COCKTAIL

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

Last update: January 2025

Cocktail is a collaboration utility to facilitate collaborative remote sensing data collection, analysis, and evaluation under constrained resources. Cocktail is designed to facilitate resource-constrained collaborative inquiry on optical satellite assets, unlike the EOreader that also processes radar (SAR) data.

Collaboration is supported because Cocktail is a lightweight package that will run an even modest cloud-based Linux systems (16CPU and 32GB RAM). Cocktail will vary parameters of a classifier in batch mode and store the classifier performances for each setting so you can collaboratively find the best configuration. Cocktail keeps track of the large collection of parameters across the individual processing modules in a single file such that remote collaborators can reliably replicate a workflow. Moreover, 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.

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. Cocktail allows you to set up 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.

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 allows you to mix assets.

Installation, testing and usage:

1 - VM
Create a virtual computer.
Choose Ubuntu 18.04 LTS (or higher), with at least 16GB of RAM (test system: 8CPUs and 32GB RAM)

2 – Get the code from GitHub

git clone https://github.com/realtechsupport/cocktail.git

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.
Cocktail uses OTB 8.0.1 with python 3.8. See environment.yml for other details.

Create a conda environment with the settings in the environment.yml file.

conda env create -f environment.yml

6 – Setup and check the directory structure
After cloning the repository, your directory structure should look like this:

```
|-- miniconda2
|-- cocktail
        |-- code
        |-- data
                |-- auth
                        pcloud_auth.txt
                        sentinel_auth.txt
                        planet_auth.txt
                |-- collection
                |-- rasterimages
                        sentinel2
                        landsat8
                        planet
                |-- rawsat
                |-- vectorimages
                        |-- roi
                settings.txt
                somefile.geojson
                colormap.txt
                colormap.qml
        |-- results
        |-- setup
        README.md
        LICENCE
        Install+Use.pdf
```

7 – libdraco.so

This is only pertinent if you encounter the error stating ‘required libdraco.so.8’.

Navigate to the library directory*:*
cd /home/username/miniconda3/envs/OTB/lib/

Create a symbolic link*: ln -s libdraco.so.9 libdraco.so.8*
Verify the link*: ls -l | grep libdraco*

Then adjust the paths to reflect your current installation. From the setup folder run the adjust_datapaths script:

python3 adjust_datapaths.py

Follow the prompt. All paths will be adjusted to your current installation and the settings.txt file will be updated accordingly.

8 – Sample data

You can get sample data here:
https://filedn.com/lqzjnYhpY3yQ7BdfTulG1yY/FOSS4G_2023/

Put raster and vector files (including shapefiles) into the data/collection directory. 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 classification steps.

Change the fake username and passwords in the auth folder according to your needs. PC.txt is for pCloud, sent.txt and planet.txt for Planet Labs.

8b – Additional opensource data

If you want additional sentinel-2 datasets, use the openEO modules. This requires an account with ESA (European Space Agency). Add that information to the sentinel.txt file. Edit the search parameters in the openeo_getdata file accordingly. Run

python3 openeo_getdata.py

Put data from sentinel and planet into the collection directory. Add geojson references files (for planet downloads) and colormaps as needed to the data directory.

Note:
The import syntax changed in the latest version of GDAL:

import gdal -> from osgeo import gdal

9 - 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.

c) Update settings

Update the settings and file parameters (in settings.txt) corresponding to your system.

nano settings.txt

Then verify that all is ok

python3 settings_test.py

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 set up 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.

If you are using pCloud to move results from the VM to the pCloud server, install the library in the base and in the OTB environment with the correct python version (now 3.8)

```
pip install pcloud
conda activate OTB
```

Planet Labs datasets

If you have an account with Planet Labs, add the API key to a file in the auth folder. To access the Planet Labs imagery, you need that key. Then run the two scripts below:

python3 planet_get_previews.py

Get thumbnail-size previews of a scene based on the AOI, start date, number of days and cloud cover. Follow the prompt to enter an AOI (the geojson file), a target date, the number of days to extend the search and the maximum cloud cover acceptable. If you have external storage enabled (pCloud), the thumbnail images will be copied there, and you can preview them. To download the actual image, do this:

python3 planet_place_order.py

This script will download the geotif image, based on the image id identified in the preview, collected with the planet_get_preview module above. At the prompt, enter the image ID and the AOI. The image and metadata will be saved to the rasterimages directory. A copy of the .tif file will be transferred to external storage, if enabled.

Operations

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

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)
1b) Or just identify a single image / band that you want to clip.
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 either your zipped raster bands or a single band. If you operate on a single input, enter the location of that file at the prompt.
Inputs will be clipped with the file specified in the settings.txt file. If you input a raster band collection, all resultant bands will be zipped and saved in the rasterimages directory.

Normalized Difference Built-up Index (NDBI) on Sentinel and Landsat series

1) Activate OTB environment

conda activate OTB

2) Run the program

python3 otb_ndbi.py

Follow the prompts to enter a satellite source. The script will perform the Normalized Difference Built-up Index (NDBI) for urban development on the asset.

NDBI difference map on Sentinel and Landsat series

1) Same requirements here as items 1 and 2 above.

2) Run the program

python3 otb_difference_ndbi.py

Follow the prompt to enter two zipped satellite asset collections from different times and the same source. The script will perform the Normalized Difference Built-up Index (NDBI) for urban development on each asset, calculate the difference between the results and then overlay that difference data on the NIR band of the newer image. In both NDBI operations, the final image will be sent to external storage, if enabled in the settings.

Pixel based classification (SVM or RF)

These routines assume you have created adequate training samples in a corresponding shape file.
Update the settings.txt file if you want to change SVM or RF hyperparameters..

1) Put all inputs (raster images and shapefiles) in the collection directory.

2) Enable the OTB environment:

conda activate OTB

3a) Run the program

python3 otb_raster_classify.py

Follow the prompts to set the input .tif image and the classifier choice (Random Forest [rf] or Support Vector Machine [libsvm])

area2_0612_2020.tif rf

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

Follow the prompts to set the input .tif image and the classifier choice (Random Forest [rf] or Support Vector Machine [libsvm]) as above.

4) You can also vary the parameters of the classifiers specified in the settings file and batch execute either rf or libsvm classifiers for each of the possible combinations with a “script of scripts”. Edit the file otb_vary_raster_classify.py to set the parameters you want to include the approach (use texture or not) and the classifier (rf or libsvm).
Then run the script:

python3 otb_vary_raster_classify.py

This script pauses after each iteration to ensure that the timestamps (1minute resolution) generated for the image results are unique. Results from all permutations of the selected parameters under the chosen classifier are saved to the results folder. If you installed pCloud and enabled uploading, these results are saved to that environment.

If you want only to create a model that you can then use to classify various input images, use the routines otb_raster_train.py or otb_raster+texture_train.py together with otb_raster_classify_with_model.py

Run the script:

python3 otb_raster_train.py

You can add one or more pairs of raster images together with a corresponding shapefile. The last entry is the classifier choice: Random Forest or Support Vector Machine. The script otb_raster+texture_train adds texture information gleaned from the raster image based on Haralick texture features. However, the routine accepts one image and shapefile pair. In either case, the routines will create a model in the results directory and copy it to external storage, if enabled.

Run the script:

python3 otb_raster_classify_with_model.py

to classify a raster image based on the previously created model. At the prompt enter the raster image to classify and then the model to apply – the model just created in the previous step, for example.

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 “pointszipfile” (the zipped vector shape file) to your data.

3) 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.

Enabled by default in the settings.txt file, summary statistics (precision, recall, f1-score) of the classifier performance across all categories are calculated, saved for reference (and transferred to external storage, if enabled).