RIOSS Project
Version 1.0
Installation Manual

Luís Felipe Ferreira de Mendonça
Marcos Reinan de Assis Conceição
contact@rioss.org

April 2021

A Brief Overview of RIOSS……………………………………………………….….3

Installation Steps………………………………………………………………..4

System Requirement…………………………………………………………….4

Download and Extract code……………………………………………………..5

Set Paths………………………………………………………………………...5

Sentinel 1 Image Data…………………………………………………………..6

Run RIOSS Code……………………………………………………………….7

Window Size Adjusts…………………………………………………………..7

1. A Brief Overview of RIOSS

RIOSS is a set of open-source routines capable of identifying oil-like spills based on random forest classifiers trained and tested on SENTINEL-1 SAR imagery. Both European Space Agency’s (ESA) Sentinel-1 A and B satellite data were used to develop our dataset and apply the proposed methodologies. These satellites have a C-band synthetic aperture radar (SAR) operating in the wide range and TOPS mode. Preprocessing of images makes use of ESA SNAP’s Sentinel Toolboxes. An optimized feature space to serve as input to such classification models, in terms of variance and computational efficiency. It involved an extensive search from 42 image attribute definitions based on their correlations and classifier-based importance estimates. This number included statistical, shape, fractal geometry, texture, and gradient-based kinds of attributes, all of them related to the imaged phenomena's physical aspects. Mixed adaptive thresholding was performed to calculate some of the features studied, returning consistent dark spot segmentation results. This process helped us to develop optimal random forest models, resulting in up to 90% accuracy. In a continual process, we mean to get even more reliable results.

2. Installation steps

The Radar Image Oil Spill Seeker (RIOSS) is a program developed in Python by Conceição, et al (2021) as a machine learning system for detecting oil spills on sea surface through SAR images. This system was developed to support scientific studies on ocean oil spills, through an open and collaborative code, which is available on RIOSS' GitHub. In addition, a more complex operational version of RIOSS runs in nearly real time, at Satellite Oceanography Laboratory of the Federal University of Bahia, to detect possible oil spills on Brazilian coast.

2.1. System Requirement

The project was developed and supported in Linux Architecture (tested on Ubuntu 18.04 LTS or later). The software basic requirements are:

FORTRAN 90/95 compiler.
Installation code: ***sudo apt-get install gfortran***

GCC.
Installation code: ***sudo apt-get install gcc***

G++.
Installation code: ***sudo apt-get install g++***

NETCDF library.
Installation code: ***sudo apt-get install netcdf-bin netcdf-dev***

PYTHON.
Installation code: ***sudo apt-get install python***

PIP (python installation manager).
Installation code: ***sudo apt install python3-pip***

Python libraries: cython numpy numba pandas scikit-learn scikit-image scipy matplotlib cartopy seaborn tqdm pillow
Installation code: ***sudo pip*** **library-name-here**

Development Python Tools.
Installation code: ***sudo apt install build-essential libssl-dev libffi-dev python3-dev***

2.2. Download and Extract Code

The RIOSS source code is available for download from RIOSS Project website. (**https://rioss.org/download.html**). The python code is provided in *.zip* format and must be unzipped.

The extract operation will create the directories:

- debug
- example
- models
- routines
- test_case

"WARNING: Our code and parameters are independent. Modifying RIOSS code may cause execution problems. We recommend only modifying the parameter file."

2.3. Set Paths

The code does not need to be compiled. If the libraries have been installed correctly, we just need change the destination folders.

Open the path EXAMPLE and copy the file *parameters.example.py* to RIOSS root path as parameters.py. **Execution code: *cp /home/user/RIOSS/EXAMPE/parameters.example.py /home/user/RIOSS/parameters.py***

Open, with text editor, the file *parameters.py* and edit the line 16 as shown in the example below.

DATA = join(HOME, 'RIOSS/test_case') # home

3. Sentinel 1 Image Data

1. The RIOSS code was, initially, developed for analysis of oil spills in Sentinel 1 constellation SAR images, from the European Space Agency (ESA). This data can be acquired through direct access, with registration on the Copernicus Open Access Hub Available in: ***(https://scihub.copernicus.eu/dhus).***

Users without prior knowledge of SAR data, we suggest downloading of GRD format. The spatial resolution of GRD products corresponds to the mid-range value at mid-orbit altitude, averaged over all swaths (SM/WV) or sub-swaths (IW/EW). The range resolution is ground range. The equivalent number of looks (ENL) for IW and EW GRD products corresponds to an average over all sub-swaths.

Our research are conducted with SLC-IW data. Single Look Complex (SLC) products have spatial resolutions that depend on acquisition mode. IW-SLC products have all bursts in all sub-swaths are re-sampled to a common pixel spacing grid in range and azimuth.

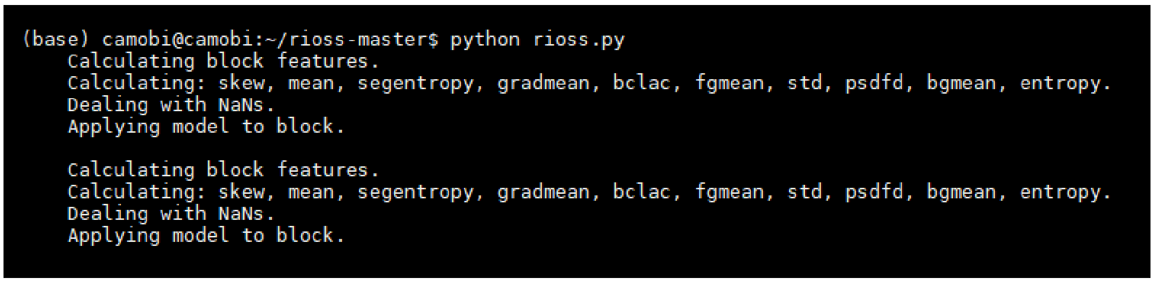
The pre-processing steps can be performed in the SNAP toolbox (ESA) software. Available in: ***(https://step.esa.int/main/download/snap-download/).*** To process the Sentinel-1 SAR IW-SLC data, we execute the SNAP toolbox commands:

- Thermal Noise Removal
- Apply Orbit File
- Radiometric Calibration
- TOPS Deburst
- Multilooking
- Converts Bands to from dB

4. Run RIOSS Code

After set the folders and process the SLC Sentinel-1 data, or direct use of the Sentinel-1 GRD data, the user will be able to run the rioss.py file.

When running the rioss.py, the user should see a sequence of commands on the screen, as shown the figure below.



```
(base) camobi@camobi:~/rioss-master$ python rioss.py
    Calculating block features.
    Calculating: skew, mean, segentropy, gradmean, bclac, fgmean, std, psdfd, bgmean, entropy.
    Dealing with NaNs.
    Applying model to block.

    Calculating block features.
    Calculating: skew, mean, segentropy, gradmean, bclac, fgmean, std, psdfd, bgmean, entropy.
    Dealing with NaNs.
    Applying model to block.
```

At the end, the execution will be creating a new folder /maps in your home, edited in parameters.py "DATA = join (HOME, 'RIOSS/test_case')". This folder will have the name of day on which the RIOSS code was run.

Example: "RIOSS/test_case/ maps/2021050215_rioss/"

Inside the *"maps/2021050215_rioss/"* folder the user must find three new paths:

/bin – Binary files generated by the classifier algorithm.

/img - Figure with the oil spill probability occurrence.

/results – File in CSV format with location of high probabilities found.

5. Window Size Adjusts

RIOSS allows the user to choose the size attribute classification window. In parameters.py file the user can edit the window size on line 54. Small oil spills are usually best identified in windows below 512x512. We suggest testing classification windows of 128x128, 256x256, 512x512 and 2014x2014.

Example: "W = 512". To classification window size 512x512 pixels.