4D Use Case: Phase 3: RANSAC Filtering

Outputs from the ASIFT Executable have a large # of “false positive” matching keypoints. An implementation of the RANSAC algorithm filters these points to generate a subset of keypoints with higher confidence.

(NOTE: Some RANSAC results, especially in poorly-fit images, generate non-optimal homography matrices that require unrealistic distortions between source & target images. These can be filtered out, but this is not yet implemented).

## RANSAC Filter

Starting with the “data\_matches.csv” output from the ASIFT Executable in the last phase, this runs a single RANSAC filter on the results, saving the outputs in a “data\_matches\_FILTERED.csv” file.

Usage in Python:

```

def filter\_ASIFT\_matches(matching\_keypoints\_filename,

output\_CSV=None,

output\_good\_points\_only=True,

verbose=True,

ransac\_reprojection\_threshold=3,

eliminate\_nodata\_matches=True,

img1\_filename=None,

img1\_nodata=None,

img2\_filename=None,

img2\_nodata=None):

```

Usage standalone:

```

usage: RANSAC\_filter.py [-h] [-ransac\_threshold RANSAC\_THRESHOLD]

[-img1\_filename IMG1\_FILENAME]

[-img1\_nodata IMG1\_NODATA]

[-img2\_filename IMG2\_FILENAME]

[-img2\_nodata IMG2\_NODATA] [--verbose]

[--filter\_nodata\_matches]

matches\_filename output\_filename

Filter keypoint matches using the RANSAC filter. Output a subset of keypoints that pass RANSAC filtering.

positional arguments:

matches\_filename Name of the CSV file from the ASIFT executable with

matching keypoints (typically 'data\_matches.csv').

output\_filename Name of the output CSV file to store the RANSAC-

filtered matches.

optional arguments:

-h, --help show this help message and exit

-ransac\_threshold RANSAC\_THRESHOLD

Pixel threshold for RANSAC filter (Default 3).

-img1\_filename IMG1\_FILENAME

The name of the source image, for filtering out nodata

values.

-img1\_nodata IMG1\_NODATA

The nodata value to use in the source image (default:

find in image metadata)

-img2\_filename IMG2\_FILENAME

The name of the target image, for filtering out nodata

values.

-img2\_nodata IMG2\_NODATA

The nodata value to use in the target image (default:

find in image metadata)

--verbose Increase output verbosity

--filter\_nodata\_matches

Filter out points adjacent to nodata values.

```

## Matches Reader

A small utility for reading the “data\_matches.csv” outputs from the ASIFT Executable and creating OpenCV Keypoint objects for the RANSAC filter.

Usage in Python:

```

def read\_matches\_file(filename, include\_keypoints=False):

'''Reads an ASIFT Executable match file.

Inputs:

- filename: name of the "matches.csv" file

- include\_keypoints: (bool), If False, just return a numpy array of matches from the file.

If True, return two lists of cv2.KeyPoint objects along with the numpy array of matches from the file.

Return Values:

(If include\_keypoints = False):

- numpy array of data from the .csv file.

(If include\_keypoints = True):

- list of cv2.KeyPoint objects from source image

- list of cv2.KeyPonit objects from target image

- numpy array of data from the .csv file.

```

Usage standalone:

*(Standalone version not implemented)*

## DrawMatches

A utility for drawing image pairs and writing them to disk, using filtered or unfiltered results. Useful for visualization of results.

Contains two functions with slightly-different implementations

Usage in Python:

```

def drawMatches\_from\_file(im1\_name, im2\_name, matches\_file, output\_img\_name, gap\_pixels = 20,

gap\_color = 255,

im1\_XOff=None,

im1\_YOff=None,

im1\_XSize=None,

im1\_YSize=None,

im2\_XOff=None,

im2\_YOff=None,

im2\_XSize=None,

im2\_YSize=None,

verbose=True):

def drawMatches\_from\_keypoints(im1\_name, im2\_name, keys1, keys2, matches, output\_img\_name, gap\_pixels = 15,

gap\_color = None,

im1\_XOff=None,

im1\_YOff=None,

im1\_XSize=None,

im1\_YSize=None,

im2\_XOff=None,

im2\_YOff=None,

im2\_XSize=None,

im2\_YSize=None,

verbose=True):

```

## KeyPoint Recombiner

(not yet implemented) Takes RANSAC results between individual tiles of the “source” and “target” images, recombines them, transforms the coordinates back into whole-image-space. Performs another RANSAC filter to maintain internal geometric consistency between the image matches. Outputs a final matches CSV to output back to the user. (Optional: If the target image is geo-located, provides image coordinates as well as geographic coordinates of keypoints in the source image.)

Usage in Python:

*(Not yet implemented, coming soon)*

Usage standalone:

*(Not yet implemented, coming soon)*

## Dependencies

The code requires Python 2.7+ to execute.

Code within the phases depends upon the following external Python libraries:

* GDAL v2.2+ : the GeoSpatial Data Abstraction Library ([www.gdal.org](http://www.gdal.org)), and associated Python “gdal” and “osgeo” bindings
* OpenCV v3.2+ : Open Computer Vision libraries and associated Python “cv2” bindings

## License

(TODO: Fill in the open-source license to be used here.)

## Acknowledgements

This project is funded by the NSF’s ICEBERG Cyber-Infrastructure grant.

## Revisions:

1.0 – 2019.02.10 – Draft document by Mike MacFerrin