The Image Downloader Project

GEOM 4009 Team Report

Philip Ishola, Shea Timmins, Zacharie Sauvé, Collin Godsell

GEOM 4009 - April 012, 2024

Introduction

This project was initiated under the Polar Continental Shelf Program (PCSP), a division of Natural Resources Canada that has been operating in the Arctic since 1958. The client, Tanya Lemieux, provided us with an overview of PCSP's objectives, emphasizing their commitment to supporting scientific research in the Arctic. Their operations include coordinating logistical support across various research projects to enhance scientific knowledge of the region and contribute to Canada's sovereignty over it. Given the challenging weather conditions in the Arctic, PCSP sought a technological solution to improve operational efficiency and safety.

The core requirement from the client was the development of the Image Downloader Project. This project aimed at creating a tool for downloading satellite imagery to aid in the planning and coordination of logistical operations in the Arctic. Specifically, the client required a system capable of automatically fetching near-real-time satellite images over designated regions of interest. This tool was envisaged as a component of a larger initiative to consolidate PCSP's data into a singular interactive platform, thereby facilitating easier access and manipulation of data for planning purposes.

The purpose of our project was to design, develop, and test a Python-based script that could meet the client's needs. This script was intended to download satellite images, convert them into a usable format, georeference the images, and then save them in a KMZ file format compatible with Google Earth Pro. Moreover, it was essential that this script could operate automatically at predetermined intervals, integrating seamlessly with tools such as Windows Scheduler to ensure consistent data retrieval without manual intervention.

The scope of our work encompassed the initial design and development phases, including interfacing with NOAA’s Web Services API and NASA’s Earth Data API to access near-real-time satellite imagery. The development process was iterative, involving the construction of a proof-of-concept tool that demonstrated the feasibility of automating satellite image collection for PCSP’s operational needs.

Throughout this project, we engaged with various data types and formats. Initially, we focused on handling High-Definition File (HDF) formats and converting these into GeoTIFF files for easier manipulation. Our work also included the integration of common Python libraries such as subprocess, numpy, and rasterio along with wget GNU for the purpose of accessing APIs, handling images, and performing geospatial operations.

This report presents a comprehensive overview of our project from inception through to completion, detailing the methodologies adopted, the challenges encountered, and the outcomes achieved. Through our collaborative efforts, we have developed a tool that not only meets the immediate needs of PCSP but also lays the groundwork for future enhancements and expansions of their data consolidation and access initiatives.

Documentation

Dependencies:

The Image Downloader Tool is dependent on the Image downloader environment file that is provided on GitHub. The environment file contains a few packages that the tool requires when running, that must be included in your devices conda environment. The packages/software required to run the tool are:

- Python>=3.8
- Numpy
- Matplotlib
- Pandas
- Fiona
- Shapely
- Geopandas
- Geoplot
- Geojson
- Rasterio
- Subprocess
- Shutil

Installation:

The hardware required to use the tool is a computer with, preferably, a Windows operating system as the tool was designed and implemented in Windows. A Linux operating system may run the script but a few modifications to the script are required due to operating system syntax. A 'README' file is provided for context on the installation process, as well as configuration. The tool makes use of a few exterior tools/softwares/documents such as the HEG tool and Task Scheduler, as well a configuration file and a parameter file. Included with the tool are a few setup demonstrations for modifying the configuration file, setting up the task scheduler and installing/setting up the HEG tool. The demonstrations provide a working knowledge of how to set up these workspaces for using the ImageDownloader tool, for more context on the individual use of the HEG tool, please see the HEG documentation website.

Troubleshooting/FAQs:

Here are some common errors that might occur with the current script.

1. **Why is the Metadata file list empty?**

   The program will not be able to run as there is no new metadata file uploaded yet on the LANCE website for that day. Be patient as they will upload it at a certain time.

2. **The tool finished running but why is the kmz file empty?**

   This could be due to the LANCE website not uploading all the necessary bands to create a true colour image. Also check that all file paths have no spaces in the names.

3. **Why is the kmz black?**

   If the whole screen is black this might be due to the image being taken at night. If part of the screen is black this is due to the satellite only passing by a section of your area of interest and not the whole area. This can be seen by a smooth curve along the image.

4. **Why is there not a new image every 5 minutes?**

   The satellite takes a new image every 5 minutes, but if it is not within the area of interest the program does not recognize it as a new image so it will run the newest image that falls within your area of interest.

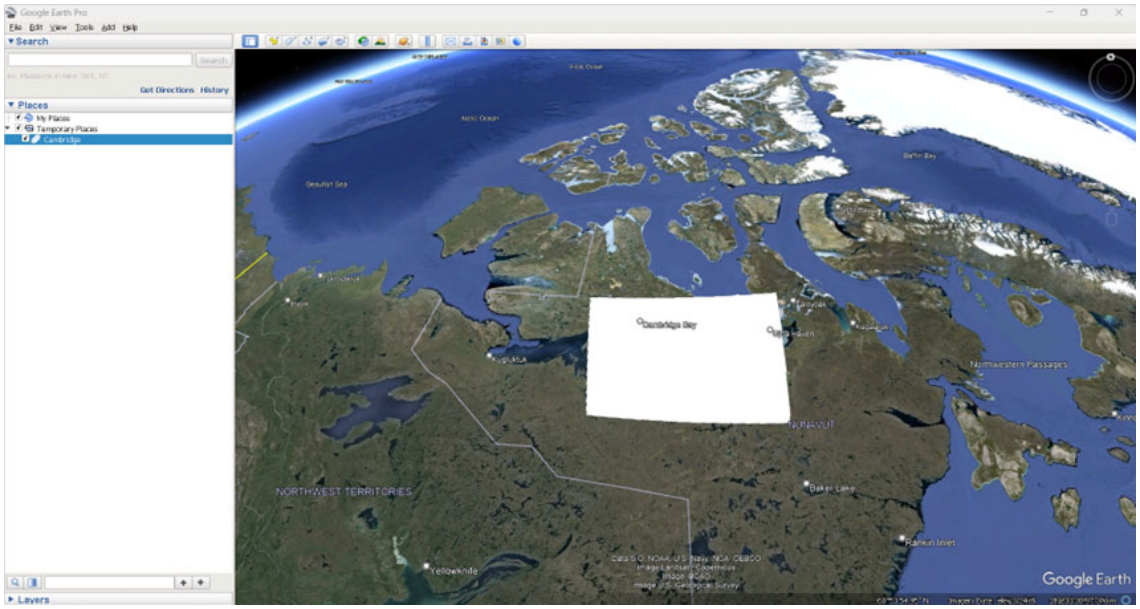
Please reach out via the Image-Downloader-Project GitHub page under the 'Issues' tab for any troubleshooting needs or frequently asked questions. GitHub provides a comprehensive forum for project contributors and users to communicate/share ideas on troubleshooting/FAQs which allows other internet users to communicate/resolve issues in the future.

Sphinx Output:

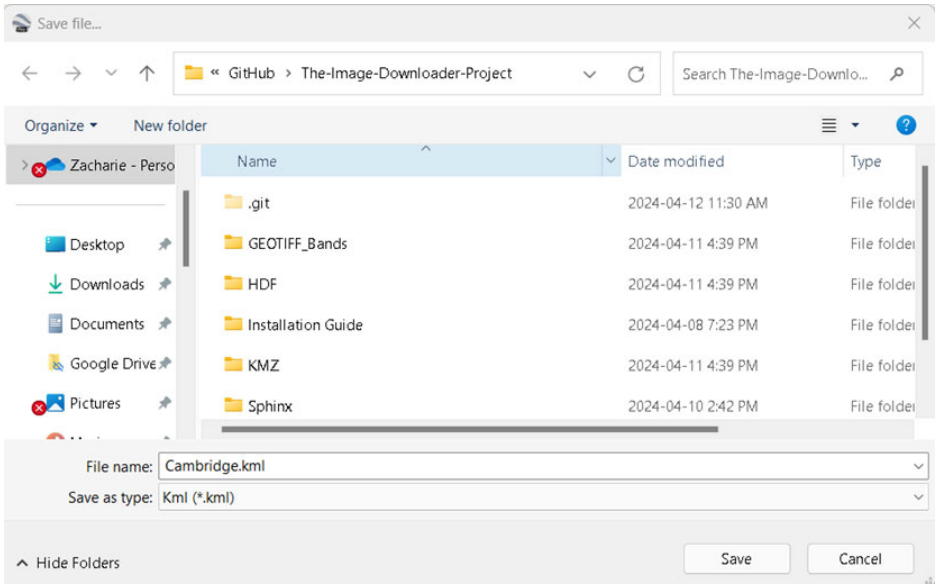
An html called **ImageDownloaderProject.html** that has auto-generated documentation from the source code which outlines all of the docstrings provided for each function the scripts employs. This can be found in the GitHub repository.

User example:

1. After installing the imagedownloader_env.yml, and the HegTool (see GitHub Installation Guide folder). Open up Google Earth and create a polygon with the 'Add Polygon' tool. Select the four corners of your area of interest that you would like.



2. Then save the kml into the Image Downloader Project folder.



3. Open the config.cfg in a text editor and fill in all the necessary components. Make sure all paths do not have spaces in the names as this will cause the HegTool not to work.

```
# Config file for ImageDownloaderProject.py

[Paths]
# add the file path for the parameter file location to run HEGTool
parameter_file = C:/Users/zachs/Documents/GitHub/The-Image-Downloader-Project/image_project_swath.prm

# add the file path for the folder you want the 3 output TIFF Bands to go
GeoTIFF_folder = C:/Users/zachs/Documents/GitHub/The-Image-Downloader-Project/GEOTIFF_Bands

# add the file path for the folder you want the merged TIFF file to go
TIFF_Final = C:/Users/zachs/Documents/GitHub/The-Image-Downloader-Project/TIFF_Final

# add the file path for the gdal_translate.exe which can be found in the /..../.../anaconda/envs/imagedownloader_env/bin/gdal_translate.exe
gdal_translate_path = C:/Users/zachs/anaconda3/envs/GEOM4009/Library/bin/gdal_translate.exe

# add the file path for the folder you want the KMZ to go
kmz_folder = C:/Users/zachs/Documents/GitHub/The-Image-Downloader-Project/KMZ

[Names]
#Add the names for output that you want for the three TIFF bands for the HegTool
base_filenames =
    MODIS_SWATH_TYPE_L2_Band1.tif
    MODIS_SWATH_TYPE_L2_Band4.tif
    MODIS_SWATH_TYPE_L2_Band3.tif


[HegTool]
# Set the working directory for the HegTool /..../.../HEGTool/HEG_Win/bin/
HEGTool_directory = C:/Users/zachs/Desktop/HEGTool/HEG_Win/bin/

# Set the environment variables for the HegTool
# Set this to point to /..../.../HEGTool/HEG_Win/bin/
MRTBINDIR = C:/Users/zachs/Desktop/HEGTool/HEG_Win/bin/

# Set this to point to /..../.../HEGTool/HEG_Win/TOOLKIT_MTD/
PGSHOME = C:/Users/zachs/Desktop/HEGTool/HEG_Win/TOOLKIT_MTD/

# Set this to point to /..../.../HEGTool/HEG_Win/data/
MRTDATADIR = C:/Users/zachs/Desktop/HEGTool/HEG_Win/data/
```

```
[LANCE]
# Create an account with the NASA Earth Data, and create a authentication token with your profile
auth_token =
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlbWFpbF9hZGRyZXNzIjoiemFjaHNhdXZlMUBnbWFpbC5jb20iLCJpc3MiOiJBUFMgT0F1dGggYIEF1dGhlbnRpY2F0b3IiLCJpYXQiOjE3MTEzOTA2NzEsIm5iZiI6MTcxMTM5MDY3MSwiZXhwIjo
xODY5MDcwNjcxLCJ1aWQiOiJ6c2F1dmUiLCJ0b2tlbkhyZWF0b3I1OiJ6c2F1dmUifQ.wGxsqDm_RsqrCQy4a4K5Mred_lxU0QC54fa2AksjGlQ

# This url path to the MODIS satelite http folder request for MYD09 data. Do not modify unless changing download folder
base_HDF_url = https://nrt3.modaps.eosdis.nasa.gov/api/v2/content/archives/allData/61/MYD09/Recent

# add the file path for the folder you want the HDF file to go
download_HDF_folder = C:/Users/zachs/Documents/GitHub/The-Image-Downloader-Project/HDF

# This url path is to the MODIS aqua metadata folder containing the geolocations. Do not modify.
base_txt_url = https://nrt3.modaps.eosdis.nasa.gov/api/v2/content/archives/geoMetaMODIS/61/AQUA/

# add the file path to folder and name you want the metadata to go
metadata_file = C:/Users/zachs/Documents/GitHub/The-Image-Downloader-Project/aqua.txt


# run through with different test_times
# If you choose the empty test time it will search for the closest image UTC time
# If you choose a date and time it will search for the closest image before that set time
#test_time = 2024-04-06 07:00
#test_time = 2024-03-30 14:00
#test_time = 2024-03-30 16:00
test_time =

[BoundingBox]
# add the file path for the folder you have the AOI kml
kml_AOI_file = C:/Users/zachs/Documents/GitHub/The-Image-Downloader-Project/cambridge.kml
```

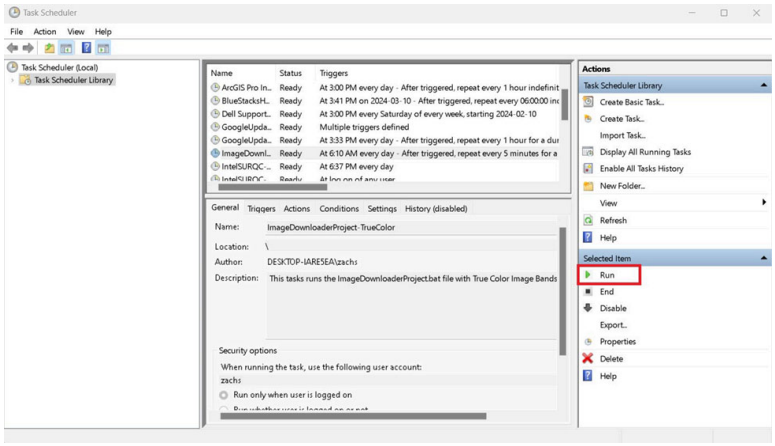
4. Open ImageDownloader.bat in a text editor, and update the python.exe environment, ImageDownloaderProject.py and config.cfg file paths.

```
@echo off
:start
"C:\Users\zachs\anaconda3\envs\imagedownloader\python.exe" "C:\Users\zachs\Documents\GitHub\The-Image-Downloader-Project
\ImageDownloaderProject.py" "C:\Users\zachs\Documents\GitHub\The-Image-Downloader-Project\config.cfg"
timeout /t 300 /nobreak >nul
goto starts
```

5. Create a task in the Task Scheduler using the instructions found in the GitHub Installation Guide folder.

6. Make sure the task is enabled (The Run and End option will show up on the right hand side under the ‘Selected Item’ drop down menu).

7. Then right-click the ImageDownloader task you created and click Run.



Workflow

The script follows a streamlined methodology which begins by downloading and reading a text metadata file that contains the Granule/Location information required to download the imagery. Once the metadata is secured, the script creates and sends a request to NASA’s LANCE service, which requests the download of the corresponding HDF. After the HDF has downloaded, it is loaded into the HEG tool for conversion into GeoTiff. Each band is stored into a separate GeoTiff file then stacked into a True-Colour (or other desired band combination) image. Upon stacking, the image is then exported as a KMZ for visualisation/use in Google Earth Pro. Figure 1, visualises the methodology of the script in simple terms.

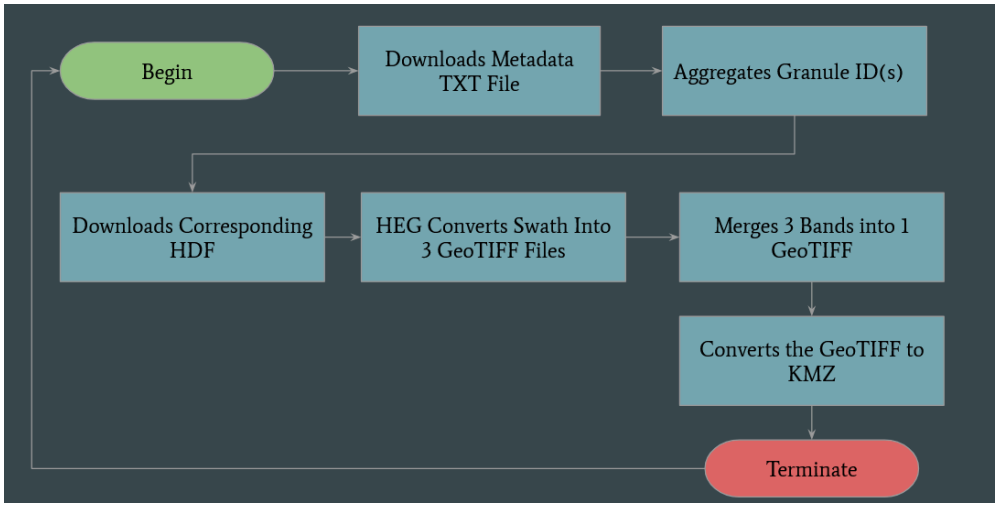


Figure 1. A flowchart outlining the raw workflow of the Image Downloader Tool.

Below are the steps formatted in a bulleted list for easier interpretation:

- Request and download metadata text file via NASA's LANCE API.
- Aggregate the Granule ID(s) based on the KML region of interest provided.
- Create a request using the metadata information to download the corresponding HDF file for the region of interest.
- Load the downloaded HDF file into HEG to convert the entire swath into three bands (stored separately in GeoTiffs).
- Stack the three bands into a single image displaying true colour. Note the band combination can be changed in the parameter file for pseudo colours, false colour composite, etc.
- Finally, output the single image as a KMZ for use in Google Earth Pro.
- The process repeats itself in the task scheduler, at specified intervals, until ceased.

Discussion

Challenges:

One of the challenges we encountered involved interacting with the Global Imagery Browse Services (GIBS) API to fetch NRT satellite imagery. The central challenge stemmed from the API's processing of time formats for NRT data requests. Despite our strict compliance with the documented standards, the API consistently rejected our submissions due to the specified time format not being recognized as valid. This issue was pivotal, given the project's reliance on timely and accurate satellite data to inform logistical decisions in the Arctic's challenging environment.

The project necessitated converting Hierarchical Data Format (HDF) formats into GeoTIFFs, a process vital for the subsequent georeferencing and KMZ conversion steps. Leveraging the HegTool for these conversions introduced complexities, as it required processing individual bands separately and later stacking them into a cohesive image. This multi-step process demanded a deep dive into command-line automation within Python, meticulous parameter file manipulation, and ensuring the integrity of the merged GeoTIFF outputs.

Limitations:

The most prevalent limitation that was encountered is that the script is totally dependent on the LANCE MODIS online database. In addition to the LANCE limitation, another prevalent issue is with the study location. The PCSP is interested in Arctic regions, but requires the viewing to be Google Earth, which uses coordinate latitude and longitude (Mercator Projection) to display the imagery. Since the imagery is displayed without the Arctic/Polar Stereographic Projection (or another relevant Arctic projection), the imagery is warped/confused by the presence of the pole. This results in an area of interest that wraps around the entire globe as either the longitude or the latitude coordinates cross over the pole.

Additionally, the script could be modified to not only request imagery from the AQUA satellite, but also from the accompanying TERRA satellite for better data availability and coverage. This was one major limitation of the current release of the Image Downloader Tool, as it lacks full data coverage potential without the second satellite. Finally, the output image is not a smooth image and composed of varying resolutions. Therefore, addressing the resolution of the image is a priority for accurate interpretation.

Future Work:

For future implementations of the Image Downloader Tool, it is suggested that the aggregated swath (for Arctic regions of interest namely) should be projected into a projection such as EPSG:3995 (Polar Stereographic Projection), before it is fed to the HEG tool. This would ensure that the specified region of interest is preserved without the excessive no data values. The result could then be returned to the Mercator Projection for use in Google Earth.

Furthermore, the script can be modified (following the same procedures for acquiring AQUA imagery) to acquire TERRA imagery for the region of interest as well. This would further the coverage area, and also allow users to choose between satellite products or acquire both. Additionally, LANCE offers other NRT data from alternative satellite instruments such as CALIPSO, VIIRS or MISR (which is useful for clouds). These products could also be accessed in a similar fashion to MODIS AQUA, and could be used in tandem with the MODIS data to provide a more accurate assessment on weather conditions.

Another paramount modification that must be made to the tool in future implementations is smoothing the resulting imagery with one single resolution. A proposed method for smoothing the resulting image is to use GDAL Warp, with either a cubic or nearest neighbour resampling method to smooth the image to a single resolution.

Additionally, adding a few ‘quality of life’ checks to the script would be pertinent. For example, checking to ensure that the HDF file that is being requested for download contains all the correct band information before passing it to the HEG tool. Another check that could improve precision, is ensuring that at least 30% of the region of interest is covered. Using a conditional statement, the spatial condition could be checked based on if the imagery is within 30% or less of the area covered by the region of interest, it could be downloaded along with the prior HDF for contrast. Finally, an input section could be added to specify which bands are required to comprise the image, allowing for alternate types of imagery aside from strictly true colour.

Employing an agile scrum methodology, we navigated through task allocation, progress tracking, and iterative development, emphasising open communication and regular updates through Discord and team meetings. We did not have any interactions with the client outside of the initial meeting where the project was introduced to us and our presentation of the product. This was due to the fact that we did not have a completed working product until close to the end project date. A majority of the decisions of what satellite imagery to download and how to process that imagery into a KML(Z) version was left up to us to decide.

Conclusion

To conclude, the Image Downloader Tool was designed to acquire NRT satellite imagery of a specified region of interest (KML) for use by the PCSP in planning and logistics. The script delivered offers a working foundation/proof of concept tool that could be expanded to meet the client's end vision of a unified platform for operations. The report outlines the workflow and processes taken to develop the first release, as well as address limitations and suggestions for future development. The project has been released publicly on GitHub for PCSP and other users to improve upon and make use of.

Acknowledgements

Many thanks to professor Derek Mueller for all his help with the coding and the direction he gave us in order to complete this project.

We would also like to acknowledge MODAPS Support over at NASA for their help in navigating their online MODIS database.

Sources

Access Basics - Global Imagery Browse Services (GIBS). (n.d.).

https://nasa-gibs.github.io/gibs-api-docs/access-basics/

Center, H. T. a. I. (n.d.). *HDF-EOS to GeoTIFF Conversion Tool (HEG)*.

https://www.hdfeos.org/software/heg.php

NASA, & Thome, K. T., [610WebDev]. (2024, March 2). *MODIS | Terra* (N. R. Rayne, Ed.).

TERRA - The EOS Flagship. Retrieved February 19, 2024, from

https://terra.nasa.gov/about/terra-instruments/modis

Modifying text documents:

https://www.geeksforgeeks.org/python-program-to-replace-specific-line-in-file/

Converting the config file text document from Windows CR LF ro Unix LF line endings:

https://stackoverflow.com/questions/36422107/how-to-convert-crlf-to-lf-on-a-windows-machine-in-pyt

hon

Merging the three bands into one image:

Color — rasterio documentation

python - Scale a numpy array with from -0.1 - 0.2 to 0-255 - Stack Overflow