CanoPy technical manual

CanoPy is the Python module for the Georgia Canopy Analysis 2009 project sponsored by the Georgia Forestry Commission (GFC). For further information about this project, please refer to the CanoPy page.

This document outlines the uses and methodology of the functions contained within the CanoPy module. To learn how to use this module, please read the CanoPy user manual. Also refer to the CanoPy tutorial.

Authors

- Owen Smith
- Huidae Cho, Ph.D., GISP, PE (MD), CFM, M.ASCE, Ph.D.

Requirements

- ArcGIS Desktop 10.x
- ArcPy
- Python 2 standard module: os
- Feature Analyst™ by the Textron Systems
- Automated Feature Extraction (AFE) models trained using Feature Analyst

We are currently planning on developing a fully open source solution without using ArcGIS and Feature Analyst.

canopy_config module

Contained in `canopy_config.py` are all the data paths that CanoPy functions operate with. Example configuration can be found in `canopy_config-example.py` and copied into `canopy_config.py`. This module will be imported by the CanoPy module.

phyregs_layer

- Type: `str`
- Layer containing polygon features for all physiographic regions
- Required attribute fields:
  - NAME (Text)
  - PHYSIO_ID (Long)
  - AREA (Float)
- Example: `phyregs_layer = 'Physiographic_Districts_GA'`

naipqq_layer

- Type: `str`
- Layer containing polygon features for all NAIP quarter quad (QQ) tiles
- Required attribute field:
  - `FileName` (Text)
- Example: `naipqq_layer = 'naip_ga_2009_1m_m4b'`

naipqq_phyregs_field

- Type: `str`
- New field name for assigning physiographic region IDs to the `naipqq_layer`
- This output text field will be created in the `naipqq_layer` by `canopy.assign_phyregs_to_naipqq()`. `PHYSIO_ID`s from the `physregs_layer` in which a `naipqq_layer` polygon is contained are output into this field.
- Output format: `,#,#,…,`
- Example: `naipqq_phyregs_field = 'phyregs'`

naip_path

- Type: `str`
- Input folder in which NAIP imagery is stored
- The structure of this input folder is defined by USDA, the original source of NAIP imagery. Under this folder are multiple 5-digit numeric folders that contain actual imagery GeoTIFF files.

```
F:/Georgia/ga/
              34083/
                    m_3408301_ne_17_1_20090929.tif
                    m_3408301_ne_17_1_20090930.tif
                    ...
              34084/
                    m_3408407_ne_16_1_20090930.tif
                    m_3408407_nw_16_1_20090930.tif
                    ...
              ...
```

- Example: `naip_path = 'F:/Georgia/ga'`

spatref_wkid

- Type: `int`
- Desired output coordinate system in WKID format
- Well-Known IDs (WKIDs) are numeric identifiers for coordinate systems administered by Esri. This variable specifies the target spatial reference for output files. The WKID used for the GFC canopy project is 102039 (USA Contiguous Albers Equal Area Conic USGS version).
- Example: `spatref_wkid = 102039`

project_path

- Type: `str`
- Folder path with which all other output paths are determined
- The default structure of the project folder is defined as follows:

```
C:/.../ (project_path)
       Data/
            Physiographic_Districts_GA.shp (added as a layer)
       Results/
               gtpoints_Winder_Slope.shp
               ...
       2009 Analysis/ (analysis_path)
                     Data/
                          naip_ga_2009_1m_m4b.shp (added as a layer)
                          snaprast.tif (snaprast_path)
                     Results/ (results_path)
                             Winder_Slope/ (physiographic region name)
                                          Inputs/
                                                 reprojected NAIP tiles
                                          Outputs/
                                                  intermediate output
tiles
canopy_2009_Winder_Slope.tif
                                ...
       2019 Analysis/ (analysis_path)
                     Data/
                          naip_ga_2019_1m_m4b.shp (added as a layer)
                          snaprast.tif (snaprast_path)
                     Results/ (results_path)
                             Winder_Slope/ (physiographic region name)
                                          Inputs/
                                                 reprojected NAIP tiles
                                          Outputs/
                                                  intermediate output
tiles
canopy_2019_Winder_Slope.tif
                                ...
       ...
```

- **NOTE:** Output folder must be manually created. It is used when running Feature Analyst and is **_NOT_** created by CanoPy.
- Example: `project_path = 'C:/work/Research/GFC Canopy Assessment'`

analysis_path_format

- Type: `str`
- Format of the analysis path for one year
- Example: `analysis_path_format = '%s/%%d Analysis' % project_path`

analysis_year

- Type: `int`
- Year for analysis
- Example: `analysis_year = 2009`

snaprast_path

- Type: `str`
- Snap raster to which all output tiles will be snapped
- This input/output raster is used to snap NAIP tiles to a consistent grid system. If this file does not already exist, the filename part of `snaprast_path` must be `r` + the filename of an existing original NAIP tile so that `canopy.reproject_naip_tiles()` can automatically create it based on the folder structure of the NAIP imagery data (`naip_path`).
- Example: `snaprast_path = '%s/Data/rm_3408504_nw_16_1_20090824.tif' % analysis_path`

results_path

- Type: `str`
- Folder where all results will be stored
- Example: `results_path = '%s/Results' % analysis_path`

canopy module

All functions designed for preproccessing NAIP imagery and for postprocessing trained/classified canopy tiles in addition to utility functions are contained within `canopy.py`.

NOTE: The `physregs_layer` and `naipqq_layer` must be added to an ArcMap or ArcGIS Pro dataframe for CanoPy functions to run.

assign_phyregs_to_naipqq()

This function adds the phyregs field to the NAIP QQ shapefile and populates it with physiographic region IDs that intersect each NAIP tile.

Arguments: None

Config variables assigned with `canopy_config`:

- `phyregs_layer = canopy_config.phyregs_layer`
- `naipqq_layer = canopy_config.naipqq_layer`
- `naipqq_phyregs_field = canopy_config.naipqq_phyregs_field`

Process:

1. The data fields of the input NAIP QQ shapefile are read using `arcpy.ListFields` and a new

text field titled `naip_phyregs_field` is added. If the field already exists, it is deleted and a new field is created.
2. Using `arcpy.CalculateField_managment`, a comma (`,`) is inserted into the newly created `naip_phyregs_field`. This becomes important as the format for the `naip_phyregs_field` must be `,#,#,…,` to allow for SQL statments in following functions to be able to read the `naip_phyregs_field` properly. The SQL selections will allow for the right NAIP tiles to be computed as the NAIP QQ shapedfile has a corresponding field for ile names.
3. All selections are cleared and each NAIP QQ polygon will contain the `naip_phyregs_field` filled with the IDs of physiographic regions that the QQ tile intersects.

reproject_naip_tiles(phyreg_ids)

This function reprojects and snaps the NAIP tiles that intersect selected physiographic regions.

Arguments:

- `phyreg_ids` (list of `int`): IDs of physiographic regions to process

Config variables assigned with `canopy_config`:

- `phyregs_layer = canopy_config.phyregs_layer`
- `naipqq_layer = canopy_config.naipqq_layer`
- `naipqq_phyregs_field = canopy_config.naipqq_phyregs_field`
- `spatref_wkid = canopy_config.spatref_wkid`
- `snaprast_path = canopy_config.snaprast_path`
- `naip_path = canopy_config.naip_path`
- `results_path = canopy_config.results_path`

Process:

1. The spatial reference desired is set using the WKID specified in the `canopy_config` using `arcpy.SpatialReference` which reads the WKID.
2. If the snap raster does not exist within the `snaprast_path`, it is created and `arcpy.env.snapRaster` is used to set all output cell alignments to match the snap.
3. All NAIP tiles intersecting the input `phyreg_ids` are selected using an SQL clause to select the `phyreg_ids`.
4. The `FileName` field from each selected NAIP QQ polygon is read.
5. Using `arcpy.ProjectRaster_managment`, the selected NAIP are reprojected to the specified WKID and saved as outputs and the prefix `r` (**r**eprojected) is added to the filename.
6. The outputs of this function are saved in an inputs folder and are what will used by Textron's Feature Analysis.

convert_afe_to_final_tiles(phyreg_ids)

This function converts Textron's Feature Analyst classified outputs to final GeoTIFF files

Arguments:

- `phyreg_ids` (list of `int`): IDs of physiographic regions to process

Config variables assigned with `canopy_config`:

- `phyregs_layer = canopy_config.phyregs_layer`
- `naipqq_layer = canopy_config.naipqq_layer`
- `naipqq_phyregs_field = canopy_config.naipqq_phyregs_field`
- `snaprast_path = canopy_config.snaprast_path`
- `results_path = canopy_config.results_path`

Process:

1. All NAIP tiles in the desired physiographic region are first selected using an SQL statement to select the input physiographic IDs.
2. The filenames from the NAIP QQ shapefile with the reprojected prefix are used to as the outputs folder created to save the classified imagery is walked through.
3. Conversion is necessary as some AFE models used in feature analysis output GeoTIFF files and some output shapefiles.
    1. If the file is a shapefile, it is converted to raster with classes 1 and 0.
    2. If the file is a GeoTIFF file, the values are reclassified from 1 to 0 and 2 to 1.
    3. If the file has already run through this function and has the appropriate prefix, nothing happens to it.
4. Outputs are saved in the outputs folder with the prefix `fr` (**f**inal **r**eprojected).

clip_final_tiles(phyreg_ids)

This function clips final tiles to their respective NAIP QQ area to eliminate overlap.

Arguments:

- `phyreg_ids` (list of `int`): IDs of physiographic regions to process

Config variables assigned with `canopy_config`:

- `phyregs_layer = canopy_config.phyregs_layer`
- `naipqq_layer = canopy_config.naipqq_layer`
- `naipqq_phyregs_field = canopy_config.naipqq_phyregs_field`
- `snaprast_path = canopy_config.snaprast_path`
- `results_path = canopy_config.results_path`

Process:

1. First, the `OID` field of the entire NAIP QQ shapefile is encoded.
2. All NAIP tiles in the desired physiographic region are first selected using an SQL statement to select the input physiographic IDs.
3. The output files from `canopy.convert_afe_to_final_tiles` are looped over and, using the corresponding `OID` field, are then clipped to their respective NAIP QQ polygons.
4. If the tile has already been clipped and has the appropriate prefix, it will be skipped. If not, the tile will be clipped and the output GeoTIFF will have the prefix `cfr` (**c**lipped **f**inal **r**eprojected).

mosaic_clipped_final_tiles(phyreg_ids)

This function mosaics clipped final GeoTIFF and then clips the mosaicked files to their corresponding physiographic regions

Arguments:

- `phyreg_ids` (list of `int`): IDs of physiographic regions to process

Config variables assigned with `canopy_config`:

- `phyregs_layer = canopy_config.phyregs_layer`
- `naipqq_layer = canopy_config.naipqq_layer`
- `naipqq_phyregs_field = canopy_config.naipqq_phyregs_field`
- `analysis_year = canopy_config.analysis_year`
- `snaprast_path = canopy_config.snaprast_path`
- `results_path = canopy_config.results_path`

Process:

1. All NAIP tiles in the input physiographic regions are first selected using an SQL statement to select the input physiographic IDs.
2. If the mosaicked file with the analysis year set by the `canopy_config` file exists, the function ends. If no mosaiked layer with the analysis year exists, the process continues.
3. Input tiles to be mosiacked are products from `canopy.clip_final_tiles` with the prefix `cfr`.
4. Mosiacking occurs using `arcpy.MosaicToNewRaster` to create the output raster as a new 2 bit GeoTIFF file.
5. The new mosaiked data set is clipped to the outline of the physiographic region with the corresponding physiographic ID.

convert_afe_to_canopy_tif(phyreg_ids)

This function is a wrapper function that converts AFE outputs to the final canopy GeoTIFF file by invoking `convert_afe_to_final_tiles()`, `clip_final_tiles()`, and `mosaic_clipped_final_tiles()` in the correct order.

Arguments:

- `phyreg_ids` (list of `int`): IDs of physiographic regions to process

Config variables assigned with `canopy_config`:

- `phyregs_layer = canopy_config.phyregs_layer`
- `naipqq_layer = canopy_config.naipqq_layer`
- `naipqq_phyregs_field = canopy_config.naipqq_phyregs_field`
- `analysis_year = canopy_config.analysis_year`
- `snaprast_path = canopy_config.snaprast_path`
- `results_path = canopy_config.results_path`

generate_gtpoints(phyreg_ids, point_density, max_points=400,

min_points=200)

This function generates randomized points for ground truthing with fields for corresponding years populated with the corresponding canopy value at each point.

Arguments:

- `phyreg_ids` (list of `int`): IDs of physiographic regions to process
- `point_density` (`int`): Number of randomly generated points per square kilometer
- `max_points` (`int, default=400`): Maximum number of points generated per region
- `min_points` (`int, default=200`): Minimum number of points generated per region

Config variables assigned with `canopy_config`:

- `phyregs_layer = canopy_config.phyregs_layer`
- `naipqq_layer = canopy_config.naipqq_layer`
- `spatref_wkid = canopy_config.spatref_wkid`
- `analysis_year = canopy_config.analysis_year`
- `results_path = canopy_config.results_path`

Process:

1. The physiographic regions are selected using the input physiographic IDs.
2. Random points in each region are created using `arcpy.CreateRandomPoints`.
3. Fields are created in each point shapefile with the field name of GT (e.g., GT).
4. The attributes of the `naip_layer` are spatially joined with the created random points shapefile and saved as a new point layer.
5. A new spatially joined point shapefile allows for the file names of each point's corresponding classified NAIP QQ to be read and converted to a NumPy array. The conversion of each NAIP QQ to a NumPy array allows the function to handle the memory requirements of the regions.
6. Each point is converted to corresponding rows and columns within the corresponding NAIP QQ using `calculate_row_column()`.
7. The value of the NumPy cell at each point's row and column is then added to the GT attribute field of the spatially joined shapefile.
8. After all point values are read, all fields except FID, Shape, and GT are deleted in the spatially joined shapefile.
9. The originally generated point shapefile is deleted.

software