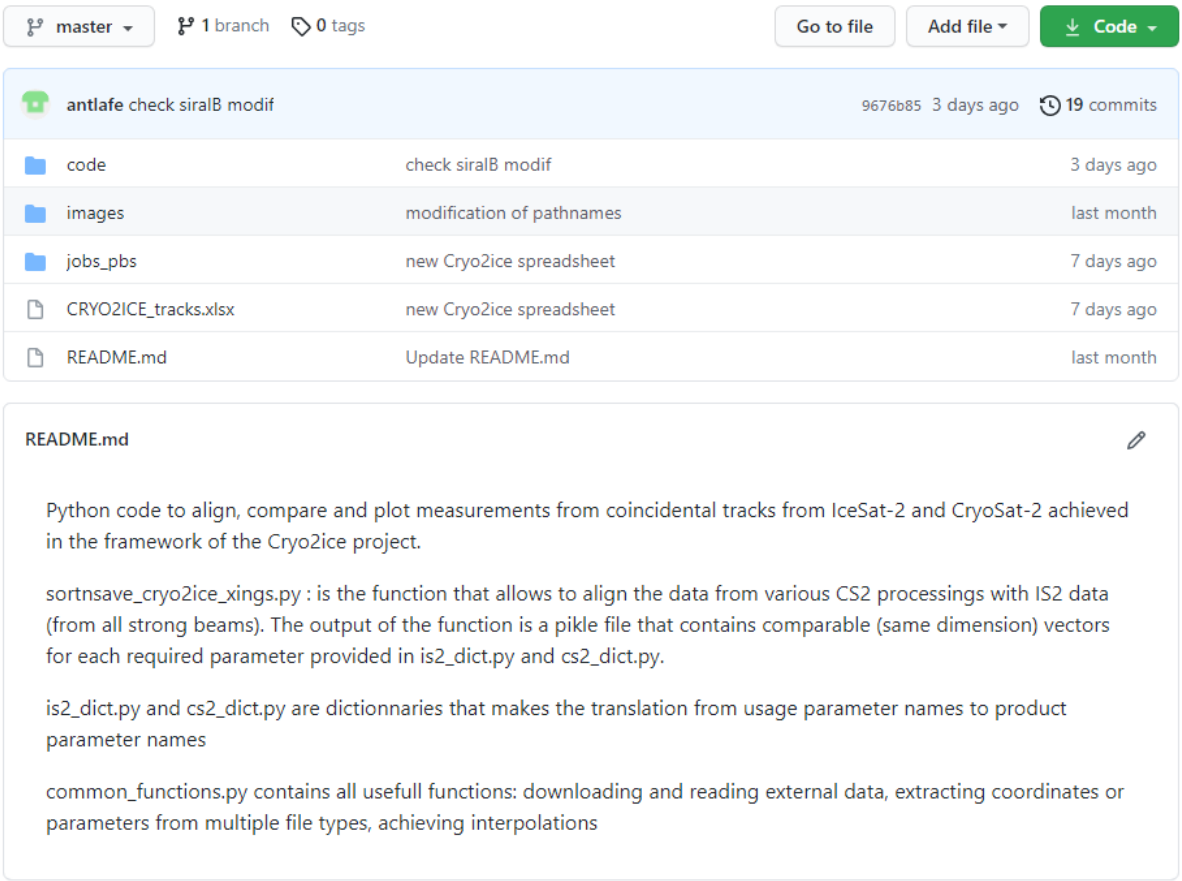
CRYO2ICE

Software user manual

1. Downloading scripts

All functions can be found and downloaded here:

https://github.com/antlafe/projet_cryo2ice



The python packages used can be found in the Anaconda distribution.

```
> git init
> git clone https://github.com/antlafe/projet_cryo2ice
```

2. Functions

a. List of functions

Name	Usage	comment
sortnsave_cryo2ice_xings.py	Main function which aligns CS2/IS2 tracks together.	Create pikle file of aligned CS2/IS2 data that can be read by other functions
animation_cs2_is2_scatters.py	Animates CS2 and IS2 data and plots the required parameter.	Reads pickle file to display an animation (see https://twitter.com/esa_cryosat/status/1359446279210827777)
statistics_cryo2ice.py	Display plots / histograms / scatters from IS2 aligment matrix ..etc	Reads pickle file
statistics_cryo2ice_mean.py	Display plots / histograms / scatters from IS2 weighted mean ..etc	Reads pickle file
stats_tools.py	Display functions: Maps, scatters, histograms	Display function library
common_functions.py	Catalogue of common functions: reading files, small operations...etc	
grid_data.py	Gridding function (used for IS2)	Adapted from LEGOS function
plot_xings.py	Plotting Xover out of the netcdf file of Xings calculated by the LEGOS algorithm: ct_xings	
get_CS2_cryo2ice.py	Finds collocated tracks for CryoSat-2 files	

b. Dictionaries

Name	Usage	comment
cs2_dict.py	CryoSat-2 parameter dictionary	
is2_dict.py	IceSat-2 parameter dictionary	
saral_dict.py	Saral parameter dictionary	
path_dict.py	Paths dictionary	To modify depending on where data are located

c. Auxiliary functions

Name	Usage	comment
grid_and_filter.py	Gridding and smoothing function	From LEGOS
parserObjects.py		Online package
W99.py / W99_sd.txt		

d. Auxiliary files

CRYO2ICE_tracks.xlsx contain information about the collocated tracks (see 3.b)

3. Downloading the data

a. All data

Collocated tracks between the CryoSat-2 and IceSat-2 missions are identified thanks to the Cryo2Ice website (https://cryo2ice.org/).

IceSat-2 ATL07 and ATL10 data are downloaded from the NSIDC website (https://nsidc.org/data/) for each Reference Ground tracks.

CryoSat-2 L2 baseline-D data are downloaded from ftp://science-pds.cryosat.esa.int/.

Other products (AWI, CPOM, UOB) are specifically ordered to the institutes that produces them.

b. Collocated tracks

CRYO2ICE_tracks.xlsx is a spreadsheet that lists all collocated tracks information: absolute orbit number, date & time and filenames for both CryoSat-2 and IceSat-2.

Collocated tracks are to be found every **20 orbits for IceSAT-2** and every **19 orbits for CryoSat-2**. As the Reference ground tracks (RGT) number is provided in the file name for IceSAT, it is quite simple to find the right files.

Filing the xlsx file:

1/ Extending the IS2 RGT number series by incrementing by 20.

2/ Copying the CS2 files using the bellow bash loop and adding to the excel file

```
(for rgt in {0612..1200..20}; do echo ${rgt}; cp
/work/ALT/odatis/seaice/users/laforga/data/IS2/ATL10/202102/ATL10-
01_202102????????_${rgt}1001_*.h5 ./; done)
```

3/ Extending the CS2 RGT number series by incrementing by 19.

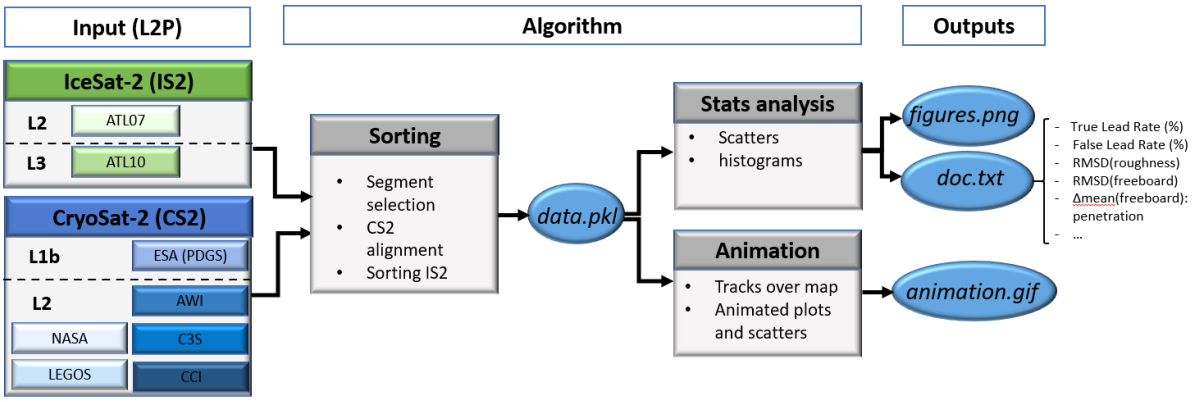
4/ Copying the CS2 Baseline-D files using the function: **get_CS2_cryo2ice.py** and copy into the excel file.

(As the RGT number is not available in CryoSat-2 files, the collocated tracks can either be found using datetime provided in the spreadsheet or by opening each file and reading the absolute orbit number. This task is achieved by the function: **get_CS2_cryo2ice.py** where both these options are available.)

5/ You can then use the function **get_CS2_cryo2ice.py** to get other type of files (it will then use equator time to deduct the right files)

4. Logic

The objective of the scripts is to make comparable different altimetric products from the two missions CryoSat-2 and IceSat-2 over the coincidental tracks of the resonant orbits (CRYO2ICE). This task is achieved by the function: sortnsave_cryo2ice_xings.py. This function produces a pickle file of aligned CS2/IS2 data which can then be used to make various comparisons and plots.



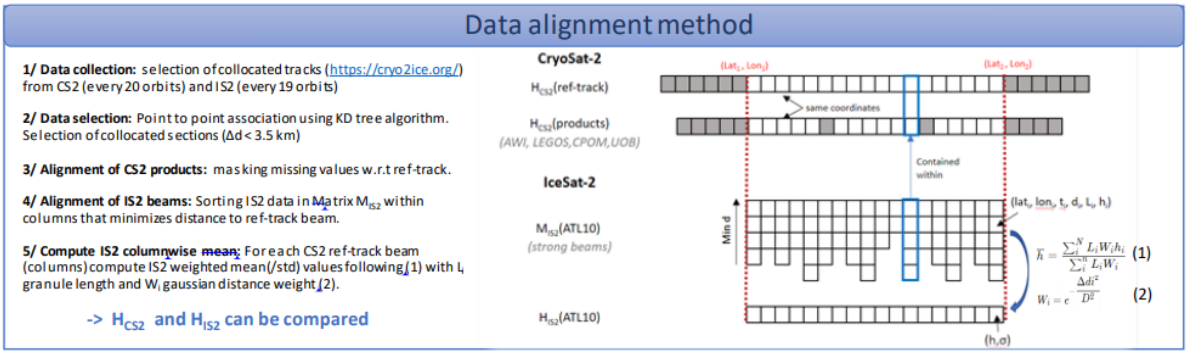
The logic is presented in the ATBD document.

5. Description of the functions

sortnsave_cryo2ice_xings.py

1. Description

This is the main function of the project. It allows to align, sort and save the IS2 data relatively to the CS2 data (see ATBD for more detail on the methodology). The aligned data are saved in a dictionary (pickle file) which is then read by all other functions (animation_cs2_is2_scatters.py, statistics_cryo2ice_mean.py)



2. Usage

Python sortnsave_cryo2ice.py [options]

-s satellite name

-g gdr

-d dates: YYYYMMDD,YYYYMMDD

-o outpath: location where to save .pkl file

-sw [OPTION]: add swath data (only SLA for the moment)

-x [OPTION]: do cross-over analysis

3. Prerequisites

1/ Download collocated tracks, one per day maximum (collocated track every 1.5days) (see 3.b) and sort in monthly repertories the files

Ex:

PATH/CRYO2ICE/CS2/YYYYMM/files.ext

2/ Add non corresponding dates btw CS2/IS2 in MIDNIGHT_DATES dictionary in the path_dict.py file for each IS2 GDR [see spreadsheet: CRYO2ICE_tracks.xlsx last column]

3/ Modify path_dict.py with correct path repertories

4/ If new product: filepattern, path and parameters names in dictionnaries (is2_dict.py or cs2_dict.py for IS2 and CS2)

5/ Check global attributes of sortnsave function

flag_1hz: *if True, the data are aligned on 1hz measurements and the 20hz are averaged*

If False, the data are aligned on 20hz measurements and 1hz are interpolated.

flag_IS2_mean *= True # compute mean weighted value for each IS2 parameter (to get a same size 1D arrays between CS2 and IS2); if False: provide a matrix that contain each individual IS2 granules sorted by order of closest to beam centre*

MAX_DIST_OF_COLLOC_DATA*= 5.5 # km maximum acceptable distance for the collocated tracks*

LAT_MIN *= 55 # deg North minimum latitude to look for collocated tracks*

6/ Check parameters to be aligned in cs2_dict.py and is2_dict.py

4. Examples

python sortnsave_cryo2ice.py -s CS2 -g LEGOS_SAM,AWI -s IS2 –g ATL10 -s SARAL,S3 -d20201001,20201007

-> Aligns CS2 GDR LEGOS_SAM and AWI to IS2 GDR ATL10 and compute cross-over statistics for S3 and SARAL over the collocated tracks for period 20201001,20201007.

python -m ipdb sortnsave_cryo2ice_xings.py -s CS2 -g ESA_BD_GDR -s SARAL -gLEGOS_T50 -sIS2 -gATL07,ATL10 -d20201103,20201103 -ofn test

->

Input data

- Data/
  - **CS2/** [All CryoSat-2 files]
    - **GDR/**
      - YYYYMM/files
    - ...

- **IS2/** [All IceSat-2 files]
  - **GDR/**
    - YYYYMM/files
  - ...
- **CRYO2ICE/**
  - **Cryo2Ice/** [Aligned .pkl files]
    - ProName/
      - Data_dict.pkl
      - Info_param.pkl
      - Status.txt
  - **CS2/** [CryoSat-2 collocated files]
    - **GDR/**
      - YYYYMM/files
    - ...
  - **IS2/** [IceSat-2 collocated files]
    - **GDR/**
      - YYYYMM/files
    - ...

Output

```
Data_dict = {
'CS2':
        {
                # product wise aligned data
                'ESA_BD_GDR': {'id', 'ref_idx', 'cs2_idx', 'latref', 'lonref', 'latref_full', 'lonref_full',
                'time', 'surface_type', 'lpe', 'radar_h', 'geoid', 'quality_flag', 'sic', 'sla', 'lon', 'lat', 'pole',
                'u10', 'radar_fb', 'earth', 'ocean', 'dac', 'mss', 'isa', 'ssb', 'swh', 'load'}
                'AWI': {}
                ...
                 # cross-over data
                'xings': {}
                # swath data (10km segments)
                'swath': {}
        },
'IS2':
        {
               # product wise aligned data
               'ATL10': {}
               # cross-over data
               'xings': {}
                # swath data (10km segments)
               'swath': {}


        },
```

```
'dates': {},
}
```

cs2_dict.py / is2_dict.py

These scripts are the dictionaries where the names of the parameter for each product are specified. When calling a specific product (or GDR: ESA_BD_GDR, AWI, ATL10 ..ect) all parameter listed in these dictionaries will be processed and aligned with the sortnsave_xings.py function.

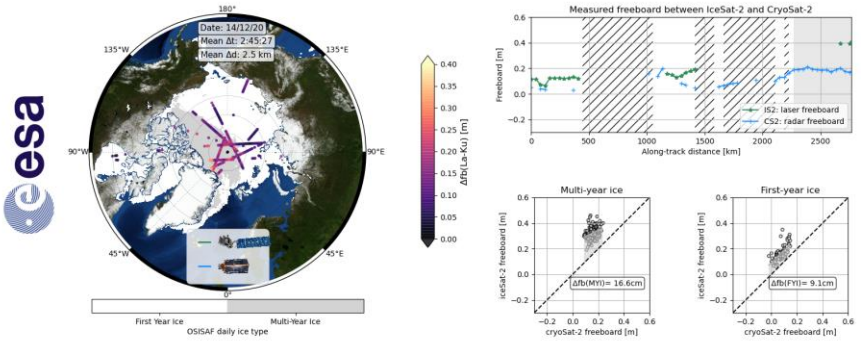
path_dict.py

This dictionary specifies the path for the various repertory called by all the function of the project. The advantage of this file is to make the function usable by anyone if you provide the right path for the data in this file.

animation_cs2_is2_scatters.py / animation_cs2_is2_scatters.py

1. Description

These functions produce an animation out of the aligned CRYO2ICE data (pickle file produce with sortnsave_xings.py). It displays the collocated CS2/IS2 tracks on an Arctic map and the parameters values along that track.



2. Usage

python animation_cryo2ice_mean.py [options]

-g CS2 GDR name [product name]

-p parameter CS2

-g IS2 GDR name [product name]

-b beams to show

-p IS2 parameter

-d dates: YYYYMMDD,YYYYMMDD

-o outpath: location where to save animation (.mp4)

-f INPUT FILE (pickle file)

3. Pre-requisites

You need to run sortnsave_xings.py function

4. examples

python -m pdb animation_cryo2ice_mean.py -g ESA_BD_GDR -p radar_fb -g ATL10 -b b1,b2,b3 -p laser_fb -d20201205,20201230 -f NovMar_ESA -o test

statistics_cryo2ice.py / statistics_cryo2ice_mean.py

1. Description

These scripts are the laboratory. They read the pickle file to make all sort of comparisons in between CS2/IS2 and external data-sets to produce all type of figures and statistics.

2. Usage

python statistics_cryo2ice_mean.py [options]

-f INPUT FILE (pickle file)

-g CS2 GDR name

-p type of comparison. There is a list of available comparisons, you can make some new ones.

-d dates: YYYYMMDD,YYYYMMDD

Pre-requisites

You need to run sortnsave_xings.py function

3. Examples

python -m pdb statistics_cryo2ice_mean.py -f NovMar_ESA -g ESA_BD_GDR -p simba -d20201101,20210331

stats_tools.py

1. Description

Collection of statistical display functions: maps, histograms, scatter plot, rolling median to be called by other scripts to make comparisons.

common_functions.py

1. Description

Collection of commun functions to download and read data and to do some basic operations. To be imported and used by all other functions.

grid_data.py

1. Description

Gridding function. Mostly used to grid IS2 since it is not in the LEGOS database yet. For other mission, I use the ct_grid_tracks function from the ct_tools.

2. Usage

python grid_data.py [options]

-s satellite name

-g GDR

-d YYYYMM

-b beam [for iS2]

-p parameters to grid

-hp hemisphere code [01: Arctic, 02: Antarctic]

-o pathout

3. Pre-requisites

Add parameters you wish to grid in dict.py and provide data location in path_dict.py dictionnary

4. Examples

python grid_data.py -s IS2 -g ATL10 -d 202102 -b b2 -p laser_fb,surface_h,gaussian_w -hp 01 -o ./

get_CS2_cryo2ice.py

1. Description

Function to find CryoSat-2 collocated tracks from information in spreadsheet or by its absolute orbit number provided in the global attributes of ESA official product.

Rq: collocated tracks are to be found every 19 orbits.

2. Usage

python get_CS2_cryo2ice.py [options]

-p CS2 Product name

-pn location of data

-d YYYYMM

3. Pre-requisites

For products other than ESA official products, equator crossing time of the collocated track must be specified in the spreadsheet.

4. examples

python -m pdb get_CS2_cryo2ice.py -p CPOM –pn ~/Documents/work/projet_cryo2ice/data/CS2/CPOM/202011/ -d 202011

Appendix

Production of LaKaKu gridded product

There are two functions to make these products.

grid_data_lakaku.pbs : main pbs function coded in bash which produces each grid: IS2, CryoSat-2: SAM+, TFMRA50, SARAL using the LEGOS database.

Usage: qsub –N prodLaKu grid_data_lakaku.pbs to launch on HAL

LaKu_grid2netcdf.py: Python function that gather and format gridded parameters to product the final netcdf product.