4D Use Case: Phase 1: Image Pre-Processing

## Overview

Functions and scripts contained in the 4D “PHASE\_1\_IMAGE\_PREPROCESSING” directory.

## Tile Planner

File: Tile\_Planner.py

Images are divided into “tiles,” the details of which are stored in a “Tiles” CSV file in a Scratch directory. Tiles are optimized to stay within a range of sizes, minimize “no data” portions of the image, and avoid “sliver” tiles at the edges of images. Individual tiles may be written to disk. Either way, their information is saved for later use in the “Tiles\_XXX.csv” files in a folder in the Scratch Directory.

Usage in Python:

```

def create\_tiles\_from\_image(img\_filename,

csv\_file=None,

tilesize=3000,

nodata=None,

minimum\_size=1000,

pyramid\_levels=0,

write\_tiles\_to\_disk=True,

overwrite=False,

contrast\_enhance=None, # (choices so far: None or 'equalize')

verbose=True):

Given an image file name:

1) Create a directory where the processing on this tile will take place.

2) Create a set of pyramid images (if pyramid\_levels > 0) to add to the tiles queue.

3) For each pyramid image, tile it up into "tilesize" portions, no smaller than "minimum\_size" on any one dimension (if smaller, will tack onto an adjacent tile).

4) Check sizes and nodata values in each tile. If nodata isn't specified, get it from the image if it's there.

5) Output all individual tiles to a CSV file, return the CSV file name to the calling function.

csv\_file can be a local filename without a path (if so, put it in the scratch directory), or a full path name, in which case it will stay where it was said.

If csv\_file is None, the file will be given a the same name as the scratch directory and put in there.

Inputs:

img\_filename : name of the image file to process

csv\_file : name of the CSV file to write out the tile information

tilesize: The standard approximate size (in pixels) of individual tiles. Tiles may be larger if combined with slivers on either side.

minimum\_size : minimum size (in pixels) that a tile can have. If the tile less than this size in width or height, it is considered a “sliver” (usually at the image edge) and combined with the tile next to it.

pyramid\_levels : Number of factor-of-two pyramid levels to create when processing the image. 0 implies only the original image. 3 implies scale-factors of 1 (original), 2, 4, and 8. All pyramid images are tiled the same as the original.

write\_tiles\_to\_disk : (True/False) It is not always necessary to write tiles to disk. If False, the boundaries of the tiles are simply saved in the “csv\_file”, and tiles are read individually in GDAL.

overwrite : (True/False) Only used if “write\_tiles\_to\_disk” is true. If True, overwrite the tiles even if they already exist for this image. If False, if tiles already exist, do not overwrite.

contrast\_enhance : Contrast-enhance the tiles before writing out to disk. Choices so far: None (no contrast enhancement) and “equalize” (use an histogram equalization routine). Other choices can still be written in here.

verbose : If True, increase verbosity in the output.

```

Usage standalone:

```

usage: python Tile\_Planner.py [-h] [-tilesize TILESIZE] [-csv\_file CSV\_FILE]

[-nodata NODATA] [-contrast\_enhance CONTRAST\_ENHANCE]

[-minimum\_size MINIMUM\_SIZE]

[-pyramid\_levels PYRAMID\_LEVELS] [--dont\_write\_tiles]

[--overwrite] [--verbose]

image

Create a CSV defining tiles of an image and create a csv summary file.

positional arguments:

image Path to the source image (tif or png)

optional arguments:

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

-tilesize TILESIZE Size (pixels) of the tiles (default: 3000)

-csv\_file CSV\_FILE Path to the output CSV file of tile descriptors

(default: put in same directory as the tiles.)

-nodata NODATA No-data value in the source image (default: -999, maps

to None)

-contrast\_enhance CONTRAST\_ENHANCE

Method of contrast-enhancing output tiles. Only

applied if tiles are written to disk. Choices so far:

'none' or 'equalize'. Default: 'none'

-minimum\_size MINIMUM\_SIZE

Minimum size tolerance (pixels) on any axis to avoid

slivers (default: 1000). Slivers less than that size

will be appended to neighboring tiles.

-pyramid\_levels PYRAMID\_LEVELS

Number of factor-of-two pyramid levels to create tiles

(default 0, meaning original size only)

--dont\_write\_tiles Skip writing tiles to disk. If set, just create the

CSV with tile descriptions (does not delete tiles if

they already exist). (Default: write the tiles.)

--overwrite Overwrite pyramids and tiles if they already exist for

this image. (default False)

--verbose Increase output verbosity

```

## Tile Writer

File: Tile\_Writer.py

(Optional) Writes the tiles out to disk. Performs contrast enhancement on each individual tile if requested.

Usage in Python:

```

def write\_tile(src\_filename,

dst\_filename,

data\_array=None,

contrast\_enhance="equalize",

nodata=None,

xoff=None,

yoff=None,

xsize=None,

ysize=None,

overwrite=False,

verbose=True):

```

Usage standalone:

```

usage: python Tile\_Writer.py [-h] [-nodata NODATA] [-xoff XOFF] [-xsize XSIZE]

[-yoff YOFF] [-ysize YSIZE]

[-contrast\_enhance CONTRAST\_ENHANCE] [--overwrite]

[--verbose]

source dest

Select a tile from an image, contrast-enhance if requested, output to a new image.

positional arguments:

source Path to the source image (tif or png)

dest Path to the desination tile to be created (tif or png)

optional arguments:

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

-nodata NODATA No-data value in the source image (default: -999,

translates to None)

-xoff XOFF X-offset for the upper-left corner of the tile, in

pixels

-xsize XSIZE Y-offset for the upper-left corner of the tile, in

pixels

-yoff YOFF X-size of the tile, in pixels

-ysize YSIZE Y-size of the tile, in pixels

-contrast\_enhance CONTRAST\_ENHANCE

Contrast enhancement, choices: none, equalize (default

none)

--overwrite Overwrite tile if already on disk

--verbose Increase the output verbosity

```

## Contrast Enhancer

Tile may have their contrast enhanced to aid the keypoint matching algorithms, and/or to optimize matching in particular areas of the image (example: rock) over others (snow and/or water).

Usage in Python:

```

def enhance\_contrast(data, algorithm="equalize", output\_min=None, output\_max=None):

data : an MxN numpy array of pixel values

algorithm : the algorithm to use. Right now onoly “equalize” is implemented.

output\_min : the minimum value to output in the image. Useful if 0 is “nodata” in the image (to avoid pixel-value conflicts)

output\_max : the maximum value to output in the image.

```

Usage standalone:

*(Standalone not implemented.)*

## Pyramid Builder

This utility is optional. Before tiling, images may be downscaled into pyramids, to perform image matching on lower-resolution versions of the image (useful for initial processing and sometimes helpful for the image-matching algorithms). Pyramid images are saved in the same Scratch directory as the tiles.

Usage in Python:

```

def write\_image\_pyramid(src\_img, num\_levels, dest\_dir=None, nodata=None, overwrite=False, verbose=True):

```

Usage standalone:

```

usage: python Image\_Pyramid.py [-h] [-dest\_dir DEST\_DIR] [-nodata NODATA] [--verbose]

[--overwrite]

source\_img N

Create N levels of pyramid images from a source image, save it to destination

filenames beginning with 'dest\_filebase'.

positional arguments:

source\_img Path to the source image (tif or png)

N Number of factor-of-two levels to pyramid. 0 is only the

original file. (default: 1)

optional arguments:

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

-dest\_dir DEST\_DIR Path to the directory of the destination image tiles

(default: same directory as source image)

-nodata NODATA Nodata value in the image (default: use NoDataValue

found in image)

--verbose Increase the output verbosity

--overwrite If present, overwrite the tiles created if image was

already processed previously. (default: Keep tiles there

if already used.)

```

## Scratch Directory Manager

A utility for organizing sub-folders in the Scratch directory. These routines are useful to save temporary results for images that may be re-used (i.e. an airborne “source” photo that will be searched over multiple satellite “target” images, and vice-versa). Tiles may be written once and re-used on multiple runs before being deleted.

Usage in Python:

```

m = SCRATCH\_Manager()

dirpath = m.add\_directory(img\_filename, verbose=True)

dirpath = m.lookup(img\_filename)

m.remove\_directory(img\_filename, verbose=False)

m.remove\_all\_directories(verbose=True)

```

Usage standalone:

```

usage: python Scratch\_Directory\_Manager.py [-h] [-add ADD] [-remove REMOVE]

[-lookup LOOKUP] [--remove\_all]

[--verbose]

Add, remove, or look up the temporary working directory, which stores

intermediate results for a given image file.

optional arguments:

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

-add ADD A filename. Adds a directory to store tiles and/or temporary

results for the given file.

-remove REMOVE An image filename. Remove a directory for the given

filename, typically after processing is done.

-lookup LOOKUP Look up the directory for the given filename and print it to

the screen.

--remove\_all Clean up the scratch directory and remove all files &

subfolders.

--verbose Increase the output verbosity```

## Run Case Planner

Take the lists of tiles output by the “Tile Generator” for both the source and target images, and matches them into pairs to be executed against each other in the next phase. Saves the output to a new CSV file. If the user/coder only wishes to select certain subsets of tiles to run against each other (only at certain pyramid levels, or only one “target” tile against which to search all the “source” tiles, e.g.), this is the place where those selections are made.

Usage in Python:

```

def generate\_run\_case\_CSVs(source\_csv,

target\_csv,

output\_csv = None,

row\_filter="all",

source\_pyramid\_levels=None,

target\_pyramid\_levels=None,

verbose=True):

```

Usage standalone:

```

usage: python Run\_Case\_Planner.py [-h] [-filter FILTER]

[-source\_pyramid\_levels SOURCE\_PYRAMID\_LEVELS]

[-target\_pyramid\_levels TARGET\_PYRAMID\_LEVELS]

[--verbose]

source\_csv target\_csv output\_csv

Take two image tile CSV files generated by 'Tile\_Planner.py', and select

individual tiles to run against each other.

positional arguments:

source\_csv Filename for the 'Tiles\_[].csv' file of the source

image.

target\_csv Filename for the 'Tiles\_[].csv' file of the target

image.

output\_csv Filename for the CSV file for tile pairs to be run by

the ASIFT Executable.

optional arguments:

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

-filter FILTER Filter to use. Options: "all", "pyramid\_levels"

(default: all)

-source\_pyramid\_levels SOURCE\_PYRAMID\_LEVELS

If -filter pyramid\_levels is used: Comma-separated

list of pyramid levels to use from the source image.

Example: 2,4,8,16 or "2, 4, 8, 16". Default to using

them all.

-target\_pyramid\_levels TARGET\_PYRAMID\_LEVELS

If -filter pyramid\_levels is used: Comma-separated

list of pyramid levels to use from the target image.

Example: 2,4,8,16 or "2, 4, 8, 16". Default to using

them all.

--verbose Increase the output verbosity

```

## Dependencies

The code requires Python 2.7+ to execute.

External Python Libraries:

scikit-image (skimage): used by Contrast\_Enhancer.py

gdal (osgeo): used by Image\_Pyramid.py, Tile\_Planner.py, Tile\_Writer.py. Requires installation of GDAL Libraries (www.gdal.org) v2.2+

## 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