: run_cleanup_scripts run_fixit_scripts
CONSITENCY CHECKS TODO:

- check that annotmatches marked as False do not have the same name for similar viewpoints.
- check that photobombs are have different names
- warn if scenery matches have the same name

wbia.other.ibsfuncs.**fix_coco_species**(*ibs*, ***kwargs*)

wbia.other.ibsfuncs.**fix_exif_data**(*ibs*, *gid_list*)
TODO CALL SCRIPT

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) – list of image ids

CommandLine: python -m wbia.other.ibsfuncs –exec-fix_exif_data

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='lynx')
>>> gid_list = ibs.get_valid_gids()
>>> result = fix_exif_data(ibs, gid_list)
>>> print(result)
```

wbia.other.ibsfuncs.**fix_ggr_qr_codes**(*ibs*, *imageset_qr_dict*)

wbia.other.ibsfuncs.**fix_invalid_annotmatches**(*ibs*)

wbia.other.ibsfuncs.**fix_invalid_name_texts**(*ibs*)
Ensure that no name text is empty or '____'

Parameters **ibs** (IBEISController) – wbia controller object

CommandLine: python -m wbia.other.ibsfuncs –test-fix_invalid_names

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> result = fix_invalid_name_texts(ibs)
>>> print(result)
```

ibs.set_name_texts(nid_list[3], '____') ibs.set_name_texts(nid_list[2], '')

wbia.other.ibsfuncs.**fix_invalid_nids**(*ibs*)

Make sure that all rowids are greater than 0

We can only handle there being a name with rowid 0 if it is UNKNOWN. In this case we safely delete it, but anything more complicated needs to be handled anually

Parameters **ibs** (IBEISController) – wbia controller object

CommandLine: python -m wbia.other.ibsfuncs –test-fix_invalid_nids

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> result = fix_invalid_nids(ibs)
>>> print(result)
```

wbia.other.ibsfuncs.**fix_remove_visual_dupliate_annotations**(*ibs*)

depricate because duplicate visual_uuids are no longer allowed to be duplicates

Add to clean database?

removes visually duplicate annotations

Parameters **ibs** (IBEISController) –

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('GZ_ALL')
>>> fix_remove_visual_dupliate_annotations(ibs)
```

wbia.other.ibsfuncs.**fix_unknown_exemplars**(*ibs*)

Goes through all of the annotations, and sets their exemplar flag to 0 if it is associated with an unknown annotation

wbia.other.ibsfuncs.**fix_zero_features**(*ibs*)

wbia.other.ibsfuncs.**flag_aids_count**(*ibs*, *aid_list*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*int*) – list of annotation ids
- **pre_unixtime_sort** (*bool*) –

Returns

Return type list

CommandLine: python -m wbia.other.ibsfuncs –test-flag_aids_count

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> gzc_flag_list = flag_aids_count(ibs, aid_list)
>>> result = gzc_flag_list
>>> print(result)
[False, True, False, False, True, False, True, True, False, True, False, True,␣
↪True]
```

wbia.other.ibsfuncs.**get_aids_with_groundtruth**(*ibs*)
returns aids with valid groundtruth

wbia.other.ibsfuncs.**get_annot_bbox_area**(*ibs*, *aid_list*)

wbia.other.ibsfuncs.**get_annot_been_adjusted**(*ibs*, *aid_list*)
Returns if a bounding box has been adjusted from defaults set in use_images_as_annotations Very hacky very heurstic.

wbia.other.ibsfuncs.**get_annot_encounter_text**(*ibs*, *aids*)
Encounter identifier for annotations

wbia.other.ibsfuncs.**get_annot_fgweights_subset**(*ibs*, *aid_list*, *fxs_list*, *config2_=None*)

wbia.other.ibsfuncs.**get_annot_info**(*ibs*, *aid_list*, *default=False*, *reference_aid=None*, ***kwargs*)

Parameters

- **ibs** (*wbia.IBEISController*) – wbia controller object
- **aid_list** (*list*) – list of annotation rowids
- **default** (*bool*) – (default = False)

Returns infodict_list

Return type list

CommandLine: python -m wbia.other.ibsfuncs –exec-get_annot_info –tb

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ibs.get_valid_aids()[0:2]
>>> default = True
>>> infodict_list = ibs.get_annot_info(1, default)
>>> result = ('infodict_list = %s' % (ut.repr2(infodict_list, nl=4),))
>>> print(result)
```

wbia.other.ibsfuncs.**get_annot_instancelist**(*ibs*, *aid_list*)

wbia.other.ibsfuncs.**get_annot_intermediate_viewpoint_stats**(*ibs*, *aids*, *size=2*)

```
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> aids = available_aids
```

wbia.other.ibsfuncs.**get_annot_lazy_dict**(*ibs*, *aid*, *config2_=None*)

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **aid** (*int*) – annotation id
- **config2** (*dict*) – (default = None)

Returns metadata

Return type ut.LazyDict

CommandLine: python -m wbia.other.ibsfuncs –exec-get_annot_lazy_dict –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid = 1
>>> config2_ = None
>>> metadata = get_annot_lazy_dict(ibs, aid, config2_)
>>> result = ('metadata = %s' % (ut.repr3(metadata),))
>>> print(result)
```

wbia.other.ibsfuncs.**get_annot_lazy_dict2**(*ibs*, *aid*, *config=None*)

DEPRICATE FOR ibs.annots

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **aid** (*int*) – annotation id
- **config** (*dict*) – (default = None)

Returns metadata

Return type ut.LazyDict

CommandLine: python -m wbia.other.ibsfuncs –exec-get_annot_lazy_dict2 –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid = 1
>>> config = {'dim_size': 450}
>>> metadata = get_annot_lazy_dict2(ibs, aid, config)
>>> result = ('metadata = %s' % (ut.repr3(metadata),))
>>> print(result)
```

wbia.other.ibsfuncs.**get_annot_occurrence_text**(*ibs*, *aids*)

Occurrence identifier for annotations

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **aids** (*list*) – list of annotation rowids

Returns occur_texts

Return type list

CommandLine: python -m wbia.other.ibsfuncs get_annot_occurrence_text –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aids = ibs.get_valid_aids()
>>> occur_texts = get_annot_occurrence_text(ibs, aids)
>>> result = ('occur_texts = %s' % (ut.repr2(occur_texts),))
>>> print(result)
```

wbia.other.ibsfuncs.**get_annot_pair_lazy_dict**(*ibs*, *qaid*, *daid*, *qconfig2_=None*, *dconfig2_=None*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **qaid** (*int*) – query annotation id
- **daid** –
- **qconfig2** (*dict*) – (default = None)
- **dconfig2** (*dict*) – (default = None)

CommandLine: python -m wbia.other.ibsfuncs –exec-get_annot_pair_lazy_dict

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
```

(continues on next page)

(continued from previous page)

```
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> qaid, daid = ibs.get_valid_aids()[0:2]
>>> qconfig2_ = None
>>> dconfig2_ = None
>>> result = get_annot_pair_lazy_dict(ibs, qaid, daid, qconfig2_, dconfig2_)
>>> print(result)
```

wbia.other.ibsfuncs.**get_annot_primary_imageset**(*ibs*, *aid_list=None*)

wbia.other.ibsfuncs.**get_annot_quality_viewpoint_subset**(*ibs*, *aid_list=None*, *annots_per_view=2*, *max_annots=None*, *verbose=False*, *prog_hook=None*, *allow_unknown=False*)

CommandLine: python -m wbia.other.ibsfuncs –exec-get_annot_quality_viewpoint_subset –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ut.exec_funckw(get_annot_quality_viewpoint_subset, globals())
>>> ibs = wbia.opendb('testdb2')
>>> new_flag_list = get_annot_quality_viewpoint_subset(ibs)
>>> result = sum(new_flag_list)
>>> print(result)
38
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ut.exec_funckw(get_annot_quality_viewpoint_subset, globals())
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = [1]
>>> new_flag_list = get_annot_quality_viewpoint_subset(ibs, aid_list, allow_
↪unknown=True)
>>> result = sum(new_flag_list)
>>> print(result)
1
```

wbia.other.ibsfuncs.**get_annot_stats_dict**(*ibs*, *aids*, *prefix=''*, *forceall=False*, *old=True*, *use_hist=False*, ***kwargs*)

stats for a set of annots

Parameters

- **ibs** (*wbia.IBEISController*) – wbia controller object
- **aids** (*list*) – list of annotation rowids
- **prefix** (*str*) – (default = '')

Kwargs: hashid, per_name, per_qual, per_vp, per_name_vpedge, per_image, min_name_hourdist

Returns aid_stats_dict

Return type dict

CommandLine: python -m wbia get_annot_stats_dict –db WWF_Lynx –all python -m wbia get_annot_stats_dict –db EWT_Cheetahs –all python -m wbia get_annot_stats_dict –db PZ_PB_RF_TRAIN –all python -m wbia get_annot_stats_dict –db PZ_Master1 –all

python -m wbia.other.ibsfuncs –exec-get_annot_stats_dict python -m wbia.other.ibsfuncs –exec-get_annot_stats_dict –db PZ_PB_RF_TRAIN –use-hist=True –old=False –per_name_vpedge=False python -m wbia.other.ibsfuncs –exec-get_annot_stats_dict –db PZ_PB_RF_TRAIN –use-hist=False –old=False –per_name_vpedge=False

python -m wbia.other.ibsfuncs –exec-get_annot_stats_dict –db PZ_MTEST –use-hist –per_name_vpedge=False python -m wbia.other.ibsfuncs –exec-get_annot_stats_dict –db PZ_MTEST –use-hist –per_name_vpedge=False

python -m wbia.other.ibsfuncs –exec-get_annot_stats_dict –db PZ_Master1 –per_name_vpedge=True python -m wbia.other.ibsfuncs –exec-get_annot_stats_dict –db PZ_Master1 –min_name_hourdist=True python -m wbia.other.ibsfuncs –exec-get_annot_stats_dict –db GZ_ALL –min_name_hourdist=True –all python -m wbia.other.ibsfuncs –exec-get_annot_stats_dict –db GZ_Master1 –all python -m wbia.other.ibsfuncs –exec-get_annot_stats_dict –db PZ_Master1 –min_name_hourdist=True –all python -m wbia.other.ibsfuncs –exec-get_annot_stats_dict –db NNP_MasterGIRM_core –min_name_hourdist=True –all

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aids = ibs.annots().aids
>>> stats = ibs.get_annot_stats_dict(aids)
>>> import ubelt as ub
>>> print('annot_stats = {}'.format(ub.repr2(stats, nl=1)))
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aids = wbia.testdata_aids(ibs=ibs)
>>> prefix = ''
>>> kwkeys = ut.parse_func_kwarg_keys(get_annot_stats_dict)
>>> #default = True if ut.get_argflag('--all') else None
>>> default = None if ut.get_argflag('--notall') else True
>>> kwargs = ut.argparse_dict(dict(zip(kwkeys, [default] * len(kwkeys))))
>>> #ut.argparse_funckw(ibs.get_annot_stats_dict)
>>> print('kwargs = %r' % (kwargs,))
>>> old = ut.get_argval('--old', default=True)
>>> use_hist = ut.get_argval('--use_hist', default=True)
>>> aid_stats_dict = get_annot_stats_dict(ibs, aids, prefix, use_hist=use_hist,␣
↪old=old, **kwargs)
```

(continues on next page)

(continued from previous page)

```
>>> result = ('aid_stats_dict = %s' % (ub.repr2(aid_stats_dict, strkeys=True,␣
↪strvals=True, nl=2, precision=2),))
>>> print(result)
```

wbia.other.ibsfuncs.**get_annot_vecs_subset**(*ibs*, *aid_list*, *fxs_list*, *config2_=None*)

wbia.other.ibsfuncs.**get_annotconfig_stats**(*ibs*, *qaids*, *daids*, *verbose=False*, *combined=False*, *combo_gt_info=True*, *combo_enc_info=False*, *combo_dists=True*, *split_matchable_data=True*, ***kwargs*)

Gets statistics about a query / database set of annotations

USEFUL DEVELOPER FUNCTION

TODO: this function should return non-string values in dictionaries. The print function should do string conversions

Parameters

- **ibs** (IBEISController) – wbia controller object
- **qaids** (*list*) – query annotation ids
- **daids** (*list*) – database annotation ids

SeeAlso: wbia.dbinfo.print_qd_info ibs.get_annot_stats_dict ibs.print_annotconfig_stats(qaid_list, daid_list)

CommandLine: python -m wbia.other.ibsfuncs get_annotconfig_stats –db PZ_MTEST -a default python -m wbia.other.ibsfuncs get_annotconfig_stats –db testdb1 -a default python -m wbia.other.ibsfuncs get_annotconfig_stats –db PZ_MTEST -a controlled python -m wbia.other.ibsfuncs get_annotconfig_stats –db PZ_FlankHack -a default:qaids=allgt python -m wbia.other.ibsfuncs get_annotconfig_stats –db PZ_MTEST -a controlled:per_name=2,min_gt=4

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> from wbia.init import main_helpers
>>> kwargs = {'per_enc': True, 'enc_per_name': True}
>>> ibs, qaids, daids = main_helpers.testdata_expanded_aids(
...     defaultdb='testdb1', a='default:qsize=3')
>>> stat_dict = get_annotconfig_stats(ibs, qaids, daids, **kwargs)
>>> stats_str2 = ut.repr2(stat_dict, si=True, nl=True, nobr=False)
>>> print(stats_str2)
```

wbia.other.ibsfuncs.**get_annotpair_speeds**(*ibs*, *aid_pairs*, *unique_aids=None*)

wbia.other.ibsfuncs.**get_annots_per_name_stats**(*ibs*, *aid_list*, ***kwargs*)

wbia.other.ibsfuncs.**get_consecutive_newname_list_via_species**(*ibs*, *imgsetid=None*, *location_text=None*, *wildbook_existing_name_list=[]*)

Just creates the nams, but does not set them

Parameters **ibs** (IBEISController) – wbia controller object

CommandLine: python -m wbia.other.ibsfuncs –test-get_consecutive_newname_list_via_species

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> ibs._clean_species()
>>> imgsetid = None
>>> new_nid_list, new_name_list = get_consecutive_newname_list_via_species(ibs,␣
↪imgsetid=imgsetid)
>>> result = ut.repr2((new_nid_list, new_name_list), nl=1)
>>> print(result)
(
    [1, 2, 3, 4, 5, 6, 7],
    ['IBEIS_PZ_0001', 'IBEIS_PZ_0002', 'IBEIS_UNKNOWN_0001', 'IBEIS_UNKNOWN_0002',
↪ 'IBEIS_GZ_0001', 'IBEIS_PB_0001', 'IBEIS_UNKNOWN_0003'],
)
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> ibs._clean_species()
>>> ibs.delete_all_imagesets()
>>> ibs.compute_occurrences(config={'use_gps': False, 'seconds_thresh': 600})
>>> imgsetid = ibs.get_valid_imgsetids()[1]
>>> new_nid_list, new_name_list = get_consecutive_newname_list_via_species(ibs,␣
↪imgsetid=imgsetid)
>>> result = ut.repr2((new_nid_list, new_name_list), nl=1)
>>> print(result)
(
    [4, 5, 6, 7],
    ['IBEIS_UNKNOWN_Occurrence_1_0001', 'IBEIS_GZ_Occurrence_1_0001', 'IBEIS_PB_
↪Occurrence_1_0001', 'IBEIS_UNKNOWN_Occurrence_1_0002'],
)
```

wbia.other.ibsfuncs.**get_database_species**(*ibs*, *aid_list=None*)

CommandLine: python -m wbia.other.ibsfuncs –test-get_database_species

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> result = ut.repr2(ibs.get_database_species(), nl=False)
>>> print(result)
['____', 'bear_polar', 'zebra_grevys', 'zebra_plains']
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('PZ_MTEST')
>>> result = ut.repr2(ibs.get_database_species(), nl=False)
>>> print(result)
['zebra_plains']
```

wbia.other.ibsfuncs.**get_database_species_count**(*ibs*, *aid_list=None*, *BATCH_SIZE=25000*)

CommandLine: python -m wbia.other.ibsfuncs –test-get_database_species_count

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> #print(ut.repr2(wbia.opendb('PZ_Master0').get_database_species_count()))
>>> ibs = wbia.opendb('testdb1')
>>> result = ut.repr2(ibs.get_database_species_count(BATCH_SIZE=2), nl=False)
>>> print(result)
{'zebra_plains': 6, '____': 3, 'zebra_grevys': 2, 'bear_polar': 2}
```

wbia.other.ibsfuncs.**get_dbinfo_str**(*ibs*)

wbia.other.ibsfuncs.**get_dbname_alias**(*ibs*)

convinience for plots

wbia.other.ibsfuncs.**get_dir_size**

//stackoverflow.com/a/34580363

Type REF

Type https

wbia.other.ibsfuncs.**get_dominant_species**(*ibs*, *aid_list*)

Parameters **aid_list** (*int*) – list of annotation ids

CommandLine: python -m wbia.other.ibsfuncs –test-get_dominant_species

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> result = get_dominant_species(ibs, aid_list)
>>> print(result)
zebra_plains
```

wbia.other.ibsfuncs.**get_extended_viewpoints**(*base_yaw_text*, *towards='front'*, *num1=0*, *num2=None*, *include_base=True*)

Given a viewpoint returns the acceptable viewpoints around it

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> yaw_text_list = ['left', 'right', 'back', 'front']
>>> towards = 'front'
>>> num1 = 1
>>> num2 = 0
>>> include_base = False
>>> extended_yaws_list = [get_extended_viewpoints(base_yaw_text, towards, num1,␣
↪num2, include_base)
>>>                       for base_yaw_text in yaw_text_list]
>>> result = ('extended_yaws_list = %s' % (ut.repr2(extended_yaws_list),))
>>> print(result)
extended_yaws_list = [['frontleft'], ['frontright'], ['backleft'], ['frontleft']]
```

wbia.other.ibsfuncs.**get_image_annotation_bboxes**(*ibs*, *gid_list*)

wbia.other.ibsfuncs.**get_image_annotation_thetas**(*ibs*, *gid_list*)

wbia.other.ibsfuncs.**get_image_instancelist**(*ibs*, *gid_list*)

wbia.other.ibsfuncs.**get_image_lazydict**(*ibs*, *gid*, *config=None*)

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **aid** (*int*) – annotation id
- **config** (*dict*) – (default = None)

Returns metadata

Return type ut.LazyDict

CommandLine: python -m wbia.other.ibsfuncs –exec-get_annot_lazy_dict2 –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> gid = 1
```

wbia.other.ibsfuncs.**get_image_time_statstr**(*ibs*, *gid_list=None*)

wbia.other.ibsfuncs.**get_infostr**(*ibs*)

Returns sort printable database information

Parameters **ibs** (IBEISController) – wbia controller object

Returns infostr

Return type str

wbia.other.ibsfuncs.**get_location_text**(*ibs*, *location_text*, *default_location_text*)

wbia.other.ibsfuncs.**get_missing_gids**(*ibs*, *gid_list=None*)

Finds gids with broken links to the original data.

Parameters

- **ibs** (IBEISController) – wbia controller object

- **gid_list** (*list*) – (default = None)

CommandLine: python -m wbia.other.ibsfuncs –exec-get_missing_gids –db GZ_Master1

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> #ibs = wbia.opendb('GZ_Master1')
>>> gid_list = ibs.get_valid_gids()
>>> bad_gids = ibs.get_missing_gids(gid_list)
>>> print('#bad_gids = %r / %r' % (len(bad_gids), len(gid_list)))
```

wbia.other.ibsfuncs.**get_num_annots_per_name**(*ibs*, *aid_list*)

Returns the number of annots per name (IN THIS LIST)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*int*) – list of annotation ids

CommandLine: python -m wbia.other.ibsfuncs –exec-get_num_annots_per_name python -m wbia.other.ibsfuncs –exec-get_num_annots_per_name –db PZ_Master1

Example

```
>>> # UNSTABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ibs.get_valid_aids(is_known=True)
>>> num_annots_per_name, unique_nids = get_num_annots_per_name(ibs, aid_list)
>>> per_name_hist = ut.dict_hist(num_annots_per_name)
>>> items = per_name_hist.items()
>>> items = sorted(items)[::-1]
>>> key_list = ut.get_list_column(items, 0)
>>> val_list = ut.get_list_column(items, 1)
>>> min_per_name = dict(zip(key_list, np.cumsum(val_list)))
>>> result = ('per_name_hist = %s' % (ut.repr2(per_name_hist),))
>>> print(result)
>>> print('min_per_name = %s' % (ut.repr2(min_per_name),))
per_name_hist = {
    1: 5,
    2: 2,
}
```

wbia.other.ibsfuncs.**get_primary_database_species**(*ibs*, *aid_list=None*, *speed-hack=True*)

Parameters **aid_list** (*list*) – list of annotation ids (default = None)

CommandLine: python -m wbia.other.ibsfuncs –test-get_primary_database_species

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = None
>>> primary_species = get_primary_database_species(ibs, aid_list)
>>> result = primary_species
>>> print('primary_species = %r' % (primary_species,))
>>> print(result)
zebra_plains
```

wbia.other.ibsfuncs.**get_primary_species_viewpoint**(*species*, *plus=0*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **species** –

Returns primary_viewpoint

Return type str

CommandLine: python -m wbia.other.ibsfuncs –exec-get_primary_species_viewpoint

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> species = wbia.const.TEST_SPECIES.ZEB_PLAIN
>>> aid_subset = get_primary_species_viewpoint(species, 0)
>>> result = ('aid_subset = %s' % (str(aid_subset),))
>>> print(result)
aid_subset = left
```

wbia.other.ibsfuncs.**get_quality_filterflags**(*ibs*, *aid_list*, *minqual*, *unknown_ok=True*)

DEPRICATE

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*int*) – list of annotation ids
- **minqual** (*str*) – qualtext
- **unknown_ok** (*bool*) – (default = False)

Returns qual_flags

Return type iter

CommandLine: python -m wbia.other.ibsfuncs –exec-get_quality_filterflags

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ibs.get_valid_aids()[0:20]
>>> minqual = 'junk'
>>> unknown_ok = False
>>> qual_flags = list(get_quality_filterflags(ibs, aid_list, minqual, unknown_ok))
>>> result = ('qual_flags = %s' % (str(qual_flags),))
>>> print(result)
```

wbia.other.ibsfuncs.**get_quality_viewpoint_filterflags**(*ibs*, *aid_list*, *minqual*, *valid_yaws*)

wbia.other.ibsfuncs.**get_special_imgsetids**(*ibs*)

wbia.other.ibsfuncs.**get_species_dbs**(*species_prefix*)

wbia.other.ibsfuncs.**get_two_annots_per_name_and_singletons**(*ibs*, *onlygt=False*)

makes controlled subset of data

DEPRICATE

CONTROLLED TEST DATA

Build data for experiment that tries to rule out as much bad data as possible

Returns a controlled set of annotations that conforms to

- number of annots per name
- uniform species
- viewpoint restrictions
- quality restrictions
- time delta restrictions

CommandLine: python -m wbia.other.ibsfuncs –test-get_two_annots_per_name_and_singletons python -m wbia.other.ibsfuncs –test-get_two_annots_per_name_and_singletons –db GZ_ALL python -m wbia.other.ibsfuncs –test-get_two_annots_per_name_and_singletons –db PZ_Master0 –onlygt

Ignore: sys.argv.extend([’–db’, ‘PZ_MTEST’])

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_Master0')
>>> aid_subset = get_two_annots_per_name_and_singletons(ibs, onlygt=ut.get_
↪argflag('--onlygt'))
>>> wbia.other.dbinfo.get_dbinfo(ibs, aid_list=aid_subset, with_contrib=False)
>>> result = str(aid_subset)
>>> print(result)
```

wbia.other.ibsfuncs.**get_unflat_am_aidpairs**(*ibs*, *aids_list*)

Gets only aid pairs that have some reviewed/matched status

wbia.other.ibsfuncs.**get_unflat_am_rowids**(*ibs*, *aids_list*)

wbia.other.ibsfuncs.**get_unflat_annots_hourdists_list**(*ibs*, *aids_list*)

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> ibs = testdata_ibs('testdb1')
>>> nid_list = get_valid_multiton_nids_custom(ibs)
>>> aids_list_ = ibs.get_name_aids(nid_list)
>>> aids_list = [(aids) for aids in aids_list_]
>>> ibs.get_unflat_annots_hourdists_list(aids_list)
```

wbia.other.ibsfuncs.**get_unflat_annots_kmdists_list**(*ibs*, *aids_list*)

wbia.other.ibsfuncs.**get_unflat_annots_speeds_list**(*ibs*, *aids_list*)
DEPRICATE. SLOWER

wbia.other.ibsfuncs.**get_unflat_annots_speeds_list2**(*ibs*, *aids_list*)
much faster than original version

_ = ibs.get_unflat_annots_speeds_list2(aids_list)

%timeit ibs.get_unflat_annots_speeds_list(aids_list) 3.44 s per loop

%timeit ibs.get_unflat_annots_speeds_list2(aids_list) 665 ms per loop

%timeit ibs.get_unflat_annots_speeds_list(aids_list[0:1]) 12.8 ms %timeit ibs.get_unflat_annots_speeds_list2(aids_list[0:1]) 6.51 ms

assert ibs.get_unflat_annots_speeds_list([]) == ibs.get_unflat_annots_speeds_list2([])

ibs.get_unflat_annots_speeds_list([[]]) ibs.get_unflat_annots_speeds_list2([[]])

wbia.other.ibsfuncs.**get_unflat_annots_timedelta_list**(*ibs*, *aids_list*)

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> ibs = testdata_ibs('NNP_Master3')
>>> nid_list = get_valid_multiton_nids_custom(ibs)
>>> aids_list_ = ibs.get_name_aids(nid_list)
>>> aids_list = [(aids) for aids in aids_list_]
```

wbia.other.ibsfuncs.**get_unflat_case_tags**(*ibs*, *aids_list*)
Gets only aid pairs that have some reviewed/matched status

wbia.other.ibsfuncs.**get_ungrouped_gids**(*ibs*)
CommandLine: python -m wbia.other.ibsfuncs –test-get_ungrouped_gids

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> ibs.delete_all_imagesets()
>>> ibs.compute_occurrences(config={'use_gps': False, 'seconds_thresh': 600})
>>> ibs.update_special_imagesets()
```

(continues on next page)

(continued from previous page)

```
>>> # Now we want to remove some images from a non-special imageset
>>> nonspecial_imgsetids = [i for i in ibs.get_valid_imgsetids() if i not in ibs.
↪get_special_imgsetids()]
>>> print("Nonspecial EIDs %r" % nonspecial_imgsetids)
>>> images_to_remove = ibs.get_imageset_gids(nonspecial_imgsetids[0:1])[0][0:1]
>>> print("Removing %r" % images_to_remove)
>>> ibs.unrelate_images_and_imagesets(images_to_remove,nonspecial_imgsetids[0:1]␣
↪* len(images_to_remove))
>>> ibs.update_special_imagesets()
>>> ungr_imgsetid = ibs.get_imageset_imgsetids_from_text(const.UNGROUPED_IMAGES_
↪IMAGESETTEXT)
>>> print("Ungrouped gids %r" % ibs.get_ungrouped_gids())
>>> print("Ungrouped imgsetid %d contains %r" % (ungr_imgsetid, ibs.get_imageset_
↪gids([ungr_imgsetid])))
>>> ungr_gids = ibs.get_imageset_gids([ungr_imgsetid])[0]
>>> assert(sorted(images_to_remove) == sorted(ungr_gids))
```

wbia.other.ibsfuncs.**get_valid_multiton_nids_custom**(*ibs*)

wbia.other.ibsfuncs.**get_viewpoint_filterflags**(*ibs*, *aid_list*, *valid_yaws*, *unknown_ok=True*, *assume_unique=False*)

Parameters

- **ibs** (*IBEISController*) – wbia controller object
- **aid_list** (*int*) – list of annotation ids
- **valid_yaws** –
- **unknown_ok** (*bool*) – (default = True)

Returns aid_list - list of annotation ids

Return type int

CommandLine: python -m wbia.other.ibsfuncs –exec-get_viewpoint_filterflags python -m wbia.other.ibsfuncs –exec-get_viewpoint_filterflags –db NNP_Master3

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='Spotted_Dolfin_Master')
>>> aid_list = ibs.get_valid_aids()[0:20]
>>> valid_yaws = ['left']
>>> unknown_ok = False
>>> yaw_flags = list(get_viewpoint_filterflags(ibs, aid_list, valid_yaws, unknown_
↪ok))
>>> result = ('yaw_flags = %s' % (str(yaw_flags),))
>>> print(result)
```

wbia.other.ibsfuncs.**get_yaw_viewtexts**(*yaw_list*)

Parameters **yaw_list** (*list of angles*) –

CommandLine: python -m wbia.other.ibsfuncs –test-get_yaw_viewtexts

---

Todo: rhombicubeoctehedron

https://en.wikipedia.org/wiki/Rhombicuboctahedron

up, down, front, left, back, right, front-left, back-left, back-right, front-right, up-front, up-left, up-back, up-right, up-front-left, up-back-left, up-back-right, up-front-right, down-front, down-left, down-back, down-right, down-front-left, down-back-left, down-back-right, down-front-right,

---

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import numpy as np
>>> yaw_list = [0.0, np.pi / 2, np.pi / 4, np.pi, 3.15, -.4, -8, .2, 4, 7, 20,␣
↪None]
>>> text_list = get_yaw_viewtexts(yaw_list)
>>> result = ut.repr2(text_list, nl=False)
>>> print(result)
['right', 'front', 'frontright', 'left', 'left', 'backright', 'back', 'right',
↪'backleft', 'frontright', 'frontright', None]
```

wbia.other.ibsfuncs.**group_annots_by_known_names**(*ibs*, *aid_list*, *checks=True*)

FIXME; rectify this #>>> import wbia # NOQA

CommandLine: python -m wbia.other.ibsfuncs –test-group_annots_by_known_names

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(db='testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
>>> known_aids_list, unknown_aids = group_annots_by_known_names(ibs, aid_list)
>>> result = ut.repr2(sorted(known_aids_list)) + '\n'
>>> result += ut.repr2(unknown_aids)
>>> print(result)
[[2, 3], [5, 6], [7], [8], [10], [12], [13]]
[11, 9, 4, 1]
```

wbia.other.ibsfuncs.**group_annots_by_multi_prop**(*ibs*, *aids*, *getter_list*)

Performs heirachical grouping of annotations based on properties

Parameters

- **ibs** (*IBEISController*) – wbia controller object
- **aids** (*list*) – list of annotation rowids
- **getter_list** (*list*) –

Returns multiprop2_aids

Return type dict

CommandLine: python -m wbia.other.ibsfuncs –exec-group_annots_by_multi_prop –db PZ_Master1 –props=viewpoint_code,name_rowids –keys1 frontleft python -m wbia.other.ibsfuncs –exec-group_annots_by_multi_prop

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aids = ibs.get_valid_aids(is_known=True)
>>> #getter_list = [ibs.get_annot_name_rowids, ibs.get_annot_viewpoints]
>>> props = ut.get_argval('--props', type_=list, default=['viewpoint_code', 'name_
↪rowids'])
>>> getter_list = [getattr(ibs, 'get_annot_' + prop) for prop in props]
>>> print('getter_list = %r' % (getter_list,))
>>> #getter_list = [ibs.get_annot_viewpoints, ibs.get_annot_name_rowids]
>>> multiprop2_aids = group_annots_by_multi_prop(ibs, aids, getter_list)
>>> get_dict_values = lambda x: list(x.values())
>>> # a bit convoluted
>>> keys1 = ut.get_argval('--keys1', type_=list, default=list(multiprop2_aids.
↪keys()))
>>> multiprop2_num_aids = ut.hmap_vals(len, multiprop2_aids)
>>> prop2_num_aids = ut.hmap_vals(get_dict_values, multiprop2_num_aids, max_
↪depth=len(props) - 2)
>>> #prop2_num_aids_stats = ut.hmap_vals(ut.get_stats, prop2_num_aids)
>>> prop2_num_aids_hist = ut.hmap_vals(ut.dict_hist, prop2_num_aids)
>>> prop2_num_aids_cumhist = ut.map_dict_vals(ut.dict_hist_cumsum, prop2_num_aids_
↪hist)
>>> print('prop2_num_aids_hist[%s] = %s' % (keys1,  ut.repr2(ut.dict_subset(prop2_
↪num_aids_hist, keys1))))
>>> print('prop2_num_aids_cumhist[%s] = %s' % (keys1,  ut.repr2(ut.dict_
↪subset(prop2_num_aids_cumhist, keys1))))
```

wbia.other.ibsfuncs.**group_annots_by_name**(*ibs*, *aid_list*, *distinguish_unknowns=True*, *assume_unique=False*)

This function is probably the fastest of its siblings

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*list*) –
- **distinguish_unknowns** (*bool*) –

Returns grouped_aids, unique_nids

Return type tuple

CommandLine: python -m wbia.other.ibsfuncs –test-group_annots_by_name

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> distinguish_unknowns = True
>>> grouped_aids, unique_nids = group_annots_by_name(ibs, aid_list, distinguish_
↪unknowns)
>>> result = str([aids.tolist() for aids in grouped_aids])
```

(continues on next page)

(continued from previous page)

```
>>> result += '\n' + str(unique_nids.tolist())
>>> print(result)
[[11], [9], [4], [1], [2, 3], [5, 6], [7], [8], [10], [12], [13]]
[-11, -9, -4, -1, 1, 2, 3, 4, 5, 6, 7]
```

wbia.other.ibsfuncs.**group_annots_by_name_dict**(*ibs*, *aids*)

wbia.other.ibsfuncs.**group_annots_by_prop**(*ibs*, *aids*, *getter_func*)

wbia.other.ibsfuncs.**group_annots_by_prop_and_name**(*ibs*, *aids*, *getter_func*)

wbia.other.ibsfuncs.**group_prop_edges**(*prop2_nid2_aids*, *prop_basis*, *size=2*, *wrap=True*)
from wbia.other.ibsfuncs import * # NOQA getter_func = ibs.get_annot_viewpoints prop_basis = list(const.VIEWTEXT_TO_YAW_RADIANS.keys()) size = 2 wrap = True

wbia.other.ibsfuncs.**import_folder**(*ibs*, *path*, *recursive=True*, ***kwargs*)

wbia.other.ibsfuncs.**inspect_ggr_qr_codes**(*ibs*, **args*, ***kwargs*)
Inspect QR codes in each imageset.

Parameters

- **ibs** (IBEISController) – wbia controller object
- **imageset_rowid_list** (*list*) – imageset rowid list

CommandLine: python -m wbia.other.ibsfuncs inspect_ggr_qr_codes

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> default_dbdir = join('/', 'data', 'wbia', 'GGR2-IBEIS')
>>> dbdir = ut.get_argval('--dbdir', type_=str, default=default_dbdir)
>>> ibs = wbia.opendb(dbdir=dbdir)
>>> ibs.inspect_ggr_qr_codes()
```

wbia.other.ibsfuncs.**inspect_nonzero_yaws**(*ibs*)
python dev.py –dbdir /raid/work2/PZ_Master –cmd –show

wbia.other.ibsfuncs.**is_aid_unknown**(*ibs*, *aid_list*)
Returns if an annotation has been given a name (even if that name is temporary)

wbia.other.ibsfuncs.**is_nid_unknown**(*ibs*, *nid_list*)

wbia.other.ibsfuncs.**lookup_annot_vecs_subset**(*ibs*, *unflat_aids*, *unflat_fxs*, *annots=None*, *config2_=None*)
unflat_aids = naids_list unflat_fxs = nfxs_list annots = data_annots **config2_ = data_config2_**

unflat_aids = cm.filtnorm_aids[0] unflat_fxs = cm.filtnorm_fxs[0]

wbia.other.ibsfuncs.**make_next_imageset_text**(*ibs*)
Creates what the next imageset name would be but does not add it to the database

Parameters **ibs** (IBEISController) – wbia controller object

CommandLine: python -m wbia.other.ibsfuncs –test-make_next_imageset_text

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> new_imagesettext = make_next_imageset_text(ibs)
>>> result = new_imagesettext
>>> print(result)
New ImageSet 0
```

wbia.other.ibsfuncs.**make_next_name**(*ibs*, *num=None*, *str_format=2*, *species_text=None*, *location_text=None*)

Creates a number of names which are not in the database, but does not add them

Parameters

- **ibs** (IBEISController) – wbia controller object
- **num** (*None*) –
- **str_format** (*int*) – either 1 or 2

Returns next_name

Return type str

CommandLine: python -m wbia.other.ibsfuncs –test-make_next_name

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs1 = wbia.opendb('testdb1')
>>> ibs2 = wbia.opendb('PZ_MTEST')
>>> ibs3 = wbia.opendb('NAUT_test')
>>> ibs1._clean_species()
>>> ibs2._clean_species()
>>> ibs3._clean_species()
>>> num = None
>>> str_format = 2
>>> next_name1 = make_next_name(ibs1, num, str_format)
>>> next_name2 = make_next_name(ibs2, num, str_format)
>>> next_name3 = make_next_name(ibs3, num, str_format)
>>> next_name4 = make_next_name(ibs1, num, str_format, const.TEST_SPECIES.ZEB_
↪GREVY)
>>> name_list = [next_name1, next_name2, next_name3, next_name4]
>>> next_name_list1 = make_next_name(ibs2, 5, str_format)
>>> temp_nids = ibs2.add_names(['WBIA_PZ_0045', 'WBIA_PZ_0048'])
>>> next_name_list2 = make_next_name(ibs2, 5, str_format)
>>> ibs2.delete_names(temp_nids)
>>> next_name_list3 = make_next_name(ibs2, 5, str_format)
>>> # FIXME: nautiluses are not working right
>>> names = (name_list, next_name_list1, next_name_list2, next_name_list3)
>>> result = ut.repr4(names)
>>> print(result)
(
    ['IBEIS_UNKNOWN_0008', 'IBEIS_UNKNOWN_0042', 'IBEIS_UNKNOWN_0004', 'IBEIS_GZ_
↪0008'],
```

(continues on next page)

(continued from previous page)

```
    ['IBEIS_UNKNOWN_0042', 'IBEIS_UNKNOWN_0043', 'IBEIS_UNKNOWN_0044', 'IBEIS_
↪UNKNOWN_0045', 'IBEIS_UNKNOWN_0046'],
    ['IBEIS_UNKNOWN_0044', 'IBEIS_UNKNOWN_0045', 'IBEIS_UNKNOWN_0046', 'IBEIS_
↪UNKNOWN_0047', 'IBEIS_UNKNOWN_0048'],
    ['IBEIS_UNKNOWN_0042', 'IBEIS_UNKNOWN_0043', 'IBEIS_UNKNOWN_0044', 'IBEIS_
↪UNKNOWN_0045', 'IBEIS_UNKNOWN_0046'],
)
```

wbia.other.ibsfuncs.**make_next_nids**(*ibs*, *num=None*, *str_format=2*, *species_text=None*, *location_text=None*)

makes name and adds it to the database returning the newly added name rowid(s)

CAUTION; changes database state
SeeAlso: make_next_name

wbia.other.ibsfuncs.**merge_ggr_staged_annots**(*ibs*, *min_overlap=0.25*, *reviews_required=3*, *liberal_aoi=False*)

Merge the staged annotations into a single set of actual annotations (with AoI)

Parameters **ibs** (IBEISController) – wbia controller object

CommandLine: python -m wbia.other.ibsfuncs merge_ggr_staged_annots

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> from os.path import expanduser
>>> import wbia  # NOQA
>>> # default_dbdir = join('/', 'data', 'wbia', 'GGR2-IBEIS')
>>> default_dbdir = expanduser(join('~', 'data', 'GGR2-IBEIS'))
>>> dbdir = ut.get_argval('--dbdir', type_=str, default=default_dbdir)
>>> ibs = wbia.opendb(dbdir=dbdir)
>>> new_aid_list, broken_gid_list = ibs.merge_ggr_staged_annots()
>>> print('Encountered %d invalid gids: %r' % (len(broken_gid_list), broken_gid_
↪list, ))
```

wbia.other.ibsfuncs.**merge_ggr_staged_annots_cluster**(*ibs*, *user_id_list*, *user_dict*, *aid_list*, *index_list*, *min_overlap=0.25*)

wbia.other.ibsfuncs.**merge_ggr_staged_annots_marriage**(*ibs*, *user_id_list*, *user_dict*, *aid_list*, *index_list*, *min_overlap=0.1*)

wbia.other.ibsfuncs.**merge_names**(*ibs*, *merge_name*, *other_names*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **merge_name** (*str*) –
- **other_names** (*list*) –

CommandLine: python -m wbia.other.ibsfuncs –test-merge_names

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> merge_name = 'zebra'
>>> other_names = ['occl', 'jeff']
>>> result = merge_names(ibs, merge_name, other_names)
>>> print(result)
>>> ibs.print_names_table()
```

wbia.other.ibsfuncs.**new_imagesets_from_images**(*ibs*, *gids_list*)

Parameters **gids_list** (*list*) –

wbia.other.ibsfuncs.**nms_aids**(*ibs*, *aid_list*, ***kwargs*)

wbia.other.ibsfuncs.**nms_boxes**(*ibs*, *indices*, *bboxes*, *thetas*, *confs*, *classes*, *nms_thresh=0.2*, *nms_aware=None*, *verbose=False*, ***kwargs*)

wbia.other.ibsfuncs.**overwrite_ggr_unixtimes_from_gps**(*ibs*, *gmt_offset=3.0*, **args*, ***kwargs*)

Sync image time offsets using QR codes sync data

Parameters **ibs** (IBEISController) – wbia controller object

CommandLine: python -m wbia.other.ibsfuncs overwrite_ggr_unixtimes_from_gps

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> default_dbdir = join('/', 'data', 'wbia', 'GGR2-IBEIS')
>>> dbdir = ut.get_argval('--dbdir', type_=str, default=default_dbdir)
>>> ibs = wbia.opendb(dbdir=dbdir)
>>> ibs.overwrite_ggr_unixtimes_from_gps()
```

wbia.other.ibsfuncs.**overwrite_unixtimes_from_gps**(*ibs*, *gid_list*, *gmt_offset=3.0*)

Sync image time offsets using QR codes sync data

Parameters **ibs** (IBEISController) – wbia controller object

CommandLine: python -m wbia.other.ibsfuncs overwrite_unixtimes_from_gps

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> default_dbdir = join('/', 'data', 'wbia', 'GGR2-IBEIS')
>>> dbdir = ut.get_argval('--dbdir', type_=str, default=default_dbdir)
>>> ibs = wbia.opendb(dbdir=dbdir)
>>> ibs.overwrite_unixtimes_from_gps()
```

wbia.other.ibsfuncs.**overwrite_unixtimes_from_gps_worker**(*path*)

wbia.other.ibsfuncs.**parse_annot_config_stats_filter_kws**(*ibs*)

wbia.other.ibsfuncs.**parse_annot_stats_filter_kws**(*ibs*)

wbia.other.ibsfuncs.**parse_ggr_name**(*ibs*, *imageset_text*, *verbose=False*, *allow_short=False*, *require_short=False*)

wbia.other.ibsfuncs.**partition_annots_into_corresponding_groups**(*ibs*, *aid_list1*, *aid_list2*)

Used for grouping one-vs-one training pairs and corerspondence filtering

Parameters

- **ibs** (wbia.control.IBEISControl.IBEISController) – wbia controller object
- **aid_list1** (*int*) – list of annotation ids
- **aid_list2** (*int*) – list of annotation ids

Returns

4 lists of lists. In the first two each list is a list of aids grouped by names and the names correspond with each other. In the last two are the annots that did not correspond with anything in the other list.

Return type tuple

CommandLine: python -m wbia.other.ibsfuncs –exec-partition_annots_into_corresponding_groups

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_MTEST')
>>> grouped_aids = list(map(list, ibs.group_annots_by_name(ibs.get_valid_
↪aids())[0]))
>>> grouped_aids = [aids for aids in grouped_aids if len(aids) > 3]
>>> # Get some overlapping groups
>>> import copy
>>> aids_group1 = copy.deepcopy((ut.get_list_column_slice(grouped_aids[0:5],␣
↪slice(0, 2))))
>>> aids_group2 = copy.deepcopy((ut.get_list_column_slice(grouped_aids[2:7],␣
↪slice(2, None))))
>>> # Ensure there is a singleton in each
>>> ut.delete_items_by_index(aids_group1[0], [0])
>>> ut.delete_items_by_index(aids_group2[-1], [0])
>>> aid_list1 = ut.flatten(aids_group1)
>>> aid_list2 = ut.flatten(aids_group2)
>>> #aid_list1 = [1, 2, 8, 9, 60]
>>> #aid_list2 = [3, 7, 20]
>>> groups = partition_annots_into_corresponding_groups(ibs, aid_list1, aid_list2)
>>> result = ut.repr2(groups)
>>> print(result)
[[10, 11], [17, 18], [22, 23]]
[[12, 13, 14, 15], [19, 20, 21], [24, 25, 26]]
[[2], [5, 6]]
[[29, 30, 31, 32], [49]]
```

wbia.other.ibsfuncs.**partition_annots_into_singleton_multiton**(*ibs*, *aid_list*)

aid_list = **aid_list_**

wbia.other.ibsfuncs.**partition_ordered_list_equal_sum**(*a*, *k*)

Partition a sorted list a into k partitions

Reference: https://stackoverflow.com/a/35518205 https://gist.github.com/laowantong/ee675108eee64640e5f94f00d8edbcb4

CommandLine: python -m wbia.other.ibsfuncs partition_ordered_list_equal_sum

Example

```
>>> # DISABLE_DOCTEST
>>> import random
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> a = [random.randint(0,20) for x in range(50)]
>>> k = 10
>>> print('Partitioning {0} into {1} partitions'.format(a, k))
>>> b = partition_ordered_list_equal_sum(a, k)
>>> print('The best partitioning is {0}\n    With heights {1}\n'.format(b,␣
↪list(map(sum, b))))
```

wbia.other.ibsfuncs.**partition_ordered_list_equal_sum_recursive**(*vals*, *ids*, *k*, *level*)

wbia.other.ibsfuncs.**postinject_func**(*ibs*)

Parameters **ibs** (IBEISController) –

CommandLine: python -m wbia.other.ibsfuncs –test-postinject_func

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> ibs.delete_empty_nids()  # a test run before this forgot to do this
>>> aids_list = ibs.get_name_aids(ibs.get_valid_nids())
>>> # indirectly test postinject_func
>>> thetas_list = ibs.get_unflat_annot_thetas(aids_list)
>>> result = str(thetas_list)
>>> print(result)
[[0.0, 0.0], [0.0, 0.0], [0.0], [0.0], [0.0], [0.0], [0.0]]
```

wbia.other.ibsfuncs.**prepare_annotgroup_review**(*ibs*, *aid_list*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*int*) – list of annotation ids

Returns (src_ag_rowid, dst_ag_rowid) - source and dest annot groups

Return type tuple

CommandLine: python -m wbia.other.ibsfuncs –test-prepare_annotgroup_review

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> result = prepare_annotgroup_review(ibs, aid_list)
>>> print(result)
```

wbia.other.ibsfuncs.**princeton_cameratrap_ocr_bottom_bar**(*ibs*, *gid_list=None*)

wbia.other.ibsfuncs.**princeton_cameratrap_ocr_bottom_bar_accuracy**(*ibs*, *offset=61200*, ***kwargs*)

wbia.other.ibsfuncs.**princeton_cameratrap_ocr_bottom_bar_csv**(*ibs*, *prefix='/data/raw/unprocessed/horses/'*, *threshold=0.39*)

wbia.other.ibsfuncs.**princeton_cameratrap_ocr_bottom_bar_parser**(*raw*)

wbia.other.ibsfuncs.**princeton_process_encounters**(*ibs*, *input_file_path*, *assert_valid=True*, ***kwargs*)

wbia.other.ibsfuncs.**princeton_process_individuals**(*ibs*, *input_file_path*, ***kwargs*)

wbia.other.ibsfuncs.**print_alr_table**(*ibs*, ***kwargs*)
Dumps alr table to stdout

wbia.other.ibsfuncs.**print_annot_stats**(*ibs*, *aids*, *prefix=''*, *label=''*, ***kwargs*)

wbia.other.ibsfuncs.**print_annotation_table**(*ibs*, *verbosity=1*, *exclude_columns=[]*, *include_columns=[]*)
Dumps annotation table to stdout

Parameters

- **ibs** (IBEISController) –
- **verbosity** (*int*) –

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> verbosity = 1
>>> print_annotation_table(ibs, verbosity)
```

wbia.other.ibsfuncs.**print_annotconfig_stats**(*ibs*, *qaids*, *daids*, ***kwargs*)
SeeAlso: ibs.get_annotconfig_stats

wbia.other.ibsfuncs.**print_annotmatch_table**(*ibs*)
Dumps annotation match table to stdout

Parameters **ibs** (IBEISController) – wbia controller object

CommandLine: python -m wbia.other.ibsfuncs –exec-print_annotmatch_table python -m wbia.other.ibsfuncs –exec-print_annotmatch_table –db PZ_Master1

Example

```
>>> # SCRIPT
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> result = print_annotmatch_table(ibs)
>>> print(result)
```

wbia.other.ibsfuncs.**print_chip_table**(*ibs*)
Dumps chip table to stdout

wbia.other.ibsfuncs.**print_config_table**(*ibs*, ***kwargs*)
Dumps config table to stdout

wbia.other.ibsfuncs.**print_contributor_table**(*ibs*, *verbosity=1*, *exclude_columns=[]*)
Dumps annotation table to stdout

Parameters

- **ibs** (IBEISController) –
- **verbosity** (*int*) –

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> verbosity = 1
>>> print_contributor_table(ibs, verbosity)
```

wbia.other.ibsfuncs.**print_dbinfo**(*ibs*, ***kwargs*)

wbia.other.ibsfuncs.**print_egpairs_table**(*ibs*, ***kwargs*)
Dumps egpairs table to stdout

wbia.other.ibsfuncs.**print_feat_table**(*ibs*)
Dumps chip table to stdout

wbia.other.ibsfuncs.**print_image_table**(*ibs*, ***kwargs*)
Dumps chip table to stdout

wbia.other.ibsfuncs.**print_imageset_table**(*ibs*, ***kwargs*)
Dumps imageset table to stdout
Kwargs: exclude_columns (list):

wbia.other.ibsfuncs.**print_infostr**(*ibs*, ***kwargs*)

wbia.other.ibsfuncs.**print_lblannot_table**(*ibs*, ***kwargs*)
Dumps lblannot table to stdout

wbia.other.ibsfuncs.**print_name_table**(*ibs*, ***kwargs*)
Dumps name table to stdout

wbia.other.ibsfuncs.**print_partition_sizes_recursive**(*vals*, *k*, *level=0*, *index=0*)

wbia.other.ibsfuncs.**print_party_table**(*ibs*, ***kwargs*)
Dumps chip table to stdout

wbia.other.ibsfuncs.**print_species_table**(*ibs*, ***kwargs*)
Dumps species table to stdout

wbia.other.ibsfuncs.**print_tables**(*ibs*, *exclude_columns=None*, *exclude_tables=None*)

wbia.other.ibsfuncs.**purge_ggr_unixtime_out_of_bounds**(*ibs*, **args*, ***kwargs*)

wbia.other.ibsfuncs.**query_ggr_gids_between_dates**(*ibs*, *gid_list=None*, *date1=(2018, 1, 27)*, *date2=(2018, 1, 29)*, *local_offset=-8.0*, *gmt_offset=3.0*)

wbia.other.ibsfuncs.**remove_aids_of_viewpoint**(*ibs*, *aid_list*, *invalid_yaws*)
Removes aids that do not have a valid yaw

TODO; rename to valid_viewpoint because this func uses category labels

wbia.other.ibsfuncs.**remove_groundtrue_aids**(*ibs*, *aid_list*, *ref_aid_list*)
removes any aids that are known to match

wbia.other.ibsfuncs.**report_sightings**(*ibs*, *complete=True*, *include_images=False*, *kaia=False*, ***kwargs*)

wbia.other.ibsfuncs.**report_sightings_str**(*ibs*, ***kwargs*)

wbia.other.ibsfuncs.**run_integrity_checks**(*ibs*)
Function to run all database consistency checks

wbia.other.ibsfuncs.**search_annot_notes**(*ibs*, *pattern*, *aid_list=None*)

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('PZ_Master0')
>>> pattern = ['gash', 'injury', 'scar', 'wound']
>>> valid_aid_list = ibs.search_annot_notes(pattern)
>>> print(valid_aid_list)
>>> print(ibs.get_annot_notes(valid_aid_list))
```

wbia.other.ibsfuncs.**search_ggr_qr_codes**(*ibs*, *imageset_rowid_list=None*, *timeout=None*, ***kwargs*)
Search for QR codes in each imageset.

Parameters

- **ibs** (IBEISController) – wbia controller object
- **imageset_rowid_list** (*list*) – imageset rowid list

CommandLine: python -m wbia.other.ibsfuncs search_ggr_qr_codes
Reference: https://www.learnopencv.com/barcode-and-qr-code-scanner-using-zbar-and-opencv/

macOS: brew install zbar

or

curl -O https://ayera.dl.sourceforge.net/project/zbar/zbar/0.10/zbar-0.10.tar.bz2 tar -xvjf zbar-0.10.tar.bz2 cd zbar-0.10/ CPPFLAGS="-I/opt/local/include" LDFLAGS="-L/opt/local/lib" ./configure –disable-video –without-qt –without-python –without-gtk –with-libiconv-prefix=/opt/local –with-jpeg=yes –prefix=$VIRTUAL_ENV make make install sudo ln $VIRTUAL_ENV/lib/libzbar.dylib /opt/local/lib/libzbar.dylib sudo ln $VIRTUAL_ENV/include/zbar.h /opt/local/include/zbar.h

Ubuntu: sudo apt-get install libzbar-dev libzbar0

pip install pyzbar

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> default_dbdir = join('/', 'data', 'wbia', 'GGR2-IBEIS')
>>> dbdir = ut.get_argval('--dbdir', type_=str, default=default_dbdir)
>>> ibs = wbia.opendb(dbdir=dbdir)
>>> ibs.search_ggr_qr_codes()
```

wbia.other.ibsfuncs.**search_ggr_qr_codes_worker**(*imageset_rowid*, *imageset_text*, *values*, *gid_list*, *filepath_list*, *note_list*, *timeout*)

wbia.other.ibsfuncs.**set_annot_names_to_different_new_names**(*ibs*, *aid_list*, ***kwargs*)

wbia.other.ibsfuncs.**set_annot_names_to_next_name**(*ibs*, *aid_list*)

wbia.other.ibsfuncs.**set_annot_names_to_same_new_name**(*ibs*, *aid_list*)

wbia.other.ibsfuncs.**set_exemplars_from_quality_and_viewpoint**(*ibs*, *aid_list=None*, *exemplars_per_view=None*, *imgsetid=None*, *dry_run=False*, *verbose=True*, *prog_hook=None*)

Automatic exemplar selection algorithm based on viewpoint and quality

References

implement maximum diversity approximation instead http://www.csbio.unc.edu/mcmillan/pubs/ICDM07_Pan.pdf

CommandLine: python -m wbia.other.ibsfuncs –test-set_exemplars_from_quality_and_viewpoint python -m wbia.other.ibsfuncs –test-set_exemplars_from_quality_and_viewpoint:1

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> #ibs = wbia.opendb('PZ_MUGU_19')
>>> ibs = wbia.opendb('PZ_MTEST')
>>> dry_run = True
>>> verbose = False
>>> old_sum = sum(ibs.get_annot_exemplar_flags(ibs.get_valid_aids()))
>>> new_flag_list = ibs.set_exemplars_from_quality_and_viewpoint(dry_run=dry_run)
>>> new_sum = sum(new_flag_list)
>>> print('old_sum = %r' % (old_sum,))
>>> print('new_sum = %r' % (new_sum,))
>>> zero_flag_list = ibs.set_exemplars_from_quality_and_viewpoint(exemplars_per_
↪view=0, dry_run=dry_run)
```

(continues on next page)

(continued from previous page)

```
>>> assert sum(zero_flag_list) == 0
>>> result = new_sum
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> dry_run = True
>>> verbose = False
>>> old_sum = sum(ibs.get_annot_exemplar_flags(ibs.get_valid_aids()))
>>> new_flag_list = ibs.set_exemplars_from_quality_and_viewpoint(dry_run=dry_run)
>>> # 2 of the 11 annots are unknown and should not be exemplars
>>> ut.assert_eq(sum(new_flag_list), 9)
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb2')
>>> dry_run = True
>>> verbose = False
>>> imgsetid = None
>>> aid_list = ibs.get_valid_aids(imgsetid=imgsetid)
>>> new_flag_list = ibs.set_exemplars_from_quality_and_viewpoint(aid_list, dry_
↪run=dry_run)
>>> old_flag_list = ibs.get_annot_exemplar_flags(aid_list)
>>> new_exemplar_aids = ut.compress(aid_list, new_flag_list)
>>> new_exemplar_qualtexts = ibs.get_annot_quality_texts(new_exemplar_aids)
>>> assert 'junk' not in new_exemplar_qualtexts, 'should not have junk exemplars'
>>> assert 'poor' not in new_exemplar_qualtexts, 'should not have poor exemplars'
>>> #assert len(new_aid_list) == len(new_flag_list)
>>> # 2 of the 11 annots are unknown and should not be exemplars
>>> #ut.assert_eq(len(new_aid_list), 9)
```

wbia.other.ibsfuncs.**sync_ggr_with_qr_codes**(*ibs*, *local_offset=-8.0*, *gmt_offset=3.0*, **args*, ***kwargs*)

Sync image time offsets using QR codes sync data

Parameters **ibs** (IBEISController) – wbia controller object

CommandLine: python -m wbia.other.ibsfuncs sync_ggr_with_qr_codes

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> default_dbdir = join('/', 'data', 'wbia', 'GGR2-IBEIS')
>>> dbdir = ut.get_argval('--dbdir', type_=str, default=default_dbdir)
```

(continues on next page)

(continued from previous page)

```
>>> ibs = wbia.opendb(dbdir=dbdir)
>>> ibs.sync_ggr_with_qr_codes()
```

wbia.other.ibsfuncs.**testdata_ibs**(*defaultdb='testdb1'*)

wbia.other.ibsfuncs.**unflat_map**(*method*, *unflat_rowids*, ***kwargs*)

Uses an wbia lookup function with a non-flat rowid list. In essence this is equivilent to map(method, unflat_rowids). The utility of this function is that it only calls method once. This is more efficient for calls that can take a list of inputs

Parameters

- **method** (*method*) – wbia controller method
- **unflat_rowids** (*list*) – list of rowid lists

Returns unflat_vals

Return type list of values

CommandLine: python -m wbia.other.ibsfuncs –test-unflat_map

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb1')
>>> method = ibs.get_annot_name_rowids
>>> unflat_rowids = ibs.get_name_aids(ibs.get_valid_nids())
>>> unflat_vals = unflat_map(method, unflat_rowids)
>>> result = str(unflat_vals)
>>> print(result)
[[1, 1], [2, 2], [3], [4], [5], [6], [7]]
```

wbia.other.ibsfuncs.**update_all_image_special_imageset**(*ibs*)

wbia.other.ibsfuncs.**update_exemplar_special_imageset**(*ibs*)

wbia.other.ibsfuncs.**update_reviewed_unreviewed_image_special_imageset**(*ibs*, *reviewed=True*, *unreviewed=True*)

Creates imageset of images that have not been reviewed and that have been reviewed (wrt detection)

wbia.other.ibsfuncs.**update_special_imagesets**(*ibs*, *use_more_special_imagesets=False*)

wbia.other.ibsfuncs.**update_species_imagesets**(*ibs*)

wbia.other.ibsfuncs.**update_ungrouped_special_imageset**(*ibs*)

Parameters **ibs** (IBEISController) – wbia controller object

CommandLine: python -m wbia.other.ibsfuncs –test-update_ungrouped_special_imageset

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia  # NOQA
>>> ibs = wbia.opendb('testdb9')
>>> result = update_ungrouped_special_imageset(ibs)
>>> print(result)
```

wbia.other.ibsfuncs.**use_images_as_annotations**(*ibs*, *gid_list*, *name_list=None*, *nid_list=None*, *notes_list=None*, *adjust_percent=0.0*, *tags_list=None*)
Adds an annotation the size of the entire image to each image. adjust_percent - shrinks the ANNOTATION by percentage on each side

wbia.other.ibsfuncs.**vd**(*ibs*)

wbia.other.ibsfuncs.**view_dbdir**(*ibs*)

wbia.other.ibsfuncs.**viewpoint_diff**(*ori1*, *ori2*)
convert distance in radians to distance in viewpoint category

1.10.10 Module contents

wbia.other.**IMPORT_TUPLES = [('dbinfo', None), ('duct_tape', None), ('detectfuncs', None),**
cd /home/joncrall/code/wbia/wbia/other makeinit.py –modname=wbia.other
Type Regen Command

wbia.other.**reassign_submodule_attributes**(*verbose=True*)
why reloading all the modules doesnt do this I don't know

wbia.other.**reload_subs**(*verbose=True*)
Reloads wbia.other and submodules

wbia.other.**rrrr**(*verbose=True*)
Reloads wbia.other and submodules

1.11 wbia.plottool package

1.11.1 Subpackages

1.11.1.1 wbia.plottool.tests package

1.11.1.1.1 Submodules

1.11.1.1.2 wbia.plottool.tests.test_helpers module

wbia.plottool.tests.test_helpers.**dummy_bbox**(*img*, *shiftxy=(0.0, 0.0)*, *scale=0.25*)
Default to rectangle that has a quarter-width/height border.

wbia.plottool.tests.test_helpers.**imread_many**(*imgpaths*)

1.11.1.1.3 wbia.plottool.tests.test_interact_multi_image module

1.11.1.1.4 wbia.plottool.tests.test_viz_image2 module

1.11.1.1.5 wbia.plottool.tests.test_viz_images module

1.11.1.1.6 Module contents

1.11.2 Submodules

1.11.3 wbia.plottool.__MPL_INIT__ module

Notes

To use various backends certian packages are required

PyQt ...

Tk pip install sudo apt-get install tk sudo apt-get install tk-dev

Wx pip install wxPython

GTK pip install PyGTK pip install pygobject pip install pygobject

Cairo pip install pycairo pip install py2cairo pip install cairocffi sudo apt-get install libcairo2-dev

CommandLine: python -m wbia.plottool.draw_func2 –exec-imshow –show –mplbe=GTKAgg python -m wbia.plottool.draw_func2 –exec-imshow –show –mplbe=TkAgg python -m wbia.plottool.draw_func2 –exec-imshow –show –mplbe=WxAgg python -m wbia.plottool.draw_func2 –exec-imshow –show –mplbe=WebAgg python -m wbia.plottool.draw_func2 –exec-imshow –show –mplbe=gdk python -m wbia.plottool.draw_func2 –exec-imshow –show –mplbe=cairo

wbia.plottool.__MPL_INIT__.**get_pyqt**()

wbia.plottool.__MPL_INIT__.**get_target_backend**()

wbia.plottool.__MPL_INIT__.**init_matplotlib**(*verbose=False*)

wbia.plottool.__MPL_INIT__.**print_all_backends**()

wbia.plottool.__MPL_INIT__.**profile**(*func*)

1.11.4 wbia.plottool.__main__ module

wbia.plottool.__main__.**plottool_main**()

1.11.5 wbia.plottool._cv2_impaint module

wbia.plottool._cv2_impaint.**cached_impaint**(*bgr_img*, *cached_mask_fpath=None*, *label_colors=None*, *init_mask=None*, *aug=False*, *refine=False*)

wbia.plottool._cv2_impaint.**demo**()
CommandLine: python -m wbia.plottool.interact_impaint –test-demo

References

http://docs.opencv.org/trunk/doc/py_tutorials/py_gui/py_mouse_handling/py_mouse_handling.html

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.interact_impaint import *  # NOQA
>>> # build test data
>>> # execute function
>>> result = demo()
>>> # verify results
>>> print(result)
```

wbia.plottool._cv2_impaint.**impaint_mask**(*img*, *label_colors=None*, *init_mask=None*, *init_label=None*)

CommandLine: python -m wbia.plottool.interact_impaint –test-impaint_mask

References

http://docs.opencv.org/trunk/doc/py_tutorials/py_gui/py_mouse_handling/py_mouse_handling.html

TODO: Slider for transparency TODO: Label selector

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.interact_impaint import *  # NOQA
>>> import utool as ut
>>> import vtool as vt
>>> img_fpath = ut.grab_test_imgpath('lena.png')
>>> img = vt.imread(img_fpath)
>>> label_colors = [255, 200, 100, 0]
>>> result = impaint_mask(img, label_colors)
>>> # verify results
>>> print(result)
```

1.11.6 wbia.plottool._oldimpaint module

1.11.7 wbia.plottool.abstract_interaction module

Known Interactions that use AbstractInteraction: pt.MatchInteraction2 pt.MultiImageInteraction wbia.NameInteraction

class wbia.plottool.abstract_interaction.**AbstractInteraction**(***kwargs*)

Bases: object

An interaction is meant to take up an entire figure

overwrite either self.plot(fnum, pnum) or self.staic_plot(fnum, pnum) or show_page

LEFT_BUTTON = 1

MIDDLE_BUTTON = 2

MOUSE_BUTTONS = {1: 'left', 2: 'middle', 3: 'right'}

RIGHT_BUTTON = 3

append_button(*text*, *divider=None*, *rect=None*, *callback=None*, *size='9%'*, *location='bottom'*, *ax=None*, ***kwargs*)
Adds a button to the current page

bring_to_front()

clean_scope()
Removes any widgets saved in the interaction scope

clear_parent_axes(*ax*)
for clearing axes that we appended anything to

close()

connect_callbacks()

draw()

enable_pan(*ax*)

enable_pan_and_zoom(*ax*)

enable_zoom(*ax*)

on_click(*event*)

on_click_inside(*event*, *ax*)

on_click_outside(*event*)

on_click_release(*event*)

on_close(*event=None*)

on_drag(*event=None*)

on_drag_inside(*event=None*)

on_drag_start(*event=None*)

on_drag_stop(*event=None*)

on_draw(*event=None*)

on_key_press(*event*)

on_motion(*event*)

on_scroll(*event*)

print_status()

reset_mouse_state()

show()

show_page(**args*)
Hack: this function should probably not be defined, but it is for convinience of a developer. Override this or create static plot function (preferably override)

show_popup_menu(*options*, *event*)
context menu

start()

update()

class wbia.plottool.abstract_interaction.**AbstractPagedInteraction**(*nPages=None*, *draw_hud=True*, ***kwargs*)

Bases: *wbia.plottool.abstract_interaction.AbstractInteraction*

make_hud()
Creates heads up display

next_page(*event*)

on_key_press(*event*)

prepare_page(*fulldraw=True*)

prev_page(*event*)

wbia.plottool.abstract_interaction.**matches_hotkey**(*key*, *hotkeys*)

wbia.plottool.abstract_interaction.**pretty_hotkey_map**(*hotkeys*)

wbia.plottool.abstract_interaction.**register_interaction**(*self*)

wbia.plottool.abstract_interaction.**unregister_interaction**(*self*)

1.11.8 wbia.plottool.color_funcs module

wbia.plottool.color_funcs.**add_alpha**(*colors*)

wbia.plottool.color_funcs.**adjust_hsv_of_rgb**(*rgb*, *hue_adjust=0.0*, *sat_adjust=0.0*, *val_adjust=0.0*)

works on a single rgb tuple

Parameters

- **rgb** (*tuple*) –
- **hue_adjust** (*float*) –
- **sat_adjust** (*float*) –
- **val_adjust** (*float*) –

Returns new_rgb

Return type

?

CommandLine: python -m wbia.plottool.color_funcs –test-adjust_hsv_of_rgb –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.color_funcs import *  # NOQA
>>> import wbia.plottool as pt
>>> # build test data
>>> rgb_list = [pt.DEEP_PINK[0:3], pt.DARK_YELLOW[0:3], pt.DARK_GREEN[0:3]]
>>> hue_adjust = -0.1
>>> sat_adjust = +0.5
>>> val_adjust = -0.1
>>> # execute function
>>> new_rgb_list = [adjust_hsv_of_rgb(rgb, hue_adjust, sat_adjust, val_adjust)␣
↪for rgb in rgb_list]
```

(continues on next page)

(continued from previous page)

```
>>> import wbia.plottool as pt
>>> if pt.show_was_requested():
>>>     color_list = rgb_list + new_rgb_list
>>>     testshow_colors(color_list)
>>> # verify results
>>> result = str(new_rgb)
>>> print(result)
```

Ignore: print(np.array([-.1, 0.0, .1, .5, .9, 1.0, 1.1])) print(np.array([-.1, 0.0, .1, .5, .9, 1.0, 1.1]) % 1.0) print(divmod(np.array([-.1, 0.0, .1, .5, .9, 1.0, 1.1]), 1.0)) print(1 + np.array([-.1, 0.0, .1, .5, .9, 1.0, 1.1]) % 1.0)

wbia.plottool.color_funcs.**adjust_hsv_of_rgb255**(*rgb255*, **args*, ***kwargs*)

CommandLine: python -m wbia.plottool.color_funcs –test-adjust_hsv_of_rgb255 –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.color_funcs import *  # NOQA
>>> import wbia.plottool as pt
>>> # build test data
>>> rgb = (220, 220, 255)
>>> hue_adjust =  0.0
>>> sat_adjust = -0.05
>>> val_adjust =  0.0
>>> # execute function
>>> new_rgb = adjust_hsv_of_rgb255(rgb, hue_adjust, sat_adjust, val_adjust)
>>> # verify results
>>> result = str(new_rgb)
>>> print(result)
>>> import wbia.plottool as pt
>>> if pt.show_was_requested():
>>>     color_list = [to_base01(rgb), to_base01(new_rgb)]
>>>     testshow_colors(color_list)
```

wbia.plottool.color_funcs.**assert_base01**(*channels*)

wbia.plottool.color_funcs.**assert_base255**(*channels*)

wbia.plottool.color_funcs.**brighten**(**args*, ***kwargs*)

wbia.plottool.color_funcs.**brighten_rgb**(*rgb*, *amount*)

wbia.plottool.color_funcs.**convert_255_to_hex**(*color255*)

```
>>> color255 = [255, 51, 0]
```

target_rgb01 = pt.FALSE_RED[0:3] target_rgb = np.array([[target_rgb01]]).astype(np.float32) / 25 target_lab = vt.convert_colorspace(target_rgb, ‘lab’, ‘rgb’)

# Find closest CSS color in LAB space dist_lab = {} dist_rgb = {} css_colors = ub.map_vals(convert_hex_to_255, mcolors.CSS4_COLORS) for k, c in css_colors.items():
    rgb = np.array([[c]]).astype(np.float32) / 255 lab = vt.convert_colorspace(rgb, ‘lab’, ‘rgb’) dist_lab[k] = np.sqrt(((target_lab - lab) ** 2).sum()) dist_rgb[k] = np.sqrt(((target_rgb - rgb) ** 2).sum())
best_keys = ub.argsort(dist_lab) ub.odict(zip(best_keys, ub.take(dist_lab, best_keys)))

wbia.plottool.color_funcs.**convert_hex_to_255**(*hex_color*)
hex_color = '#6A5AFFAF'

wbia.plottool.color_funcs.**darken_rgb**(*rgb*, *amount*)

wbia.plottool.color_funcs.**desaturate_rgb**(*rgb*, *amount*)
CommandLine: python -m wbia.plottool.color_funcs –test-desaturate_rgb –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.color_funcs import *  # NOQA
>>> rgb = (255.0 / 255.0, 100 / 255.0, 0 / 255.0)
>>> amount = .5
>>> new_rgb = desaturate_rgb(rgb, amount)
>>> # xdoctest: +REQUIRES(--show)
>>> color_list = [rgb, new_rgb, desaturate_rgb(rgb, .7)]
>>> testshow_colors(color_list)
>>> # verify results
>>> result = ut.repr2(new_rgb)
>>> print(result)
(1.0, 0.696078431372549, 0.5)
```

(1.0, 0.41599384851980004, 0.039215686274509776)

wbia.plottool.color_funcs.**distinct_colors**(*N*, *brightness=0.878*, *randomize=True*, *hue_range=(0.0, 1.0)*, *cmap_seed=None*)

Parameters

- **N** (*int*) –
- **brightness** (*float*) –

Returns RGB_tuples

Return type list

CommandLine: python -m wbia.plottool.color_funcs –test-distinct_colors –N 2 –show –hue-range=0.05,.95 python -m wbia.plottool.color_funcs –test-distinct_colors –N 3 –show –hue-range=0.05,.95 python -m wbia.plottool.color_funcs –test-distinct_colors –N 4 –show –hue-range=0.05,.95 python -m wbia.plottool.color_funcs –test-distinct_colors –N 3 –show –no-randomize python -m wbia.plottool.color_funcs –test-distinct_colors –N 4 –show –no-randomize python -m wbia.plottool.color_funcs –test-distinct_colors –N 6 –show –no-randomize python -m wbia.plottool.color_funcs –test-distinct_colors –N 20 –show

References

http://blog.jianhuashao.com/2011/09/generate-n-distinct-colors.html

CommandLine: python -m wbia.plottool.color_funcs –exec-distinct_colors –show python -m wbia.plottool.color_funcs –exec-distinct_colors –show –no-randomize –N 50 python -m wbia.plottool.color_funcs –exec-distinct_colors –show –cmap_seed=foobar

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.color_funcs import *  # NOQA
>>> # build test data
>>> N = ut.get_argval('--N', int, 2)
>>> randomize = not ut.get_argflag('--no-randomize')
>>> brightness = 0.878
>>> # execute function
>>> cmap_seed = ut.get_argval('--cmap_seed', str, default=None)
>>> hue_range = ut.get_argval('--hue-range', list, default=(0.00, 1.0))
>>> RGB_tuples = distinct_colors(N, brightness, randomize, hue_range, cmap_
↪seed=cmap_seed)
>>> # verify results
>>> assert len(RGB_tuples) == N
>>> result = str(RGB_tuples)
>>> print(result)
>>> ut.quit_if_noshow()
>>> color_list = RGB_tuples
>>> testshow_colors(color_list)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.color_funcs.**ensure_base01**(*color*)

always returns a base 01 color

Note, some colors cannot be determined to be either 255 or 01 if they are in float format.

Parameters **color** –

Returns color01

Return type

?

CommandLine: python -m wbia.plottool.color_funcs ensure_base01

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.color_funcs import *  # NOQA
>>> ensure_base01('g')
>>> ensure_base01('orangered')
>>> ensure_base01('#AAAAAA')
>>> ensure_base01([0, 0, 0])
>>> ensure_base01([1, 1, 0, 0])
>>> ensure_base01([1., 1., 0., 0.])
>>> ensure_base01([.7, .2, 0., 0.])
```

wbia.plottool.color_funcs.**ensure_base255**(*color*)

always returns a base 255 color

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.color_funcs import *  # NOQA
>>> ensure_base255('g')
>>> ensure_base255('orangered')
```

(continues on next page)

(continued from previous page)

```
>>> ensure_base255('#AAAAAA')
>>> ensure_base255([0, 0, 0])
>>> ensure_base255([1, 1, 0, 0])
>>> ensure_base255([.9, 1., 0., 0.])
>>> ensure_base255([1., 1., 0., 0.])  # FIXME
>>> ensure_base255([.7, .2, 0., 0.])
```

wbia.plottool.color_funcs.**is_base01**(*channels*)
check if a color is in base 01

wbia.plottool.color_funcs.**is_base255**(*channels*)
check if a color is in base 01

wbia.plottool.color_funcs.**lighten_rgb**(*rgb*, *amount*)
CommandLine: python -m wbia.plottool.color_funcs –test-lighten_rgb –show python -m wbia.plottool.color_funcs –test-lighten_rgb

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.color_funcs import *  # NOQA
>>> # build test data
>>> rgb = np.array((255.0 / 255.0, 100 / 255.0, 0 / 255.0))
>>> amount = .1
>>> # execute function
>>> new_rgb = lighten_rgb(rgb, amount)
>>> import wbia.plottool as pt
>>> if pt.show_was_requested():
>>>     color_list = [rgb, new_rgb, lighten_rgb(rgb, .5)]
>>>     testshow_colors(color_list)
>>> # verify results
>>> result = ut.repr2(new_rgb, with_dtype=False)
>>> print(result)
```

wbia.plottool.color_funcs.**show_all_colormaps**()
Displays at a 90 degree angle. Weird

FIXME: Remove call to pylab

References

http://wiki.scipy.org/Cookbook/Matplotlib/Show_colormaps http://matplotlib.org/examples/color/colormaps_reference.html

Notes

cmaps = [(‘Perceptually Uniform Sequential’,

[‘viridis’, ‘inferno’, ‘plasma’, ‘magma’]),

(‘Sequential’, [‘Blues’, ‘BuGn’, ‘BuPu’, ‘GnBu’, ‘Greens’, ‘Greys’, ‘Oranges’, ‘OrRd’, ‘PuBu’, ‘PuBuGn’, ‘PuRd’, ‘Purples’, ‘RdPu’, ‘Reds’, ‘YlGn’, ‘YlGnBu’, ‘YlOrBr’, ‘YlOrRd’]),

(‘Sequential (2)’, [‘afmhot’, ‘autumn’, ‘bone’, ‘cool’, ‘copper’, ‘gist_heat’, ‘gray’, ‘hot’, ‘pink’, ‘spring’, ‘summer’, ‘winter’]),

('Diverging', ['BrBG', 'bwr', 'coolwarm', 'PiYG', 'PRGn', 'PuOr', 'RdBu', 'RdGy', 'RdYlBu', 'RdYlGn', 'Spectral', 'seismic']),

('Qualitative', ['Accent', 'Dark2', 'Paired', 'Pastel1', 'Pastel2', 'Set1', 'Set2', 'Set3']),

('Miscellaneous', ['gist_earth', 'terrain', 'ocean', 'gist_stern', 'brg', 'CMRmap', 'cubehelix', 'gnuplot', 'gnuplot2', 'gist_ncar', 'nipy_spectral', 'jet', 'rainbow', 'gist_rainbow', 'hsv', 'flag', 'prism'])]

CommandLine: python -m wbia.plottool.color_funcs –test-show_all_colormaps –show python -m wbia.plottool.color_funcs –test-show_all_colormaps –show –type=Miscellaneous python -m wbia.plottool.color_funcs –test-show_all_colormaps –show –cmap=RdYlBu

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.color_funcs import *  # NOQA
>>> import wbia.plottool as pt
>>> show_all_colormaps()
>>> pt.show_if_requested()
```

wbia.plottool.color_funcs.**testshow_colors**(*rgb_list*, *gray=False*)

colors = ['r', 'b', 'purple', 'orange', 'deeppink', 'g']

colors = list(mcolors.CSS4_COLORS.keys())
CommandLine: python -m wbia.plottool.color_funcs testshow_colors –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.color_funcs import *  # NOQA
>>> colors = ut.get_argval('--colors', type_=list, default=['k', 'r'])
>>> ut.quit_if_noshow()
>>> rgb_list = ut.emap(ensure_base01, colors)
>>> testshow_colors(rgb_list)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.color_funcs.**to_base01**(*color255*)

converts base 255 color to base 01 color

wbia.plottool.color_funcs.**to_base255**(*color01*, *assume01=False*)

converts base 01 color to base 255 color

1.11.9 wbia.plottool.custom_constants module

wbia.plottool.custom_constants.**FontProp**(**args*, ***kwargs*)

overwrite fontproperties with custom settings

Kwargs: fname=u'', name=u'', style=u'normal', variant=u'normal', weight=u'normal', stretch=u'normal', size=u'medium'

wbia.plottool.custom_constants.**golden_wh**(*x*)

returns a width / height with a golden aspect ratio

wbia.plottool.custom_constants.**golden_wh2**(*sz*)

1.11.10 wbia.plottool.custom_figure module

wbia.plottool.custom_figure.**cla**()

wbia.plottool.custom_figure.**clf**()

wbia.plottool.custom_figure.**customize_figure**(*fig*, *docla*)

wbia.plottool.custom_figure.**customize_fontprop**(*font_prop*, ***fontkw*)

wbia.plottool.custom_figure.**ensure_fig**(*fnum=None*)

wbia.plottool.custom_figure.**figure**(*fnum=None*, *pnum=(1, 1, 1)*, *docla=False*, *title=None*, *figtitle=None*, *doclf=False*, *projection=None*, ***kwargs*)

http://matplotlib.org/users/gridspec.html

Parameters

- **fnum** (*int*) – fignum = figure number
- **pnum** (*int, str, or tuple(int, int, int)*) – plotnum = plot tuple
- **docla** (*bool*) – (default = False)
- **title** (*str*) – (default = None)
- **figtitle** (*None*) – (default = None)
- **doclf** (*bool*) – (default = False)
- **projection** (*None*) – (default = None)

Returns fig

Return type

?

CommandLine: python -m wbia.plottool.custom_figure –exec-figure:0 –show python -m wbia.plottool.custom_figure –exec-figure:1 –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.custom_figure import *  # NOQA
>>> fnum = 1
>>> fig = figure(fnum, (2, 2, 1))
>>> gca().text(0.5, 0.5, "ax1", va="center", ha="center")
>>> fig = figure(fnum, (2, 2, 2))
>>> gca().text(0.5, 0.5, "ax2", va="center", ha="center")
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.custom_figure import *  # NOQA
>>> fnum = 1
>>> fig = figure(fnum, (2, 2, 1))
>>> gca().text(0.5, 0.5, "ax1", va="center", ha="center")
>>> fig = figure(fnum, (2, 2, 2))
```

(continues on next page)

(continued from previous page)

```
>>> gca().text(0.5, 0.5, "ax2", va="center", ha="center")
>>> fig = figure(fnum, (2, 4, (1, slice(1, None))))
>>> gca().text(0.5, 0.5, "ax3", va="center", ha="center")
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.custom_figure.**gca**()

wbia.plottool.custom_figure.**gcf**()

wbia.plottool.custom_figure.**get_ax**(*fnum=None*, *pnum=None*)

wbia.plottool.custom_figure.**get_fig**(*fnum=None*)
DEPRICATE use ensure_fig

wbia.plottool.custom_figure.**get_image_from_figure**(*fig*)
saves figure data to an ndarray

References

http://stackoverflow.com/questions/7821518/save-plot-to-numpy-array

wbia.plottool.custom_figure.**prepare_figure_for_save**(*fnum*, *dpi=None*, *figsize=None*, *fig=None*)
so bad

wbia.plottool.custom_figure.**prepare_figure_fpath**(*fig*, *fpath*, *fnum*, *usetitle*, *defaultext*, *verbose*, *dpath=None*)

wbia.plottool.custom_figure.**sanitize_img_ext**(*ext*, *defaultext=None*)

wbia.plottool.custom_figure.**sanitize_img_fname**(*fname*)
Removes bad characters from images fnames

wbia.plottool.custom_figure.**save_figure**(*fnum=None*, *fpath=None*, *fpath_strict=None*, *usetitle=False*, *overwrite=True*, *defaultext=None*, *verbose=1*, *dpi=None*, *figsize=None*, *saveax=None*, *fig=None*, *dpath=None*)
Helper to save the figure image to disk. Tries to be smart about filename lengths, extensions, overwrites, etc...

DEPCIATE

Parameters

- **fnum** (*int*) – figure number
- **fpath** (*str*) – file path string
- **fpath_strict** (*str*) – uses this exact path
- **usetitle** (*bool*) – uses title as the fpath
- **overwrite** (*bool*) – default=True
- **defaultext** (*str*) – default extension
- **verbose** (*int*) – verbosity flag
- **dpi** (*int*) – dots per inch
- **figsize** (*tuple(int, int)*) – figure size
- **saveax** (*bool or Axes*) – specifies if the axes should be saved instead of the figure

References

for saving only a specific Axes http://stackoverflow.com/questions/4325733/save-a-subplot-in-matplotlib http://robotics.usc.edu/~ampereir/wordpress/?p=626 http://stackoverflow.com/questions/1271023/resize-a-figure-automatically-in-matplotlib

wbia.plottool.custom_figure.**set_figtitle**(*figtitle*, *subtitle=''*, *forcefignum=True*, *incanvas=True*, *size=None*, *fontfamily=None*, *fontweight=None*, *fig=None*, *font=None*)

Parameters

- **figtitle** –
- **subtitle** (*str*) – (default = '')
- **forcefignum** (*bool*) – (default = True)
- **incanvas** (*bool*) – (default = True)
- **fontfamily** (*None*) – (default = None)
- **fontweight** (*None*) – (default = None)
- **size** (*None*) – (default = None)
- **fig** (*None*) – (default = None)

CommandLine: python -m wbia.plottool.custom_figure set_figtitle –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.custom_figure import *  # NOQA
>>> import wbia.plottool as pt
>>> fig = pt.figure(fnum=1, doclf=True)
>>> result = pt.set_figtitle(figtitle='figtitle', fig=fig)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.custom_figure.**set_ticks**(*xticks*, *yticks*)

wbia.plottool.custom_figure.**set_title**(*title=''*, *ax=None*, ***fontkw*)

wbia.plottool.custom_figure.**set_xlabel**(*lbl*, *ax=None*, ***kwargs*)

Parameters

- **lbl** –
- **ax** (*None*) – (default = None)
- ****kwargs** –

CommandLine: python -m wbia.plottool.custom_figure set_xlabel

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.custom_figure import *  # NOQA
>>> import wbia.plottool as pt
```

(continues on next page)

(continued from previous page)

```
>>> fig = pt.figure()
>>> pt.adjust_subplots(fig=fig, bottom=.5)
>>> ax = pt.gca()
>>> lbl = 'a\nab\nabc'
>>> result = set_xlabel(lbl, ax)
>>> xaxis = ax.get_xaxis()
>>> xlabel = xaxis.get_label()
>>> xlabel.set_horizontalalignment('left')
>>> xlabel.set_x(0)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.custom_figure.**set_xticks**(*tick_set*)

wbia.plottool.custom_figure.**set_ylabel**(*lbl*, *ax=None*, ***kwargs*)

wbia.plottool.custom_figure.**set_yticks**(*tick_set*)

1.11.11 wbia.plottool.draw_func2 module

Lots of functions for drawing and plotting visiony things

class wbia.plottool.draw_func2.**OffsetImage2**(*arr*, *zoom=1*, *cmap=None*, *norm=None*, *interpolation=None*, *origin=None*, *filternorm=1*, *filterrad=4.0*, *resample=False*, *dpi_cor=True*, ***kwargs*)

Bases: matplotlib.offsetbox.OffsetBox

TODO: If this works reapply to mpl

draw(*renderer*)
Draw the children

get_children()
Return a list of the child .*Artist*s.

get_data()

get_extent(*renderer*)
Return a tuple `width, height, xdescent, ydescent` of the box.

get_offset()
return offset of the container.

get_window_extent(*renderer*)
get the bounding box in display space.

get_zoom()

set(***, *agg_filter=<UNSET>*, *alpha=<UNSET>*, *animated=<UNSET>*, *clip_box=<UNSET>*, *clip_on=<UNSET>*, *clip_path=<UNSET>*, *data=<UNSET>*, *gid=<UNSET>*, *height=<UNSET>*, *in_layout=<UNSET>*, *label=<UNSET>*, *offset=<UNSET>*, *path_effects=<UNSET>*, *picker=<UNSET>*, *rasterized=<UNSET>*, *sketch_params=<UNSET>*, *snap=<UNSET>*, *transform=<UNSET>*, *url=<UNSET>*, *visible=<UNSET>*, *width=<UNSET>*, *zoom=<UNSET>*, *zorder=<UNSET>*)
Set multiple properties at once.

Supported properties are

Properties: agg_filter: a filter function, which takes a (m, n, 3) float array and a dpi value, and returns a (m, n, 3) array alpha: scalar or None animated: bool clip_box: *.Bbox* clip_on: bool clip_path: Patch or (Path, Transform) or None data: unknown figure: *~matplotlib.figure.Figure* gid: str height: float in_layout: bool label: object offset: (float, float) or callable path_effects: *.AbstractPathEffect* picker: None or bool or float or callable rasterized: bool sketch_params: (scale: float, length: float, randomness: float) snap: bool or None transform: *.Transform* url: str visible: bool width: float zoom: unknown zorder: float

set_data(*arr*)

set_zoom(*zoom*)

class wbia.plottool.draw_func2.**RenderingContext**(***savekw*)

Bases: object

wbia.plottool.draw_func2.**absolute_lbl**(*x_*, *y_*, *txt*, *roffset=(-0.02, -0.02)*, *alpha=0.6*, ***kwargs*)

alternative to relative text

wbia.plottool.draw_func2.**absolute_text**(*pos*, *text*, *ax=None*, ***kwargs*)

wbia.plottool.draw_func2.**add_alpha**(*colors*)

wbia.plottool.draw_func2.**adjust_subplots**(*left=None*, *right=None*, *bottom=None*, *top=None*, *wspace=None*, *hspace=None*, *use_argv=False*, *fig=None*)

Kwargs: left (float): left side of the subplots of the figure right (float): right side of the subplots of the figure bottom (float): bottom of the subplots of the figure top (float): top of the subplots of the figure wspace (float): width reserved for blank space between subplots hspace (float): height reserved for blank space between subplots

wbia.plottool.draw_func2.**append_phantom_legend_label**(*label*, *color*, *type_='circle'*, *alpha=1.0*, *ax=None*)

adds a legend label without displaying an actor

Parameters

- **label** –
- **color** –
- **loc** (*str*) –

CommandLine: python -m wbia.plottool.draw_func2 –test-append_phantom_legend_label –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import wbia.plottool as pt
>>> label = 'some label'
>>> color = 'b'
>>> loc = 'upper right'
>>> fig = pt.figure()
>>> ax = pt.gca()
>>> result = append_phantom_legend_label(label, color, loc, ax=ax)
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.quit_if_noshow()
>>> pt.show_phantom_legend_labels(ax=ax)
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**ax_absolute_text**(*x_*, *y_*, *txt*, *ax=None*, *roffset=None*, ***kwargs*)
Base function for text
Kwargs: horizontalalignment in ['right', 'center', 'left'], verticalalignment in ['top'] color

wbia.plottool.draw_func2.**axes_bottom_button_bar**(*ax*, *text_list=[]*)

wbia.plottool.draw_func2.**axes_extent**(*axs*, *pad=0.0*)
Get the full extent of a group of axes, including axes labels, tick labels, and titles.

wbia.plottool.draw_func2.**cartoon_stacked_rects**(*xy*, *width*, *height*, *num=4*, *shift=None*, ***kwargs*)
pt.figure() xy = (.5, .5) width = .2 height = .2 ax = pt.gca() ax.add_collection(col)

wbia.plottool.draw_func2.**color_orimag**(*gori*, *gmag=None*, *gmag_is_01=None*, *encoding='rgb'*, *p=0.5*)

Parameters

- **gori** (*ndarray*) – orientation values at pixels between 0 and tau
- **gmag** (*ndarray*) – orientation magnitude
- **gmag_is_01** (*bool*) – True if gmag is in the 0 and 1 range. if None we try to guess
- **p** (*float*) – power to raise normalized weights to for visualization purposes

Returns rgb_ori or bgr_ori

Return type ndarray

CommandLine: python -m wbia.plottool.draw_func2 –test-color_orimag –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import wbia.plottool as pt
>>> import vtool as vt
>>> # build test data
>>> gori = np.array([[ 0.        ,  0.        ,  3.14159265,  3.14159265,  0.
↪    ],
...                  [ 1.57079633,  3.92250052,  1.81294053,  3.29001537,  1.
↪57079633],
...                  [ 4.71238898,  6.15139659,  0.76764078,  1.75632531,  1.
↪57079633],
...                  [ 4.71238898,  4.51993581,  6.12565345,  3.87978382,  1.
↪57079633],
...                  [ 0.        ,  0.        ,  0.        ,  0.        ,  0.
↪    ]])
>>> gmag = np.array([[ 0.        ,  0.02160321,  0.00336692,  0.06290751,  0.
↪    ],
...                      [ 0.02363726,  0.04195344,  0.29969492,  0.53007415,  0.
↪0426679 ],
...                      [ 0.00459386,  0.32086307,  0.02844123,  0.24623816,  0.
↪27344167],
...                      [ 0.04204251,  0.52165989,  0.25800464,  0.14568752,  0.
↪023614  ],
...                      [ 0.        ,  0.05143869,  0.2744546 ,  0.01582246,  0.
↪       ]])
>>> # execute function
>>> p = 1
```

(continues on next page)

(continued from previous page)

```
>>> bgr_ori1 = color_orimag(gori, gmag, encoding='bgr', p=p)
>>> bgr_ori2 = color_orimag(gori, None, encoding='bgr')
>>> legendimg = pt.make_ori_legend_img().astype(np.float32) / 255.0
>>> gweights_color = np.dstack([gmag] * 3).astype(np.float32)
>>> img, _, _ = vt.stack_images(bgr_ori2, gweights_color, vert=False)
>>> img, _, _ = vt.stack_images(img, bgr_ori1, vert=False)
>>> img, _, _ = vt.stack_images(img, legendimg, vert=True, modifysize=True)
>>> # verify results
>>> pt.imshow(img, pnum=(1, 2, 1))
>>> # Hack orientation offset so 0 is downward
>>> gradx, grady = np.cos(gori + TAU / 4.0), np.sin(gori + TAU / 4.0)
>>> pt.imshow(bgr_ori2, pnum=(1, 2, 2))
>>> pt.draw_vector_field(gradx, grady, pnum=(1, 2, 2), invert=False)
>>> color_orimag_colorbar(gori)
>>> pt.set_figtitle('weighted and unweighted orientaiton colors')
>>> pt.update()
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**color_orimag_colorbar**(*gori*)

wbia.plottool.draw_func2.**colorbar**(*scalars*, *colors*, *custom=False*, *lbl=None*, *ticklabels=None*, *float_format='%.2f'*, ***kwargs*)

adds a color bar next to the axes based on specific scalars

Parameters

- **scalars** (*ndarray*) –
- **colors** (*ndarray*) –
- **custom** (*bool*) – use custom ticks

Kwargs: See plt.colorbar

Returns matplotlib colorbar object

Return type cb

CommandLine: python -m wbia.plottool.draw_func2 –exec-colorbar –show python -m wbia.plottool.draw_func2 –exec-colorbar:1 –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> from wbia.plottool import draw_func2 as df2
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> scalars = np.array([-1, -2, 1, 1, 2, 7, 10])
>>> cmap_ = 'plasma'
>>> logscale = False
>>> custom = True
>>> reverse_cmap = True
>>> val2_customcolor  = {
...         -1: UNKNOWN_PURP,
...         -2: LIGHT_BLUE,
...     }
>>> colors = scores_to_color(scalars, cmap_=cmap_, logscale=logscale, reverse_
↪cmap=reverse_cmap, val2_customcolor=val2_customcolor)
>>> colorbar(scalars, colors, custom=custom)
```

(continues on next page)

(continued from previous page)

```
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> from wbia.plottool import draw_func2 as df2
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import wbia.plottool as pt
>>> scalars = np.linspace(0, 1, 100)
>>> cmap_ = 'plasma'
>>> logscale = False
>>> custom = False
>>> reverse_cmap = False
>>> colors = scores_to_color(scalars, cmap_=cmap_, logscale=logscale,
>>>                          reverse_cmap=reverse_cmap)
>>> colors = [pt.lighten_rgb(c, .3) for c in colors]
>>> colorbar(scalars, colors, custom=custom)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**customize_colormap**(*data*, *base_colormap*)

wbia.plottool.draw_func2.**dark_background**(*ax=None*, *doubleit=False*, *force=False*)

Parameters

- **ax** (*None*) – (default = None)
- **doubleit** (*bool*) – (default = False)

CommandLine: python -m wbia.plottool.draw_func2 –exec-dark_background –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import wbia.plottool as pt
>>> fig = pt.figure()
>>> pt.dark_background()
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**distinct_markers**(*num*, *style='astrisk'*, *total=None*, *offset=0*)

Parameters **num** –

CommandLine: python -m wbia.plottool.draw_func2 –exec-distinct_markers –show python -m wbia.plottool.draw_func2 –exec-distinct_markers –mstyle=star –show python -m wbia.plottool.draw_func2 –exec-distinct_markers –mstyle=polygon –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import wbia.plottool as pt
>>> style = ut.get_argval('--mstyle', type_=str, default='astrisk')
>>> marker_list = distinct_markers(10, style)
>>> x_data = np.arange(0, 3)
>>> for count, (marker) in enumerate(marker_list):
>>>     pt.plot(x_data, [count] * len(x_data), marker=marker, markersize=10,␣
↪linestyle='', label=str(marker))
>>> pt.legend()
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**draw_bbox**(*bbox*, *lbl=None*, *bbox_color=(1, 0, 0)*, *lbl_bgcolor=(0, 0, 0)*, *lbl_txtcolor=(1, 1, 1)*, *draw_arrow=True*, *theta=0*, *ax=None*, *lw=2*)

wbia.plottool.draw_func2.**draw_border**(*ax*, *color=array([0., 1., 0., 1.])*, *lw=2*, *offset=None*, *adjust=True*)

draws rectangle border around a subplot

wbia.plottool.draw_func2.**draw_boxedX**(*xywh=None*, *color=array([1., 0., 0., 1.])*, *lw=2*, *alpha=0.5*, *theta=0*, *ax=None*)

draws a big red x

wbia.plottool.draw_func2.**draw_keypoint_gradient_orientations**(*rchip*, *kpt*, *sift=None*, *mode='vec'*, *kptkw={}*, *siftkw={}*, ***kwargs*)

Extracts a keypoint patch from a chip, extract the gradient, and visualizes it with respect to the current mode.

wbia.plottool.draw_func2.**draw_keypoint_patch**(*rchip*, *kp*, *sift=None*, *warped=False*, *patch_dict={}*, ***kwargs*)

Parameters

- **rchip** (*ndarray[uint8_t, ndim=2]*) – rotated annotation image data
- **kp** (*ndarray[float32_t, ndim=1]*) – a single keypoint
- **sift** (*None*) – (default = None)
- **warped** (*bool*) – (default = False)
- **patch_dict** (*dict*) – (default = {})

Returns ax

Return type

?

CommandLine: python -m wbia.plottool.draw_func2 –test-draw_keypoint_patch –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import vtool as vt
>>> rchip = vt.imread(ut.grab_test_imgpath('lena.png'))
```

(continues on next page)

(continued from previous page)

```
>>> kp = [100, 100, 20, 0, 20, 0]
>>> sift = None
>>> warped = True
>>> patch_dict = {}
>>> ax = draw_keypoint_patch(rchip, kp, sift, warped, patch_dict)
>>> result = ('ax = %s' % (str(ax),))
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**draw_kpts2**(*kpts*, *offset=(0, 0)*, *scale_factor=1*, *ell=True*, *pts=False*, *rect=False*, *eig=False*, *ori=False*, *pts_size=2*, *ell_alpha=0.6*, *ell_linewidth=1.5*, *ell_color=None*, *pts_color=array([1., 0.49803922, 0., 1.])*, *color_list=None*, *pts_alpha=1.0*, *siftkw={}*, *H=None*, *weights=None*, *cmap_='hot'*, *ax=None*, ***kwargs*)

thin wrapper around mpl_keypoint.draw_keypoints

FIXME: seems to be off by (.5, .5) translation

Parameters

- **kpts** –
- **offset** (*tuple*) –
- **scale_factor** (*int*) –
- **ell** (*bool*) –
- **pts** (*bool*) –
- **rect** (*bool*) –
- **eig** (*bool*) –
- **ori** (*bool*) –
- **pts_size** (*int*) –
- **ell_alpha** (*float*) –
- **ell_linewidth** (*float*) –
- **ell_color** (*None*) –
- **pts_color** (*ndarray*) –
- **color_list** (*list*) –

Example

```
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> from wbia.plottool import draw_func2 as df2
>>> offset = (0, 0)
>>> scale_factor = 1
>>> ell = True
>>> ell=True
>>> pts=False
>>> rect=False
>>> eig=False
```

(continues on next page)

(continued from previous page)

```
>>> ell=True
>>> pts=False
>>> rect=False
>>> eig=False
>>> ori=False
>>> pts_size=2
>>> ell_alpha=.6
>>> ell_linewidth=1.5
>>> ell_color=None
>>> pts_color=df2.ORANGE
>>> color_list=None
```

wbia.plottool.draw_func2.**draw_line_segments**(*segments_list*, ***kwargs*)

segments_list - list of [xs,ys,...] defining the segments

wbia.plottool.draw_func2.**draw_line_segments2**(*pts1*, *pts2*, *ax=None*, ***kwargs*)

draws *N* line segments

Parameters

- **pts1** (*ndarray*) – Nx2
- **pts2** (*ndarray*) – Nx2
- **ax** (*None*) – (default = None)

CommandLine: python -m wbia.plottool.draw_func2 draw_line_segments2 –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import wbia.plottool as pt
>>> pts1 = np.array([(1, 1), (0, 0)])
>>> pts2 = np.array([(2, 2), (1, 0)])
>>> pt.figure(fnum=None)
>>> #segments = [np.array((xy1, xy2)) for xy1, xy2 in zip(pts1, pts2)]
>>> #draw_line_segments(segments)
>>> draw_line_segments2(pts1, pts2)
>>> import wbia.plottool as pt
>>> pt.quit_if_noshow()
>>> ax = pt.gca()
>>> pt.set_axis_limit(-1, 3, -1, 3, ax)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**draw_lines2**(*kpts1*, *kpts2*, *fm=None*, *fs=None*, *kpts2_offset=(0, 0)*, *color_list=None*, *scale_factor=1*, *lw=1.4*, *line_alpha=0.35*, *H1=None*, *H2=None*, *scale_factor1=None*, *scale_factor2=None*, *ax=None*, ***kwargs*)

wbia.plottool.draw_func2.**draw_patches_and_sifts**(*patch_list*, *sift_list*, *fnum=None*, *pnum=(1, 1, 1)*)

wbia.plottool.draw_func2.**draw_stems**(*x_data=None*, *y_data=None*, *setlims=True*, *color=None*, *markersize=None*, *bottom=None*, *marker=None*, *linestyle='-'*)

Draws stem plot

Parameters

- **x_data** (*None*) –
- **y_data** (*None*) –
- **setlims** (*bool*) –
- **color** (*None*) –
- **markersize** (*None*) –
- **bottom** (*None*) –

References

http://exnumerus.blogspot.com/2011/02/how-to-quickly-plot-multiple-line.html

CommandLine: python -m wbia.plottool.draw_func2 –test-draw_stems –show python -m wbia.plottool.draw_func2 –test-draw_stems

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> x_data = np.append(np.arange(1, 10), np.arange(1, 10))
>>> rng = np.random.RandomState(0)
>>> y_data = sorted(rng.rand(len(x_data)) * 10)
>>> # y_data = np.array([ut.get_nth_prime(n) for n in x_data])
>>> setlims = False
>>> color = [1.0, 0.0, 0.0, 1.0]
>>> markersize = 2
>>> marker = 'o'
>>> bottom = None
>>> result = draw_stems(x_data, y_data, setlims, color, markersize, bottom,␣
↪marker)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**draw_text**(*text_str*, *rgb_textFG=(0, 0, 0)*, *rgb_textBG=(1, 1, 1)*)

wbia.plottool.draw_func2.**draw_text_annotations**(*text_list*, *pos_list*, *bbox_offset_list=[0, 0]*, *pos_offset_list=[0, 0]*, *bbox_align_list=[0, 0]*, *color_list=None*, *textprops={}*)

Hack fixes to issues in text annotations

wbia.plottool.draw_func2.**draw_vector_field**(*gx*, *gy*, *fnum=None*, *pnum=None*, *title=None*, *invert=True*, *stride=1*)

CommandLine: python -m wbia.plottool.draw_func2 draw_vector_field –show python -m wbia.plottool.draw_func2 draw_vector_field –show –fname=zebra.png –fx=121 –stride=3

Example

```
>>> # DISABLE_DOCTEST
>>> import wbia.plottool as pt
>>> import utool as ut
```

(continues on next page)

(continued from previous page)

```
>>> import vtool as vt
>>> patch = vt.testdata_patch()
>>> gx, gy = vt.patch_gradient(patch, gaussian_weighted=False)
>>> stride = ut.get_argval('--stride', default=1)
>>> pt.draw_vector_field(gx, gy, stride=stride)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**ensure_divider**(*ax*)
Returns previously constructed divider or creates one

wbia.plottool.draw_func2.**ensure_fnum**(*fnum*)

wbia.plottool.draw_func2.**execstr_global**()

wbia.plottool.draw_func2.**extract_axes_extents**(*fig*, *combine=False*, *pad=0.0*)
CommandLine:

python -m wbia.plottool.draw_func2 extract_axes_extents python -m wbia.plottool.draw_func2 extract_axes_extents –save foo.jpg

Notes: contour does something weird to axes with contour:

axes_extents = Bbox([[-0.839827203337, -0.00555555555556], [7.77743055556, 6.97227277762]])

without contour axes_extents = Bbox([[0.0290607810781, -0.00555555555556], [7.77743055556, 5.88]])

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import wbia.plottool as pt
>>> import matplotlib.gridspec as gridspec
>>> import matplotlib.pyplot as plt
>>> pt.qtensure()
>>> fig = plt.figure()
>>> gs = gridspec.GridSpec(17, 17)
>>> specs = [
>>>     gs[0:8,  0:8], gs[0:8,  8:16],
>>>     gs[9:17, 0:8], gs[9:17, 8:16],
>>> ]
>>> rng = np.random.RandomState(0)
>>> X = (rng.rand(100, 2) * [[8, 8]]) + [[6, -14]]
>>> x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
>>> y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
>>> xx, yy = np.meshgrid(np.arange(x_min, x_max), np.arange(y_min, y_max))
>>> yynan = np.full(yy.shape, fill_value=np.nan)
>>> xxnan = np.full(yy.shape, fill_value=np.nan)
>>> cmap = plt.cm.RdYlBu
>>> norm = plt.Normalize(vmin=0, vmax=1)
>>> for count, spec in enumerate(specs):
>>>     fig.add_subplot(spec)
>>>     plt.plot(X.T[0], X.T[1], 'o', color='r', markeredgecolor='w')
>>>     Z = rng.rand(*xx.shape)
```

(continues on next page)

(continued from previous page)

```
>>>     plt.contourf(xx, yy, Z, cmap=cmap, norm=norm, alpha=1.0)
>>>     plt.title('full-nan decision point')
>>>     plt.gca().set_aspect('equal')
>>> gs = gridspec.GridSpec(1, 16)
>>> subspec = gs[:, -1:]
>>> cax = plt.subplot(subspec)
>>> sm = plt.cm.ScalarMappable(cmap=cmap)
>>> sm.set_array(np.linspace(0, 1))
>>> plt.colorbar(sm, cax)
>>> cax.set_ylabel('ColorBar')
>>> fig.suptitle('SupTitle')
>>> subkw = dict(left=.001, right=.9, top=.9, bottom=.05, hspace=.2, wspace=.1)
>>> plt.subplots_adjust(**subkw)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**fig_relative_text**(*x*, *y*, *txt*, ***kwargs*)

wbia.plottool.draw_func2.**fnum_generator**(*base=1*)

wbia.plottool.draw_func2.**get_all_markers**()

CommandLine: python -m wbia.plottool.draw_func2 –exec-get_all_markers –show

References

http://matplotlib.org/1.3.1/examples/pylab_examples/line_styles.html http://matplotlib.org/api/markers_api.html#matplotlib.markers.MarkerStyle.markers

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import wbia.plottool as pt
>>> marker_dict = get_all_markers()
>>> x_data = np.arange(0, 3)
>>> for count, (marker, name) in enumerate(marker_dict.items()):
>>>     pt.plot(x_data, [count] * len(x_data), marker=marker, linestyle='',␣
↪label=name)
>>> pt.legend()
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**get_axis_bbox**(*ax=None*, ***kwargs*)

returns in figure coordinates?

wbia.plottool.draw_func2.**get_axis_xy_width_height**(*ax=None*, *xaug=0*, *yaug=0*, *waug=0*, *haug=0*)

gets geometry of a subplot

wbia.plottool.draw_func2.**get_binary_svm_cmap**()

wbia.plottool.draw_func2.**get_num_rc**(*nSubplots=None*, *nRows=None*, *nCols=None*)

Gets a constrained row column plot grid

Parameters

- **nSubplots** (*None*) – (default = None)

- **nRows** (*None*) – (default = None)
- **nCols** (*None*) – (default = None)

Returns (nRows, nCols)

Return type tuple

CommandLine: python -m wbia.plottool.draw_func2 get_num_rc

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> cases = [
>>>     dict(nRows=None, nCols=None, nSubplots=None),
>>>     dict(nRows=2, nCols=None, nSubplots=5),
>>>     dict(nRows=None, nCols=2, nSubplots=5),
>>>     dict(nRows=None, nCols=None, nSubplots=5),
>>> ]
>>> for kw in cases:
>>>     print('----')
>>>     size = get_num_rc(**kw)
>>>     if kw['nSubplots'] is not None:
>>>         assert size[0] * size[1] >= kw['nSubplots']
>>>     print('**kw = %s' % (ut.repr2(kw),))
>>>     print('size = %r' % (size,))
```

wbia.plottool.draw_func2.**get_orientation_color**(*radians_list*)

Parameters **radians_list** (*list*) –

CommandLine: python -m wbia.plottool.draw_func2 –test-get_orientation_color

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> # build test data
>>> radians_list = np.linspace(-1, 10, 10)
>>> # execute function
>>> result = get_orientation_color(radians_list)
>>> # verify results
>>> print(result)
```

wbia.plottool.draw_func2.**get_pnum_func**(*nRows=1*, *nCols=1*, *base=0*)

wbia.plottool.draw_func2.**imshow**(*img*, *fnum=None*, *title=None*, *figtitle=None*, *pnum=None*, *interpolation='nearest'*, *cmap=None*, *heatmap=False*, *data_colorbar=False*, *darken=None*, *update=False*, *xlabel=None*, *redraw_image=True*, *ax=None*, *alpha=None*, *norm=None*, ***kwargs*)

Parameters

- **img** (*ndarray*) – image data
- **fnum** (*int*) – figure number
- **title** (*str*) –

- **figtitle** (*None*) –
- **pnum** (*tuple*) – plot number
- **interpolation** (*str*) – other interpolations = nearest, bicubic, bilinear
- **cmap** (*None*) –
- **heatmap** (*bool*) –
- **data_colorbar** (*bool*) –
- **darken** (*None*) –
- **update** (*bool*) – (default = False)
- **redraw_image** (*bool*) – used when calling imshow over and over. if false doesnt do the image part.

Returns (fig, ax)

Return type tuple

Kwargs: docla, doclf, projection

Returns (fig, ax)

Return type tuple

CommandLine: python -m wbia.plottool.draw_func2 –exec-imshow –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import vtool as vt
>>> img_fpath = ut.grab_test_imgpath('carl.jpg')
>>> img = vt.imread(img_fpath)
>>> (fig, ax) = imshow(img)
>>> result = ('(fig, ax) = %s' % (str((fig, ax)),))
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**imshow_null**(*msg=None*, *ax=None*, ***kwargs*)

Parameters

- **msg** (*None*) – (default = None)
- **ax** (*None*) – (default = None)
- ****kwargs** – fnum, title, figtitle, pnum, interpolation, cmap, heatmap, data_colorbar, darken, update, xlabel, redraw_image, alpha, docla, doclf, projection, use_gridspec

CommandLine: python -m wbia.plottool.draw_func2 imshow_null –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> msg = None
```

(continues on next page)

(continued from previous page)

```
>>> ax = None
>>> result = imshow_null(msg, ax)
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**interpolated_colormap**(*color_frac_list*, *resolution=64*, *space='lch-ab'*)

http://stackoverflow.com/questions/12073306/customize-colorbar-in-matplotlib

CommandLine: python -m wbia.plottool.draw_func2 interpolated_colormap –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import wbia.plottool as pt
>>> color_frac_list = [
>>>     (pt.TRUE_BLUE, 0),
>>>     #(pt.WHITE, .5),
>>>     (pt.YELLOW, .5),
>>>     (pt.FALSE_RED, 1.0),
>>> ]
>>> color_frac_list = [
>>>     (pt.RED, 0),
>>>     (pt.PINK, .1),
>>>     (pt.ORANGE, .2),
>>>     (pt.GREEN, .5),
>>>     (pt.TRUE_BLUE, .7),
>>>     (pt.PURPLE, 1.0),
>>> ]
>>> color_frac_list = [
>>>     (pt.RED,      0/6),
>>>     (pt.YELLOW,   1/6),
>>>     (pt.GREEN,    2/6),
>>>     (pt.CYAN,     3/6),
>>>     (pt.BLUE,     4/6),  # FIXME doesn't go in correct direction
>>>     (pt.MAGENTA,  5/6),
>>>     (pt.RED,      6/6),
>>> ]
>>> color_frac_list = [
>>>     ((1, 0, 0, 0),  0/6),
>>>     ((1, 0, .001/255, 0),  6/6), # hack
>>> ]
>>> space = 'hsv'
>>> color_frac_list = [
>>>     (pt.BLUE,   0.0),
>>>     (pt.GRAY,   0.5),
>>>     (pt.YELLOW, 1.0),
>>> ]
>>> color_frac_list = [
>>>     (pt.GREEN,  0.0),
>>>     (pt.GRAY,   0.5),
>>>     (pt.RED,    1.0),
>>> ]
>>> space = 'lab'
```

(continues on next page)

(continued from previous page)

```
>>> #resolution = 16 + 1
>>> resolution = 256 + 1
>>> cmap = interpolated_colormap(color_frac_list, resolution, space)
>>> import wbia.plottool as pt
>>> pt.quit_if_noshow()
>>> a = np.linspace(0, 1, resolution).reshape(1, -1)
>>> pylab.imshow(a, aspect='auto', cmap=cmap, interpolation='nearest')  # ,␣
↪origin="lower")
>>> plt.grid(False)
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**is_texmode**()

wbia.plottool.draw_func2.**label_to_colors**(*labels_*)
returns a unique and distinct color corresponding to each label

wbia.plottool.draw_func2.**legend**(*loc='best'*, *fontproperties=None*, *size=None*, *fc='w'*, *alpha=1*, *ax=None*, *handles=None*)

Parameters

- **loc** (*str*) – (default = 'best')
- **fontproperties** (*None*) – (default = None)
- **size** (*None*) – (default = None)

CommandLine: python -m wbia.plottool.draw_func2 –exec-legend –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> loc = 'best'
>>> import wbia.plottool as pt
>>> xdata = np.linspace(-6, 6)
>>> ydata = np.sin(xdata)
>>> pt.plot(xdata, ydata, label='sin')
>>> fontproperties = None
>>> size = None
>>> result = legend(loc, fontproperties, size)
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**lowerright_text**(*txt*)

wbia.plottool.draw_func2.**make_bbox**(*bbox*, *theta=0*, *bbox_color=None*, *ax=None*, *lw=2*, *alpha=1.0*, *align='center'*, *fill=None*, ***kwargs*)

wbia.plottool.draw_func2.**make_bbox_positioners**(*y=0.02*, *w=0.08*, *h=0.02*, *xpad=0.05*, *startx=0*, *stopx=1*)

wbia.plottool.draw_func2.**make_fnum_nextgen**(*base=1*)

wbia.plottool.draw_func2.**make_ori_legend_img**()
creates a figure that shows which colors are associated with which keypoint rotations.

a rotation of 0 should point downward (becuase it is relative the the (0, 1) keypoint eigenvector. and its color should be red due to the hsv mapping
CommandLine: python -m wbia.plottool.draw_func2 –test-make_ori_legend_img –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import wbia.plottool as pt
>>> # build test data
>>> # execute function
>>> img_BGR = make_ori_legend_img()
>>> # verify results
>>> pt.imshow(img_BGR)
>>> pt.iup()
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**make_pnum_nextgen**(*nRows=None*, *nCols=None*, *base=0*, *nSubplots=None*, *start=0*)

Parameters

- **nRows** (*None*) – (default = None)
- **nCols** (*None*) – (default = None)
- **base** (*int*) – (default = 0)
- **nSubplots** (*None*) – (default = None)
- **start** (*int*) – (default = 0)

Returns pnum_next

Return type iterator

CommandLine: python -m wbia.plottool.draw_func2 –exec-make_pnum_nextgen –show
GridParams:

```
>>> param_grid = dict(
>>>     nRows=[None, 3],
>>>     nCols=[None, 3],
>>>     nSubplots=[None, 9],
>>> )
>>> combos = ut.all_dict_combinations(param_grid)
```

GridExample:

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> base, start = 0, 0
>>> pnum_next = make_pnum_nextgen(nRows, nCols, base, nSubplots, start)
>>> pnum_list = list( (pnum_next() for _ in it.count()) )
>>> print((nRows, nCols, nSubplots))
>>> result = ('pnum_list = %s' % (ut.repr2(pnum_list),))
>>> print(result)
```

wbia.plottool.draw_func2.**next_fnum**(*new_base=None*)

wbia.plottool.draw_func2.**overlay_icon**(*icon*, *coords=(0, 0)*, *coord_type='axes'*, *bbox_alignment=(0, 0)*, *max_asize=None*, *max_dsize=None*, *as_artist=True*)

Overlay a species icon

References

http://matplotlib.org/examples/pylab_examples/demo_annotation_box.html http://matplotlib.org/users/annotations_guide.html /usr/local/lib/python2.7/dist-packages/matplotlib/offsetbox.py

Parameters

- **icon** (*ndarray or str*) – image icon data or path
- **coords** (*tuple*) – (default = (0, 0))
- **coord_type** (*str*) – (default = 'axes')
- **bbox_alignment** (*tuple*) – (default = (0, 0))
- **max_dsize** (*None*) – (default = None)

CommandLine: python -m wbia.plottool.draw_func2 –exec-overlay_icon –show –icon zebra.png python -m wbia.plottool.draw_func2 –exec-overlay_icon –show –icon lena.png python -m wbia.plottool.draw_func2 –exec-overlay_icon –show –icon lena.png –artist

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import wbia.plottool as pt
>>> pt.plot2(np.arange(100), np.arange(100))
>>> icon = ut.get_argval('--icon', type_=str, default='lena.png')
>>> coords = (0, 0)
>>> coord_type = 'axes'
>>> bbox_alignment = (0, 0)
>>> max_dsize = None  # (128, None)
>>> max_asize = (60, 40)
>>> as_artist = not ut.get_argflag('--noartist')
>>> result = overlay_icon(icon, coords, coord_type, bbox_alignment,
>>>                       max_asize, max_dsize, as_artist)
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**pad_axes**(*pad*, *xlim=None*, *ylim=None*)

wbia.plottool.draw_func2.**param_plot_iterator**(*param_list*, *fnum=None*, *projection=None*)

wbia.plottool.draw_func2.**parse_fontkw**(***kwargs*)

Kwargs: fontsize, fontfamilty, fontproperties

wbia.plottool.draw_func2.**plot**(**args*, ***kwargs*)

wbia.plottool.draw_func2.**plot2**(*x_data*, *y_data*, *marker='o'*, *title_pref=''*, *x_label='x'*, *y_label='y'*, *unitbox=False*, *flipx=False*, *flipy=False*, *title=None*, *dark=None*, *equal_aspect=True*, *pad=0*, *label=''*, *fnum=None*, *pnum=None*, **args*, ***kwargs*)

don't forget to call pt.legend

Kwargs: linewidth (float):

wbia.plottool.draw_func2.**plot_bars**(*y_data*, *nColorSplits=1*)

wbia.plottool.draw_func2.**plot_descriptor_signature**(*vec*, *title=''*, *fnum=None*, *pnum=None*)

signature general for for any descriptor vector.

Parameters

- **vec** (*ndarray*) –
- **title** (*str*) – (default = '')
- **fnum** (*int*) – figure number(default = None)
- **pnum** (*tuple*) – plot number(default = None)

Returns ax

Return type AxesSubplot

CommandLine: python -m wbia.plottool.draw_func2 –test-plot_descriptor_signature –show
SeeAlso: plot_sift_signature

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import vtool as vt
>>> vec = ((np.random.RandomState(0).rand(258) - .2) * 4)
>>> title = 'test sift histogram'
>>> fnum = None
>>> pnum = None
>>> ax = plot_descriptor_signature(vec, title, fnum, pnum)
>>> result = ('ax = %s' % (str(ax),))
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**plot_fmatch**(*xywh1*, *xywh2*, *kpts1*, *kpts2*, *fm*, *fs=None*, *fm_norm=None*, *lbl1=None*, *lbl2=None*, *fnum=None*, *pnum=None*, *rect=False*, *colorbar_=True*, *draw_border=False*, *cmap=None*, *H1=None*, *H2=None*, *scale_factor1=None*, *scale_factor2=None*, *ax=None*, ***kwargs*)

Overlays the matching features over chips that were previously plotted.

Parameters

- **xywh1** (*tuple*) – location of rchip1 in the axes
- **xywh2** (*tuple*) – location or rchip2 in the axes
- **kpts1** (*ndarray*) – keypoints in rchip1
- **kpts2** (*ndarray*) – keypoints in rchip1
- **fm** (*list*) – feature matches
- **fs** (*list*) – features scores
- **fm_norm** (*None*) – (default = None)
- **lbl1** (*None*) – rchip1 label
- **lbl2** (*None*) – rchip2 label
- **fnum** (*None*) – figure number
- **pnum** (*None*) – plot number
- **rect** (*bool*) –
- **colorbar** (*bool*) –

- **draw_border** (*bool*) –
- **cmap** (*None*) – (default = None)
- **H1** (*None*) – (default = None)
- **H2** (*None*) – (default = None)
- **scale_factor1** (*None*) – (default = None)
- **scale_factor2** (*None*) – (default = None)

Kwargs: draw_pts, draw_ell, draw_lines, show_nMatches, all_kpts

Returns None

Return type

?

CommandLine: python -m wbia.plottool.draw_func2 –exec-plot_fmatch

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> xywh1 = '?'
>>> xywh2 = '?'
>>> kpts1 = '?'
>>> kpts2 = '?'
>>> fm = '?'
>>> fs = None
>>> fm_norm = None
>>> lbl1 = None
>>> lbl2 = None
>>> fnum = None
>>> pnum = None
>>> rect = False
>>> colorbar_ = True
>>> draw_border = False
>>> cmap = None
>>> H1 = None
>>> H2 = None
>>> scale_factor1 = None
>>> scale_factor2 = None
>>> plot_fmatch(xywh1, xywh2, kpts1, kpts2, fm, fs, fm_norm, lbl1, lbl2,
>>>             fnum, pnum, rect, colorbar_, draw_border, cmap, h1, h2,
>>>             scale_factor1, scale_factor2)
>>> result = ('None = %s' % (str(None),))
>>> print(result)
```

wbia.plottool.draw_func2.**plot_func**(*funcs*, *start=0*, *stop=1*, *num=100*, *setup=None*, *fnum=None*, *pnum=None*)

plots a numerical function in a given range

Parameters

- **funcs** (*list of function*) – live python function
- **start** (*int*) – (default = 0)
- **stop** (*int*) – (default = 1)

- **num** (*int*) – (default = 100)

CommandLine: python -m wbia.plottool.draw_func2 –exec-plot_func –show –range=-1,1 –func=np.exp python -m wbia.plottool.draw_func2 –exec-plot_func –show –range=-1,1 –func=scipy.special.logit python -m wbia.plottool.draw_func2 –exec-plot_func –show –range=0,1 –func="lambda x: scipy.special.expit(((x * 2) - 1.0) * 6)" python -m wbia.plottool.draw_func2 –exec-plot_func –show –range=0,1 –func="lambda x: scipy.special.expit(-6 + 12 * x)" python -m wbia.plottool.draw_func2 –exec-plot_func –show –range=0,4 –func="lambda x: vt.logistic_01((-1 + x) * 2)" python -m wbia.plottool.draw_func2 –exec-plot_func –show –range=0,1 –func="lambda x: np.tan((x - .5) * np.pi)" –ylim=-10,10 python -m wbia.plottool.draw_func2 –exec-plot_func –show –range=0,3 –func=np.tan python -m wbia.plottool.draw_func2 –exec-plot_func –show –range=0,50 –func="lambda x: np.exp(-x / 50)" python -m wbia.plottool.draw_func2 –exec-plot_func –show –range=-8,8 –func=vt.beaton_tukey_loss python -m wbia.plottool.draw_func2 –exec-plot_func –show –range=-8,8 –func=vt.beaton_tukey_weight,vt.beaton_tukey_loss

python -m wbia.plottool plot_func –show –range=-1,1 –setup="from wbia.algo.smk.smk_pipeline import SMK" –func=lambda u: SMK.selectivity(u, 3.0, 0)

python -m wbia.plottool plot_func –show –range=-1,1 –func "lambda u: sign(u) * abs(u)**3.0 * greater_equal(u, 0)" "lambda u: (sign((u+1)/2) * abs((u+1)/2)**3.0 * greater_equal(u, 0+.5))"

alpha=3 thresh=-1

python -m wbia.plottool plot_func –show –range=-1,1 –func "lambda u: sign(u) * abs(u)**$alpha * greater_equal(u, $thresh)" "lambda u: (sign(u) * abs(u)**$alpha * greater_equal(u, $thresh) + 1) / 2" "lambda u: sign((u+1)/2) * abs((u+1)/2)**$alpha * greater_equal(u, $thresh)"

python -m wbia.plottool plot_func –show –range=4,100 –func "lambda n: log2(n)""lambda n: log2(log2(n))""lambda n: log2(n)/log2(log2(n))""lambda n: log2(n) ** 2""lambda n: n"

python -m wbia.plottool plot_func –show –range=4,1000000 –func "lambda n: log2(n)""lambda n: n ** (1/3)"

python -m wbia.plottool plot_func –show –range=0,10 –func "lambda x: (3 * (x ** 2) - 18 * (x) - 81) / ((x ** 2) - 54) "

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import scipy
>>> import scipy.special  # NOQA
>>> func_list = ut.get_argval('--func', type_=list, default=['np.exp'])
>>> setup = ut.get_argval('--setup', type_=str, default=None)
>>> #funcs = [eval(f) for f in func_list]
>>> funcs = func_list
>>> start, stop = ut.get_argval('--range', type_=list, default=[-1, 1])
>>> start, stop = eval(str(start)), eval(str(stop))
>>> num = 1000
>>> result = plot_func(funcs, start, stop, num, setup=setup)
>>> print(result)
>>> import plottool as pt
>>> pt.quit_if_noshow()
>>> ylim = ut.get_argval('--ylim', type_=list, default=None)
>>> import wbia.plottool as pt
>>> None if ylim is None else plt.gca().set_ylim(*ylim)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**plot_hist**(*data*, *bins=None*, *nbins=10*, *weights=None*)

wbia.plottool.draw_func2.**plot_histpdf**(*data*, *label=None*, *draw_support=False*, *nbins=10*)

wbia.plottool.draw_func2.**plot_sift_signature**(*sift*, *title=''*, *fnum=None*, *pnum=None*)

Plots a SIFT descriptor as a histogram and distinguishes different bins into different colors

Parameters

- **sift** (*ndarray[dtype=np.uint8]*) –
- **title** (*str*) – (default = '')
- **fnum** (*int*) – figure number(default = None)
- **pnum** (*tuple*) – plot number(default = None)

Returns ax

Return type AxesSubplot

CommandLine: python -m wbia.plottool.draw_func2 –test-plot_sift_signature –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import vtool as vt
>>> sift = vt.demodata.testdata_dummy_sift(1, np.random.RandomState(0))[0]
>>> title = 'test sift histogram'
>>> fnum = None
>>> pnum = None
>>> ax = plot_sift_signature(sift, title, fnum, pnum)
>>> result = ('ax = %s' % (str(ax),))
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**plot_surface3d**(*xgrid*, *ygrid*, *zdata*, *xlabel=None*, *ylabel=None*, *zlabel=None*, *wire=False*, *mode=None*, *contour=False*, *dark=False*, *rstride=1*, *cstride=1*, *pnum=None*, *labelkw=None*, *xlabelkw=None*, *ylabelkw=None*, *zlabelkw=None*, *titlekw=None*, **args*, ***kwargs*)

References

http://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html

CommandLine: python -m wbia.plottool.draw_func2 –exec-plot_surface3d –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import wbia.plottool as pt
>>> import vtool as vt
>>> shape=(19, 19)
```

(continues on next page)

(continued from previous page)

```
>>> sigma1, sigma2 = 2.0, 1.0
>>> ybasis = np.arange(shape[0])
>>> xbasis = np.arange(shape[1])
>>> xgrid, ygrid = np.meshgrid(xbasis, ybasis)
>>> sigma = [sigma1, sigma2]
>>> gausspatch = vt.gaussian_patch(shape, sigma=sigma)
>>> title = 'ksize=%r, sigma=%r' % (shape, (sigma1, sigma2),)
>>> pt.plot_surface3d(xgrid, ygrid, gausspatch, rstride=1, cstride=1,
>>>                   cmap=mpl.cm.coolwarm, title=title)
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**pnum_generator**(*nRows=1*, *nCols=1*, *base=0*, *nSubplots=None*, *start=0*)

Parameters

- **nRows** (*int*) – (default = 1)
- **nCols** (*int*) – (default = 1)
- **base** (*int*) – (default = 0)
- **nSubplots** (*None*) – (default = None)

Yields *tuple* – pnum

CommandLine: python -m wbia.plottool.draw_func2 –exec-pnum_generator –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> nRows = 3
>>> nCols = 2
>>> base = 0
>>> pnum_ = pnum_generator(nRows, nCols, base)
>>> result = ut.repr2(list(pnum_), nl=1, nobr=True)
>>> print(result)
(3, 2, 1),
(3, 2, 2),
(3, 2, 3),
(3, 2, 4),
(3, 2, 5),
(3, 2, 6),
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> nRows = 3
>>> nCols = 2
>>> pnum_ = pnum_generator(nRows, nCols, start=3)
>>> result = ut.repr2(list(pnum_), nl=1, nobr=True)
>>> print(result)
(3, 2, 4),
(3, 2, 5),
(3, 2, 6),
```

wbia.plottool.draw_func2.**postsetup_axes**(*use_legend=True*, *bg=None*)

wbia.plottool.draw_func2.**presetup_axes**(*x_label='x'*, *y_label='y'*, *title_pref=''*, *title=None*, *equal_aspect=False*, *ax=None*, ***kwargs*)

wbia.plottool.draw_func2.**print_valid_cmaps**()

wbia.plottool.draw_func2.**quit_if_noshow**()

wbia.plottool.draw_func2.**relative_text**(*pos*, *text*, *ax=None*, *offset=None*, ***kwargs*)

Places text on axes in a relative position

Parameters

- **pos** (*tuple*) – relative xy position
- **text** (*str*) – text
- **ax** (*None*) – (default = None)
- **offset** (*None*) – (default = None)
- ****kwargs** – horizontalalignment, verticalalignment, roffset, ha, va, fontsize, fontproperties, fontproperties, clip_on

CommandLine: python -m wbia.plottool.draw_func2 relative_text –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import wbia.plottool as pt
>>> x = .5
>>> y = .5
>>> txt = 'Hello World'
>>> pt.figure()
>>> ax = pt.gca()
>>> family = 'monospace'
>>> family = 'CMU Typewriter Text'
>>> fontproperties = mpl.font_manager.FontProperties(family=family,
>>>                                                  size=42)
>>> result = relative_text((x, y), txt, ax, halign='center',
>>>                        fontproperties=fontproperties)
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**remove_patches**(*ax=None*)

deletes patches from axes

wbia.plottool.draw_func2.**render_figure_to_image**(*fig*, ***savekw*)

wbia.plottool.draw_func2.**reverse_colormap**(*cmap*)

References

http://nbviewer.ipython.org/github/kwinkunks/notebooks/blob/master/Matteo_colourmaps.ipynb

wbia.plottool.draw_func2.**rotate_plot**(*theta=0.7853981633974483*, *ax=None*)

Parameters

- **theta** –

- **ax** (*None*) –

CommandLine: python -m wbia.plottool.draw_func2 –test-rotate_plot

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> # build test data
>>> ax = gca()
>>> theta = TAU / 8
>>> plt.plot([1, 2, 3, 4, 5], [1, 2, 3, 2, 2])
>>> # execute function
>>> result = rotate_plot(theta, ax)
>>> # verify results
>>> print(result)
>>> show_if_requested()
```

wbia.plottool.draw_func2.**save_parts**(*fig*, *fpath*, *grouped_axes=None*, *dpi=None*)

FIXME: this works in mpl 2.0.0, but not 2.0.2

Parameters

- **fig** –
- **fpath** (*str*) – file path string
- **dpi** (*None*) – (default = None)

Returns subpaths

Return type list

CommandLine: python -m wbia.plottool.draw_func2 save_parts

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import wbia.plottool as pt
>>> import matplotlib as mpl
>>> import matplotlib.pyplot as plt
>>> def testimg(fname):
>>>     return plt.imread(mpl.cbook.get_sample_data(fname))
>>> fnames = ['grace_hopper.png', 'ada.png'] * 4
>>> fig = plt.figure(1)
>>> for c, fname in enumerate(fnames, start=1):
>>>     ax = fig.add_subplot(3, 4, c)
>>>     ax.imshow(testimg(fname))
>>>     ax.set_title(fname[0:3] + str(c))
>>>     ax.set_xticks([])
>>>     ax.set_yticks([])
>>> ax = fig.add_subplot(3, 1, 3)
>>> ax.plot(np.sin(np.linspace(0, np.pi * 2)))
>>> ax.set_xlabel('xlabel')
>>> ax.set_ylabel('ylabel')
>>> ax.set_title('title')
>>> fpath = 'test_save_parts.png'
```

(continues on next page)

(continued from previous page)

```
>>> adjust_subplots(fig=fig, wspace=.3, hspace=.3, top=.9)
>>> subpaths = save_parts(fig, fpath, dpi=300)
>>> fig.savefig(fpath)
>>> ut.startfile(subpaths[0])
>>> ut.startfile(fpath)
```

wbia.plottool.draw_func2.**scores_to_cmap**(*scores*, *colors=None*, *cmap_='hot'*)

wbia.plottool.draw_func2.**scores_to_color**(*score_list*, *cmap_='hot'*, *logscale=False*, *reverse_cmap=False*, *custom=False*, *val2_customcolor=None*, *score_range=None*, *cmap_range=(0.1, 0.9)*)

Other good colormaps are 'spectral', 'gist_rainbow', 'gist_ncar', 'Set1', 'Set2', 'Accent' # TODO: plasma

Parameters

- **score_list** (*list*) –
- **cmap** (*str*) – defaults to hot
- **logscale** (*bool*) –
- **cmap_range** (*tuple*) – restricts to only a portion of the cmap to avoid extremes

Returns <class '_ast.ListComp'>

SeeAlso: python -m wbia.plottool.color_funcs –test-show_all_colormaps –show –type "Perceptually Uniform Sequential"

CommandLine: python -m wbia.plottool.draw_func2 scores_to_color –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import wbia.plottool as pt
>>> ut.exec_funckw(pt.scores_to_color, globals())
>>> score_list = np.array([-1, -2, 1, 1, 2, 10])
>>> # score_list = np.array([0, .1, .11, .12, .13, .8])
>>> # score_list = np.linspace(0, 1, 100)
>>> cmap_ = 'plasma'
>>> colors = pt.scores_to_color(score_list, cmap_)
>>> import vtool as vt
>>> imgRGB = vt.atleast_nd(np.array(colors)[:, 0:3], 3, tofront=True)
>>> imgRGB = imgRGB.astype(np.float32)
>>> imgBGR = vt.convert_colorspace(imgRGB, 'BGR', 'RGB')
>>> pt.imshow(imgBGR)
>>> pt.show_if_requested()
```

Example

```
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> score_list = np.array([-1, -2, 1, 1, 2, 10])
>>> cmap_ = 'hot'
>>> logscale = False
>>> reverse_cmap = True
>>> custom = True
```

(continues on next page)

(continued from previous page)

```
>>> val2_customcolor  = {
...         -1: UNKNOWN_PURP,
...         -2: LIGHT_BLUE,
...     }
```

wbia.plottool.draw_func2.**set_axis_extent**(*extents*, *ax=None*)

Parameters **extents** – xmin, xmax, ymin, ymax

wbia.plottool.draw_func2.**set_axis_limit**(*xmin*, *xmax*, *ymin*, *ymax*, *ax=None*)

wbia.plottool.draw_func2.**set_figsize**(*w*, *h*, *dpi*)

wbia.plottool.draw_func2.**show_chipmatch2**(*rchip1*, *rchip2*, *kpts1=None*, *kpts2=None*, *fm=None*, *fs=None*, *fm_norm=None*, *title=None*, *vert=None*, *fnum=None*, *pnum=None*, *heatmap=False*, *modifysize=False*, *new_return=False*, *draw_fmatch=True*, *darken=None*, *H1=None*, *H2=None*, *sel_fm=[]*, *ax=None*, *heatmask=False*, *white_background=False*, ***kwargs*)

Draws two chips and the feature matches between them. feature matches kpts1 and kpts2 use the (x,y,a,c,d)

Parameters

- **rchip1** (*ndarray*) – rotated annotation 1 image data
- **rchip2** (*ndarray*) – rotated annotation 2 image data
- **kpts1** (*ndarray*) – keypoints for annotation 1 [x, y, a=1, c=0, d=1, theta=0]
- **kpts2** (*ndarray*) – keypoints for annotation 2 [x, y, a=1, c=0, d=1, theta=0]
- **fm** (*list*) – list of feature matches as tuples (qfx, dfx)
- **fs** (*list*) – list of feature scores
- **fm_norm** (*None*) – (default = None)
- **title** (*str*) – (default = None)
- **vert** (*None*) – (default = None)
- **fnum** (*int*) – figure number(default = None)
- **pnum** (*tuple*) – plot number(default = None)
- **heatmap** (*bool*) – (default = False)
- **modifysize** (*bool*) – (default = False)
- **new_return** (*bool*) – (default = False)
- **draw_fmatch** (*bool*) – (default = True)
- **darken** (*None*) – (default = None)
- **H1** (*None*) – (default = None)
- **H2** (*None*) – (default = None)
- **sel_fm** (*list*) – (default = [])
- **ax** (*None*) – (default = None)
- **heatmask** (*bool*) – (default = False)

- ****kwargs** – all_kpts, lbl1, lbl2, rect, colorbar_, draw_border, cmap, scale_factor1, scale_factor2, draw_pts, draw_ell, draw_lines, ell_alpha, colors

Returns (xywh1, xywh2, sf_tup)

Return type tuple

CommandLine: python -m wbia.plottool.draw_func2 show_chipmatch2 –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import wbia.plottool as pt
>>> import vtool as vt
>>> rchip1 = vt.imread(ut.grab_test_imgpath('easy1.png'))
>>> rchip2 = vt.imread(ut.grab_test_imgpath('easy2.png'))
>>> kpts1 = np.array([
>>>     [10,  10,   30,   0,    30,    0.  ],
>>>     [ 355.89,  142.95,   10.46,   -0.63,    8.59,    0.  ],
>>>     [ 356.35,  147.  ,    8.38,    1.08,   11.68,    0.  ],
>>>     [ 361.4 ,  150.64,    7.44,    3.45,   13.63,    0.  ]
>>> ], dtype=np.float64)
>>> kpts2 = np.array([
>>>     [ 10,   10,   30,   0,    30,    0.  ],
>>>     [ 376.98,   50.61,   11.91,   -2.9 ,    9.77,    0.  ],
>>>     [ 377.59,   54.89,    9.7 ,   -1.4 ,   13.72,    0.  ],
>>>     [ 382.8 ,   58.2 ,    7.87,   -0.31,   15.23,    0.  ]
>>> ], dtype=np.float64)
>>> fm = None
>>> fs = None
>>> H1 = np.array([
>>>     [ -4.68815126e-01,   7.80306795e-02,  -2.23674587e+01],
>>>     [  4.54394231e-02,  -7.67438835e-01,   5.92158624e+01],
>>>     [  2.12918867e-04,  -8.64851418e-05,  -6.21472492e-01]])
>>> H1 = None
>>> H2 = None
>>> #H_half = np.array([[.2, 0, 0], [0, .2, 0], [0, 0, 1]])
>>> #H1 = H_half
>>> #H2 = H_half
>>> kwargs = dict(H1=H1, H2=H2, fm=fm, draw_lines=True, draw_ell=True)
>>> kwargs.update(ell_linewidth=5, lw=10, line_alpha=[1, .3, .3, .3])
>>> result = show_chipmatch2(rchip1, rchip2, kpts1, kpts2, **kwargs)
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**show_histogram**(*data*, *bins=None*, ***kwargs*)

CommandLine: python -m wbia.plottool.draw_func2 –test-show_histogram –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> # build test data
>>> data = np.array([1, 24, 0, 0, 3, 4, 5, 9, 3, 0, 0, 0, 0, 2, 2, 2, 0, 0, 1, 1,␣
↪0, 0, 0, 3,])
>>> bins = None
```

(continues on next page)

(continued from previous page)

```
>>> # execute function
>>> result = show_histogram(data, bins)
>>> # verify results
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**show_if_requested**(*N=1*)
Used at the end of tests. Handles command line arguments for saving figures
Referencse: http://stackoverflow.com/questions/4325733/save-a-subplot-in-matplotlib

wbia.plottool.draw_func2.**show_kpts**(*kpts*, *fnum=None*, *pnum=None*, ***kwargs*)
Show keypoints in a new figure. Note: use draw_kpts2 to overlay keypoints on a existing figure.
Parameters **kpts** (*ndarray[float32_t, ndim=2]*) – keypoints

CommandLine: xdoctest -m ~/code/plottool/plottool/draw_func2.py show_kpts

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.draw_func2 import *  # NOQA
>>> import vtool as vt
>>> kpts = vt.demodata.get_dummy_kpts()
>>> result = show_kpts(kpts)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.draw_func2.**show_phantom_legend_labels**(*ax=None*, ***kwargs*)

wbia.plottool.draw_func2.**show_signature**(*sig*, ***kwargs*)

wbia.plottool.draw_func2.**show_was_requested**()
returns True if –show is specified on the commandline or you are in IPython (and presumably want some sort of interaction

wbia.plottool.draw_func2.**small_xticks**(*ax=None*)

wbia.plottool.draw_func2.**small_yticks**(*ax=None*)

wbia.plottool.draw_func2.**space_xticks**(*nTicks=9*, *spacing=16*, *ax=None*)

wbia.plottool.draw_func2.**space_yticks**(*nTicks=9*, *spacing=32*, *ax=None*)

wbia.plottool.draw_func2.**test_save**()
CommandLine: python -m wbia.plottool.draw_func2 test_save –show python -m wbia.plottool.draw_func2 test_save

wbia.plottool.draw_func2.**udpate_adjust_subplots**()
DEPRICATE

updates adjust_subplots based on command line

wbia.plottool.draw_func2.**unique_rows**(*arr*)

References

http://stackoverflow.com/questions/16970982/find-unique-rows-in-numpy-array

wbia.plottool.draw_func2.**update_figsize**()
updates figsize based on command line

wbia.plottool.draw_func2.**upperleft_text**(*txt*, *alpha=0.6*, *color=None*)

wbia.plottool.draw_func2.**upperright_text**(*txt*, *offset=None*, *alpha=0.6*)

wbia.plottool.draw_func2.**variation_trunctate**(*data*)

wbia.plottool.draw_func2.**width_from**(*num*, *pad=0.05*, *start=0*, *stop=1*)

1.11.12 wbia.plottool.draw_sv module

wbia.plottool.draw_sv.**get_blended_chip**(*chip1*, *chip2*, *M*)
warps chip1 into chip2 space

wbia.plottool.draw_sv.**show_sv**(*chip1*, *chip2*, *kpts1*, *kpts2*, *fm*, *homog_tup=None*, *aff_tup=None*, *mx=None*, *show_assign=True*, *show_lines=True*, *show_kpts=True*, *show_aff=None*, *fnum=1*, *refine_method=None*, ***kwargs*)
Visualizes spatial verification
CommandLine: python -m vtool.spatial_verification –test-spatially_verify_kpts –show

wbia.plottool.draw_sv.**show_sv_simple**(*chip1*, *chip2*, *kpts1*, *kpts2*, *fm*, *inliers*, *mx=None*, *fnum=1*, *vert=None*, ***kwargs*)
CommandLine: python -m wbia.plottool.draw_sv –test-show_sv_simple –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.draw_sv import *  # NOQA
>>> import vtool as vt
>>> kpts1, kpts2, fm, aff_inliers, chip1, chip2, xy_thresh_sqrd = vt.testdata_
↪matching_affine_inliers()
>>> inliers = aff_inliers
>>> mx = None
>>> fnum = 1
>>> vert = None  # ut.get_argval('--vert', type_=bool, default=None)
>>> result = show_sv_simple(chip1, chip2, kpts1, kpts2, fm, inliers, mx, fnum,␣
↪vert=vert)
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

1.11.13 wbia.plottool.fig_presenter module

wbia.plottool.fig_presenter.**all_figures_bring_to_front**()

wbia.plottool.fig_presenter.**all_figures_show**()

wbia.plottool.fig_presenter.**all_figures_tight_layout**()

wbia.plottool.fig_presenter.**all_figures_tile**(*max_rows=None*, *row_first=True*, *no_tile=False*, *monitor_num=None*, *percent_w=None*, *percent_h=None*, *hide_toolbar=True*)
Lays out all figures in a grid. if wh is a scalar, a golden ratio is used

wbia.plottool.fig_presenter.**bring_to_front**(*fig*)

wbia.plottool.fig_presenter.**close_all_figures**()

wbia.plottool.fig_presenter.**close_figure**(*fig*)

wbia.plottool.fig_presenter.**draw**()

wbia.plottool.fig_presenter.**get_all_figures**()

wbia.plottool.fig_presenter.**get_all_qt4_wins**()

wbia.plottool.fig_presenter.**get_all_windows**()
Returns all mpl figures and registered qt windows

wbia.plottool.fig_presenter.**get_figure_window**(*fig*)

wbia.plottool.fig_presenter.**get_geometry**(*fnum*)

wbia.plottool.fig_presenter.**get_main_win_base**()

wbia.plottool.fig_presenter.**iup**()

wbia.plottool.fig_presenter.**iupdate**()

wbia.plottool.fig_presenter.**present**(**args*, ***kwargs*)
basically calls show if not embeded.
Kwargs: max_rows, row_first, no_tile, monitor_num, percent_w, percent_h, hide_toolbar
CommandLine: python -m wbia.plottool.fig_presenter present

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.fig_presenter import *  # NOQA
>>> result = present()
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.fig_presenter.**register_qt4_win**(*win*)

wbia.plottool.fig_presenter.**reset**()

wbia.plottool.fig_presenter.**set_geometry**(*fnum*, *x*, *y*, *w*, *h*)

wbia.plottool.fig_presenter.**show**()

wbia.plottool.fig_presenter.**show_figure**(*fig*)

wbia.plottool.fig_presenter.**unregister_qt4_win**(*win*)

wbia.plottool.fig_presenter.**update**()

1.11.14 wbia.plottool.interact_annotations module

Interactive tool to draw mask on an image or image-like array.

Todo:

- need concept of subannotation
- need to take options on a right click of an annotation

- add support for arbitrary polygons back in .
- rename species_list to label_list or category_list
- Just use metadata instead of species / category / label

Need to incorporate parts into metadata

Notes

3. Change bounding box and update continuously to the original image the new ANNOTATIONs

2. Make new window and frames inside, double click to pull up normal window with editing start with just taking in 6 images and ANNOTATIONs

1. ANNOTATION ID number, then list of 4 tuples

python -m utool.util_inspect check_module_usage –pat=”interact_annotations.py”

References

Adapted from matplotlib/examples/event_handling/poly_editor.py Jan 9 2014: taken from: https://gist.github.com/tonysyu/3090704

CommandLine: python -m wbia.plottool.interact_annotations –test-test_interact_annots –show

class wbia.plottool.interact_annotations.**AnnotPoly**(*ax*, *num*, *verts*, *theta*, *species*, *fc=(0, 0, 0)*, *line_color=(1, 1, 1)*, *line_width=4*, *is_orig=False*, *metadata=None*, *valid_species=None*, *manager=None*)

Bases: matplotlib.patches.Polygon, utool.util_dev.NiceRepr

Helper to represent an annotation polygon wbia –aidcmd=’Interact image’ –aid=1

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.interact_annotations import *  # NOQA
>>> verts = vt.verts_from_bbox([0, 0, 10, 10])
>>> poly = AnnotPoly(None, 0, verts, 0, '____')
```

add_to_axis(*ax*)

axes_init(*ax*)

calc_handle_display_coords()

calc_tag_position()

CommandLine: python -m wbia.plottool.interact_annotations –test-calc_tag_position –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.interact_annotations import *  # NOQA
>>> poly = ut.DynStruct()
>>> poly.basecoords = vt.verts_from_bbox([0, 0, 400, 400], True)
>>> poly.theta = 0
>>> poly.xy = vt.verts_from_bbox([0, 0, 400, 400], True)
>>> tagpos = poly.calc_tag_position()
>>> print('tagpos = %r' % (tagpos,))
```

draw_self(*ax*, *show_species_tags=False*, *editable=True*)

get_poly_mask(*shape*)

increment_species(*amount=1*)

is_near_handle(*xy_pt*, *max_dist*)

move_poly(*dx*, *dy*, *ax*)

move_to_back()

print_info()

remove_from_axis(*ax*)

resize_poly(*x*, *y*, *idx*, *ax*)

Resize a rectangle using idx as the given anchor point. Respects current rotation.

CommandLine: python -m wbia.plottool.interact_annotations –exec-resize_poly –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.interact_annotations import *  # NOQA
>>> (h, w) = img.shape[0:2]
>>> x1, y1 = 10, 10
>>> x2, y2 = w - 10,  h - 10
>>> coords = ((x1, y1), (x1, y2), (x2, y2), (x2, y1))
>>> x = 3 * w / 4
>>> y = 3 * h / 4
>>> idx = 3
>>> resize_poly(poly, x, y, idx)
>>> update_UI()
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

rotate_poly(*dtheta*, *ax*)

set(*, *agg_filter=<UNSET>*, *alpha=<UNSET>*, *animated=<UNSET>*, *antialiased=<UNSET>*, *capstyle=<UNSET>*, *clip_box=<UNSET>*, *clip_on=<UNSET>*, *clip_path=<UNSET>*, *closed=<UNSET>*, *color=<UNSET>*, *edgecolor=<UNSET>*, *facecolor=<UNSET>*, *fill=<UNSET>*, *gid=<UNSET>*, *hatch=<UNSET>*, *in_layout=<UNSET>*, *joinstyle=<UNSET>*, *label=<UNSET>*, *linestyle=<UNSET>*, *linewidth=<UNSET>*, *path_effects=<UNSET>*, *picker=<UNSET>*, *rasterized=<UNSET>*, *sketch_params=<UNSET>*, *snap=<UNSET>*, *species=<UNSET>*, *transform=<UNSET>*, *url=<UNSET>*, *visible=<UNSET>*, *xy=<UNSET>*, *zorder=<UNSET>*)

Set multiple properties at once.

Supported properties are

Properties: agg_filter: a filter function, which takes a (m, n, 3) float array and a dpi value, and returns a (m, n, 3) array alpha: scalar or None animated: bool antialiased or aa: bool or None capstyle: *.CapStyle* or {'butt', 'projecting', 'round'} clip_box: *.Bbox* clip_on: bool clip_path: Patch or (Path, Transform) or None closed: bool color: color edgecolor or ec: color or None facecolor or fc: color or None figure: *.Figure* fill: bool gid: str hatch: {'/', '', 'l', '-', '+', 'x', 'o', 'O', '.', '*'} in_layout: bool joinstyle: *.JoinStyle* or {'miter', 'round', 'bevel'} label: object linestyle or ls: {'-', '–', '-.', ':', '', (offset, on-off-seq), ...} linewidth or lw: float or None path_effects: *.AbstractPathEffect* picker: None or bool or float or callable rasterized: bool sketch_params: (scale: float, length: float, randomness: float) snap: bool or None species: unknown transform: *.Transform* url: str visible: bool xy: (N, 2) array-like zorder: float

set_species(*text*)

size

update_color(*selected=False*, *editing_parts=False*)

update_display_coords()

update_lines()

class wbia.plottool.interact_annotations.**AnnotationInteraction**(*img*, *img_ind=None*, *commit_callback=None*, *verts_list=None*, *bbox_list=None*, *theta_list=None*, *species_list=None*, *metadata_list=None*, *line_width=4*, *line_color=(1, 1, 1)*, *face_color=(0, 0, 0)*, *fnum=None*, *default_species='____'*, *next_callback=None*, *prev_callback=None*, *do_mask=False*, *valid_species=[]*, ***kwargs*)

Bases: *wbia.plottool.abstract_interaction.AbstractInteraction*

An interactive polygon editor.

SeeAlso: wbia.viz.interact.interact_annotations2 (ensure that any updates here are propogated there)

Parameters **verts_list** (*list*) – list of lists of (float, float) List of (x, y) coordinates used as vertices of the polygon.

add_action_buttons()

add_new_poly(*event=None*, *full=False*)
Adds a new annotation to the image

connect_mpl_callbacks(*canvas*)
disconnects matplotlib callbacks specified in the self.mpl_callback_ids dict

delete_current_poly(*event=None*)
Removes an annotation

disconnect_mpl_callbacks(*canvas*)
disconnects all connected matplotlib callbacks

draw_artists()

draw_callback(*event*)

edit_poly_parts(*poly*)

editable_polys

get_most_recently_added_poly()

get_poly_under_cursor(*x*, *y*)
get the index of the vertex under cursor if within max_dist tolerance

handle_polygon_creation(*bbox_list*, *theta_list*, *species_list*, *metadata_list*)
Maintain original input

in_edit_parts_mode

is_poly_pickable(*artist*, *event*)

new_polygon(*verts*, *theta*, *species*, *fc=(0, 0, 0)*, *line_color=(1, 1, 1)*, *line_width=4*, *is_orig=False*, *metadata=None*)
verts - list of (x, y) tuples

next_image(*event*)

on_click(*event*)
python -m wbia.viz.interact.interact_annotations2 –test-ishow_image2 –show

on_click_release(*event*)

on_figure_leave(*event*)

on_key_press(*event*)

on_motion(*event*)

on_pick(*event*)
Makes selected polygon translucent

prev_image(*event*)

reinitialize_figure(*fnum=None*)

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

save_and_exit(*event*, *do_close=True*)
The Save and Exit Button

write a callback to redraw viz for bbox_list

show()

start()

toggle_species_label()

uneditable_polys

update_UI()

update_callbacks(*next_callback*, *prev_callback*)

update_image_and_callbacks(*img*, *bbox_list*, *theta_list*, *species_list*, *metadata_list*, *next_callback*, *prev_callback*)

wbia.plottool.interact_annotations.**apply_mask**(*img*, *mask*)

wbia.plottool.interact_annotations.**apply_polarDelta**(*poldelt*, *cart*)

wbia.plottool.interact_annotations.**calc_display_coords**(*oldcoords*, *theta*)

wbia.plottool.interact_annotations.**check_dims**(*ax*, *xy_pt*, *margin=0.5*)
checks if bounding box dims are ok

Allow the bounding box to go off the image so orientations can be done correctly

wbia.plottool.interact_annotations.**check_min_wh**(*coords*)
Depends on hardcoded indices, which is inelegant, but we're already depending on those for the FUDGE_FACTORS array above 0—-1 | | 3—-2

wbia.plottool.interact_annotations.**check_valid_coords**(*ax*, *coords_list*)

wbia.plottool.interact_annotations.**default_vertices**(*img*, *polys=None*, *mouseX=None*, *mouseY=None*)
Default to rectangle that has a quarter-width/height border.

wbia.plottool.interact_annotations.**enforce_dims**(*ax*, *xy_pt*, *margin=0.5*)
ONLY USE THIS ON UNROTATED RECTANGLES, as to do otherwise may yield arbitrary polygons

wbia.plottool.interact_annotations.**is_within_distance_from_line**(*pt*, *line*, *max_dist*)

wbia.plottool.interact_annotations.**points_center**(*pts*)

wbia.plottool.interact_annotations.**polarDelta**(*p1*, *p2*)

wbia.plottool.interact_annotations.**pretty_hotkey_map**(*hotkeys*)

wbia.plottool.interact_annotations.**rotate_points_around**(*points*, *theta*, *ax*, *ay*)

References

http://www.euclideanspace.com/maths/geometry/affine/aroundPoint/matrix2d/

wbia.plottool.interact_annotations.**test_interact_annots**()
CommandLine: python -m wbia.plottool.interact_annotations –test-test_interact_annots –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.interact_annotations import *  # NOQA
>>> import wbia.plottool as pt
>>> # build test data
>>> # execute function
>>> self = test_interact_annots()
>>> # verify results
>>> print(self)
>>> pt.show_if_requested()
```

1.11.15 wbia.plottool.interact_helpers module

wbia.plottool.interact_helpers.**begin_interaction**(*type_*, *fnum*)

wbia.plottool.interact_helpers.**clicked_inside_axis**(*event*)

wbia.plottool.interact_helpers.**clicked_outside_axis**(*event*)

wbia.plottool.interact_helpers.**connect_callback**(*fig*, *callback_type*, *callback_fn*)
wrapper around fig.canvas.mpl_connect

References

http://matplotlib.org/users/event_handling.html button_press_event button_release_event draw_event key_press_event key_release_event motion_notify_event pick_event resize_event scroll_event figure_enter_event figure_leave_event axes_enter_event axes_leave_event

wbia.plottool.interact_helpers.**detect_keypress**(*fig*)

wbia.plottool.interact_helpers.**disconnect_callback**(*fig*, *callback_type*, ***kwargs*)

1.11.16 wbia.plottool.interact_impaint module

helpers for painting on top of images for groundtruthing

References

http://stackoverflow.com/questions/22232812/drawing-on-image-with-matplotlib-and-opencv-update-image http://stackoverflow.com/questions/34933254/force-matplotlib-to-block-in-a-pyqt-thread-process http://matplotlib.org/examples/user_interfaces/embedding_in_qt4.html http://stackoverflow.com/questions/22410663/block-qmainwindow-while-child-widget-is-alive-pyqt http://stackoverflow.com/questions/20289939/pause-execution-until-button-press

class wbia.plottool.interact_impaint.**PaintInteraction**(*img*, ***kwargs*)
Bases: *wbia.plottool.abstract_interaction.AbstractInteraction*

References

http://stackoverflow.com/questions/22232812/drawing-on-image-with-mpl
CommandLine: python -m wbia.plottool.interact_impaint –exec-draw_demo –show
apply_stroke(*x*, *y*, *color*)

do_blit()

on_click_inside(*event*, *ax*)

on_close(*event=None*)

on_drag_inside(*event*)

on_drag_stop(*event*)

on_draw(*event*)

on_key_press(*event*)

on_scroll(*event*)

static_plot(*fnum=None*, *pnum=(1, 1, 1)*)

update_image()

update_title()

wbia.plottool.interact_impaint.**draw_demo**()
CommandLine: python -m wbia.plottool.interact_impaint –exec-draw_demo –show

Example

```
>>> # SCRIPT
>>> from wbia.plottool.interact_impaint import *  # NOQA
>>> result = draw_demo()
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.interact_impaint.**impaint_mask2**(*img*, *init_mask=None*)
python -m wbia.plottool.interact_impaint –exec-draw_demo –show

1.11.17 wbia.plottool.interact_keypoints module

class wbia.plottool.interact_keypoints.**KeypointInteraction**(*chip*, *kpts*, *vecs*, *fnum=0*, *figtitle=None*, ***kwargs*)
Bases: *wbia.plottool.abstract_interaction.AbstractInteraction*
CommandLine: python -m wbia.plottool.interact_keypoints –exec-KeypointInteraction –show python -m wbia.plottool.interact_keypoints –exec-KeypointInteraction –show –fname=lena.png

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.interact_keypoints import *  # NOQA
>>> import numpy as np
>>> import wbia.plottool as pt
>>> import utool as ut
>>> import pyhesaff
>>> import vtool as vt
>>> kpts, vecs, imgBGR = pt.viz_keypoints.testdata_kpts()
>>> ut.quit_if_noshow()
>>> #pt.interact_keypoints.ishow_keypoints(imgBGR, kpts, vecs, ori=True, ell_
↪alpha=.4, color='distinct')
>>> pt.interact_keypoints.KeypointInteraction(imgBGR, kpts, vecs, ori=True, ell_
↪alpha=.4, autostart=True)
>>> pt.show_if_requested()
```

on_click_inside(*event*, *ax*)

on_click_outside(*event*)

plot(*fnum=None*, *pnum=(1, 1, 1)*, ***kwargs*)

wbia.plottool.interact_keypoints.**ishow_keypoints**(*chip*, *kpts*, *desc*, *fnum=0*, *figtitle=None*, *nodraw=False*, ***kwargs*)
TODO: Depricate in favor of the class

CommandLine: python -m wbia.plottool.interact_keypoints –test-ishow_keypoints –show python -m wbia.plottool.interact_keypoints –test-ishow_keypoints –show –fname zebra.png

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.interact_keypoints import *  # NOQA
>>> import numpy as np
>>> import wbia.plottool as pt
>>> import utool as ut
>>> import pyhesaff
>>> import vtool as vt
>>> kpts, vecs, imgBGR = pt.viz_keypoints.testdata_kpts()
>>> ut.quit_if_noshow()
>>> #pt.interact_keypoints.ishow_keypoints(imgBGR, kpts, vecs, ori=True, ell_
↪alpha=.4, color='distinct')
>>> pt.interact_keypoints.ishow_keypoints(imgBGR, kpts, vecs, ori=True, ell_
↪alpha=.4)
>>> pt.show_if_requested()
```

1.11.18 wbia.plottool.interact_matches module

Unfinished non-wbia dependent version of interact matches

class wbia.plottool.interact_matches.**MatchInteraction2**(*rchip1*, *rchip2*, *kpts1*, *kpts2*, *fm*, *fs*, *fsv*, *vecs1*, *vecs2*, *H1=None*, *H2=None*, *fnum=None*, ***kwargs*)

Bases: *wbia.plottool.abstract_interaction.AbstractInteraction*

TODO: replace functional version with this class

Plots a chip result and sets up callbacks for interaction.
SeeAlso: wbia.viz.interact.interact_matches.MatchInteraction
CommandLine: python -m wbia.plottool.interact_matches –test-MatchInteraction2 –show

Example

```
>>> # xdoctest: +REQUIRES(module:wbia, --slow)
>>> from wbia.plottool.interact_matches import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb('testdb1')
>>> qreq_ = ibs.new_query_request([1], [2, 3, 4, 5], cfgdict=dict(query_rotation_
↪heuristic=True))
>>> cm = qreq_.execute()[0]
>>> qaid = cm.qaid
>>> daid = cm.get_top_aids()[0]
>>> rchip1 = ibs.get_annot_chips([qaid], config2_=qreq_.extern_query_config2)[0]
>>> rchip2 = ibs.get_annot_chips([daid], config2_=qreq_.extern_data_config2)[0]
>>> kpts1 = ibs.get_annot_kpts([qaid], config2_=qreq_.extern_query_config2)[0]
>>> kpts2 = ibs.get_annot_kpts([daid], config2_=qreq_.extern_data_config2)[0]
>>> vecs1 = ibs.get_annot_vecs([qaid], config2_=qreq_.extern_query_config2)[0]
>>> vecs2 = ibs.get_annot_vecs([daid], config2_=qreq_.extern_data_config2)[0]
```

(continues on next page)

(continued from previous page)

```
>>> fm = cm.aid2_fm[daid]
>>> fs = cm.aid2_fs[daid]
>>> fsv = cm.aid2_fsv[daid]
>>> H1 = cm.aid2_H[daid]
>>> self = MatchInteraction2(rchip1, rchip2, kpts1, kpts2, fm, fs, fsv,
>>>                          vecs1, vecs2, H1)
>>> self.show_page()
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

chipmatch_view(*fnum=None*, *pnum=(1, 1, 1)*, *verbose=None*, ***kwargs_*)
just visualizes the matches using some type of lines

get_popup_options()

on_click_inside(*event*, *ax*)

on_click_outside(*event*)

plot(**args*, ***kwargs*)

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

select_ith_match(*mx*)
Selects the ith match and visualizes and prints information concerning features weights, keypoint details, and sift descriptions

wbia.plottool.interact_matches.**show_keypoint_gradient_orientations**(*ibs*, *rchip*, *kp*, *vec*, *fnum=None*, *pnum=None*, *config2_=None*)

1.11.19 wbia.plottool.interact_multi_image module

class wbia.plottool.interact_multi_image.**MultiImageInteraction**(*gpath_list*, *nPerPage=4*, *bboxes_list=None*, *thetas_list=None*, *verts_list=None*, *gid_list=None*, *nImgs=None*, *fnum=None*, *context_option_funcs=None*, *xlabel_list=None*, *vizkw=None*, ***kwargs*)
Bases: *wbia.plottool.abstract_interaction.AbstractInteraction*
CommandLine: python -m wbia.plottool.interact_multi_image –exec-MultiImageInteraction –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.interact_multi_image import *  # NOQA
>>> import utool as ut
>>> TEST_IMAGES_URL = 'https://wildbookiarepository.azureedge.net/data/testdata.
↪zip'
>>> test_image_dir = ut.grab_zipped_url(TEST_IMAGES_URL, appname='utool')
>>> # test image paths
>>> imgpaths       = ut.list_images(test_image_dir, fullpath=True,␣
↪recursive=False)
>>> bboxes_list = [[]] * len(imgpaths)
>>> #bboxes_list[0] = [(-200, -100, 400, 400)]
>>> bboxes_list[0] = [(20, 10, 400, 400)]
>>> iteract_obj = MultiImageInteraction(imgpaths, nPerPage=4,
>>>                                     bboxes_list=bboxes_list)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

dump_to_disk(*dpath*, *num=None*, *prefix='temp_img'*)

make_hud()
Creates heads up display

next_page(*event*)

on_click_inside(*event*, *ax*)

on_key_press(*event*)

plot_image(*index*)

prepare_page(*pagenum*)
Gets indexes for the pagenum ready to be displayed

prev_page(*event*)

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

show_page(*pagenum=None*)
Displays a page of matches

update_images(*img_ind*, *updated_bbox_list*, *updated_theta_list*, *changed_annottups*, *new_annottups*)
Insert code for viz_image2 redrawing here

1.11.20 wbia.plottool.interactions module

class wbia.plottool.interactions.**ExpandableInteraction**(*fnum=None*, *_pnumiter=None*, *interactive=None*, ***kwargs*)
Bases: *wbia.plottool.abstract_interaction.AbstractInteraction*

Append a list of functions that draw plots and this interaction will plot them in appropriate subplots and let you click on them to zoom in.

Parameters

- **fnum** (*int*) – figure number(default = None)
- **_pnumiter** (*None*) – (default = None)

- **interactive** (*None*) – (default = None)
- ****kwargs** – nRows, nCols

CommandLine: python -m wbia.plottool.interactions –exec-ExpandableInteraction –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.interactions import *  # NOQA
>>> import numpy as np
>>> import wbia.plottool as pt
>>> inter = pt.interactions.ExpandableInteraction()
>>> inter.append_plot(ut.partial(pt.plot_func, np.sin, stop=np.pi * 2))
>>> inter.append_plot(ut.partial(pt.plot_func, np.cos, stop=np.pi * 2))
>>> inter.append_plot(ut.partial(pt.plot_func, np.tan, stop=np.pi * 2))
>>> inter.start()
>>> pt.show_if_requested()
```

append_partial (*func*, **args*, ***kwargs*)
Register a plotting function with default arguments

Parameters

- **func** (*callable*) – plotting function (does NOT need fnum/pnum).
- ***args** – args to be passed to func
- ****kwargs** – kwargs to be passed to func

append_plot (*func*, *pnum=None*, *ishow_func=None*, *px=None*)
Register a plotting function

Parameters

- **func** (*callable*) – must take fnum and pnum as keyword arguments.
- **pnum** (*tuple*) – plot num / gridspec. Defaults based on append order
- **ishow_func** (*callable*) – an interactive version of func
- **px** (*int*) – None or a plot index into (nRows, nCols)

on_click (*event*)

show_page ()
Hack: this function should probably not be defined, but it is for convinience of a developer. Override this or create static plot function (preferably override)

class wbia.plottool.interactions.**PanEvents** (*ax=None*)
Bases: object

pan_on_motion (*event*)

pan_on_press (*event*)

pan_on_release (*event*)

wbia.plottool.interactions.**check_if_subinteract** (*func*)

wbia.plottool.interactions.**pan_factory** (*ax=None*)

wbia.plottool.interactions.**zoom_factory** (*ax=None*, *zoomable_list=[]*, *base_scale=1.1*)

References

https://gist.github.com/tacaswell/3144287 http://stackoverflow.com/questions/11551049/matplotlib-plot-zooming-with-scroll-wheel

1.11.21 wbia.plottool.mpl_keypoint module

class wbia.plottool.mpl_keypoint.**HomographyTransform**(*H*, *axis=None*, *use_rmin=True*)

Bases: matplotlib.transforms.Transform

References

http://stackoverflow.com/questions/28401788/using-homogeneous-transforms-non-affine-with-matplotlib-patches?noredirect=1#comment45156353_28401788 http://matplotlib.org/users/transforms_tutorial.html

input_dims = 2

is_separable = False

output_dims = 2

transform_non_affine(*input_xy*)

The input and output are Nx2 numpy arrays.

transform_path_non_affine(*path*)

Apply the non-affine part of this transform to *.Path path*, returning a new *.Path*.

`transform_path(path)` is equivalent to `transform_path_affine(transform_path_non_affine(valu`

wbia.plottool.mpl_keypoint.**draw_keypoints**(*ax*, *kpts_*, *scale_factor=1.0*, *offset=(0.0*, *0.0)*, *rotation=0.0*, *ell=True*, *pts=False*, *rect=False*, *eig=False*, *ori=False*, *sifts=None*, *siftkw={}*, *H=None*, ***kwargs*)

draws keypoints extracted by pyhesaff onto a matplotlib axis

FIXME: There is probably a matplotlib bug here. If you specify two different alphas in a collection, whatever the last alpha was gets applied to everything

Parameters

- **ax** (*mpl.Axes*) –
- **kpts** (*ndarray*) – keypoints [[x, y, a, c, d, theta], ...]
- **scale_factor** (*float*) –
- **offset** (*tuple*) –
- **rotation** (*float*) –
- **ell** (*bool*) –
- **pts** (*bool*) –
- **rect** (*bool*) –
- **eig** (*bool*) –
- **ori** (*bool*) –
- **sifts** (*None*) –

References

http://stackoverflow.com/questions/28401788/transforms-non-affine-patch
CommandLine: python -m wbia.plottool.mpl_keypoint draw_keypoints –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.mpl_keypoint import *  # NOQA
>>> from wbia.plottool.mpl_keypoint import _draw_patches, _draw_pts  # NOQA
>>> import wbia.plottool as pt
>>> import vtool as vt
>>> imgBGR = vt.get_star_patch(jitter=True)
>>> H = np.array([[1, 0, 0], [.5, 2, 0], [0, 0, 1]])
>>> H = np.array([[.8, 0, 0], [0, .8, 0], [0, 0, 1]])
>>> H = None
>>> TAU = 2 * np.pi
>>> kpts_ = vt.make_test_image_keypoints(imgBGR, scale=.5, skew=2, theta=TAU / 8.
↪0)
>>> scale_factor=1.0
>>> #offset=(0.0, -4.0)
>>> offset=(0.0, 0.0)
>>> rotation=0.0
>>> ell=True
>>> pts=True
>>> rect=True
>>> eig=True
>>> ori=True
>>> # make random sifts
>>> sifts = mpl_sift.testdata_sifts()
>>> siftkw = {}
>>> kwargs = dict(ori_color=[0, 1, 0], rect_color=[0, 0, 1],
>>>               eig_color=[1, 1, 0], pts_size=.1)
>>> w, h = imgBGR.shape[0:2][::-1]
>>> imgBGR_ = imgBGR if H is None else vt.warpAffine(
>>>     imgBGR, H, (int(w * .8), int(h * .8)))
>>> fig, ax = pt.imshow(imgBGR_ * 255)
>>> draw_keypoints(ax, kpts_, scale_factor, offset, rotation, ell, pts,
...                rect, eig, ori, sifts, siftkw, H=H, **kwargs)
>>> pt.iup()
>>> pt.show_if_requested()
```

wbia.plottool.mpl_keypoint.**eigenvector_actors**(*invVR_aff2Ds*)

wbia.plottool.mpl_keypoint.**ellipse_actors**(*invVR_aff2Ds*)

wbia.plottool.mpl_keypoint.**get_invVR_aff2Ds**(*kpts*, *H=None*)
Returns matplotlib keypoint transformations (circle -> ellipse)

Example

```
>>> # Test CV2 ellipse vs mine using MSER
>>> import vtool as vt
>>> import cv2
>>> import wbia.plottool as pt
```

(continues on next page)

(continued from previous page)

```
>>> img_fpath = ut.grab_test_imgpath(ut.get_argval('--fname', default='zebra.png
↪'))
>>> imgBGR = vt.imread(img_fpath)
>>> imgGray = cv2.cvtColor(imgBGR, cv2.COLOR_BGR2GRAY)
>>> mser = cv2.MSER_create()
>>> regions, bboxs = mser.detectRegions(imgGray)
>>> region = regions[0]
>>> bbox = bboxs[0]
>>> vis = imgBGR.copy()
>>> vis[region.T[1], region.T[0], :] = 0
>>> hull = cv2.convexHull(region.reshape(-1, 1, 2))
>>> cv2.polylines(vis, [hull], 1, (0, 255, 0))
>>> ell = cv2.fitEllipse(region)
>>> cv2.ellipse(vis, ell, (255))
>>> ((cx, cy), (rx, ry), degrees) = ell
>>> # Convert diameter to radians
>>> rx /= 2
>>> ry /= 2
>>> # Make my version of ell
>>> theta = np.radians(degrees)  # opencv lives in radians
>>> S = vt.scale_mat3x3(rx, ry)
>>> T = vt.translation_mat3x3(cx, cy)
>>> R = vt.rotation_mat3x3(theta)
>>> #R = np.eye(3)
>>> invVR = T.dot(R).dot(S)
>>> kpts = vt.flatten_invV_mats_to_kpts(np.array([invVR]))
>>> pt.imshow(vis)
>>> # MINE IS MUCH LARGER (by factor of 2)) WHY?
>>> # we start out with a unit circle not a half circle
>>> pt.draw_keypoints(pt.gca(), kpts, pts=True, ori=True, eig=True, rect=True)
```

wbia.plottool.mpl_keypoint.**orientation_actors**(*kpts*, *H=None*)
creates orientation actors w.r.t. the gravity vector

wbia.plottool.mpl_keypoint.**pass_props**(*dict1*, *dict2*, **args*)

wbia.plottool.mpl_keypoint.**rectangle_actors**(*invVR_aff2Ds*)

1.11.22 wbia.plottool.mpl_sift module

wbia.plottool.mpl_sift.**draw_sift_on_patch**(*patch*, *sift*, ***kwargs*)

wbia.plottool.mpl_sift.**draw_sifts**(*ax*, *sifts*, *invVR_aff2Ds=None*, ***kwargs*)
Gets sift patch collections, transforms them and then draws them.
CommandLine: python -m wbia.plottool.mpl_sift –test-draw_sifts –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.plottool.mpl_sift import *  # NOQA
>>> # build test data
>>> import wbia.plottool as pt
>>> pt.figure(1)
>>> ax = pt.gca()
>>> ax.set_xlim(-1.1, 1.1)
```

(continues on next page)

(continued from previous page)

```
>>> ax.set_ylim(-1.1, 1.1)
>>> sifts = testdata_sifts()
>>> sifts[:, 0:8] = 0
>>> invVR_aff2Ds = None
>>> kwargs = dict(multicolored_arms=False)
>>> kwargs['arm1_lw'] = 3
>>> kwargs['stroke'] = 5
>>> result = draw_sifts(ax, sifts, invVR_aff2Ds, **kwargs)
>>> ax.set_aspect('equal')
>>> print(result)
>>> pt.show_if_requested()
```

wbia.plottool.mpl_sift.**get_sift_collection**(*sift*, *aff=None*, *bin_color=array([0., 0., 0., 1.])*, *arm1_color=array([1., 0., 0., 1.])*, *arm2_color=array([0., 0., 0., 1.])*, *arm_alpha=1.0*, *arm1_lw=1.0*, *arm2_lw=2.0*, *stroke=1.0*, *circ_alpha=0.5*, *fidelity=256*, *scaling=True*, ***kwargs*)

Creates a collection of SIFT matplotlib patches

get_sift_collection

Parameters

- **sift** –
- **aff** (*None*) –
- **bin_color** (*ndarray*) –
- **arm1_color** (*ndarray*) –
- **arm2_color** (*ndarray*) –
- **arm_alpha** (*float*) –
- **arm1_lw** (*float*) –
- **arm2_lw** (*float*) –
- **circ_alpha** (*float*) –
- **fidelity** (*int*) – quantization factor

Returns coll_tup

Return type

?

CommandLine: python -m wbia.plottool.mpl_sift –test-get_sift_collection

Example

```
>>> from wbia.plottool.mpl_sift import *  # NOQA
>>> sift = testdata_sifts()[0]
>>> aff = None
>>> bin_color = np.array([ 0.,  0.,  0.,  1.])
>>> arm1_color = np.array([ 1.,  0.,  0.,  1.])
>>> arm2_color = np.array([ 0.,  0.,  0.,  1.])
>>> arm_alpha = 1.0
```

(continues on next page)

(continued from previous page)

```
>>> arm1_lw = 0.5
>>> arm2_lw = 1.0
>>> circ_alpha = 0.5
>>> coll_tup = get_sift_collection(sift, aff, bin_color, arm1_color,
>>>                                arm2_color, arm_alpha, arm1_lw,
>>>                                arm2_lw, circ_alpha)
>>> print(coll_tup)
```

wbia.plottool.mpl_sift.**render_sift_on_patch**(*patch*, *sift*)

wbia.plottool.mpl_sift.**testdata_sifts**()

1.11.23 wbia.plottool.nx_helpers module

Helpers for graph plotting

References

http://www.graphviz.org/content/attrs http://www.graphviz.org/doc/info/attrs.html

Ignore: http://www.graphviz.org/pub/graphviz/stable/windows/graphviz-2.38.msi pip uninstall pydot pip uninstall pyparsing pip install -Iv https://pypi.python.org/packages/source/p/pyparsing/pyparsing-1.5.7.tar.gz#md5=9be0fcdcc595199c646ab317c1d9a709 pip install pydot sudo apt-get install libgraphviz4 libgraphviz-dev -y sudo apt-get install libgraphviz-dev pip install pygraphviz sudo pip3 install pygraphviz

> –install-option="–include-path=/usr/include/graphviz" –install-option="–library-path=/usr/lib/graphviz/"

python -c "import pygraphviz; print(pygraphviz.__file__)" python3 -c "import pygraphviz; print(pygraphviz.__file__)"

class wbia.plottool.nx_helpers.**GRAPHVIZ_KEYS**

Bases: object

E = {'URL', 'arrowhead', 'arrowsize', 'arrowtail', 'color', 'colorscheme', 'comment',

G = {'Damping', 'K', 'URL', '_background', 'bb', 'bgcolor', 'center', 'charset', 'clus

N = {'URL', 'area', 'color', 'colorscheme', 'comment', 'distortion', 'fillcolor', 'fix

class wbia.plottool.nx_helpers.**GraphVizLayoutConfig**(***kwargs*)

Bases: *wbia.dtool.base.Config*

Ignore:

Node Props:

colorscheme CEGN string NaN

fontcolor CEGN color NaN fontname CEGN string NaN fontsize CEGN double NaN

label CEGN lblString NaN

nojustify CEGN bool NaN style CEGN style NaN color CEN colorcolorList NaN

fillcolor CEN colorcolorList NaN

layer CEN layerRange NaN

penwidth CEN double NaN

radientangle CGN int NaN

labelloc CGN string NaN

margin CGN doublepoint NaN sortv CGN int NaN

peripheries CN int NaN

showboxes EGN int dot only

comment EGN string NaN

pos EN pointsplineType NaN

xlabel EN lblString NaN

ordering GN string dot only

group N string dot only pin N bool fdp | neato only

distortion N double NaN

fixedsize N boolstring NaN

height N double NaN image N string NaN

imagescale N boolstring NaN

orientation N double NaN regular N bool NaN

samplepoints N int NaN

shape N shape NaN

shapefile N string NaN

sides N int NaN skew N double NaN

width N double NaN z N double NaN

static get_param_info_list()

wbia.plottool.nx_helpers.**apply_graph_layout_attrs**(*graph*, *layout_info*)

wbia.plottool.nx_helpers.**draw_network2**(*graph*, *layout_info*, *ax*, *as_directed=None*, *hacknoedge=False*, *hacknode=False*, *verbose=None*, ***kwargs*)

Kwargs: use_image, arrow_width, fontsize, fontweight, fontname, fontfamilty, fontproperties
fancy way to draw networkx graphs without directly using networkx

python -m wbia.annotmatch_funcs review_tagged_joins –dpath ~/latex/crall-candidacy-2015/ –save figures4/mergecase.png –figsize=15,15 –clipwhite –diskshow # python -m dtool –tf DependencyCache.make_graph –show

wbia.plottool.nx_helpers.**dump_nx_ondisk**(*graph*, *fpath*)

wbia.plottool.nx_helpers.**ensure_nonhex_color**(*orig_color*)

wbia.plottool.nx_helpers.**format_anode_pos**(*xy*, *pin=True*)

wbia.plottool.nx_helpers.**get_explicit_graph**(*graph*)

Parameters **graph** (*nx.Graph*) –

wbia.plottool.nx_helpers.**get_nx_layout**(*graph*, *layout*, *layoutkw=None*, *verbose=None*)

wbia.plottool.nx_helpers.**make_agraph**(*graph_*)

wbia.plottool.nx_helpers.**netx_draw_images_at_positions**(*img_list*, *pos_list*, *size_list*, *color_list*, *framewidth_list*)

Overlays images on a networkx graph

References

https://gist.github.com/shobhit/3236373 http://matplotlib.org/examples/pylab_examples/demo_annotation_box.html http://stackoverflow.com/questions/11487797/mpl-overlay-small-image http://matplotlib.org/api/text_api.html http://matplotlib.org/api/offsetbox_api.html

wbia.plottool.nx_helpers.**nx_agraph_layout**(*orig_graph*, *inplace=False*, *verbose=None*, *return_agraph=False*, *groupby=None*, ***layoutkw*)

Uses graphviz and custom code to determine position attributes of nodes and edges.

Parameters **groupby** (*str*) – if not None then nodes will be grouped by this attributes and groups will be layed out separately and then stacked together in a grid

Ignore: orig_graph = graph graph = layout_graph

References

http://www.graphviz.org/content/attrs http://www.graphviz.org/doc/info/attrs.html

CommandLine: python -m wbia.plottool.nx_helpers nx_agraph_layout –show

Doctest:

```
>>> # FIXME failing-test (22-Jul-2020) This test is failing and it's not␣
↪clear how to fix it
>>> # xdoctest: +SKIP
>>> # xdoctest: +REQUIRES(module:pygraphviz)
>>> from wbia.plottool.nx_helpers import *  # NOQA
>>> import wbia.plottool as pt
>>> import networkx as nx
>>> import utool as ut
>>> n, s = 9, 4
>>> offsets = list(range(0, (1 + n) * s, s))
>>> node_groups = [ut.lmap(str, range(*o)) for o in ut.itertwo(offsets)]
>>> edge_groups = [ut.combinations(nodes, 2) for nodes in node_groups]
>>> graph = nx.Graph()
>>> [graph.add_nodes_from(nodes) for nodes in node_groups]
>>> [graph.add_edges_from(edges) for edges in edge_groups]
>>> for count, nodes in enumerate(node_groups):
...     nx.set_node_attributes(graph, name='id', values=ut.dzip(nodes,␣
↪[count]))
>>> layoutkw = dict(prog='neato')
>>> graph1, info1 = nx_agraph_layout(graph.copy(), inplace=True, groupby='id
↪', **layoutkw)
>>> graph2, info2 = nx_agraph_layout(graph.copy(), inplace=True, **layoutkw)
>>> graph3, _ = nx_agraph_layout(graph1.copy(), inplace=True, **layoutkw)
>>> nx.set_node_attributes(graph1, name='pin', values='true')
>>> graph4, _ = nx_agraph_layout(graph1.copy(), inplace=True, **layoutkw)
>>> if pt.show_was_requested():
>>>     pt.show_nx(graph1, layout='custom', pnum=(2, 2, 1), fnum=1)
>>>     pt.show_nx(graph2, layout='custom', pnum=(2, 2, 2), fnum=1)
>>>     pt.show_nx(graph3, layout='custom', pnum=(2, 2, 3), fnum=1)
>>>     pt.show_nx(graph4, layout='custom', pnum=(2, 2, 4), fnum=1)
>>>     pt.show_if_requested()
```

(continues on next page)

(continued from previous page)

```
>>> g1pos = nx.get_node_attributes(graph1, 'pos')['1']
>>> g4pos = nx.get_node_attributes(graph4, 'pos')['1']
>>> g2pos = nx.get_node_attributes(graph2, 'pos')['1']
>>> g3pos = nx.get_node_attributes(graph3, 'pos')['1']
>>> print('g1pos = {!r}'.format(g1pos))
>>> print('g4pos = {!r}'.format(g4pos))
>>> print('g2pos = {!r}'.format(g2pos))
>>> print('g3pos = {!r}'.format(g3pos))
>>> assert np.all(g1pos == g4pos), 'points between 1 and 4 were pinned so␣
↪they should be equal'
>>> #assert np.all(g2pos != g3pos), 'points between 2 and 3 were not pinned,␣
↪so they should be different'
```

assert np.all(nx.get_node_attributes(graph1, ‘pos’)[‘1’] == nx.get_node_attributes(graph4, ‘pos’)[‘1’])
assert np.all(nx.get_node_attributes(graph2, ‘pos’)[‘1’] == nx.get_node_attributes(graph3, ‘pos’)[‘1’])

wbia.plottool.nx_helpers.**parse_aedge_layout_attrs**(*aedge*, *translation=None*)
parse grpahviz splineType

wbia.plottool.nx_helpers.**parse_anode_layout_attrs**(*anode*)

wbia.plottool.nx_helpers.**parse_html_graphviz_attrs**()

wbia.plottool.nx_helpers.**parse_point**(*ptstr*)

wbia.plottool.nx_helpers.**patch_pygraphviz**()
Hacks around a python3 problem in 1.3.1 of pygraphviz

wbia.plottool.nx_helpers.**show_nx**(*graph*, *with_labels=True*, *fnum=None*, *pnum=None*, *layout='agraph'*, *ax=None*, *pos=None*, *img_dict=None*, *title=None*, *layoutkw=None*, *verbose=None*, ***kwargs*)

Parameters

- **graph** (*networkx.Graph*) –
- **with_labels** (*bool*) – (default = True)
- **fnum** (*int*) – figure number(default = None)
- **pnum** (*tuple*) – plot number(default = None)
- **layout** (*str*) – (default = ‘agraph’)
- **ax** (*None*) – (default = None)
- **pos** (*None*) – (default = None)
- **img_dict** (*dict*) – (default = None)
- **title** (*str*) – (default = None)
- **layoutkw** (*None*) – (default = None)
- **verbose** (*bool*) – verbosity flag(default = None)

Kwargs: use_image, framewidth, modify_ax, as_directed, hacknoedge, hacknode, arrow_width, fontsize, fontweight, fontname, fontfamilty, fontproperties

CommandLine: python -m wbia.plottool.nx_helpers show_nx –show python -m dtool –tf DependencyCache.make_graph –show python -m wbia.scripts.specialdraw double_depcache_graph –show –testmode python -m vtool.clustering2 unsupervised_multicut_labeling –show

Example

```
>>> # ENABLE_DOCTEST
>>> # xdoctest: +REQUIRES(module:pygraphviz)
>>> from wbia.plottool.nx_helpers import *  # NOQA
>>> import networkx as nx
```

(continues on next page)

(continued from previous page)

```
>>> graph = nx.DiGraph()
>>> graph.add_nodes_from(['a', 'b', 'c', 'd'])
>>> graph.add_edges_from({'a': 'b', 'b': 'c', 'b': 'd', 'c': 'd'}.items())
>>> nx.set_node_attributes(graph, name='shape', values='rect')
>>> nx.set_node_attributes(graph, name='image', values={'a': ut.grab_test_imgpath(
↪'carl.jpg')})
>>> nx.set_node_attributes(graph, name='image', values={'d': ut.grab_test_imgpath(
↪'lena.png')})
>>> #nx.set_node_attributes(graph, name='height', values=100)
>>> with_labels = True
>>> fnum = None
>>> pnum = None
>>> e = show_nx(graph, with_labels, fnum, pnum, layout='agraph')
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

1.11.24 wbia.plottool.other module

wbia.plottool.other.**color_orimag**(*gori*, *gmag*)

wbia.plottool.other.**draw_hist_subbin_maxima**(*hist*, *centers=None*)

1.11.25 wbia.plottool.plot_helpers module

wbia.plottool.plot_helpers.**del_plotdat**(*ax*, *key*)
sets internal property to a matplotlib axis

wbia.plottool.plot_helpers.**draw**()

wbia.plottool.plot_helpers.**ensureqt**()

wbia.plottool.plot_helpers.**get_bbox_centers**(*bbox_list*)

wbia.plottool.plot_helpers.**get_plotdat**(*ax*, *key*, *default=None*)
returns internal property from a matplotlib axis

wbia.plottool.plot_helpers.**get_plotdat_dict**(*ax*)
sets internal property to a matplotlib axis

wbia.plottool.plot_helpers.**get_square_row_cols**(*nSubplots*, *max_cols=None*, *fix=False*, *inclusive=True*)

Parameters
- **nSubplots** –
- **max_cols** (*None*) –

Returns (None, None)

Return type tuple

CommandLine: python -m wbia.plottool.plot_helpers –test-get_square_row_cols

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plot_helpers import *  # NOQA
>>> # build test data
```

(continues on next page)

(continued from previous page)

```
>>> nSubplots = 9
>>> nSubplots_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
>>> max_cols = None
>>> # execute function
>>> rc_list = [get_square_row_cols(nSubplots, fix=True) for nSubplots in␣
↪nSubplots_list]
>>> # verify results
>>> result = repr(np.array(rc_list).T)
>>> print(result)
array([[1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3],
       [1, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4]])
```

wbia.plottool.plot_helpers.**kp_info**(*kp*)

wbia.plottool.plot_helpers.**qt4ensure**()

wbia.plottool.plot_helpers.**qtensure**()

wbia.plottool.plot_helpers.**set_plotdat**(*ax*, *key*, *val*)
sets internal property to a matplotlib axis

1.11.26 wbia.plottool.plots module

wbia.plottool.plots.**colorline**(*x*, *y*, *z=None*, *cmap=<matplotlib.colors.LinearSegmentedColormap object>*, *norm=<matplotlib.colors.Normalize object>*, *linewidth=1*, *alpha=1.0*)
Plot a colored line with coordinates x and y Optionally specify colors in the array z Optionally specify a colormap, a norm function and a line width

References

nbviewer.ipython.org/github/dpsanders/matplotlib-examples/blob/master/colorline.ipynb
CommandLine: python -m wbia.plottool.plots –test-colorline –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> import wbia.plottool as pt
>>> # build test data
>>> x = np.array([1, 3, 3, 2, 5]) / 5.0
>>> y = np.array([1, 2, 1, 3, 5]) / 5.0
>>> z = None
>>> cmap = df2.plt.get_cmap('hsv')
>>> norm = plt.Normalize(0.0, 1.0)
>>> linewidth = 1
>>> alpha = 1.0
>>> # execute function
>>> pt.figure()
>>> result = colorline(x, y, z, cmap)
>>> # verify results
>>> print(result)
>>> pt.dark_background()
>>> pt.show_if_requested()
```

wbia.plottool.plots.**demo_fonts**()

CommandLine: python -m wbia.plottool.plots demo_fonts –show

References

http://stackoverflow.com/questions/8753835/list-of-fonts-avail-mpl

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> demo_fonts()
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.present()
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.plots.**draw_hist_subbin_maxima**(*hist*, *centers=None*, *bin_colors=None*, *maxima_thresh=None*, *remove_endpoints=True*, ***kwargs*)

Parameters

- **hist** (*ndarray*) –
- **centers** (*None*) –

CommandLine: python -m wbia.plottool.plots –test-draw_hist_subbin_maxima –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> import wbia.plottool as pt
>>> hist = np.array([    6.73, 8.69, 0.00, 0.00, 34.62, 29.16, 0.00, 0.00, 6.73,␣
↪8.69])
>>> centers = np.array([-0.39, 0.39, 1.18, 1.96,  2.75,  3.53, 4.32, 5.11, 5.89,␣
↪6.68])
>>> bin_colors = pt.df2.plt.get_cmap('hsv')(centers / vt.TAU)
>>> use_darkbackground = True
>>> maxima_thresh = .8
>>> result = draw_hist_subbin_maxima(hist, centers, bin_colors,
>>>                                  maxima_thresh,
>>>                                  use_darkbackground=use_darkbackground)
>>> print(result)
>>> pt.show_if_requested()
```

wbia.plottool.plots.**draw_histogram**(*bin_labels*, *bin_values*, *xlabel=''*, *ylabel='Freq'*, *xtick_rotation=0*, *transpose=False*, ***kwargs*)

Parameters

- **bin_labels** –
- **bin_values** –
- **xlabel** (*unicode*) – (default = u'')
- **ylabel** (*unicode*) – (default = u'Freq')
- **xtick_rotation** (*int*) – (default = 0)
- **transpose** (*bool*) – (default = False)

Kwargs: fnum, pnum, kind, spread_list, title, titlesize, labelsize, legendsize, ticksize, num_xticks, num_yticks, yticklabels, xticklabels, ytick_rotation, xpad, ypad, xpad_factor, ypad_factor, ypad_high, ypad_low, xpad_high, xpad_low, xscale, yscale, legend_loc, legend_alpha, use_darkbackground, lightbg

CommandLine: python -m wbia.plottool.plots –exec-draw_histogram –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> bin_labels = ['label1', 'label2']
>>> bin_values = [.4, .6]
>>> xlabel = ''
>>> ylabel = 'Freq'
>>> xtick_rotation = 0
>>> transpose = False  # True
>>> kwargs = dict(use_darkbackground=False)
>>> result = draw_histogram(bin_labels, bin_values, xlabel, ylabel, xtick_
↪rotation, transpose, **kwargs)
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.plots.**draw_subextrema**(*ydata*, *xdata=None*, *op='max'*, *bin_colors=None*, *thresh_factor=None*, *normalize_x=True*, *flat=True*)

Parameters

- **ydata** (*ndarray*) –
- **xdata** (*None*) –

CommandLine: python -m wbia.plottool.plots –test-draw_subextrema –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> import vtool as vt
>>> import wbia.plottool as pt
>>> ydata = np.array([    6.73, 8.69, 0.00, 0.00, 34.62, 29.16, 0.01, 0.00, 6.73,␣
↪8.69])
>>> xdata = np.array([-0.39, 0.39, 1.18, 1.96,  2.75,  3.53, 4.32, 5.11, 5.89, 6.
↪68])
>>> bin_colors = pt.df2.plt.get_cmap('hsv')(xdata / vt.TAU)
>>> use_darkbackground = True
>>> thresh_factor = .01
>>> op = 'max'
>>> ut.exec_funckw(draw_subextrema, globals())
>>> result = draw_subextrema(ydata, xdata, bin_colors=bin_colors,
>>>                          thresh_factor=thresh_factor, op=op)
>>> print(result)
>>> pt.show_if_requested()
```

wbia.plottool.plots.**draw_time_distribution**(*unixtime_list*, *bw=None*)

wbia.plottool.plots.**draw_time_histogram**(*unixtime_list*, ***kwargs*)

wbia.plottool.plots.**draw_timedelta_pie**(*timedeltas*, *bins=None*, *fnum=None*, *pnum=(1, 1, 1)*, *label=''*)

Parameters

- **timedeltas** (*list*) –
- **bins** (*None*) – (default = None)

CommandLine: python -m wbia.plottool.plots –exec-draw_timedelta_pie –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> timedeltas = np.array([    1.,    14.,    17.,    34.,     4.,    36.,    34.,
↪     2.,
...                         3268.,    34., np.nan,    33.,     5.,     2.,    16.,
↪     5.,
...                           35.,    64.,   299.,    35.,     2.,     5.,    34.,
↪    12.,
...                            1.,     8.,     6.,     7.,    11.,     5.,    46.,
↪    47.,
...                           22.,     3.,  np.nan,    11.], dtype=np.float64) **␣
↪2
>>> bins = None
>>> result = draw_timedelta_pie(timedeltas, bins)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.plots.**estimate_pdf**(*data*, *bw_factor*)

wbia.plottool.plots.**get_good_logyscale_kwargs**(*y_data*, *adaptive_knee_scaling=False*)

wbia.plottool.plots.**interval_line_plot**(*xdata, ydata_mean, y_data_std, color=[1, 0, 0], label=None, marker='o', linestyle='-'*)

Parameters

- **xdata** (*ndarray*) –
- **ydata_mean** (*ndarray*) –
- **y_data_std** (*ndarray*) –

SeeAlso: pt.multi_plot (using the spread_list kwarg)
CommandLine: python -m wbia.plottool.plots –test-interval_line_plot –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> xdata = [1, 2, 3, 4, 5, 6, 7, 8]
>>> ydata_mean = [2, 3, 4, 3, 3, 2, 2, 2]
>>> y_data_std = [1, 2, 1, 1, 3, 2, 2, 1]
>>> result = interval_line_plot(xdata, ydata_mean, y_data_std)
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.plots.**interval_stats_plot**(*param2_stat_dict*, *fnum=None*, *pnum=(1, 1, 1)*, *x_label=''*, *y_label=''*, *title=''*)

interval plot for displaying mean, range, and std

Parameters

- **fnum** (*int*) – figure number

- **pnum** (*tuple*) – plot number

CommandLine: python -m wbia.plottool.plots –test-interval_stats_plot python -m wbia.plottool.plots –test-interval_stats_plot –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> import wbia.plottool as pt
>>> # build test data
>>> param2_stat_dict = {
...     0.5: dict([('max', 0.0584), ('min', 0.0543), ('mean', 0.0560), ('std', 0.
↪00143),]),
...     0.6: dict([('max', 0.0593), ('min', 0.0538), ('mean', 0.0558), ('std', 0.
↪00178),]),
...     0.7: dict([('max', 0.0597), ('min', 0.0532), ('mean', 0.0556), ('std', 0.
↪00216),]),
...     0.8: dict([('max', 0.0601), ('min', 0.0525), ('mean', 0.0552), ('std', 0.
↪00257),]),
...     0.9: dict([('max', 0.0604), ('min', 0.0517), ('mean', 0.0547), ('std', 0.
↪00300),]),
...     1.0: dict([('max', 0.0607), ('min', 0.0507), ('mean', 0.0541), ('std', 0.
↪00345),])
... }
>>> fnum = None
>>> pnum = (1, 1, 1)
>>> title = 'p vs score'
>>> x_label = 'p'
>>> y_label = 'score diff'
>>> result = interval_stats_plot(param2_stat_dict, fnum, pnum, x_label, y_label,␣
↪title)
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.plots.**is_default_dark_bg**()

wbia.plottool.plots.**multi_plot**(*xdata=None*, *ydata_list=[]*, ***kwargs*)

plots multiple lines, bars, etc...

This is the big function that implements almost all of the heavy lifting in this file. Any function not using this should probably find a way to use it. It is pretty general and relatively clean.

Parameters

- **xdata** (*ndarray*) – can also be a list of arrays
- **ydata_list** (*list of ndarrays*) – can also be a single array

Kwargs:

Misc: fnum, pnum, use_legend, legend_loc
Labels: xlabel, ylabel, title, figtitle ticksize, titlesize, legendsize, labelsize
Grid: gridlinewidth, gridlinestyle
Ticks: num_xticks, num_yticks, tickwidth, ticklength, ticksize
Data: xmin, xmax, ymin, ymax, spread_list # can append _list to any of these plot_kw_keys = ['label', 'color', 'marker', 'markersize', 'markeredgewidth', 'linewidth', 'linestyle']
kind = ['bar', 'plot', ...]
if kind='plot': spread

if kind='bar': stacked, width

References

matplotlib.org/examples/api/barchart_demo.html

CommandLine: python -m wbia.plottool.plots multi_plot:0 –show python -m wbia.plottool.plots multi_plot:1 –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> xdata = [1, 2, 3, 4, 5]
>>> ydata_list = [[1, 2, 3, 4, 5], [3, 3, 3, 3, 3], [5, 4, np.nan, 2, 1], [4, 3,␣
↪np.nan, 1, 0]]
>>> kwargs = {'label_list': ['spamΣ', 'eggs', 'jamμ', 'pram'],  'linestyle': '-'}
>>> #fig = multi_plot(xdata, ydata_list, title='$\phi_1(\\vec{x})$', xlabel='\nfds
↪', **kwargs)
>>> fig = multi_plot(xdata, ydata_list, title='ΣΣΣμμμ', xlabel='\nfdsΣΣΣμμμ',␣
↪**kwargs)
>>> result = ('fig = %s' % (str(fig),))
>>> fig2 = multi_plot([1, 2, 3], [4, 5, 6], fnum=4)
>>> result = ('fig = %s' % (str(fig),))
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> fig = multi_plot([1, 2, 3], [4, 5, 6])
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.plots.**plot_densities**(*prob_list*, *prob_lbls=None*, *prob_colors=None*, *xdata=None*, *prob_thresh=None*, *score_thresh=None*, *figtitle='plot_probabilities'*, *fnum=None*, *pnum=(1, 1, 1)*, *fill=False*, ***kwargs*)

Input: a list of scores (either chip or descriptor)

Concatenates and sorts the scores Sorts and plots with different types of scores labeled

Parameters

- **prob_list** (*list*) –
- **prob_lbls** (*None*) – (default = None)
- **prob_colors** (*None*) – (default = None)
- **xdata** (*None*) – (default = None)
- **prob_thresh** (*None*) – (default = None)
- **figtitle** (*str*) – (default = 'plot_probabilities')
- **fnum** (*int*) – figure number(default = None)
- **pnum** (*tuple*) – plot number(default = (1, 1, 1))
- **fill** (*bool*) – (default = False)

CommandLine: python -m wbia.plottool.plots –exec-plot_probabilities –show –lightbg

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> prob_list = [[.01, .02, .03, .04, .03, .06, .03, .04]]
>>> prob_lbls = ['prob']
>>> prob_colors = None
>>> xdata = None
>>> prob_thresh = None
>>> figtitle = 'plot_probabilities'
>>> fnum = None
>>> pnum = (1, 1, 1)
>>> fill = True
>>> score_thresh = None
>>> result = plot_probabilities(prob_list, prob_lbls, prob_colors, xdata, prob_
↪thresh, score_thresh, figtitle, fnum, pnum, fill)
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.plots.**plot_multiple_scores**(*known_nd_data*, *known_target_points*, *nd_labels*, *target_label*, *title=None*, *use_legend=True*, *color_list=None*, *marker_list=None*, *report_max=True*, ***kwargs*)

Plots nd-data in 2d using multiple contour lines

CommandLine: python -m wbia.plottool.plots –test-plot_multiple_scores –show

python -m wbia.plottool.plots –exec-plot_rank_cumhist –adjust=.15 –dpi=512 –figsize=11,4 –clip-white –dpath ~/latex/crall-candidacy-2015/ –save “figures/tmp.jpg” –diskshow

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> known_nd_data = np.array([[  1,   2,   4,   7,   1,   2,   4,   7,   1,   2,
↪ 4,   7,   1,
...                            2,   4,   7,   1,   2,   4,   7],
...                           [ 50,  50,  50,  50, 100, 100, 100, 100, 200, 200,
↪200, 200, 300,
...                            300, 300, 300, 500, 500, 500, 500]], dtype=np.
↪int64).T
>>> known_target_points = np.array([35, 32, 32, 30, 33, 32, 33, 30, 32, 31, 31,
↪32, 36, 33, 33, 32, 33,
...                              33, 32, 31], dtype=np.int64)
>>> label_list = ['custom', 'custom:sv_on=False']
>>> nd_labels = [u'K', u'dsize']
>>> target_label = 'score'
>>> fnum = None
>>> pnum = None
>>> use_legend = True
>>> title = 'test'
>>> result = plot_multiple_scores(known_nd_data, known_target_points, nd_labels,
↪target_label, title=title)
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.plots.**plot_pdf**(*data*, *draw_support=True*, *scale_to=None*, *label=None*, *color=0*, *nYTicks=3*)

wbia.plottool.plots.**plot_probabilities**(*prob_list*, *prob_lbls=None*, *prob_colors=None*, *xdata=None*, *prob_thresh=None*, *score_thresh=None*, *figtitle='plot_probabilities'*, *fnum=None*, *pnum=(1, 1, 1)*, *fill=False*, ***kwargs*)

Input: a list of scores (either chip or descriptor)

Concatenates and sorts the scores Sorts and plots with different types of scores labeled

Parameters

- **prob_list** (*list*) –
- **prob_lbls** (*None*) – (default = None)
- **prob_colors** (*None*) – (default = None)
- **xdata** (*None*) – (default = None)
- **prob_thresh** (*None*) – (default = None)
- **figtitle** (*str*) – (default = 'plot_probabilities')
- **fnum** (*int*) – figure number(default = None)
- **pnum** (*tuple*) – plot number(default = (1, 1, 1))
- **fill** (*bool*) – (default = False)

CommandLine: python -m wbia.plottool.plots –exec-plot_probabilities –show –lightbg

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> prob_list = [[.01, .02, .03, .04, .03, .06, .03, .04]]
>>> prob_lbls = ['prob']
>>> prob_colors = None
>>> xdata = None
>>> prob_thresh = None
>>> figtitle = 'plot_probabilities'
>>> fnum = None
>>> pnum = (1, 1, 1)
>>> fill = True
>>> score_thresh = None
>>> result = plot_probabilities(prob_list, prob_lbls, prob_colors, xdata, prob_
↪thresh, score_thresh, figtitle, fnum, pnum, fill)
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.plots.**plot_probs**(*prob_list*, *prob_lbls=None*, *prob_colors=None*, *xdata=None*, *prob_thresh=None*, *score_thresh=None*, *figtitle='plot_probabilities'*, *fnum=None*, *pnum=(1, 1, 1)*, *fill=False*, ***kwargs*)

Input: a list of scores (either chip or descriptor)

Concatenates and sorts the scores Sorts and plots with different types of scores labeled

Parameters

- **prob_list** (*list*) –
- **prob_lbls** (*None*) – (default = None)
- **prob_colors** (*None*) – (default = None)
- **xdata** (*None*) – (default = None)
- **prob_thresh** (*None*) – (default = None)
- **figtitle** (*str*) – (default = 'plot_probabilities')

- **fnum** (*int*) – figure number(default = None)
- **pnum** (*tuple*) – plot number(default = (1, 1, 1))
- **fill** (*bool*) – (default = False)

CommandLine: python -m wbia.plottool.plots –exec-plot_probabilities –show –lightbg

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> prob_list = [[.01, .02, .03, .04, .03, .06, .03, .04]]
>>> prob_lbls = ['prob']
>>> prob_colors = None
>>> xdata = None
>>> prob_thresh = None
>>> figtitle = 'plot_probabilities'
>>> fnum = None
>>> pnum = (1, 1, 1)
>>> fill = True
>>> score_thresh = None
>>> result = plot_probabilities(prob_list, prob_lbls, prob_colors, xdata, prob_
↪thresh, score_thresh, figtitle, fnum, pnum, fill)
>>> print(result)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.plots.**plot_rank_cumhist**(*cdf_list*, *label_list*, *color_list=None*, *marker_list=None*, *edges=None*, *xlabel=''*, *ylabel='cumfreq'*, *use_legend=True*, *num_xticks=None*, *kind='bar'*, ***kwargs*)

Plots CMC curves TODO rename to plot_cmc

CommandLine: python -m wbia.plottool.plots –test-plot_rank_cumhist –show

python -m wbia.plottool.plots –exec-plot_rank_cumhist –adjust=.15 –dpi=512 –figsize=11,4 –clipwhite –dpath ~/latex/crall-candidacy-2015/ –save "figures/tmp.jpg" –diskshow

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> cdf_list = np.array(
>>>     [[ 88,  92,  93,  96,  96,  96,  96,  98,  99,  99, 100, 100, 100],
>>>      [ 79,  82,  82,  85,  86,  87,  87,  87,  88,  89,  90,  90,  90]])
>>> edges = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
>>> label_list = ['custom', 'custom:sv_on=False']
>>> fnum = None
>>> pnum = None
>>> plot_rank_cumhist(cdf_list, label_list, edges=edges, fnum=fnum, pnum=pnum)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.plots.**plot_score_histograms**(*scores_list*, *score_lbls=None*, *score_markers=None*, *score_colors=None*, *markersizes=None*, *fnum=None*, *pnum=(1, 1, 1)*, *title=None*, *score_label='score'*, *score_thresh=None*, *overlay_prob_given_list=None*, *overlay_score_domain=None*, *logscale=False*, *histnorm=False*, ***kwargs*)

Accumulates scores into histograms and plots them
CommandLine: python -m wbia.plottool.plots –test-plot_score_histograms –show
Ignore:

```
>>> score_label = 'score'
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> rng = np.random.RandomState(seed=0)
>>> tp_support = rng.normal(loc=6.5, size=(256,))
>>> tn_support = rng.normal(loc=3.5, size=(256,))
>>> scores_list = [tp_support, tn_support]
>>> logscale = True
>>> title = 'plot_scores_histogram'
>>> result = plot_score_histograms(scores_list, title=title,
>>>                                logscale=logscale)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
>>> print(result)
```

wbia.plottool.plots.**plot_search_surface**(*known_nd_data*, *known_target_points*, *nd_labels*, *target_label*, *fnum=None*, *pnum=None*, *title=None*)

3D Function

Parameters

- **known_nd_data** – should be integral for now
- **known_target_points** –
- **nd_labels** –
- **target_label** –
- **fnum** (*int*) – figure number(default = None)

Returns ax
Return type

?

CommandLine: python -m wbia.plottool.plots –exec-plot_search_surface –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> known_nd_data = np.array([x.flatten() for x in np.meshgrid(*[np.linspace(-20,␣
↪20, 10).astype(np.int32), np.linspace(-20, 20, 10).astype(np.int32)])]).T
>>> # complicated polynomial target
```

(continues on next page)

(continued from previous page)

```
>>> known_target_points = -.001 * known_nd_data.T[0] ** 4 + .25 * known_nd_data.
↪T[1] ** 2 - .0005 * known_nd_data.T[1] ** 4 + .001 * known_nd_data.T[1] ** 3
>>> nd_labels = ['big-dim', 'small-dim']
>>> target_label = ['score']
>>> fnum = 1
>>> ax = plot_search_surface(known_nd_data, known_target_points, nd_labels,␣
↪target_label, fnum)
```

wbia.plottool.plots.**plot_sorted_scores**(*scores_list*, *score_lbls=None*, *score_markers=None*, *score_colors=None*, *markersizes=None*, *fnum=None*, *pnum=(1, 1, 1)*, *logscale=True*, *figtitle=None*, *score_label='score'*, *thresh=None*, *use_stems=None*, ***kwargs*)

Concatenates and sorts the scores Sorts and plots with different types of scores labeled

Parameters

- **scores_list** (*list*) – a list of scores
- **score_lbls** (*None*) –
- **score_markers** (*None*) –
- **score_colors** (*None*) –
- **markersizes** (*None*) –
- **fnum** (*int*) – figure number
- **pnum** (*tuple*) – plot number
- **logscale** (*bool*) –
- **figtitle** (*str*) –

CommandLine: python -m wbia.plottool.plots –test-plot_sorted_scores –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> rng = np.random.RandomState(seed=0)
>>> tp_support = rng.normal(loc=6.5, size=(256,))
>>> tn_support = rng.normal(loc=3.5, size=(256,))
>>> scores_list = [tp_support, tn_support]
>>> score_lbls = None
>>> score_markers = None
>>> score_colors = None
>>> markersizes = None
>>> fnum = None
>>> pnum = (1, 1, 1)
>>> logscale = True
>>> figtitle = 'plot_sorted_scores'
>>> result = plot_sorted_scores(scores_list, score_lbls, score_markers,
>>>                             score_colors, markersizes, fnum, pnum,
>>>                             logscale, figtitle)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
>>> print(result)
```

wbia.plottool.plots.**plot_stems**(*x_data*, *y_data*, *fnum=None*, *pnum=(1, 1, 1)*, ***kwargs*)

Example

```
>>> import wbia.plottool as pt
>>> x_data = [1, 1, 2, 3, 3, 3, 4, 4, 5]
>>> y_data = [1, 2, 1, 2, 1, 4, 4, 5, 1]
>>> pt.plots.plot_stems(x_data, y_data)
>>> pt.show_if_requested()
```

wbia.plottool.plots.**set_logyscale_from_data**(*y_data*)

wbia.plottool.plots.**word_histogram2**(*text_list*, *weight_list=None*, ***kwargs*)

Parameters **text_list** (*list*) –

References

stackoverflow.com/questions/17430105/autofmt-xdate-deletes-x-axis-labels-of-all-subplots
CommandLine: python -m wbia.plottool.plots –exec-word_histogram2 –show –lightbg

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> text_list = []
>>> item_list = text_list = ['spam', 'eggs', 'ham', 'jam', 'spam', 'spam', 'spam',
↪ 'eggs', 'spam']
>>> weight_list = None
>>> #weight_list = [.1, .2, .3, .4, .5, .5, .4, .3, .1]
>>> #text_list = [x.strip() for x in ut.lorium_ipsum().split()]
>>> result = word_histogram2(text_list, weight_list)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

wbia.plottool.plots.**wordcloud**(*text*, *size=None*, *fnum=None*, *pnum=None*, *ax=None*)

References

bioinfoexpert.com/?p=592 sudo pip install git+git://github.com/amueller/word_cloud.git

Parameters

- **text** (*str or dict*) – raw text or dictionary of frequencies
- **fnum** (*int*) – figure number(default = None)
- **pnum** (*tuple*) – plot number(default = None)

CommandLine: python -m wbia.plottool.plots –exec-wordcloud –show python -m wbia.plottool.plots –exec-wordcloud –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> text = '''
        Normally, Frost-Breath-type cards are only good in aggressive decks,
        but add an Eldrazi Scion into the mix and that all changes. I'm not
```

(continues on next page)

(continued from previous page)

```
        adverse to playing a card that ramps my mana, can trade for an x/1, and
        does so while keeping me alive. I still would rather be beating down if
        I'm including this in my deck, but I think it does enough different
        things at a good rate that you are more likely to play it than not.
        Cards that swing a race this drastically are situationally awesome, and
        getting the Eldrazi Scion goes a long way toward mitigating the cost of
        drawing this when you aren't in a race (which is the reason
        non-aggressive decks tend to avoid this effect).
    '''
>>> fnum = None
>>> pnum = None
>>> result = wordcloud(text, fnum, pnum)
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
>>> print(result)
```

wbia.plottool.plots.**zoom_effect01**(*ax1*, *ax2*, *xmin*, *xmax*, ***kwargs*)

connect ax1 & ax2. The x-range of (xmin, xmax) in both axes will be marked. The keywords parameters will be used ti create patches.

Parameters

- **ax1** (*mpl.axes*) – the main axes
- **ax2** (*mpl.axes*) – the zoomed axes
- **(xmin,xmax)** – the limits of the colored area in both plot axes.

Returns (c1, c2, bbox_patch1, bbox_patch2, p)

Return type tuple

References

matplotlib.org/users/annotations_guide.html

CommandLine: python -m wbia.plottool.plots zoom_effect01 –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.plots import *  # NOQA
>>> import wbia.plottool as pt
>>> cdf_list = np.array(
>>>     [[10, 15, 40, 42, 50, 88,  92,  93,  96,  96,  96,  96,  98,  99,  99,␣
↪100, 100, 100],
>>>      [20, 30, 31, 66, 75, 79,  82,  82,  85,  86,  87,  87,  87,  88,  89, ␣
↪90,  90,  90]])
>>> edges = list(range(0, len(cdf_list[0]) + 1))
>>> label_list = ['custom', 'custom:sv_on=False']
>>> fnum = 1
>>> numranks = len(cdf_list[0])
>>> top = 3
>>> plot_rank_cumhist(cdf_list, label_list, edges=edges, xmin=.9, num_
↪xticks=numranks, fnum=fnum, pnum=(2, 1, 1), kind='plot', ymin=0, ymax=100)
>>> ax1 = pt.gca()
>>> plot_rank_cumhist(cdf_list.T[0:top].T, label_list, edges=edges[0:top + 1],␣
↪xmin=.9, num_xticks=top, fnum=fnum, pnum=(2, 1, 2), kind='plot', ymin=0,␣
↪ymax=100)
>>> ax2 = pt.gca()
```

(continues on next page)

(continued from previous page)

```
>>> xmin = 1
>>> xmax = top
>>> (c1, c2, bbox_patch1, bbox_patch2, p) = zoom_effect01(ax1, ax2, xmin, xmax)
>>> result = ('(c1, c2, bbox_patch1, bbox_patch2, p) = %s' % (ut.repr2((c1, c2,␣
↪bbox_patch1, bbox_patch2, p)),))
>>> print(result)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.show_if_requested()
```

1.11.27 wbia.plottool.screeninfo module

wbia.plottool.screeninfo.**get_avail_geom**(*monitor_num=None*, *percent_w=1.0*, *percent_h=1.0*)

wbia.plottool.screeninfo.**get_monitor_geom**(*monitor_num=0*)

Parameters **monitor_num** (*int*) – (default = 0)

Returns geom

Return type tuple

CommandLine: python -m wbia.plottool.screeninfo get_monitor_geom –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.screeninfo import *  # NOQA
>>> monitor_num = 0
>>> geom = get_monitor_geom(monitor_num)
>>> result = ('geom = %s' % (ut.repr2(geom),))
>>> print(result)
```

wbia.plottool.screeninfo.**get_monitor_geometries**()

wbia.plottool.screeninfo.**get_number_of_monitors**()

wbia.plottool.screeninfo.**get_resolution_info**(*monitor_num=0*)

Parameters **monitor_num** (*int*) – (default = 0)

Returns info

Return type dict

CommandLine: python -m wbia.plottool.screeninfo get_resolution_info –show xrandr | grep ‘ connected’ grep “NVIDIA” /var/log/Xorg.0.log

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.screeninfo import *  # NOQA
>>> monitor_num = 1
>>> for monitor_num in range(get_number_of_monitors()):
>>>     info = get_resolution_info(monitor_num)
>>>     print('monitor(%d).info = %s' % (monitor_num, ut.repr3(info,␣
↪precision=3)))
```

wbia.plottool.screeninfo.**get_stdpxls**()

wbia.plottool.screeninfo.**get_valid_fig_positions**(*num_wins*, *max_rows=None*, *row_first=True*, *monitor_num=None*, *percent_w=1.0*, *percent_h=1.0*)

Returns a list of bounding boxes where figures can be placed on the screen

wbia.plottool.screeninfo.**get_xywh_pads**()

wbia.plottool.screeninfo.**infer_monitor_specs**(*res_w*, *res_h*, *inches_diag*)

monitors = [dict(name='work1', inches_diag=23, res_w=1920, res_h=1080), dict(name='work2', inches_diag=24, res_w=1920, res_h=1200),

dict(name='hp-129', inches_diag=25, res_w=1920, res_h=1080), dict(name='?-26', inches_diag=26, res_w=1920, res_h=1080), dict(name='?-27', inches_diag=27, res_w=1920, res_h=1080),

] for info in monitors:

name = info['name'] inches_diag = info['inches_diag'] res_h = info['res_h'] res_w = info['res_w'] print('—') print(name) inches_w = inches_diag * res_w / np.sqrt(res_h**2 + res_w**2) inches_h = inches_diag * res_h / np.sqrt(res_h**2 + res_w**2) print('inches diag = %.2f' % (inches_diag)) print('inches WxH = %.2f x %.2f' % (inches_w, inches_h))

#inches_w = inches_diag * res_w/sqrt(res_h**2 + res_w**2)

1.11.28 wbia.plottool.test_colorsys module

wbia.plottool.test_colorsys.**TEST_COLORSYS**()

1.11.29 wbia.plottool.test_vtk_poly module

wbia.plottool.test_vtk_poly.**rhombic_dodecahedron**()

wbia.plottool.test_vtk_poly.**rhombicuboctahedron**()

1.11.30 wbia.plottool.viz_featrow module

wbia.plottool.viz_featrow.**draw_feat_row**(*chip*, *fx*, *kp*, *sift*, *fnum*, *nRows*, *nCols=None*, *px=None*, *prevsift=None*, *origsift=None*, *aid=None*, *info=''*, *type_=None*, *shape_labels=False*, *vecfield=False*, *multicolored_arms=False*, *draw_chip=False*, *draw_warped=True*, *draw_unwarped=True*, *draw_desc=True*, *rect=True*, *ori=True*, *pts=False*, ***kwargs*)

SeeAlso: wbia.viz.viz_nearest_descriptors ~/code/wbia/wbia/viz/viz_nearest_descriptors.py

CommandLine:

Use this to find the fx you want to visualize python -m wbia.plottool.interact_keypoints –test-ishow_keypoints –show –fname zebra.png

Use this to visualize the featrow python -m wbia.plottool.viz_featrow –test-draw_feat_row –show python -m wbia.plottool.viz_featrow –test-draw_feat_row –show –fname zebra.png –fx=121 –feat-all –no-sift python -m wbia.plottool.viz_featrow –test-draw_feat_row –dpath figures –save ~/latex/crall-candidacy-2015/figures/viz_featrow.jpg

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.viz_featrow import *  # NOQA
>>> import wbia.plottool as pt
>>> # build test data
>>> kpts, vecs, imgBGR = pt.viz_keypoints.testdata_kpts()
>>> chip = imgBGR
>>> print('There are %d features' % (len(vecs)))
>>> fx = ut.get_argval('--fx', type_=int, default=0)
>>> kp = kpts[fx]
>>> sift = vecs[fx]
>>> fnum = 1
>>> nRows = 1
>>> nCols = 2
>>> px = 0
>>> if True:
>>>     from wbia.scripts.thesis import TMP_RC
>>>     import matplotlib as mpl
>>>     mpl.rcParams.update(TMP_RC)
>>> hack = ut.get_argflag('--feat-all')
>>> sift = sift if not ut.get_argflag('--no-sift') else None
>>> draw_desc = sift is not None
>>> kw = dict(
>>>     prevsift=None, origsift=None, aid=None, info='', type_=None,
>>>     shape_labels=False, vecfield=False, multicolored_arms=True,
>>>     draw_chip=hack, draw_unwarped=hack, draw_warped=True, draw_desc=draw_desc
>>> )
>>> # execute function
>>> result = draw_feat_row(chip, fx, kp, sift, fnum, nRows, nCols, px,
>>>                        rect=False, ori=False, pts=False, **kw)
>>> # verify results
>>> print(result)
>>> pt.show_if_requested()
```

wbia.plottool.viz_featrow.**formatdist**(*val*)

wbia.plottool.viz_featrow.**precisionstr**(*c='E'*, *pr=2*)

1.11.31 wbia.plottool.viz_image2 module

wbia.plottool.viz_image2.**draw_chip_overlay**(*ax*, *bbox*, *theta*, *text*, *is_sel*)
Draw an annotation around a chip in the image

wbia.plottool.viz_image2.**draw_image_overlay**(*ax*, *bbox_list=[]*, *theta_list=None*, *text_list=None*, *sel_list=None*, *draw_lbls=True*)

wbia.plottool.viz_image2.**show_image**(*img*, *bbox_list=[]*, *title=''*, *theta_list=None*, *text_list=None*, *sel_list=None*, *draw_lbls=True*, *fnum=None*, *annote=True*, ***kwargs*)
Driver function to show images

1.11.32 wbia.plottool.viz_keypoints module

wbia.plottool.viz_keypoints.**show_keypoints**(*chip*, *kpts*, *fnum=0*, *pnum=None*, ***kwargs*)

Parameters

- **chip** (*ndarray[uint8_t, ndim=2]*) – annotation image data
- **kpts** (*ndarray[float32_t, ndim=2]*) – keypoints
- **fnum** (*int*) – figure number(default = 0)
- **pnum** (*tuple*) – plot number(default = None)

Kwargs: ddd, title, figtitle, interpolation, cmap, heatmap, data_colorbar, darken, update, redraw_image, docla, doclf, projection, sel_fx

CommandLine: python -m wbia.plottool.viz_keypoints –exec-show_keypoints

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.plottool.viz_keypoints import *  # NOQA
>>> import vtool as vt
>>> kpts, vecs, chip = testdata_kpts()
>>> fnum = 0
>>> pnum = None
>>> result = show_keypoints(chip, kpts, fnum, pnum)
>>> print(result)
```

wbia.plottool.viz_keypoints.**testdata_kpts**()

1.11.33 Module contents

Wrappers around matplotlib

wbia.plottool.**reassign_submodule_attributes**(*verbose=1*)
Updates attributes in the __init__ modules with updated attributes in the submodules.

wbia.plottool.**reload_subs**(*verbose=1*)
Reloads wbia.plottool and submodules

wbia.plottool.**rrrr**(*verbose=1*)
Reloads wbia.plottool and submodules

1.12 wbia.scripts package

1.12.1 Submodules

1.12.2 wbia.scripts._neighbor_experiment module

wbia.scripts._neighbor_experiment.**augment_nnindexer_experiment**()

References

http://answers.opencv.org/question/44592/flann-index-training-fails-with-segfault/

CommandLine: utprof.py -m wbia.algo.hots._neighbor_experiment –test-augment_nnindexer_experiment python -m wbia.algo.hots._neighbor_experiment –test-augment_nnindexer_experiment

python -m wbia.algo.hots._neighbor_experiment –test-augment_nnindexer_experiment –db PZ_MTEST –diskshow –adjust=.1 –save “augment_experiment_{db}.png” –dpath=’.’ –dpi=180

–figsize=9,6 python -m wbia.algo.hots._neighbor_experiment –test-augment_nnindexer_experiment –db PZ_Master0 –diskshow –adjust=.1 –save “augment_experiment_{db}.png” –dpath=’.’ –dpi=180 –figsize=9,6 –nosave-flann –show python -m wbia.algo.hots._neighbor_experiment –test-augment_nnindexer_experiment –db PZ_Master0 –diskshow –adjust=.1 –save “augment_experiment_{db}.png” –dpath=’.’ –dpi=180 –figsize=9,6 –nosave-flann –show

python -m wbia.algo.hots._neighbor_experiment –test-augment_nnindexer_experiment –db PZ_Master0 –diskshow –adjust=.1 –save “augment_experiment_{db}.png” –dpath=’.’ –dpi=180 –figsize=9,6 –nosave-flann –no-api-cache –nocache-uuids

python -m wbia.algo.hots._neighbor_experiment –test-augment_nnindexer_experiment –db PZ_MTEST –show python -m wbia.algo.hots._neighbor_experiment –test-augment_nnindexer_experiment –db PZ_Master0 –show

RUNS THE SEGFAULTING CASE python -m wbia.algo.hots._neighbor_experiment –test-augment_nnindexer_experiment –db PZ_Master0 –show # Debug it gdb python run -m wbia.algo.hots._neighbor_experiment –test-augment_nnindexer_experiment –db PZ_Master0 –show gdb python run -m wbia.algo.hots._neighbor_experiment –test-augment_nnindexer_experiment –db PZ_Master0 –diskshow –adjust=.1 –save “augment_experiment_{db}.png” –dpath=’.’ –dpi=180 –figsize=9,6

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots._neighbor_experiment import *  # NOQA
>>> # execute function
>>> augment_nnindexer_experiment()
>>> # verify results
>>> ut.show_if_requested()
```

wbia.scripts._neighbor_experiment.**flann_add_time_experiment**()
builds plot of number of annotations vs indexer build time.

TODO: time experiment

CommandLine: python -m wbia.algo.hots._neighbor_experiment –test-flann_add_time_experiment –db PZ_MTEST –show python -m wbia.algo.hots._neighbor_experiment –test-flann_add_time_experiment –db PZ_Master0 –show utprof.py -m wbia.algo.hots._neighbor_experiment –test-flann_add_time_experiment –show

valgrind –tool=memcheck –suppressions=valgrind-python.supp python -m wbia.algo.hots._neighbor_experiment –test-flann_add_time_experiment –db PZ_MTEST –no-with-reindex

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots._neighbor_experiment import *  # NOQA
>>> import wbia
>>> #ibs = wbia.opendb('PZ_MTEST')
>>> result = flann_add_time_experiment()
>>> # verify results
>>> print(result)
>>> ut.show_if_requested()
```

wbia.scripts._neighbor_experiment.**pyflann_remove_and_save**()

References

Logic goes here ~/code/flann/src/cpp/flann/algorithms/kdtree_index.h

~/code/flann/src/cpp/flann/util/serialization.h ~/code/flann/src/cpp/flann/util/dynamic_bitset.h

Bindings go here ~/code/flann/src/cpp/flann/flann.cpp ~/code/flann/src/cpp/flann/flann.h

Contains stuff for the flann namespace like flann::log_level # Also has Index with # Matrix<ElementType> features; SEEMS USEFUL ~/code/flann/src/cpp/flann/flann.hpp

Wrappers go here ~/code/flann/src/python/pyflann/flann_ctypes.py ~/code/flann/src/python/pyflann/index.py

~/local/build_scripts/flannscripts/autogen_bindings.py
Greping: cd ~/code/flann/src grep -ER cleanRemovedPoints * grep -ER **removed_points_** *
CommandLine: python -m wbia.algo.hots._neighbor_experiment –exec-pyflann_remove_and_save

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots._neighbor_experiment import *  # NOQA
>>> pyflann_remove_and_save()
```

wbia.scripts._neighbor_experiment.**pyflann_test_remove_add**()
CommandLine: python -m wbia.algo.hots._neighbor_experiment –exec-pyflann_test_remove_add

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots._neighbor_experiment import *  # NOQA
>>> pyflann_test_remove_add()
```

wbia.scripts._neighbor_experiment.**pyflann_test_remove_add2**()
CommandLine: python -m wbia.algo.hots._neighbor_experiment –exec-pyflann_test_remove_add2

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots._neighbor_experiment import *  # NOQA
>>> pyflann_test_remove_add2()
```

wbia.scripts._neighbor_experiment.**subindexer_time_experiment**()
builds plot of number of annotations vs indexer build time.

TODO: time experiment

wbia.scripts._neighbor_experiment.**trytest_incremental_add**(*ibs*)
Parameters **ibs** (IBEISController) –

CommandLine: python -m wbia.algo.hots._neighbor_experiment –test-test_incremental_add

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots.neighbor_index_cache import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('PZ_MTEST')
>>> result = test_incremental_add(ibs)
>>> print(result)
```

wbia.scripts._neighbor_experiment.**trytest_multiple_add_removes**()
CommandLine: python -m wbia.algo.hots._neighbor_experiment –exec-test_multiple_add_removes

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots._neighbor_experiment import *  # NOQA
>>> result = test_multiple_add_removes()
>>> print(result)
```

1.12.3 wbia.scripts._thesis_helpers module

class wbia.scripts._thesis_helpers.**DBInputs**(*dbname=None*)
Bases: object

classmethod draw(*expt_name*, *dbnames*, **args*)
CommandLine: python -m wbia Chap3.draw nsum –dbs=GZ_Master1,PZ_Master1 python -m wbia Chap3.draw foregroundness –dbs=GZ_Master1,PZ_Master1 –diskshow python -m wbia Chap3.draw kexpt –dbs=GZ_Master1 –diskshow

python -m wbia Chap4.draw importance GZ_Master1

python -m wbia Chap4.draw hard_cases GZ_Master1,PZ_Master1 match_state,photobomb_state –diskshow

Example: # >>> # Script # >>> from wbia.scripts.thesis import * # NOQA # >>> expt_name = ut.get_argval('–expt', type_=str, pos=1) # >>> dbnames = ut.get_argval(('–dbs', '–db'), type_=list, default=[]) # >>> Chap3.draw(expt_name, dbnames)

classmethod draw_serial(*expt_name*, *dbnames*, **args*)

ensure_results(*expt_name=None*, *nocompute=None*)
Subclasses must obey the measure_<expt_name>, draw_<expt_name> contract

ensure_setup()

classmethod measure(*expt_name*, *dbnames*, **args*)
CommandLine: python -m wbia Chap3.measure all –dbs=GZ_Master1 python -m wbia Chap3.measure all –dbs=PZ_Master1

python -m wbia Chap3.measure nsum –dbs=GZ_Master1,PZ_Master1 python -m wbia Chap3.measure foregroundness –dbs=GZ_Master1,PZ_Master1

Example: # >>> # Script # >>> from wbia.scripts.thesis import * # NOQA # >>> expt_name = ut.get_argval('–expt', type_=str, pos=1) # >>> dbnames = ut.get_argval(('–dbs', '–db'), type_=list, default=[]) # >>> ChapX.measure(expt_name, dbnames)

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

classmethod vd()
CommandLine: python -m wbia Chap3.vd

class wbia.scripts._thesis_helpers.**ExpandingSample**(*qaids*, *dname_encs*, *confusor_pool*)

Bases: utool.util_dev.NiceRepr

expand(*denc_per_name=[1], extra_dbsize_fracs=[0]*)

class wbia.scripts._thesis_helpers.**Tabular**(*data=None*, *colfmt=None*, *hline=None*, *caption=''*, *index=True*, *escape=True*)

Bases: object

add_multicolumn_header(*size_col_name*)

size_col_name is a list of tuples indicating the number of columns, column format, and text.

as_parts()

as_table(*caption=None*)

as_tabular()

as_text()

rrr(*verbose=True*, *reload_module=True*)

special class reloading function This function is often injected as rrr of classes

wbia.scripts._thesis_helpers.**ave_str**(*mean*, *std*, *precision=2*)

wbia.scripts._thesis_helpers.**dbname_to_species_nice**(*dbname*)

wbia.scripts._thesis_helpers.**find_minority_class_ccs**(*infr*)

wbia.scripts._thesis_helpers.**join_tabular**(*parts*, *hline=False*, *align=True*)

wbia.scripts._thesis_helpers.**split_tabular**(*text*)

wbia.scripts._thesis_helpers.**test_mcc**()

wbia.scripts._thesis_helpers.**upper_one**(*s*)

1.12.4 wbia.scripts.classify_shark module

class wbia.scripts.classify_shark.**ClfProblem**(*ds*)

Bases: object

Harness for researching a classification problem

classifier_test(*clf*, *test_idx*)

fit_new_classifier(*train_idx*)

References

http://leon.bottou.org/research/stochastic http://blog.explainmydata.com/2012/06/ntrain-24853-ntest-25147-ncorrupt.html http://scikit-learn.org/stable/modules/svm.html#svm-classification http://scikit-learn.org/stable/modules/grid_search.html

fit_new_linear_svm(*train_idx*)

gen_crossval_idxs(*n_folds=2*)

gridsearch_linear_svm_params(*train_idx*)

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.classify_shark import *  # NOQA
>>> from wbia.scripts import classify_shark
>>> ds = classify_shark.get_sharks_dataset('binary')
>>> problem = classify_shark.ClfProblem(ds)
>>> problem.print_support_info()
```

print_support_info()

stratified_2sample_idxs(*frac=0.2*, *split_frac=0.75*)

class wbia.scripts.classify_shark.**ClfSingleResult**(*ds=None*, *test_idx=None*, *y_true=None*, *y_pred=None*, *y_conf=None*)

Bases: object

Reports the results of a classification problem

Example

```
>>> # DISABLE_DOCTEST
>>> result = ClfSingleResult()
```

compile_results()

print_report()

class wbia.scripts.classify_shark.**WhaleSharkInjuryModel**

Bases: object

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.classify_shark import *  # NOQA
>>> from wbia.scripts import classify_shark
>>> ds = classify_shark.get_sharks_dataset('binary', 'chip')
>>> problem = classify_shark.ClfProblem(ds)
>>> problem.print_support_info()
>>> ibs = ds.ibs
```

augment(*Xb*, *yb=None*)

X_valid, y_valid = dataset.subset('valid') num = 10 Xb = X_valid[:num] Xb = Xb / 255.0 if ut.is_int(Xb) else Xb Xb = Xb.astype(np.float32, copy=True) yb = None if yb is None else yb.astype(np.int32, copy=True) # Rescale the batch data to the range 0 to 1 **Xb_**, **yb_** = model.augment(Xb) **yb_** = None >>> ut.quit_if_noshow() >>> import wbia.plottool as pt >>> pt.qt4ensure() >>> from wbia_cnn import augment >>> augment.show_augmented_patches(Xb, **Xb_**, yb, **yb_**, data_per_label=1) >>> ut.show_if_requested()

def_inception()

def_lenet()

def_resnet()

init_arch(*verbose=False*, ***kwargs*)

CommandLine: python -m wbia.scripts.classify_shark WhaleSharkInjuryModel.init_arch python -m wbia.scripts.classify_shark WhaleSharkInjuryModel.init_arch –show

python -m wbia.scripts.classify_shark shark_net –dry –show python -m wbia.scripts.classify_shark shark_net –vd

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.classify_shark import *  # NOQA
>>> verbose = True
>>> data_shape = tuple(ut.get_argval('--datashape', type_=list,
>>>                                  default=(224, 224, 3)))
>>> model = WhaleSharkInjuryModel(batch_size=64, output_dims=2,
>>>                               data_shape=data_shape)
>>> model.init_arch()
>>> model.print_model_info_str()
>>> ut.quit_if_noshow()
>>> model.show_arch(fullinfo=False)
>>> ut.show_if_requested()
```

special_output()

wbia.scripts.classify_shark.**get_model_state**(*clf*)

wbia.scripts.classify_shark.**get_shark_dataset**(*target_type='binary'*, *data_type='chip'*)

```
>>> from wbia.scripts.classify_shark import *  # NOQA
>>> target_type = 'binary'
>>> data_type = 'hog'
>>> dataset = get_shark_dataset(target_type)
```

wbia.scripts.classify_shark.**get_shark_labels_and_metadata**(*target_type=None*, *ibs=None*, *config=None*)

```
>>> from wbia.scripts.classify_shark import *  # NOQA
>>> target_type = 'multiclass3'
>>> data_type = 'hog'
```

wbia.scripts.classify_shark.**inspect_results**(*ds*, *result_list*)

wbia.scripts.classify_shark.**predict_svc_ovr**(*clf*, *data*)

wbia.scripts.classify_shark.**predict_ws_injury_interim_svm**(*ibs*, *aids*, ***kwargs*)
Returns relative confidence

wbia.scripts.classify_shark.**set_model_state**(*clf*, *model_state*)

wbia.scripts.classify_shark.**shark_net**(*dry=False*)
CommandLine: python -m wbia.scripts.classify_shark shark_net python -m wbia.scripts.classify_shark shark_net –dry python -m wbia.scripts.classify_shark shark_net –vd –monitor

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.classify_shark import *  # NOQA
>>> shark_net()
```

wbia.scripts.classify_shark.**shark_svm**()

References

http://scikit-learn.org/stable/model_selection.html

Todo:
- Change unreviewed healthy tags to healthy-likely

CommandLine: python -m wbia.scripts.classify_shark shark_svm –show python -m wbia.scripts.classify_shark shark_svm

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.classify_shark import *  # NOQA
>>> shark_svm()
>>> ut.show_if_requested()
```

1.12.5 wbia.scripts.fix_annotation_orientation_issue module

wbia.scripts.fix_annotation_orientation_issue.**fix_annotation_orientation**(*ibs*, *min_percentage=0.95*)

Fixes the annotations that are outside the bounds of the image due to a changed image orientation flag in the database

CommandLine: python -m wbia.scripts.fix_annotation_orientation_issue fix_annotation_orientation

Example

```
>>> # ENABLE_DOCTEST
>>> import wbia
>>> from wbia.scripts.fix_annotation_orientation_issue import *  # NOQA
>>> ibs = wbia.opendb()
>>> unfixable_gid_list = fix_annotation_orientation(ibs)
>>> assert len(unfixable_gid_list) == 0
```

1.12.6 wbia.scripts.getshark module

wbia.scripts.getshark.**add_new_images**(*ibs*, *miss_info*, *species*)

wbia.scripts.getshark.**check_annot_disagree**(*single_info*, *single_annots*, *key1*, *prop2*, *repl2*, *is_set*, *key2=None*, *DRY=True*)

wbia.scripts.getshark.**download_missing_images**(*parsed*, *num=None*)

wbia.scripts.getshark.**get_injur_categories**(*single_annots*, *verbose=False*)

wbia.scripts.getshark.**get_injured_tags**(*tags_list*, *include_healthy=False*, *invert=False*)
tags_list = single_info['tags'] tags_list = single_annots.case_tags info_injur_tags = parse_injury_categories() annot_injur_tags = parse_injury_categories(single_annots.case_tags)

wbia.scripts.getshark.**parse_shark_fname_tags**(*orig_fname_list*, *dev=False*)
Parses potential tags from the filename. If dev mode is on, then it prints out other potential tags you might add.

```
>>> orig_fname_list = parsed['orig_fname']
>>> dev = True
>>> tags = parse_shark_fname_tags(orig_fname_list, dev=dev)
```

wbia.scripts.getshark.**parse_whaleshark_org**()
Read list of all images from wildbook

Combines old and new

```
>>> from wbia.scripts.getshark import *  # NOQA
```

wbia.scripts.getshark.**parse_whaleshark_org_keywords**()

wbia.scripts.getshark.**parse_whaleshark_org_old**()

wbia.scripts.getshark.**parse_wildbook**(*images_url*, *keyword_url=None*)
Read list of all images from wildbook

Combines old and new

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.getshark import *  # NOQA
>>> url = images_url = 'http://www.mantamatcher.org/listImages.jsp'
```

Example

```
>>> # DISABLE_DOCTEST
>>> images_url = 'http://www.whaleshark.org/listImages.jsp'
>>> keyword_url = 'http://www.whaleshark.org/getKeywordImages.jsp'
```

wbia.scripts.getshark.**parse_wildbook_images**(*url*)

Example

```
>>> # DISABLE_DOCTEST
>>> url = 'www.whaleshark.org/listImages.jsp'
>>> url = images_url = 'http://www.mantamatcher.org/listImages.jsp'
>>> parse_wildbook_images(url)
```

wbia.scripts.getshark.**postprocess_corrupted**(*parsed_dl*)

wbia.scripts.getshark.**postprocess_extfilter**(*parsed*)

wbia.scripts.getshark.**postprocess_filenames**(*parsed*, *download_dir*)

wbia.scripts.getshark.**postprocess_rectify_duplicates**(*unmerged*)
Rectify duplicate uuid information

wbia.scripts.getshark.**postprocess_tags_build**(*parsed*)

wbia.scripts.getshark.**postprocess_tags_filter**(*parsed*)

wbia.scripts.getshark.**postprocess_uuids**(*parsed_dl*)

wbia.scripts.getshark.**sync_annot_info**(*ibs*, *single_annots*, *single_info*, *species*, *DRY*)
sync *info* from wildbook into *annots* from IA.

wbia.scripts.getshark.**sync_existing_images**(*ibs*, *hit_info*, *species*, *DRY*)

wbia.scripts.getshark.**sync_wildbook**()
MAIN ENTRY POINT

Syncronizes our wbia database with a wildbook database like whaleshark.org

#cd ~/work/WS_ALL python -m wbia.scripts.getshark

cd /media/raid/raw/WhaleSharks_WB/

```
>>> from wbia.scripts.getshark import *  # NOQA
```

1.12.7 wbia.scripts.getshark_old module

wbia.scripts.getshark_old.**detect_sharks**(*ibs*, *gids*)

wbia.scripts.getshark_old.**get_injured_sharks**()

```
>>> from wbia.scripts.getshark import *  # NOQA
```

wbia.scripts.getshark_old.**purge_ensure_one_annot_per_images**(*ibs*)
pip install Pipe

wbia.scripts.getshark_old.**shark_misc**()

wbia.scripts.getshark_old.**train_part_detector**()
Problem: healthy sharks usually have a mostly whole body shot injured sharks usually have a close up shot. This distribution of images is likely what the injur-shark net is picking up on.
The goal is to train a detector that looks for things that look like the distribution of injured sharks.

We will run this on healthy sharks to find the parts of

1.12.8 wbia.scripts.labelShark module

wbia.scripts.labelShark.**classifyShark**(*ibs*, *gid_list*)

1.12.9 wbia.scripts.name_recitifer module

wbia.scripts.name_recitifer.**find_consistent_labeling**(*grouped_oldnames*, *extra_prefix='_extra_name'*, *verbose=False*)
Solves a a maximum bipirtite matching problem to find a consistent name assignment that minimizes the number of annotations with different names. For each new grouping of annotations we assign

For each group of annotations we must assign them all the same name, either from

To reduce the running time

Parameters **grouped_oldnames** (*list*) – A group of old names where the grouping is based on new names. For instance:

Given: aids = [1, 2, 3, 4, 5] old_names = [0, 1, 1, 1, 0] new_names = [0, 0, 1, 1, 0]

The grouping is [[0, 1, 0], [1, 1]]

This lets us keep the old names in a split case and re-use exising names and make minimal changes to current annotation names while still being consistent with the new and improved grouping.

The output will be: [0, 1]

Meaning that all annots in the first group are assigned the name 0 and all annots in the second group are assigned the name 1.

References

http://stackoverflow.com/questions/1398822/assignment-problem-numpy

CommandLine: python -m wbia.scripts.name_recitifer find_consistent_labeling

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.scripts.name_recitifer import *  # NOQA
>>> grouped_oldnames = testdata_oldnames(25, 15,  5, n_per_incon=5)
>>> new_names = find_consistent_labeling(grouped_oldnames, verbose=1)
>>> grouped_oldnames = testdata_oldnames(0, 15,  5, n_per_incon=1)
>>> new_names = find_consistent_labeling(grouped_oldnames, verbose=1)
>>> grouped_oldnames = testdata_oldnames(0, 0, 0, n_per_incon=1)
>>> new_names = find_consistent_labeling(grouped_oldnames, verbose=1)
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.scripts.name_recitifer import *  # NOQA
>>> ydata = []
>>> xdata = list(range(10, 150, 50))
>>> for x in xdata:
>>>     print('x = %r' % (x,))
>>>     grouped_oldnames = testdata_oldnames(x, 15,  5, n_per_incon=5)
>>>     t = ut.Timerit(3, verbose=1)
>>>     for timer in t:
>>>         with timer:
>>>             new_names = find_consistent_labeling(grouped_oldnames)
>>>     ydata.append(t.ave_secs)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.qtensure()
>>> pt.multi_plot(xdata, [ydata])
>>> ut.show_if_requested()
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.scripts.name_recitifer import *  # NOQA
>>> grouped_oldnames = [['a', 'b', 'c'], ['b', 'c'], ['c', 'e', 'e']]
>>> new_names = find_consistent_labeling(grouped_oldnames, verbose=1)
>>> result = ut.repr2(new_names)
>>> print(new_names)
['a', 'b', 'e']
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.scripts.name_recitifer import *  # NOQA
>>> grouped_oldnames = [['a', 'b'], ['a', 'a', 'b'], ['a']]
>>> new_names = find_consistent_labeling(grouped_oldnames)
>>> result = ut.repr2(new_names)
>>> print(new_names)
['b', 'a', '_extra_name0']
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.scripts.name_recitifer import *  # NOQA
>>> grouped_oldnames = [['a', 'b'], ['e'], ['a', 'a', 'b'], [], ['a'], ['d']]
>>> new_names = find_consistent_labeling(grouped_oldnames)
>>> result = ut.repr2(new_names)
>>> print(new_names)
['b', 'e', 'a', '_extra_name0', '_extra_name1', 'd']
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.scripts.name_recitifer import *  # NOQA
>>> grouped_oldnames = [[], ['a', 'a'], [],
>>>                     ['a', 'a', 'a', 'a', 'a', 'a', 'a', 'b'], ['a']]
>>> new_names = find_consistent_labeling(grouped_oldnames)
>>> result = ut.repr2(new_names)
>>> print(new_names)
['_extra_name0', 'a', '_extra_name1', 'b', '_extra_name2']
```

wbia.scripts.name_recitifer.**find_consistent_labeling_old**(*grouped_oldnames*, *extra_prefix='_extra_name'*, *verbose=False*)

wbia.scripts.name_recitifer.**reasign_names1**(*ibs*, *aid_list=None*, *old_img2_names=None*, *common_prefix=''*)

Changes the names in the IA-database to correspond to an older naming convention. If splits and merges were preformed tries to find the maximally consistent renaming scheme.

Notes

For each annotation: * get the image * get the image full path * strip the full path down to the file name prefix:
[example /foo/bar/pic.jpg -> pic]

- make the name of the individual associated with that annotation be the file name prefix
- save the new names to the image analysis database
- wildbook will make a request to get all of the annotations, image file names, image names and animal ids

CommandLine: python -m wbia.scripts.name_recitifer rectify_names –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.name_recitifer import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = None
>>> common_prefix = ''
>>> old_img2_names = None #['img_fred.png', ']
>>> result = reasign_names1(ibs, aid_list, img_list, name_list)
```

wbia.scripts.name_recitifer.**reasign_names2**(*ibs*, *gname_name_pairs*, *aid_list=None*)

Notes

- Given a list of pairs: image file names (full path), animal name.
- Go through all the images in the database and create a dictionary

that associates the file name (full path) of the image in the database with the
annotation or annotations associated with that image.

- Go through the list of pairs: For each image file name, look up in the dictionary the image file name and assign the annotation associated with the image file name the animal name
- Throughout this, keep a list of annotations that have been changed
- Wildbook will issue a pull request to get these annotation.

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.name_recitifer import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = None
>>> common_prefix = ''
>>> gname_name_pairs = [
>>>     ('easy1.JPG', 'easy'),
>>>     ('easy2.JPG', 'easy'),
>>>     ('easy3.JPG', 'easy'),
>>>     ('hard1.JPG', 'hard')
>>> ]
>>> changed_pairs = reasign_names2(gname_name_pairs)
```

wbia.scripts.name_recitifer.**simple_munkres**(*part_oldnames*)
Defines a munkres problem to solve name rectification.

Notes

We create a matrix where each rows represents a group of annotations in the same PCC and each column represents an original name. If there are more PCCs than original names the columns are padded with extra values. The matrix is first initialized to be negative infinity representing impossible assignments. Then for each column representing a padded name, we set we its value to 1 indicating that each new name could be assigned to a padded name for some small profit. Finally, let f_{rc} be the the number of annotations in row r with an original name of c. Each matrix value (r, c) is set to $f_{rc} + 1$ if $f_{rc} > 0$, to represent how much each name "wants" to be labeled with a particular original name, and the extra one ensures that these original names are always preferred over padded names.
CommandLine: python -m wbia.scripts.name_recitifer simple_munkres

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.scripts.name_recitifer import *  # NOQA
>>> part_oldnames = [['a', 'b'], ['b', 'c'], ['c', 'a', 'a']]
>>> new_names = simple_munkres(part_oldnames)
>>> result = ut.repr2(new_names)
>>> print(new_names)
['b', 'c', 'a']
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.scripts.name_recitifer import *  # NOQA
>>> part_oldnames = [[], ['a', 'a'], [],
>>>                  ['a', 'a', 'a', 'a', 'a', 'a', 'a', 'b'], ['a']]
>>> new_names = simple_munkres(part_oldnames)
>>> result = ut.repr2(new_names)
>>> print(new_names)
[None, 'a', None, 'b', None]
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.scripts.name_recitifer import *  # NOQA
>>> part_oldnames = [[], ['b'], ['a', 'b', 'c'], ['b', 'c'], ['c', 'e', 'e']]
>>> new_names = find_consistent_labeling(part_oldnames)
>>> result = ut.repr2(new_names)
>>> print(new_names)
['_extra_name0', 'b', 'a', 'c', 'e']
```

Profit Matrix b a c e _0

0 -10 -10 -10 -10 1 1 2 -10 -10 -10 1 2 2 2 2 -10 1 3 2 -10 2 -10 1 4 -10 -10 2 3 1

wbia.scripts.name_recitifer.**testdata_oldnames**(*n_incon_groups=10*, *n_con_groups=2*, *n_per_con=5*, *n_per_incon=5*, *con_sep=4*, *n_empty_groups=0*)

1.12.10 wbia.scripts.postdoc module

class wbia.scripts.postdoc.**GraphExpt**(*dbname=None*)
Bases: *wbia.scripts._thesis_helpers.DBInputs*

Todo:

- **[ ] Experimental analysis of duration of each phase and state of** graph.
- **[ ] Experimental analysis of phase 3, including how far we can get** with automatic decision making and do we discover new merges? If there are potential merges, can we run phase iii with exactly the same ordering as before: ordering by probability for automatically decidable and then by positive probability for others. This should work for phase 3 and therefore allow a clean combination of the three phases and our termination criteria. I just thought of this so don't really have it written cleanly above.
- **[ ] Experimental analysis of choice of automatic decision thresholds.** by lowering the threshold we increase the risk of mistakes. Each mistake costs some number of manual reviews (perhaps 2-3), but if the frequency of errors is low then we could be saving ourselves a lot of manual reviews.

item OTHER SPECIES

CommandLine: python -m wbia GraphExpt.measure all PZ_MTEST
Ignore:

```
>>> from wbia.scripts.postdoc import *
>>> self = GraphExpt('PZ_MTEST')
>>> self._precollect()
>>> self._setup()
```

base_dpath = '/home/docs/Desktop/graph_expt'

draw_graphsim()
CommandLine:
python -m wbia GraphExpt.measure graphsim GZ_Master1 python -m wbia GraphExpt.draw graphsim GZ_Master1 –diskshow

python -m wbia GraphExpt.draw graphsim PZ_MTEST –diskshow python -m wbia GraphExpt.draw graphsim GZ_Master1 –diskshow python -m wbia GraphExpt.draw graphsim PZ_Master1 –diskshow

Ignore:

```
>>> from wbia.scripts.postdoc import *
>>> self = GraphExpt('GZ_Master1')
>>> self = GraphExpt('PZ_MTEST')
```

draw_graphsim2()
CommandLine: python -m wbia GraphExpt.draw graphsim2 –db PZ_MTEST –diskshow python -m wbia GraphExpt.draw graphsim2 GZ_Master1 –diskshow python -m wbia GraphExpt.draw graphsim2 PZ_Master1 –diskshow

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.thesis import *
>>> dbname = ut.get_argval('--db', default='GZ_Master1')
>>> self = GraphExpt(dbname)
```

(continues on next page)

(continued from previous page)

```
>>> self.draw_graphsim2()
>>> ut.show_if_requested()
```

measure_all()

measure_graphsim()

CommandLine: python -m wbia GraphExpt.measure graphsim GZ_Master1 1

Ignore:

```
>>> from wbia.scripts.postdoc import *
>>> #self = GraphExpt('PZ_MTEST')
>>> #self = GraphExpt('GZ_Master1')
>>> self = GraphExpt.measure('graphsim', 'PZ_Master1')
>>> self = GraphExpt.measure('graphsim', 'GZ_Master1')
>>> self = GraphExpt.measure('graphsim', 'PZ_MTEST')
```

rrr(*verbose=True*, *reload_module=True*)

special class reloading function This function is often injected as rrr of classes

class wbia.scripts.postdoc.**VerifierExpt**(*dbname=None*)

Bases: *wbia.scripts._thesis_helpers.DBInputs*

Collect data from experiments to visualize

python -m wbia VerifierExpt.measure all PZ_Master1.GZ_Master1,GIRM_Master1,MantaMatcher,RotanTurtles,humpbacks_fb,L
python -m wbia VerifierExpt.measure all GIRM_Master1,PZ_Master1,LF_ALL python -m wbia VerifierExpt.measure all LF_ALL python -m wbia VerifierExpt.measure all PZ_Master1

python -m wbia VerifierExpt.measure all MantaMatcher python -m wbia VerifierExpt.draw all MantaMatcher

python -m wbia VerifierExpt.draw rerank PZ_Master1

python -m wbia VerifierExpt.measure all RotanTurtles python -m wbia VerifierExpt.draw all RotanTurtles

Ignore:

```
>>> from wbia.scripts.postdoc import *
>>> fpath = ut.glob(ut.truepath('~/Desktop/mtest_plots'), '*.pkl')[0]
>>> self = ut.load_data(fpath)
```

agg_dbnames = ['PZ_Master1', 'GZ_Master1', 'MantaMatcher', 'RotanTurtles', 'humpbacks_

classmethod agg_dbstats()

CommandLine: python -m wbia VerifierExpt.agg_dbstats python -m wbia VerifierExpt.measure_dbstats

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.postdoc import *  # NOQA
>>> result = VerifierExpt.agg_dbstats()
>>> print(result)
```

classmethod agg_results(*task_key*)

python -m wbia VerifierExpt.agg_results python -m wbia VerifierExpt.agg_results –link link-paper-final

GZ_Master1,LF_ALL,MantaMatcher,RotanTurtles,humpbacks_fb,GIRM_Master1

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.postdoc import *  # NOQA
>>> task_key = 'match_state'
>>> result = VerifierExpt.agg_results(task_key)
>>> print(result)
```

base_dpath = '/home/docs/latex/crall-iccvw-2017/figures'

custom_single_hard_case()

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.postdoc import *
>>> defaultdb = 'PZ_PB_RF_TRAIN'
>>> #defaultdb = 'GZ_Master1'
>>> defaultdb = 'PZ_MTEST'
>>> self = VerifierExpt.collect(defaultdb)
>>> self.dbname = 'PZ_PB_RF_TRAIN'
```

draw_all()

CommandLine: python -m wbia VerifierExpt.draw_all –db PZ_MTEST python -m wbia VerifierExpt.draw_all –db PZ_PB_RF_TRAIN python -m wbia VerifierExpt.draw_all –db GZ_Master1 python -m wbia VerifierExpt.draw_all –db PZ_Master1

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.postdoc import *
>>> dbname = ut.get_argval('--db', default='PZ_MTEST')
>>> dbnames = ut.get_argval('--dbs', type_=list, default=[dbname])
>>> for dbname in dbnames:
>>>     print('dbname = %r' % (dbname,))
>>>     self = VerifierExpt(dbname)
>>>     self.draw_all()
```

draw_class_score_hist()

Plots distribution of positive and negative scores

draw_hard_cases(*task_key='match_state'*)

draw hard cases with and without overlay

python -m wbia VerifierExpt.draw hard_cases GZ_Master1 match_state python -m wbia VerifierExpt.draw hard_cases PZ_Master1 match_state python -m wbia VerifierExpt.draw hard_cases PZ_Master1 photobomb_state python -m wbia VerifierExpt.draw hard_cases GZ_Master1 photobomb_state

python -m wbia VerifierExpt.draw hard_cases RotanTurtles match_state

```
>>> from wbia.scripts.postdoc import *
>>> self = VerifierExpt('PZ_MTEST')
>>> task_key = 'match_state'
>>> self.draw_hard_cases(task_key)
```

draw_mcc_thresh(*task_key*)

python -m wbia VerifierExpt.draw mcc_thresh GZ_Master1 match_state python -m wbia VerifierExpt.draw mcc_thresh PZ_Master1 match_state

python -m wbia VerifierExpt.draw mcc_thresh GZ_Master1 photobomb_state python -m wbia VerifierExpt.draw mcc_thresh PZ_Master1 photobomb_state

draw_rerank()

draw_roc(*task_key='match_state'*)

python -m wbia VerifierExpt.draw roc GZ_Master1 photobomb_state python -m wbia VerifierExpt.draw roc GZ_Master1 match_state

python -m wbia VerifierExpt.draw roc PZ_MTEST

classmethod draw_tagged_pair()

measure_all()

CommandLine: python -m wbia VerifierExpt.measure all GZ_Master1,MantaMatcher,RotanTurtles,LF_ALL python -m wbia VerifierExpt.measure all GZ_Master1

Ignore: from wbia.scripts.postdoc import * self = VerifierExpt('PZ_MTEST') self.measure_all()

measure_dbstats()

python -m wbia VerifierExpt.measure dbstats GZ_Master1 python -m wbia VerifierExpt.measure dbstats PZ_Master1 python -m wbia VerifierExpt.measure dbstats MantaMatcher python -m wbia VerifierExpt.measure dbstats RotanTurtles

Ignore:

```
>>> from wbia.scripts.postdoc import *
>>> #self = VerifierExpt('GZ_Master1')
>>> self = VerifierExpt('MantaMatcher')
```

measure_hard_cases(*task_key*)

Find a failure case for each class

CommandLine: python -m wbia VerifierExpt.measure hard_cases GZ_Master1 match_state python -m wbia VerifierExpt.measure hard_cases GZ_Master1 photobomb_state python -m wbia VerifierExpt.draw hard_cases GZ_Master1 match_state python -m wbia VerifierExpt.draw hard_cases GZ_Master1 photobomb_state

python -m wbia VerifierExpt.measure hard_cases PZ_Master1 match_state python -m wbia VerifierExpt.measure hard_cases PZ_Master1 photobomb_state python -m wbia VerifierExpt.draw hard_cases PZ_Master1 match_state python -m wbia VerifierExpt.draw hard_cases PZ_Master1 photobomb_state

python -m wbia VerifierExpt.measure hard_cases PZ_MTEST match_state python -m wbia VerifierExpt.draw hard_cases PZ_MTEST photobomb_state

python -m wbia VerifierExpt.draw hard_cases RotanTurtles match_state python -m wbia VerifierExpt.draw hard_cases MantaMatcher match_state

Ignore:

```
>>> task_key = 'match_state'
>>> task_key = 'photobomb_state'
>>> from wbia.scripts.postdoc import *
>>> self = VerifierExpt('GZ_Master1')
>>> self._setup()
```

measure_rerank()

```
>>> from wbia.scripts.postdoc import *
>>> defaultdb = 'PZ_Master1'
>>> defaultdb = 'GZ_Master1'
>>> self = VerifierExpt(defaultdb)
>>> self._setup()
>>> self.measure_rerank()
```

measure_thresh(*pblm*)

ranking_hyperparamm_search()

```
>>> from wbia.scripts.postdoc import *
>>> self = VerifierExpt('humpbacks_fb')
```

```
>>> self = VerifierExpt('MantaMatcher')
```

```
>>> self = VerifierExpt('RotanTurtles')
```

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

task_nice_lookup = {'match_state': {'match': 'Positive', 'nomatch': 'Negative', 'no

write_metrics(*task_key='match_state'*)
Writes confusion matricies
CommandLine: python -m wbia VerifierExpt.draw metrics PZ_PB_RF_TRAIN match_state python -m wbia VerifierExpt.draw metrics GZ_Master1 photobomb_state

python -m wbia VerifierExpt.draw metrics PZ_Master1,GZ_Master1 photobomb_state,match_state
Ignore:

```
>>> from wbia.scripts.postdoc import *
>>> self = VerifierExpt('PZ_Master1')
>>> task_key = 'match_state'
```

write_sample_info()
python -m wbia VerifierExpt.draw sample_info GZ_Master1

wbia.scripts.postdoc.**draw_match_states**()

wbia.scripts.postdoc.**entropy_potential**(*infr*, *u*, *v*, *decision*)
Returns the number of edges this edge would invalidate

from wbia.algo.graph import demo infr = demo.demodata_infr(pcc_sizes=[5, 2, 4, 2, 2, 1, 1, 1]) infr.refresh_candidate_edges() infr.params['redun.neg'] = 1 infr.params['redun.pos'] = 1 infr.apply_nondynamic_update()

ut.qtensure() infr.show(show_cand=True, groupby='name_label')

u, v = 1, 7 decision = 'positive'

wbia.scripts.postdoc.**plot_cmcs**(*cdfs*, *labels*, *fnum=1*, *pnum=(1, 1, 1)*, *ymin=0.4*)

wbia.scripts.postdoc.**prepare_cdfs**(*cdfs*, *labels*)

wbia.scripts.postdoc.**review_pz**()

1.12.11 wbia.scripts.rsync_wbiadb module

CommandLine: python -m wbia.scripts.rsync_wbiadb python -m wbia.scripts.rsync_wbiadb –dryrun

wbia.scripts.rsync_wbiadb.**rsync_ibsdb_main**()

wbia.scripts.rsync_wbiadb.**sync_wbiadb**(*remote_uri*, *dbname*, *mode='pull'*, *workdir=None*, *port=22*, *dryrun=False*)

syncs an wbiadb without syncing the cache or the chip directory (or the top level image directory because it shouldnt exist unlese it is an old hots database)

1.12.12 wbia.scripts.specialdraw module

wbia.scripts.specialdraw.**double_depcache_graph**()

CommandLine: python -m wbia.scripts.specialdraw double_depcache_graph –show –testmode

python -m wbia.scripts.specialdraw double_depcache_graph –save=figures5/doubledepc.png –dpath ~/latex/cand/ –diskshow –figsize=8,20 –dpi=220 –testmode –show –clipwhite python -m wbia.scripts.specialdraw double_depcache_graph –save=figures5/doubledepc.png –dpath ~/latex/cand/ –diskshow –figsize=8,20 –dpi=220 –testmode –show –clipwhite –arrow-width=.5

python -m wbia.scripts.specialdraw double_depcache_graph –save=figures5/doubledepc.png –dpath ~/latex/cand/ –diskshow –figsize=8,20 –dpi=220 –testmode –show –clipwhite –arrow-width=5

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.specialdraw import *  # NOQA
>>> result = double_depcache_graph()
>>> print(result)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

wbia.scripts.specialdraw.**draw_graph_id**()

CommandLine: python -m wbia.scripts.specialdraw draw_graph_id –show

wbia.scripts.specialdraw.**draw_inconsistent_pcc**()

CommandLine: python -m wbia.scripts.specialdraw draw_inconsistent_pcc –show

wbia.scripts.specialdraw.**event_space**()

pip install matplotlib-venn

wbia.scripts.specialdraw.**featweight_fig**()

CommandLine: python -m wbia.scripts.specialdraw featweight_fig –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.specialdraw import *  # NOQA
>>> featweight_fig()
>>> ut.show_if_requested()
```

wbia.scripts.specialdraw.**general_identify_flow**()

CommandLine: python -m wbia.scripts.specialdraw general_identify_flow –show –save pairsim.png –dpi=100 –diskshow –clipwhite

python -m wbia.scripts.specialdraw general_identify_flow –dpi=200 –diskshow –clipwhite –dpath ~/latex/cand/ –figsize=20,10 –save figures4/pairprob.png –arrow-width=2.0

Example

```
>>> # SCRIPT
>>> from wbia.scripts.specialdraw import *  # NOQA
>>> general_identify_flow()
>>> ut.quit_if_noshow()
>>> ut.show_if_requested()
```

wbia.scripts.specialdraw.**graph_iden_cut_demo**()

CommandLine: python -m wbia.scripts.specialdraw graph_iden_cut_demo –show –precut python -m wbia.scripts.specialdraw graph_iden_cut_demo –show –postcut

python -m wbia.scripts.specialdraw graph_iden_cut_demo –precut –save=precut.png –clipwhite python -m wbia.scripts.specialdraw graph_iden_cut_demo –postcut –save=postcut.png –clipwhite

Example

```
>>> # SCRIPT
>>> from wbia.scripts.specialdraw import *  # NOQA
>>> graph_iden_cut_demo()
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

wbia.scripts.specialdraw.**graphcut_flow**()

Returns name

Return type

?

CommandLine: python -m wbia.scripts.specialdraw graphcut_flow –show python -m wbia.scripts.specialdraw graphcut_flow –show –save cutflow.png –diskshow –clipwhite python -m wbia.scripts.specialdraw graphcut_flow –save figures4/cutiden.png –diskshow –clipwhite –dpath ~/latex/crall-candidacy-2015/ –figsize=24,10 –arrow-width=2.0

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.specialdraw import *  # NOQA
>>> graphcut_flow()
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

wbia.scripts.specialdraw.**intraoccurrence_connected**()

CommandLine: python -m wbia.scripts.specialdraw intraoccurrence_connected –show python -m wbia.scripts.specialdraw intraoccurrence_connected –show –smaller

python -m wbia.scripts.specialdraw intraoccurrence_connected –precut –save=precut.jpg python -m wbia.scripts.specialdraw intraoccurrence_connected –postcut –save=postcut.jpg

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.specialdraw import *  # NOQA
>>> result = intraoccurrence_connected()
>>> print(result)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

wbia.scripts.specialdraw.**k_redun_demo**()

python -m wbia.scripts.specialdraw k_redun_demo –save=kredun.png python -m wbia.scripts.specialdraw k_redun_demo –show

Example

```
>>> # SCRIPT
>>> from wbia.scripts.specialdraw import *  # NOQA
>>> k_redun_demo()
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

wbia.scripts.specialdraw.**lighten_hex**(*hexcolor*, *amount*)

wbia.scripts.specialdraw.**merge_viewpoint_graph**()

CommandLine: python -m wbia.scripts.specialdraw merge_viewpoint_graph –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.specialdraw import *  # NOQA
>>> result = merge_viewpoint_graph()
>>> print(result)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

wbia.scripts.specialdraw.**multidb_montage**()

CommandLine: python -m wbia.scripts.specialdraw multidb_montage –save montage.jpg –dpath ~/slides –diskshow –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.specialdraw import *  # NOQA
>>> multidb_montage()
```

wbia.scripts.specialdraw.**nx_makenode**(*graph*, *name*, ***attrkw*)

wbia.scripts.specialdraw.**redun_demo2**()
python -m wbia.scripts.specialdraw redun_demo2 –show

wbia.scripts.specialdraw.**redun_demo3**()
python -m wbia.scripts.specialdraw redun_demo3 –show python -m wbia.scripts.specialdraw redun_demo3 –saveparts=~/slides/incon_redun.jpg –dpi=300

wbia.scripts.specialdraw.**scalespace**()
THIS DOES NOT SHOW A REAL SCALE SPACE PYRAMID YET. FIXME.

Returns imgBGRA_warped

Return type

?

CommandLine: python -m wbia.scripts.specialdraw scalespace –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.specialdraw import *  # NOQA
>>> imgBGRA_warped = scalespace()
>>> result = ('imgBGRA_warped = %s' % (ut.repr2(imgBGRA_warped),))
>>> print(result)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

wbia.scripts.specialdraw.**setcover_example**()
CommandLine: python -m wbia.scripts.specialdraw setcover_example –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.specialdraw import *  # NOQA
>>> result = setcover_example()
>>> print(result)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

wbia.scripts.specialdraw.**show_id_graph**()
CommandLine: python -m wbia.scripts.specialdraw show_id_graph –show python -m wbia.scripts.specialdraw show_id_graph –show

Example

```
>>> # SCRIPT
>>> from wbia.scripts.specialdraw import *  # NOQA
>>> show_id_graph()
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

wbia.scripts.specialdraw.**simple_vsone_matches**()
CommandLine:

python -m wbia.scripts.specialdraw simple_vsone_matches –show –db GZ_Master1 –aids=2811,2810

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.specialdraw import *  # NOQA
>>> simple_vsone_matches()
>>> ut.show_if_requested()
```

wbia.scripts.specialdraw.**system_diagram**()
CommandLine: python -m wbia.scripts.specialdraw system_diagram –show

1.12.13 wbia.scripts.thesis module

class wbia.scripts.thesis.**Chap3**(*dbname=None*)
Bases: *wbia.scripts._thesis_helpers.DBInputs*, *wbia.scripts.thesis.Chap3Draw*, *wbia.scripts.thesis.Chap3Measures*

classmethod agg_dbstats()
CommandLine: python -m wbia Chap3.agg_dbstats python -m wbia Chap3.measure_dbstats

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.thesis import *  # NOQA
>>> result = Chap3.agg_dbstats()
>>> print(result)
```

base_dpath = '/home/docs/latex/crall-thesis-2017/figures3'

classmethod draw_agg_baseline()
CommandLine: python -m wbia Chap3.draw_agg_baseline –diskshow

Example

```
>>> # SCRIPT
>>> from wbia.scripts.thesis import *  # NOQA
>>> Chap3.draw_agg_baseline()
```

measure_all()

Example

from wbia.scripts.thesis import * self = Chap3('PZ_Master1') self.measure_all() self = Chap3('GZ_Master1') self.measure_all() self = Chap3('GIRM_Master1') self.measure_all()

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

classmethod run_all()
CommandLine: python -m wbia Chap3.run_all

class wbia.scripts.thesis.**Chap3Draw**
Bases: object

draw_all()
CommandLine: python -m wbia Chap3.draw_all –dbs=GZ_Master1,PZ_Master1,GIRM_Master1

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.thesis import *  # NOQA
>>> dbname = ut.get_argval('--db', default='PZ_MTEST')
>>> dbnames = ut.get_argval('--dbs', type_=list, default=[dbname])
>>> for dbname in dbnames:
>>>     print('dbname = %r' % (dbname,))
>>>     self = Chap3(dbname)
>>>     self.draw_all()
```

draw_baseline()

draw_foregroundness()
wbia Chap3.measure foregroundness –dbs=GZ_Master1,PZ_Master1 wbia Chap3.draw foregroundness –dbs=GZ_Master1,PZ_Master1

draw_invar()
wbia Chap3.measure invar –dbs=GZ_Master1,PZ_Master1 wbia Chap3.draw invar –dbs=GZ_Master1,PZ_Master1

draw_kexpt()
wbia Chap3.measure kexpt –dbs=GZ_Master1,PZ_Master1 wbia Chap3.draw kexpt –dbs=GZ_Master1,PZ_Master1 –diskshow

draw_nsum()
wbia Chap3.measure nsum –dbs=GZ_Master1,PZ_Master1 wbia Chap3.draw nsum –dbs=GZ_Master1,PZ_Master1

draw_nsum_simple()
wbia Chap3.measure nsum –dbs=GZ_Master1,PZ_Master1 wbia Chap3.draw nsum –dbs=GZ_Master1,PZ_Master1
Ignore:

```
>>> from wbia.scripts.thesis import *  # NOQA
>>> self = Chap3('PZ_Master1')
```

draw_smk()
wbia Chap3.measure smk –dbs=GZ_Master1,PZ_Master1 wbia Chap3.draw smk –dbs=GZ_Master1,PZ_Master1

draw_time_distri()
CommandLine: python -m wbia Chap3.draw_time_distri –dbs=GZ_Master1,PZ_Master1,GIRM_MasterV python -m wbia Chap3.draw_time_distri –dbs=GIRM_Master1 python -m wbia Chap3.draw_time_distri –dbs=GZ_Master1 python -m wbia Chap3.draw_time_distri –dbs=PZ_Master1 python -m wbia Chap3.draw_time_distri –dbs=humpbacks_fb

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.thesis import *  # NOQA
>>> dbname = ut.get_argval('--db', default='PZ_MTEST')
>>> dbnames = ut.get_argval('--dbs', type_=list, default=[dbname])
>>> for dbname in dbnames:
>>>     print('dbname = %r' % (dbname,))
>>>     self = Chap3(dbname)
>>>     self.draw_time_distri()
```

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

class wbia.scripts.thesis.**Chap3Measures**
Bases: object

draw_foregroundness_intra()
python -m wbia Chap3.measure foregroundness_intra –dbs=GZ_Master1,PZ_Master1 python -m wbia Chap3.draw foregroundness_intra –dbs=GZ_Master1,PZ_Master1 –diskshow

measure_baseline()

```
>>> from wbia.scripts.thesis import *
>>> self = Chap3('GZ_Master1')
>>> self._precollect()
```

measure_dbsize()

measure_dbstats()

measure_foregroundness()

measure_foregroundness_intra()

measure_invar()

measure_kexpt()

measure_nsum()
python -m wbia Chap3.measure nsum –dbs=GZ_Master1,PZ_Master1 python -m wbia Chap3.draw nsum –dbs=GZ_Master1,PZ_Master1 –diskshow

from wbia.scripts.thesis import * self = Chap3('GZ_Master1') self = Chap3('PZ_Master1') self = Chap3('PZ_MTEST') self._precollect()

measure_smk()
python -m wbia Chap3.measure smk –dbs=GZ_Master1,PZ_Master1 python -m wbia Chap3.draw smk –dbs=GZ_Master1,PZ_Master1 –diskshow

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

class wbia.scripts.thesis.**Chap4**(*dbname=None*)
Bases: *wbia.scripts._thesis_helpers.DBInputs*

Collect data from experiments to visualize

TODO: redo save/loading of measurments
Ignore:

```
>>> from wbia.scripts.thesis import *
>>> fpath = ut.glob(ut.truepath('~/Desktop/mtest_plots'), '*.pkl')[0]
>>> self = ut.load_data(fpath)
```

base_dpath = '/home/docs/latex/crall-thesis-2017/figures4'

custom_single_hard_case()

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.thesis import *
>>> defaultdb = 'PZ_PB_RF_TRAIN'
>>> #defaultdb = 'GZ_Master1'
>>> defaultdb = 'PZ_MTEST'
>>> self = Chap4.collect(defaultdb)
>>> self.dbname = 'PZ_PB_RF_TRAIN'
```

draw_all()

CommandLine: python -m wbia Chap4.draw_all –db PZ_MTEST python -m wbia Chap4.draw_all –db PZ_PB_RF_TRAIN python -m wbia Chap4.draw_all –db GZ_Master1 python -m wbia Chap4.draw_all –db PZ_Master1

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.thesis import *
>>> dbname = ut.get_argval('--db', default='PZ_MTEST')
>>> dbnames = ut.get_argval('--dbs', type_=list, default=[dbname])
>>> for dbname in dbnames:
>>>     print('dbname = %r' % (dbname,))
>>>     self = Chap4(dbname)
>>>     self.draw_all()
```

draw_class_score_hist()

Plots distribution of positive and negative scores

draw_hard_cases(*task_key*)

draw hard cases with and without overlay

python -m wbia Chap4.draw hard_cases GZ_Master1 match_state python -m wbia Chap4.draw hard_cases PZ_Master1 match_state python -m wbia Chap4.draw hard_cases PZ_Master1 photobomb_state python -m wbia Chap4.draw hard_cases GZ_Master1 photobomb_state

```
>>> from wbia.scripts.thesis import *
>>> self = Chap4('PZ_MTEST')
>>> task_key = 'match_state'
>>> self.draw_hard_cases(task_key)
```

draw_mcc_thresh(*task_key*)

python -m wbia Chap4.draw mcc_thresh GZ_Master1 match_state python -m wbia Chap4.draw mcc_thresh PZ_Master1 match_state

python -m wbia Chap4.draw mcc_thresh GZ_Master1 photobomb_state python -m wbia Chap4.draw mcc_thresh PZ_Master1 photobomb_state

draw_prune()

CommandLine: python -m wbia Chap4.draw importance GZ_Master1

python -m wbia Chap4.draw importance PZ_Master1 photobomb_state python -m wbia Chap4.draw importance PZ_Master1 match_state

python -m wbia Chap4.draw prune GZ_Master1,PZ_Master1 python -m wbia Chap4.draw prune PZ_Master1

```
>>> from wbia.scripts.thesis import *
>>> self = Chap4('PZ_Master1')
>>> self = Chap4('GZ_Master1')
>>> self = Chap4('PZ_MTEST')
```

draw_rerank()

draw_roc(*task_key*)

python -m wbia Chap4.draw roc GZ_Master1 photobomb_state python -m wbia Chap4.draw roc GZ_Master1 match_state

classmethod draw_tagged_pair()

draw_wordcloud(*task_key*)

measure_all()

CommandLine: python -m wbia Chap4.measure_all –db PZ_PB_RF_TRAIN python -m wbia Chap4.measure_all –db PZ_MTEST python -m wbia Chap4.measure_all

python -m wbia Chap4.measure_all –db GZ_Master1

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.thesis import *
>>> dbname = ut.get_argval('--db', default='PZ_MTEST')
>>> dbnames = ut.get_argval('--dbs', type_=list, default=[dbname])
>>> for dbname in dbnames:
>>>     print('dbname = %r' % (dbname,))
>>>     self = Chap4(dbname)
>>>     self.measure_all()
```

measure_hard_cases(*task_key*)

Find a failure case for each class

CommandLine: python -m wbia Chap4.measure hard_cases GZ_Master1 match_state python -m wbia Chap4.measure hard_cases GZ_Master1 photobomb_state python -m wbia Chap4.draw hard_cases GZ_Master1 match_state python -m wbia Chap4.draw hard_cases GZ_Master1 photobomb_state

python -m wbia Chap4.measure hard_cases PZ_Master1 match_state python -m wbia Chap4.measure hard_cases PZ_Master1 photobomb_state python -m wbia Chap4.draw hard_cases PZ_Master1 match_state python -m wbia Chap4.draw hard_cases PZ_Master1 photobomb_state

python -m wbia Chap4.measure hard_cases PZ_MTEST match_state python -m wbia Chap4.draw hard_cases PZ_MTEST photobomb_state

python -m wbia Chap4.measure hard_cases MantaMatcher match_state

Ignore:

```
>>> task_key = 'match_state'
>>> task_key = 'photobomb_state'
>>> from wbia.scripts.thesis import *
>>> self = Chap4('GZ_Master1')
>>> self._setup()
```

measure_prune()

```
>>> from wbia.scripts.thesis import *
>>> self = Chap4('GZ_Master1')
>>> self = Chap4('PZ_Master1')
>>> self = Chap4('PZ_MTEST')
```

measure_rerank()

```
>>> from wbia.scripts.thesis import *
>>> defaultdb = 'PZ_Master1'
>>> defaultdb = 'GZ_Master1'
>>> self = Chap4(defaultdb)
>>> self._setup()
>>> self.measure_rerank()
```

measure_thresh(*pblm*)

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

task_nice_lookup = {'match_state': {'match': 'Positive', 'nomatch': 'Negative', 'no

write_importance(*task_key*)
python -m wbia Chap4.draw importance GZ_Master1,PZ_Master1 match_state

python -m wbia Chap4.draw importance GZ_Master1 match_state python -m wbia Chap4.draw importance PZ_Master1 match_state

python -m wbia Chap4.draw importance GZ_Master1 photobomb_state python -m wbia Chap4.draw importance PZ_Master1 photobomb_state

write_metrics(*task_key='match_state'*)
CommandLine: python -m wbia Chap4.draw metrics PZ_PB_RF_TRAIN match_state python -m wbia Chap4.draw metrics GZ_Master1 photobomb_state

python -m wbia Chap4.draw metrics PZ_Master1,GZ_Master1 photobomb_state,match_state
Ignore:

```
>>> from wbia.scripts.thesis import *
>>> self = Chap4('PZ_Master1')
>>> task_key = 'match_state'
```

write_metrics2(*task_key='match_state'*)
CommandLine: python -m wbia Chap4.draw metrics PZ_PB_RF_TRAIN match_state python -m wbia Chap4.draw metrics2 PZ_Master1 photobomb_state python -m wbia Chap4.draw metrics2 GZ_Master1 photobomb_state

python -m wbia Chap4.draw metrics2 GZ_Master1 photobomb_state

write_sample_info()
python -m wbia Chap4.draw sample_info GZ_Master1

class wbia.scripts.thesis.**Chap5**(*dbname=None*)

Bases: *wbia.scripts._thesis_helpers.DBInputs*

python -m wbia Chap5.measure all GZ_Master1 python -m wbia Chap5.measure all PZ_Master1 python -m wbia Chap5.draw all GZ_Master1 python -m wbia Chap5.draw all PZ_Master1 –comp Leviathan

python -m wbia Chap5.draw error_graph_analysis GZ_Master1 python -m wbia Chap5.draw error_graph_analysis PZ_Master1 –comp Leviathan

base_dpath = '/home/docs/latex/crall-thesis-2017/figures5'

draw_all()

CommandLine: python -m wbia Chap5.draw all GZ_Master1 python -m wbia Chap5.draw error_graph_analysis GZ_Master1

python -m wbia Chap5.draw all PZ_Master1 python -m wbia Chap5.draw error_graph_analysis PZ_Master1

Ignore:

```
>>> from wbia.scripts.thesis import *
>>> self = Chap4('GZ_Master1')
```

draw_error_graph_analysis()

CommandLine: python -m wbia Chap5.draw error_graph_analysis GZ_Master1 python -m wbia Chap5.draw error_graph_analysis PZ_Master1

Ignore:

```
>>> from wbia.scripts.thesis import *
>>> self = Chap5('GZ_Master1')
>>> self = Chap5('PZ_Master1')
```

draw_refresh()

CommandLine: python -m wbia Chap5.draw refresh GZ_Master1 –diskshow python -m wbia Chap5.draw refresh PZ_Master1 –diskshow

draw_simulation()

CommandLine: python -m wbia Chap5.draw simulation PZ_MTEST –diskshow python -m wbia Chap5.draw simulation GZ_Master1 –diskshow python -m wbia Chap5.draw simulation PZ_Master1 –diskshow

Ignore:

```
>>> from wbia.scripts.thesis import *
>>> self = Chap5('GZ_Master')
```

draw_simulation2()

CommandLine: python -m wbia Chap5.draw_simulation2 –db PZ_MTEST –show python -m wbia Chap5.draw_simulation2 –db GZ_Master1 –show python -m wbia Chap5.draw_simulation2 –db PZ_Master1 –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.scripts.thesis import *
>>> dbname = ut.get_argval('--db', default='GZ_Master1')
>>> self = Chap5(dbname)
>>> self.draw_simulation2()
>>> ut.show_if_requested()
```

measure_all()

measure_dbstats()
python -m wbia Chap5.draw dbstats GZ_Master1

python -m wbia Chap5.measure dbstats PZ_Master1 python -m wbia Chap5.draw dbstats PZ_Master1
Ignore:

```
>>> from wbia.scripts.thesis import *
>>> self = Chap5('GZ_Master1')
```

measure_simulation()
CommandLine: python -m wbia Chap5.measure simulation GZ_Master1 python -m wbia Chap5.measure simulation PZ_Master1
Ignore:

```
>>> from wbia.scripts.thesis import *
>>> self = Chap5('GZ_Master1')
```

print_error_analysis()
Ignore:

```
>>> from wbia.scripts.thesis import *
>>> self = Chap5('GZ_Master1')
>>> self = Chap5('PZ_Master1')
```

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

write_dbstats()
TODO: write info about what dataset was used
CommandLine: python -m wbia Chap5.measure dbstats PZ_Master1 python -m wbia Chap5.measure dbstats PZ_Master1

python -m wbia Chap5.measure simulation GZ_Master1 python -m wbia Chap5.draw dbstats –db GZ_Master1 –diskshow
Ignore:

```
>>> from wbia.scripts.thesis import *
>>> self = Chap5('GZ_Master1')
```

write_error_tables()
CommandLine: python -m wbia Chap5.draw error_tables PZ_Master1 python -m wbia Chap5.draw error_tables GZ_Master1
Ignore:

```
>>> from wbia.scripts.thesis import *
>>> from wbia.scripts.thesis import _ranking_hist, _ranking_cdf
>>> self = Chap5('GZ_Master1')
```

class wbia.scripts.thesis.**Sampler**
Bases: object

class wbia.scripts.thesis.**SplitSample**(*qaids*, *daids*)
Bases: utool.util_dev.NiceRepr

wbia.scripts.thesis.**feat_alias**(*k*)

wbia.scripts.thesis.**label_alias**(*k*)

wbia.scripts.thesis.**plot_cmcs**(*cdfs*, *labels*, *fnum=1*, *pnum=(1, 1, 1)*, *ymin=0.4*)

wbia.scripts.thesis.**plot_cmcs2**(*cdfs*, *labels*, *fnum=1*, ***kwargs*)

wbia.scripts.thesis.**prepare_cdfs**(*cdfs*, *labels*)

1.12.14 Module contents

1.13 wbia.templates package

1.13.1 Submodules

1.13.2 wbia.templates.generate_notebook module

CommandLine: # Generate and start an IPython notebook python -m wbia –tf autogen_ipynb –ipynb –db <dbname> [-a <acfg>] [-t <pcfg>]

python -m wbia –tf autogen_ipynb –ipynb –db seaturtles -a default2:qhas_any=(left,right),sample_occur=True,occur_offset=[0,1,2

CommandLine: # to connect to a notebook on a remote machine that does not have the # appropriate port exposed you must start an SSH tunnel. # Typically a jupyter-notebook runs on port 8888. # Run this command on your local machine. ssh -N -f -L localhost:<local_port>:localhost:<remote_port> <remote_user>@<remote_host>

E.G. ssh -N -f -L localhost:8889:localhost:8888 joncrall@hyrule.cs.rpi.edu # Now you can connect locally firefox localhost:8889

Running a server: jupyter-notebook password jupyter-notebook –no-browser –NotebookApp.iopub_data_rate_limit=100000000 –NotebookApp.token=

To allow remote jupyter-noteobok connections jupyter notebook –generate-config

Really need to do jupyter hub

need to set c.NotebookApp.port = 8888 c.NotebookApp.open_browser = False c.NotebookApp.ip = '*'

wbia.templates.generate_notebook.**autogen_ipynb**(*ibs*, *launch=None*, *run=None*)

Autogenerates standard IBEIS Image Analysis IPython notebooks.

CommandLine: python -m wbia autogen_ipynb –run –db lynx python -m wbia autogen_ipynb –run –db lynx

python -m wbia autogen_ipynb –ipynb –db PZ_MTEST -p :proot=smk,num_words=64000 default python -m wbia autogen_ipynb –ipynb –db PZ_MTEST –asreport python -m wbia autogen_ipynb –ipynb –db PZ_MTEST –noexample –withtags python -m wbia autogen_ipynb –ipynb –db PZ_MTEST

python -m wbia autogen_ipynb –ipynb –db STS_SandTigers

python -m wbia autogen_ipynb –db PZ_MTEST # TODO: Add support for dbdir to be specified python -m wbia autogen_ipynb –db ~/work/PZ_MTEST

python -m wbia autogen_ipynb –ipynb –db Oxford -a default:qhas_any=(query,),dpername=1,exclude_reference=True,dmin
python -m wbia autogen_ipynb –ipynb –db PZ_MTEST -a default -t best:lnbnn_normalizer=[None,normlnbnn-test]

python -m wbia.templates.generate_notebook –exec-autogen_ipynb –db wd_peter_blinston –ipynb

python -m wbia autogen_ipynb –db PZ_Master1 –ipynb python -m wbia autogen_ipynb –db PZ_Master1 -a timectrl:qindex=0:100 -t best best:normsum=True –ipynb –noexample python -m wbia autogen_ipynb –db PZ_Master1 -a timectrl –run jupyter-notebook Experiments-lynx.ipynb killall python

python -m wbia autogen_ipynb –db humpbacks –ipynb -t default:proot=BC_DTW -a default:has_any=hasnotch python -m wbia autogen_ipynb –db humpbacks –ipynb -t default:proot=BC_DTW default:proot=vsmany -a default:has_any=hasnotch,mingt=2,qindex=0:50 –noexample

python -m wbia autogen_ipynb –db testdb_curvrank –ipynb -t default:proot=CurvRankDorsal python -m wbia autogen_ipynb –db testdb_curvrank –ipynb -t default:proot=CurvRankFluke python -m wbia autogen_ipynb –db PW_Master –ipynb -t default:proot=CurvRankDorsal

python -m wbia autogen_ipynb –db testdb_identification –ipynb -t default:proot=Deepsense

Ignore: python -m wbia autogen_ipynb –db WS_ALL

Example

```
>>> # SCRIPT
>>> from wbia.templates.generate_notebook import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> result = autogen_ipynb(ibs)
>>> print(result)
```

wbia.templates.generate_notebook.**get_default_cell_template_list**(*ibs*)

Defines the order of ipython notebook cells

wbia.templates.generate_notebook.**make_wbia_cell_list**(*ibs*)

wbia.templates.generate_notebook.**make_wbia_notebook**(*ibs*)

Parameters **ibs** (*wbia.IBEISController*) – wbia controller object

CommandLine: python -m wbia.templates.generate_notebook –exec-make_wbia_notebook –db wd_peter_blinston –asreport python -m wbia –tf –exec-make_wbia_notebook python -m wbia –tf make_wbia_notebook –db lynx jupyter-notebook tmp.ipynb runipy tmp.ipynb –html report.html runipy –pylab tmp.ipynb tmp2.ipynb sudo pip install runipy python -c "import runipy; print(runipy.__version__)"

Example

```
>>> # SCRIPT
>>> from wbia.templates.generate_notebook import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> notebook_str = make_wbia_notebook(ibs)
>>> print(notebook_str)
```

1.13.3 wbia.templates.notebook_cells module

ComamndLine: python -m wbia –tf autogen_ipynb –ipynb –db PZ_MTEST –ipynb

wbia.templates.notebook_cells.**dataset_summary_stats_hacktest**()

import wbia ibs = wbia.opendb('WWF_Lynx_Copy') # import wbia # ibs = wbia.opendb('WWF_Lynx') a = [
'default:max_timestamp=now,minqual=good,require_timestamp=True,view=left,dcrossval_enc=1,joinme=1',
'default:max_timestamp=now,minqual=good,require_timestamp=True,view=left,dcrossval_enc=2,joinme=2',
'default:max_timestamp=now,minqual=good,require_timestamp=True,view=left,dcrossval_enc=3,joinme=3',
'default:max_timestamp=now,minqual=good,require_timestamp=True,view=left,dcrossval_enc=4,joinme=4',

'default:max_timestamp=now,minqual=good,require_timestamp=True,view=right,dcrossval_enc=1,joinme=1',
'default:max_timestamp=now,minqual=good,require_timestamp=True,view=right,dcrossval_enc=2,joinme=2',
'default:max_timestamp=now,minqual=good,require_timestamp=True,view=right,dcrossval_enc=3,joinme=3',
'default:max_timestamp=now,minqual=good,require_timestamp=True,view=right,dcrossval_enc=4,joinme=4',
] acfg_list, expanded_aids_list = wbia.expt.experiment_helpers.get_annotcfg_list(
ibs, acfg_name_list=a, verbose=0)
expt_aids = sorted(set(ut.total_flatten(expanded_aids_list))) print(ut.repr2(ibs.get_annot_stats_dict(expt_aids, strkeys=True, nl=2, use_hist=True)))

expt_qaids = sorted(set(ut.total_flatten(ut.take_column(expanded_aids_list, 0)))) # print(ut.repr2(ibs.get_annot_stats_dict(expt_qaids, strkeys=True, nl=2, use_hist=True)))

1.13.4 wbia.templates.notebook_helpers module

wbia.templates.notebook_helpers.**custom_globals**()

wbia.templates.notebook_helpers.**make_cells_wider**()

1.13.5 Module contents

wbia.templates.**IMPORT_TUPLES = [('generate_notebook', None), ('notebook_cells', None)]**
cd /home/joncrall/code/wbia/wbia/templates makeinit.py –modname=wbia.templates
Type Regen Command

wbia.templates.**reassign_submodule_attributes**(*verbose=True*)
why reloading all the modules doesnt do this I don't know

wbia.templates.**reload_subs**(*verbose=True*)
Reloads wbia.templates and submodules

wbia.templates.**rrrr**(*verbose=True*)
Reloads wbia.templates and submodules

1.14 wbia.viz package

1.14.1 Subpackages

1.14.1.1 wbia.viz.interact package

1.14.1.1.1 Submodules

1.14.1.1.2 wbia.viz.interact.interact_annotations2 module

1.14.1.1.3 wbia.viz.interact.interact_chip module

1.14.1.1.4 wbia.viz.interact.interact_image module

1.14.1.1.5 wbia.viz.interact.interact_matches module

1.14.1.1.6 wbia.viz.interact.interact_name module

1.14.1.1.7 wbia.viz.interact.interact_qres module

1.14.1.1.8 wbia.viz.interact.interact_query_decision module

1.14.1.1.9 wbia.viz.interact.interact_sver module

1.14.1.1.10 Module contents

1.14.2 Submodules

1.14.3 wbia.viz.viz_chip module

wbia.viz.viz_chip.**HARDCODE_SHOW_PB_PAIR**()
python -m wbia.viz.viz_chip HARDCODE_SHOW_PB_PAIR –show

Example

```
>>> # SCRIPT
>>> from wbia.viz.viz_chip import *  # NOQA
>>> import wbia.plottool as pt
>>> HARDCODE_SHOW_PB_PAIR()
>>> pt.show_if_requested()
```

wbia.viz.viz_chip.**show_chip**(*ibs*, *aid*, *in_image=False*, *annote=True*, *title_suffix=''*, *weight_label=None*, *weights=None*, *config2_=None*, ***kwargs*)
Driver function to show chips

Parameters

- **ibs** (*wbia.IBEISController*) –
- **aid** (*int*) – annotation rowid
- **in_image** (*bool*) – displays annotation with the context of its source image
- **annote** (*bool*) – enables overlay annoations

- **title_suffix** (*str*) –
- **weight_label** (*None*) – (default = None)
- **weights** (*None*) – (default = None)
- **config2** (*dict*) – (default = None)

Kwargs: enable_chip_title_prefix, nokpts, kpts_subset, kpts, text_color, notitle, draw_lbls, show_aidstr, show_gname, show_name, show_nid, show_exemplar, show_num_gt, show_quality_text, show_viewcode, fnum, title, figtitle, pnum, interpolation, cmap, heatmap, data_colorbar, darken, update, xlabel, redraw_image, ax, alpha, docla, doclf, projection, pts, ell color (3/4-tuple, ndarray, or str): colors for keypoints

CommandLine: python -m wbia.viz.viz_chip show_chip –show –ecc python -c "import utool as ut; ut.print_auto_docstr('wbia.viz.viz_chip', 'show_chip')" python -m wbia.viz.viz_chip show_chip –show –db NNP_Master3 –aids 14047 –no-annote python -m wbia.viz.viz_chip show_chip –show –db NNP_Master3 –aids 14047 –no-annote

python -m wbia.viz.viz_chip show_chip –show –db PZ_MTEST –aid 1 –bgmethod=cnn python -m wbia.viz.viz_chip show_chip –show –db PZ_MTEST –aid 1 –bgmethod=cnn –scale_max=30

python -m wbia.viz.viz_chip show_chip –show –db PZ_MTEST –aid 1 –ecc –draw_lbls=False –notitle –save=~/slides/lnbnn_query.jpg –dpi=300

Example

```
>>> # xdoctest: +REQUIRES(module:wbia_cnn)
>>> # VIZ_TEST
>>> from wbia.viz.viz_chip import *  # NOQA
>>> import numpy as np
>>> import vtool as vt
>>> in_image = False
>>> ibs, aid_list, kwargs, config2_ = testdata_showchip()
>>> aid = aid_list[0]
>>> if True:
>>>     import matplotlib as mpl
>>>     from wbia.scripts.thesis import TMP_RC
>>>     mpl.rcParams.update(TMP_RC)
>>> if ut.get_argflag('--ecc'):
>>>     kpts = ibs.get_annot_kpts(aid, config2_=config2_)
>>>     weights = ibs.get_annot_fgweights([aid], ensure=True, config2_=config2_
↪)[0]
>>>     kpts = ut.random_sample(kpts[weights > .9], 200, seed=0)
>>>     ecc = vt.get_kpts_eccentricity(kpts)
>>>     scale = 1 / vt.get_scales(kpts)
>>>     #s = ecc if config2_.affine_invariance else scale
>>>     s = scale
>>>     colors = pt.scores_to_color(s, cmap_='jet')
>>>     kwargs['color'] = colors
>>>     kwargs['kpts'] = kpts
>>>     kwargs['ell_linewidth'] = 3
>>>     kwargs['ell_alpha'] = .7
>>> show_chip(ibs, aid, in_image=in_image, config2_=config2_, **kwargs)
>>> pt.show_if_requested()
```

wbia.viz.viz_chip.**show_many_chips**(*ibs*, *aid_list*, *config2_=None*, *fnum=None*, *pnum=None*, *vert=True*)

CommandLine: python -m wbia.viz.viz_chip –test-show_many_chips python -m wbia.viz.viz_chip –test-show_many_chips –show python -m wbia.viz.viz_chip –test-show_many_chips –show –db NNP_Master3 –aids=13276,14047,14489,14906,10194,10201,12656,10150,11002,15315,7191,13127,15591,12838,13970

–no-annote –dpath figures –save ~/latex/crall-candidacy-2015/figures/challengechips.jpg '–caption=challenging images'

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.viz.viz_chip import *  # NOQA
>>> import numpy as np
>>> in_image = False
>>> ibs, aid_list, kwargs, config2_ = testdata_showchip()
>>> # execute function
>>> show_many_chips(ibs, aid_list, config2_)
>>> ut.show_if_requested()
```

wbia.viz.viz_chip.**testdata_showchip**()

1.14.4 wbia.viz.viz_graph module

1.14.5 wbia.viz.viz_graph2 module

1.14.6 wbia.viz.viz_helpers module

wbia.viz.viz_helpers.**get_aidstrs**(*aid_list*, ***kwargs*)

wbia.viz.viz_helpers.**get_annot_kpts_in_imgspace**(*ibs*, *aid_list*, *config2_=None*, *ensure=True*)

Transforms keypoints so they are plotable in imagespace

wbia.viz.viz_helpers.**get_annot_text**(*ibs*, *aid_list*, *draw_lbls*)

wbia.viz.viz_helpers.**get_annot_texts**(*ibs*, *aid_list*, ***kwargs*)

Add each type of text_list to the strings list

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*int*) – list of annotation ids

Returns annotation_text_list

Return type list

CommandLine: python -m wbia.viz.viz_helpers –test-get_annot_texts

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.viz.viz_helpers import *  # NOQA
>>> import wbia
>>> import collections
>>> ibs = wbia.opendb('testdb1')
>>> # Default all kwargs to true
>>> class KwargsProxy(object):
...     def get(self, a, b):
...         return True
>>> kwargs_proxy = KwargsProxy()
>>> aid_list = ibs.get_valid_aids()[::3]
>>> # execute function
```

(continues on next page)

(continued from previous page)

```
>>> annotation_text_list = get_annot_texts(ibs, aid_list, kwargs_proxy=kwargs_
↪proxy)
>>> # verify results
>>> result = ut.repr2(annotation_text_list, nl=1)
>>> print(result)
[
    'aid1, gname=easy1.JPG, name=____, nid=-1, , nGt=0, quality=UNKNOWN, view=left
↪',
    'aid4, gname=hard1.JPG, name=____, nid=-4, , nGt=0, quality=UNKNOWN, view=left
↪',
    'aid7, gname=jeff.png, name=jeff, nid=3, EX, nGt=0, quality=UNKNOWN,␣
↪view=unknown',
    'aid10, gname=occl2.JPG, name=occl, nid=5, EX, nGt=0, quality=UNKNOWN,␣
↪view=left',
    'aid13, gname=zebra.jpg, name=zebra, nid=7, EX, nGt=0, quality=UNKNOWN,␣
↪view=unknown',
]
```

wbia.viz.viz_helpers.**get_bbox_centers**(*bbox_list*)

wbia.viz.viz_helpers.**get_bboxes**(*ibs*, *aid_list*, *offset_list=None*)

wbia.viz.viz_helpers.**get_chips**(*ibs*, *aid_list*, *in_image=False*, *config2_=None*, *as_fpath=False*)

wbia.viz.viz_helpers.**get_image_titles**(*ibs*, *gid_list*)

wbia.viz.viz_helpers.**get_kpts**(*ibs*, *aid_list*, *in_image=False*, *config2_=None*, *ensure=True*, *kpts_subset=None*, *kpts=None*)

wbia.viz.viz_helpers.**get_nidstrs**(*nid_list*, ***kwargs*)

wbia.viz.viz_helpers.**get_query_text**(*ibs*, *cm*, *aid2*, *truth*, ***kwargs*)

returns title based on the query chip and result

Parameters

- **ibs** (IBEISController) – wbia controller object
- **cm** (ChipMatch) – object of feature correspondences and scores
- **aid2** (*int*) – annotation id
- **truth** (*int*) – 0, 1, 2

Kwargs: qaid, score, rawscore, aid2_raw_rank, show_truth, name_score, name_rank, show_name_score, show_name_rank, show_timedelta

Returns query_text

Return type str

CommandLine: python -m wbia.viz.viz_helpers –exec-get_query_text

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.viz.viz_helpers import *  # NOQA
>>> import wbia
>>> cm, qreq_ = wbia.testdata_cm()
>>> aid2 = cm.get_top_aids()[0]
>>> truth = 1
>>> query_text = get_query_text(ibs, cm, aid2, truth)
>>> result = ('query_text = %s' % (str(query_text),))
>>> print(result)
```

wbia.viz.viz_helpers.**get_timedelta_str**(*ibs*, *aid1*, *aid2*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid1** (*int*) – annotation id
- **aid2** (*int*) – annotation id

Returns timedelta_str

Return type str

CommandLine: python -m wbia.viz.viz_helpers –test-get_timedelta_str

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.viz.viz_helpers import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid1, aid2 = 1, 8
>>> timedelta_str = get_timedelta_str(ibs, aid1, aid2)
>>> result = str(timedelta_str)
>>> print(result)
td(2 hours 28 minutes 22 seconds)
```

td(+2:28:22) td(02:28:22)

wbia.viz.viz_helpers.**get_truth_color**(*truth*, *base255=False*, *lighten_amount=None*)

wbia.viz.viz_helpers.**get_vsstr**(*qaid*, *aid*)

wbia.viz.viz_helpers.**is_unknown**(*ibs*, *nid_list*)

wbia.viz.viz_helpers.**kp_info**(*kp*)

wbia.viz.viz_helpers.**register_FNUMS**(*FNUMS_*)

wbia.viz.viz_helpers.**show_keypoint_gradient_orientations**(*ibs*, *aid*, *fx*, *fnum=None*, *pnum=None*, *config2_=None*)

1.14.7 wbia.viz.viz_hough module

wbia.viz.viz_hough.**show_hough_image**(*ibs*, *gid*, *species=None*, *fnum=None*, ***kwargs*)

wbia.viz.viz_hough.**show_probability_chip**(*ibs*, *aid*, *species=None*, *fnum=None*, *config2_=None*, *blend=False*, ***kwargs*)

TODO: allow species override in controller

CommandLine: python -m wbia.viz.viz_hough –exec-show_probability_chip –cnn –show python -m wbia.viz.viz_hough –exec-show_probability_chip –cnn –show –db PZ_Master1 python -m wbia.viz.viz_hough –exec-show_probability_chip –cnn –show –db PZ_Master1 –aid 9970

Example

```
>>> # SCRIPT
>>> from wbia.viz.viz_hough import *  # NOQA
>>> import wbia
>>> from wbia.viz import viz_chip
>>> ibs, aid_list, kwargs, config2_ = viz_chip.testdata_showchip()
```

(continues on next page)

(continued from previous page)

```
>>> fnum = 1
>>> species = None
>>> aid = aid_list[0]
>>> fig, ax = show_probability_chip(ibs, aid, species, fnum, blend=True, **kwargs)
>>> ut.show_if_requested()
```

1.14.8 wbia.viz.viz_image module

wbia.viz.viz_image.**draw_image_overlay**(*ibs*, *ax*, *gid*, *sel_aids*, *draw_lbls=True*, *annote=True*)

wbia.viz.viz_image.**drive_test_script**(*ibs*)

Test script where we drive around and take pictures of animals both in a given database and not in a given databse to make sure the system works.

CommandLine: python -m wbia.viz.viz_image –test-drive_test_script python -m wbia.viz.viz_image –test-drive_test_script –db PZ_MTEST –show python -m wbia.viz.viz_image –test-drive_test_script –db GIR_Tanya –show python -m wbia.viz.viz_image –test-drive_test_script –db GIR_Master0 –show python -m wbia.viz.viz_image –test-drive_test_script –db PZ_Master0 –show python -m wbia.viz.viz_image –test-drive_test_script –db PZ_FlankHack –show

python -m wbia.viz.viz_image –test-drive_test_script –db PZ_FlankHack –show python -m wbia.viz.viz_image –test-drive_test_script –dbdir /raid/work2/GIR_Master –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.viz.viz_image import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb()
>>> drive_test_script(ibs)
```

wbia.viz.viz_image.**get_annot_annotations**(*ibs*, *aid_list*, *sel_aids=[]*, *draw_lbls=True*)

wbia.viz.viz_image.**show_image**(*ibs*, *gid*, *sel_aids=[]*, *fnum=None*, *annote=True*, *draw_lbls=True*, *notitle=False*, *rich_title=False*, *pnum=(1, 1, 1)*, ***kwargs*)

Driver function to show images

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid** (*int*) – image row id
- **sel_aids** (*list*) –
- **fnum** (*int*) – figure number
- **annote** (*bool*) –
- **draw_lbls** (*bool*) –

Returns (fig, ax)

Return type tuple

CommandLine: python -m wbia.viz.viz_image –test-show_image –show python -m wbia.viz.viz_image –test-show_image –show –db GZ_ALL python -m wbia.viz.viz_image –test-show_image –show –db GZ_ALL –gid 100 python -m wbia.viz.viz_image –test-show_image –show –db PZ_MTEST –aid 10

python -m wbia.viz.viz_image –test-show_image –show –db PZ_MTEST –aid 91 –no-annot –rich-title python -m wbia.viz.viz_image –test-show_image –show –db GIR_Tanya –aid 1 –no-annot –rich-title

Example

```
>>> # SLOW_DOCTEST
>>> # VIZ_TEST
>>> from wbia.viz.viz_image import *  # NOQA
>>> import wbia
>>> # build test data
>>> ibs = wbia.opendb(ut.get_argval('--db', str, 'testdb1'))
>>> #gid = ibs.get_valid_gids()[0]
>>> gid = ut.get_argval('--gid', int, 1)
>>> aid = ut.get_argval('--aid', int, None)
>>> if aid is not None:
>>>     gid = ibs.get_annot_gids(aid)
>>> sel_aids = []
>>> fnum = None
>>> annote = not ut.get_argflag('--no-annot')
>>> rich_title = ut.get_argflag('--rich-title')
>>> draw_lbls = True
>>> # execute function
>>> (fig, ax) = show_image(ibs, gid, sel_aids, fnum, annote, draw_lbls, rich_
↪title)
>>> pt.show_if_requested()
```

wbia.viz.viz_image.**show_multi_images**(*ibs*, *gid_list*, *fnum=None*, ***kwargs*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **gid_list** (*list*) –
- **fnum** (*int*) – figure number(default = None)

CommandLine: python -m wbia.viz.viz_image –test-show_multi_images –db NNP_Master3 –gids=7409,7448,4670,7497,7496,7464,7446,7442 –show python -m wbia.viz.viz_image –test-show_multi_images –db NNP_Master3 –gids=1,2,3 –show

Ignore:

```
>>> # print to 8 gids sorted by num aids
>>> import wbia
>>> ibs = wbia.opendb('NNP_Master3')
>>> gid_list = ibs.get_valid_gids()
>>> aids_list = ibs.get_image_aids(gid_list)
>>> index_list = ut.list_argsort(list(map(len, aids_list)))[::-1]
>>> gid_list = ut.take(gid_list, index_list[0:8])
>>> print(','.join(map(str, gid_list)))
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.viz.viz_image import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> gid_list = ut.get_argval('--gids', list, default=[1, 2])
>>> fnum = None
>>> result = show_multi_images(ibs, gid_list, fnum, draw_lbls=False, notitle=True,
↪ sel_aids='all')
>>> print(result)
>>> ut.show_if_requested()
```

1.14.9 wbia.viz.viz_matches module

wbia.viz.viz_matches.**annotate_matches2**(*ibs*, *aid1*, *aid2*, *fm*, *fs*, *offset1=(0, 0)*, *offset2=(0, 0)*, *xywh2=None*, *xywh1=None*, *qreq_=None*, ***kwargs*)

TODO: use this as the main function.

wbia.viz.viz_matches.**annotate_matches3**(*ibs*, *aid_list*, *bbox_list*, *offset_list*, *name_fm_list*, *name_fs_list*, *qreq_=None*, ***kwargs*)

TODO: use this as the main function.

wbia.viz.viz_matches.**get_data_annot_pair_info**(*ibs*, *aid_list*, *qreq_*, *draw_fmatches*, *scale_down=False*, *kpts2_list=None*, *as_fpath=False*)

wbia.viz.viz_matches.**get_multitruth**(*ibs*, *aid_list*)

wbia.viz.viz_matches.**get_query_annot_pair_info**(*ibs*, *qaid*, *qreq_*, *draw_fmatches*, *kpts1=None*, *as_fpath=False*)

wbia.viz.viz_matches.**show_matches**(*ibs*, *cm*, *aid2*, *sel_fm=[]*, *qreq_=None*, ***kwargs*)

DEPRICATE

shows single annotated match result.

Parameters

- **ibs** (IBEISController) –
- **cm** (ChipMatch) – object of feature correspondences and scores
- **aid2** (*int*) – result annotation id
- **sel_fm** (*list*) – selected features match indices

Kwargs: vert (bool)

Returns (ax, xywh1, xywh2)

Return type tuple

wbia.viz.viz_matches.**show_matches2**(*ibs*, *aid1*, *aid2*, *fm=None*, *fs=None*, *fm_norm=None*, *sel_fm=[]*, *H1=None*, *H2=None*, *qreq_=None*, ***kwargs*)

TODO: DEPRICATE and use special case of show_name_matches Integrate ChipMatch

Used in: Found 1 line(s) in '/home/joncrall/code/wbia_cnn/wbia_cnn/ingest_wbia.py': ingest_wbia.py : 827 | >>> wbia.viz.viz_matches.show_matches2(ibs, aid1, aid2, fm=None, kpts1=kpts1, kpts2=kpts2) ————————- Found 4 line(s) in '/home/joncrall/code/wbia/wbia/viz/viz_matches.py': viz_matches.py : 423 |def show_matches2(ibs, aid1, aid2, fm=None, fs=None, fm_norm=None, sel_fm=[], viz_matches.py : 430 | python -m wbia.viz.viz_matches –exec-show_matches2 –show viz_matches.py : 431 | python -m wbia –tf ChipMatch.ishow_single_annotmatch show_matches2 –show viz_matches.py : 515 | return show_matches2(ibs, aid1, aid2, fm, fs, qreq_=qreq_, **kwargs) ————————- Found 1 line(s) in '/home/joncrall/code/wbia/wbia/viz/interact/interact_matches.py': interact_matches.py : 372 | tup = viz.viz_matches.show_matches2(ibs, self.qaid, self.daid, ————————- Found 2 line(s) in '/home/joncrall/code/wbia/wbia/algo/hots/chip_match.py': chip_match.py : 204 | viz_matches.show_matches2(**qreq_**.ibs, cm.qaid, daid, qreq_=qreq_, chip_match.py : 219 | wbia.viz.viz_matches.show_matches2 ————————- Found 1 line(s) in '/home/joncrall/code/wbia/wbia/algo/hots/scoring.py': scoring.py : 562 | viz.viz_matches.show_matches2(**qreq_**.ibs, qaid, daid, fm, fs,

CommandLine: python -m wbia.viz.viz_matches –exec-show_matches2 –show python -m wbia –tf ChipMatch.ishow_single_annotmatch show_matches2 –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.algo.hots.chip_match import *  # NOQA
>>> import wbia
>>> cm, qreq_ = wbia.testdata_cm(defaultdb='PZ_MTEST', default_qaids=[18])
>>> cm.score_name_nsum(qreq_)
>>> daid = cm.get_top_aids()[0]
>>> cm.show_single_annotmatch(qreq_, daid)
>>> ut.show_if_requested()
```

wbia.viz.viz_matches.**show_multichip_match**(*rchip1*, *rchip2_list*, *kpts1*, *kpts2_list*, *fm_list*, *fs_list*, *featflag_list*, *fnum=None*, *pnum=None*, ***kwargs*)

move to df2 rchip = rchip1 H = H1 = None target_wh = None

wbia.viz.viz_matches.**show_name_matches**(*ibs*, *qaid*, *name_daid_list*, *name_fm_list*, *name_fs_list*, *name_H1_list*, *name_featflag_list*, *qreq_=None*, ***kwargs*)

Called from chip_match.py

Parameters

- **ibs** (IBEISController) – wbia controller object
- **qaid** (*int*) – query annotation id
- **name_daid_list** (*list*) –
- **name_fm_list** (*list*) –
- **name_fs_list** (*list*) –
- **name_H1_list** (*list*) –
- **name_featflag_list** (*list*) –
- **qreq** (QueryRequest) – query request object with hyper-parameters(default = None)

Kwargs: draw_fmatches, name_rank, fnum, pnum, colorbar_, nonvote_mode, fastmode, show_matches, fs, fm_norm, lbl1, lbl2, rect, draw_border, cmap, H1, H2, scale_factor1, scale_factor2, draw_pts, draw_ell, draw_lines, show_nMatches, all_kpts, in_image, show_query, draw_lbl, name_annot_scores, score, rawscore, aid2_raw_rank, show_name, show_nid, show_aid, show_annot_score, show_truth, name_score, show_name_score, show_name_rank, show_timedelta

CommandLine: python -m wbia.viz.viz_matches –exec-show_name_matches python -m wbia.viz.viz_matches –test-show_name_matches –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.viz.viz_matches import *  # NOQA
>>> from wbia.algo.hots import chip_match
>>> from wbia.algo.hots import name_scoring
>>> import vtool as vt
>>> from wbia.algo.hots import _pipeline_helpers as plh  # NOQA
>>> import numpy as np
>>> func = chip_match.ChipMatch.show_single_namematch
>>> sourcecode = ut.get_func_sourcecode(func, stripdef=True, stripret=True,
>>>                                     strip_docstr=True)
>>> setup = ut.regex_replace('viz_matches.show_name_matches', '#', sourcecode)
>>> homog = False
>>> print(ut.indent(setup, '>>> '))
>>> ibs, qreq_, cm_list = plh.testdata_post_sver('PZ_MTEST', qaid_list=[1])
```

(continues on next page)

(continued from previous page)

```
>>> cm = cm_list[0]
>>> cm.score_name_nsum(qreq_)
>>> dnid = ibs.get_annot_nids(cm.qaid)
>>> # +--- COPIED SECTION
>>> locals_ = locals()
>>> var_list = ut.exec_func_src(
>>>     func, locals_=locals_,
>>>     sentinal='name_annot_scores = cm.annot_score_list.take(sorted_groupxs')
>>> exec(ut.execstr_dict(var_list))
>>> # L___ COPIED SECTION
>>> kwargs = {}
>>> show_name_matches(ibs, qaid, name_daid_list, name_fm_list,
>>>                   name_fs_list, name_h1_list, name_featflag_list,
>>>                   qreq_=qreq_, **kwargs)
>>> ut.quit_if_noshow()
>>> ut.show_if_requested()
```

1.14.10 wbia.viz.viz_name module

wbia.viz.viz_name.**show_multiple_chips**(*ibs*, *aid_list*, *in_image=True*, *fnum=0*, *sel_aids=[]*, *subtitle=''*, *annote=False*, ***kwargs*)

CommandLine: python -m wbia.viz.viz_name –test-show_multiple_chips –show –no-inimage python -m wbia.viz.viz_name –test-show_multiple_chips –show –db NNP_Master3 –aids=6435,9861,137,6563,9167,12547,9332,12598,13285 –no-inimage –notitle python -m wbia.viz.viz_name –test-show_multiple_chips –show –db NNP_Master3 –aids=137,6563,12547,9332,12598,13285 –no-inimage –notitle –adjust=.05 python -m wbia.viz.viz_name –test-show_multiple_chips –show –db NNP_Master3 –aids=6563,9332,13285,12598 –no-inimage –notitle –adjust=.05 –rc=1,4 python -m wbia.viz.viz_name –test-show_multiple_chips –show –db PZ_Master0 –aids=1288 –no-inimage –notitle –adjust=.05 python -m wbia.viz.viz_name –test-show_multiple_chips –show –db PZ_Master0 –aids=4020,4839 –no-inimage –notitle –adjust=.05

python -m wbia.viz.viz_name –test-show_multiple_chips –db NNP_Master3 –aids=6524,6540,6571,6751 –no-inimage –notitle –adjust=.05 –diskshow

python -m wbia.viz.viz_name –test-show_multiple_chips –db PZ_MTEST -a default:index=0:4 –show –aids=1 –doboth –show –no-inimage

python -m wbia.viz.viz_name –test-show_multiple_chips –db PZ_MTEST –aids=1 –doboth –show –no-inimage python -m wbia.viz.viz_name –test-show_multiple_chips –db PZ_MTEST –aids=1 –doboth –rc=2,1 –show –no-inimage python -m wbia.viz.viz_name –test-show_multiple_chips –db PZ_MTEST –aids=1 –doboth –rc=2,1 –show –notitle –trydrawline –no-draw_lbls python -m wbia.viz.viz_name –test-show_multiple_chips –db PZ_MTEST –aids=1,2 –doboth –show –notitle –trydrawline

python -m wbia.viz.viz_name –test-show_multiple_chips –db PZ_MTEST –aids=1,2,3,4,5 –doboth –rc=2,5 –show –chrlbl –trydrawline –qualtitle –no-figtitle –notitle –doboth –doboth –show

python -m wbia.viz.viz_name –test-show_multiple_chips –db NNP_Master3 –aids=15419 –doboth –rc=2,1 –show –notitle –trydrawline –no-draw_lbls

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.viz.viz_name import *  # NOQA
>>> import wbia
```

(continues on next page)

(continued from previous page)

```
>>> ibs, aid_list, in_image = testdata_multichips()
>>> if True:
>>>     import matplotlib as mpl
>>>     from wbia.scripts.thesis import TMP_RC
>>>     mpl.rcParams.update(TMP_RC)
>>> fnum = 0
>>> sel_aids = []
>>> subtitle = ''
>>> annote = False
>>> fig = show_multiple_chips(ibs, aid_list, in_image, fnum, sel_aids, subtitle,␣
↪annote)
>>> ut.quit_if_noshow()
>>> fig.canvas.draw()
>>> ut.show_if_requested()
```

wbia.viz.viz_name.**show_name**(*ibs*, *nid*, *in_image=True*, *fnum=0*, *sel_aids=[]*, *subtitle=''*, *annote=False*, *aid_list=None*, *index_list=None*, ***kwargs*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **nid** –
- **in_image** (*bool*) –
- **fnum** (*int*) – figure number
- **sel_aids** (*list*) –
- **subtitle** (*str*) –
- **annote** (*bool*) –

CommandLine:

python -m wbia.viz.viz_name –test-show_name –dpath ~/latex/crall-candidacy-2015 –save 'figures/{name}.jpg' –no-figtitle –notitle –db NNP_Master3 –figsize=9,4 –clipwhite –dpi=180 –adjust=.05 –index_list=[0,1,2,3] –rc=2,4 –append temp_out_figure.tex –name=IBEIS_PZ_0739 –no-draw_lbls –doboth –no-inimage –diskshow

python -m wbia.viz.viz_name –test-show_name –no-figtitle –notitle –db NNP_Master3 –figsize=9,4 –clipwhite –dpi=180 –adjust=.05 –index_list=[0,1,2,3] –rc=2,4 –append temp_out_figure.tex –name=IBEIS_PZ_0739 –no-draw_lbls –doboth –no-inimage –show

python -m wbia.viz.viz_name –test-show_name –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.viz.viz_name import *  # NOQA
>>> ibs, nid, in_image, index_list = testdata_showname()
>>> fnum = 0
>>> sel_aids = []
>>> subtitle = ''
>>> annote = False
>>> # execute function
>>> show_name(ibs, nid, in_image, fnum, sel_aids, subtitle, annote, index_
↪list=index_list)
>>> ut.show_if_requested()
```

wbia.viz.viz_name.**show_name_of**(*ibs*, *aid*, ***kwargs*)

wbia.viz.viz_name.**testdata_multichips**()

wbia.viz.viz_name.**testdata_showname**()

1.14.11 wbia.viz.viz_nearest_descriptors module

wbia.viz.viz_nearest_descriptors.**get_annotfeat_nn_index**(*ibs*, *qaid*, *qfx*, *qreq_=None*)

wbia.viz.viz_nearest_descriptors.**show_nearest_descriptors**(*ibs*, *qaid*, *qfx*, *fnum=None*, *stride=5*, *qreq_=None*, ***kwargs*)

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **qaid** (*int*) – query annotation id
- **qfx** (*int*) – query feature index
- **fnum** (*int*) – figure number
- **stride** (*int*) –
- **consecutive_distance_compare** (*bool*) –

CommandLine: # Find a good match to inspect python -m wbia.viz.interact.interact_matches –test-testdata_match_interact –show –db PZ_MTEST –qaid 3

Now inspect it python -m wbia.viz.viz_nearest_descriptors –test-show_nearest_descriptors –show –db PZ_MTEST –qaid 3 –qfx 879 python -m wbia.viz.viz_nearest_descriptors –test-show_nearest_descriptors –show python -m wbia.viz.viz_nearest_descriptors –test-show_nearest_descriptors –db PZ_MTEST –qaid 3 –qfx 879 –diskshow –save foo.png –dpi=256

SeeAlso: plottool.viz_featrow ~/code/plottool/plottool/viz_featrow.py

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.viz.viz_nearest_descriptors import *  # NOQA
>>> import wbia
>>> # build test data
>>> if True:
>>>     import matplotlib as mpl
>>>     from wbia.scripts.thesis import TMP_RC
>>>     mpl.rcParams.update(TMP_RC)
>>> qreq_ = wbia.testdata_qreq_()
>>> ibs = wbia.opendb('PZ_MTEST')
>>> qaid = qreq_.qaids[0]
>>> qfx = ut.get_argval('--qfx', type_=None, default=879)
>>> fnum = None
>>> stride = 5
>>> # execute function
>>> skip = False
>>> result = show_nearest_descriptors(ibs, qaid, qfx, fnum, stride,
>>>                                   draw_chip=True,
>>>                                   draw_warped=True,
>>>                                   draw_unwarped=False,
>>>                                   draw_desc=False, qreq_=qreq_)
>>> # verify results
>>> print(result)
>>> pt.show_if_requested()
```

wbia.viz.viz_nearest_descriptors.**show_top_featmatches**(*qreq_*, *cm_list*)

Parameters

- **qreq** (*wbia.QueryRequest*) – query request object with hyper-parameters
- **cm_list** (*list*) –

SeeAlso: python -m wbia –tf TestResult.draw_feat_scoresep –show –db PZ_MTEST -t best:lnbnn_on=True,lnbnn_normalizer=normlnbnn-test -a default –sephack

python -m wbia –tf TestResult.draw_feat_scoresep –show –db PZ_Master1 -t best:lnbnn_on=True -a timectrl –sephack python -m wbia –tf TestResult.draw_feat_scoresep –show –db PZ_MTEST -t best:lnbnn_on=True -a default:size=30 –sephack python -m wbia –tf TestResult.draw_feat_scoresep –show –db PZ_MTEST -t best:K=1,Knorm=5,lnbnn_on=True -a default:size=30 –sephack python -m wbia –tf TestResult.draw_feat_scoresep –show –db PZ_MTEST -t best:K=1,Knorm=3,lnbnn_on=True -a default –sephack

CommandLine: python -m wbia.viz.viz_nearest_descriptors –exec-show_top_featmatches –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.viz.viz_nearest_descriptors import *  # NOQA
>>> import wbia
>>> cm_list, qreq_ = wbia.testdata_cmlist(defaultdb='PZ_MTEST',
>>>                                       a=['default:has_none=mother,size=30'])
>>> show_top_featmatches(qreq_, cm_list)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

1.14.12 wbia.viz.viz_other module

1.14.13 wbia.viz.viz_qres module

wbia.viz.viz_qres.**show_qres**(*ibs*, *cm*, *qreq_=None*, ***kwargs*)

Display Query Result Logic Defaults to: query chip, groundtruth matches, and top matches

Parameters

- **ibs** (*wbia.IBEISController*) – wbia controller object
- **cm** (*wbia.ChipMatch*) – object of feature correspondences and scores
- **qreq** (*wbia.QueryRequest*) – query request object with hyper-parameters(default = None)

Kwargs: annot_mode, figtitle, make_figtitle, aug, top_aids, all_kpts, show_query, in_image, sidebyside, name_scoring, max_nCols, failed_to_match, fnum in_image (bool) show result in image view if True else chip view annot_mode (int):

if annot_mode == 0, then draw lines and ellipse elif annot_mode == 1, then dont draw lines or ellipse elif annot_mode == 2, then draw only lines elif annot_mode == 3, draw heatmask only

See: viz_matches.show_name_matches, viz_helpers.get_query_text

Returns fig

Return type mpl.Figure

CommandLine: ./main.py –query 1 -y –db PZ_MTEST –noshow-qres python -m wbia.viz.viz_qres show_qres –show python -m wbia.viz.viz_qres show_qres –show –top-aids=10 –db=PZ_MTEST –sidebyside –annot_mode=0 –notitle –no-viz_name_score –qaids=5 –max_nCols=2 –adjust=.01,.01,.01

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.viz.viz_qres import *  # NOQA
>>> import wbia
>>> cm, qreq_ = wbia.testdata_cm()
>>> kwargs = dict(
>>>     top_aids=ut.get_argval('--top-aids', type_=int, default=3),
>>>     sidebyside=not ut.get_argflag('--no-sidebyside'),
>>>     annot_mode=ut.get_argval('--annot_mode', type_=int, default=1),
>>>     viz_name_score=not ut.get_argflag('--no-viz_name_score'),
>>>     simplemode=ut.get_argflag('--simplemode'),
>>>     max_nCols=ut.get_argval('--max_nCols', type_=int, default=None)
>>> )
>>> ibs = qreq_.ibs
>>> fig = show_qres(ibs, cm, show_query=False, qreq_=qreq_, **kwargs)
>>> ut.show_if_requested()
```

wbia.viz.viz_qres.**show_qres_analysis**(*ibs*, *cm*, *qreq_=None*, ***kwargs*)

Wrapper around show_qres.

KWARGS: aid_list - show matches against aid_list (default top 3)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **cm** (ChipMatch) – object of feature correspondences and scores
- **qreq** (QueryRequest) – query request object with hyper-parameters(default = None)

Kwargs: N, show_gt, show_query, aid_list, figtitle, viz_name_score, viz_name_score

CommandLine: python -m wbia.viz.viz_qres –exec-show_qres_analysis –show

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.viz.viz_qres import *  # NOQA
>>> import wbia
>>> cm, qreq_ = wbia.testdata_cm(
>>>     defaultdb='PZ_MTEST', default_qaids=[1],
>>>     default_daids=[2, 3, 4, 5, 6, 7, 8, 9])
>>> kwargs = dict(show_query=False, viz_name_score=True,
>>>               show_timedelta=True, N=3, show_gf=True)
>>> ibs = qreq_.ibs
>>> show_qres_analysis(ibs, cm, qreq_, **kwargs)
>>> ut.show_if_requested()
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.viz.viz_qres import *  # NOQA
>>> import wbia
>>> cm, qreq_ = wbia.testdata_cm(
>>>     defaultdb='PZ_MTEST', default_qaids=[1],
>>>     default_daids=[2])
>>> kwargs = dict(show_query=False, viz_name_score=True,
```

(continues on next page)

(continued from previous page)

```
>>>                show_timedelta=True, N=3, show_gf=True)
>>> ibs = qreq_.ibs
>>> show_qres_analysis(ibs, cm, qreq_, **kwargs)
>>> ut.show_if_requested()
```

wbia.viz.viz_qres.**show_qres_top**(*ibs*, *cm*, *qreq_=None*, ***kwargs*)
Wrapper around show_qres.

1.14.14 wbia.viz.viz_sver module

wbia.viz.viz_sver.**show_sver**(*ibs*, *aid1*, *aid2*, *chipmatch_FILT=None*, *aid2_svtup=None*, *config2_=None*, ***kwargs*)
Compiles IBEIS information and sends it to plottool
CommandLine: python -m wbia.viz.viz_sver –test-show_sver –show

Example

```
>>> # SLOW_DOCTEST
>>> from wbia.viz.viz_sver import *    # NOQA
>>> import wbia
>>> import utool as ut
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> aid1, aid2 = aid_list[0:2]
>>> chipmatch_FILT = None
>>> aid2_svtup = None
>>> kwargs = {}
>>> show_sver(ibs, aid1, aid2)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> exec(pt.present())
```

1.14.15 Module contents

1.15 wbia.web package

1.15.1 Submodules

1.15.2 wbia.web.apis module

Dependencies: flask, tornado

wbia.web.apis.**annot_src_api**(*rowid=None*, *fresh=False*, ***kwargs*)
Returns the image file of annot <aid>

Example

```
>>> # xdoctest: +REQUIRES(--slow)
>>> # xdoctest: +REQUIRES(--web-tests)
>>> from wbia.web.app import *  # NOQA
>>> import wbia
>>> with wbia.opendb_with_web('testdb1') as (ibs, client):
...     resp = client.get('/api/annot/src/1/')
>>> print(resp.data)
b'\xff\xd8\xff\xe0\x00\x10JFIF...
```

RESTful: Method: GET URL: /api/annot/src/<rowid>/

wbia.web.apis.**api_test_datasets_id**(*ibs*, *dataset*, **args*, ***kwargs*)

wbia.web.apis.**background_src_api**(*rowid=None*, *fresh=False*, ***kwargs*)
Returns the image file of annot <aid>

Example

```
>>> # xdoctest: +REQUIRES(--slow)
>>> # xdoctest: +REQUIRES(--web-tests)
>>> # xdoctest: +REQUIRES(module:wbia_cnn)
>>> from wbia.web.app import *  # NOQA
>>> import wbia
>>> with wbia.opendb_with_web('testdb1') as (ibs, client):
...     resp = client.get('/api/background/src/1/')
>>> print(resp.data)
b'\xff\xd8\xff\xe0\x00\x10JFIF...
```

RESTful: Method: GET URL: /api/annot/src/<rowid>/

wbia.web.apis.**heartbeat**(*ibs*, **args*, ***kwargs*)

wbia.web.apis.**hello_world**(**args*, ***kwargs*)

Example

```
>>> # xdoctest: +REQUIRES(--web-tests)
>>> from wbia.web.app import *  # NOQA
>>> import wbia
>>> import requests
>>> import wbia
>>> with wbia.opendb_with_web('testdb1') as (ibs, client):
...     resp = client.get('/api/test/helloworld/?test0=0')
...     payload = {
...         'test1' : 'test1',
...         'test2' : None,  # NOTICE test2 DOES NOT SHOW UP
...     }
...     resp = client.post('/api/test/helloworld/', data=payload)
```

wbia.web.apis.**image_conv_feature_api**(*rowid=None*, *model='resnet50'*, ***kwargs*)
RESTful: Method: GET URL: /api/image/feature/json/<uuid>/

wbia.web.apis.**image_conv_feature_api_json**(*uuid=None*, *model='resnet50'*, ***kwargs*)
RESTful: Method: GET URL: /api/image/feature/json/<uuid>/

wbia.web.apis.**image_src_api**(*rowid=None*, *thumbnail=False*, *fresh=False*, ***kwargs*)
Returns the image file of image <gid>

Example

```
>>> from wbia.web.app import *  # NOQA
>>> import wbia
>>> with wbia.opendb_with_web('testdb1') as (ibs, client):
...     resp = client.get('/api/image/src/1/')
>>> print(resp.data)
b'\xff\xd8\xff\xe0\x00\x10JFIF...
```

RESTful: Method: GET URL: /api/image/src/<rowid>/

wbia.web.apis.**image_src_api_ext**(**args*, ***kwargs*)

wbia.web.apis.**image_src_api_json**(*uuid=None*, ***kwargs*)
Returns the image file of image <gid>

Example

```
>>> # xdoctest: +REQUIRES(--web-tests)
>>> from wbia.web.app import *  # NOQA
>>> import wbia
>>> with wbia.opendb_with_web('testdb1') as (ibs, client):
...     resp = client.get('/api/image/src/json/0a9bc03d-a75e-8d14-0153-
↪e2949502aba7/')
>>> print(resp.data)
b'\xff\xd8\xff\xe0\x00\x10JFIF...
```

RESTful: Method: GET URL: /api/image/src/<gid>/

wbia.web.apis.**image_upload**(*cleanup=True*, ***kwargs*)
Returns the gid for an uploaded image.

Parameters

- **image** (*image binary*) – the POST variable containing the binary (multi-form) image data
- ****kwargs** – Arbitrary keyword arguments; the kwargs are passed down to the add_images function

Returns

gid corresponding to the image submitted. lexigraphical order.

Return type gid (rowids)

RESTful: Method: POST URL: /api/upload/image/

wbia.web.apis.**image_upload_zip**(***kwargs*)
Returns the gid_list for image files submitted in a ZIP archive. The image archive should be flat (no folders will be scanned for images) and must be smaller than 100 MB. The archive can submit multiple images, ideally in JPEG format to save space. Duplicate image uploads will result in the duplicate images receiving the same gid based on the hashed pixel values.

Parameters

- **image_zip_archive** (*binary*) – the POST variable containing the binary (multi-form) image archive data
- ****kwargs** – Arbitrary keyword arguments; the kwargs are passed down to the add_images function

Returns

the list of gids corresponding to the images submitted. The gids correspond to the image names sorted in lexigraphical order.

Return type gid_list (list if rowids)

RESTful: Method: POST URL: /api/image/zip

wbia.web.apis.**web_embed**(**args*, ***kwargs*)

1.15.3 wbia.web.apis_detect module

Dependencies: flask, tornado.

wbia.web.apis_detect.**aoi_cnn**(*ibs*, *aid_list*, *testing=False*, *model_tag='candidacy'*, ***kwargs*)

wbia.web.apis_detect.**commit_detection_results**(*ibs*, *gid_list*, *results_list*, *note=None*, *update_json_log=True*)

wbia.web.apis_detect.**commit_detection_results_filtered**(*ibs*, *gid_list*, *filter_species_list=None*, *filter_viewpoint_list=None*, *note=None*, *update_json_log=True*)

wbia.web.apis_detect.**commit_localization_results**(*ibs*, *gid_list*, *results_list*, *note=None*, *labeler_algo='pipeline'*, *labeler_model_tag=None*, *viewpoint_model_tag=None*, *use_labeler_species=False*, *orienter_algo=None*, *orienter_model_tag=None*, *assigner_algo=None*, *assigner_model_tag=None*, *update_json_log=True*, *apply_nms_post_use_labeler_species=True*, ***kwargs*)

wbia.web.apis_detect.**detect_cnn_json**(*ibs*, *gid_list*, *detect_func*, *config={}*, ***kwargs*)

Run animal detection in each image and returns json-ready formatted results, does not return annotations.

Parameters **gid_list** (*list*) – list of image ids to run detection on

Returns dict of detection results (not annotations)

Return type results_dict (list)

CommandLine: python -m wbia.web.apis_detect –test-detect_cnn_yolo_json

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.web.apis_detect import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> gid_list = ibs.get_valid_gids()[0:2]
>>> results_dict = ibs.detect_cnn_yolo_json(gid_list)
>>> print(results_dict)
```

wbia.web.apis_detect.**detect_cnn_json_wrapper**(*ibs*, *image_uuid_list*, *detect_func*, ***kwargs*)

Detect with CNN (general).

REST: Method: GET URL: /api/detect/cnn/yolo/json/

Parameters **image_uuid_list** (*list*) – list of image uuids to detect on.

wbia.web.apis_detect.**detect_cnn_lightnet**(*ibs*, *gid_list*, *model_tag=None*, *commit=True*, *testing=False*, ***kwargs*)

Run animal detection in each image. Adds annotations to the database as they are found.

Parameters **gid_list** (*list*) – list of image ids to run detection on

Returns

list of lists of annotation ids detected in each image

Return type aids_list (list)

CommandLine: python -m wbia.web.apis_detect –test-detect_cnn_lightnet –show

RESTful: Method: PUT, GET URL: /api/detect/cnn/lightnet/

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.web.apis_detect import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('PZ_MTEST')
>>> gid_list = ibs.get_valid_gids()[:5]
>>> aids_list = ibs.detect_cnn_lightnet(gid_list)
>>> if ut.show_was_requested():
>>>     import wbia.plottool as pt
>>>     from wbia.viz import viz_image
>>>     for fnum, gid in enumerate(gid_list):
>>>         viz_image.show_image(ibs, gid, fnum=fnum)
>>>     pt.show_if_requested()
>>> # Remove newly detected annotations
>>> ibs.delete_annots(ut.flatten(aids_list))
```

wbia.web.apis_detect.**detect_cnn_lightnet_image_uris_json**(*ibs*, *image_uris*, *config={}*, ***kwargs*)

wbia.web.apis_detect.**detect_cnn_lightnet_json**(*ibs*, *gid_list*, *config={}*, ***kwargs*)

wbia.web.apis_detect.**detect_cnn_lightnet_json_wrapper**(*ibs*, *image_uuid_list*, ***kwargs*)

wbia.web.apis_detect.**detect_cnn_yolo**(*ibs*, *gid_list*, *model_tag=None*, *commit=True*, *testing=False*, ***kwargs*)

Run animal detection in each image. Adds annotations to the database as they are found.

Parameters **gid_list** (*list*) – list of image ids to run detection on

Returns

list of lists of annotation ids detected in each image

Return type aids_list (list)

CommandLine: python -m wbia.web.apis_detect –test-detect_cnn_yolo –show

RESTful: Method: PUT, GET URL: /api/detect/cnn/yolo/

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.web.apis_detect import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('PZ_MTEST')
>>> gid_list = ibs.get_valid_gids()[:5]
>>> aids_list = ibs.detect_cnn_yolo(gid_list)
>>> if ut.show_was_requested():
```

(continues on next page)

(continued from previous page)

```
>>>     import wbia.plottool as pt
>>>     from wbia.viz import viz_image
>>>     for fnum, gid in enumerate(gid_list):
>>>         viz_image.show_image(ibs, gid, fnum=fnum)
>>>     pt.show_if_requested()
>>> # Remove newly detected annotations
>>> ibs.delete_annots(ut.flatten(aids_list))
```

wbia.web.apis_detect.**detect_cnn_yolo_exists**(*ibs*, *gid_list*, *testing=False*)

Check to see if a detection has been completed.

Parameters **gid_list** (*list*) – list of image ids to run detection on

Returns

list of flags for if the detection has been run on the image

Return type flag_list (list)

CommandLine: python -m wbia.web.apis_detect –test-detect_cnn_yolo_exists

RESTful: Method: GET URL: /api/detect/cnn/yolo/exists/

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.web.apis_detect import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('PZ_MTEST')
>>> gid_list = ibs.get_valid_gids()
>>> depc = ibs.depc_image
>>> aids_list = ibs.detect_cnn_yolo(gid_list[:3], testing=True)
>>> result = ibs.detect_cnn_yolo_exists(gid_list[:5])
>>> ibs.delete_annots(ut.flatten(aids_list))
>>> print(result)
[True, True, True, False, False]
```

wbia.web.apis_detect.**detect_cnn_yolo_json**(*ibs*, *gid_list*, *config={}*, ***kwargs*)

wbia.web.apis_detect.**detect_cnn_yolo_json_wrapper**(*ibs*, *image_uuid_list*, ***kwargs*)

wbia.web.apis_detect.**detect_ws_injury**(*ibs*, *gid_list*)

Classify if a whale shark is injured.

Parameters **gid_list** (*list*) – list of image ids to run classification on

Returns

predictions is list of strings representing a possible tag. confidences is a list of floats of correspoinding cofidence to the prediction

Return type result_list (dictionary)

wbia.web.apis_detect.**detection_lightnet_test**(*ibs*, *config={}*)

wbia.web.apis_detect.**detection_yolo_test**(*ibs*, *config={}*)

wbia.web.apis_detect.**get_species_with_detectors**(*ibs*)

Get valid species for detection.

RESTful: Method: GET URL: /api/detect/species/

wbia.web.apis_detect.**get_working_species**(*ibs*)

Get working species for detection.

RESTful: Method: GET URL: /api/detect/species/working/

wbia.web.apis_detect.**has_species_detector**(*ibs*, *species_text*)
TODO: extend to use non-constant species.
RESTful: Method: GET URL: /api/detect/species/enabled/

wbia.web.apis_detect.**labeler_cnn**(*ibs*, *aid_list*, *testing=False*, *algo='pipeline'*, *model_tag='candidacy'*, ***kwargs*)

wbia.web.apis_detect.**log_detections**(*ibs*, *aid_list*, *fallback=True*)

wbia.web.apis_detect.**models_cnn**(*ibs*, *config_dict*, *parse_classes_func*, *parse_line_func*, *check_hash=False*, *hidden_models=[]*, ***kwargs*)

wbia.web.apis_detect.**models_cnn_lightnet**(*ibs*, ***kwargs*)
Return the models (and their labels) for the YOLO CNN detector
RESTful: Method: PUT, GET URL: /api/labels/cnn/lightnet/

wbia.web.apis_detect.**models_cnn_yolo**(*ibs*, ***kwargs*)
Return the models (and their labels) for the YOLO CNN detector
RESTful: Method: PUT, GET URL: /api/labels/cnn/yolo/

wbia.web.apis_detect.**process_detection_html**(*ibs*, ***kwargs*)
Process the return from the detection review interface. Pass the POST result from the detection review form directly to this function unmodified.

Returns Same format as *func:start_detect_image*
Return type detection results (dict)

RESTful: Method: POST URL: /api/review/detect/cnn/yolo/

wbia.web.apis_detect.**review_detection_html**(*ibs*, *image_uuid*, *result_list*, *callback_url*, *callback_method='POST'*, *include_jquery=False*, *config=None*)
Return the detection review interface for a particular image UUID and a list of results for that image.

Parameters

- **image_uuid** (*UUID*) – the UUID of the image you want to review detections for
- **result_list** (*list of dict*) – list of detection results returned by the detector
- **callback_url** (*str*) – URL that the review form will submit to (action) when the user is complete with their review
- **callback_method** (*str*) – HTTP method the review form will submit to (method). Defaults to 'POST'

Returns json response with the detection web interface in html
Return type template (html)

RESTful: Method: GET URL: /api/review/detect/cnn/yolo/

wbia.web.apis_detect.**review_detection_test**(*image_uuid=None*, *result_list=None*, *callback_url=None*, *callback_method='POST'*, *callback_detailed=False*, ***kwargs*)

wbia.web.apis_detect.**wic_cnn**(*ibs*, *gid_list*, *testing=False*, *algo='cnn'*, *model_tag='candidacy'*, ***kwargs*)

wbia.web.apis_detect.**wic_cnn_json**(*ibs*, *gid_list*, *config={}*, ***kwargs*)

1.15.4 wbia.web.apis_engine module

wbia.web.apis_engine.**ensure_simple_server**(*port=5832*)
CommandLine: python -m wbia.web.apis_engine –exec-ensure_simple_server python -m utool.util_web –exec-start_simple_webserver

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.web.apis_engine import *  # NOQA
>>> result = ensure_simple_server()
>>> print(result)
```

wbia.web.apis_engine.**ensure_uuid_list**(*list_*)

wbia.web.apis_engine.**start_add_images**(*ibs*, *image_uri_list*, *callback_url=None*, *callback_method=None*, *callback_detailed=False*, *lane='fast'*, *jobid=None*, ***kwargs*)

REST: Method: GET/api/engine/image/json/ URL:

Parameters

- **image_uri_list** (*list*) – list of image urls to import.
- **callback_url** (*url*) – url that will be called when detection succeeds or fails

wbia.web.apis_engine.**start_detect_image_lightnet**(*ibs*, *image_uuid_list*, *callback_url=None*, *callback_method=None*, *callback_detailed=False*, *lane='fast'*, *jobid=None*, ***kwargs*)

REST: Method: GET/api/engine/detect/cnn/lightnet/ URL:

Parameters

- **image_uuid_list** (*list*) – list of image uuids or urls to detect on.
- **callback_url** (*url*) – url that will be called when detection succeeds or fails

wbia.web.apis_engine.**start_detect_image_test_lightnet**(*ibs*)

wbia.web.apis_engine.**start_detect_image_test_yolo**(*ibs*)

wbia.web.apis_engine.**start_detect_image_yolo**(*ibs*, *image_uuid_list*, *callback_url=None*, *callback_method=None*, *callback_detailed=False*, *lane='fast'*, *jobid=None*, ***kwargs*)

REST: Method: GET URL: /api/engine/detect/cnn/yolo/

Parameters

- **image_uuid_list** (*list*) – list of image uuids to detect on.
- **callback_url** (*url*) – url that will be called when detection succeeds or fails

wbia.web.apis_engine.**start_identify_annots**(*ibs*, *qannot_uuid_list*, *dannot_uuid_list=None*, *pipecfg={}*, *callback_url=None*, *callback_method=None*, *callback_detailed=False*, *lane='slow'*, *jobid=None*)

REST: Method: GET URL: /api/engine/query/annot/rowid/

Parameters

- **qannot_uuid_list** (*list*) – specifies the query annotations to identify.
- **dannot_uuid_list** (*list*) – specifies the annotations that the algorithm is allowed to use for identification. If not specified all annotations are used. (default=None)
- **pipecfg** (*dict*) – dictionary of pipeline configuration arguments (default=None)

CommandLine: # Run as main process python -m wbia.web.apis_engine –exec-start_identify_annots:0 # Run using server process python -m wbia.web.apis_engine –exec-start_identify_annots:1

Split into multiple processes python -m wbia.web.apis_engine –main –bg python -m wbia.web.apis_engine –exec-start_identify_annots:1 –fg

python -m wbia.web.apis_engine –exec-start_identify_annots:1 –domain http://52.33.105.88

python -m wbia.web.apis_engine –exec-start_identify_annots:1 –duuids=[] python -m wbia.web.apis_engine –exec-start_identify_annots:1 –domain http://52.33.105.88 –duuids=03a17411-c226-c960-d180-9fafef88c880

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.web.apis_engine import *  # NOQA
>>> from wbia.web import apis_engine
>>> import wbia
>>> ibs, qaids, daids = wbia.testdata_expanded_aids(
>>>     defaultdb='PZ_MTEST', a=['default:qsize=2,dsize=10'])
>>> qannot_uuid_list = ibs.get_annot_uuids(qaids)
>>> dannot_uuid_list = ibs.get_annot_uuids(daids)
>>> pipecfg = {}
>>> ibs.initialize_job_manager()
>>> jobid = ibs.start_identify_annots(qannot_uuid_list, dannot_uuid_list, pipecfg)
>>> result = ibs.wait_for_job_result(jobid, timeout=None, freq=2)
>>> print(result)
>>> import utool as ut
>>> #print(ut.to_json(result))
>>> ibs.close_job_manager()
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.web.apis_engine import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')  # , domain='http://52.33.105.88')
>>> aids = ibs.get_valid_aids()[0:2]
>>> qaids = aids[0:1]
>>> daids = aids
>>> query_config_dict = {
>>>     #'pipeline_root' : 'BC_DTW'
>>> }
>>> qreq_ = ibs.new_query_request(qaids, daids, cfgdict=query_config_dict)
>>> cm_list = qreq_.execute()
```

Example

```
>>> # xdoctest: +REQUIRES(--web-tests)
>>> # xdoctest: +REQUIRES(--job-engine-tests)
>>> from wbia.web.apis_engine import *  # NOQA
>>> import wbia
>>> with wbia.opendb_bg_web('testdb1', managed=True) as web_ibs:  # , domain=
↪'http://52.33.105.88')
...     aids = web_ibs.send_wbia_request('/api/annot/', 'get')[0:2]
...     uuid_list = web_ibs.send_wbia_request('/api/annot/uuid/', type_='get',␣
↪aid_list=aids)
...     quuid_list = ut.get_argval('--quuids', type_=list, default=uuid_list)
...     duuid_list = ut.get_argval('--duuids', type_=list, default=uuid_list)
```

(continues on next page)

(continued from previous page)

```
...     data = dict(
...         qannot_uuid_list=quuid_list, dannot_uuid_list=duuid_list,
...         pipecfg={},
...         callback_url='http://127.0.1.1:5832'
...     )
...     # Start callback server
...     bgserver = ensure_simple_server()
...     # --
...     jobid = web_ibs.send_wbia_request('/api/engine/query/annot/rowid/',␣
↪**data)
...     status_response = web_ibs.wait_for_results(jobid, delays=[1, 5, 30])
...     print('status_response = %s' % (status_response,))
...     result_response = web_ibs.read_engine_results(jobid)
...     print('result_response = %s' % (result_response,))
...     cm_dict = result_response['json_result'][0]
...     print('Finished test')
...     bgserver.terminate2()
Waiting for server to be up. count=0
...
status_response = {'status': 'ok', 'jobid': '...', 'jobstatus': 'completed'}
result_response = ...
Finished test
```

Ignore: qaids = daids = ibs.get_valid_aids() jobid = ibs.start_identify_annots(**payload)

wbia.web.apis_engine.**start_identify_annots_query**(*ibs*, *query_annot_uuid_list=None*, *query_annot_name_list=None*, *database_annot_uuid_list=None*, *database_annot_name_list=None*, *database_imgsetid=None*, *matching_state_list=[]*, *query_config_dict={}*, *echo_query_params=True*, *include_qaid_in_daids=True*, *callback_url=None*, *callback_method=None*, *callback_detailed=False*, *lane='slow'*, *jobid=None*)

REST: Method: GET URL: /api/engine/query/graph/

Parameters

- **query_annot_uuid_list** (*list*) – specifies the query annotations to identify.
- **query_annot_name_list** (*list*) – specifies the query annotation names
- **database_annot_uuid_list** (*list*) – specifies the annotations that the algorithm is allowed to use for identification. If not specified all annotations are used. (default=None)
- **database_annot_name_list** (*list*) – specifies the database annotation names (default=None)
- **matching_state_list** (*list of tuple*) – the list of matching state 3-tuples corresponding to the query_annot_uuid_list (default=None)
- **query_config_dict** (*dict*) – dictionary of algorithmic configuration arguments. (default=None)
- **echo_query_params** (*bool*) – flag for if to return the original query parameters with the result

CommandLine: # Normal mode python -m wbia.web.apis_engine start_identify_annots_query # Split mode

wbia –web python -m wbia.web.apis_engine start_identify_annots_query –show –domain=localhost

Example

```
>>> # DISABLE_DOCTEST
>>> # xdoctest: +REQUIRES(--job-engine-tests)
>>> from wbia.web.apis_engine import *  # NOQA
>>> import wbia
>>> #domain = 'localhost'
>>> domain = None
>>> with wbia.opendb_bg_web('testdb1', domain=domain, managed=True) as web_ibs:
↪# , domain='http://52.33.105.88')
...     aids = web_ibs.send_wbia_request('/api/annot/', 'get')[0:3]
...     uuid_list = web_ibs.send_wbia_request('/api/annot/uuid/', type_='get',␣
↪aid_list=aids)
...     quuid_list = ut.get_argval('--quuids', type_=list, default=uuid_list)[0:1]
...     duuid_list = ut.get_argval('--duuids', type_=list, default=uuid_list)
...     query_config_dict = {
...         #'pipeline_root' : 'BC_DTW'
...     }
...     data = dict(
...         query_annot_uuid_list=quuid_list, database_annot_uuid_list=duuid_list,
...         query_config_dict=query_config_dict,
...     )
...     jobid = web_ibs.send_wbia_request('/api/engine/query/graph/', **data)
...     print('jobid = %r' % (jobid,))
...     status_response = web_ibs.wait_for_results(jobid)
...     result_response = web_ibs.read_engine_results(jobid)
...     print('result_response = %s' % (ut.repr3(result_response),))
...     inference_result = result_response['json_result']
...     if isinstance(inference_result, str):
...         print(inference_result)
...     cm_dict = inference_result['cm_dict']
...     quuid = quuid_list[0]
...     cm = cm_dict[str(quuid)]
```

wbia.web.apis_engine.**start_identify_annots_query_complete**(*ibs*, *annot_uuid_list=None*, *annot_name_list=None*, *matching_state_list=[]*, *query_config_dict={}*, *k=5*, *echo_query_params=True*, *callback_url=None*, *callback_method=None*, *callback_detailed=False*, *lane='slow'*, *jobid=None*)

REST: Method: GET URL: /api/engine/query/complete/

Parameters

- **annot_uuid_list** (*list*) – specifies the query annotations to identify.
- **annot_name_list** (*list*) – specifies the query annotation names

- **matching_state_list** (*list of tuple*) – the list of matching state 3-tuples corresponding to the query_annot_uuid_list (default=None)
- **query_config_dict** (*dict*) – dictionary of algorithmic configuration arguments. (default=None)
- **echo_query_params** (*bool*) – flag for if to return the original query parameters with the result

wbia.web.apis_engine.**start_labeler_cnn**(*ibs*, *annot_uuid_list*, *callback_url=None*, *callback_method=None*, *callback_detailed=False*, *lane='fast'*, *jobid=None*, ***kwargs*)

wbia.web.apis_engine.**start_predict_ws_injury_interim_svm**(*ibs*, *annot_uuid_list*, *callback_url=None*, *callback_method=None*, *callback_detailed=False*, *lane='fast'*, *jobid=None*, ***kwargs*)

REST: Method: POST URL: /api/engine/classify/whaleshark/injury/

Parameters

- **annot_uuid_list** (*list*) – list of annot uuids to detect on.
- **callback_url** (*url*) – url that will be called when detection succeeds or fails

CommandLine: python -m wbia.web.apis_engine start_predict_ws_injury_interim_svm

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.web.apis_engine import *  # NOQA
>>> from wbia.web import apis_engine
>>> import wbia
>>> ibs, qaids, daids = wbia.testdata_expanded_aids(
>>>     defaultdb='WS_ALL', a=['default:qsize=2,dsize=10'])
>>> annot_uuid_list = ibs.get_annot_uuids(qaids)
>>> ibs.initialize_job_manager()
>>> jobid = ibs.start_predict_ws_injury_interim_svm(annot_uuid_list)
>>> result = ibs.wait_for_job_result(jobid, timeout=None, freq=2)
>>> print(result)
>>> import utool as ut
>>> #print(ut.to_json(result))
>>> ibs.close_job_manager()
```

wbia.web.apis_engine.**start_review_query_chips_best**(*ibs*, *annot_uuid*, *database_imgsetid=None*, *callback_url=None*, *callback_method=None*, *callback_detailed=False*, *lane='slow'*, *jobid=None*, ***kwargs*)

wbia.web.apis_engine.**start_web_query_all**(*ibs*)

REST: Method: GET URL: /api/engine/query/web/

wbia.web.apis_engine.**start_wic_image**(*ibs*, *image_uuid_list*, *callback_url=None*, *callback_method=None*, *callback_detailed=False*, *lane='fast'*, *jobid=None*, ***kwargs*)

REST: Method: GET URL: /api/engine/wic/cnn/

Parameters

- **image_uuid_list** (*list*) – list of image uuids to detect on.
- **callback_url** (*url*) – url that will be called when detection succeeds or fails

wbia.web.apis_engine.**start_wildbook_sync**(*ibs*, ***kwargs*)

REST: Method: GET URL: /api/engine/wildbook/sync/

wbia.web.apis_engine.**web_check_annot_uuids_with_names**(*annot_uuid_list*, *name_list*)

wbia.web.apis_engine.**web_check_uuids**(*ibs*, *image_uuid_list=[]*, *qannot_uuid_list=[]*, *dannot_uuid_list=[]*)

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **image_uuid_list** (*list*) – (default = [])
- **qannot_uuid_list** (*list*) – (default = [])
- **dannot_uuid_list** (*list*) – (default = [])

CommandLine: python -m wbia.web.apis_engine –exec-web_check_uuids –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.web.apis_engine import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> image_uuid_list = []
>>> qannot_uuid_list = ibs.get_annot_uuids([1, 1, 2, 3, 2, 4])
>>> dannot_uuid_list = ibs.get_annot_uuids([1, 2, 3])
>>> try:
>>>     web_check_uuids(ibs, image_uuid_list, qannot_uuid_list,
>>>                     dannot_uuid_list)
>>> except controller_inject.WebDuplicateUUIDException:
>>>     pass
>>> else:
>>>     raise AssertionError('Should have gotten WebDuplicateUUIDException')
>>> try:
>>>     web_check_uuids(ibs, [1, 2, 3], qannot_uuid_list,
>>>                     dannot_uuid_list)
>>> except controller_inject.WebMissingUUIDException as ex:
>>>     pass
>>> else:
>>>     raise AssertionError('Should have gotten WebMissingUUIDException')
>>> print('Successfully reported errors')
```

1.15.5 wbia.web.apis_json module

Dependencies: flask, tornado

class wbia.web.apis_json.**ParseError**(*value*)

Bases: object

wbia.web.apis_json.**add_annotmatch_json**(*ibs*, *match_annot_uuid1_list*, *match_annot_uuid2_list*, *match_evidence_decision_list=None*, *match_meta_decision_list=None*, *match_confidence_list=None*, *match_user_list=None*, *match_tag_list=None*, *match_modified_list=None*, *match_count_list=None*)

wbia.web.apis_json.**add_annots_json**(*ibs*, *image_uuid_list*, *annot_bbox_list*, *annot_theta_list*, *annot_viewpoint_list=None*, *annot_quality_list=None*, *annot_species_list=None*, *annot_multiple_list=None*, *annot_interest_list=None*, *annot_name_list=None*, ***kwargs*)

REST: Method: POST URL: /api/annot/json/
Ignore: sudo pip install boto

Parameters

- **image_uuid_list** (*list of str*) – list of image UUIDs to be used in IBEIS IA
- **annot_bbox_list** (*list of 4-tuple*) – list of bounding box coordinates encoded as a 4-tuple of the values (xtl, ytl, width, height) where xtl is the 'top left corner, x value' and ytl is the 'top left corner, y value'.
- **annot_theta_list** (*list of float*) – list of radian rotation around center. Defaults to 0.0 (no rotation).
- **annot_species_list** (*list of str*) – list of species for the annotation, if known. If the list is partially known, use None (null in JSON) for unknown entries.
- **annot_name_list** (*list of str*) – list of names for the annotation, if known. If the list is partially known, use None (null in JSON) for unknown entries.
- ****kwargs** – key-value pairs passed to the ibs.add_annots() function.

CommandLine: python -m wbia.web.app –test-add_annots_json

Example

```
>>> # DISABLE_DOCTEST
>>> import wbia
>>> from wbia.control.IBEISControl import *  # NOQA
>>> web_instance = wbia.opendb(db='testdb1')
>>> _payload = {
>>>     'image_uuid_list': [
>>>         uuid.UUID('7fea8101-7dec-44e3-bf5d-b8287fd231e2'),
>>>         uuid.UUID('c081119a-e08e-4863-a710-3210171d27d6'),
>>>     ],
>>>     'annot_uuid_list': [
>>>         uuid.UUID('fe1547c5-1425-4757-9b8f-b2b4a47f552d'),
>>>         uuid.UUID('86d3959f-7167-4822-b99f-42d453a50745'),
>>>     ],
>>>     'annot_bbox_list': [
>>>         [0, 0, 1992, 1328],
>>>         [0, 0, 1194, 401],
>>>     ],
>>> }
>>> aid_list = wbia.web.app.add_annots_json(web_instance, **_payload)
>>> print(aid_list)
>>> print(web_instance.get_annot_image_uuids(aid_list))
```

(continues on next page)

(continued from previous page)

```
>>> print(web_instance.get_annot_uuids(aid_list))
>>> print(web_instance.get_annot_bboxes(aid_list))
```

wbia.web.apis_json.**add_images_json**(*ibs*, *image_uri_list*, *image_unixtime_list=None*, *image_gps_lat_list=None*, *image_gps_lon_list=None*, ***kwargs*)

REST: Method: POST URL: /api/image/json/

Ignore: sudo pip install boto

Parameters

- **image_uri_list** (*list*) – list of string image uris, most likely HTTP(S) or S3 encoded URLs. Alternatively, this can be a list of dictionaries (JSON objects) that specify AWS S3 stored assets. An example below:

  image_uri_list = ['http://domain.com/example/asset1.png', '/home/example/Desktop/example/asset2.jpg', 's3://s3.amazon.com/example-bucket-2/asset1-in-bucket-2.tif', {

  'bucket' : 'example-bucket-1', 'key' : 'example/asset1.png', 'auth_domain' : None, # Uses 127.0.0.1 'auth_access_id' : None, # Uses system default 'auth_secret_key' : None, # Uses system default

  }, {

  'bucket' : 'example-bucket-1', 'key' : 'example/asset2.jpg', # if unspecified, auth uses 127.0.0.1 and system defaults

  }, {

  'bucket' : 'example-bucket-2', 'key' : 'example/asset1-in-bucket-2.tif', 'auth_domain' : 's3.amazon.com', 'auth_access_id' : '____________________', 'auth_secret_key' : '____',

  },

  ]

  Note that you cannot specify AWS authentication access ids or secret keys using string uri's. For specific authentication methods, please use the latter list of dictionaries.
- **image_time_posix_list** (*list of int*) – list of image's POSIX timestamps
- **image_gps_lat_list** (*list of float*) – list of image's GPS latitude values
- **image_gps_lon_list** (*list of float*) – list of image's GPS longitude values
- ****kwargs** – key-value pairs passed to the ibs.add_images() function.

CommandLine: python -m wbia.web.apis_json –test-add_images_json

,"bucket":"flukebook-prod-asset-store","key":""

Example

```
>>> # FIXME failing-test (03-Aug-2020) boto.exception.NoAuthHandlerFound: No␣
↪handler was ready to authenticate
>>> # xdoctest: +SKIP
>>> from wbia.control.IBEISControl import *  # NOQA
>>> import wbia
>>> import uuid
>>> web_instance = wbia.opendb(db='testdb1')
>>> _payload = {
>>>     'image_uri_list': [
>>>         'https://upload.wikimedia.org/wikipedia/commons/4/49/Zebra_running_
↪Ngorongoro.jpg',
>>>         {
>>>             'bucket'          : 'test-asset-store',
>>>             'key'             : 'caribwhale/20130903-JAC-0002.JPG',
>>>         },
>>>         {
>>>             'bucket'          : 'flukebook-prod-asset-store',
>>>             'key'             : '3/a/3a76b0e8-1c64-403d-ace1-679cf2f081c0/f2.
↪jpg',
>>>         },
>>>     ],
>>> }
>>> gid_list = wbia.web.apis_json.add_images_json(web_instance, **_payload)
>>> print(gid_list)
>>> print(web_instance.get_image_uuids(gid_list))
>>> print(web_instance.get_image_uris(gid_list))
>>> print(web_instance.get_image_paths(gid_list))
>>> print(web_instance.get_image_uris_original(gid_list))
```

wbia.web.apis_json.**add_imagesets_json**(*ibs*, *imageset_text_list*, *imageset_uuid_list=None*, *config_rowid_list=None*, *imageset_notes_list=None*, *imageset_occurence_flag_list=None*)

Adds a list of imagesets.

Parameters

- **imagesettext_list** (*list*) –
- **imageset_uuid_list** (*list*) –
- **config_rowid_list** (*list*) –
- **notes_list** (*list*) –

Returns added imageset uuids

Return type imageset_uuid_list (list)

RESTful: Method: POST URL: /api/imageset/json/

wbia.web.apis_json.**add_names_json**(*ibs*, *name_text_list*, *name_uuid_list=None*, *name_note_list=None*)

wbia.web.apis_json.**add_parts_json**(*ibs*, *annot_uuid_list*, *part_bbox_list*, *part_theta_list*, ***kwargs*)

REST: Method: POST URL: /api/part/json/

Ignore: sudo pip install boto

Parameters

- **annot_uuid_list** (*list of str*) – list of annot UUIDs to be used in IBEIS IA
- **part_uuid_list** (*list of str*) – list of annotations UUIDs to be used in IBEIS IA
- **part_bbox_list** (*list of 4-tuple*) – list of bounding box coordinates encoded as a 4-tuple of the values (xtl, ytl, width, height) where xtl is the 'top left corner, x value' and ytl is the 'top left corner, y value'.

- **part_theta_list** (*list of float*) – list of radian rotation around center. Defaults to 0.0 (no rotation).
- ****kwargs** – key-value pairs passed to the ibs.add_annots() function.

wbia.web.apis_json.**add_review_json**(*ibs*, *review_annot_uuid1_list*, *review_annot_uuid2_list*, *review_evidence_decision_list*, *review_meta_decision_list=None*, *review_uuid_list=None*, *review_user_list=None*, *review_user_confidence_list=None*, *review_tags_list=None*, *review_client_start_time_posix=None*, *review_client_end_time_posix=None*, *review_server_start_time_posix=None*, *review_server_end_time_posix=None*)

wbia.web.apis_json.**add_species_json**(*ibs*, *species_nice_list*, *species_text_list=None*, *species_code_list=None*, *species_uuid_list=None*, *species_note_list=None*, *skip_cleaning=False*)

wbia.web.apis_json.**annotation_src_api_json**(*ibs*, *uuid=None*)

wbia.web.apis_json.**chaos_imageset**(*ibs*)

REST: Method: POST URL: /api/image/json/

Parameters **image_uuid_list** (*list of str*) – list of image UUIDs to be delete from IBEIS

wbia.web.apis_json.**delete_annots_json**(*ibs*, *annot_uuid_list*)

REST: Method: DELETE URL: /api/annot/json/

Parameters **annot_uuid_list** (*list of str*) – list of annot UUIDs to be delete from IBEIS

wbia.web.apis_json.**delete_images_json**(*ibs*, *image_uuid_list*)

REST: Method: DELETE URL: /api/image/json/

Parameters **image_uuid_list** (*list of str*) – list of image UUIDs to be delete from IBEIS

wbia.web.apis_json.**delete_imageset_json**(*ibs*, *imageset_uuid_list*)

REST: Method: DELETE URL: /api/imageset/json/

Parameters **imageset_uuid_list** (*list of str*) – list of imageset UUIDs to be delete from IBEIS

wbia.web.apis_json.**delete_name_json**(*ibs*, *name_uuid_list*)

REST: Method: DELETE URL: /api/name/json/

Parameters **name_uuid_list** (*list of str*) – list of name UUIDs to be delete from IBEIS

wbia.web.apis_json.**delete_species_json**(*ibs*, *species_uuid_list*)

REST: Method: DELETE URL: /api/species/json/

Parameters **species_uuid_list** (*list of str*) – list of species UUIDs to be delete from IBEIS

wbia.web.apis_json.**get_annot_age_months_est_json**(*ibs*, *annot_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_annot_age_months_est_max_json**(*ibs*, *annot_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_annot_age_months_est_max_texts_json**(*ibs*, *annot_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_annot_age_months_est_min_json**(*ibs*, *annot_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_annot_age_months_est_min_texts_json**(*ibs*, *annot_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_annot_age_months_est_texts_json**(*ibs*, *annot_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_annot_aids_from_uuid_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_bboxes_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_detect_confidence_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_exemplar_flags_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_gids_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_hashid_uuid_json**(*ibs*, *annot_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_annot_image_contributor_tag_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_image_gps_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_image_names_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_image_paths_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_image_set_texts_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_image_unixtimes_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_image_uuids_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_imgset_uuids_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_imgsetids_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_interest_json**(*ibs*, *annot_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_annot_multiple_json**(*ibs*, *annot_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_annot_name_rowids_json**(*ibs*, *annot_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_annot_name_texts_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_nids_json**(*ibs*, *annot_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_annot_notes_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_num_verts_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_qualities_json**(*ibs*, *annot_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_annot_quality_texts_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_reviewed_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_rotated_verts_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_sex_json**(*ibs*, *annot_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_annot_sex_texts_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_species_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_species_rowids_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_species_texts_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_species_uuids_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_thetas_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_verts_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_viewpoint_texts_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_viewpoints_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_yaw_texts_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_annot_yaws_json**(*ibs*, *annot_uuid_list*)

wbia.web.apis_json.**get_contributor_rowids_from_uuid_json**(*ibs*, *contributor_uuid_list*)

wbia.web.apis_json.**get_image_aids_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_aids_of_species_json**(*ibs*, *image_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_image_annot_uuids_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_annot_uuids_of_species_json**(*ibs*, *image_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_image_detect_confidence_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_gids_from_uuid_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_gnames_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_gps_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_hash_json**(*ibs*, *image_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_image_heights_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_imagesettext_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_imgset_uuids_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_imgsetids_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_lat_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_location_codes_json**(*ibs*, *image_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_image_lon_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_name_uuids_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_nids_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_notes_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_num_annotations_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_orientation_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_orientation_str_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_paths_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_reviewed_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_sizes_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_species_rowids_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_species_uuids_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_timedelta_posix_json**(*ibs*, *image_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_image_unixtimes_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_uris_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_uris_original_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_image_uuids_with_annot_uuids**(*ibs*, *gid_list=None*)

wbia.web.apis_json.**get_image_widths_json**(*ibs*, *image_uuid_list*)

wbia.web.apis_json.**get_imageset_aids_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_annot_uuids_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_duration_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_end_time_posix_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_gids_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_gps_lats_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_gps_lons_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_image_uuids_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_imgsetids_from_text_json**(*ibs*, *imageset_text_list*, ***kwargs*)

wbia.web.apis_json.**get_imageset_imgsetids_from_uuid_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_name_uuids_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_nids_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_note_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_num_aids_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_num_annots_reviewed_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_num_gids_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_num_imgs_reviewed_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_num_names_with_exemplar_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_occurrence_flags_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_processed_flags_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_shipped_flags_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_smart_waypoint_ids_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_smart_xml_contents_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_smart_xml_fnames_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_start_time_posix_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_imageset_text_json**(*ibs*, *imageset_uuid_list*)

wbia.web.apis_json.**get_name_age_months_est_max_json**(*ibs*, *name_uuid_list*)

wbia.web.apis_json.**get_name_age_months_est_min_json**(*ibs*, *name_uuid_list*)

wbia.web.apis_json.**get_name_aids_json**(*ibs*, *name_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_name_alias_texts_json**(*ibs*, *name_uuid_list*)

wbia.web.apis_json.**get_name_annot_uuids_json**(*ibs*, *name_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_name_exemplar_aids_json**(*ibs*, *name_uuid_list*)

wbia.web.apis_json.**get_name_exemplar_name_uuids_json**(*ibs*, *name_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_name_gids_json**(*ibs*, *name_uuid_list*)

wbia.web.apis_json.**get_name_image_uuids_json**(*ibs*, *name_uuid_list*)

wbia.web.apis_json.**get_name_imgset_uuids_json**(*ibs*, *name_uuid_list*)

wbia.web.apis_json.**get_name_imgsetids_json**(*ibs*, *name_uuid_list*)

wbia.web.apis_json.**get_name_nids_with_gids_json**(*ibs*, *nid_list=None*)

wbia.web.apis_json.**get_name_notes_json**(*ibs*, *name_uuid_list*)

wbia.web.apis_json.**get_name_num_annotations_json**(*ibs*, *name_uuid_list*)

wbia.web.apis_json.**get_name_num_exemplar_annotations_json**(*ibs*, *name_uuid_list*)

wbia.web.apis_json.**get_name_rowids_from_text_json**(*ibs*, *name_text_list*, ***kwargs*)

wbia.web.apis_json.**get_name_rowids_from_uuid_json**(*ibs*, *name_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_name_sex_json**(*ibs*, *name_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_name_sex_text_json**(*ibs*, *name_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_name_temp_flag_json**(*ibs*, *name_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_name_texts_json**(*ibs*, *name_uuid_list*, ***kwargs*)

wbia.web.apis_json.**get_species_codes_json**(*ibs*, *species_uuid_list*)

wbia.web.apis_json.**get_species_nice_json**(*ibs*, *species_uuid_list*)

wbia.web.apis_json.**get_species_notes_json**(*ibs*, *species_uuid_list*)

wbia.web.apis_json.**get_species_rowids_from_text_json**(*ibs*, *species_text_list*, ***kwargs*)

wbia.web.apis_json.**get_species_rowids_from_uuids_json**(*ibs*, *species_uuid_list*)

wbia.web.apis_json.**get_species_texts_json**(*ibs*, *species_uuid_list*)

wbia.web.apis_json.**get_valid_annot_uuids_json**(*ibs*, ***kwargs*)

wbia.web.apis_json.**get_valid_image_uuids_json**(*ibs*, ***kwargs*)

wbia.web.apis_json.**get_valid_imageset_uuids_json**(*ibs*, ***kwargs*)

wbia.web.apis_json.**get_valid_name_uuids_json**(*ibs*, ***kwargs*)

wbia.web.apis_json.**get_valid_part_uuids_json**(*ibs*, ***kwargs*)

wbia.web.apis_json.**image_base64_api_json**(*ibs*, *uuid=None*, *thumbnail=False*, *fresh=False*, ***kwargs*)

wbia.web.apis_json.**labeler_cnn_json_wrapper**(*ibs*, *annot_uuid_list*, ***kwargs*)

wbia.web.apis_json.**set_annot_bboxes_json**(*ibs*, *annot_uuid_list*, *bbox_list*, ***kwargs*)

wbia.web.apis_json.**set_annot_interest_json**(*ibs*, *annot_uuid_list*, *flag_list*, ***kwargs*)

wbia.web.apis_json.**set_annot_multiple_json**(*ibs*, *annot_uuid_list*, *flag_list*, ***kwargs*)

wbia.web.apis_json.**set_annot_name_texts_json**(*ibs*, *annot_uuid_list*, *name_text_list*, ***kwargs*)

wbia.web.apis_json.**set_annot_note_json**(*ibs*, *annot_uuid_list*, *annot_note_list*, ***kwargs*)

wbia.web.apis_json.**set_annot_quality_texts_json**(*ibs*, *annot_uuid_list*, *quality_text_list*, ***kwargs*)

wbia.web.apis_json.**set_annot_species_json**(*ibs*, *annot_uuid_list*, *species_text_list*, ***kwargs*)

wbia.web.apis_json.**set_annot_tag_text_json**(*ibs*, *annot_uuid_list*, *annot_tags_list*, ***kwargs*)

wbia.web.apis_json.**set_annot_thetas_json**(*ibs*, *annot_uuid_list*, *theta_list*, ***kwargs*)

wbia.web.apis_json.**set_annot_viewpoints_json**(*ibs*, *annot_uuid_list*, *viewpoint_list*, ***kwargs*)

wbia.web.apis_json.**set_annotmatch_confidence_json**(*ibs*, *match_annot_uuid1_list*, *match_annot_uuid2_list*, *match_confidence_list*, ***kwargs*)

wbia.web.apis_json.**set_annotmatch_count_json**(*ibs*, *match_annot_uuid1_list*, *match_annot_uuid2_list*, *match_count_list*, ***kwargs*)

wbia.web.apis_json.**set_annotmatch_evidence_decision_json**(*ibs*, *match_annot_uuid1_list*, *match_annot_uuid2_list*, *match_decision_list*, ***kwargs*)

wbia.web.apis_json.**set_annotmatch_meat_decision_json**(*ibs*, *match_annot_uuid1_list*, *match_annot_uuid2_list*, *match_decision_list*, ***kwargs*)

wbia.web.apis_json.**set_annotmatch_posixtime_modified_json**(*ibs*, *match_annot_uuid1_list*, *match_annot_uuid2_list*, *match_modified_list*, ***kwargs*)

wbia.web.apis_json.**set_annotmatch_reviewer_json**(*ibs*, *match_annot_uuid1_list*, *match_annot_uuid2_list*, *match_user_list*, ***kwargs*)

wbia.web.apis_json.**set_annotmatch_tag_text_json**(*ibs*, *match_annot_uuid1_list*, *match_annot_uuid2_list*, *match_tags_list*, ***kwargs*)

wbia.web.apis_json.**set_exemplars_from_quality_and_viewpoint_json**(*ibs*, *annot_uuid_list*, *annot_name_list*, ***kwargs*)

wbia.web.apis_json.**set_image_imagesettext_json**(*ibs*, *image_uuid_list*, *imageset_text_list*)

wbia.web.apis_json.**set_image_imgset_uuids_json**(*ibs*, *image_uuid_list*, *imageset_uuid_list*)

wbia.web.apis_json.**set_image_imgsetids_json**(*ibs*, *image_uuid_list*, *imageset_rowid_list*)

wbia.web.apis_json.**set_name_notes_json**(*ibs*, *name_uuid_list*, *name_note_list*, ***kwargs*)

wbia.web.apis_json.**set_name_texts_json**(*ibs*, *name_uuid_list*, *name_text_list*, ***kwargs*)

wbia.web.apis_json.**set_part_bboxes_json**(*ibs*, *part_uuid_list*, *bbox_list*, ***kwargs*)

wbia.web.apis_json.**set_part_quality_texts_json**(*ibs*, *part_uuid_list*, *quality_text_list*, ***kwargs*)

wbia.web.apis_json.**set_part_thetas_json**(*ibs*, *part_uuid_list*, *theta_list*, ***kwargs*)

wbia.web.apis_json.**set_part_types_json**(*ibs*, *part_uuid_list*, *type_text_list*, ***kwargs*)

wbia.web.apis_json.**set_part_viewpoints_json**(*ibs*, *part_uuid_list*, *viewpoint_list*, ***kwargs*)

1.15.6 wbia.web.apis_query module

Dependencies: flask, tornado

SeeAlso: routes.review_identification

wbia.web.apis_query.**add_annots_query_chips_graph_v2**(*ibs*, *graph_uuid*, *annot_uuid_list*)

wbia.web.apis_query.**delete_query_chips_graph_v2**(*ibs*, *graph_uuid*)

wbia.web.apis_query.**ensure_review_image**(*ibs*, *aid*, *cm*, *qreq_*, *view_orientation='vertical'*, *draw_matches=True*, *draw_heatmask=False*, *verbose=False*)

” Create the review image for a pair of annotations
CommandLine: python -m wbia.web.apis_query ensure_review_image –show

Example

```
>>> # SCRIPT
>>> from wbia.web.apis_query import *  # NOQA
>>> import wbia
>>> cm, qreq_ = wbia.testdata_cm('PZ_MTEST', a='default:dindex=0:10,qindex=0:1')
>>> ibs = qreq_.ibs
>>> aid = cm.get_top_aids()[0]
>>> tt = ut.tic('make image')
>>> image, _ = ensure_review_image(ibs, aid, cm, qreq_)
>>> ut.toc(tt)
>>> ut.quit_if_noshow()
>>> print('image.shape = %r' % (image.shape,))
>>> print('image.dtype = %r' % (image.dtype,))
>>> ut.print_object_size(image)
>>> import wbia.plottool as pt
>>> pt.imshow(image)
>>> ut.show_if_requested()
```

wbia.web.apis_query.**ensure_review_image_v2**(*ibs*, *match*, *draw_matches=False*, *draw_heatmask=False*, *view_orientation='vertical'*, *overlay=True*)

wbia.web.apis_query.**get_graph_client_query_chips_graph_v2**(*ibs*, *graph_uuid*)

wbia.web.apis_query.**get_recognition_query_aids**(*ibs*, *is_known*, *species=None*)
DEPCIRATE

RESTful: Method: GET URL: /api/query/annot/rowid/

wbia.web.apis_query.**log_render_status**(*ibs*, **args*)

wbia.web.apis_query.**process_graph_match_html**(*ibs*, ***kwargs*)

RESTful: Method: POST URL: /api/review/query/graph/

wbia.web.apis_query.**process_graph_match_html_v2**(*ibs*, *graph_uuid*, ***kwargs*)

wbia.web.apis_query.**query_chips**(*ibs*, *qaid_list=None*, *daid_list=None*, *cfgdict=None*, *use_cache=None*, *use_bigcache=None*, *qreq_=None*, *return_request=False*, *verbose=False*, *save_qcache=None*, *prog_hook=None*, *return_cm_dict=False*, *return_cm_simple_dict=False*)

Submits a query request to the hotspotter recognition pipeline. Returns a list of QueryResult objects.

Parameters

- **qaid_list** (*list*) – a list of annotation ids to be submitted as queries
- **daid_list** (*list*) – a list of annotation ids used as the database that will be searched
- **cfgdict** (*dict*) – dictionary of configuration options used to create a new QueryRequest if not already specified
- **use_cache** (*bool*) – turns on/off chip match cache (default: True)
- **use_bigcache** (*bool*) – turns one/off chunked chip match cache (default: True)
- **qreq_** (*QueryRequest*) – optional, a QueryRequest object that overrides all previous settings
- **return_request** (*bool*) – returns the request which will be created if one is not already specified
- **verbose** (*bool*) – default=False, turns on verbose printing

Returns

a list of ChipMatch objects containing the matching annotations, scores, and feature matches

Return type list

Returns(2):

tuple: (cm_list, qreq_) - a list of query results and optionally the QueryRequest object used

RESTful: Method: PUT URL: /api/query/chip/

CommandLine: python -m wbia.web.apis_query –test-query_chips

Test speed of single query python -m wbia –tf IBEISController.query_chips –db PZ_Master1 -a default:qindex=0:1,dindex=0:500 –nocache-hs

python -m wbia –tf IBEISController.query_chips –db PZ_Master1 -a default:qindex=0:1,dindex=0:3000 –nocache-hs

python -m wbia.web.apis_query –test-query_chips:1 –show python -m wbia.web.apis_query –test-query_chips:2 –show

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +SKIP
>>> from wbia.control.IBEISControl import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_()
>>> ibs = qreq_.ibs
>>> cm_list = qreq_.execute()
>>> cm = cm_list[0]
```

(continues on next page)

(continued from previous page)

```
>>> ut.quit_if_noshow()
>>> cm.ishow_analysis(qreq_)
>>> ut.show_if_requested()
```

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +SKIP
>>> import wbia
>>> from wbia.control.IBEISControl import *  # NOQA
>>> qaid_list = [1]
>>> daid_list = [1, 2, 3, 4, 5]
>>> ibs = wbia.opendb_test(db='testdb1')
>>> qreq_ = ibs.new_query_request(qaid_list, daid_list)
>>> cm = ibs.query_chips(qaid_list, daid_list, use_cache=False, qreq_=qreq_)[0]
>>> ut.quit_if_noshow()
>>> cm.ishow_analysis(qreq_)
>>> ut.show_if_requested()
```

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +SKIP
>>> import wbia
>>> from wbia.control.IBEISControl import *  # NOQA
>>> qaid_list = [1]
>>> daid_list = [1, 2, 3, 4, 5]
>>> ibs = wbia.opendb_test(db='testdb1')
>>> cfgdict = {'pipeline_root':'BC_DTW'}
>>> qreq_ = ibs.new_query_request(qaid_list, daid_list, cfgdict=cfgdict,␣
↪verbose=True)
>>> cm = ibs.query_chips(qreq_=qreq_)[0]
>>> ut.quit_if_noshow()
>>> cm.ishow_analysis(qreq_)
>>> ut.show_if_requested()
```

wbia.web.apis_query.**query_chips_dict**(*ibs*, **args*, ***kwargs*)

Runs query_chips, but returns a json compatible dictionary

RESTful: Method: GET URL: /api/query/chip/dict/

wbia.web.apis_query.**query_chips_graph**(*ibs*, *qaid_list*, *daid_list*, *user_feedback=None*, *query_config_dict={}*, *echo_query_params=True*, *cache_images=True*, *n=20*, *view_orientation='horizontal'*, *return_summary=True*, ***kwargs*)

wbia.web.apis_query.**query_chips_graph_complete**(*ibs*, *aid_list*, *query_config_dict={}*, *k=5*, ***kwargs*)

wbia.web.apis_query.**query_chips_graph_match_thumb**(*extern_reference*, *query_annot_uuid*, *database_annot_uuid*, *version*)

wbia.web.apis_query.**query_chips_graph_v2**(*ibs*, *annot_uuid_list=None*, *query_config_dict={}*, *review_callback_url=None*, *review_callback_method='POST'*, *finished_callback_url=None*, *finished_callback_method='POST'*, *creation_imageset_rowid_list=None*, *backend='graph_algorithm'*, ***kwargs*)

CommandLine: python -m wbia.web.apis_query –test-query_chips_graph_v2:0

python -m wbia reset_mtest_graph

python -m wbia –db PZ_MTEST –web –browser –url=/review/identification/hardcase/ python -m wbia –db PZ_MTEST –web –browser –url=/review/identification/graph/

Example

```
>>> # xdoctest: +REQUIRES(--web-tests)
>>> from wbia.web.apis_query import *
>>> import wbia
>>> # Open local instance
>>> ibs = wbia.opendb('PZ_MTEST')
>>> uuid_list = ibs.annots().uuids[0:10]
>>> data = dict(annot_uuid_list=uuid_list)
>>> # Start up the web instance
>>> with wbia.opendb_with_web(db='PZ_MTEST') as (ibs, client):
...     resp = client.post('/api/query/graph/v2/', data=data)
>>> resp.json
{'status': {'success': False, 'code': 608, 'message': 'Invalid image and/or␣
↪annotation UUIDs (0, 1)', 'cache': -1}, 'response': {'invalid_image_uuid_list':␣
↪[], 'invalid_annot_uuid_list': [[0, '...']]}}
```

Example

```
>>> # DEBUG_SCRIPT
>>> from wbia.web.apis_query import *
>>> # Hack a flask context
>>> current_app = ut.DynStruct()
>>> current_app.GRAPH_CLIENT_DICT = {}
>>> old = query_chips_graph_v2.__globals__.get('current_app', None)
>>> query_chips_graph_v2.__globals__['current_app'] = current_app
>>> import wbia
>>> ibs = wbia.opendb('PZ_MTEST')
>>> #ut.exec_funckw(query_chips_graph_v2, globals())
>>> # Run function in main process
>>> query_chips_graph_v2(ibs)
>>> # Reset context
>>> query_chips_graph_v2.__globals__['current_app'] = old
```

wbia.web.apis_query.**query_chips_simple_dict**(*ibs*, **args*, ***kwargs*)

Runs query_chips, but returns a json compatible dictionary

Parameters **as** **query_chips** (*same*) –

RESTful: Method: GET URL: /api/query/chip/dict/simple/

SeeAlso: query_chips

CommandLine: python -m wbia.web.apis_query –test-query_chips_simple_dict:0 python -m wbia.web.apis_query –test-query_chips_simple_dict:1

python -m wbia.web.apis_query –test-query_chips_simple_dict:0 –humpbacks

Example

```
>>> # xdoctest: +REQUIRES(--web-tests)
>>> from wbia.control.IBEISControl import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> #qaid = ibs.get_valid_aids()[0:3]
>>> qaids = ibs.get_valid_aids()
>>> daids = ibs.get_valid_aids()
>>> dict_list = ibs.query_chips_simple_dict(qaids, daids)
>>> qgids = ibs.get_annot_image_rowids(qaids)
>>> qnids = ibs.get_annot_name_rowids(qaids)
>>> for dict_, qgid, qnid in list(zip(dict_list, qgids, qnids)):
>>>     dict_['qgid'] = qgid
>>>     dict_['qnid'] = qnid
>>>     dict_['dgid_list'] = ibs.get_annot_image_rowids(dict_['daid_list'])
>>>     dict_['dnid_list'] = ibs.get_annot_name_rowids(dict_['daid_list'])
>>>     dict_['dgname_list'] = ibs.get_image_gnames(dict_['dgid_list'])
>>>     dict_['qgname'] = ibs.get_image_gnames(dict_['qgid'])
>>> result  = ut.repr2(dict_list, nl=2, precision=2, hack_liststr=True)
>>> result = result.replace('u\'', '"').replace('\'', '"')
>>> print(result)
```

Example

```
>>> # xdoctest: +SKIP
>>> # FIXME failing-test (04-Aug-2020) This test hangs when running together with␣
↪the test above
>>> from wbia.control.IBEISControl import *  # NOQA
>>> import wbia
>>> # Start up the web instance
>>> with wbia.opendb_bg_web(db='testdb1', managed=True) as web_ibs:
...     cmdict_list = web_ibs.send_wbia_request('/api/query/chip/dict/simple/',␣
↪type_='get', qaid_list=[1], daid_list=[1, 2, 3])
>>> print(cmdict_list)
>>> assert 'score_list' in cmdict_list[0]
```

wbia.web.apis_query.**query_chips_test**(*ibs*, *aid=None*, *limited=False*, *census_annotations=True*, ***kwargs*)

CommandLine: python -m wbia.web.apis_query query_chips_test

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +SKIP
>>> from wbia.control.IBEISControl import *  # NOQA
>>> import wbia
>>> qreq_ = wbia.testdata_qreq_(defaultdb='testdb1')
```

(continues on next page)

(continued from previous page)

```
>>> ibs = qreq_.ibs
>>> result_dict = ibs.query_chips_test()
>>> print(result_dict)
```

wbia.web.apis_query.**query_graph_v2_callback**(*graph_client*, *callback_type*)

wbia.web.apis_query.**query_graph_v2_latest_logs**(*future*)

wbia.web.apis_query.**query_graph_v2_on_request_review**(*future*)

wbia.web.apis_query.**remove_annots_query_chips_graph_v2**(*ibs*, *graph_uuid*, *annot_uuid_list*)

wbia.web.apis_query.**review_graph_match_config_v2**(*ibs*, *graph_uuid*, *aid1=None*, *aid2=None*, *view_orientation='vertical'*, *view_version=1*)

wbia.web.apis_query.**review_graph_match_html**(*ibs*, *review_pair*, *cm_dict*, *query_config_dict*, *_internal_state*, *callback_url*, *callback_method='POST'*, *view_orientation='vertical'*, *include_jquery=False*)

Parameters

- **ibs** (*wbia.IBEISController*) – image analysis api
- **review_pair** (*dict*) – pair of annot uuids
- **cm_dict** (*dict*) –
- **query_config_dict** (*dict*) –
- **_internal_state** –
- **callback_url** (*str*) –
- **callback_method** (*unicode*) – (default = u'POST')
- **view_orientation** (*unicode*) – (default = u'vertical')
- **include_jquery** (*bool*) – (default = False)

CommandLine: python -m wbia.web.apis_query review_graph_match_html –show

wbia –web python -m wbia.web.apis_query review_graph_match_html –show –domain=localhost

Example

```
>>> # xdoctest: +REQUIRES(--web-tests)
>>> # xdoctest: +REQUIRES(--job-engine-tests)
>>> # DISABLE_DOCTEST
>>> # Disabled because this test uses opendb_bg_web, which hangs the test runner␣
↪and leaves zombie processes
>>> from wbia.web.apis_query import *  # NOQA
>>> import wbia
>>> web_ibs = wbia.opendb_bg_web('testdb1')  # , domain='http://52.33.105.88')
>>> aids = web_ibs.send_wbia_request('/api/annot/', 'get')[0:2]
>>> uuid_list = web_ibs.send_wbia_request('/api/annot/uuid/', type_='get', aid_
↪list=aids)
>>> quuid_list = uuid_list[0:1]
>>> duuid_list = uuid_list
>>> query_config_dict = {
>>>     # 'pipeline_root' : 'BC_DTW'
>>> }
>>> data = dict(
```

(continues on next page)

(continued from previous page)

```
>>>     query_annot_uuid_list=quuid_list, database_annot_uuid_list=duuid_list,
>>>     query_config_dict=query_config_dict,
>>> )
>>> jobid = web_ibs.send_wbia_request('/api/engine/query/graph/', **data)
>>> print('jobid = %r' % (jobid,))
>>> status_response = web_ibs.wait_for_results(jobid)
>>> result_response = web_ibs.read_engine_results(jobid)
>>> inference_result = result_response['json_result']
>>> print('inference_result = %r' % (inference_result,))
>>> auuid2_cm = inference_result['cm_dict']
>>> quuid = quuid_list[0]
>>> class_dict = auuid2_cm[str(quuid)]
>>> # Get information in frontend
>>> #ibs = wbia.opendb('testdb1')
>>> #cm = match_obj = wbia.ChipMatch.from_dict(class_dict, ibs=ibs)
>>> #match_obj.print_rawinfostr()
>>> # Make the dictionary a bit more managable
>>> #match_obj.compress_top_feature_matches(num=2)
>>> #class_dict = match_obj.to_dict(ibs=ibs)
>>> cm_dict = class_dict
>>> # Package for review
>>> review_pair = {'annot_uuid_1': quuid, 'annot_uuid_2': duuid_list[1]}
>>> callback_method = u'POST'
>>> view_orientation = u'vertical'
>>> include_jquery = False
>>> kw = dict(
>>>     review_pair=review_pair,
>>>     cm_dict=cm_dict,
>>>     query_config_dict=query_config_dict,
>>>     _internal_state=None,
>>>     callback_url = None,
>>> )
>>> html_str = web_ibs.send_wbia_request('/api/review/query/graph/', type_='get',␣
↪**kw)
>>> web_ibs.terminate2()
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.render_html(html_str)
>>> ut.show_if_requested()
```

Example

```
>>> # DISABLE_DOCTEST
>>> # xdoctest: +REQUIRES(--job-engine-tests)
>>> # This starts off using web to get information, but finishes the rest in␣
↪python
>>> from wbia.web.apis_query import *  # NOQA
>>> import wbia
>>> ut.exec_funckw(review_graph_match_html, globals())
>>> web_ibs = wbia.opendb_bg_web('testdb1')  # , domain='http://52.33.105.88')
>>> aids = web_ibs.send_wbia_request('/api/annot/', 'get')[0:2]
>>> uuid_list = web_ibs.send_wbia_request('/api/annot/uuid/', type_='get', aid_
↪list=aids)
>>> quuid_list = uuid_list[0:1]
>>> duuid_list = uuid_list
```

(continues on next page)

(continued from previous page)

```
>>> query_config_dict = {
>>>     # 'pipeline_root' : 'BC_DTW'
>>> }
>>> data = dict(
>>>     query_annot_uuid_list=quuid_list, database_annot_uuid_list=duuid_list,
>>>     query_config_dict=query_config_dict,
>>> )
>>> jobid = web_ibs.send_wbia_request('/api/engine/query/graph/', **data)
>>> status_response = web_ibs.wait_for_results(jobid)
>>> result_response = web_ibs.read_engine_results(jobid)
>>> web_ibs.terminate2()
>>> # NOW WORK IN THE FRONTEND
>>> inference_result = result_response['json_result']
>>> auuid2_cm = inference_result['cm_dict']
>>> quuid = quuid_list[0]
>>> class_dict = auuid2_cm[str(quuid)]
>>> # Get information in frontend
>>> ibs = wbia.opendb('testdb1')
>>> cm = wbia.ChipMatch.from_dict(class_dict, ibs=ibs)
>>> cm.print_rawinfostr()
>>> # Make the dictionary a bit more managable
>>> cm.compress_top_feature_matches(num=1)
>>> cm.print_rawinfostr()
>>> class_dict = cm.to_dict(ibs=ibs)
>>> cm_dict = class_dict
>>> # Package for review ( CANT CALL DIRECTLY BECAUSE OF OUT OF CONTEXT )
>>> review_pair = {'annot_uuid_1': quuid, 'annot_uuid_2': duuid_list[1]}
>>> x = review_graph_match_html(ibs, review_pair, cm_dict,
>>>                             query_config_dict, _internal_state=None,
>>>                             callback_url=None)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.render_html(html_str)
>>> ut.show_if_requested()
```

wbia.web.apis_query.**review_graph_match_html_alias**(**args*, ***kwargs*)

wbia.web.apis_query.**review_graph_match_html_v2**(*ibs*, *graph_uuid*, *callback_url=None*, *callback_method='POST'*, *view_orientation='vertical'*, *view_version=1*, *include_jquery=False*)

wbia.web.apis_query.**review_query_chips_best**(*ibs*, *aid*, *database_imgsetid=None*, ***kwargs*)

wbia.web.apis_query.**review_query_chips_test**(***kwargs*)

wbia.web.apis_query.**sync_query_chips_graph_v2**(*ibs*, *graph_uuid*)

wbia.web.apis_query.**view_graphs_status**(*ibs*)

1.15.7 wbia.web.apis_sync module

Dependencies: flask, tornado

SeeAlso: routes.review_identification

wbia.web.apis_sync.**detect_remote_sync_images**(*ibs*, *gid_list=None*, *only_sync_missing_images=True*)

wbia.web.apis_sync.**sync_get_training_data**(*ibs*, *species_name*, *force_update=False*, ***kwargs*)

wbia.web.apis_sync.**sync_get_training_data_uuid_list**(*ibs*, *auuid_list*, *force_update=False*, ***kwargs*)

1.15.8 wbia.web.app module

Dependencies: flask, tornado

class wbia.web.app.**TimedWSGIContainer**(*wsgi_application: WSGIAppType*)

Bases: tornado.wsgi.WSGIContainer

wbia.web.app.**start_from_wbia**(*ibs*, *port=None*, *browser=None*, *precache=None*, *url_suffix=None*, *start_job_queue=None*, *start_web_loop=True*)

Parse command line options and start the server.

CommandLine: python -m wbia –db PZ_MTEST –web python -m wbia –db PZ_MTEST –web –browser

wbia.web.app.**start_tornado**(*ibs*, *port=None*, *browser=None*, *url_suffix=None*, *start_web_loop=True*, *fallback=True*)

Initialize the web server

wbia.web.app.**start_web_annot_groupreview**(*ibs*, *aid_list*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*list*) – list of annotation rowids

CommandLine: python -m wbia.tag_funcs –exec-start_web_annot_groupreview –db PZ_Master1 python -m wbia.tag_funcs –exec-start_web_annot_groupreview –db GZ_Master1 python -m wbia.tag_funcs –exec-start_web_annot_groupreview –db GIRM_Master1

Example

```
>>> # SCRIPT
>>> from wbia.tag_funcs import *  # NOQA
>>> import wbia
>>> #ibs = wbia.opendb(defaultdb='PZ_Master1')
>>> ibs = wbia.opendb(defaultdb='GZ_Master1')
>>> #aid_list = ibs.get_valid_aids()
>>> # -----
>>> any_tags = ut.get_argval('--tags', type_=list, default=['Viewpoint'])
>>> min_num = ut.get_argval('--min_num', type_=int, default=1)
>>> prop = any_tags[0]
>>> filtered_annotmatch_rowids = filter_annotmatch_by_tags(ibs, None, any_
↪tags=any_tags, min_num=min_num)
>>> aid1_list = (ibs.get_annotmatch_aid1(filtered_annotmatch_rowids))
>>> aid2_list = (ibs.get_annotmatch_aid2(filtered_annotmatch_rowids))
>>> aid_list = list(set(ut.flatten([aid2_list, aid1_list])))
>>> result = start_web_annot_groupreview(ibs, aid_list)
>>> print(result)
```

1.15.9 wbia.web.appfuncs module

class wbia.web.appfuncs.**NavbarClass**
Bases: object

wbia.web.appfuncs.**check_valid_function_name**(*string*)

wbia.web.appfuncs.**convert_nmea_to_json**(*nmea_str*, *filename*, *GMT_OFFSET=0*)

wbia.web.appfuncs.**convert_tuple_to_viewpoint**(*viewpoint_tuple*)

wbia.web.appfuncs.**convert_viewpoint_to_tuple**(*viewpoint_text*)

wbia.web.appfuncs.**decode_refer_url**(*encoded*)

wbia.web.appfuncs.**default_species**(*ibs*)

wbia.web.appfuncs.**embed_image_html**(*imgBGR*, *target_width=1200.0*, *target_height=800.0*)
Creates an image embedded in HTML base64 format.

wbia.web.appfuncs.**encode_refer_url**(*decoded*)

wbia.web.appfuncs.**get_review_annot_args**(*is_reviewed_func*, *speed_hack=False*)
Helper to return aids in an imageset or a group review

wbia.web.appfuncs.**get_review_image_args**(*is_reviewed_func*)
Helper to return gids in an imageset or a group review

wbia.web.appfuncs.**imageset_annot_canonical**(*ibs*, *aid_list*, *canonical_part_type='__CANONICAL__'*)

wbia.web.appfuncs.**imageset_annot_demographics_processed**(*ibs*, *aid_list*)

wbia.web.appfuncs.**imageset_annot_processed**(*ibs*, *aid_list*)

wbia.web.appfuncs.**imageset_annot_quality_processed**(*ibs*, *aid_list*)

wbia.web.appfuncs.**imageset_annot_viewpoint_processed**(*ibs*, *aid_list*)

wbia.web.appfuncs.**imageset_image_cameratrap_processed**(*ibs*, *gid_list*)

wbia.web.appfuncs.**imageset_image_processed**(*ibs*, *gid_list*, *is_staged=False*, *reviews_required=3*)

wbia.web.appfuncs.**imageset_image_staged_progress**(*ibs*, *gid_list*, *reviews_required=3*)

wbia.web.appfuncs.**imageset_part_contour_processed**(*ibs*, *part_rowid_list*, *reviewed_flag_progress=True*)

wbia.web.appfuncs.**imageset_part_type_processed**(*ibs*, *part_rowid_list*, *reviewed_flag_progress=True*)

wbia.web.appfuncs.**movegroup_aid**(*ibs*, *aid*, *src_ag*, *dst_ag*)

wbia.web.appfuncs.**resize_via_web_parameters**(*image*)

wbia.web.appfuncs.**send_csv_file**(*string*, *filename*)

wbia.web.appfuncs.**template**(*template_directory=None*, *template_filename=None*, ***kwargs*)

1.15.10 wbia.web.graph_server module

class wbia.web.graph_server.**Actor**
Bases: object

classmethod executor()
Creates an asychronous instance of this Actor and returns the executor to manage it.

handle(*message*)
This method recieves, handles, and responds to the messages sent from the executor. This function can return arbitrary values. These values can be accessed from the main thread using the Future object returned when the message was posted to this actor by the executor.

wbia.web.graph_server.**GRAPH_ACTOR_CLASS**
alias of *wbia.web.graph_server.ThreadActor*

class wbia.web.graph_server.**GraphActor**(**args*, ***kwargs*)
Bases: *wbia.web.graph_server.ThreadActor*

add_aids(*aids*, ***kwargs*)

feedback(***feedback*)

handle(*message*)
This method recieves, handles, and responds to the messages sent from the executor. This function can return arbitrary values. These values can be accessed from the main thread using the Future object returned when the message was posted to this actor by the executor.

logs()

metadata()

remove_aids(*aids*, ***kwargs*)

resume()

start(*dbdir*, *aids='all'*, *config={}*, ***kwargs*)

status()

class wbia.web.graph_server.**GraphAlgorithmActor**(**args*, ***kwargs*)
Bases: *wbia.web.graph_server.GraphActor*
CommandLine: python -m wbia.web.graph_server GraphAlgorithmActor
Doctest:

```
>>> from wbia.web.graph_server import *
>>> actor = GraphAlgorithmActor()
>>> payload = testdata_start_payload()
>>> # Start the process
>>> start_resp = actor.handle(payload)
>>> print('start_resp = {!r}'.format(start_resp))
>>> # Respond with a user decision
>>> user_request = actor.handle({'action': 'resume'})
>>> # Wait for a response and  the GraphAlgorithmActor in another proc
>>> edge, priority, edge_data = user_request[0]
>>> user_resp_payload = _testdata_feedback_payload(edge, 'match')
>>> content = actor.handle(user_resp_payload)
>>> actor.infr.dump_logs()
```

Doctest:

```
>>> # xdoctest: +REQUIRES(module:wbia_cnn, --slow)
>>> from wbia.web.graph_server import *
>>> import wbia
>>> actor = GraphAlgorithmActor()
>>> config = {
>>>     'manual.n_peek'    : 1,
```

(continues on next page)

(continued from previous page)

```
>>>     'manual.autosave' : False,
>>>     'ranking.enabled' : False,
>>>     'autoreview.enabled' : False,
>>>     'redun.enabled'   : False,
>>>     'redun.enabled'   : False,
>>>     'queue.conf.thresh' : 'absolutely_sure',
>>>     'algo.hardcase' : True,
>>> }
>>> # Start the process
>>> dbdir = wbia.sysres.db_to_dbdir('PZ_MTEST')
>>> payload = {'action': 'start', 'dbdir': dbdir, 'aids': 'all',
>>>            'config': config, 'init': 'annotmatch'}
>>> start_resp = actor.handle(payload)
>>> print('start_resp = {!r}'.format(start_resp))
>>> # Respond with a user decision
>>> user_request = actor.handle({'action': 'resume'})
>>> print('user_request = {!r}'.format(user_request))
>>> # Wait for a response and  the GraphAlgorithmActor in another proc
>>> edge, priority, edge_data = user_request[0]
>>> user_resp_payload = _testdata_feedback_payload(edge, 'match')
>>> content = actor.handle(user_resp_payload)
>>> actor.infr.dump_logs()
>>> actor.infr.status()
```

add_aids(*aids*, ***kwargs*)

feedback(***feedback*)

logs()

metadata()

remove_aids(*aids*, ***kwargs*)

resume()

start(*dbdir*, *aids='all'*, *config={}*, *graph_uuid=None*, ***kwargs*)

status()

class wbia.web.graph_server.**GraphAlgorithmClient**(*aids*, *actor_config={}*, *image-sets=None*, *graph_uuid=None*, *callbacks={}*, *autoinit=False*)

Bases: *wbia.web.graph_server.GraphClient*

CommandLine: python -m wbia.web.graph_server GraphAlgorithmClient

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.web.graph_server import *
>>> import wbia
>>> client = GraphAlgorithmClient(aids='all', autoinit=True)
>>> # Start the GraphAlgorithmActor in another proc
>>> payload = testdata_start_payload()
>>> client.post(payload).result()
>>> future = client.post({'action': 'resume'})
>>> future.add_done_callback(_test_foo)
>>> user_request = future.result()
```

(continues on next page)

(continued from previous page)

```
>>> # Wait for a response and  the GraphAlgorithmActor in another proc
>>> edge, priority, edge_data = user_request[0]
>>> user_resp_payload = _testdata_feedback_payload(edge, 'match')
>>> future = client.post(user_resp_payload)
>>> future.result()
>>> # Debug by getting the actor over a mp.Pipe
>>> future = client.post({'action': 'debug'})
>>> actor = future.result()
>>> actor.infr.dump_logs()
>>> #print(client.post({'action': 'logs'}).result())
```

Ignore: # >>> from wbia.web.graph_server import * # >>> import wbia # >>> client = GraphAlgorithmClient(autoinit=True) # >>> # Start the GraphAlgorithmActor in another proc # >>> client.post(testdata_start_payload(list(range(1, 10)))).result() # >>> # # >>> future = client.post({'action': 'resume'}) # >>> user_request = future.result() # >>> # The infr algorithm needs a review # >>> edge, priority, edge_data = user_request[0] # >>> # # >>> client.post(_testdata_feedback_payload(edge, 'match')) # >>> client.post({'action': 'resume'}) # >>> client.post(_testdata_feedback_payload(edge, 'match')) # >>> client.post(_testdata_feedback_payload(edge, 'match')) # >>> client.post({'action': 'resume'}) # >>> client.post(_testdata_feedback_payload(edge, 'match')) # >>> client.post({'action': 'wait', 'num': float(30)}) # >>> client.post({'action': 'resume'}) # >>> client.post(_testdata_feedback_payload(edge, 'match')) # >>> client.post(_testdata_feedback_payload(edge, 'match')) # >>> client.post({'action': 'resume'}) # >>> client.post(_testdata_feedback_payload(edge, 'match')) # >>> client.post({'action': 'resume'}) # >>> client.post(_testdata_feedback_payload(edge, 'match')) # >>> client.post({'action': 'resume'}) # >>> client.post(_testdata_feedback_payload(edge, 'match')) # >>> client.post(_testdata_feedback_payload(edge, 'match')) # >>> client.post({'action': 'resume'}) # >>> client.post(_testdata_feedback_payload(edge, 'match')) # >>> client.post({'action': 'resume'}) # >>> client.post(_testdata_feedback_payload(edge, 'match')) # >>> client.post({'action': 'resume'}) # >>> client.post(_testdata_feedback_payload(edge, 'match')) # >>> client.post({'action': 'resume'}) # >>> client.post(_testdata_feedback_payload(edge, 'match')) # >>> client.post({'action': 'resume'}) # >>> client.post(_testdata_feedback_payload(edge, 'match')) # >>> client.post({'action': 'resume'}) # >>> client.post(_testdata_feedback_payload(edge, 'match')) # >>> client.post({'action': 'resume'})

actor_cls
alias of *GraphAlgorithmActor*

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

sync(*ibs*)

class wbia.web.graph_server.**GraphClient**(*aids*, *actor_config={}*, *imagesets=None*, *graph_uuid=None*, *callbacks={}*, *autoinit=False*)

Bases: object

actor_cls
alias of *GraphActor*

add_aids()

check(*edge*)

cleanup()

initialize()

post(*payload*)

refresh_metadata()

refresh_status()

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

sample(*previous_edge_list=[]*, *max_previous_edges=10*)

shutdown()

sync(*ibs*)

update(*data_list*)

class wbia.web.graph_server.**ProcessActor**
Bases: *wbia.web.graph_server.Actor*

classmethod executor(**args*, ***kwargs*)
Creates an asychronous instance of this Actor and returns the executor to manage it.

class wbia.web.graph_server.**ProcessActorExecutor**(*actor_class*, **args*, ***kwargs*)
Bases: concurrent.futures.process.ProcessPoolExecutor

post(*payload*)

class wbia.web.graph_server.**ThreadActor**
Bases: *wbia.web.graph_server.Actor*

classmethod executor(**args*, ***kwargs*)
Creates an asychronous instance of this Actor and returns the executor to manage it.

class wbia.web.graph_server.**ThreadedActorExecutor**(*actor_class*, **args*, ***kwargs*)
Bases: concurrent.futures.thread.ThreadPoolExecutor

post(*payload*)

wbia.web.graph_server.**double_review_test**()

wbia.web.graph_server.**testdata_start_payload**(*aids='all'*)

wbia.web.graph_server.**ut_to_json_encode**(*dict_*)

1.15.11 wbia.web.job_engine module

Accepts and handles requests for tasks.

Each of the following runs in its own Thread/Process.

BASICALLY DO A CLIENT/SERVER TO SPAWN PROCESSES AND THEN A PUBLISH SUBSCRIBE TO RETURN DATA

Accepter: Receives tasks and requests Delegates tasks and responds to requests Tasks are delgated to an engine

Engine: the engine accepts requests. the engine immediately responds WHERE it will be ready. the engine sends a message to the collector saying that something will be ready. the engine then executes a task. The engine is given direct access to the data.

Collector: The collector accepts requests The collector can respond: * <ResultContent> * Results are ready. * Results are not ready. * Unknown jobid. * Error computing results. * Progress percent.

References

Simple task farm, with routed replies in pyzmq http://stackoverflow.com/questions/7809200/implementing-task-farm-messaging-pattern-with-zeromq https://gist.github.com/minrk/1358832

Notes

We are essentially goint to be spawning two processes. We can test these simultaniously using

python -m wbia.web.job_engine job_engine_tester

We can test these separately by first starting the background server python -m wbia.web.job_engine job_engine_tester –bg

Alternative: python -m wbia.web.job_engine job_engine_tester –bg –no-engine python -m wbia.web.job_engine job_engine_tester –bg –only-engine –fg-engine

And then running the forground process python -m wbia.web.job_engine job_engine_tester –fg

class wbia.web.job_engine.**JobBackend**(***kwargs*)
Bases: object

get_process_alive_status()

initialize_background_processes(*dbdir=None*, *containerized=False*, *thread=False*)

class wbia.web.job_engine.**JobInterface**(*id_*, *port_dict*, *ibs=None*)
Bases: object

get_job_id_list()

get_job_metadata(*jobid*)

get_job_result(*jobid*)

get_job_status(*jobid*)

get_job_status_dict()

get_unpacked_result(*jobid*)

initialize_client_thread()
Creates a ZMQ object in this thread. This talks to background processes.

queue_interrupted_jobs()

queue_job(*action*, *callback_url=None*, *callback_method=None*, *callback_detailed=False*, *lane='slow'*, *jobid=None*, *args=None*, *kwargs=None*)
IBEIS: This is just a function that lives in the main thread and ships off a job.
FIXME: I do not like having callback_url and callback_method specified like this with args and kwargs. If these must be there then they should be specified first, or THE PREFERED OPTION IS args and kwargs should not be specified without the * syntax
The client - sends messages, and receives replies after they have been processed by the

wait_for_job_result(*jobid*, *timeout=10*, *freq=0.1*)

wbia.web.job_engine.**calculate_timedelta**(*start*, *end*)

wbia.web.job_engine.**close_job_manager**(*ibs*)

wbia.web.job_engine.**collect_queue_loop**(*port_dict*)

wbia.web.job_engine.**collector_loop**(*port_dict*, *dbdir*, *containerized*)
Service that stores completed algorithm results

wbia.web.job_engine.**convert_to_date**(*timestamp*)

wbia.web.job_engine.**delete_shelve_lock_file**(*shelve_filepath*)

wbia.web.job_engine.**engine_loop**(*id_*, *port_dict*, *dbdir*, *containerized*, *lane*)

IBEIS: This will be part of a worker process with its own IBEISController instance.

Needs to send where the results will go and then publish the results there.
The engine_loop - receives messages, performs some action, and sends a reply, preserving the leading two message parts as routing identities

wbia.web.job_engine.**engine_queue_loop**(*port_dict*, *engine_lanes*)
Specialized queue loop

wbia.web.job_engine.**fetch_job**(*ibs*, *jobid*)

wbia.web.job_engine.**get_collector_shelve_filepaths**(*collector_data*, *jobid*)

wbia.web.job_engine.**get_job_id_list**(*ibs*)
Web call that returns the list of job ids
CommandLine: # Run Everything together python -m wbia.web.job_engine –exec-get_job_status

Start job queue in its own process python -m wbia.web.job_engine job_engine_tester –bg # Start web server in its own process ./main.py –web –fg pass # Run foreground process python -m wbia.web.job_engine –exec-get_job_status:0 –fg

Example

```
>>> # xdoctest: +REQUIRES(--web-tests)
>>> # xdoctest: +REQUIRES(--job-engine-tests)
>>> from wbia.web.job_engine import *  # NOQA
>>> import wbia
>>> with wbia.opendb_bg_web('testdb1', managed=True) as web_ibs:  # , domain=
↪'http://52.33.105.88')
...     # Test get status of a job id that does not exist
...     response = web_ibs.send_wbia_request('/api/engine/job/', jobid='badjob')
```

wbia.web.job_engine.**get_job_metadata**(*ibs*, *jobid*)
Web call that returns the metadata of a job
CommandLine: # Run Everything together python -m wbia.web.job_engine –exec-get_job_metadata

Start job queue in its own process python -m wbia.web.job_engine job_engine_tester –bg # Start web server in its own process ./main.py –web –fg pass # Run foreground process python -m wbia.web.job_engine –exec-get_job_metadata:0 –fg

Example

```
>>> # xdoctest: +REQUIRES(--web-tests)
>>> # xdoctest: +REQUIRES(--slow)
>>> # xdoctest: +REQUIRES(--job-engine-tests)
>>> # xdoctest: +REQUIRES(--web-tests)
>>> from wbia.web.job_engine import *  # NOQA
>>> import wbia
>>> with wbia.opendb_bg_web('testdb1', managed=True) as web_ibs:  # , domain=
↪'http://52.33.105.88')
...     # Test get metadata of a job id that does not exist
...     response = web_ibs.send_wbia_request('/api/engine/job/metadata/', jobid=
↪'badjob')
```

wbia.web.job_engine.**get_job_result**(*ibs*, *jobid*)
Web call that returns the result of a job

wbia.web.job_engine.**get_job_status**(*ibs*, *jobid=None*)

Web call that returns the status of a job

Returns one of: received - job has been received, but not ingested yet accepted - job has been accepted (validated) queued - job has been transferred to the engine queue working - job is being worked on by the engine publishing - job is done on the engine, pushing results to collector completed | exception - job is complete or has an error

CommandLine: # Run Everything together python -m wbia.web.job_engine –exec-get_job_status

Start job queue in its own process python -m wbia.web.job_engine job_engine_tester –bg # Start web server in its own process ./main.py –web –fg pass # Run foreground process python -m wbia.web.job_engine –exec-get_job_status:0 –fg

Example

```
>>> # xdoctest: +REQUIRES(--web-tests)
>>> # xdoctest: +REQUIRES(--job-engine-tests)
>>> from wbia.web.job_engine import *  # NOQA
>>> import wbia
>>> with wbia.opendb_bg_web('testdb1', managed=True) as web_ibs:  # , domain=
↪'http://52.33.105.88')
...     # Test get status of a job id that does not exist
...     response = web_ibs.send_wbia_request('/api/engine/job/status/', jobid=
↪'badjob')
```

wbia.web.job_engine.**get_process_alive_status**(*ibs*)

wbia.web.job_engine.**get_shelve_filepaths**(*ibs*, *jobid*)

wbia.web.job_engine.**get_shelve_lock_filepath**(*shelve_filepath*)

wbia.web.job_engine.**get_shelve_value**(*shelve_filepath*, *key*)

wbia.web.job_engine.**initialize_job_manager**(*ibs*)

Starts a background zmq job engine. Initializes a zmq object in this thread that can talk to the background processes.

Run from the webserver

CommandLine: python -m wbia.web.job_engine –exec-initialize_job_manager:0

Example

```
>>> # DISABLE_DOCTEST
>>> # xdoctest: +REQUIRES(--job-engine-tests)
>>> from wbia.web.job_engine import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> from wbia.web import apis_engine
>>> from wbia.web import job_engine
>>> ibs.load_plugin_module(job_engine)
>>> ibs.load_plugin_module(apis_engine)
>>> ibs.initialize_job_manager()
>>> print('Initializqation success. Now closing')
>>> ibs.close_job_manager()
>>> print('Closing success.')
```

wbia.web.job_engine.**initialize_process_record**(*record_filepath*, *shelve_input_filepath*, *shelve_output_filepath*, *shelve_path*, *shelve_archive_path*, *jobiface_id*)

wbia.web.job_engine.**invalidate_global_cache**(*jobid*)

wbia.web.job_engine.**job_engine_tester**()

CommandLine: python -m wbia.web.job_engine –exec-job_engine_tester python -b -m wbia.web.job_engine –exec-job_engine_tester

python -m wbia.web.job_engine job_engine_tester python -m wbia.web.job_engine job_engine_tester –bg python -m wbia.web.job_engine job_engine_tester –fg

Example

```
>>> # SCRIPT
>>> from wbia.web.job_engine import *  # NOQA
>>> job_engine_tester()
```

wbia.web.job_engine.**on_collect_request**(*ibs*, *collect_request*, *collector_data*, *shelve_path*, *containerized=False*)

Run whenever the collector recieves a message

wbia.web.job_engine.**on_engine_request**(*ibs*, *jobid*, *action*, *args*, *kwargs*, *attempts=3*, *retry_delay_min=1*, *retry_delay_max=60*)

Run whenever the engine recieves a message

wbia.web.job_engine.**rcv_multipart_json**(*sock*, *num=2*, *print=<function make_module_print_func.<locals>.print>*)

helper

wbia.web.job_engine.**retry_job**(*ibs*, *jobid*)

wbia.web.job_engine.**send_multipart_json**(*sock*, *idents*, *reply*)

helper

wbia.web.job_engine.**set_shelve_value**(*shelve_filepath*, *key*, *value*)

wbia.web.job_engine.**spawn_background_process**(*func*, **args*, ***kwargs*)

wbia.web.job_engine.**touch_shelve_lock_file**(*shelve_filepath*)

wbia.web.job_engine.**update_proctitle**(*procname*, *dbname=None*)

wbia.web.job_engine.**wait_for_shelve_lock_file**(*shelve_filepath*, *timeout=600*)

1.15.12 wbia.web.prometheus module

wbia.web.prometheus.**prometheus_increment_api**(*ibs*, *tag*)

wbia.web.prometheus.**prometheus_increment_exception**(*ibs*, *tag*)

wbia.web.prometheus.**prometheus_increment_route**(*ibs*, *tag*)

wbia.web.prometheus.**prometheus_update**(*ibs*, **args*, ***kwargs*)

1.15.13 wbia.web.routes module

Dependencies: flask, tornado

wbia.web.routes.**action**(***kwargs*)

wbia.web.routes.**action_detect**(***kwargs*)

wbia.web.routes.**action_identification**(***kwargs*)

wbia.web.routes.**api_root**(***kwargs*)

wbia.web.routes.**check_engine_identification_query_object**(*global_feedback_limit=50*)

wbia.web.routes.**commit_current_query_object_names**(*query_object*, *ibs*)

Parameters

- **query_object** (*wbia.AnnotInference*) –
- **ibs** (*wbia.IBEISController*) – image analysis api

wbia.web.routes.**dbinfo**(***kwargs*)

wbia.web.routes.**delete_query_chips_graph_v2_refer**(*graph_uuid*)

wbia.web.routes.**error404**(*exception=None*)

wbia.web.routes.**gradient_magnitude**(*image_filepath*)

wbia.web.routes.**group_review**(***kwargs*)

wbia.web.routes.**image_view_api**(*gid=None*, *thumbnail=False*, *fresh=False*, ***kwargs*)

Returns the base64 encoded image of image <gid>

RESTful: Method: GET URL: /image/view/<gid>/

wbia.web.routes.**load_identification_query_object**(*autoinit=False*, *global_feedback_limit=50*, ***kwargs*)

wbia.web.routes.**load_identification_query_object_worker**(*ibs*, ***kwargs*)

wbia.web.routes.**login**(*refer=None*, **args*, ***kwargs*)

wbia.web.routes.**logout**(***kwargs*)

wbia.web.routes.**precompute_current_review_match_images**(*ibs*, *query_object*, *global_feedback_limit=50*, *view_orientation='vertical'*)

wbia.web.routes.**precompute_web_detection_thumbnails**(*ibs*, *gid_list=None*, ***kwargs*)

wbia.web.routes.**precompute_web_viewpoint_thumbnails**(*ibs*, *aid_list=None*, ***kwargs*)

wbia.web.routes.**review**(*imgsetid=None*)

wbia.web.routes.**review_annotation**(***kwargs*)

CommandLine: python -m wbia.web.app –exec-review_annotation –db PZ_Master1

Example

```
>>> # SCRIPT
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_Master1')
>>> aid_list_ = ibs.find_unlabeled_name_members(suspect_yaws=True)
>>> aid_list = ibs.filter_aids_to_quality(aid_list_, 'good', unknown_ok=False)
>>> ibs.start_web_annot_groupreview(aid_list)
```

wbia.web.routes.**review_annotation_canonical**(*imgsetid=None*, *samples=200*, *species=None*, *version=1*, ***kwargs*)

wbia.web.routes.**review_annotation_dynamic**(***kwargs*)

wbia.web.routes.**review_cameratrap**(***kwargs*)

wbia.web.routes.**review_contour**(*part_rowid=None*, *imgsetid=None*, *previous=None*, ***kwargs*)

wbia.web.routes.**review_demographics**(*species='zebra_grevys'*, *aid=None*, ***kwargs*)

wbia.web.routes.**review_detection**(*gid=None*, *only_aid=None*, *refer_aid=None*, *imgsetid=None*, *previous=None*, *previous_only_aid=None*, *staged_super=False*, *progress=None*, ***kwargs*)

wbia.web.routes.**review_detection_canonical**(*aid=None*, *imgsetid=None*, *previous=None*, *previous_only_aid=None*, ***kwargs*)

wbia.web.routes.**review_detection_dynamic**(***kwargs*)

wbia.web.routes.**review_identification**(*aid1=None*, *aid2=None*, *use_engine=False*, *global_feedback_limit=50*, ***kwargs*)

CommandLine: python -m wbia.web.routes review_identification –db PZ_Master1 python -m wbia.web.routes review_identification –db PZ_MTEST python -m wbia.web.routes review_identification –db testdb1 –show

Example

```
>>> # SCRIPT
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> with wbia.opendb_with_web('testdb1') as (ibs, client):
...     resp = client.get('/review/identification/lnbnn/')
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.render_html(resp.data.decode('utf8'))
>>> ut.show_if_requested()
```

wbia.web.routes.**review_identification_graph**(*graph_uuid=None*, *aid1=None*, *aid2=None*, *annot_uuid_list=None*, *hardcase=None*, *view_orientation='vertical'*, *view_version=1*, *hogwild=False*, *hogwild_species=None*, *creation_imageset_rowid_list=None*, *kaia=False*, *census=False*, *backend='graph_algorithm'*, ***kwargs*)

CommandLine: python -m wbia.web.routes review_identification_graph –db PZ_Master1 python -m wbia.web.routes review_identification_graph –db PZ_MTEST python -m wbia.web.routes review_identification_graph –db testdb1 –show

python -m wbia –db PZ_MTEST –web –browser –url=/review/identification/graph/ –noengine

Example

```
>>> # SCRIPT
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> with wbia.opendb_with_web('testdb1') as (ibs, client):
...     resp = client.get('/review/identification/graph/')
>>> ut.quit_if_noshow()
```

(continues on next page)

(continued from previous page)

```
>>> import wbia.plottool as pt
>>> ut.render_html(resp.data.decode('utf8'))
>>> ut.show_if_requested()
```

wbia.web.routes.**review_identification_graph_refer**(*imgsetid*, *species=None*, *tier=1*, *year=2019*, *option=None*, *backend='graph_algorithm'*, ***kwargs*)

wbia.web.routes.**review_identification_hardcase**(**args*, ***kwargs*)

CommandLine: python -m wbia –db PZ_MTEST –web –browser –url=/review/identification/hardcase/ python -m wbia –db PZ_MTEST –web –browser –url=/review/identification/graph/

```
>>> # SCRIPT
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> with wbia.opendb_with_web('PZ_Master1') as (ibs, client):
...     resp = client.get('/review/identification/hardcase/')
```

Ignore: import wbia ibs, aids = wbia.testdata_aids('PZ_Master1', a=':species=zebra_plains') logger.info(len(aids)) infr = wbia.AnnotInference(ibs, aids=aids, autoinit='staging') infr.load_published() logger.info(ut.repr4(infr.status())) infr.qt_review_loop()

verifiers = infr.learn_evaluation_verifiers()

verif = verifiers['match_state'] edges = list(infr.edges()) real = list(infr.edge_decision_from(edges)) hardness = 1 - verif.easiness(edges, real)

wbia.web.routes.**review_part_types**(*part_rowid=None*, *imgsetid=None*, *previous=None*, *hotkeys=8*, *refresh=False*, *previous_part_types=None*, ***kwargs*)

wbia.web.routes.**review_quality**(***kwargs*)

PZ Needs Tags: 17242 14468 14427 15946 14771 14084 4102 6074 3409
GZ Needs Tags; 1302

CommandLine: python -m wbia.web.app –exec-review_quality –db PZ_Master1 python -m wbia.web.app –exec-review_quality –db GZ_Master1 python -m wbia.web.app –exec-review_quality –db GIRM_Master1

Example

```
>>> # SCRIPT
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list_ = ibs.find_unlabeled_name_members(qual=True)
>>> valid_views = ['primary', 'primary1', 'primary-1']
>>> aid_list = ibs.filter_aids_to_viewpoint(aid_list_, valid_views, unknown_
↪ok=False)
>>> ibs.start_web_annot_groupreview(aid_list)
```

wbia.web.routes.**review_species**(*hotkeys=8*, *refresh=False*, *previous_species_rowids=None*, ***kwargs*)

wbia.web.routes.**review_species_holding**(**args*, ***kwargs*)

wbia.web.routes.**review_splits**(*aid=None*, ***kwargs*)

wbia.web.routes.**review_viewpoint**(**kwargs)

CommandLine: python -m wbia.web.app –exec-review_viewpoint –db PZ_Master1

Example

```
>>> # SCRIPT
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_Master1')
>>> aid_list_ = ibs.find_unlabeled_name_members(suspect_yaws=True)
>>> aid_list = ibs.filter_aids_to_quality(aid_list_, 'good', unknown_ok=False)
>>> ibs.start_web_annot_groupreview(aid_list)
```

wbia.web.routes.**review_viewpoint2**(**kwargs)

CommandLine: python -m wbia.web.app –exec-review_viewpoint –db PZ_Master1

Example

```
>>> # SCRIPT
>>> from wbia.other.ibsfuncs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_Master1')
>>> aid_list_ = ibs.find_unlabeled_name_members(suspect_yaws=True)
>>> aid_list = ibs.filter_aids_to_quality(aid_list_, 'good', unknown_ok=False)
>>> ibs.start_web_annot_groupreview(aid_list)
```

wbia.web.routes.**review_viewpoint3**(**kwargs)

wbia.web.routes.**root**(**kwargs)

wbia.web.routes.**sightings**(*html_encode=True*, *complete=True*, *include_images=False*, *kaia=False*, **kwargs)

wbia.web.routes.**upload**(*args, **kwargs)

wbia.web.routes.**upload_zip**(**kwargs)

wbia.web.routes.**view**(**kwargs)

wbia.web.routes.**view_advanced0**(**kwargs)

wbia.web.routes.**view_advanced1**(**kwargs)

wbia.web.routes.**view_advanced2**(**kwargs)

wbia.web.routes.**view_advanced3**(**kwargs)

wbia.web.routes.**view_advanced4**(**kwargs)

wbia.web.routes.**view_annotations**(**kwargs)

wbia.web.routes.**view_graphs**(*sync=False*, **kwargs)

wbia.web.routes.**view_images**(**kwargs)

wbia.web.routes.**view_imagesets**(**kwargs)

wbia.web.routes.**view_jobs**(**kwargs)

wbia.web.routes.**view_map**(**kwargs)

wbia.web.routes.**view_names**(**kwargs)

wbia.web.routes.**view_parts**(*pid_list=None*, *aid_list=None*, *gid_list=None*, *imgsetid_list=None*, *page=1*, ***kwargs*)

wbia.web.routes.**view_viewpoints**(***kwargs*)

wbia.web.routes.**wb_counts**(***kwargs*)

wbia.web.routes.**wb_counts_alias1**(***kwargs*)

wbia.web.routes.**wb_counts_alias2**(***kwargs*)

1.15.14 wbia.web.routes_ajax module

Dependencies: flask, tornado

wbia.web.routes_ajax.**annotation_src**(*aid=None*, *ibs=None*, ***kwargs*)

wbia.web.routes_ajax.**image_src**(*gid=None*, *thumbnail=False*, *ibs=None*, ***kwargs*)

wbia.web.routes_ajax.**image_src_ext**(**args*, ***kwargs*)

wbia.web.routes_ajax.**image_src_path**(*gpath*, *orient='auto'*, ***kwargs*)

wbia.web.routes_ajax.**part_src**(*part_rowid*, ***kwargs*)

wbia.web.routes_ajax.**probchip_src**(*aid=None*, *ibs=None*, ***kwargs*)

wbia.web.routes_ajax.**set_cookie**(***kwargs*)

1.15.15 wbia.web.routes_csv module

Dependencies: flask, tornado

wbia.web.routes_csv.**download_associations_list**(***kwargs*)

wbia.web.routes_csv.**download_associations_matrix**(***kwargs*)

wbia.web.routes_csv.**download_sightings**(***kwargs*)

wbia.web.routes_csv.**get_aid_list_csv**(***kwargs*)

wbia.web.routes_csv.**get_annotation_special_kaia_dung_samples**(***kwargs*)

wbia.web.routes_csv.**get_annotation_special_megan**(***kwargs*)

wbia.web.routes_csv.**get_annotation_special_monica_laurel_max**(*desired_species=None*, ***kwargs*)

wbia.web.routes_csv.**get_associations_dict**(*ibs*, *desired_species=None*, ***kwargs*)

wbia.web.routes_csv.**get_demographic_info**(***kwargs*)

wbia.web.routes_csv.**get_gid_list_csv**(***kwargs*)

wbia.web.routes_csv.**get_gid_with_aids_csv**(***kwargs*)

wbia.web.routes_csv.**get_image_info**(***kwargs*)

wbia.web.routes_csv.**get_nid_with_gids_csv**(***kwargs*)

1.15.16 wbia.web.routes_demo module

Dependencies: flask, tornado

wbia.web.routes_demo.**demo**(**args*, ***kwargs*)

1.15.17 wbia.web.routes_experiments module

Dependencies: flask, tornado

wbia.web.routes_experiments.**experiment_init_db**(*tag*)

wbia.web.routes_experiments.**experiments_image_src**(*tag=None*, ***kwargs*)

wbia.web.routes_experiments.**experiments_interest**(*dbtag1='demo-jasonp'*, *dbtag2='demo-chuck'*, ***kwargs*)

wbia.web.routes_experiments.**experiments_voting**(***kwargs*)

wbia.web.routes_experiments.**experiments_voting_area_src**(*ibs*, *aoi=False*, ***kwargs*)

wbia.web.routes_experiments.**experiments_voting_bbox_width**(*ibs*, ***kwargs*)

wbia.web.routes_experiments.**experiments_voting_center_src**(*ibs*, *aoi=False*, ***kwargs*)

wbia.web.routes_experiments.**experiments_voting_counts**(*ibs*, ***kwargs*)

wbia.web.routes_experiments.**experiments_voting_initialize**(*enabled_list=None*)

wbia.web.routes_experiments.**experiments_voting_variance**(*ibs*, *team_index*, ***kwargs*)

wbia.web.routes_experiments.**view_experiments**(***kwargs*)

wbia.web.routes_experiments.**voting_data**(*method=3*, *option='inclusive'*, *species='all'*, *team1=True*, *team2=True*, *team3=True*, *team4=True*, *team5=True*)

wbia.web.routes_experiments.**voting_uuid_list**(*ibs*, *team_list*)

1.15.18 wbia.web.routes_submit module

Dependencies: flask, tornado

wbia.web.routes_submit.**group_review_submit**(***kwargs*)

CommandLine: python -m wbia.web.app –exec-group_review_submit

Example

```
>>> # UNSTABLE_DOCTEST
>>> from wbia.web.app import *  # NOQA
>>> import wbia
>>> import wbia.web
>>> ibs = wbia.opendb('testdb1')
>>> aid_list = ibs.get_valid_aids()[::2]
>>> ibs.start_web_annot_groupreview(aid_list)
```

wbia.web.routes_submit.**submit_annotation**(***kwargs*)

wbia.web.routes_submit.**submit_annotation_canonical**(*samples=200*, *species=None*, *version=1*, ***kwargs*)

wbia.web.routes_submit.**submit_cameratrap**(***kwargs*)

wbia.web.routes_submit.**submit_contour**(***kwargs*)

wbia.web.routes_submit.**submit_demographics**(*species='zebra_grevys'*, ***kwargs*)

wbia.web.routes_submit.**submit_detection**(***kwargs*)

wbia.web.routes_submit.**submit_identification**(***kwargs*)

wbia.web.routes_submit.**submit_identification_v2**(*graph_uuid*, ***kwargs*)

wbia.web.routes_submit.**submit_identification_v2_kaia**(*graph_uuid*, ***kwargs*)

wbia.web.routes_submit.**submit_login**(*name*, *organization*, *refer=None*, **args*, ***kwargs*)

wbia.web.routes_submit.**submit_part_types**(***kwargs*)

wbia.web.routes_submit.**submit_quality**(***kwargs*)

wbia.web.routes_submit.**submit_species**(***kwargs*)

wbia.web.routes_submit.**submit_splits**(***kwargs*)

wbia.web.routes_submit.**submit_viewpoint**(***kwargs*)

wbia.web.routes_submit.**submit_viewpoint2**(***kwargs*)

wbia.web.routes_submit.**submit_viewpoint3**(***kwargs*)

1.15.19 wbia.web.test_api module

This is a proof of concept for connecting to an authenticated Qubica Server

wbia.web.test_api.**get_api_result**(*uri*, *user_email=None*, *user_enc_pass=None*, ***kwargs*)
Make a GET API request to the server

wbia.web.test_api.**get_authorization_header**(*uri*, *user_email=None*, *user_enc_pass=None*)

wbia.web.test_api.**get_signature**(*key*, *message*)

wbia.web.test_api.**post_api_result**(*uri*, *user_email=None*, *user_enc_pass=None*, ***kwargs*)
Make a GET API request to the server

wbia.web.test_api.**run_test_api**()
CommandLine: python -m wbia.web.test_api –test-run_test_api

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.web.test_api import *  # NOQA
>>> response = run_test_api()
>>> print('Server response: %r' % (response, ))
>>> result = response
(200, u'{"status": {"cache": -1, "message": "", "code": 200, "success": true},
↪"response": "testdb1"}', <bound method Response.json of <Response [200]>>)
```

1.15.20 Module contents

1.16 wbia.__main__

Runs IBIES gui

wbia.__main__.**run_wbia**()
CommandLine: python -m wbia python -m wbia find_installed_tomcat python -m wbia get_annot_groundtruth:1

wbia.__main__.**smoke_test**()

1.17 wbia._devcmds_wbia

DEPRICATE MOST OF THIS FILE IN FAVOR OF DOCTEST SCRIPTS

wbia._devcmds_wbia.**export**(*ibs*, *aid_pairs=None*)
3 - 4 different animals 2 views of each matching keypoint coordinates on each annotation

wbia._devcmds_wbia.**openworkdirs_test**()
problems: PZ_DanExt_All PZ_DanExt_Test GZ_March2012 Wildebeest_ONLY_MATCHES
python dev.py –convert –dbdir /raid/work/PZ_Marianne –force-delete python dev.py –convert –dbdir /raid/work/SL_Siva –force-delete python dev.py –convert –dbdir /raid/work/PZ_SweatwaterSmall –force-delete

1.18 wbia._devscript

wbia._devscript.**devcmd**(**args*)
Decorator which registers a function as a developer command

wbia._devscript.**devprecmd**(**args*)
Decorator which registers a function as a developer precommand

wbia._devscript.**hack_argv**(*arg*)

1.19 wbia._wbia_object

class wbia._wbia_object.**ObjectList1D**(*rowids*, *ibs*, *config=None*, *caching=False*, *asarray=False*)
Bases: utool.util_dev.NiceRepr, utool.util_class.HashComparable2

An object that efficiently operates on a list of wbia objects using vectorized code. Single instances can be returned as ObjectScalar0D's

chunks(*chunksize*)

compress(*flags*)

disconnect()
Disconnects object from the state of the database. All information has been assumed to be preloaded.

group(*labels*)
group as list

group_indicies(*labels*)

group_items(*labels*)
group as dict

group_uuid()

loc(*rowids*)
Lookup subset by rowids

lookup_idxs(*rowids*)
Lookup subset indicies by rowids

preload(**attrs*)

scalars()

set_caching(*flag*)

take(*idxs*)
Creates a subset of the list using the specified indices.

take_column(*keys*)

view(*rowids=None*)
Like take, but returns a view proxy that maps to the original parent

class wbia._wbia_object.**ObjectScalar0D**(*obj1d*)
Bases: utool.util_dev.NiceRepr, utool.util_class.HashComparable2

This actually stores a ObjectList1D of length 1 and simply calls those functions where available

class wbia._wbia_object.**ObjectView1D**(*rowids*, *obj1d*, *cache=None*)
Bases: utool.util_dev.NiceRepr

Allows for proxy caching.

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia._wbia_object import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aids = ibs.get_valid_aids()
>>> a = self = annots = ibs.annots(aids)
>>> rowids = [1, 1, 3, 2, 1, 2]
>>> self = v = a.view(rowids)
>>> assert np.all(v.vecs[0] == v.vecs[1])
>>> assert v.vecs[0] is v.vecs[1]
>>> assert v.vecs[0] is not v.vecs[2]
```

view(*rowids*)
returns a view of a view that uses the same per-item cache

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia._wbia_object import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aids = ibs.get_valid_aids()
```

(continues on next page)

(continued from previous page)

```
>>> annots = ibs.annots(aids)
>>> self = annots.view(annots._rowids)
>>> v1 = self.view([1, 1, 2, 3, 1, 2])
>>> v2 = self.view([3, 4, 5])
>>> v3 = self.view([1, 4])
>>> v4 = self.view(3)
>>> lazy4 = v4._make_lazy_dict()
>>> assert v1.vecs[0] is v3.vecs[0]
>>> assert v2._cache is self._cache
>>> assert v2._cache is v1._cache
```

1.20 wbia.annotmatch_funcs

wbia.annotmatch_funcs.**add_annotmatch_undirected**(*ibs*, *aids1*, *aids2*, ***kwargs*)

wbia.annotmatch_funcs.**get_annot_has_reviewed_matching_aids**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)

wbia.annotmatch_funcs.**get_annot_num_reviewed_matching_aids**(*ibs*, *aid1_list*, *eager=True*, *nInput=None*)

Parameters

- **aid_list** (*int*) – list of annotation ids
- **eager** (*bool*) –
- **nInput** (*None*) –

Returns num_annot_reviewed_list

Return type list

CommandLine: python -m wbia.annotmatch_funcs –test-get_annot_num_reviewed_matching_aids

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.annotmatch_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb2')
>>> aid1_list = ibs.get_valid_aids()
>>> eager = True
>>> nInput = None
>>> num_annot_reviewed_list = get_annot_num_reviewed_matching_aids(ibs, aid_list,␣
↪eager, nInput)
>>> result = str(num_annot_reviewed_list)
>>> print(result)
```

wbia.annotmatch_funcs.**get_annot_pair_is_reviewed**(*ibs*, *aid1_list*, *aid2_list*)

Parameters

- **aid1_list** (*list*) –
- **aid2_list** (*list*) –

Returns annotmatch_reviewed_list

Return type list

CommandLine: python -m wbia.annotmatch_funcs –test-get_annot_pair_is_reviewed

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.annotmatch_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb2')
>>> aid_list = ibs.get_valid_aids()
>>> pairs = list(ut.product(aid_list, aid_list))
>>> aid1_list = ut.get_list_column(pairs, 0)
>>> aid2_list = ut.get_list_column(pairs, 1)
>>> annotmatch_reviewed_list = get_annot_pair_is_reviewed(ibs, aid1_list, aid2_
↪list)
>>> reviewed_pairs = ut.compress(pairs, annotmatch_reviewed_list)
>>> result = len(reviewed_pairs)
>>> print(result)
104
```

wbia.annotmatch_funcs.**get_annot_pair_timedelta**(*ibs*, *aid_list1*, *aid_list2*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list1** (*int*) – list of annotation ids
- **aid_list2** (*int*) – list of annotation ids

Returns timedelta_list

Return type list

CommandLine: python -m wbia.annotmatch_funcs –test-get_annot_pair_timedelta

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.annotmatch_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('PZ_MTEST')
>>> aid_list = ibs.get_valid_aids(hasgt=True)
>>> unixtimes = ibs.get_annot_image_unixtimes_asfloat(aid_list)
>>> aid_list = ut.compress(aid_list, ~np.isnan(unixtimes))
>>> gt_aids_list = ibs.get_annot_groundtruth(aid_list, daid_list=aid_list)
>>> flags = np.array(list(map(len, gt_aids_list))) > 0
>>> aid_list1 = ut.compress(aid_list, flags)[0:5]
>>> aid_list2 = ut.take_column(gt_aids_list, 0)[0:5]
>>> timedelta_list = ibs.get_annot_pair_timedelta(aid_list1, aid_list2)
>>> result = ut.repr2(timedelta_list, precision=1)
>>> print(result)
np.array([7.6e+07, 7.6e+07, 2.4e+06, 2.0e+08, 9.7e+07])
```

wbia.annotmatch_funcs.**get_annot_reviewed_matching_aids**(*ibs*, *aid_list*, *eager=True*, *nInput=None*)

Returns a list of the aids that were reviewed as candidate matches to the input aid

wbia.annotmatch_funcs.**get_annotedge_timedelta**(*ibs*, *edges*)

wbia.annotmatch_funcs.**get_annotedge_viewdist**(*ibs*, *edges*)

wbia.annotmatch_funcs.**get_annotmatch_aids**(*ibs*, *annotmatch_rowid_list*)

wbia.annotmatch_funcs.**get_annotmatch_rowid_from_edges**(*ibs*, *aid_pairs*)

Edegs are undirected

wbia.annotmatch_funcs.**get_annotmatch_rowid_from_undirected_superkey**(*ibs*, *aids1*, *aids2*)

wbia.annotmatch_funcs.**get_annotmatch_rowids_between**(*ibs*, *aids1*, *aids2*, *method=None*)

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.annotmatch_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('PZ_MTEST')
>>> aids1 = aids2 = [1, 2, 3, 4, 5, 6]
>>> rowids_between = ibs.get_annotmatch_rowids_between
>>> ams1 = sorted(rowids_between(aids1, aids2, method=1))
>>> ams2 = sorted(rowids_between(aids1, aids2, method=2))
>>> assert len(ub.find_duplicates(ams1)) == 0
>>> assert len(ub.find_duplicates(ams2)) == 0
>>> assert sorted(ams2) == sorted(ams1)
```

wbia.annotmatch_funcs.**get_annotmatch_rowids_between_groups**(*ibs*, *aids1_list*, *aids2_list*)

wbia.annotmatch_funcs.**get_annotmatch_rowids_from_aid**(*ibs*, *aid_list*, *eager=True*, *nInput=None*, *force_method=None*)

Undirected version Returns a list of the aids that were reviewed as candidate matches to the input aid aid_list = ibs.get_valid_aids()

CommandLine: python -m wbia.annotmatch_funcs –exec-get_annotmatch_rowids_from_aid python -m wbia.annotmatch_funcs –exec-get_annotmatch_rowids_from_aid:1 –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.annotmatch_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> ut.exec_funckw(ibs.get_annotmatch_rowids_from_aid, globals())
>>> aid_list = ibs.get_valid_aids()[0:4]
>>> annotmatch_rowid_list = ibs.get_annotmatch_rowids_from_aid(aid_list,
>>>                                                            eager, nInput)
>>> result = ('annotmatch_rowid_list = %s' % (str(annotmatch_rowid_list),))
>>> print(result)
```

wbia.annotmatch_funcs.**get_annotmatch_rowids_from_aid1**(*ibs*, *aid1_list*, *eager=True*, *nInput=None*)

TODO autogenerate

Returns a list of the aids that were reviewed as candidate matches to the input aid

aid_list = ibs.get_valid_aids() :param ibs: wbia controller object :type ibs: IBEISController :param aid1_list: :type aid1_list: list :param eager: (default = True) :type eager: bool :param nInput: (default = None) :type nInput: None

Returns annotmatch_rowid_list

Return type list

wbia.annotmatch_funcs.**get_annotmatch_rowids_from_aid2**(*ibs*, *aid2_list*, *eager=True*, *nInput=None*, *force_method=None*)

This one is slow because aid2 is the second part of the index Returns a list of the aids that were reviewed as candidate matches to the input aid

wbia.annotmatch_funcs.**get_annotmatch_rowids_in_cliques**(*ibs*, *aids_list*)

wbia.annotmatch_funcs.**get_match_text**(*ibs*, *aid1*, *aid2*)

wbia.annotmatch_funcs.**get_match_truth**(*ibs*, *aid1*, *aid2*)

wbia.annotmatch_funcs.**get_match_truths**(*ibs*, *aids1*, *aids2*)

Uses NIDS to verify truth. TODO: rectify with annotmatch table

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aids1** (*list*) –
- **aids2** (*list*) –

Returns

truth_codes - see wbia.constants.EVIDENCE_DECISION.INT_TO_CODE for code definitions

Return type list[int]

CommandLine: python -m wbia.other.ibsfuncs –test-get_match_truths

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.annotmatch_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aids1 = ibs.get_valid_aids()
>>> aids2 = ut.list_roll(ibs.get_valid_aids(), -1)
>>> truth_codes = get_match_truths(ibs, aids1, aids2)
>>> print('truth_codes = %s' % ut.repr2(truth_codes))
>>> target = np.array([3, 1, 3, 3, 1, 0, 0, 3, 3, 3, 3, 0, 3])
>>> assert np.all(truth_codes == target)
```

wbia.annotmatch_funcs.**set_annot_pair_as_negative_match**(*ibs*, *aid1*, *aid2*, *dryrun=False*, *on_nontrivial_split=None*, *logger_=None*)

TODO: ELEVATE THIS FUNCTION

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid1** (*int*) – annotation id
- **aid2** (*int*) – annotation id
- **dryrun** (*bool*) –

CommandLine: python -m wbia.annotmatch_funcs –test-set_annot_pair_as_negative_match

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.annotmatch_funcs import *  # NOQA
>>> import wbia
```

(continues on next page)

(continued from previous page)

```
>>> ibs = wbia.opendb('testdb1')
>>> aid1, aid2 = ibs.get_valid_aids()[0:2]
>>> dryrun = True
>>> result = set_annot_pair_as_negative_match(ibs, aid1, aid2, dryrun)
>>> print(result)
>>> ibs.delete_names(ibs.get_valid_nids()[-1])  # clean up
```

wbia.annotmatch_funcs.**set_annot_pair_as_positive_match**(*ibs*, *aid1*, *aid2*, *dryrun=False*, *on_nontrivial_merge=None*, *logger_=None*)

Safe way to perform links. Errors on invalid operations.

TODO: ELEVATE THIS FUNCTION Change into make_task_set_annot_pair_as_positive_match and it returns what needs to be done.

Need to test several cases: uknown, unknown knownA, knownA knownB, knownA unknown, knownA knownA, unknown

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid1** (*int*) – query annotation id
- **aid2** (*int*) – matching annotation id

CommandLine: python -m wbia.annotmatch_funcs –test-set_annot_pair_as_positive_match

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.annotmatch_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> aid1, aid2 = ibs.get_valid_aids()[0:2]
>>> dryrun = True
>>> status = set_annot_pair_as_positive_match(ibs, aid1, aid2, dryrun)
>>> print(status)
```

wbia.annotmatch_funcs.**set_annot_pair_as_reviewed**(*ibs*, *aid1*, *aid2*)

denote that this match was reviewed and keep whatever status it is given

1.21 wbia.annots

class wbia.annots.**AnnotGroups**(*annots_list*, *ibs*)

Bases: utool.util_dev.NiceRepr

Effciently handle operations on multiple groups of annotations

age_months_est_max

age_months_est_min

aid

aids

all_tags

annotmatch_tags

bbox_area

bboxes

case_tags

contact_aids

detect_confidence

encounter_text

exemplar_flags

gids

groundfalse

groundtruth

has_groundtruth

has_reviewed_matching_aids

hashid_semantic_uuid

hashid_uuid

hashid_visual_uuid

image_contributor_tag

image_datetime_str

image_gps

image_gps2

image_set_texts

image_unixtimes_asfloat

image_uuids

images

imgset_uuids

imgsetids

match_tags
: returns pairwise tags within the annotation group

multiple

name_uuids

names

nids

notes

num_contact_aids

num_groundtruth

num_reviewed_matching_aids

occurrence_text

otherimage_aids

parent_aid

primary_imageset

qualities

quality_texts

reviewed

reviewed_matching_aids

rrr(*verbose=True*, *reload_module=True*)
special class reloading function This function is often injected as rrr of classes

semantic_uuids

sex

sex_texts

species

species_rowids

species_texts

species_uuids

static_encounter

thetas

uuids

verts

viewpoint_code

viewpoint_int

visual_uuids

yaw_texts

yaws

yaws_asfloat

class wbia.annots.**AnnotMatches**(*rowids*, *ibs*, *config=None*, *caching=False*, *asarray=False*)
Bases: *wbia._wbia_object.ObjectList1D*

Represents a group of annotations. Efficiently accesses properties from a database using lazy evaluation.
CommandLine: python -m wbia.annots Annots

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.annots import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aids = ibs.get_valid_aids()
```

(continues on next page)

(continued from previous page)

```
>>> annots = Annots(aids, ibs)
>>> ams = annots.get_am_rowids()
>>> matches = self = ibs.matches()
>>> ed1 = matches.evidence_decision
>>> md2 = matches.meta_decision
>>> table = ibs.db.get_table_as_pandas('annotmatch')
>>> assert len(table) == len(matches)
```

aid1

aid2

case_tags

confidence

confidence_code

count

edges

evidence_decision

evidence_decision_code

meta_decision

meta_decision_code

posixtime_modified

reviewer

rrr()
Dynamic module reloading

tag_text

class wbia.annots.**Annots**(*rowids*, *ibs*, *config=None*, *caching=False*, *asarray=False*)
Bases: *wbia._wbia_object.ObjectList1D*

Represents a group of annotations. Efficiently accesses properties from a database using lazy evaluation.
CommandLine: python -m wbia.annots Annots

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.annots import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aids = ibs.get_valid_aids()
>>> a = self = annots = Annots(aids, ibs)
>>> a.preload('vecs', 'kpts', 'nids')
>>> print(Annots.mro())
>>> print(ut.depth_profile(a.vecs))
>>> print(a)
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.annots import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aids = ibs.get_valid_aids()
>>> a = self = annots = Annots(aids, ibs)
>>> a.preload('vecs', 'kpts', 'nids')
>>> a.disconnect()
>>> assert 'vecs' in a._internal_attrs.keys()
>>> assert a._ibs is None
>>> ut.assert_raises(KeyError, a._get_num_feats)
>>> a._ibs = ibs
>>> assert len(a._get_num_feats()) > 0
```

age_months_est_max

age_months_est_min

aid

aids

all_tags

annotmatch_tags

append_tags(*tags*)

bbox_area

bboxes

case_tags

chip_dlensqrd

chip_fpath

chip_size

chip_sizes

chip_thumbpath

chip_thumbtup

chips

contact_aids

detect_confidence

encounter_text

exemplar_flags

feat_rowids

featweight_rowids

fgweights

fgweights_subset

get_aidpairs()

get_am_aidpairs()

get_am_rowids(*internal=True*)
if *internal is True* returns am rowids only between annotations in this Annots object, otherwise returns any am rowid that contains any aid in this Annots object.

get_am_rowids_and_pairs()

get_name_image_closure()

get_speeds()

get_stats(***kwargs*)

gids

gps

groundfalse

groundtruth

group2(*by*)
self = annots by = annots.static_encounter encounters = annots.group2(annots.static_encounter)

has_groundtruth

has_reviewed_matching_aids

hashid_semantic_uuid

hashid_uuid

hashid_visual_uuid

hog_hog

hog_img

image_contributor_tag

image_datetime_str

image_gps

image_gps2

image_set_texts

image_unixtimes_asfloat

image_uuids

imgset_uuids

imgsetids

kpts

kpts_distinctiveness

matches(*internal=True*)

multiple

name

name_uuids

names

nid

nids

notes

num_contact_aids

num_feats

num_groundtruth

num_reviewed_matching_aids

occurrence_text

otherimage_aids

parent_aid

primary_imageset

print_stats(***kwargs*)

probchip_img

qual

qualities

quality_texts

rchip

rchip_fpath

remove_tags(*tags*)

reviewed

reviewed_matching_aids

rrr()
Dynamic module reloading

semantic_uuids

sex

sex_texts

show(**args*, ***kwargs*)

species

species_rowids

species_texts

species_uuids

static_encounter

thetas

time

unary_tags

uuids

vecs

vecs_cache

vecs_subset

verts

viewpoint_code

viewpoint_int

visual_uuids

yaw

yaw_texts

yaws

yaws_asfloat

wbia.annots.**annots**(*ibs*, *aids=None*, *uuids=None*, ***kwargs*)
Makes an Annots object

wbia.annots.**matches**(*ibs*, *ams=None*, *edges=None*, *uuid_edges=None*, ***kwargs*)
Makes an Annots object

1.22 wbia.constants

It is better to use constant variables instead of hoping you spell the same string correctly every time you use it. (Also it makes it much easier if a string name changes)

class wbia.constants.**CONFIDENCE**
Bases: object

ABSOLUTELY_SURE = 4

class CODE
Bases: object

ABSOLUTELY_SURE = 'absolutely_sure'

GUESSING = 'guessing'

NOT_SURE = 'not_sure'

PRETTY_SURE = 'pretty_sure'

UNKNOWN = 'unspecified'

CODE_TO_INT = {'absolutely_sure': 4, 'guessing': 1, 'not_sure': 2, 'pretty_sure':

CODE_TO_NICE = {'absolutely_sure': 'Doubtless', 'guessing': 'Guessing', 'not_sure':

GUESSING = 1

INT_TO_CODE = OrderedDict([(4, 'absolutely_sure'), (3, 'pretty_sure'), (2, 'not_sure')

INT_TO_NICE = OrderedDict([(4, 'Doubtless'), (3, 'Sure'), (2, 'Unsure'), (1, 'Guessing

class NICE
Bases: object

ABSOLUTELY_SURE = 'Doubtless'

```
        GUESSING = 'Guessing'
        NOT_SURE = 'Unsure'
        PRETTY_SURE = 'Sure'
        UNKNOWN = 'Unspecified'
    NICE_TO_CODE = {'Doubtless': 'absolutely_sure', 'Guessing': 'guessing', 'Sure': 'pr
    NICE_TO_INT = {'Doubtless': 4, 'Guessing': 1, 'Sure': 3, 'Unspecified': None, 'Uns
    NOT_SURE = 2
    PRETTY_SURE = 3
    UNKNOWN = None
```

class wbia.constants.**EVIDENCE_DECISION**

Bases: object

TODO: change to EVIDENCE_DECISION / VISUAL_DECISION Enumerated types of review codes and texts

Notes

Unreviewed: Not comparared yet. nomatch: Visually comparable and the different match: Visually comparable and the same notcomp: Not comparable means it is actually impossible to determine. unknown: means that it was reviewed, but we just can't figure it out.

```
    class CODE
        Bases: object
        INCOMPARABLE = 'notcomp'
        NEGATIVE = 'nomatch'
        POSITIVE = 'match'
        UNKNOWN = 'unknown'
        UNREVIEWED = 'unreviewed'
    CODE_TO_INT = {'match': 1, 'nomatch': 0, 'notcomp': 2, 'unknown': 3, 'unreviewed':
    CODE_TO_NICE = {'match': 'Positive', 'nomatch': 'Negative', 'notcomp': 'Incomparabl
    INCOMPARABLE = 2
    INT_TO_CODE = OrderedDict([(1, 'match'), (0, 'nomatch'), (2, 'notcomp'), (3, 'unknown'
    INT_TO_NICE = OrderedDict([(1, 'Positive'), (0, 'Negative'), (2, 'Incomparable'), (3,
    MATCH_CODE = {'match': 1, 'nomatch': 0, 'notcomp': 2, 'unknown': 3, 'unreviewed':
    NEGATIVE = 0
    class NICE
        Bases: object
        INCOMPARABLE = 'Incomparable'
        NEGATIVE = 'Negative'
        POSITIVE = 'Positive'
        UNKNOWN = 'Unknown'
        UNREVIEWED = 'Unreviewed'
```

NICE_TO_CODE = {'Incomparable': 'notcomp', 'Negative': 'nomatch', 'Positive': 'match

NICE_TO_INT = {'Incomparable': 2, 'Negative': 0, 'Positive': 1, 'Unknown': 3, 'Unr

POSITIVE = 1

UNKNOWN = 3

UNREVIEWED = None

class wbia.constants.META_DECISION

Bases: object

Enumerated types of review codes and texts

Notes

unreviewed: we dont have a meta decision same: we know this is the same animal through non-visual means diff: we know this is the different animal through non-visual means

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.constants import *  # NOQA
>>> assert hasattr(META_DECISION, 'CODE')
>>> assert hasattr(META_DECISION, 'NICE')
>>> code1 = META_DECISION.INT_TO_CODE[META_DECISION.NULL]
>>> code2 = META_DECISION.CODE.NULL
>>> assert code1 == code2
>>> nice1 = META_DECISION.INT_TO_NICE[META_DECISION.NULL]
>>> nice2 = META_DECISION.NICE.NULL
>>> assert nice1 == nice2
```

class CODE

Bases: object

DIFF = 'diff'

NULL = 'null'

SAME = 'same'

CODE_TO_INT = {'diff': 0, 'null': None, 'same': 1}

CODE_TO_NICE = {'diff': 'Different', 'null': 'NULL', 'same': 'Same'}

DIFF = 0

INT_TO_CODE = OrderedDict([(None, 'null'), (0, 'diff'), (1, 'same')])

INT_TO_NICE = OrderedDict([(None, 'NULL'), (0, 'Different'), (1, 'Same')])

class NICE

Bases: object

DIFF = 'Different'

NULL = 'NULL'

SAME = 'Same'

NICE_TO_CODE = {'Different': 'diff', 'NULL': 'null', 'Same': 'same'}

```
    NICE_TO_INT = {'Different':  0, 'NULL': None, 'Same':  1}
    NULL = None
    SAME = 1
class wbia.constants.PATH_NAMES
    Bases: object
    Path names for internal IBEIS database
    backups = '_ibeis_backups'
    bigcache = 'qres_bigcache_new'
    cache = '_ibeis_cache'
    chips = 'chips'
    detectimg = 'detectimg'
    distinctdir = 'distinctiveness_model'
    figures = 'figures'
    flann = 'flann'
    images = 'images'
    logs = '_ibeis_logs'
    nets = 'nets'
    qres = 'qres_new'
    scorenormdir = 'scorenorm'
    smartpatrol = 'smart_patrol'
    sqldb = '_ibeis_database.sqlite3'
    sqlstaging = '_ibeis_staging.sqlite3'
    thumbs = 'thumbs'
    trashdir = 'trashed_images'
    trees = 'trees'
    uploads = 'uploads'
class wbia.constants.QUAL
    Bases: object
    class CODE
        Bases: object
        EXCELLENT = 'excellent'
        GOOD = 'good'
        JUNK = 'junk'
        OK = 'ok'
        POOR = 'poor'
        UNKNOWN = 'unspecified'
    CODE_TO_INT = {'excellent':  5, 'good':  4, 'junk':  1, 'ok':  3, 'poor':  2, 'unspeci
```

```
    CODE_TO_NICE = {'excellent':  'Excellent', 'good':  'Good', 'junk':  'Junk', 'ok':  'O
    EXCELLENT = 5
    GOOD = 4
    INT_TO_CODE = OrderedDict([(5, 'excellent'), (4, 'good'), (3, 'ok'), (2, 'poor'), (1,
    INT_TO_NICE = OrderedDict([(5, 'Excellent'), (4, 'Good'), (3, 'OK'), (2, 'Poor'), (1,
    JUNK = 1
    class NICE
        Bases: object
        EXCELLENT = 'Excellent'
        GOOD = 'Good'
        JUNK = 'Junk'
        OK = 'OK'
        POOR = 'Poor'
        UNKNOWN = 'Unspecified'
    NICE_TO_CODE = {'Excellent':  'excellent', 'Good':  'good', 'Junk':  'junk', 'OK': 'ok
    NICE_TO_INT = {'Excellent':  5, 'Good':  4, 'Junk':  1, 'OK': 3, 'Poor':  2, 'Unspecif
    OK = 3
    POOR = 2
    UNKNOWN = None
class wbia.constants.REL_PATHS
    Bases: object
    all paths are relative to ibs.dbdir
    backups = '_ibsdb/_ibeis_backups'
    bigcache = '_ibsdb/_ibeis_cache/qres_bigcache_new'
    cache = '_ibsdb/_ibeis_cache'
    chips = '_ibsdb/_ibeis_cache/chips'
    distinctdir = '_ibsdb/_ibeis_cache/distinctiveness_model'
    figures = '_ibsdb/figures'
    flann = '_ibsdb/_ibeis_cache/flann'
    images = '_ibsdb/images'
    logs = '_ibsdb/_ibeis_logs'
    nets = '_ibsdb/nets'
    qres = '_ibsdb/_ibeis_cache/qres_new'
    thumbs = '_ibsdb/_ibeis_cache/thumbs'
    trashdir = 'trashed_images'
    trees = '_ibsdb/trees'
    uploads = '_ibsdb/uploads'
```

class wbia.constants.**TEST_SPECIES**
Bases: object

BEAR_POLAR = 'bear_polar'

GIR_MASAI = 'giraffe_masai'

ZEB_GREVY = 'zebra_grevys'

ZEB_PLAIN = 'zebra_plains'

class wbia.constants.**VIEW**
Bases: object

categorical viewpoint using the faces of a Rhombicuboctahedron

References

https://en.wikipedia.org/wiki/Rhombicuboctahedron

B = 7

BL = 6

BR = 8

class CODE
Bases: object

B = 'back'

BL = 'backleft'

BR = 'backright'

D = 'down'

DB = 'downback'

DBL = 'downbackleft'

DBR = 'downbackright'

DF = 'downfront'

DFL = 'downfrontleft'

DFR = 'downfrontright'

DL = 'downleft'

DR = 'downright'

F = 'front'

FL = 'frontleft'

FR = 'frontright'

L = 'left'

R = 'right'

U = 'up'

UB = 'upback'

UBL = 'upbackleft'

```
        UBR = 'upbackright'
        UF = 'upfront'
        UFL = 'upfrontleft'
        UFR = 'upfrontright'
        UL = 'upleft'
        UNKNOWN = 'unknown'
        UR = 'upright'
    CODE_TO_INT = {'back':  7, 'backleft':  6, 'backright':  8, 'down':  18, 'downback':
    CODE_TO_NICE = {'back':  'Back', 'backleft':  'Back-Left', 'backright':  'Back-Right',
    D = 18
    DB = 20
    DBL = 25
    DBR = 26
    DF = 19
    DFL = 23
    DFR = 24
    DIST = {(7, 7):  0, (6, 6):  0, (8, 8):  0, (18, 18):  0, (20, 20):  0, (25, 25):  0,
    DL = 21
    DR = 22
    F = 3
    FL = 4
    FR = 2
    INT_TO_CODE = OrderedDict([(None, 'unknown'), (1, 'right'), (2, 'frontright'), (3, 'fr
    INT_TO_NICE = OrderedDict([(None, 'Unknown'), (1, 'Right'), (2, 'Front-Right'), (3, 'F
    L = 5
    class NICE
        Bases: object
        B = 'Back'
        BL = 'Back-Left'
        BR = 'Back-Right'
        D = 'Down'
        DB = 'Down-Back'
        DBL = 'Down-Back-Left'
        DBR = 'Down-Back-Right'
        DF = 'Down-Front'
        DFL = 'Down-Front-Left'
```

```
        DFR = 'Down-Front-Right'
        DL = 'Down-Left'
        DR = 'Down-Right'
        F = 'Front'
        FL = 'Front-Left'
        FR = 'Front-Right'
        L = 'Left'
        R = 'Right'
        U = 'Up'
        UB = 'Up-Back'
        UBL = 'Up-Back-Left'
        UBR = 'Up-Back-Right'
        UF = 'Up-Front'
        UFL = 'Up-Front-Left'
        UFR = 'Up-Front-Right'
        UL = 'Up-Left'
        UNKNOWN = 'Unknown'
        UR = 'Up-Right'
    NICE_TO_CODE = {'Back':  'back', 'Back-Left':  'backleft', 'Back-Right':  'backright',
    NICE_TO_INT = {'Back':  7, 'Back-Left':  6, 'Back-Right':  8, 'Down':  18, 'Down-Back'
    R = 1
    U = 9
    UB = 11
    UBL = 16
    UBR = 17
    UF = 10
    UFL = 14
    UFR = 15
    UL = 12
    UNKNOWN = None
    UR = 13
    d = None
    f1 = None
    f2 = None
class wbia.constants.ZIPPED_URLS
    Bases: object
```

ASSIGNER = 'https://wildbookiarepository.azureedge.net/databases/testdb_assigner.zip'

DF_CURVRANK = 'https://wildbookiarepository.azureedge.net/databases/testdb_curvrank.zi

GZ_DISTINCTIVE = 'https://wildbookiarepository.azureedge.net/models/distinctivness_zeb

ID_EXAMPLE = 'https://wildbookiarepository.azureedge.net/databases/testdb_identificati

K7_EXAMPLE = 'https://wildbookiarepository.azureedge.net/databases/testdb_kaggle7.zip'

NAUTS = 'https://wildbookiarepository.azureedge.net/databases/NAUT_test.zip'

ORIENTATION = 'https://wildbookiarepository.azureedge.net/databases/testdb_orientation

PZ_DISTINCTIVE = 'https://wildbookiarepository.azureedge.net/models/distinctivness_zeb

PZ_MTEST = 'https://wildbookiarepository.azureedge.net/databases/PZ_MTEST.zip'

WDS = 'https://wildbookiarepository.azureedge.net/databases/wd_peter2.zip'

wbia.constants.**sentry_traces_sampler**(*sampling_context*)

1.23 wbia.core_annots

IBEIS CORE Defines the core dependency cache supported by the image analysis api

Extracts annotation chips from imaages and applies optional image normalizations.

Todo:

- interactive callback functions
- detection interface
- identification interface

Notes

HOW TO DESIGN INTERACTIVE PLOTS: decorate as interactive depc.get_property(recompute=True) instead of calling preproc as a generator and then adding, calls preproc and passes in a callback function. preproc spawns interaction and must call callback function when finished. callback function adds the rowids to the table.

Needed Tables: Chip NormChip Feats Keypoints Descriptors ProbChip

IdentifyQuery NeighborIndex QualityClassifier ViewpointClassifier

CommandLine: python -m wbia.control.IBEISControl –test-show_depc_annot_graph –show

Setup:

```
>>> from wbia.core_annots import *  # NOQA
>>> import wbia
>>> import wbia.plottool as pt
>>> ibs = wbia.opendb('testdb1')
>>> depc = ibs.depc_annot
>>> aid_list = ibs.get_valid_aids()[0:2]
```

class wbia.core_annots.**AnnotMaskConfig**(***kwargs*)

Bases: *wbia.dtool.base.Config*

class wbia.core_annots.**AoIConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

class wbia.core_annots.**CanonicalConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

class wbia.core_annots.**ChipConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

class wbia.core_annots.**ChipThumbConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

class wbia.core_annots.**ClassifierConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

class wbia.core_annots.**FeatConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_annots import *  # NOQA
>>> feat_cfg = FeatConfig()
>>> result = str(feat_cfg)
>>> print(result)
<FeatConfig(hesaff+sift)>
```

get_hesaff_params()

get_param_info_list()

class wbia.core_annots.**FeatWeightConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

class wbia.core_annots.**HOGConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

class wbia.core_annots.**IndexerConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_annots import *  # NOQA
>>> cfg = VsOneConfig()
>>> result = str(cfg)
>>> print(result)
```

get_flann_params()

class wbia.core_annots.**LabelerConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

class wbia.core_annots.**OrienterConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

class wbia.core_annots.**PartAssignmentFeatureConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

class wbia.core_annots.**ProbchipConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

class wbia.core_annots.**VsOneConfig**(**kwargs)
Bases: *wbia.dtool.base.Config*

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_annots import *  # NOQA
>>> cfg = VsOneConfig()
>>> result = str(cfg)
>>> print(result)
```

wbia.core_annots.**assigner_viewpoint_features**(*depc*, *part_aid_list*, *body_aid_list*, *config=None*)

wbia.core_annots.**assigner_viewpoint_unit_features**(*depc*, *part_aid_list*, *body_aid_list*, *config=None*)

wbia.core_annots.**cnn_probchips**(*ibs*, *species*, *inputchip_fpaths*, *smooth_thresh*, *smooth_ksize*)

wbia.core_annots.**compute_annotmask**(*depc*, *aid_list*, *config=None*)
Interaction dispatcher for annotation masks.

Parameters
- **depc** (*wbia.depends_cache.DependencyCache*) –
- **aid_list** (*list*) – list of annotation rowids
- **config** (AnnotMaskConfig) – (default = None)

Yields *(uri, int, int)* – tup

CommandLine: python -m wbia.core_annots –exec-compute_annotmask –show python -m wbia.core_annots –exec-compute_annotmask –show –edit

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_annots import *  # NOQA
>>> ibs, depc, aid_list = testdata_core()
>>> config = AnnotMaskConfig(dim_size=None)
>>> chip_config = config.chip_cfg
>>> edit = ut.get_argflag('--edit')
>>> mask = depc.get_property('annotmask', aid_list, 'img', config,␣
↪recompute=edit)[0]
>>> chip = depc.get_property('chips', aid_list, 'img', config=chip_config)[0]
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> resized = vt.resize_mask(mask, chip)
>>> blended = vt.blend_images_multiply(chip, resized)
>>> pt.imshow(blended, title='mask')
>>> pt.show_if_requested()
```

wbia.core_annots.**compute_aoi2**(*depc*, *aid_list*, *config=None*)
Extracts the Annotation of Interest (AoI) for a given input annotation

Parameters
- **depc** (*wbia.depends_cache.DependencyCache*) –
- **aid_list** (*list*) – list of annotation rowids

- **config** (*dict*) – (default = None)

Yields *(float, str)* – tup

CommandLine: wbia compute_aoi2

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_annot
>>> aid_list = ibs.get_valid_aids()[0:8]
>>> # depc.delete_property('aoi_two', aid_list)
>>> results = depc.get_property('aoi_two', aid_list, None)
>>> print(results)
```

wbia.core_annots.**compute_canonical**(*depc*, *aid_list*, *config=None*)

Extracts the detections for a given input annotation

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **gid_list** (*list*) – list of image rowids
- **config** (*dict*) – (default = None)

Yields *(float, str)* – tup

CommandLine: wbia compute_canonical

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_image
>>> gid_list = ibs.get_valid_gids()[0:8]
>>> # depc.delete_property('canonical', gid_list)
>>> results = depc.get_property('canonical', gid_list, None)
>>> print(results)
```

wbia.core_annots.**compute_chip**(*depc*, *aid_list*, *config=None*)

Extracts the annotation chip from the bounding box

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **aid_list** (*list*) – list of annotation rowids
- **config** (*dict*) – (default = None)

Yields *(uri, int, int)* – tup

CommandLine: python -m wbia.core_annots –exec-compute_chip:0 –show python -m wbia.core_annots –exec-compute_chip:0 –show –greyscale wbia –tf compute_chip –show –pad=64 –dim_size=256 –db PZ_MTEST wbia –tf compute_chip –show –pad=64 –dim_size=None –db PZ_MTEST wbia –tf compute_chip –show –db humpbacks wbia –tf compute_chip:1 –show

Doctest:

```
>>> from wbia.core_annots import *  # NOQA
>>> import wbia
>>> defaultdb = 'testdb1'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_annot
>>> config = ChipConfig.from_argv_dict(dim_size=None)
>>> aid_list = ibs.get_valid_aids()[0:8]
>>> chips = depc.get_property('chips', aid_list, 'img', config={'dim_size':␣
↪256})
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> #interact_obj = pt.interact_multi_image.MultiImageInteraction(chips,␣
↪nPerPage=4)
>>> import wbia.viz.interact.interact_chip
>>> interact_obj = wbia.viz.interact.interact_chip.interact_multichips(ibs,␣
↪aid_list, config2_=config)
>>> interact_obj.start()
>>> pt.show_if_requested()
```

Doctest:

```
>>> from wbia.core_annots import *  # NOQA
>>> import wbia
>>> defaultdb = 'testdb1'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_annot
>>> config = ChipConfig(**{'dim_size': (256, 256), 'resize_dim': 'wh'})
>>> #dlg = config.make_qt_dialog()
>>> #config = dlg.widget.config
>>> aid_list = ibs.get_valid_aids()[0:8]
>>> chips = depc.get_property('chips', aid_list, 'img', config=config,␣
↪recompute=True)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> pt.imshow(vt.stack_image_recurse(chips))
>>> pt.show_if_requested()
```

wbia.core_annots.**compute_chipthumb**(*depc*, *aid_list*, *config=None*)

Yet another chip thumb computer

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_annots import *  # NOQA
>>> import wbia
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_annot
>>> config = ChipThumbConfig.from_argv_dict(dim_size=None)
>>> aid_list = ibs.get_valid_aids()[0:2]
>>> compute_chipthumb(depc, aid_list, config)
>>> chips = depc.get_property('chips', aid_list, 'img', config={'dim_size': 256})
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> import wbia.viz.interact.interact_chip
>>> interact_obj = wbia.viz.interact.interact_chip.interact_multichips(ibs, aid_
↪list, config2_=config)
```

(continues on next page)

(continued from previous page)

```
>>> interact_obj.start()
>>> pt.show_if_requested()
```

wbia.core_annots.**compute_classifications**(*depc*, *aid_list*, *config=None*)

Extracts the detections for a given input annotation

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **gid_list** (*list*) – list of image rowids
- **config** (*dict*) – (default = None)

Yields *(float, str)* – tup

CommandLine: wbia compute_classifications

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_image
>>> gid_list = ibs.get_valid_gids()[0:8]
>>> # depc.delete_property('classifier', gid_list)
>>> results = depc.get_property('classifier', gid_list, None)
>>> print(results)
```

wbia.core_annots.**compute_dlen_sqrd**(*depc*, *aid_list*, *config=None*)

wbia.core_annots.**compute_feats**(*depc*, *cid_list*, *config=None*)

Computes features and yields results asynchronously: TODO: Remove IBEIS from this equation. Move the firewall towards the controller

Parameters

- **depc** (*dtool.DependencyCache*) –
- **cid_list** (*list*) –
- **config** (*None*) –

Returns generates param tups

Return type generator

SeeAlso: ~/code/wbia_cnn/wbia_cnn/_plugin.py

CommandLine: python -m wbia.core_annots –test-compute_feats:0 –show python -m wbia.core_annots –test-compute_feats:1

Doctest:

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_annots import *  # NOQA
>>> ibs, depc, aid_list = testdata_core()
>>> chip_config = {}
>>> config = FeatConfig()
>>> cid_list = depc.get_rowids('chips', aid_list, config=chip_config)
>>> featgen = compute_feats(depc, cid_list, config)
>>> feat_list = list(featgen)
>>> assert len(feat_list) == len(aid_list)
>>> (nFeat, kpts, vecs) = feat_list[0]
>>> assert nFeat == len(kpts) and nFeat == len(vecs)
>>> assert kpts.shape[1] == 6
```

(continues on next page)

(continued from previous page)

```
>>> assert vecs.shape[1] == 128
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> chip = depc.get_native('chips', cid_list[0:1], 'img')[0]
>>> pt.interact_keypoints.KeypointInteraction(chip, kpts, vecs,␣
↪autostart=True)
>>> ut.show_if_requested()
```

Example

```
>>> # DISABLE_DOCTEST
>>> # TIMING
>>> from wbia.core_annots import *  # NOQA
>>> ibs, depc, aid_list = testdata_core('PZ_MTEST', 100)
>>> #config = {'dim_size': 450}
>>> config = {}
>>> cid_list = depc.get_rowids('chips', aid_list, config=config)
>>> config = FeatConfig()
>>> featgen = compute_feats(depc, cid_list, config)
>>> feat_list = list(featgen)
>>> idx = 5
>>> (nFeat, kpts, vecs) = feat_list[idx]
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> chip = depc.get_native('chips', cid_list[idx:idx + 1], 'img')[0]
>>> pt.interact_keypoints.KeypointInteraction(chip, kpts, vecs, autostart=True)
>>> ut.show_if_requested()
```

```
>>> #num_feats = depc.get('feat', aid_list, 'num_feats', config=config,␣
↪recompute=True)
```

ibs.delete_annot_feats(aid_list) ibs.get_annot_feat_rowids(aid_list)

wbia.core_annots.**compute_fgweights**(*depc*, *fid_list*, *pcid_list*, *config=None*)

Parameters

- **depc** (*dtool.DependencyCache*) – depc
- **fid_list** (*list*) –
- **config** (*None*) – (default = None)

CommandLine: python -m wbia.core_annots compute_fgweights

Doctest:

```
>>> # xdoctest: +REQUIRES(module:wbia_cnn)
>>> from wbia.core_annots import *  # NOQA
>>> ibs, depc, aid_list = testdata_core()
>>> full_config = {}
>>> config = FeatConfig()
>>> fid_list = depc.get_rowids('feat', aid_list, config=full_config)
>>> pcid_list = depc.get_rowids('probchip', aid_list, config=full_config)
>>> prop_list = list(compute_fgweights(depc, fid_list, pcid_list))
>>> featweight_list = ut.take_column(prop_list, 0)
>>> result = np.array_str(featweight_list[0][0:3], precision=3)
>>> print(result)
```

wbia.core_annots.**compute_hog**(*depc*, *cid_list*, *config=None*)

Doctest:

```
>>> from wbia.core_annots import *  # NOQA
>>> ibs, depc, aid_list = testdata_core()
>>> chip_config = {}
>>> config = HOGConfig()
>>> cid_list = depc.get_rowids('chips', aid_list, config=chip_config)
>>> hoggen = compute_hog(depc, cid_list, config)
>>> hog = list(hoggen)[0]
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> hog_image = make_hog_block_image(hog, config)
>>> ut.show_if_requested()
```

wbia.core_annots.**compute_labels_annotations**(*depc*, *aid_list*, *config=None*)

Extracts the detections for a given input image

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **gid_list** (*list*) – list of image rowids
- **config** (*dict*) – (default = None)

Yields *(float, str)* – tup

CommandLine: python -m wbia.core_annots –exec-compute_labels_annotations python -m wbia.core_annots –exec-compute_labels_annotations:0 python -m wbia.core_annots –exec-compute_labels_annotations:1

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_annot
>>> aid_list = ibs.get_valid_aids()[0:8]
>>> config = {'labeler_algo': 'densenet', 'labeler_weight_filepath': 'giraffe_v1'}
>>> # depc.delete_property('labeler', aid_list)
>>> results = depc.get_property('labeler', aid_list, None, config=config)
>>> print(results)
>>> config = {'labeler_weight_filepath': 'candidacy'}
>>> # depc.delete_property('labeler', aid_list)
>>> results = depc.get_property('labeler', aid_list, None, config=config)
>>> print(results)
>>> config = {'labeler_algo': 'azure'}
>>> # depc.delete_property('labeler', aid_list)
>>> results = depc.get_property('labeler', aid_list, None, config=config)
>>> print(results)
>>> # depc.delete_property('labeler', aid_list)
>>> results = depc.get_property('labeler', aid_list, None)
>>> print(results)
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
```

(continues on next page)

(continued from previous page)

```
>>> import wbia
>>> defaultdb = 'WD_Master'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_annot
>>> aid_list = ibs.get_valid_aids()[0:8]
>>> config = {'labeler_algo': 'densenet', 'labeler_weight_filepath': 'wilddog_
↪v3+wilddog_v2+wilddog_v1'}
>>> # depc.delete_property('labeler', aid_list)
>>> results = depc.get_property('labeler', aid_list, None, config=config)
>>> print(results)
```

wbia.core_annots.**compute_neighbor_index**(*depc*, *fids_list*, *config*)

Parameters

- **depc** (*dtool.DependencyCache*) –
- **fids_list** (*list*) –
- **config** (*dtool.Config*) –

CommandLine: python -m wbia.core_annots –exec-compute_neighbor_index –show python -m wbia.control.IBEISControl –test-show_depc_annot_table_input –show –tablename=neighbor_index

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_annots import *  # NOQA
>>> import wbia
>>> ibs, aid_list = wbia.testdata_aids('testdb1')
>>> depc = ibs.depc_annot
>>> fid_list = depc.get_rowids('feat', aid_list)
>>> aids_list = tuple([aid_list])
>>> fids_list = tuple([fid_list])
>>> # Compute directly from function
>>> config = ibs.depc_annot['neighbor_index'].configclass()
>>> result1 = list(compute_neighbor_index(depc, fids_list, config))
>>> nnindexer1 = result1[0][0]
>>> # Compute using depcache
>>> result2 = ibs.depc_annot.get('neighbor_index', [aids_list], 'indexer', config,
↪ recompute=False, _debug=True)
>>> #result3 = ibs.depc_annot.get('neighbor_index', [tuple(fids_list)], 'indexer',
↪ config, recompute=False)
>>> print(result2)
>>> print(result3)
>>> assert result2[0] is not result3[0]
>>> assert nnindexer1.knn(ibs.get_annot_vecs(1), 1) is not None
>>> assert result3[0].knn(ibs.get_annot_vecs(1), 1) is not None
```

wbia.core_annots.**compute_orients_annotations**(*depc*, *aid_list*, *config=None*)

Extracts the detections for a given input image

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **gid_list** (*list*) – list of image rowids
- **config** (*dict*) – (default = None)

Yields *(float, str)* – tup

CommandLine: pytest wbia/core_annots.py::compute_orients_annotations:0 python -m xdoctest /Users/jason.parham/code/wildbook-ia/wbia/core_annots.py compute_orients_annotations:1 –orient

Doctest:

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'testdb_identification'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_annot
>>> aid_list = ibs.get_valid_aids()[-16:-8]
>>> config = {'orienter_algo': 'deepsense'}
>>> # depc.delete_property('orienter', aid_list)
>>> result_list = depc.get_property('orienter', aid_list, None,␣
↪config=config)
>>> xtl_list    = list(map(int, map(np.around, ut.take_column(result_list,␣
↪0))))
>>> ytl_list    = list(map(int, map(np.around, ut.take_column(result_list,␣
↪1))))
>>> w_list      = list(map(int, map(np.around, ut.take_column(result_list,␣
↪2))))
>>> h_list      = list(map(int, map(np.around, ut.take_column(result_list,␣
↪3))))
>>> theta_list  = ut.take_column(result_list, 4)
>>> bbox_list   = list(zip(xtl_list, ytl_list, w_list, h_list))
>>> ibs.set_annot_bboxes(aid_list, bbox_list, theta_list=theta_list)
>>> print(result_list)
```

Doctest:

```
>>> # DISABLE_DOCTEST
>>> import wbia
>>> import random
>>> import utool as ut
>>> from wbia.init import sysres
>>> import numpy as np
>>> dbdir = sysres.ensure_testdb_orientation()
>>> ibs = wbia.opendb(dbdir=dbdir)
>>> aid_list = ibs.get_valid_aids()
>>> note_list = ibs.get_annot_notes(aid_list)
>>> species_list = ibs.get_annot_species(aid_list)
>>> flag_list = [
>>>     note == 'random-01' and species == 'right_whale_head'
>>>     for note, species in zip(note_list, species_list)
>>> ]
>>> aid_list = ut.compress(aid_list, flag_list)
>>> aid_list = aid_list[:10]
>>> depc = ibs.depc_annot
>>> config = {'orienter_algo': 'plugin:orientation'}
>>> # depc.delete_property('orienter', aid_list)
>>> result_list = depc.get_property('orienter', aid_list, None,␣
↪config=config)
>>> xtl_list    = list(map(int, map(np.around, ut.take_column(result_list,␣
↪0))))
>>> ytl_list    = list(map(int, map(np.around, ut.take_column(result_list,␣
↪1))))
>>> w_list      = list(map(int, map(np.around, ut.take_column(result_list,␣
↪2))))
>>> h_list      = list(map(int, map(np.around, ut.take_column(result_list,␣
↪3))))
>>> theta_list  = ut.take_column(result_list, 4)
```

(continues on next page)

(continued from previous page)

```
>>> bbox_list   = list(zip(xtl_list, ytl_list, w_list, h_list))
>>> # ibs.set_annot_bboxes(aid_list, bbox_list, theta_list=theta_list)
>>> print(result_list)
```

wbia.core_annots.**compute_pairwise_vsone**(*depc*, *qaids*, *daids*, *config*)

Executes one-vs-one matching between pairs of annotations using the vt.PairwiseMatch object.

Doctest:

```
>>> from wbia.core_annots import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('PZ_MTEST')
>>> match_config = ut.hashdict({})
>>> qaids = [1, 4, 2]
>>> daids = [2, 5, 3]
>>> match_list = ibs.depc.get('pairwise_match', (qaids, daids),
>>>                           'match', config=match_config)
>>> m1, m2, m3 = match_list
>>> assert (m1.annot1['aid'], m1.annot2['aid']) == (1, 2)
>>> assert (m2.annot1['aid'], m2.annot2['aid']) == (4, 5)
>>> assert m1.fs.sum() > m2.fs.sum()
```

wbia.core_annots.**compute_probchip**(*depc*, *aid_list*, *config=None*)

Computes probability chips

CommandLine: python -m wbia.core_annots –test-compute_probchip –nocnn –show –db PZ_MTEST python -m wbia.core_annots –test-compute_probchip –show –fw_detector=cnn python -m wbia.core_annots –test-compute_probchip –show –fw_detector=rf –smooth_thresh=None

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_annots import *  # NOQA
>>> import wbia
>>> ibs, depc, aid_list = testdata_core()
>>> aid_list = ibs.get_valid_aids(species='zebra_plains')[0:10]
>>> config = ProbchipConfig.from_argv_dict(fw_detector='cnn', smooth_thresh=None)
>>> #probchip_fpath_list_ = ut.take_column(list(compute_probchip(depc, aid_list,␣
↪config)), 0)
>>> probchip_list_ = ut.take_column(list(compute_probchip(depc, aid_list,␣
↪config)), 0)
>>> #result = ut.repr2(probchip_fpath_list_)
>>> #print(result)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> #xlabel_list = list(map(str, [vt.image.open_image_size(p) for p in probchip_
↪fpath_list_]))
>>> #iteract_obj = pt.interact_multi_image.MultiImageInteraction(probchip_fpath_
↪list_, nPerPage=4, xlabel_list=xlabel_list)
>>> xlabel_list = [str(vt.get_size(img)) for img in probchip_list_]
>>> iteract_obj = pt.interact_multi_image.MultiImageInteraction(probchip_list_,␣
↪nPerPage=4, xlabel_list=xlabel_list)
>>> iteract_obj.start()
>>> ut.show_if_requested()
```

wbia.core_annots.**empty_probchips**(*inputchip_fpaths*)

wbia.core_annots.**gen_chip_configure_and_compute**(*ibs*, *gid_list*, *rowid_list*, *bbox_list*, *theta_list*, *config*)

wbia.core_annots.**gen_chip_worker**(*gpath*, *orient*, *M*, *new_size*, *filter_list*, *warpkw*)

wbia.core_annots.**gen_feat_worker**(*chip_fpath*, *probchip_fpath*, *hesaff_params*)

Function to be parallelized by multiprocessing / joblib / whatever. Must take in one argument to be used by multiprocessing.map_async

Parameters

- **chip_fpath** –
- **probchip_fpath** –
- **hesaff_params** –

Returns (None, kpts, vecs)

Return type tuple

CommandLine: python -m wbia.core_annots –exec-gen_feat_worker –show python -m wbia.core_annots –exec-gen_feat_worker –show –aid 1988 –db GZ_Master1 –affine-invariance=False –scale_max=30 python -m wbia.core_annots –exec-gen_feat_worker –show –aid 1988 –db GZ_Master1 –affine-invariance=False –maskmethod=None –scale_max=30

Doctest:

```
>>> from wbia.core_annots import *  # NOQA
>>> ibs, depc, aid_list = testdata_core()
>>> aid = aid_list[0]
>>> config = {}
>>> feat_config = FeatConfig.from_argv_dict()
>>> chip_fpath = ibs.depc_annot.get('chips', aid_list[0], 'img',␣
↪config=config, read_extern=False)
>>> maskmethod = ut.get_argval('--maskmethod', type_=str, default='cnn')
>>> probchip_fpath = ibs.depc_annot.get('probchip', aid_list[0], 'img',␣
↪config=config, read_extern=False) if feat_config['maskmethod'] == 'cnn'␣
↪else None
>>> hesaff_params = feat_config.asdict()
>>> # Exec function source
>>> masked_chip, num_kpts, kpts, vecs = ut.exec_func_src(
>>>     gen_feat_worker, key_list=['masked_chip', 'num_kpts', 'kpts', 'vecs
↪'],
>>>     sentinal='num_kpts = kpts.shape[0]')
>>> result = ('(num_kpts, kpts, vecs) = %s' % (ut.repr2((num_kpts, kpts,␣
↪vecs)),))
>>> print(result)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> from wbia.plottool.interactions import ExpandableInteraction
>>> interact = ExpandableInteraction()
>>> interact.append_plot(pt.interact_keypoints.KeypointInteraction(masked_
↪chip, kpts, vecs))
>>> interact.append_plot(lambda **kwargs: pt.plot_score_histograms([vt.get_
↪scales(kpts)], **kwargs))
>>> interact.start()
>>> ut.show_if_requested()
```

wbia.core_annots.**gen_featweight_worker**(*kpts*, *probchip*, *chipsize*)

Function to be parallelized by multiprocessing / joblib / whatever. Must take in one argument to be used by multiprocessing.map_async

Parameters

- **kpts** –
- **probchip** –

- **chipsize** –

CommandLine: python -m wbia.core_annots –test-gen_featweight_worker –show python -m wbia.core_annots –test-gen_featweight_worker –show –dpath figures –save ~/latex/crall-candidacy-2015/figures/gen_featweight.jpg python -m wbia.core_annots –test-gen_featweight_worker –show –db PZ_MTEST –qaid_list=1,2,3,4,5,6,7,8,9

Doctest:

```
>>> # xdoctest: +REQUIRES(module:wbia_cnn)
>>> from wbia.core_annots import *  # NOQA
>>> #test_featweight_worker()
>>> ibs, depc, aid_list = testdata_core()
>>> aid_list = aid_list[0:1]
>>> config = {'dim_size': 450, 'resize_dim': 'area', 'smooth_thresh': 0,
↪'smooth_ksize': 0}
>>> probchip = depc.get('probchip', aid_list, 'img', config=config)[0]
>>> chipsize = depc.get('chips', aid_list, ('width', 'height'),␣
↪config=config)[0]
>>> kpts = depc.get('feat', aid_list, 'kpts', config=config)[0]
>>> weights = gen_featweight_worker(kpts, probchip, chipsize)
>>> assert np.all(weights <= 1.0), 'weights cannot be greater than 1'
>>> chip = depc.get('chips', aid_list, 'img', config=config)[0]
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> fnum = 1
>>> pnum_ = pt.make_pnum_nextgen(1, 3)
>>> pt.figure(fnum=fnum, doclf=True)
>>> pt.imshow(chip, pnum=pnum_(0), fnum=fnum)
>>> pt.imshow(probchip, pnum=pnum_(2), fnum=fnum)
>>> pt.imshow(chip, pnum=pnum_(1), fnum=fnum)
>>> color_list = pt.draw_kpts2(kpts, weights=weights, ell_alpha=.3)
>>> cb = pt.colorbar(weights, color_list)
>>> cb.set_label('featweights')
>>> pt.show_if_requested()
```

wbia.core_annots.**get_annot_lrudfb_bools**(*ibs*, *aid_list*)

wbia.core_annots.**get_annot_lrudfb_unit_vector**(*ibs*, *aid_list*)

wbia.core_annots.**make_configured_annots**(*ibs*, *qaids*, *daids*, *qannot_cfg*, *dannot_cfg*, *preload=False*, *return_view_cache=False*)

Configures annotations so they can be sent to the vsone vt.matching procedure.

CommandLine: python -m wbia.core_annots make_configured_annots

Doctest:

```
>>> from wbia.core_annots import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb('testdb1')
>>> qannot_cfg = dannot_cfg = ut.hashdict({})
>>> qaids = [1, 2]
>>> daids = [3, 4]
>>> preload = True
>>> configured_lazy_annots, configured_annot_views = make_configured_annots(
>>>     ibs, qaids, daids, qannot_cfg, dannot_cfg, preload=False,
>>>     return_view_cache=True,
>>> )
>>> aid_dict = configured_lazy_annots[qannot_cfg]
>>> annot_views = configured_annot_views[qannot_cfg]
>>> annot = aid_dict[1]
```

(continues on next page)

(continued from previous page)

```
>>> assert len(annot_views._cache) == 0
>>> view = annot['view']
>>> kpts = annot['kpts']
>>> assert len(annot_views._cache) == 2
```

wbia.core_annots.**make_hog_block_image**(*hog*, *config=None*)

References

https://github.com/scikit-image/scikit-image/blob/master/skimage/feature/_hog.py

wbia.core_annots.**postprocess_mask**(*mask*, *thresh=20*, *kernel_size=20*)

Parameters **mask** (*ndarray*) –
Returns mask2
Return type ndarray

CommandLine: python -m wbia.core_annots –exec-postprocess_mask –cnn –show –aid=1 –db PZ_MTEST python -m wbia –tf postprocess_mask –cnn –show –db PZ_MTEST –adapteq=True
SeeAlso: python -m wbia_cnn –tf generate_species_background_mask –show –db PZ_Master1 –aid 9970
Ignore: input_tuple = aid_list tablename = 'probchip' config = full_config rowid_kw = dict(config=config)
Doctest:

```
>>> # xdoctest: +REQUIRES(module:wbia_cnn, --slow)
>>> from wbia.core_annots import *  # NOQA
>>> import wbia.plottool as pt
>>> ibs, depc, aid_list = testdata_core()
>>> config = ChipConfig.from_argv_dict()
>>> probchip_config = ProbchipConfig(smooth_thresh=None)
>>> chip = ibs.depc_annot.get('chips', aid_list, 'img', config)[0]
>>> mask = ibs.depc_annot.get('probchip', aid_list, 'img', probchip_
↪config)[0]
>>> mask2 = postprocess_mask(mask)
>>> ut.quit_if_noshow()
>>> fnum = 1
>>> pt.imshow(chip, pnum=(1, 3, 1), fnum=fnum, xlabel=str(chip.shape))
>>> pt.imshow(mask, pnum=(1, 3, 2), fnum=fnum, title='before',␣
↪xlabel=str(mask.shape))
>>> pt.imshow(mask2, pnum=(1, 3, 3), fnum=fnum, title='after',␣
↪xlabel=str(mask2.shape))
>>> ut.show_if_requested()
```

wbia.core_annots.**rf_probchips**(*ibs*, *aids*, *species*, *inputchip_fpaths*, *pad*, *smooth_thresh*, *smooth_ksize*)

wbia.core_annots.**testdata_core**(*defaultdb='testdb1'*, *size=2*)

1.24 wbia.core_images

IBEIS CORE IMAGE.

Defines the core dependency cache supported by the image analysis api

Extracts detection results from images and applies additional processing automatically

Ex python -m wbia.control.IBEISControl –test-show_depc_image_graph –show python -m wbia.control.IBEISControl –test-show_depc_image_graph –show –reduced

TODO:

Notes

HOW TO DESIGN INTERACTIVE PLOTS: decorate as interactive

depc.get_property(recompute=True)

instead of calling preproc as a generator and then adding, calls preproc and passes in a callback function. preproc spawns interaction and must call callback function when finished.

callback function adds the rowids to the table.

Needed Tables: Detections QualityClassifier ViewpointClassifier

class wbia.core_images.**AoIConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*

class wbia.core_images.**CameraTrapEXIFConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*

class wbia.core_images.**Chip2Config**(***kwargs*)
Bases: *wbia.dtool.base.Config*

class wbia.core_images.**Classifier2Config**(***kwargs*)
Bases: *wbia.dtool.base.Config*

class wbia.core_images.**ClassifierConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*

class wbia.core_images.**ClassifierLocalizationsConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*

class wbia.core_images.**DetectorConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*

class wbia.core_images.**Feature2Config**(***kwargs*)
Bases: *wbia.dtool.base.Config*

class wbia.core_images.**FeatureConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*

class wbia.core_images.**LabelerConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*

class wbia.core_images.**LocalizerConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*

class wbia.core_images.**LocalizerOriginalConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*

class wbia.core_images.**ThumbnailConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*

class wbia.core_images.**WebSrcConfig**(***kwargs*)
Bases: *wbia.dtool.base.Config*

wbia.core_images.**compute_cameratrap_exif**(*depc*, *gid_list*, *config=None*)

wbia.core_images.**compute_cameratrap_exif_worker**(*gpath*, *orient*, *bottom=80*, *psm=7*, *oem=1*, *whitelist='0123456789°CF/:'*)

wbia.core_images.**compute_classifications**(*depc*, *gid_list*, *config=None*)

Extract the detections for a given input image.

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **gid_list** (*list*) – list of image rowids
- **config** (*dict*) – (default = None)

Yields *(float, str)* – tup

CommandLine: wbia compute_classifications

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_image
>>> gid_list = ibs.get_valid_gids()[0:8]
>>> # depc.delete_property('classifier', gid_list)
>>> results = depc.get_property('classifier', gid_list, None)
>>> print(results)
>>> depc = ibs.depc_image
>>> config = {'classifier_algo': 'svm'}
>>> depc.delete_property('classifier', gid_list, config=config)
>>> results = depc.get_property('classifier', gid_list, None, config=config)
>>> print(results)
>>> config = {'classifier_algo': 'svm', 'classifier_weight_filepath': 'localizer-
↪zebra-10'}
>>> depc.delete_property('classifier', gid_list, config=config)
>>> results = depc.get_property('classifier', gid_list, None, config=config)
>>> print(results)
```

wbia.core_images.**compute_classifications2**(*depc*, *gid_list*, *config=None*)

Extract the multi-class classifications for a given input image.

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **gid_list** (*list*) – list of image rowids
- **config** (*dict*) – (default = None)

Yields *(np.ndarray, np.ndarray)* – tup

CommandLine: wbia compute_classifications2

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_image
>>> gid_list = ibs.get_valid_gids()[0:8]
>>> # depc.delete_property('classifier_two', gid_list)
>>> results = depc.get_property('classifier_two', gid_list, None)
>>> print(results)
```

wbia.core_images.**compute_detections**(*depc*, *gid_list*, *config=None*)

Extract the detections for a given input image.

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **gid_list** (*list*) – list of image rowids
- **config** (*dict*) – (default = None)

Yields *(float, np.ndarray, np.ndarray, np.ndarray, np.ndarray)* – tup

CommandLine: wbia compute_detections

Example

```
>>> # SLOW_DOCTEST
>>> # xdoctest: +SKIP
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> # dbdir = '/Users/bluemellophone/Desktop/GGR-IBEIS-TEST/'
>>> # dbdir = '/media/danger/GGR/GGR-IBEIS-TEST/'
>>> # ibs = wbia.opendb(dbdir=dbdir)
>>> depc = ibs.depc_image
>>> gid_list = ibs.get_valid_gids()[0:2]
>>> depc.delete_property('detections', gid_list)
>>> detects = depc.get_property('detections', gid_list, None)
>>> print(detects)
```

wbia.core_images.**compute_features**(*depc*, *gid_list*, *config=None*)

Compute features on images using pre-trained state-of-the-art models in Keras.

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **gid_list** (*list*) – list of image rowids
- **config** (*dict*) – (default = None)

Yields *(np.ndarray,)* – tup

CommandLine: wbia compute_features
CommandLine: python -m wbia.core_images compute_features –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_image
>>> print(depc.get_tablenames())
>>> gid_list = ibs.get_valid_gids()[:16]
>>> config = {'model': 'vgg16'}
>>> depc.delete_property('features', gid_list, config=config)
>>> features = depc.get_property('features', gid_list, 'vector', config=config)
>>> print(features)
>>> config = {'model': 'vgg19'}
>>> depc.delete_property('features', gid_list, config=config)
>>> features = depc.get_property('features', gid_list, 'vector', config=config)
```

(continues on next page)

(continued from previous page)

```
>>> print(features)
>>> config = {'model': 'resnet'}
>>> depc.delete_property('features', gid_list, config=config)
>>> features = depc.get_property('features', gid_list, 'vector', config=config)
>>> print(features)
>>> config = {'model': 'inception'}
>>> depc.delete_property('features', gid_list, config=config)
>>> features = depc.get_property('features', gid_list, 'vector', config=config)
>>> print(features)
```

wbia.core_images.**compute_localizations**(*depc*, *loc_orig_id_list*, *config=None*)

Extract the localizations for a given input image.

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **gid_list** (*list*) – list of image rowids
- **config** (*dict*) – (default = None)

Yields *(float, np.ndarray, np.ndarray, np.ndarray, np.ndarray)* – tup

CommandLine: wbia compute_localizations

CommandLine: python -m wbia.core_images compute_localizations –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_image
>>> print(depc.get_tablenames())
>>> gid_list = ibs.get_valid_gids()[:16]
>>> config = {'algo': 'lightnet', 'nms': True}
>>> # depc.delete_property('localizations', gid_list, config=config)
>>> detects = depc.get_property('localizations', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'combined': True}
>>> # depc.delete_property('localizations', gid_list, config=config)
>>> detects = depc.get_property('localizations', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
```

wbia.core_images.**compute_localizations_chips**(*depc*, *loc_id_list*, *config=None*)

Extract the detections for a given input image.

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **loc_id_list** (*list*) – list of localization rowids
- **config** (*dict*) – (default = None)

Yields *(float, str)* – tup

CommandLine: wbia compute_localizations_chips

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_image
>>> gid_list = ibs.get_valid_gids()[0:8]
>>> config = {'combined': True, 'localization_chip_masking': True}
>>> # depc.delete_property('localizations_chips', gid_list, config=config)
>>> results = depc.get_property('localizations_chips', gid_list, None,␣
↪config=config)
>>> print(results)
>>> config = {'combined': True, 'localization_chip_masking': False}
>>> # depc.delete_property('localizations_chips', gid_list, config=config)
>>> results = depc.get_property('localizations_chips', gid_list, None,␣
↪config=config)
>>> print(results)
```

wbia.core_images.**compute_localizations_classifications**(*depc*, *loc_id_list*, *config=None*)

Extract the detections for a given input image.

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **loc_id_list** (*list*) – list of localization rowids
- **config** (*dict*) – (default = None)

Yields *(float, str)* – tup

CommandLine: wbia compute_localizations_classifications

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_image
>>> gid_list = ibs.get_valid_gids()[0:8]
>>> config = {'algo': 'yolo'}
>>> # depc.delete_property('localizations_classifier', gid_list, config=config)
>>> results = depc.get_property('localizations_classifier', gid_list, None,␣
↪config=config)
>>> print(results)
>>> config = {'algo': 'yolo', 'classifier_masking': True}
>>> # depc.delete_property('localizations_classifier', gid_list, config=config)
>>> results = depc.get_property('localizations_classifier', gid_list, None,␣
↪config=config)
>>> print(results)
>>>
>>> depc = ibs.depc_image
>>> gid_list = list(set(ibs.get_imageset_gids(ibs.get_imageset_imgsetids_from_
↪text('TEST_SET'))))
>>> config = {'combined': True, 'classifier_algo': 'svm', 'classifier_weight_
↪filepath': None}
```

(continues on next page)

(continued from previous page)

```
>>> # depc.delete_property('localizations_classifier', gid_list, config=config)
>>> results = depc.get_property('localizations_classifier', gid_list, None,␣
↪config=config)
>>> print(results)
>>>
>>> config = {'combined': True, 'classifier_algo': 'svm', 'classifier_weight_
↪filepath': 'localizer-zebra-10'}
>>> # depc.delete_property('localizations_classifier', gid_list, config=config)
>>> results = depc.get_property('localizations_classifier', gid_list, None,␣
↪config=config)
>>> print(results)
>>>
>>> config = {'combined': True, 'classifier_algo': 'svm', 'classifier_weight_
↪filepath': 'localizer-zebra-50'}
>>> results = depc.get_property('localizations_classifier', gid_list, None,␣
↪config=config)
>>> print(results)
>>>
>>> config = {'combined': True, 'classifier_algo': 'svm', 'classifier_weight_
↪filepath': 'localizer-zebra-100'}
>>> results = depc.get_property('localizations_classifier', gid_list, None,␣
↪config=config)
>>> print(results)
```

wbia.core_images.**compute_localizations_features**(*depc*, *loc_id_list*, *config=None*)

Compute features on images using pre-trained state-of-the-art models in Keras.

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **gid_list** (*list*) – list of image rowids
- **config** (*dict*) – (default = None)

Yields *(np.ndarray,)* – tup

CommandLine: wbia compute_localizations_features

CommandLine: python -m wbia.core_images compute_localizations_features –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_image
>>> print(depc.get_tablenames())
>>> gid_list = ibs.get_valid_gids()[:16]
>>> config = {'feature2_algo': 'vgg16', 'combined': True}
>>> depc.delete_property('localizations_features', gid_list, config=config)
>>> features = depc.get_property('localizations_features', gid_list, 'vector',␣
↪config=config)
>>> print(features)
>>> config = {'feature2_algo': 'vgg19', 'combined': True}
>>> depc.delete_property('localizations_features', gid_list, config=config)
>>> features = depc.get_property('localizations_features', gid_list, 'vector',␣
↪config=config)
>>> print(features)
```

(continues on next page)

(continued from previous page)

```
>>> config = {'feature2_algo': 'resnet', 'combined': True}
>>> depc.delete_property('localizations_features', gid_list, config=config)
>>> features = depc.get_property('localizations_features', gid_list, 'vector',␣
↪config=config)
>>> print(features)
>>> config = {'feature2_algo': 'inception', 'combined': True}
>>> depc.delete_property('localizations_features', gid_list, config=config)
>>> features = depc.get_property('localizations_features', gid_list, 'vector',␣
↪config=config)
>>> print(features)
```

wbia.core_images.**compute_localizations_interest**(*depc*, *loc_id_list*, *config=None*)

Extract the detections for a given input image.

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **loc_id_list** (*list*) – list of localization rowids
- **config** (*dict*) – (default = None)

Yields *(float, str)* – tup

CommandLine: wbia compute_localizations_labels

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_image
>>> gid_list = ibs.get_valid_gids()[0:100]
>>> depc.delete_property('labeler', gid_list)
>>> results = depc.get_property('labeler', gid_list, None)
>>> results = depc.get_property('labeler', gid_list, 'species')
>>> print(results)
```

wbia.core_images.**compute_localizations_labels**(*depc*, *loc_id_list*, *config=None*)

Extract the detections for a given input image.

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **loc_id_list** (*list*) – list of localization rowids
- **config** (*dict*) – (default = None)

Yields *(float, str)* – tup

CommandLine: python -m wbia.core_images –exec-compute_localizations_labels

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_image
```

(continues on next page)

(continued from previous page)

```
>>> gid_list = ibs.get_valid_gids()[0:10]
>>> config = {'labeler_algo': 'densenet', 'labeler_weight_filepath': 'giraffe_v1'}
>>> # depc.delete_property('localizations_labeler', aid_list)
>>> results = depc.get_property('localizations_labeler', gid_list, None,␣
↪config=config)
>>> print(results)
>>> config = {'labeler_weight_filepath': 'candidacy'}
>>> # depc.delete_property('localizations_labeler', aid_list)
>>> results = depc.get_property('localizations_labeler', gid_list, None,␣
↪config=config)
>>> print(results)
```

wbia.core_images.**compute_localizations_original**(*depc*, *gid_list*, *config=None*)

Extract the localizations for a given input image.

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **gid_list** (*list*) – list of image rowids
- **config** (*dict*) – (default = None)

Yields *(float, np.ndarray, np.ndarray, np.ndarray, np.ndarray)* – tup

CommandLine: wbia compute_localizations_original

CommandLine: python -m wbia.core_images compute_localizations_original –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'PZ_MTEST'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_image
>>> print(depc.get_tablenames())
>>> gid_list = ibs.get_valid_gids()[:16]
>>> config = {'algo': 'azure', 'config_filepath': None}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'darknet', 'config_filepath': 'pretrained-v2-pascal'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'darknet', 'config_filepath': 'pretrained-v2-large-pascal'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'darknet', 'config_filepath': 'pretrained-tiny-pascal'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'darknet', 'config_filepath': 'pretrained-v2-large-coco'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
```

(continues on next page)

(continued from previous page)

```
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'darknet', 'config_filepath': 'pretrained-tiny-coco'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'yolo'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'lightnet'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'rf'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'selective-search'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'selective-search-rcnn'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'faster-rcnn', 'config_filepath': 'pretrained-vgg-pascal'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'faster-rcnn', 'config_filepath': 'pretrained-zf-pascal'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'faster-rcnn', 'config_filepath': 'pretrained-vgg-ilsvrc'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'faster-rcnn', 'config_filepath': 'pretrained-zf-ilsvrc'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'ssd', 'config_filepath': 'pretrained-300-pascal'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
```

(continues on next page)

(continued from previous page)

```
>>> print(detects)
>>> config = {'algo': 'ssd', 'config_filepath': 'pretrained-512-pascal'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'ssd', 'config_filepath': 'pretrained-300-pascal-plus'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'ssd', 'config_filepath': 'pretrained-512-pascal-plus'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'ssd', 'config_filepath': 'pretrained-300-coco'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'ssd', 'config_filepath': 'pretrained-512-coco'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'ssd', 'config_filepath': 'pretrained-300-ilsvrc'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': 'ssd', 'config_filepath': 'pretrained-500-ilsvrc'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
>>> config = {'algo': '_COMBINED'}
>>> depc.delete_property('localizations_original', gid_list, config=config)
>>> detects = depc.get_property('localizations_original', gid_list, 'bboxes',␣
↪config=config)
>>> print(detects)
```

wbia.core_images.**compute_thumbnails**(*depc*, *gid_list*, *config=None*)

Compute the thumbnail for a given input image.

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **gid_list** (*list*) – list of image rowids
- **config** (*dict*) – (default = None)

Yields *(uri, int, int)* – tup

CommandLine: wbia –tf compute_thumbnails –show –db PZ_MTEST

Example

```
>>> # ENABLE_DOCTEST
>>> # xdoctest: +REQUIRES(--weird)
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'testdb1'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_image
>>> gid_list = ibs.get_valid_gids()[0:10]
>>> thumbs = depc.get_property('thumbnails', gid_list, 'img', config={'thumbsize
↪': 221}, recompute=True)
>>> # xdoctest: +REQUIRES(--show)
>>> import wbia.plottool as pt
>>> pt.quit_if_noshow()
>>> iteract_obj = pt.interact_multi_image.MultiImageInteraction(thumbs,␣
↪nPerPage=4)
>>> iteract_obj.start()
>>> pt.show_if_requested()
```

wbia.core_images.**compute_web_src**(*depc*, *gid_list*, *config=None*)

Compute the web src

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **gid_list** (*list*) – list of image rowids
- **config** (*dict*) – (default = None)

Yields *(str)* – tup

CommandLine: wbia –tf compute_web_src –show –db PZ_MTEST

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.core_images import *  # NOQA
>>> import wbia
>>> defaultdb = 'testdb1'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_image
>>> gid_list = ibs.get_valid_gids()[0:10]
>>> thumbs = depc.get_property('web_src', gid_list, 'src', recompute=True)
>>> thumb = thumbs[0]
>>> hash_str = ut.hash_data(thumb)
>>> assert hash_str in ['yerctlgfqosrhmjpqvkbmnoocagfqsna',
↪'wcuppmpowkvhfmfcnrxdeedommihexfu', 'lerhyizhlignvvzmvqbbberyklzyfbzq'], 'Found
↪%r' % (hash_str, )
```

wbia.core_images.**draw_thumb_helper**(*thumbsize*, *gpath*, *orient*, *bbox_list*, *theta_list*, *interest_list*)

wbia.core_images.**draw_web_src**(*gpath*, *orient*)

wbia.core_images.**get_localization_aoi2**(*ibs*, *loc_id_list*, *target_size=(192, 192)*)

wbia.core_images.**get_localization_chips**(*ibs*, *loc_id_list*, *target_size=(128, 128)*, *axis_aligned=False*)

wbia.core_images.**get_localization_chips_worker**(*gid*, *img*, *bbox_list*, *theta_list*, *target_size*, *axis_aligned=False*)

wbia.core_images.**get_localization_masks**(*ibs*, *loc_id_list*, *target_size=(128, 128)*)

wbia.core_images.**get_localization_masks_worker**(*gid*, *img*, *bbox_list*, *theta_list*, *target_size*)

wbia.core_images.**load_text**(*fpath*)

wbia.core_images.**save_text**(*fpath*, *text*)

1.25 wbia.core_parts

Extracts parts chips from image and applies optional image normalizations.

wbia.core_parts.**compute_part_chip**(*depc*, *part_rowid_list*, *config=None*)

Extracts the part chip from the bounding box

Parameters

- **depc** (*wbia.depends_cache.DependencyCache*) –
- **part_rowid_list** (*list*) – list of part rowids
- **config** (*dict*) – (default = None)

Yields *(uri, int, int)* – tup

CommandLine: wbia –tf compute_part_chip

Doctest:

```
>>> from wbia.core_parts import *  # NOQA
>>> import wbia
>>> import random
>>> defaultdb = 'testdb1'
>>> ibs = wbia.opendb(defaultdb=defaultdb)
>>> depc = ibs.depc_part
>>> config = {'dim_size': None}
>>> aid_list = ibs.get_valid_aids()
>>> aid_list = aid_list[:10]
>>> bbox_list = ibs.get_annot_bboxes(aid_list)
>>> bbox_list = [
>>>     (xtl + 100, ytl + 100, w - 100, h - 100)
>>>     for xtl, ytl, w, h in bbox_list
>>> ]
>>> part_rowid_list = ibs.add_parts(aid_list, bbox_list=bbox_list)
>>> chips = depc.get_property('pchips', part_rowid_list, 'img',␣
↪config=config)
>>> for (xtl, ytl, w, h), chip in zip(bbox_list, chips):
>>>     assert chip.shape == (h, w, 3)
>>> ibs.delete_parts(part_rowid_list)
```

1.26 wbia.demodata

wbia.demodata.**ensure_demodata**()

Ensures that you have testdb1 and PZ_MTEST demo databases.

wbia.demodata.**ensure_testdata**()

wbia.demodata.**get_test_gpaths**(*ndata=None*, *names=None*, ***kwargs*)

wbia.demodata.**get_testimg_path**(*gname*)

Returns path to image in testdata

1.27 wbia.dev

mkinit ~/code/wbia/wbia

DEV SCRIPT

TODO: DEPRICATE

This is a hacky script meant to be run mostly automatically with the option of interactions.

dev.py is supposed to be a developer non-gui interface into the IBEIS software. dev.py runs experiments and serves as a scratchpad for new code and quick scripts

---

Todo: Test to find typical "good" descriptor scores. Find nearest neighbors and noramlizers for each feature in a query image. Based on ground truth and spatial verification mark feature matches as true or false. Visualize the feature scores of good matches vs bad matches. Lowe shows the pdf of correct matches and the PDF for incorrect matches. We should also show the same thing.

Done: Cache nearest neighbors so different parameters later in the pipeline dont take freaking forever.

CommandLine: python dev.py –wshow -t query –db PZ_MTEST –qaid 110 –cfg score_method:nsum prescore_method:nsum python dev.py –wshow -t query –db PZ_MTEST –qaid 110 python dev.py –wshow -t query –db PZ_MTEST –qaid 110 –cfg fg_on=True python dev.py –wshow -t query –db PZ_MTEST –qaid 110 –cfg

wbia.dev.**dev_snippets**(*main_locals*)
Common variables for convineince when interacting with IPython

wbia.dev.**devmain**()
The Developer Script A command line interface to almost everything

-w	# wait / show the gui / figures are visible
--cmd	# ipython shell to play with variables
-t	# run list of tests

wbia.dev.**get_ibslist**(*ibs*)

wbia.dev.**get_sortbystr**(*str_list*, *key_list*, *strlbl=None*, *keylbl=None*)

wbia.dev.**ggr_random_name_splits**()
CommandLine: python -m wbia.viz.viz_graph2 ggr_random_name_splits –show
Ignore: sshfs -o idmap=user lev:/ ~/lev

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.viz.viz_graph2 import *  # NOQA
>>> ggr_random_name_splits()
```

wbia.dev.**run_dev**(*ibs*)
main developer command
CommandLine: python dev.py –db PZ_Master0 –controlled –print-rankhist

wbia.dev.**run_devcmds**(*ibs*, *qaid_list*, *daid_list*, *acfg=None*)
This function runs tests passed in with the -t flag

wbia.dev.**run_devprecmds**()
Looks for pre-tests specified with the -t flag and runs them

1.28 wbia.filter_configs

1.29 wbia.images

class wbia.images.**ImageIBEISPropertyInjector**(*name*, *bases*, *dct*)
Bases: type

class wbia.images.**ImageSetAttrInjector**(*name*, *bases*, *dct*)
Bases: type

Example

```
>>> # SCRIPT
>>> from wbia import _wbia_object
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> objname = 'imageset'
>>> blacklist = []
>>> _wbia_object._find_wbia_attrs(ibs, objname, blacklist)
```

class wbia.images.**ImageSets**(*gsids*, *ibs*, *config=None*)
Bases: *wbia._wbia_object.ObjectList1D*

Represents a group of annotations. Efficiently accesses properties from a database using lazy evaluation.
CommandLine: python -m wbia.images ImageSets

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.images import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> gsids = ibs._get_all_imgsetids()
>>> self = ImageSets(gsids, ibs)
>>> print(self)
<ImageSets(num=13)>
```

aids

annots

configid

custom_filtered_aids

duration

end_time_posix

fraction_annotmatch_reviewed

fraction_imgs_reviewed

fraction_names_with_exemplar

gids

gps_lats

gps_lons

gsgrids

image_uuids

images

imgsetids_from_text

imgsetids_from_uuid

isoccurrence

name_uuids

nids

note

notes

num_aids

num_annotmatch_reviewed

num_annots_reviewed

num_gids

num_imgs_reviewed

num_names_with_exemplar

percent_annotmatch_reviewed_str

percent_imgs_reviewed_str

percent_names_with_exemplar_str

processed_flags

rrr()
Dynamic module reloading

shipped_flags

smart_waypoint_ids

smart_xml_contents

smart_xml_fnames

start_time_posix

text

uuid

uuids

class wbia.images.**Images**(*rowids*, *ibs*, *config=None*, *caching=False*, *asarray=False*)
Bases: *wbia._wbia_object.ObjectList1D*

Represents a group of annotations. Efficiently accesses properties from a database using lazy evaluation.
CommandLine: python -m wbia.images Images –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.images import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> gids = ibs.get_valid_gids()
>>> g = self = images = Images(gids, ibs)
>>> print(g.widths)
>>> print(g)
<Images(num=13)>
```

aids

aids_of_species

annot_uuids

annot_uuids_of_species

annotation_bboxes

annotation_thetas

annots

append_to_imageset(*imageset_text*)

contributor_rowid

contributor_tag

datetime

datetime_str

detect_confidence

detectpaths

enabled

exts

gid

gids

glrids

gnames

gps

gps2

gsgrids

heights

imagesettext

imgset_uuids

imgsetids

lat

location_codes

lon

missing_uuid

name_uuids

nids

notes

num_annotations

orientation

orientation_str

party_rowids

party_tag

paths

remove_from_imageset(*imageset_text*)

reviewed

rrr()
Dynamic module reloading

show(**args*, ***kwargs*)

sizes

species_rowids

species_uuids

thumbpath

thumbtup

time_statstr

timedelta_posix

unixtime

unixtime2

unixtime_asfloat

uris

uris_original

uuids

widthsimgdata

wbia.images.**images**(*ibs*, *gids=None*, *uuids=None*, ***kwargs*)
Makes an Images object

wbia.images.**imagesets**(*ibs*, *gsids=None*, *text=None*)

1.30 wbia.params

DEPRICATE THIS ENTIRE FILE

this module lists most of the command line args available for use. there are still many cases where util_arg.get_argval and util_arg.get_argflag are used instead of this module. Those command line arguments will not be represented here and they should eventually be integrated into this module (hopefully automagically)

TODO: nnkj/enerate this module automagically from

> import utool as ut utool_parse_codeblock = ut.util_arg.autogen_argparse_block(extra_args=parsed_args) ut.util_arg.reset_argrecord() import wbia parsed_args = ut.util_arg.parse_used_arg_flags_and_vals(wbia, recursive=True) wbia_parse_codeblock = ut.util_arg.autogen_argparse_block(extra_args=parsed_args)
>
> ut.util_arg.autogenerate_parse_py([utool_parse_codeblock, wbia_parse_codeblock])
>
> utool_parse_codeblock ut.util_arg
>
> print(parse_codeblock)

wbia.params.**parse_args**()

1.31 wbia.tag_funcs

wbia.tag_funcs.**append_annot_case_tags**(*ibs*, *aid_list*, *tag_list*)

> Generally appends tags to annotations. Careful not to introduce too many random tags. Maybe we should just let that happen and introduce tag-aliases
>
> Note: this is more of a set add rather than a list append
>
> TODO: remove

wbia.tag_funcs.**consolodate_annotmatch_tags**(*old_tags*)

wbia.tag_funcs.**export_tagged_chips**(*ibs*, *aid_list*, *dpath='.'*)

> DEPRICATE
>
> **CommandLine:** python -m wbia.tag_funcs –exec-export_tagged_chips –tags Hard interesting needswork –db PZ_Master1 python -m wbia.tag_funcs –exec-export_tagged_chips –logic=or –any_startswith quality occlusion –has_any lighting needswork interesting hard –db GZ_Master1 –dpath=/media/raid python -m wbia.tag_funcs –exec-export_tagged_chips –db GZ_Master1 –min_num=1 –dpath /media/raid

Example

```
>>> # SCRIPT
>>> from wbia.tag_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> kwargs = ut.argparse_dict(ut.get_kwdefaults2(filterflags_general_tags), type_
↪hint=ut.ddict(list, logic=str))
>>> ut.print_dict(kwargs, 'filter args')
>>> aid_list = ibs.filter_annots_by_tags(**kwargs)
>>> print('len(aid_list) = %r' % (len(aid_list),))
>>> dpath = ut.get_argval('--dpath', default='')
>>> all_tags = ut.flatten(ibs.get_annot_all_tags(aid_list))
>>> filtered_tag_hist = ut.dict_hist(all_tags)
>>> ut.print_dict(filtered_tag_hist, key_order_metric='val')
>>> export_tagged_chips(ibs, aid_list, dpath)
```

wbia.tag_funcs.**filter_aidpairs_by_tags**(*ibs*, *has_any=None*, *has_all=None*, *min_num=None*, *max_num=None*, *am_rowids=None*)

list(zip(aid_pairs, undirected_tags))

wbia.tag_funcs.**filter_annotmatch_by_tags**(*ibs*, *annotmatch_rowids=None*, ***kwargs*)

ignores case

Parameters

- **ibs** (IBEISController) – wbia controller object
- **flags** –

Returns list

CommandLine: python -m wbia.tag_funcs –exec-filter_annotmatch_by_tags –show python -m wbia.tag_funcs –exec-filter_annotmatch_by_tags –show –db PZ_Master1 –min-num=1 python -m wbia.tag_funcs –exec-filter_annotmatch_by_tags –show –db PZ_Master1 –tags JoinCase python -m wbia.tag_funcs –exec-filter_annotmatch_by_tags –show –db PZ_Master1 –tags SplitCase python -m wbia.tag_funcs –exec-filter_annotmatch_by_tags –show –db PZ_Master1 –tags occlusion python -m wbia.tag_funcs –exec-filter_annotmatch_by_tags –show –db PZ_Master1 –tags viewpoint python -m wbia.tag_funcs –exec-filter_annotmatch_by_tags –show –db PZ_Master1 –tags SceneryMatch python -m wbia.tag_funcs –exec-filter_annotmatch_by_tags –show –db PZ_Master1 –tags Photobomb

python -m wbia.tag_funcs –exec-filter_annotmatch_by_tags –show –db GZ_Master1 –tags needswork

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.tag_funcs import *  # NOQA
>>> import wbia
>>> #ibs = wbia.opendb(defaultdb='testdb1')
>>> ibs = wbia.opendb(defaultdb='PZ_Master1')
>>> #tags = ['Photobomb', 'SceneryMatch']
>>> has_any = ut.get_argval('--tags', type_=list, default=['SceneryMatch',
↪'Photobomb'])
>>> min_num = ut.get_argval('--min_num', type_=int, default=1)
>>> prop = has_any[0]
>>> filtered_annotmatch_rowids = filter_annotmatch_by_tags(ibs, None, has_any=has_
↪any, min_num=min_num)
>>> aid1_list = np.array(ibs.get_annotmatch_aid1(filtered_annotmatch_rowids))
>>> aid2_list = np.array(ibs.get_annotmatch_aid2(filtered_annotmatch_rowids))
>>> aid_pairs = np.vstack([aid1_list, aid2_list]).T
>>> # Dont double count
>>> xs = vt.find_best_undirected_edge_indexes(aid_pairs)
>>> aid1_list = aid1_list.take(xs)
>>> aid2_list = aid2_list.take(xs)
>>> valid_tags_list = ibs.get_annotmatch_case_tags(filtered_annotmatch_rowids)
>>> print('valid_tags_list = %s' % (ut.repr2(valid_tags_list, nl=1),))
>>> #
>>> print('Aid pairs with has_any=%s' % (has_any,))
>>> print('Aid pairs with min_num=%s' % (min_num,))
>>> print('aid_pairs = ' + ut.repr2(list(zip(aid1_list, aid2_list))))
>>> # Show timedelta info
>>> ut.quit_if_noshow()
>>> timedelta_list = ibs.get_annot_pair_timedelta(aid1_list, aid2_list)
>>> import wbia.plottool as pt
>>> pt.draw_timedelta_pie(timedelta_list, label='timestamp of tags=%r' % (has_any,
↪))
>>> ut.show_if_requested()
```

wbia.tag_funcs.**filter_annots_by_tags**(*ibs*, *aid_list=None*, ***kwargs*)

Filter / Find / Search for annotations with particular tags

CommandLine: python -m wbia.tag_funcs –exec-filter_annots_by_tags –helpx python -m wbia.tag_funcs –exec-filter_annots_by_tags –db GZ_Master1 python -m wbia.tag_funcs –exec-filter_annots_by_tags –db GZ_Master1 –min_num=1 python -m wbia.tag_funcs –exec-filter_annots_by_tags –db GZ_Master1 –has_any=lighting –has_all=lighting:underexposed –show

SeeAlso: python -m wbia.init.filter_annots –exec-filter_annots_general

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.tag_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> kwargs = ut.argparse_dict(ut.get_kwdefaults2(filterflags_general_tags), type_
↪hint=ut.ddict(list, logic=str))
>>> ut.print_dict(kwargs, 'filter args')
>>> aid_list = ibs.filter_annots_by_tags(aid_list, **kwargs)
>>> print('len(aid_list) = %r' % (len(aid_list),))
>>> # print results
>>> all_tags = ut.flatten(ibs.get_annot_all_tags(aid_list))
>>> filtered_tag_hist = ut.dict_hist(all_tags)
>>> ut.print_dict(filtered_tag_hist, key_order_metric='val')
>>> print('len(aid_list) = %r' % (len(aid_list),))
>>> print('sum(tags) = %r' % (sum(filtered_tag_hist.values()),))
>>> ut.quit_if_noshow()
>>> import wbia.viz.interact
>>> wbia.viz.interact.interact_chip.interact_multichips(ibs, aid_list)
>>> ut.show_if_requested()
```

wbia.tag_funcs.**filterflags_annot_tags**(*ibs*, *aid_list*, ***kwargs*)

Filter / Find / Search for annotations with particular tags

wbia.tag_funcs.**filterflags_general_tags**(*tags_list*, *has_any=None*, *has_all=None*, *has_none=None*, *min_num=None*, *max_num=None*, *any_startswith=None*, *any_endswith=None*, *any_match=None*, *none_match=None*, *logic='and'*)

maybe integrate into utool? Seems pretty general

Parameters

- **tags_list** (*list*) –
- **has_any** (*None*) – (default = None)
- **has_all** (*None*) – (default = None)
- **min_num** (*None*) – (default = None)
- **max_num** (*None*) – (default = None)

CommandLine: python -m wbia.tag_funcs –exec-filterflags_general_tags python -m wbia.tag_funcs –exec-filterflags_general_tags:0 –helpx python -m wbia.tag_funcs –exec-filterflags_general_tags:0 python -m wbia.tag_funcs –exec-filterflags_general_tags:0 –none_match n python -m wbia.tag_funcs –exec-filterflags_general_tags:0 –has_none=n,o python -m wbia.tag_funcs –exec-filterflags_general_tags:1 python -m wbia.tag_funcs –exec-filterflags_general_tags:2

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.tag_funcs import *  # NOQA
>>> tags_list = [['v'], [], ['P'], ['P', 'o'], ['n', 'o',], [], ['n', 'N'], ['e',
↪'i', 'p', 'b', 'n'], ['q', 'v'], ['n'], ['n'], ['N']]
>>> kwargs = ut.argparse_dict(ut.get_kwdefaults2(filterflags_general_tags), type_
↪hint=list)
>>> print('kwargs = %r' % (kwargs,))
>>> flags = filterflags_general_tags(tags_list, **kwargs)
>>> print(flags)
>>> result = ut.compress(tags_list, flags)
>>> print('result = %r' % (result,))
```

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.tag_funcs import *  # NOQA
>>> tags_list = [['v'], [], ['P'], ['P'], ['n', 'o',], [], ['n', 'N'], ['e', 'i',
↪'p', 'b', 'n'], ['n'], ['n'], ['N']]
>>> has_all = 'n'
>>> min_num = 1
>>> flags = filterflags_general_tags(tags_list, has_all=has_all, min_num=min_num)
>>> result = ut.compress(tags_list, flags)
>>> print('result = %r' % (result,))
```

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.tag_funcs import *  # NOQA
>>> tags_list = [['vn'], ['vn', 'no'], ['P'], ['P'], ['n', 'o',], [], ['n', 'N'],␣
↪['e', 'i', 'p', 'b', 'n'], ['n'], ['n', 'nP'], ['NP']]
>>> kwargs = {
>>>     'any_endswith': 'n',
>>>     'any_match': None,
>>>     'any_startswith': 'n',
>>>     'has_all': None,
>>>     'has_any': None,
>>>     'has_none': None,
>>>     'max_num': 3,
>>>     'min_num': 1,
>>>     'none_match': ['P'],
>>> }
>>> flags = filterflags_general_tags(tags_list, **kwargs)
>>> filtered = ut.compress(tags_list, flags)
>>> result = ('result = %s' % (ut.repr2(filtered),))
result = [['vn', 'no'], ['n', 'o'], ['n', 'N'], ['n'], ['n', 'nP']]
```

wbia.tag_funcs.**get_aidpair_tags**(*ibs*, *aid1_list*, *aid2_list*, *directed=True*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid1_list** (*list*) –
- **aid2_list** (*list*) –
- **directed** (*bool*) – (default = True)

Returns tags_list
Return type list

CommandLine: python -m wbia.tag_funcs –exec-get_aidpair_tags –db PZ_Master1 –tags Hard interesting

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.tag_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> has_any = ut.get_argval('--tags', type_=list, default=None)
>>> min_num = ut.get_argval('--min_num', type_=int, default=1)
>>> aid_pairs = filter_aidpairs_by_tags(ibs, has_any=has_any, min_num=1)
>>> aid1_list = aid_pairs.T[0]
>>> aid2_list = aid_pairs.T[1]
>>> undirected_tags = get_aidpair_tags(ibs, aid1_list, aid2_list, directed=False)
>>> tagged_pairs = list(zip(aid_pairs.tolist(), undirected_tags))
>>> print(ut.repr2(tagged_pairs))
>>> tag_dict = ut.groupby_tags(tagged_pairs, undirected_tags)
>>> print(ut.repr2(tag_dict, nl=2))
>>> print(ut.repr2(ut.map_dict_vals(len, tag_dict)))
```

wbia.tag_funcs.**get_annot_all_tags**(*ibs*, *aid_list=None*)

CommandLine: python -m wbia.tag_funcs –exec-get_annot_all_tags –db GZ_Master1

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.tag_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> all_tags = ut.flatten(ibs.get_annot_all_tags(aid_list))
>>> tag_hist = ut.dict_hist(all_tags)
>>> ut.print_dict(tag_hist)
```

wbia.tag_funcs.**get_annot_annotmatch_tags**(*ibs*, *aid_list*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*list*) – list of annotation rowids

Returns annotmatch_tags_list
Return type list

CommandLine: python -m wbia.tag_funcs –exec-get_annot_annotmatch_tags –db GZ_Master1

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.tag_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> all_tags = ut.flatten(get_annot_annotmatch_tags(ibs, aid_list))
```

(continues on next page)

(continued from previous page)

```
>>> tag_hist = ut.dict_hist(all_tags)
>>> ut.print_dict(tag_hist)
```

wbia.tag_funcs.**get_annot_case_tags**(*ibs*, *aid_list*)

returns list of tags. Use instead of get_annot_tag_text .. todo:: rename to get_annot_unary_tags

Parameters

- **ibs** (IBEISController) – wbia controller object
- **aid_list** (*list*) – list of annotation rowids

Returns tags_list

Return type list

CommandLine: python -m wbia.tag_funcs –exec-get_annot_case_tags

Example

```
>>> # ENABLE_DOCTEST
>>> from wbia.tag_funcs import *  # NOQA
>>> from wbia.tag_funcs import _parse_tags # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> aid_list = ibs.get_valid_aids()
>>> tags_list = get_annot_case_tags(ibs, aid_list)
>>> result = ('tags_list = %s' % (str(tags_list),))
>>> print(result)
```

Ignore: # FIXME incorrporate old tag notes aid_list = ibs.get_valid_aids() notes_list = ibs.get_annot_notes(aid_list) flags = [len(notes) > 0 for notes in notes_list] aid_list = ut.compress(aid_list, flags) notes_list = ut.compress(notes_list, flags)

import re notes_list = [note.replace('rfdetect', '') for note in notes_list] notes_list = [note.replace('<COMMA>', ';') for note in notes_list] notes_list = [note.replace('jpg', '') for note in notes_list] notes_list = [note.replace('<HARDCASE>', '') for note in notes_list] notes_list = [note.strip() for note in notes_list] notes_list = [re.sub(';;*', ';', note) for note in notes_list] notes_list = [note.strip(';') for note in notes_list] notes_list = [note.strip(':') for note in notes_list] notes_list = [note.strip() for note in notes_list]

flags = [len(notes) < 70 and len(notes) > 0 for notes in notes_list] aid_list = ut.compress(aid_list, flags) notes_list = ut.compress(notes_list, flags)

flags = ['M;' not in notes and 'F;' not in notes and 'H1' not in notes for notes in notes_list] flags = ['M;' not in notes and 'F;' not in notes and 'H1' not in notes for notes in notes_list] aid_list = ut.compress(aid_list, flags) notes_list = ut.compress(notes_list, flags)

flags = ['aliases' not in notes for notes in notes_list] aid_list = ut.compress(aid_list, flags) notes_list = ut.compress(notes_list, flags)

#flags = [not re.match(';d*;', note) for note in notes_list] flags = [not re.match(r'dd*', note) for note in notes_list] aid_list = ut.compress(aid_list, flags) notes_list = ut.compress(notes_list, flags)

flags = [not notes.startswith('Foal;') for notes in notes_list] aid_list = ut.compress(aid_list, flags) notes_list = ut.compress(notes_list, flags)

old_tags_list = [_parse_tags(note) for note in notes_list]

old_tags = list(set(ut.flatten(old_tags_list))) old_tags = sorted([tag for tag in old_tags if not re.match(r'dd*', tag)])
old_to_new = { 'gash': None, 'pose': 'novelpose', 'vocalizing': 'novelpose' 'occlusion': 'occlusion',

}

Ignore: python -m wbia.tag_funcs –exec-filter_annotmatch_by_tags –show –db PZ_Master1 –tags viewpoint

wbia.tag_funcs.**get_annot_prop**(*ibs*, *prop*, *aid_list*)

Annot tags

wbia.tag_funcs.**get_annotmatch_case_tags**(*ibs*, *annotmatch_rowids*)

Parameters

- **ibs** (IBEISController) – wbia controller object
- **annotmatch_rowids** –

Returns filtered_aid_list

Return type list

CommandLine: python -m wbia.tag_funcs –exec-get_annotmatch_case_tags

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia.tag_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='PZ_Master1')
>>> annotmatch_rowids = ibs._get_all_annotmatch_rowids()
>>> tags_list = get_annotmatch_case_tags(ibs, annotmatch_rowids)
>>> result = ('tags_list = %s' % (str(tags_list),))
>>> print(result)
tags_list = [[u'occlusion', u'pose', 'Hard', 'NonDistinct'], [], ['Hard']]
```

wbia.tag_funcs.**get_annotmatch_other_prop**(*ibs*, *prop*, *annotmatch_rowids*)

wbia.tag_funcs.**get_annotmatch_prop**(*ibs*, *prop*, *annotmatch_rowids*)

hacky getter for dynamic properties of annotmatches using notes table

Parameters

- **prop** (*str*) –
- **annotmatch_rowids** –

Returns filtered_aid_list

Return type list

CommandLine: python -m wbia.tag_funcs –exec-get_annotmatch_prop

Example

```
>>> # DISABLE_DOCTEST
>>> # Test setting and getting standard keys
>>> from wbia.tag_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> prop = 'hard'
>>> annotmatch_rowids = ibs._get_all_annotmatch_rowids()
>>> flag_list = get_annotmatch_prop(ibs, prop, annotmatch_rowids)
>>> flag_list = ('filtered_aid_list = %s' % (str(flag_list),))
>>> subset_rowids = annotmatch_rowids[::2]
>>> set_annotmatch_prop(ibs, prop, subset_rowids, [True] * len(subset_rowids))
>>> flag_list2 = get_annotmatch_prop(ibs, prop, annotmatch_rowids)
>>> print('flag_list2 = %r' % (flag_list2,))
```

Example

```
>>> # DISABLE_DOCTEST
>>> # Test setting and getting non-standard keys
>>> from wbia.tag_funcs import *  # NOQA
>>> import wbia
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> prop = 'occlusion'
>>> annotmatch_rowids = ibs._get_all_annotmatch_rowids()
>>> flag_list = get_annotmatch_prop(ibs, prop, annotmatch_rowids)
>>> flag_list = ('filtered_aid_list = %s' % (str(flag_list),))
>>> subset_rowids = annotmatch_rowids[1::2]
>>> subset_rowids1 = annotmatch_rowids[::2]
>>> set_annotmatch_prop(ibs, prop, subset_rowids1, [True] * len(subset_rowids))
>>> set_annotmatch_prop(ibs, 'pose', subset_rowids1, [True] * len(subset_rowids))
>>> flag_list2 = get_annotmatch_prop(ibs, prop, annotmatch_rowids)
>>> print('flag_list2 = %r' % (flag_list2,))
```

wbia.tag_funcs.**get_annotmatch_standard_prop**(*ibs*, *prop*, *annotmatch_rowids*)

wbia.tag_funcs.**get_available_annot_tags**()

wbia.tag_funcs.**get_cate_categories**()

wbia.tag_funcs.**get_textformat_tag_flags**(*prop*, *text_list*)
general text tag getter hack

wbia.tag_funcs.**overwrite_annot_case_tags**(*ibs*, *aid_list*, *tag_list*)
Completely replaces annotation tags. BE VERY CAREFUL WITH THIS FUNCTION

wbia.tag_funcs.**remove_all_annot_case_tags**(*ibs*, *aid_list*)

wbia.tag_funcs.**remove_annot_case_tags**(*ibs*, *aid_list*, *tag_list*)

wbia.tag_funcs.**rename_and_reduce_tags**(*ibs*, *annotmatch_rowids*)
Script to update tags to newest values
CommandLine: python -m wbia.tag_funcs –exec-rename_and_reduce_tags –db PZ_Master1
Ignore:

```
>>> from wbia.tag_funcs import *  # NOQA
>>> import wbia
>>> #ibs = wbia.opendb(defaultdb='PZ_Master1')
>>> ibs = wbia.opendb(defaultdb='testdb1')
>>> annotmatch_rowids = filter_annotmatch_by_tags(ibs, min_num=1)
>>> rename_and_reduce_tags(ibs, annotmatch_rowids)
```

wbia.tag_funcs.**set_annot_case_tags**(*ibs*, *aid_list*, *new_tags_list*)
Completely overwrite case tags

wbia.tag_funcs.**set_annot_prop**(*ibs*, *prop*, *aid_list*, *flags*)
sets nonstandard properties using the notes column

wbia.tag_funcs.**set_annotmatch_other_prop**(*ibs*, *prop*, *annotmatch_rowids*, *flags*)
sets nonstandard properties using the notes column

wbia.tag_funcs.**set_annotmatch_prop**(*ibs*, *prop*, *annotmatch_rowids*, *flags*)
hacky setter for dynamic properties of annotmatches using notes table

wbia.tag_funcs.**set_textformat_tag_flags**(*prop*, *text_list*, *flags*)
general text tag setter hack

1.32 Module contents

IBEIS: main package init

TODO: LAZY IMPORTS? http://code.activestate.com/recipes/473888-lazy-module-imports/

wbia.**import_subs**()

wbia.**reload_subs**(*verbose=True*)
Reloads wbia and submodules

wbia.**rrrr**(*verbose=True*)
Regen Command: Kinda have to work with the output of these. This module is hard to autogenerate correctly.

cd /home/joncrall/code/wbia/wbia/other makeinit.py -x web viz tests gui makeinit.py -x constants params entry_points other control dbio tests

wbia.**run_experiment**(*e='print', db='PZ_MTEST', dbdir=None, a=['unctrl'], t=['default'], initial_aids=None, qaid_override=None, daid_override=None, lazy=False, **kwargs*)
Convience function
CommandLine: wbia -e print

Parameters
- **e** (*str*) – (default = 'print')
- **db** (*str*) – (default = 'PZ_MTEST')
- **a** (*list*) – (default = ['unctrl'])
- **t** (*list*) – (default = ['default'])
- **qaid_override** (*None*) – (default = None)
- **lazy** (*bool*) – (default = False)

Returns func - live python function
Return type function

CommandLine: python -m wbia.__init__ –exec-run_experiment –show

Example

```
>>> # DISABLE_DOCTEST
>>> from wbia import *  # NOQA
>>> e = 'rank_cmc'
>>> db = 'testdb1'
>>> a = ['default:species=primary']
>>> t = ['default']
>>> initial_aids = [2, 3, 4, 7, 9, 10, 11]
>>> qaid_override = [1, 9, 10, 11, 2, 3]
>>> testres = run_experiment(e, db, a, t, qaid_override=qaid_override,
>>>                          initial_aids=initial_aids)
>>> result = ('testres = %s' % (str(testres),))
>>> print(result)
>>> ut.quit_if_noshow()
>>> testres.draw_func()
>>> ut.show_if_requested()
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

W

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z