Image Photo-Z

API Reference:

This file documents the functions created and used in the package. For a complete description of the algorithm see the "Getting to know the image-photo-z pipeline" document.

A. *catalog_gen* module: This module implements querying the SDSS SkyServer for a catalog of objects that image-photo-z may use.

- Function *query_area(outputCatalog, minRA, maxRA, minDec, maxDec)*
  Queries DR10 via CasJobs to get a catalog of objects with clean photometric and spectrometric information in the specified area of the sky.
  Parameters:
  - *outputCatalog*: *string*, Name of the output catalog file. ex.: "one_square_degree.csv".
  - *minRA*: *float*, Minimum requested RA in degrees. ex.: 180.
  - *maxRA*: *float*, Maximum requested RA in degrees. ex.: 181.
  - *minDec*: *float*, Minimum requested Dec in degrees. ex.: 45.
  - *maxDec*: *float*, Minimum requested Dec in degrees. ex.: 46.
- Function *finalize_catalog(inputCatalog, outputCatalog)*
  Finalizes the catalog generated by query_area for use by image-photo-z.
  Parameters:
  - *inputCatalog*: *string*, Name of the input catalog file. ex.: "one_square_degree.csv".
  - *outputCatalog*: *string*, Name of the output catalog file. ex.: "one_square_degree_processed.csv".

B. *image_registration* module: This module implements registring images, that is, aligning them in different bands.

- Function *register_reproject(inputImages, outputImages, refImage, processDir, headerName="header.hdr")*
  Registers input images using the reprojection routine in Montage.
  Parameters:
  - *inputImages*: *list*, List of input image names. ex.: ["u.fits", "r.fits"].
  - *outputImages*: *list*, List of output image names. ex.: ["u_reg.fits", "r_reg.fits"].
  - *refImage*: *string*, Name of the reference image, one of *inputImages*. ex.: "r.fits."
  - *processDir*: *string*, Name of the root of the directory where all images are kept. ex.: "processing".
  - *headerName*: *string*, optional, Name of the header file generated.
- Function *reg_check_diff(check, ref, out)*
  Checks image registration between *check* and *ref* by generating a difference image.
  Parameters:

- *check*: *string*, Name of the image whose registration is to be checked. ex.: "u_reg.fits".
- *ref*: *string*, Name of the reference image. ex.: "r.fits".
- *out*: *string*, Name of the generated difference image. ex.: "r_diff.fits".

C. *generate_training* module: This module generates pixel training data as required by image-photo-z.

- Function *preprocess_catalog(catalog, output)*
  Preprocess given catalog for generate_training.
  Parameters:
  - *catalog*: *string*, Input catalog filename. ex.: "one_square_degree.csv".
  - *output*: *string*, Output catalog filename. ex.: "one_square_degree_processed.csv".
- Function *generate_download_list(catalog, output, bands=['u','g','r','i','z'], rerun="301")*
  Generates a image URL list of objects in *catalog* to pass to *wget* to download.
  Parameters:
  - *catalog*: *string*, Name of the catalog file. ex.: "one_square_degree_processed.csv".
  - *output*: *string*, Name of the output file. ex.: "download.list".
  - *bands*: *list*, optional, List of filters to download images in. ex.: ['u', 'r'].
  - *rerun*: *string*, optional, String of the current rerun number for the SDSS pipeline.
- Function *download_images(catalog, downloadFolder, logfile="logfile", bands=['u','g','r','i','z'], rerun="301")*
  Generates a image URL list of objects in *catalog* to pass to *wget* to download.
  Parameters:
  - *catalog*: *string*, Name of the catalog file. ex.: "one_square_degree_processed.csv".
  - *output*: *string*, Name of the output file. ex.: "download.list".
  - *bands*: *list*, List of filters to download images in. ex.: ['u', 'r'].
- Function *make_logfile(catalog, logfile="logfile")*
  Generates a logfile of downloaded image names for further use.
  Parameters:
  - *catalog*: *string*, Name of the catalog file. ex.: "one_square_degree_processed.csv".
  - *logfile*: *string*, Name of the output logfile. ex.: "logfile".
- Function *convert_catalog_to_exp_pixels(filename, catalog, expPixelsList)*
  Converts a FITS header and the given catalog information to a list of expected pixel locations for objects.
  Parameters:
  - *filename*: *string*, Name of the FITS file whose header is to be used. ex.: "r.fits".
  - *catalog*: *string*, Name of the catalog file. ex.: "one_square_degree_processed.csv".
  - *expPixelsList*: *string*, Name of the output listfile. ex.: "sky.list".
- Function *sextract(imageFileNames, configFileNames, processDir, refBand='r', bands=['u', 'g', 'r', 'i', 'z'])*
  Runs SExtractor and associates objects from the expected pixels list to the detected objects.
  Parameters:
  - *imageFileNames*: *list*, List of the FITS input image files. ex.: ["r.fits", "g.fits"].
  - *configFileNames*: *list*, List of SExtractor config files in the same order as the FITS files. ex.:

['u.sex', 'g.sex'].
    - *processDir*: *string*, Name of the process directory in which the images are kept. ex.: "processing".
    - *refBand*: *string*, optional, Reference band for the source extraction. ex.: 'r'.
    - *bands*: *list*, optional, List of filters in which images have to be processed. ex.: ['u', 'g', 'r', 'i', 'z']
- Function *generate_training_objects(objectsFileName, segImageName, catalog, imageFileNames, catagory, outdir)*
  Generates pixel-level training data.
  Parameters:
    - *objectsFileName*: *string*, Name of the SExtractor output catalog file. ex.: "ref.cat".
    - *segImageName*: *string*, Name of the SExtractor output segmentation image. ex.: "ref_seg.fits".
    - *catalog*: *string*, Name of the catalog file. ex.: "one_square_degree_processed.csv".
    - *imageFileNames*: *list*, List of the input filenames. ex.: ["r.fits", "g.fits"].
    - *catagory*: *list*, Type of pixels to choose. Some of ["GALAXY", "STAR", "QSO"].
    - *outdir*: *string*, Name of the directory in which output results should be placed. ex.: "data".
- Function *generate_training_background(segImageNames, imageFileNames, outdir, nMaxDataPoints=1000)*
  Generates pixel-level training data.
  Parameters:
    - *segImageNames*: *list*, List of the SExtractor output segmentation image names. ex.: ["r_seg.fits", "u_seg.fits"].
    - *imageFileNames*: *list*, List of the input filenames. ex.: ["r.fits", "g.fits"].
    - *outdir*: *string*, Name of the directory in which output results should be placed. ex.: "data".
    - *nMaxDataPoints*: *int*, optional, Maximum number of background pixels to retain as data. ex.: 1000.

C. *training* module: This module trains a machine learning algorithm using data generated by generate_training. THis module currently implements training by k-nearest neighbors.

- Function *prepare_for_training_kNN_regression(catagories, dataDir, outfiles)* Prepare data for regression training.
    - *catagory*: *list*, Type of pixels to choose. Some of ["GALAXY", "STAR", "QSO"].
    - *dataDir*: *string*, Name of the directory in which input results are located. ex.: "data".
    - *outfiles*: *list*, [Name of datafile, name of targetfile]. ex.: ["training/trainingData.train", "training/trainingTargets.train"].
- Function *train_test_kNN_regression(trainData, trainTargets, testData, testTargets, testPredictions, nNeighbors=5)* Train a kNN regressor and test it using some data.
    - *trainData*: *string*, Name of the training datafile. ex.: "trainingData.train".
    - *trainTargets*: *string*, Name of the training targetfile. ex.: "trainingTargets.train".
    - *testData*: *string*, Name of the testing datafile. ex.: "testingData.train".
    - *testTargets*: *string*, Name of the testing targetfile. ex.: "testingTargets.train".
    - *testPredictions*: *string*, Name of the testing prediction file. ex.: "testingPredictions.train".
    - *nNeighbors*: *int*, optional, Number of neighbors to use. ex.: 5.
- Function *generate_kNN_output_regression(testingPredictions, testingTargets, outfile)* Generate an

output file consisting of predicted object averages and actual values from regression output files for further processing.
    - *testPredictions*: *string*, Name of the testing prediction file. ex.: "testingPredictions.train".
    - *testingTargets*: *string*, Name of the testing targetfile. ex.: "testingTargets.train".
    - *outfile*: *string*, Name of the output file. ex.: "kNNOutput_regression.csv".
- Function *prepare_for_training_kNN_classification(catagories, dataDir, outfiles)* Prepare data for classification training.
    - *catagory*: *list*, Type of pixels to choose. Some of ["GALAXY", "STAR", "QSO"].
    - *dataDir*: *string*, Name of the directory in which input results are located. ex.: "data".
    - *outfiles*: *list*, [Name of datafile, name of targetfile]. ex.: ["training/trainingData.train", "training/trainingTargets.train"].
- Function *train_test_kNN_classification(trainData, trainTargets, testData, testTargets, testPredictions, nNeighbors=5)* Train a kNN classifier and test it using some data.
    - *trainData*: *string*, Name of the training datafile. ex.: "trainingData.train".
    - *trainTargets*: *string*, Name of the training targetfile. ex.: "trainingTargets.train".
    - *testData*: *string*, Name of the testing datafile. ex.: "testingData.train".
    - *testTargets*: *string*, Name of the testing targetfile. ex.: "testingTargets.train".
    - *testPredictions*: *string*, Name of the testing prediction file. ex.: "testingPredictions.train".
    - *nNeighbors*: *int*, optional, Number of neighbors to use. ex.: 5.
- Function *generate_kNN_output_classification(testingPredictions, testingTargets, outfile round_off=0)* Generate an output file consisting of predicted object averages and actual values from classification output files for further processing.
    - *testPredictions*: *string*, Name of the testing prediction file. ex.: "testingPredictions.train".
    - *testingTargets*: *string*, Name of the testing targetfile. ex.: "testingTargets.train".
    - *outfile*: *string*, Name of the output file. ex.: "kNNOutput_classification.csv".
    - *round_off*: *bool*, Switch to select whether or not to round-off classification average results. One of "True" and "False".

C. *get_images* module: This module gets images from SDSS servers via Montage utilities.

- Function *get_image_lists(obj, width, height, survey="SDSS", bands=['u','g','r','i','z'])* Generate image lists for the requested coordinates or object.
    - *obj*: *string*, Name of the requested object or coordinate pair. ex.: "m31", "180.24, 52.18".
    - *width*: *float*, Width of the requested frame in degrees. ex.: 1.2.
    - *height*: *float*, Height of the requested frame in degrees. ex.: 1.2.
    - *survey*: *string*, optional, Name of the survey. One of "SDSS", "DSS", "2MASS", "DPOSS".
    - *band*: *list*, optional, List of the filter names in which images are to be obtained. A list of all filters for all surveys is given here.
        - SDSS: u, g, r, i, z
        - 2MASS: j, h, k
        - DPOSS: f, j, n
        - DSS: DSS1, DSS1R, DSS1B, DSS2, DSS2B, DSS2R, DSS2IR