Table of Contents

1. Msc-thesis

Introduction

This document provides an overview of different folders: in which folder data is stored and where Python scripts can be found.

1.1.Data

This folder contains different types of data used during this thesis

1.1.1. Buoy

The wave buoy data is stored in this folder. Wave buoy data from the North Sea was too big to upload on github and is therefore not included. Buoy data from Portugal is included.

1.1.1.1. OCEANSAR

This folder contains wave buoy data for different case studies, used to run OCEANSAR. Location of these wave spectra is defined in the configfile. Each file is a numpy array which consists of a row of frequency bins with variance density, mean wave direction and mean spreading.

1.1.1.2. PORTUGAL

The wave buoy data from the two Monican wave buoys is stored in this folder. This data was used for case studies along the Portuguese coast.

1.1.2. Configfile

This folder contains OCEANSAR configuration files for case studies in the North Sea.

1.1.2.1. Portugal

This folder contains OCEANSAR configuration files for case studies along the Portuguese coast.

1.2.Scripts

This folder contains all the scripts which were used during this thesis. The only script which is normally executed is : *main_script.py*. All the other functions are imported into this script and executed. The following scripts are found:

Main_script.py

This is the main script which is used to process a single or multiple S-1 SLC images. Input is a folder which contains the unzipped satellite products (each file is stored in a folder with .SAFE extension). It will also find wave buoy data on the same time stamp as a satellite product. The script executes the following steps, indicated behind each step is the function which executes this step:

1. Load the satellite product
2. Read meta-data from the satellite XML file (*preprocessing.py)*
3. Read satellite geo-reference points and determine if wave buoy is located in a sub-swath, if the wave buoy is not located in a sub-swath, the next swath is processed. (*latlon.py)*
4. Calculate the heading of the satellite using geo-reference points. (*latlon.py)*
5. Define additional heading according to ascending or descending pass. (*latlon.py)*
6. Load buoy data from Portugal or the North sea at the time stamp of satellite overpass. (*buoy.py)*
7. Create save path and copy meta-data to save path. (*preprocessing.py)*
8. Transform buoy data from frequency-direction to wave-number domain. Plot 2D wave spectra . (*buoy.py)*
9. Load S-1 SAR image and calibrate image (*preprocessing.py)*
10. Deramp and demodulate calibrated SAR image. (*preprocessing.py)*
11. Processing entire sub-swath, calculate cross-spectra for every imagette. (*process.py)*
12. Plot imagette cross-spectra on sub-swath image. (*process.py)*
13. Plot intensity image and plot variance per imagette. (*process.py)*
14. Calculate the variance per imagette, imagette with lowest variance is further processed. Alternatively an imagette can manually be set by changing meta['az_loc'] or meta['ra_loc'] to the desired imagette. (*process.py)*
15. Start processing of a single imagette, output are different sub-look cross-spectra. (*process.py)*
16. Create plot of the 3 different sub-looks. (*process.py)*
17. Create 1D and 2D plots of cross-spectra. (*process.py)*

NOTE: Due to the size of the Sentinel-1 data it was not possible to store an SLC image on github. A test case can be downloaded from the Copernicus Scihub with a Copernicus scihub account: https://scihub.copernicus.eu/dhus/odata/v1/Products('7cb73b7d-780a-4c04-af28-549b4b8b6ea8')/$value (this is the SLC image used during the case study on 2014-11-25 from this thesis). After downloading the image needs to be unzipped before it can be processed.

Data is stored in the folder *processed* in the data directory. Below is a short overview of the different scripts:

Preprocessing.py:

This script contains routines to read XML data from a satellite product, as well as routines to calibrate and deramp the image with the meta-data.

Processing.py

This script processes different imagettes. It has routines to create sub-looks from imagettes and calculate the cross spectra. It also plots the cross-spectra and saves the resulting images.

Buoy.py

This script is used to process wave buoy data. First it transforms the buoy data from frequency-direction to wave number domain. Next it creates a plot of this 2D wave spectrum.

Latlon.py

This script determines the georeferenced information of a satellite product. It calculates the heading of the satellite image and it has routines to determine if a wave buoy is located within an Satellite image sub-swath.

OCEANSAR_processing.py

This script was used to process the OCEANSAR SLC images. It contains several routines which are also found in the scripts *processing.py and process.py*. The function *process_oceansar_data* is responsible for processing of the data. It follows the same steps as used for the Sentinel-1 SAR data. Output is 2D cross-spectra from which an image is plotted.

1.2.1. OCEANSAR

This folder contains two scripts which originate from the OCEANSAR module (*https://github.com/pakodekker/oceansar)* but show small changes to accommodate the date used during this thesis. The file oceansar_sarsim.py is used to run the OCEANSAR simulator, input is a list of configuration files (which are stored in the folder *configfile)*. The file *buoy.py* contains routines to process the wave buoy data, adjustments were made in the rotation of the wave buoy data to change between the frame of reference between the wave buoy data and the SAR data. **NOTE:** The latest version of OCEANSAR has an updated version of this script, so the *buoy.py* script in this folder is obsolete.

1.2.2. Extra

This folder contains several additional scripts which were used during this thesis. The script *load_spectra_buoy.py* is used to transform the wave buoy data from different sources into a single format which is used to create 2D wave spectra.

The file *sentinel_download.py* contains routines to download satellite images from the *Sentinel Scihub* using the *sentinelsat* package.

This folder also contains the geoJSON objects which were used to download the satellite products.