Guide: Using PGC GitHub: mosaic
URL: https://www.pgc.umn.edu/guides/pgc-coding-and-utilities/using-pgc-github-mosaic/
Last Modified: December 17, 2019
Export Date: August 4, 2024

Introduction

In this guide, you will learn what software you need to run the pgc mosaic script, where to access the required software, and how to use the script with a sample workflow.

The PGC hosts several open source codes on GitHub, a company that hosts software development. Projects, source codes, changes and versions can be accessed in PGC's Repositories online.

Quick Links

- Polar Geospatial Center GitHub Homepage
- Polar Geospatial Center GitHub Imagery Utilities
- Mac and Linux: Installation of PGC's GDAL Stack
- Windows: OSGeo4W Installation

About

A mosaic is combination of two or more images. The Polar Geospatial Center has developed a Python script that mosaics multiple input images into a set of non-overlapping output tile images. It can sort the images according to several factors including: cloud cover, sun elevation angle, off-nadir angle, probability of overexposure, and proximity to a specific date. This tool assumes that images have been orthorectified. All preprocessing should be completed before mosaicking.

Requirements

The pgc_mosaic.py tool can run within a personal computing environment (desktop computer) or in a cluster computing environment. The code is built to run primarily on a Linux HPC cluster running Maui/Torque for queue management. This tool will also work on a windows platform.

> Please note that the code is tightly coupled to the systems on which it was developed. You should have no expectation of it running on another system without some patching.

Software:

This tool is built on the GDAL/OGR image processing API using Python. GDAL 2.1 is required for this tool to function.

If you are using a Linux system you will need to download the PGC optimized GDAL toolchain. The list of software installed with the optimized GDAL toolchain can be found here. A script is provided to install all required packages. If you have not ran a shell script in a Linux terminal follow this guide here. After installation of the PGC optimized GDAL toolchain, the mosaic tool can be ran through the Linux terminal. There are plenty of free, online tutorials for Linux terminal if you are new to command line interfaces.

If you are using a Windows system, it is recommended that you use OSGeo4W. This will provide a Windows

environment to use the tool. You can get the installers here. The express installation will provide the most high profile OSGeo4W packages. However, it will not allow for control over install location, proxies, and cache directory selection. The advanced install will allow for more control. The PGC mosaic tool will run with either install type. After installation of OSGeo4W, the mosaic tool can be ran through the OSGeo4W Shell. As with Linux, there are numerous online resources for using a Windows command line interface.

Script Details

The pgc_mosaic utility runs batch image mosaicking or submits them to a PBS or SLURM HPC cluster for processing. This utility contains three scripts:

1. pgc_mosaic.py – initializes the output mosaic, creates cutlines, and run the subtile processes.
2. pgc_mosaic_query_index.py – takes mosaic parameters and a shapefile index and determines which images will contribute to the resulting mosaic. The resulting list can be used to reduce the number of images that are run through the orthorectification script to those that will be eventually used.
3. pgc_mosaic_build_tile.py – builds an individual mosaic tile. This script is invoked by pgc_mosaic.

Instruction on how to only run parts of the toolset are included in the sample workflow below.

The mosaic toolset uses a scoring algorithm to determine which images to use in the mosaic. The four variables used are: cloud cover, sun elevation, off nadir angle, and distance from the target day of the year. Each variable is normalized to a value between 0 and 1 and multiplied by a weighting factor relative to its importance. The scoring algorithm creates a suitability score on which images are selected. Images with greater than 50% cloud cover are automatically excluded. Variable value preferences include:

- Cloud cover – lower is preferred.
- Sun elevation – higher is preferred.
- Off nadir angle – lower is preferred.
- Distance from target day of the year/target year – smaller is preferred.

Users also have the ability to exclude images with high exposure settings (specific commands shown in sample workflow). This is useful in Antarctica where the albedo can render images useless. Overexposure is defined as:

$$(TDI / \text{Sun Elevation}) > \text{Sensor – band Threshold}$$

For panchromatic images, the pan TDI is used. For multispectral images, the green TDI is used. The sensor-band thresholds were derived empirically and are as such:

Sensor	Band	Threshold Value
WorldView-1	Pan	1400
WorldView-2	Pan	1400
WorldView-2	Green	400
WorldView-3	Pan	1400
WorldView-3	Green	400
QuickBird	Pan	500
QuickBird	Green	25

Sensor	Band	Threshold Value
GeoEye-1	Pan	None
GeoEye-1	Green	170

Submission scripts to PBS and SLURM can be found at the PGC GitHub page. Including the command **“- – pbs”** will submit the task to PBS, and including the command **“- -slurm”** will submit the task to SLURM. When submitting a job to a cluster where there is storage local to the processing node it is recommended to include the **“- -wd”** command. This will allow you to set a local working directory which will allow for increased processing time.

A description of the commands can be found here. Information regarding common commands are detailed in the sample workflow below. Additionally, the script description can be viewed in a command terminal by using the **“-h”** or **“- -help”** command. For example:
`C:\python user\pathname\pgc_mosaic.py -h`

Sample Workflow

Before you begin you will need to gather all your image files and place them in a single folder. Note for these commands that Linux users will have forward slashes (/) in the pathname, and pathnames will begin with “/mnt/” instead of using the drive name as in Windows (D:\).

1. Once you have gathered all image files, you will need to open either the OSGeo4W shell if you are using Windows, or your Linux terminal if you are using Linux. Once you have opened the window, type “python” followed by the pathname for the pgc_mosaic.py script.
   `C:\>python user\pathname\pgc_mosaic.py`

   This will tell the computer to use Python to run the script, which is found in the location you specified. Dragging and dropping the pgc_mosaic.py file into the terminal will automatically populate the file pathname.

2. Next we will specify the output pixel resolution. The default is the same resolution as the input file.
   To indicate the projection we can either use **“-r”** or **“- -resolution”** followed by the resolution in the x and y directions. For example:

   To keep the default cell resolution:
   `c:\>python user\pathname\pgc_mosaic.py`

   To change the resolution to 5m x 5m pixels:
   `C:\>python user\pathname\pgc_mosaic.py -r 5.0 5.0`

3. Now we will specify the extent of the output mosaic. The default is the union of all inputs. To state which DEM to use, we can use the **“-e”** or **“- -extent”** flag followed by the minimum x value, maximum x value, minimum y value, and maximum y value. The units for the extent are determined by the coordinate system. For example:
   To set output extent to union of all inputs:
   `C:\>python user\pathname\pgc_mosaic.py -r 5.0 5.0`

   To set output extent to some extent based on the input coordinate system:
   `C:\>python user\pathname\pgc_mosaic.py -r 5.0 5.0 -e 10 1500 0 100>/code>`

4. The next step is to designate the tile size used in the mosaic. The default tile size is 40,000 times the output resolution. The units for tile size are determined by the coordinate system. To indicate the tile size we can use the **“-t”** or **“- -tilesize”** flag followed by the size in the x direction and size in the y direction. For example:
   To set the output tile size to 20,000m x 20,000m with a coordinate reference system of EPSG 3031:
   ```
   C:\>python user\pathname\pgc_mosaic.py -r 5.0 5.0 -e 10 1500 0 100 --tilesize 20000 20000
   ```

5. The next command that you may include is specifying the number of bands to include in the mosaic. The default number is the same as the first image. However, reducing the number of bands can reduce storage space. This command is activated including the **“-b”** or **“- -bands”** command followed by the number you wish to include. For example:
   To specify the use of only the first four bands:
   ```
   c:\>python user\pathname\pgc_mosaic.py -r 5.0 5.0 -e 10 1500 0 100 --tilesize 20000 20000 -b 4
   ```

6. Now we will designate the time of year to use as the target date for image suitability ranking. To do this you use the **“- -tday”** command followed by the month and day of as **“mm-dd”**. This is not a required command. It can be used as:
   To specify target date as April 5th:
   ```
   C:\>python user\pathname\pgc_mosaic.py -r 5.0 5.0 -e 10 1500 0 100 --tilesize 20000 20000 --tday 04-05
   ```

   You can also set the target year to use. Use the **“- -tyear”** command followed by the year. For example:
   ```
   C:\>python user\pathname\pgc_mosaic.py -r 5.0 5.0 -e 10 1500 0 100 --tilesize 20000 20000 --tyear 2018
   ```

7. As mentioned above, high albedo in Antarctica may render images useless. If you are mosaicking imagery over Antarctica, it may be beneficial to use exposure settings in the metadata to inform the suitability score. This is done by invoking the **“- -use-exposure”** command. For example:
   ```
   C:\>python user\pathname\pgc_mosaic.py -r 5.0 5.0 -e 10 1500 0 100 --tilesize 20000 20000 --tday 04-05 --use-exposure
   ```

8. As mentioned in the script details, you are able to specify which of the steps of the utility you would like to run. This may be helpful if you are looking to see which images you will need to orthorectify before running the mosaic utility. Checking which images will be used may reduce overall processing time by reducing the amount of images that need to be orthorectified. This command is used by including **“- -mode”** followed by the mode type. The four mode types include:

   ALL – all steps are completed (this is the default setting)

   SHP – create shapefiles (creates list of images used in the mosaic

   MOSAIC – creates tiled TIFs

   TEST – creates log file only

   These can be incorporated by:
   Using all steps:
   ```
   C:\>python user\pathname\pgc_mosaic.py -r 5.0 5.0 -e 10 1500 0 100 --tilesize
   ```

```
20000 20000 --tday 04-05 --use-exposure
```

Creating a shapefile only:

```
C:\>python user\pathname\pgc_mosaic.py -r 5.0 5.0 -e 10 1500 0 100 --tilesize
20000 20000 --tday 04-05 --use-exposure --mode SHP
```

Creating tiled TIFs:

```
C:\>python user\pathname\pgc_mosaic.py -r 5.0 5.0 -e 10 1500 0 100 --tilesize
20000 20000 --tday 04-05 --use-exposure --mode MOSAIC
```

Creating test logs only:

```
C:\>python user\pathname\pgc_mosaic.py -r 5.0 5.0 -e 10 1500 0 100 --tilesize
20000 20000 --tday 04-05 --use-exposure --mode TEST
```

9. The last step in setting up the pgc_mosaic tool is specifying the location of the input rasters and the desired output mosaic name. Type the pathname to the folder containing the raster files followed by the desired by the desired output name. You can drag and drop the folders into the terminal to have their pathname's appear. Lastly, make sure that the proper syntax is used (like spaces between commands). The example below lists commands with proper syntax:

   ```
   C:\>python user\pathname\pgc_mosaic.py -r 5.0 5.0 -e 10 1500 0 100 --tilesize
   20000 20000 --tday 04-05 --use-exposure --mode TEST
   user\pathname\ionput_directory output_mosaic_name
   ```

   To run the tool, press the Enter key. It is important to determine the exact commands you will need before running the tool. You will never need to use all available commands together.

Additional Resources

Mosaic Explained (ESRI):

http://desktop.arcgis.com/en/arcmap/10.3/manage-data/raster-and-images/what-is-a-mosaic.htm

Example of Polar Mosaics:

https://www.ospo.noaa.gov/Products/imagery/mosaics.html