CanoPy FOSS

A Random Forests Based Canopy Classification System Utilizing NAIP Imagery Within A Python Framework

Owen Smith

GISC 4903 Special Topics in GIS
Instructor: Huidae Cho

About

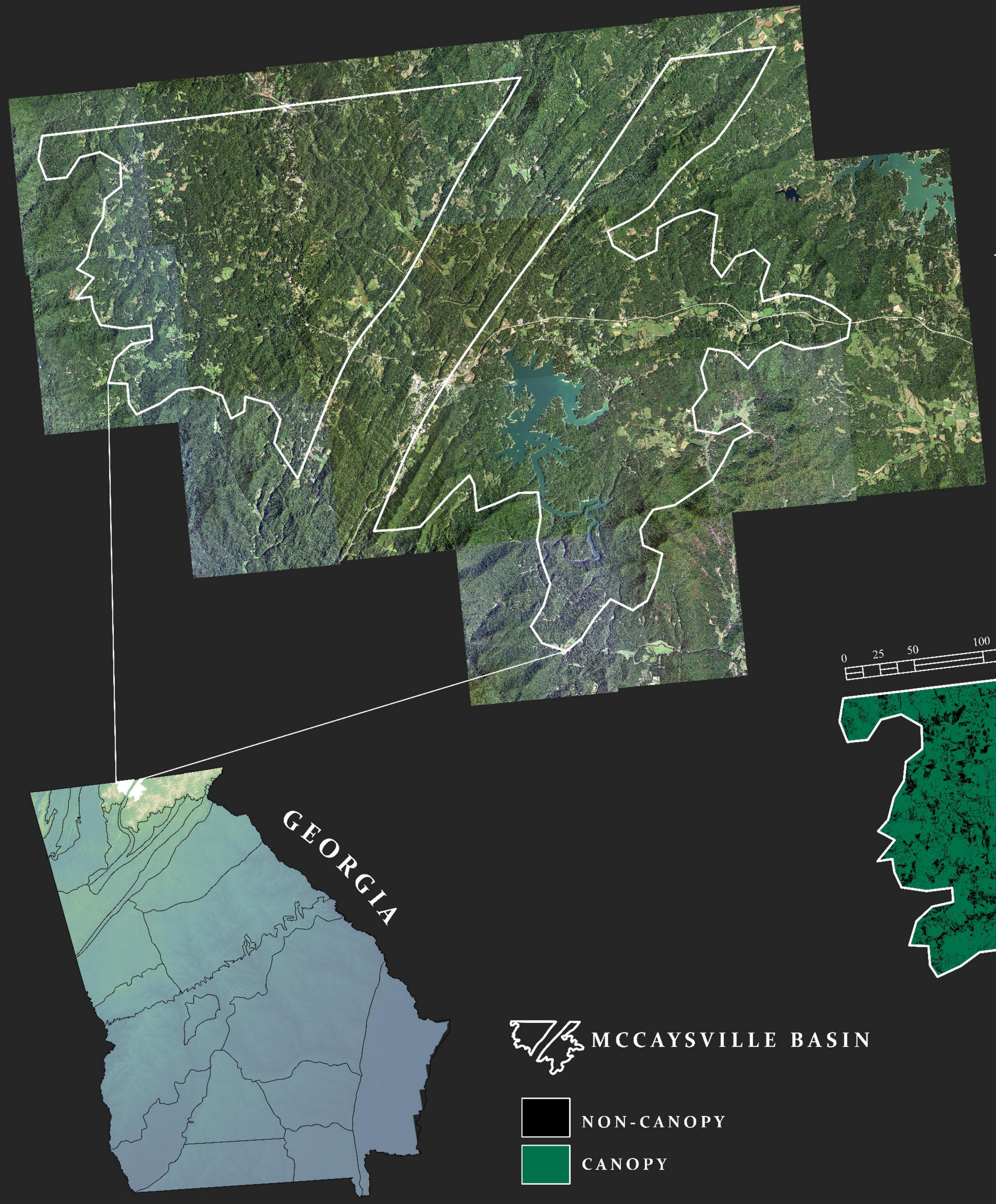
Tree canopy monitoring is an important aspect of maintaining any environment. On a large scale the monitoring and data creation can be a difficult and time-consuming process to undertake. Proprietary software dedicated to completing such tasks may be expensive and with a lack the insight needed into the innerworkings of their classification algorithms leading to open source alternatives. The Python programming language provides an ideal platform for the creation of a scalable method capable of accurately processing the large amounts of data to classify canopy on a large scale. The Scikit-learn Extra Trees classifier is utilized alongside the Geospatial Data Abstraction Library (GDAL) API to enact the method created. The framework created is based around the classification of imagery from the National Agriculture Imagery Program (NAIP) due to its high spatial resolution. The framework is believed to be comparable to proprietary systems in both accuracy and computational time.

Configuration

CanoPy Foss is split into 3 separate .py files – config.py, training.py, & canopy_foss.py.
The configuration is the most important file as it consists of all the file paths that will be used for I/O.

Configuration Parameter		Description
proj	→	EPSG code of projection in which final data will be reprojected to.
workspace	→	Directory where process will output results and read data from
naip_dir	→	Folder that contains all NAIP
results	→	Folder where all regions folders will be created
class_directory	→	Folder within region folders that will contain final outputs after classification
data	→	Folder where all reference and training data is stored
phyreg_lyr	→	Physiographic districts shapefile
clip_naip	→	Original NAIP QQ shapefile to use for clipping
naipqq_shp	→	Joined NAIP QQ tile with PHYSIO_ID's to query filenames
training_raster	→	Rasterized training data
training_fit_raster	→	ARVI raster which training data applies to

A **Scalable** & **Accurate** Open Source Python **Canopy Classification** Module For NAIP Imagery.

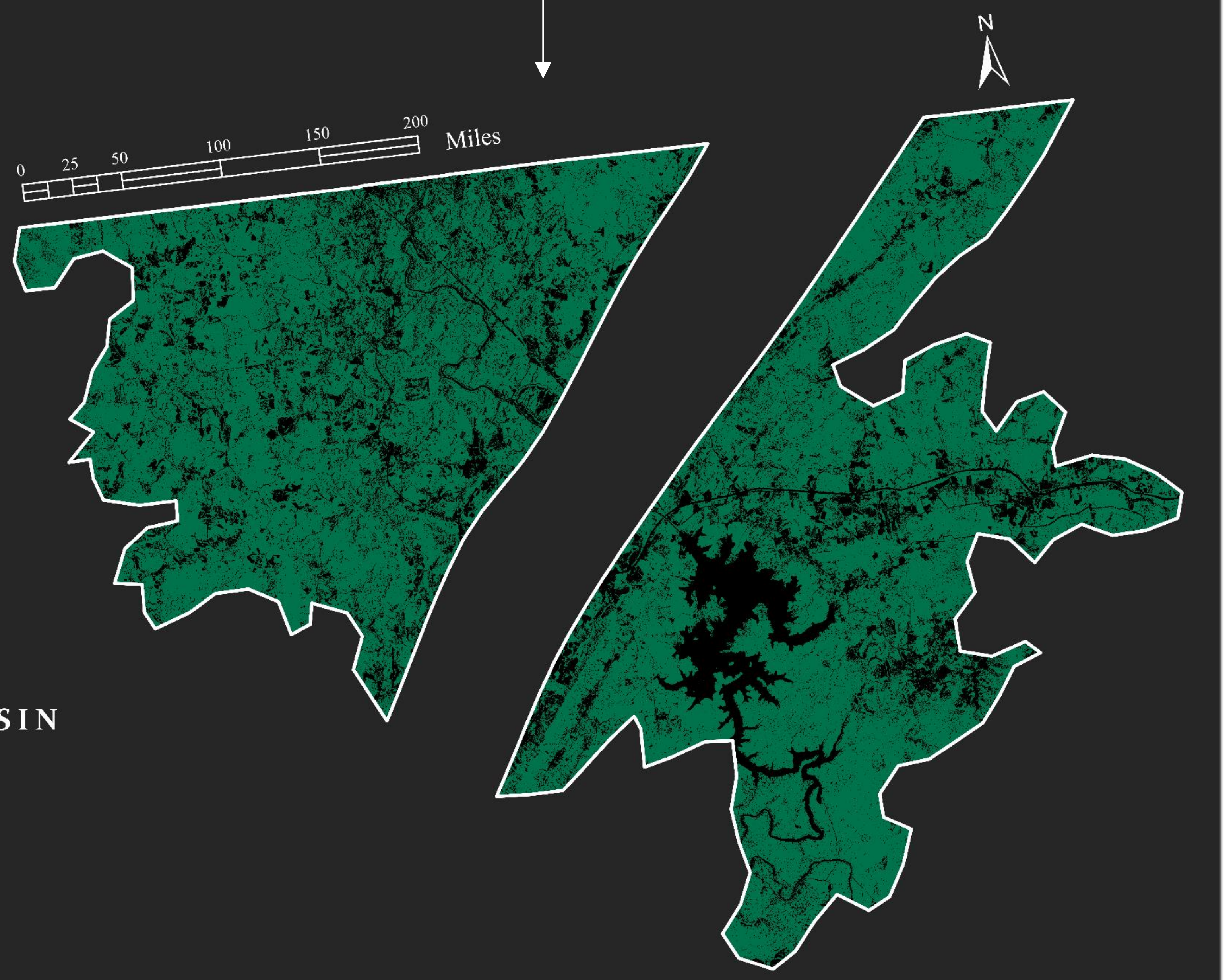


CanoPy FOSS

```
>>> import canopy_foss as cf
>>> cf.ARVI(phy_id=8)
>>> cf.batch_extra_trees(phy_id=8)
>>> cf.clip_reproject_classified_tiles(phy_id=8)
>>> cf.mosaic_tiles(phy_id=8)
>>> cf.clip_mosaic(phy_id=8)
```

OR

```
>>> import canopy_foss as cf
>>> cf. create_canopy_dataset(phy_id=8)
```



MCCAYSVILLE BASIN

⬛ NON-CANOPY

🟩 CANOPY

Source Code Available Here

Dependencies

- GDAL /OGR
- Rindcalc
- NumPy
- Scikit-Learn

Process

Join NAIP Quarter Quad (QQ) shapefile with Physiographic District shapefile

↓

Create & test training data with training.py

↓

Configure data paths in config.py

↓

Batch create ARVI rasters for physiographic district

$$ARVI = \frac{(NIR - (2 * Red) + Blue)}{(NIR + (2 * Red) + Blue)}$$

↓

Classify with Extra Trees classifier

↓

Reproject to specified projection & clip each tile to its QQ to remove overlap

↓

Mosaic tiles

↓

Clip mosaiced tiles to outline of physiographic district

Results

The created process is exponentially faster than proprietary software such as Textron's Feature Analyst while still maintaining a high degree of accuracy. CanoPy FOSS's total computation time for the largest physiographic district in GA is ~ 14 ½ hours while Textron's takes ~ 36 hours just to classify.

Data provided by the USDA Aerial Photography Program