Guide: Using PGC GitHub: pansharpening
URL: https://www.pgc.umn.edu/guides/pgc-coding-and-utilities/using-pgc-github-pansharpening/
Last Modified: October 4, 2023
Export Date: April 25, 2024

Introduction

In this guide, you will learn what software you need to run the pgc pansharpen 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

High-resolution satellites, like those operated by Maxar, provide two types of bands: multispectral and panchromatic. The multispectral bands consist of ranges of wavelengths in the electromagnetic spectrum (i.e. red, blue, green, etc.), and these are subsequently called the red band, blue band, and so forth. The panchromatic band is a single band that combines reflectance across the different color bands, hence "pan" chromatic. Panchromatic bands have higher spatial resolutions than multispectral bands because the broad spectral range permits sensors to obtain a high signal to noise ratio. However, panchromatic bands lack the ability to be displayed in true color. Pansharpening is the process in which the panchromatic band is used to increase the spatial resolution of multispectral bands, which allows for higher resolution to true or false color images.

> Please note that the Python script to batch pansharpen satellite imagery developed by the Polar Geospatial Center will also automatically orthorectify the imagery. It is important to note that pansharpening will change the digital numbers for each pixel. **This means that pansharpening should only be used to improve resolution for visual interpretation**.

Requirements

The pgc_pansharpen.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 pansharpening 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 pansharpen tool will run with either install type. After installation of OSGeo4W, the pansharpening 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_pansharpen utility runs batch image orthorectification, conversion, and pansharpening or submits them to a PBS or SLURM HPC cluster for processing. First, the utility applies the orthorectification process to both the panchromatic and multispectral image in a pair, and then pansharpens them using the GDAL tool gdal_pansharpen. 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.

Sample Workflow

Before you begin you will need to gather all your NTF 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 the NTF 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_pansharpen.py script.`C:\>python user\pathname\pgc_pansharpen.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_pansharpen.py file into the terminal will automatically populate the file pathname.

2. Next we will specify the output coordinate reference system. Geodetic Parameter Datasets (EPSGs) are a codified list of projections and coordinate reference systems. By selecting a specific EPSG code, an appropriate projection can be selected for the AOI the imagery lies within. It is very important to use an appropriate coordinate system, especially when working close to the poles. For example, when using the WGS84 geographic coordinate system, Antarctica and Greenland appear much larger than they are in reality. By using a projection specifically designed for polar regions, this distortion can be limited, and the imagery can be displayed more accurately. Most commonly used EPSG codes at PGC include:

3031- Antarctic Polar Stereographic
3413- Arctic Polar Stereographic
3338- Alaska Conformal Conic

To indicate the projection we can either use **“-p”** or **“- -epsg=”** followed by the EPSG number.

```
C:\>python user\pathname\pgc_pansharpen.py -p 3031
```

Or

```
C:\>python user\pathname\pgc_pansharpen.py --epsg=3031
```

3. Now we will specify the digital elevation model (DEM) to use to adjust the imagery. A DEM is a raster surface that is representative of the earth’s surface. Each pixel contains an assigned value that represents the elevation of the pixels location. The DEM is key in adjusting for errors in imagery due to elevation as extreme changes in elevation decrease the accuracy of the imagery. A DEM is not required, and if it is not specified, it will use an average elevation as specified from the image metadata. This is usually effective in areas near sea level with minimal elevation change. However, the use of a DEM is recommended.To state which DEM to use, we can use the **“-d”** or **“- -dem=”** flag followed by the pathname to DEM file.

   ```
   C:\>python user\pathname\pgc_pansharpen.py -p 3031 -d C:\file\pathname\DEM.tif
   ```

   Or

   ```
   C:\>python user\pathname\pgc_pansharpen.py -p 3031 --dem=C:\file\pathname\DEM.tif
   ```

4. The next step is to identify the output bit depth. The available bit depths are 8, 16, and 32. A larger bit depth means that a larger array of possible color values for each pixel can be represented. With 16 bit imagery, the possible number of values is 65,536, with 8 bit, it is simplified to 256. Usually, if the imagery is being used for simple visual analysis, 8 bit will suffice, and no input is needed in the script. If more complex analyses are being performed on the imagery, 16 bit or 32 bit float may be desired. Note that 16 bit imagery takes up significantly more disk space than 8 bit, and 32 bit takes up even more. While 8 bit creates a product similar to the NTF size, 16/32 bit will often create a product larger than the original NTF, and approximately twice the size of what 8 bit imagery would produce. Available options for this command include:

   UInt16 – 16 bit output
   Float32 – 32 bit floating point output
   No command is required for 8 bit output.

   To designate an output bit depth we can use the **“-t”** or **“- -outtype=”** flag followed by one of the previously mentioned command options. For example:

   For 8 bit output: No Command Needed

   ```
   C:\>python user\pathname\pgc_pansharpen.py -p 3031 -d C:\file\pathname\DEM.tif
   ```

   For 16 bit output:

   ```
   C:\>python user\pathname\pgc_pansharpen.py -p 3031 -d C:\file\pathname\DEM.tif -t UInt16
   ```

   For 32 bit output:

   ```
   C:\>python user\pathname\pgc_pansharpen.py -p 3031 -d C:\file\pathname\DEM.tif -t Float32
   ```

5. As mentioned above, the pgc_pansharpen tool can also correct radiometric settings. Some images will

appear brighter or darker than others due to variation in acquisition time, atmospheric differences, and other image differences. These differences become apparent between scenes and make visual analysis difficult. When using 8 bit images, multiple values from the original image are aggregated into a single value. In many images, these values tend to cluster in a specific portion of the possible spectrum of values. A pixel stretch will stretch these values out across all 256 available values. This ensures similar quality between different scenes. For 16 bit imagery, the maximum number of values are already used, and usually no stretch is necessary.The three most commonly used stretches include:

No Stretch (**ns**): Scales the DN values to the output data type. Usually used when a 16 bit output is desired.

Reflectance (**rf**): Calculates top of atmosphere reflectance and scales it to the output data type. This option is optimized for snow-covered images.

Modified Reflectance (**mr**): Same as reflectance, but with a histogram stretch applied that brightens the lower end of the dynamic range. This option is optimized for areas with less snow cover (i.e. more diverse land covers).

To indicate the projection we can either use **"-c"** or **"- -stretch="** followed by one of the command options listed above. For example:

For No Stretch:

```
C:\>python user\pathname\pgc_pansharpen.py -p 3031 -d C:\file\pathname\DEM.tif
-c ns
```

For Reflectance Stretch:

```
C:\>python user\pathname\pgc_pansharpen.py -p 3031 -d C:\file\pathname\DEM.tif
-c rf
```

For Modified Reflectance:

```
C:\>python user\pathname\pgc_pansharpen.py -p 3031 -d C:\file\pathname\DEM.tif
-c mr
```

6. Now we will specify the output file format. Different computer programs and processing steps may require different file formats. Check which format you require. If no command is input the default format is GeoTIFF. If you are using the imagery in ArcMap, ArcGIS Pro, or QGIS the default GeoTIFF setting will be fine. Other available options include:

   HFA File Format (**HFA**): This is a file format specific to ERDAS IMAGINE (.img format).

   ENVI File Format (**ENVI**): This is specific to Exelis Visual Information Solution (ENVI). Binary file with .envi extension.

   GeoTIFF File Format (**GTiff**): This type of file is generally uniform for multiple platforms.

   JPEG 2000 File Format (**JP2OpenJPEG**): Lossless JPEG2000 format using OpenJPEG2 driver. JPEG2000 files are also generally accepted file types for multiple platforms.

   To specify the file format we can either use **"-f"** or **"- -format="** followed by one of the command options listed above. For example:

   For HFA:

```
C:\>python user\pathname\pgc_pansharpen.py -p 3031 -d C:\file\pathname\DEM.tif
-c mr -f HFA
```

For ENVI:

```
C:\>python user\pathname\pgc_pansharpen.py -p 3031 -d C:\file\pathname\DEM.tif
-c mr -f ENVI
```

For GeoTiff: No Command Needed

```
C:\>python user\pathname\pgc_pansharpen.py -p 3031 -d C:\file\pathname\DEM.tif
-c mr
```

7. This next command is rarely needed. The Resample command denotes how cells should receive values. The default option is the nearest neighbor sampling method. This method assigns the nearest unorthorectified pixel value to the correct orthorectified pixel. This leaves the initial pixel values unaltered, but occasional errors may lead to an inappropriate pixel assignment. Information on resampling methods can be found here. Available command options include:

   Nearest Neighbor (**near**): Resamples using the nearest neighbor method

   Bilinear Interpolation (**bilinear**): Resamples using the bilinear interpolation method

   Cubic Convolution (**cubic**): Resamples using the cubic convolution method

   Cubic Spline (**cubicspline**): Resamples using the cubic spline method

   Lanczos Resampling (**lanczos**): Resamples using the Lanczos resampling method

   To indicate the resampling method we use **"- -resample="** followed by one of the command options listed above. For Example:

   For Nearest Neighbor: No Command Needed

   ```
   C:\>python user\pathname\pgc_pansharpen.py -p 3031 -d C:\file\pathname\DEM.tif
   -c mr
   ```

   For Bilinear Interpolation:

   ```
   C:\>python user\pathname\pgc_pansharpen.py -p 3031 -d C:\file\pathname\DEM.tif
   -c mr --resample=bilinear
   ```

   For Cubic Convolution:

   ```
   C:\>python user\pathname\pgc_pansharpen.py -p 3031 -d C:\file\pathname\DEM.tif
   -c mr --resample=cubic
   ```

8. Another optional command is **"- -rgb".** This command removes all bands besides the red, green, and blue bands. This is done to create a smaller file, and should be used if you are only looking to display the true color image. All other bands (coastal blue, yellow, NIR, etc.) will not be kept. To specify that only the red, green, and blue bands should be kept, use the **"- -rgb"** command. For Example:

   ```
   C:\>python user\pathname\pgc_pansharpen.py -p 3031 -d C:\file\pathname\DEM.tif
   -c mr --resample=cubic --rgb
   ```

9. The last step in setting up the pgc_pansharpen tool is specifying the location of the NTF files and the desired output location. Type the pathname to the folder containing the NTF files followed by the pathname to the desired output folder. 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_pansharpen.py -p 3031 -d C:\file\pathname\DEM.tif -c mr --resample=cubic --rgb D:\input\Imagery\NTF D:\output\folder\pansharpened
```

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

Pansharpening Explained:

https://en.wikipedia.org/wiki/Pansharpened_image

Bit Depth Explained:

https://apollomapping.com/2012/September/article8.html

Resampling Strategies:

http://www.extension.org/pages/9628/remote-sensing-resampling-methods#.U5iyjvldV8E
http://www.ldv.ei.tum.de/uploads/media/Vorlesung_3.4_Resampling.pdf
http://en.wikipedia.org/wiki/Lanczos_resampling

DEM General Overview:

http://en.wikipedia.org/wiki/Digital_elevation_model

RAMP DEM Information and Download:

http://nsidc.org/data/nsidc-0082

AlaskaNED Information:

http://ned.usgs.gov/

SRTM Information and Download:

http://www2.jpl.nasa.gov/srtm/

ERDAS IMAGINE Website:

http://www.hexagongeospatial.com/products/ERDAS-IMAGINE/details.aspx