\_p

**TSX2StaMPS: Data preparation for StaMPS PSI processing of high-resolution TerraSAR-X data with SNAP**

# Table of Contents

Inhalt

[Table of Contents 2](#_Toc119999005)

[1 Introduction 3](#_Toc119999006)

[2 Data download 4](#_Toc119999007)

[3 Download and installation 4](#_Toc119999007)

[4 Step by step 5](#_Toc119999008)

[5 References 15](#_Toc119999009)

# Introduction

This tutorial represents the extension of the tutorials of the Research and User Support for Sentinel core products (RUS) service, which provides a free and open scalable platform in a powerful computing environment and hosts a suite of open-source toolboxes pre- installed on virtual machines, to handle and process data derived from the Copernicus Sentinel satellites constellation (Serco Italia SPA 2020).

Persistent Scatterer Interferometry (PSI) is a powerful advanced DInSAR technique able to measure and monitor displacements of the Earth’s surface over time with high accuracy (Serco Italia SPA 2020). Hooper et al. (2004) proposed a novel PS selection using phase characteristics, which is suitable to find low-amplitude natural targets with phase stability that cannot be identified by amplitude-based algorithms. This work originated one of the most widely used PSI software packages, StaMPS. 1

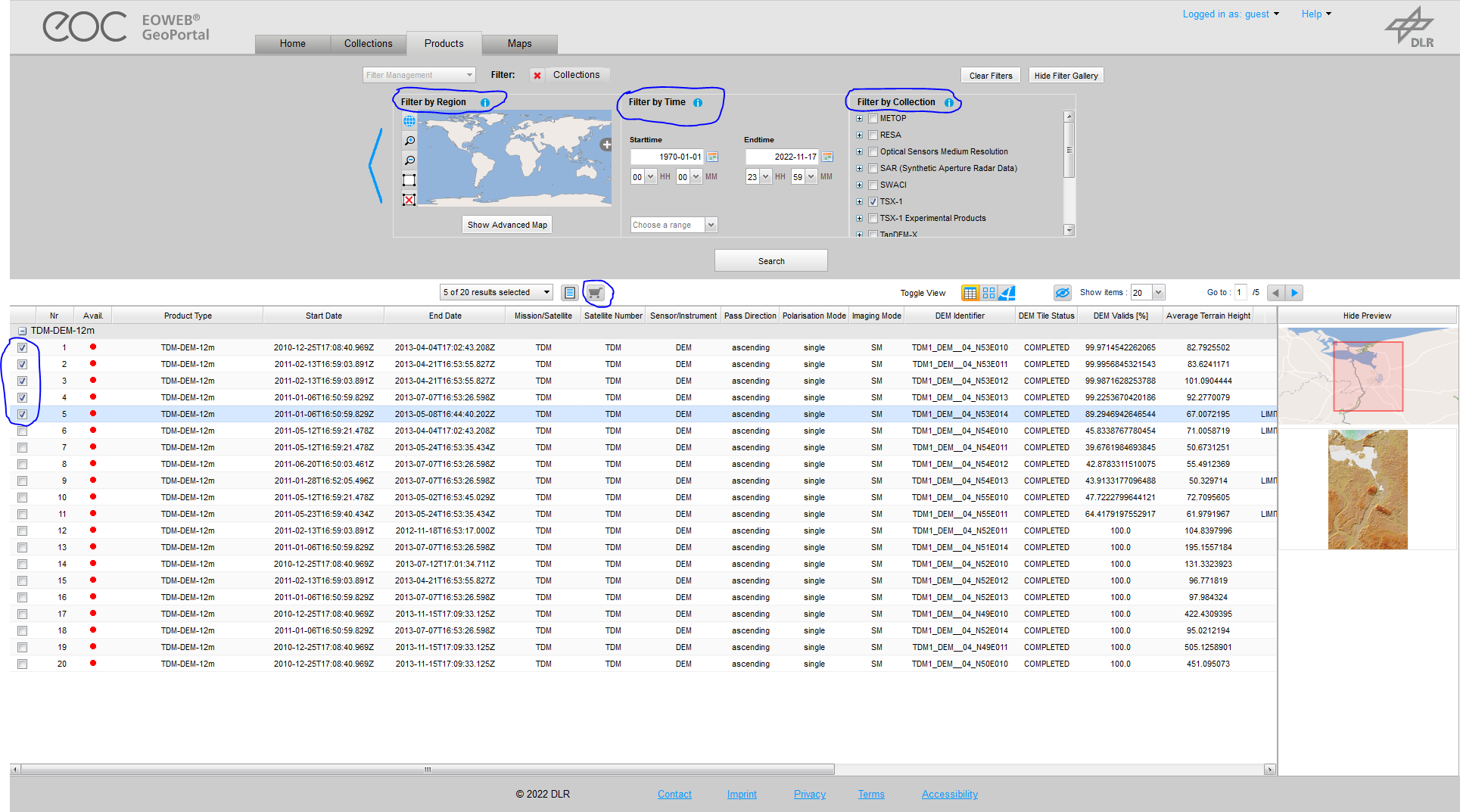
SNAP2StaMPS is a Python workflow developed by José Manuel Delgado Blasco and Michael Foumelis in collaboration with Prof. A. Hooper to automate the pre-processing of Sentinel-1 SLC data and their preparation for ingestion to StaMPS (Serco Italia SPA 2020). However, not only Sentinel-1 data could be processed within StaMPS, but other data like TerraSAR-X too. In this regard, the original SNAP2StaMPS package was complemented with a new package called “TSX2StaMPS”, which focuses on the preprocessing of high-resolution TerraSAR-X data for the use in StaMPS. Since some of the processing steps and explanations contain parts of the SNAP2StaMPS package and tutorials, these abstracts are marked here with a red star (**\***) and refer to the following SNAP2StaMPS tutorial:

Serco Italia SPA (2020). *SNAP2StaMPS: Data preparation for StaMPS PSI processing with SNAP - Mexico City 2020 (version 1.1).* Retrieved from RUS Lectures at [https://rus-](https://rus-copernicus.eu/portal/the-rus-library/learn-by-yourself/) [copernicus.eu/portal/the-rus-library/learn-by-yourself/](https://rus-copernicus.eu/portal/the-rus-library/learn-by-yourself/)

# Data download

To download high-resolution TSX data, register as a new user at the EOWEB GeoPortal hosted by the German Aerospace Center (DLR): <https://eoweb.dlr.de/egp/>.

According to your user privileges, select the data (platform, start time, end time, geographical region) you want to download and put the selected scenes in the cart.



<https://eoweb.dlr.de/guestegp/main#mainWindowtabExplore>

After ordering the data, you can download the provided “.tar.gz” files by using FTPS. For further information see the following link for downloading data provided by DLR, e.g. via FileZilla: <https://eoweb.dlr.de/egp/docs/user/downloading_ordered_data.html>

# Download and installation

The TSX2StaMPS package is part of Snap2StaMPS, which can be downloaded via the following link on GitHub:

https://github.com/mdelgadoblasco/snap2stamps

To install the package, follow the instructions given in the Snap2StaMPS repository on Github. TSX2StaMPS is also available as standalone package:

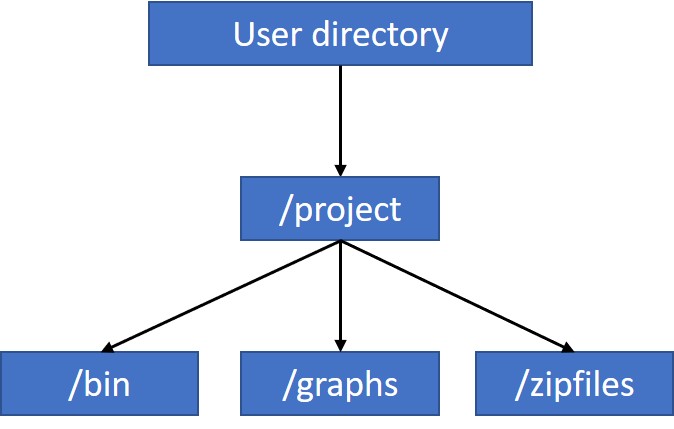
<https://github.com/jziemer1996/TSX2StaMPS.git>

For further information, please check out the information and tutorial provided in this repository.

# Step by step

#### Package structure

The package is structured as followed:



TSX2stamps consists of a project folder, in which three subdirectories can be found, ***“bin”***, ***“graphs”*** and ***“zipfiles”***. The downloaded TSX scenes can be downloaded and moved into the ***“zipfiles”*** directory. In the ***“bin”*** folder, you`ll find all processing steps provided as python scripts and a project\_stripmap.conf file, where all processing parameters can be defined by the user:

* **Project\_stripmap.conf**

#### stripmap\_step\_0a\_unpack\_sar\_scenes.py

#### stripmap\_step\_0b\_secondaries\_prep.py

#### stripmap\_step\_1\_subset\_sar.py

#### stripmap\_step\_1b\_masterselection.py

#### stripmap\_step\_2\_coreg\_sar.py

#### stripmap\_step\_3\_ifg\_sar.py

#### stripmap\_step\_4\_plotting\_all.py

#### stripmap\_step\_5\_stamps\_export.py

The python scripts can be executed via cmd or Windows Powershell. The stripmap\_step\_5\_stamps\_export.py script is the last step of the workflow and prepares the preprocessed data for ingestion in StaMPS. The scripts are based on SNAP graphs for the processing of the TSX scenes. These XML graphs are stored in the /graphs folder and are ordered according to the processing workflow:

* **1) stripmap\_TSX\_Subset.xml**
* **2) stripmap\_DEM\_Assisted\_Coregistration(\_extDEM).xml**
* **3) stripmap\_Interferogram\_TopoPhase(\_extDEM).xml**
* **4) stripmap\_plot\_[...].xml**
  + **[...]: “split”, “coreg” or “ifg”**
* **5) stripmap\_Export.xml**

*The Following part is directly taken from the* [*SNAP2StaMPS - manual*](https://github.com/mdelgadoblasco/snap2stamps/blob/master/Manual/SNAP2StaMPS_User_Manual.pdf) *and StaMPS manual.****\****

StaMPS is compatible with the output generated by SNAP after the version 6.0. SNAP allows the user to define a series of xml files that contain user-defined processing workflow by using its Graph Builder. These files can be used to run SNAP processing in batch mode by using the GPT command (Graph Processing Tool).

[TSX2stamps](https://github.com/mdelgadoblasco/snap2stamps) contains a set of graphs, together with python wrappers that allow you to automatise the processing chain for single master interferograms compatible with StaMPS PSI.

#### Hardware requirements\*

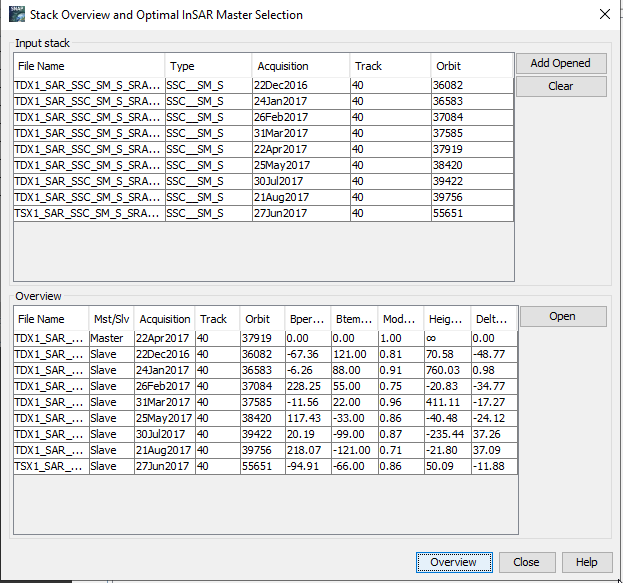
There are no specific hardware requirements but note that TerraSAR-X data are quite large, and their processing requires significant resources. For example, the interferogram generation step, which is the most computationally demanding will likely require a machine with a minimum of 16 GB RAM.

#### Master selection\*

Before we start with the pre-processing of the data, we need to select an optimal master image. The master image is selected such that the distribution of the perpendicular baseline values is as low as possible as well as maximizing the (expected) stack coherence of the interferometric stack. Selection of the "optimal" master should lead to improved visual interpretation of the interferograms and assist quality assessment.

SNAP contains a tool to perform the optimal master selection for us while also providing the overview of the temporal and perpendicular baselines of all the products with respect to it.

Go to **Radar**  **Interferometric**  **InSAR Stack Overview** and click **Add Opened** to load all products (they will not be ordered by date, but it is not important). Then click **Overview**.

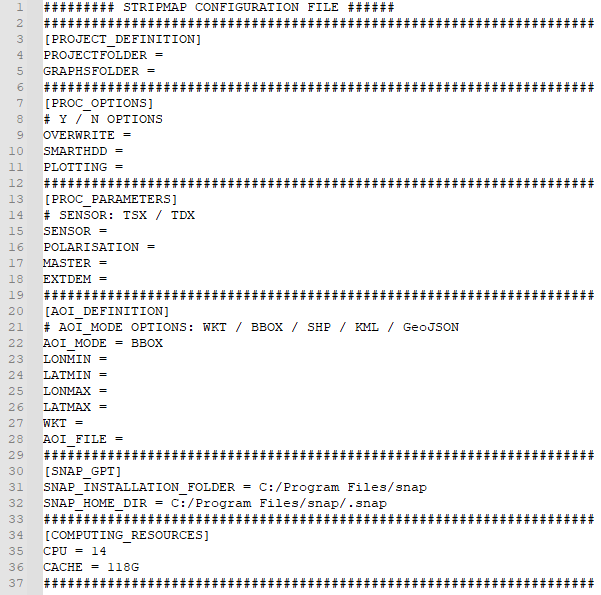


In the Mst/Slv column you can see, which TSX scene was selected as master. Please keep that scene in mind for later processing!

#### TSX2StaMPS project configuration\*

To run the next steps in a more automated mode the TSX2stamps package includes a ***project.conf*** file where all necessary user inputs are defined. Let’s now set it up.

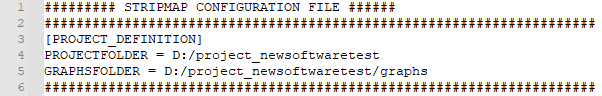
You can find the file in the TSX2stamps **“bin”** folder and have to edit it according to your data and paths. Right-click ***project.conf*** and select ***Open in Mousepad.*** At this moment all the settings are empty.



In the **PROJECT DEFINITION** section, we set the default project folder to the ***Processing*** folder and set the graphs folder to the TSX2stamps graph location.

#### PROJECTFOLDER=/…/Project

#### GRAPHSFOLDER=/…/Project/graphs





TIP: Your paths must not contain any spaces if they do, enclose them into quote marks “path”.

In the **PROCESSING OPTIONS** section, you can specify if you want to overwrite the results, if you’re using a smart HDD and if you want to plot some of the results. The plotting option refers to the question if you generally want to plot the results, but not which of them. You can decide this later on!

#### 

In the **PROCESSING PARAMETERS** section, you can choose the sensor type from which you want to process the data (TDX or TSX). Further, the polarization can be chosen. In line 17, you have to specify the name of your subsetted master scene. This is important since a master scene is needed for coregistration. However, the software will automatically ask in step 4) “stripmap\_step\_1b\_masterselection.py”, which scene you want to select as master image. A new folder called “master” will automatically created in your project folder and the selected master scene will be moved. You only have to specify the name of this file in the project\_stripmap.conf file. Note that all files resulting from the subsetting look as follows: yyyymmdd\_sub.dim. Consequently, this naming convention must also be overtaken in the project\_stripmap.conf file.

#### MASTER=/…/Project/master

#### Further, you can use an external DEM, which you can add in the line below (line 18).

#### Note: This tutorial is based on the processing with an External DEM with high spatial resolution and was not tested for the auto-download of the SRTM DEM. We strongly recommend using an External DEM in higher spatial resolution than SRTM for TSX/TDX data (if possible) since these coarse resolution DEMs may lead to inaccuracies in the results.

#### 

In the **AOI BBOX DEFINITION** you can define a spatial subset of the area of interest. You can choose between different AOI modes you want to use. The standard is to define a bounding box (BBOX). Further options to define the area of interest are available including WKT, SHP, KML and GeoJSON. The polygon of interest can be specified using the WKT parameter, the AOI\_FILE supports the abovementioned data types.

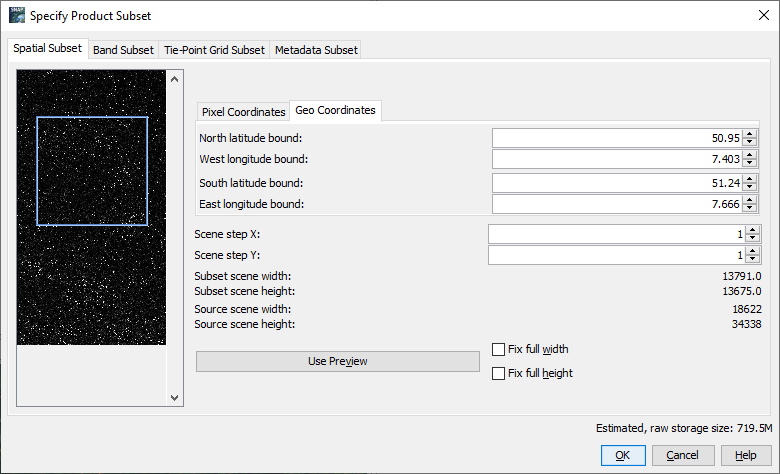
When using the bounding box, four parameters, ***Latmin, Lonmin, Latmax*** and ***Lonmax*** have to be specified. The subset is based on geographical coordinates. To find out the extend of your subset, go to SNAP —> **Graph Builder —> Right click —> Add —> Raster —> Subset.** There you can specify the extend of the SAR scene to be subsetted.

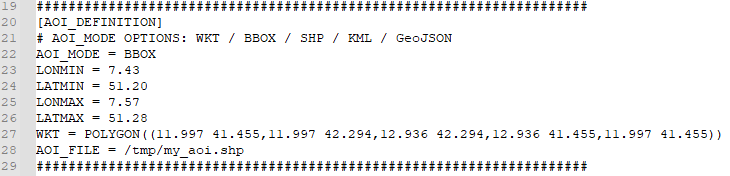
#### LONMIN (West longitude bound) = 7.403

#### LATMIN (North latitude bound) = 50.95

***LONMAX (East longitude bound) = 7.666***

***LATMAX (South latitude bound) = 51.24***





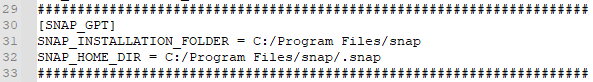


TIP: Be careful to copy minus signs correctly and remove all spaces.

In the SNAP GPT path, we need to point to the SNAP installation.

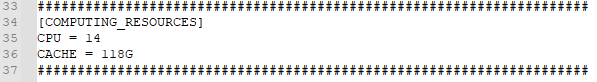
SNAP\_INSTALLATION\_FOLDER=***/…/snap/***

SNAP\_HOME\_DIR=***/…/snap/.snap***



Finally, in the COMPUTING RESOURCES TO EMPLOY section, we can set the CPU number and Cache size to use (See  NOTE 4). This needs to be selected based on your VM/PC:

CPU=12 CACHE=32G



Now, go to **File** —> **Save** and save the changes made to the ***project\_stripmap.conf*** file. We can now start with the automated processing using the python scripts.

#### Unpack TSX scenes

In the first step, the downloaded TSX scenes have to be unpacked in the /unzip folder. Finally, the unpacked scenes are moved to a newly created folder called ***“slaves”***, where further processing is done. The first step is executed using the *stripmap\_step\_0a\_unpack\_sar\_scenes.py* script provided in the /bin folder. To execute the script, paste the following command in the terminal.

python3 stripmap\_step\_0a\_unpack\_sar\_scenes.py –F project\_stripmap.conf

When the processing is completed, the bin path ending with $ will appear. Leave the terminal window open and check the ***“slaves”*** folder.

#### Slaves preparation\*

In the next step, we need to divide the slave images into folders with the name corresponding to the acquisition date in format ***<yyyymmdd>***. This is necessary for the subsetting in step 3. To call the second python script paste the following command in the terminal.

python3 stripmap\_step\_0b\_secondaries\_prep.py –F project\_stripmap.conf

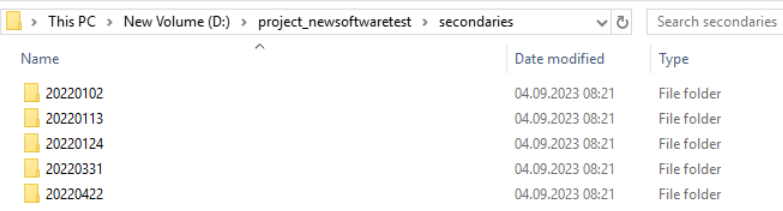
Then press **ENTER** to run the command. *The processing will take few seconds depending on your VM (here: 16 cores; 128 GB RAM).*



TIP: **The TDX2stamps requires Python 2.7.** You can type “python -V” and “python2 -V” commands to your command line to find the associated version.

When the processing is completed, the bin path ending with $ will appear again. Leave the terminal window open and check, if the ***“slaves”*** folder contains a folder for each slave image.

#### /…/project/secondaries



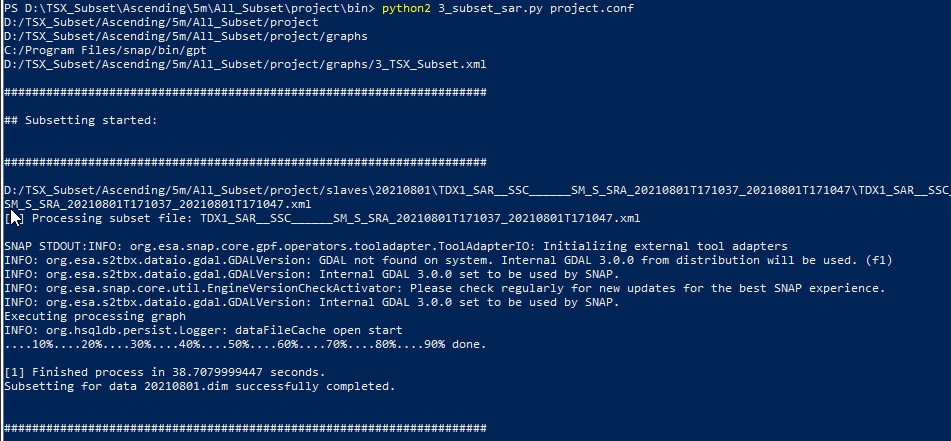
#### Subsetting

Next, you could create a subset of the downloaded TSX scenes. This is not mandatory, but is recommended, since it can significantly reduce the processing time of the following steps. Again, this step is run in the terminal window by using the *stripmap\_step\_1\_subset\_sar.py* script which refers to the SNAP graph “stripmap\_TSX\_Subset.xml” (predefined in the ***“graphs”*** folder). The python script will automate the processing by looping over the slave images and updating the input and output accordingly and then running the graph for each slave image. It will create a new folder named “subset” in the project directory, where all subsetted files are stored in format ***<yyyymmdd>***, with the extent specified in the AOI BBOX DEFINITION mentioned above.

Go back to the **terminal** window and run the following command.

python3 stripmap\_step\_1\_subset\_sar.py –F project\_stripmap.conf

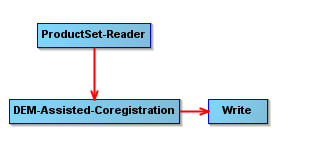
Then press **ENTER** to run the command. When the processing finishes successfully, something like the message below should appear. *The processing will take approximately 40 seconds per image depending on your subset and VM (here: 16 cores; 128 GB RAM).*



In the end of this process, don’t forget to insert the name of your master scene (see section 4.3) in line 17 of the project\_stripmap.conf file. The correct folder content is important here, since each slave is coregistered with the master scene in the following step.

#### Master-slave coregistration\*

In the next step we need to co-register each subsetted slave image with the master image. You can see the workflow below.



* + 1. DEM-Assisted-Coregistration

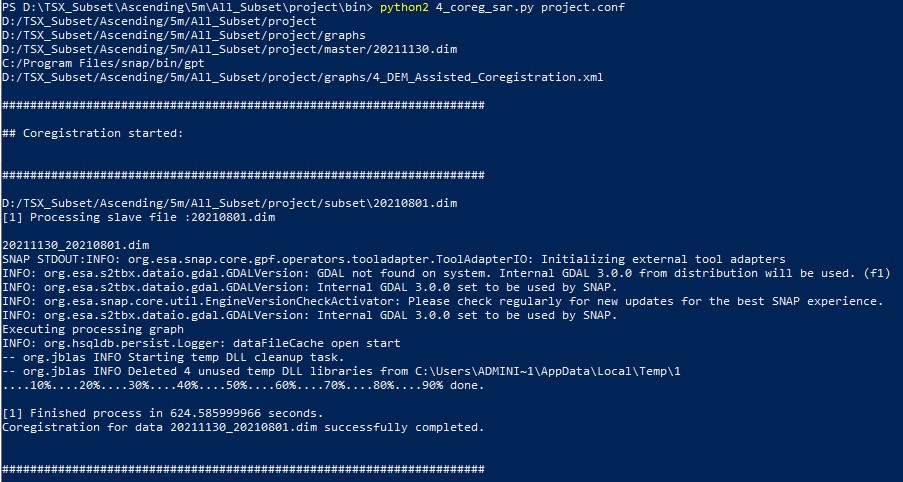
This operator co-registers two TSX products (master and slave) using a Digital Elevation Model (DEM) *(for further information see SNAP Help).* Note, that in this package, an external DEM is used as a default. To specify your DEM you want to use, open the “stripmap\_DEM\_Assisted\_Coregistration(\_extDEM).xml” graph in the ***“graphs”*** folder, and paste the folder path, where the DEM is stored and the name of the DEM into line 11. However, you could use the DEMs provided via Auto Download in SNAP. If so, the XML graph needs to be adjusted accordingly (lines 9-11). The script will be executed by using the *stripmap\_step\_2\_coreg\_sar.py* script within the bin folder.

* + 1. Run

Go back to the **terminal** window and run the following command. Note that this is one of the most time- demanding steps in the pre-processing chain. When the processing is completed, a new folder ***“coreg”*** is created in the ***Project*** folder. This folder contains the coregistered and debursted products.

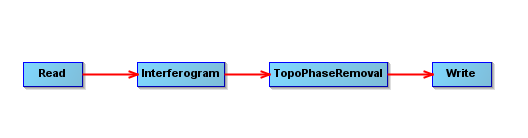
python3 stripmap\_step\_2\_coreg\_sar.py –F project\_stripmap.conf

Press **ENTER** to run the command. When the processing finishes successfully, something like the message below should appear. *The processing will take approximately 10 minutes per slave image depending on your subset and VM (here: 16 cores; 128 GB RAM).*



#### Interferogram generation\*

Finally, interferograms have to be generated from the coregistered data pairs created in the previous step. This will be done by using the *stripmap\_step\_3\_ifg\_sar.py* script in the bin folder, which corresponds to the “stripmap\_Interferogram\_TopoPhase(\_extDEM).xml” graph located in the graphs folder.

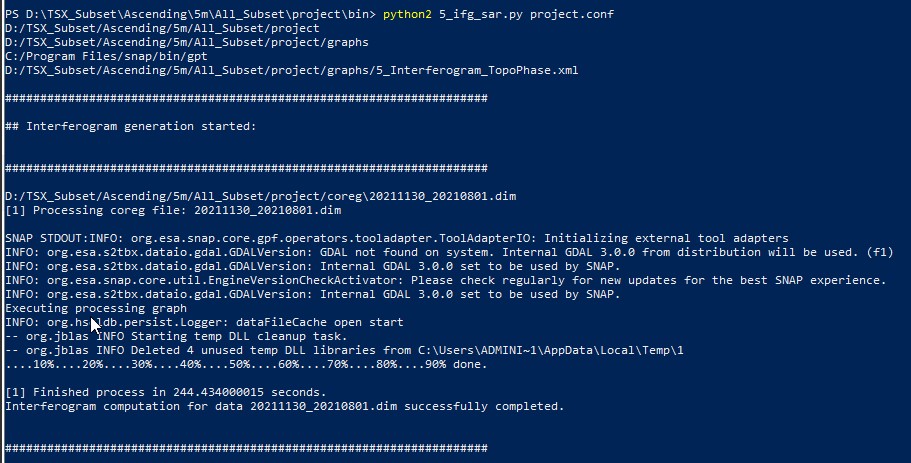


In the first step, the interferograms will be created using the files stored in the ***“coreg”*** folder. Finally, the topographic phase will be removed using the (external) DEM specified above. Again, you need to change the folder path and name of your DEM in line 42 of the “stripmap\_Interferogram\_TopoPhase(\_extDEM).xml” graph. As mentioned above, you don’t have to use your own DEM here, since a bunch of DEMs are provided via the Auto Download function in SNAP. If so, the XML graph needs to be adjusted accordingly (lines 41-42). When the processing is completed, a new folder ***“ifg”*** is created in the ***Project*** folder. This folder contains the interferogram with the topographic phase removed. The products in both folders are named ***<masterDate\_slaveDate.dim.***

Go back to the **terminal** window and run the following command.

python3 stripmap\_step\_3\_ifg\_sar.py project\_stripmap.conf

Then press **ENTER** to run the command. Leave the terminal window open. When the processing finishes successfully, something like the message below should appear. *The processing will take approximately 5 minutes per coregistered image-pair depending on your subset and VM (here: 16 cores; 128 GB RAM).*



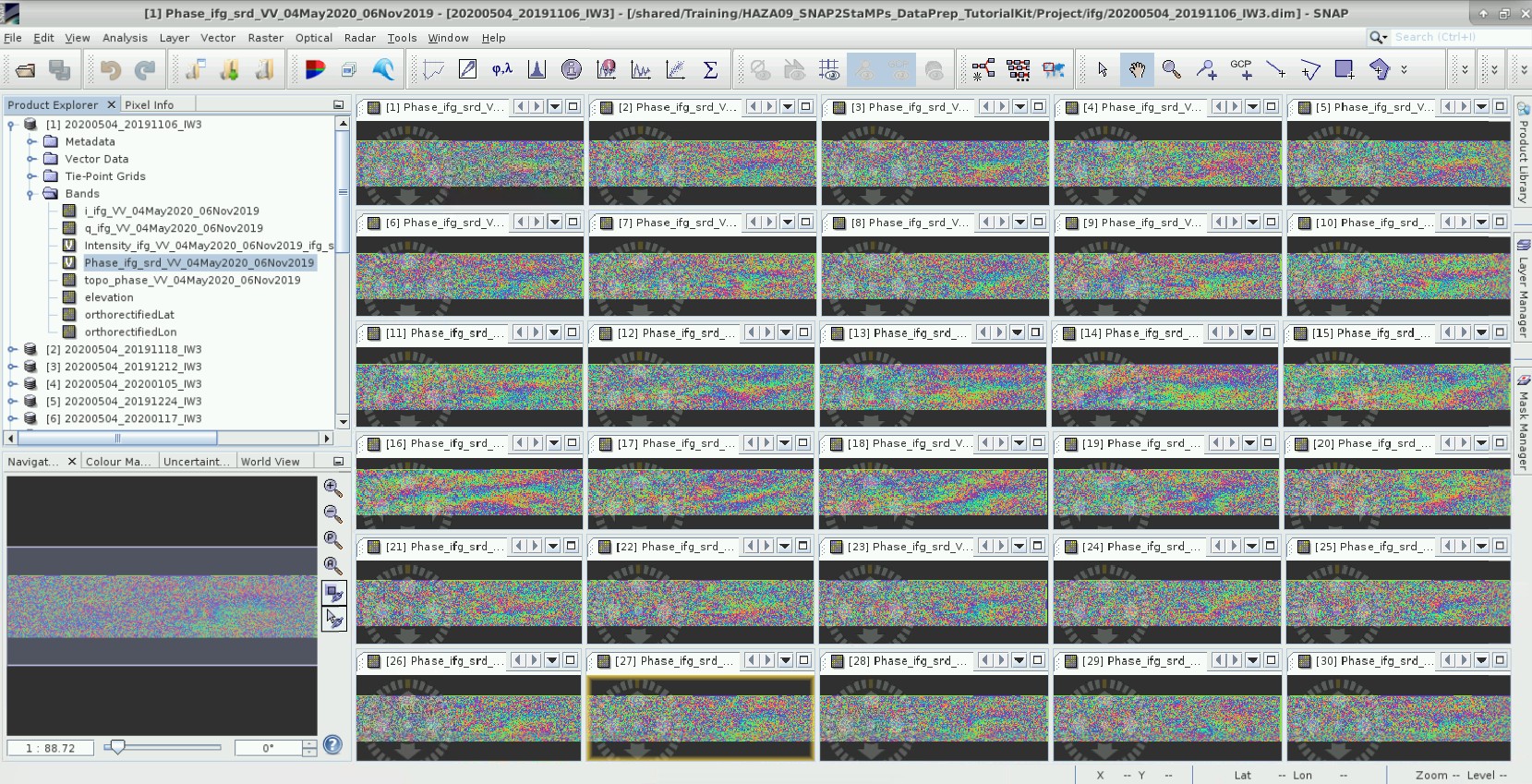
* + 1. Check the interferograms

Now, before the files are exported we need to test if all the interferograms have been correctly completed. Open SNAP graphical interface and load all the products:

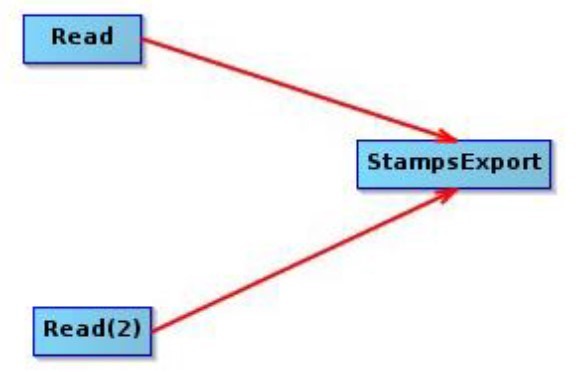
#### /…/project/ifg/

Then open the ***“Phase\_ifg\_\*\*\*\*\*”*** band for each product. You do not need to open all at once as below, you merely need to check for empty interferograms.

In the image below you can see that all interferograms have been correctly created. If in your dataset you find an empty interferogram note the name and consequently remove the file from the ***“coreg”*** and ***“ifg”*** folders.



#### StaMPS export\*



Now we need to prepare the data into a StaMPS compatible format. This is done by using a SNAP tool for StaMPS export.

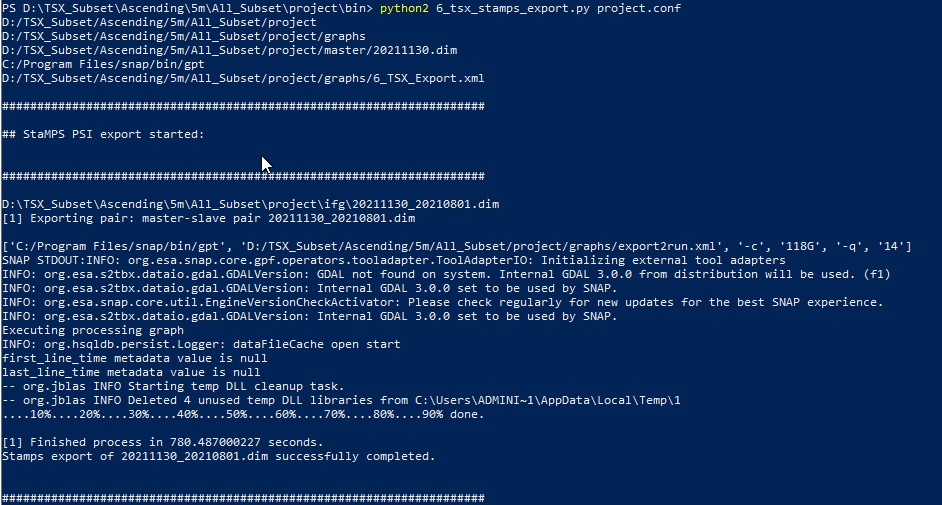
The inputs are:

* the coregistered master-slave pair
* its corresponding interferogram with the elevation and orthorectified latitude and longitude bands

Go back to the **terminal** window and run the following command.

python3 stripmap\_step\_5\_stamps\_export.py project\_stripmap.conf

Then press **ENTER** to run the command. When the processing finishes successfully, something like the message below should appear. *The processing will take again quite long time - approximately a 15 minutes per slave image depending on your VM (here: 16 cores; 128 GB RAM).*



When the processing is completed, a new folder named ***“INSAR\_<masterDate>”*** was created in the ***Project*** folder. It contains the final output structure - four folders: ***rslc***, ***diff0***, ***geo*** and ***dem***. When the processing was finished successfully, you can close your command line interface and start with StaMPS PSI processing.

#### 4.11 Plotting the results (optional)

#### You can plot the results of the subsetting, the coregistration and the interferogram generation by using the

#### stripmap\_step\_4\_plotting\_all.py script. These scripts can be executed as follows:

* Plot the subsets:

python3 stripmap\_step\_4\_plotting\_all.py –F project\_stripmap.conf –M ‘split’

* Plot the coregistered data:

python3 stripmap\_step\_4\_plotting\_all.py –F project\_stripmap.conf –M ‘coreg’

* Plot the phase of the generated interferograms (as greyscale):

python3 stripmap\_step\_4\_plotting\_all.py –F project\_stripmap.conf –M ‘ifg’

The plotted images can be found in new folders called “plot\_split”, “plot\_coreg” and “plot\_ifg” in the project folder of your project.

#### 4.11 Auto-run TSX2stamps (optional)

Since running the software step by step can be time-consuming, the new version allows to automatically run all scripts of the Snap2StaMPS workflow, either using Sentinel-1 or TSX data. The config files have to be modified in the same way as explained in section 4.4 To run all the scripts in one command, the auto\_run.py script in the /bin folder has to be executed as follows:

python3 auto\_run.py –F project\_stripmap.conf

# References

#### Software and Data download

**German Aerospace Center (2022).** EOWEB GeoPortal (EGP). <https://eoweb.dlr.de/egp/>

**SNAP2StaMPS Manual and download** - https://github.com/mdelgadoblasco/snap2stamps

[**StaMPS Manual**](https://homepages.see.leeds.ac.uk/~earahoo/stamps/StaMPS_Manual_v4.1b1.pdf)[**StaMPS Download**](https://github.com/dbekaert/StaMPS)

#### PSI resources

**Crosetto, M., Monserrat, O., Cuevas-González, M., Devanthéry, N., & Crippa, B. (2016). Persistent Scatterer Interferometry: A review***. ISPRS Journal of Photogrammetry and Remote Sensing*, 115, 78–89. <https://doi.org/10.1016/j.isprsjprs.2015.10.011>

**Delgado Blasco, J. M., Foumelis, M., Stewart, C., & Hooper, A. (2019). Measuring Urban Subsidence in the Rome Metropolitan Area (Italy) with Sentinel-1 SNAP-StaMPS Persistent Scatterer Interferometry**. *Remote Sensing*, 11(2), 129. <https://doi.org/10.3390/rs11020129>

**Jia, H., & Liu, L. (2016). A technical review on persistent scatterer interferometry**. *Journal of Modern Transportation*, 24(2), 153–158. <https://doi.org/10.1007/s40534-016-0108-4>

#### Tutorials

**GIS-Blog** - Matthias Schlögl – Using StaMPS/MTI for PSI Analysis (post series) - [https://www.gis-](https://www.gis-blog.com/stamps-1/) [blog.com/stamps-1/](https://www.gis-blog.com/stamps-1/)

1. **Hooper** – StaMPS Persistent Scatterer Exercise (2015) – ESA Land Training Course <http://seom.esa.int/landtraining2015/files/Day_4/D4P2a_LTC2015_Hooper.pdf>

**Serco Italia SPA (2020).** SNAP2StaMPS: Data preparation for StaMPS PSI processing with SNAP - Mexico City 2020 (version 1.1). Retrieved from RUS Lectures at <https://rus-copernicus.eu/portal/the-rus-library/learn-by-yourself/>