COCKTAIL

GDAL, OTB and QGIS in a virtual environment for resource constrained collaboration

Last update: 7 July 2022

Private sector, high-resolution, daily-updated satellite assets have become a significant resource for remote sensing operations. A case in point is PlanetScope (PS), currently operating the largest collection of small Earth-imaging satellites. Cocktail facilitates the combination of free and commercial satellite assets to monitor an area of interest with a low-cost system, and then switches to a high-resolution asset only if a condition of concern or interest has been detected and not adequately understood in the first data set. Not unlike the tip and cue concept in which one monitors an area of interest with one sensor and then 'tips' another complementary sensor platform to acquire another image over the same area, Cocktail tips and cues based on economic constraints.

Cocktail is designed as a compendium to other GIS investigation environments, combines GDAL, OTB and GQIS elements, and is intended to be deployed remotely. The combination requires some attention. The Orfeo machine learning library and the translator library for raster and vector geospatial data formats GDAL do not play well with the open source geographic information system QGIS outside of a resource intensive GUI environment. The systems have differing and partially conflicting dependencies, and the GUI environment makes rapid testing across the various tool boxes cumbersome. This fact becomes apparent, for example, when developing a pipeline to perform vector-based satellite image classification with a neural network under OTB. Cocktail allows you to setup a low-cost virtual computer that can support GDAL, OTB and QGIS functionality and how to apply various classifiers including vector support machines, random forests, neural networks and various band math operations across GDAL, OTB and QGIS.

Cocktail is a remote sensing opensource utility but does not offer many of the functionalities - such as processing SAR constellation data - of the EOreader, for example. Cocktail is designed to facilitate resource constrained collaborative inquiry on optical satellite assets. The large collection of parameters set across the individual modules are saved in a single file such that collaborators can reliably replicate a workflow. Cocktail can be linked with an external cloud storage provider to move assets out of the compute environment for low-cost storage. The code repository implements a solution for this issue using pCloud. Other providers can likewise be attached.

Here is a brief description of the steps required to get started:

1 - VM
Create a virtual computer.
Choose Ubuntu 18.04 LTS, with at least 16GB of RAM

2 – Get the files from GitHub
At the prompt type: *git clone https://github.com/realtechsupport/gdal-otb-qgis-combo.git*
Enter the username: *realtechsupport*
Enter the access token: *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*

3 - Install QGIS
Make the file basics+qgis.sh executable.

chmod +x basics+qgis.sh

Run that file

sh basics+qgis.sh

4 - Install OTB in a virtual environment with conda
Make the file conda-packages.sh executable.

chmod +x conda-packages.sh

Run that file

sh conda-packages.sh

5 - Create a conda environment with defined dependencies.
(OTB 7.2 requires python 3.7, for example)
Create a conda environment with the settings in the environment.yml file.

conda env create -f environment.yml

6 - Customize
Add other libraries to the OTB environment as needed

conda install -c conda-forge pillow

(Enable the OTB environment to install geojson and geopandas.
The geopandas install may take some time…)

conda install -c conda-forge geojson
conda install -c conda-forge geopandas

7 – Setup the directory structure
The directories code, data, rasterimages, vectorfiles, collection and results should be ready made in your repository directory. Load zipped shapefile assets into the collections directory. Place image assets (.tif) in the rasterimages directory and vector assets (.shp) into the vectorfiles directory. Add the corresponding .dbf, .prj, .shx files for each .shp file.

Use the settings.txt file (in the data directory) to set your file names, process preferences and classification parameters. The content of this file is parsed and passed to the various classification routines.

8 - Test
Adjust the parameters in the file data/settings.txt to work with your data directory.

a) Test QGIS

Run the qgis test in the base environment.

python3 qgis_test.py

You should see a list of operations supported by QGIS.

b) Test OTB

Activate the OTB environment.

conda activate OTB

Run the otb test

python3 otb_test.py

You should see a list of operations supported by OTB as well as the output of the settings.txt file.

You can now use QGIS in the base installation and enable the conda environment to access OTB functionality. You can also move across environments to access libraries from either environment with python scripts as outlined below (see vector_classify_top.sh):

echo "Starting OTB-QGIS pipeline...\n\n"
conda run -n OTB python3 /home/code/otb_code_A.py
python3 /home/code/qgis_code_A.py
conda run -n OTB python3 /home/code/otb_code_B.py
python3 /home/code/qgis_code_B.py

Since the directory structure you setup will be identical in both the QGIS and OTB environments, intermediate filed produced will be available to processes running in either environment, allowing for data to be shared at no extra computational cost. All settings across the script modules are stored in the 'settings.txt' resource file and imported to the individual modules.

After installation and testing you can perform operations as such (provided you have already prepared your data accordingly.)

If you are using pCloud to move results from the VM to the pCloud server, install the library in the OTB environment with the correct python version (here 3.7).

conda activate OTB
python3.7 -m pip install pcloud

Working with Cocktail

All examples below assume that you set your input and process parameters in the settings.txt file.

Get free satellite imagery from the European Space Agency
Perform steps 1-4 once.

1) Create an account with Copernicus Open Access Hub (https://scihub.copernicus.eu/dhus)
You will need a login and pswd to access sentinel2 files
2) Generate a geojson file for area of interest (https://geojson.io/)
Save the file as sat_roi.geojson and place a copy on into your data directory.
3) Put the ESA credentials in a file in the auth folder.
4) Generate a shapefile with a specific area of interest (ROI) to clip the asset to a specific area.
Place that file in the collection directory.

Repeat this next step periodically (Sentinel2 has a combined revisit frequency of 5 days).
You can also call it via a cron job.

4) Run the script:
    python3 sentinel2_getdata.py

Enter choices at the prompt
Inputs: bandselection, now, uuid

If the second input is empty, the end date in settings.txt will be used.
You can cue on the sentinel asset and simply get the latest available summary image (TCI) to see if the data is useful.

To do this, type
    tci now

If you have setup pCloud (or someother provider) and the TCI passes a simple statistical test (see code for details), that TCI will be saved to pCloud. If that image suggests good data, get all bands with these inputs:

    all now

Alternatively
    tci

Will get the last TCI between dates set in the settings.txt file

    all

Will get all bands between dates in the settings.txt, and

    71d7edef-292c-4f4e-af40-72c3f0e9eb64 (sample)

Will get all bands of a unique identifier (uuid)

Sentinel's SAFE format with the JPEG2000 images will be converted to geotif (.tif) will be clipped with the specified region of interest specified in the settings.txt file, compressed (.zip) and saved to the collection directory (area2_0726_2021_sentinel2.zip, for example).

If you selected 'tci', and the operation was successful, you should see a message like this:

```
Image passes test. Percentage of non-black pixels: 66
Sending to pCloud..
2022-07-07 20:09:46,697 - pycloud - INFO - Using pCloud API endpoint: https://api.pcloud.com/

Uploaded:
['/home/blc/cocktail/data/rawsat/S2B_MSIL1C_20210706T021609_N0301_R003_T50LKR_20210711T143413.SAFE/GRANULE/
L1C_T50LKR_A022623_20210706T022834/IMG_DATA/T50LKR_20210706T021609_TCI_roi.jpeg']

Adding log entry...
Deleting temporary files to save storage space.
```

You should also have the reduced format .jpeg image in your cloud storage as a reference.

Depending on the area you select and your cloud coverage threshold, you might not find any good data within a given time window. Vary your settings. Also, ESA limits the data you can download. Caveat.

Once you do get a good test image, download and save the complete package by running

python3 sentinel2_getdata.py with the prompts
all now

… as mentioned above. You can use the result for band operations or classifications (see below). But the resolution of the sentinel data may be insufficient – in that case you would seek an alternate source such as Planet Labs. Finding viable free data (tip) can will reduce the number to times you grab non-free higher resolution data from a private provider (cue). Use the same ROI file and the date of the downloaded TCI to get data only on dates where you area is clearly visible and accessible. That is how the tip and cue concept is translated in this context.

Certainly this is a poor-man's approach to making the most of expensive datasets, and it may miss some opportunities as the Sentinel revisit frequency is 5 days compared to the 1day rate of Planet Labs. Yet for many applications such as monitoring change in vegetation growth or urban development, this difference might not matter.

Clip a collection of satellite image raster bands to a specific region of interest (roi)

1) Get your choice of satellite imagery (Sentinel2, Landsat, Planet) and, convert to .tif (see above), zip the individual bands together to one file (say sat_bands.zip)
2) Create an ROI file that is contained within the extent of the satellite imagery. Geojson.io is a useful tool for this step (say sat_clip.zip).
3) Move the sat_bands.zip and sat_clip.zip files to the collection directory
5) Update the settings.txt file with the name of the sat_clip.zip file
6) Activate OTB environment:
conda activate OTB

6) Run the python script
python3 otb_clip.py

Follow the prompts to input your zipped raster band data choice. The bands will be clipped with the file specified in the settings.txt file, zipped and saved in the rasterimages/files directory.

Band operations (on Sentinel and Landsat series)

1) Activate OTB environment
conda activate OTB

2) Place a zipped folder with all bands available into the collection directory. Or just use the sentinel2_getdata.py for sentinel2 assets.
(Naming convention: area_monthday_year_satllite,zip > area2_0726_2021_sentinel2.zip)

3) Run the program
python3 otb_bandoperations.py

Follow the prompts to enter two variables: the satellite source and a band-math operation. Currently supported operations are: Normalized Difference Built-up Index (NDBI) for urban development and the Normalized Difference Vegetation Index (NDVI) for plant and tree coverage.

Pixel based classification (SVM or RF)

These routines assume you have created adequate training samples in a corresponding shape file, and update the settings.txt file accordingly.

1) Open the settings.txt file and change the parameters "raster image" and "raster shapefile" to your data. Make sure all the data paths are correct.

2) Enable the OTB environment:

conda activate OTB

3a) Run the program with the selected classifier as input argument (libsvm or rf):

python3 otb_raster_classify.py libsvm

3b) Run the related script that generates Haralick texture features derived from the image information and concatenates that information with the input image. Texture information can improve classification results on some images.

python3 otb_raster+texture_classify.py libsvm

4) You can also vary the parameters of the classifiers specified in the large'settings.txt' file and batch execute either rf or libsvm classifiers for each of the possible combinations. Edit the file otb_vary_raster_classify.py to set the parameters you want to include the classifier of choice. Then run the script:

*python3 otb*_vary_raster_classify.*py*

Results from all permutations of the selected parameters under the chosen classifier will be saved to the results folder. If you installed pCloud and enabled uploading, these results can be saved to that environment.

Object based classification (ANN)

These routines assume you have created adequate training samples in a corresponding shape file.

1) Deactivate the OTB environment first

conda deactivate

2) Open the settings.txt file and change the parameters "raster image" and "points file" to your data.

3) Check the paths in the vector_classify_top.sh file. Run the program:

sh vector_classify_top.sh

This script runs all the processes across multiple otb and qgis scripts to perform the object-based classification.

In both pixel and object-based classification, the performance of the classifier is evaluated (recall, precision, f1-score) for each category and saved to the results folder.