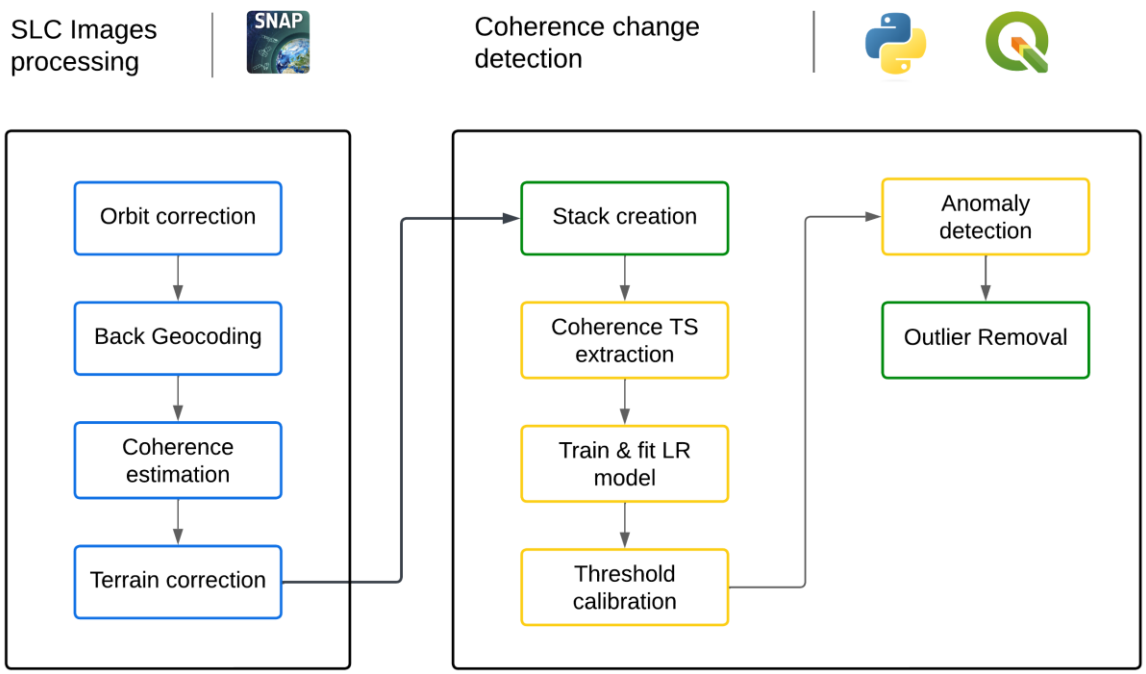
The methodology developed for this work can be split into 2 parts (as shown in the figure below). First, SLC image processing and the generation of the Coherence image stack with ESA'S SNAP and QGIS, followed by the anomaly detection in the coherence timeseries done with a python script which returns the pixels identified as "damaged buildings".



1) SLC processing and Coherence stack creation:

1.1) SLC Images processing:

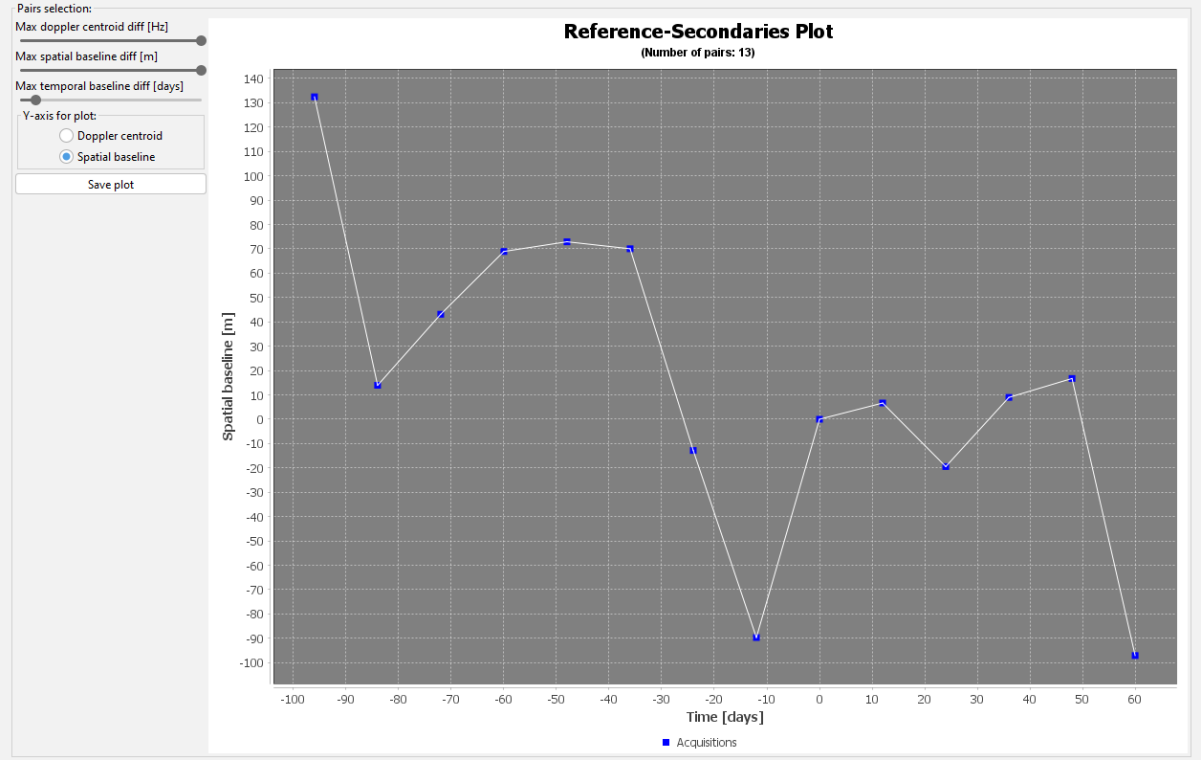
Our methodology follows a standard workflow for InSAR, starting by applying orbits corrections (**Apply Orbit File** operator in SNAP) and **splitting** the SLC images keeping only the bursts covering our area of interest (AOI) (**at least two bursts even if the AOI is covered by just one**).

Secondly, we need to **co-registe**r our images and create an interferometric stack. Starting by "**back geocoding**" the images (RADAR > coregistration > S1 TOPS coregistration > S1 Back Geocoding) while putting the master image on top of the list of inputs (you can use the **Stack Preview** tool to identify which image to use as a Master).

Afterwards you can run the created stack through **Enhanced Spectral Diversity** (optional but still recommended when working with longer timeseries) for an improved coregistration (RADAR > coregistration > S1 TOPS coregistration > S1 Enhanced Spectral Diversity).

The next step would be the **TOPSAR Deburst** (RADAR > Sentinel-1 TOPS > S1 TOPS Deburst) to merge the selected bursts, you can then crop the debursted stack keeping only your AOI (RASTER > SUBSET).

Then we generate our coherence images using the **Multitemporal InSAR** tool in SNAP by adjusting the parameters to the **lowest temporal baseline possible and highest baseline & doppler centroid**, this will ensure that out interferometric pairs will be formed in chronological order, meaning coherence will be measured between each image and the one acquired right after it, this is crucial if we want to keep temporal decoerrelation/loss of coherence to a minimum and thus making the coherence values measured at different dates comparable. Depending on which version of SNAP & microwave toolbox you're using, it might be required downloaded an elevation band first (Right-click on your file in the product explorer window)

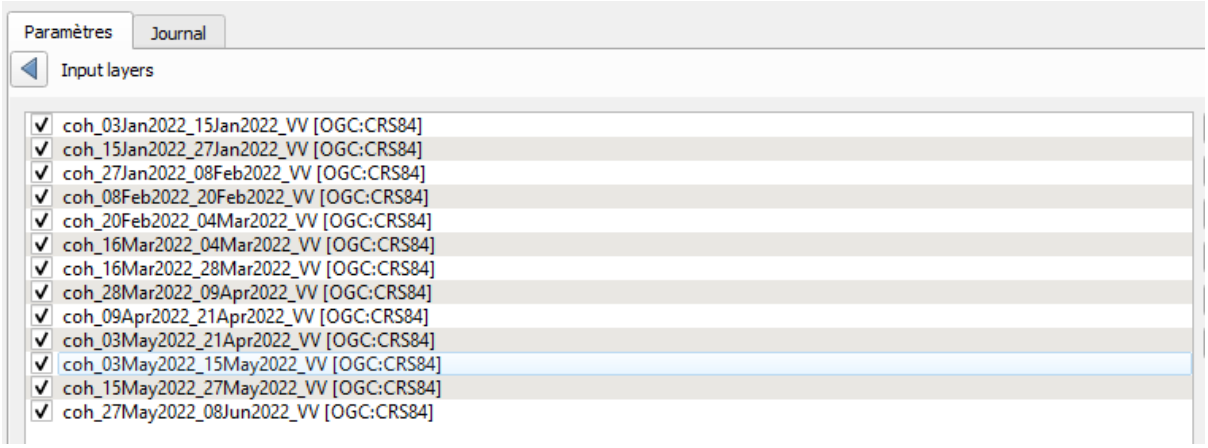


The final step in SNAP is terrain correction (RADAR > Geometric > Terrain Correction > Range-Doppler Terrain Correction) before it I recommend running band subset keeping only the coherence & elevation bands.

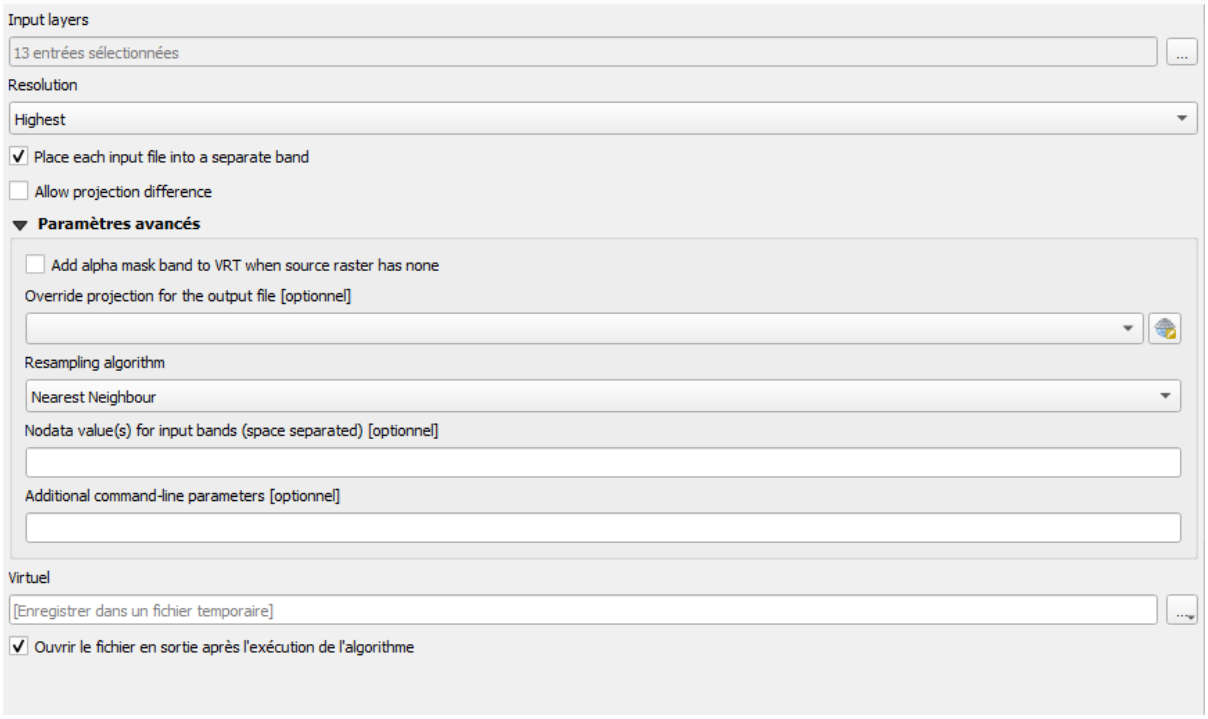
1.2) Coherence Stack creation:

After the SLC image processing in SNAP, we need to create a .tif files will where each band include the coherence values for one date.

First, we need to load our coherence images into QGIS (.img files found in your image folder). Then create a raster combining all the raster images, where each band represents the coherence of one of the interferometric pairs formed. For this, we will use the virtual raster tool in QGIS. (RASTER > MISC > VIRTUAL RASTER) **IT IS IMPORT to sort the coherence images introduced in the virtual raster tool by chronological order.** (see image below)



Also make sure to **check** **“Place each input file into a separate band”**.



You can then export the .vrt file created as a .tif file.

2) Anomaly detection in coherence timeseries:

2.1) Coherence timeseries extraction:

The first cell/part in the python script extract the coherence time series from the coherence stack, which would be saved in a dataframe where each row is the coherence time serie for one pixel. Its coordinates are also stored in the last 2 columns. It is also possible exports the dataframe as csv file for later use.

2.2) Anomaly detection:

In the second cell we will use our dataframe to train & fit a linear regression model using SkLearn library, I have found that the Lasso model works best for this application. The resulting linear regression series will (for pixels over built up areas) represent the coherence timeseries if no damages occurred.

The last part of our code will, for every date (& pixel), measure the difference between the coherence values & the linear regression, then compare it to a threshold value defined by the user (from testing I have found that a value between 0.4-0.3 works best). I have also introduced another condition that will ensure that comparison is done only for built up areas, the coherence values for the pre-event dates need to exceed a threshold value (I recommend using a threshold of 0.5-0.7).

The pixel identified as “damaged buildings” will be saved in a list, then put in a dataframe then exported as a .csv file. You can then import it in your GIS using the last 2 columns as your X &Y.