User's Guide to COSI-CORR Python code
Co-registration of Optically Sensed Images and Correlation
Correlation 3D Example

Marie Cadoux, Saif Aati, Jean-Philippe Avouac

California Institute Of Technology

1200 East California Blvd, Pasadena, CA 91125, USA

July, 2023

Abstract

This document is a user's guide to creating a 3D correlation map from an example. All the input parameters and results are presented to help the user to understand the code. An example is treated to measure the ground surface displacement of a coseismic deformation: Izmit earthquake. This code is also adapted to measure the ground surface deformation of a slow landslide, an ice flow and a sand dune migration.

Contents

1. Introduction

Correlation_3D_example is a file containing an example of code allowing us to get a 3D correlation map, in a field of geology: a coseismic deformation (Izmit Earthquake). This code is also adapted to treat a sand dune migration, a slow landslide and an ice flow. With the 3D correlation map, we get the horizontal ground surface deformation, East/West and North/South and the vertical ground surface deformation. For this example, all the input parameters and the results are present. The goal of this folder is to help the user with the first uses of the code.

To get a 3D displacement map it is necessary to follow the workflow below (Figure 1), written by (Aati et al., 2022).

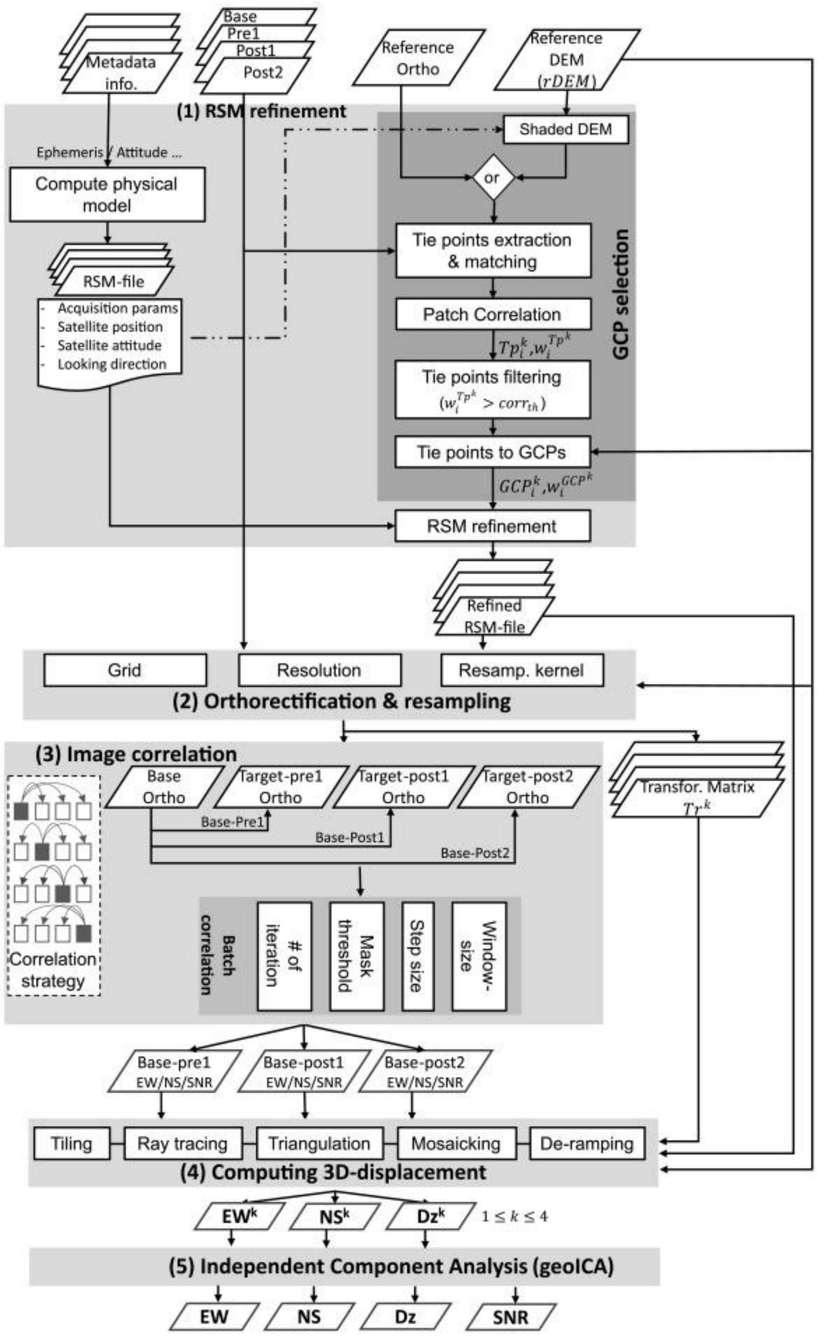

Figure 1: Workflow to get the 3D correlation map

Four no orthorectified images (raw images) in minimum (or a multiple of 4) that come from the same kind of sensor (Spot, Sentinel, Landsat…) with their metadata (information about the imaging system), a reference orthorectified image and a DEM are required. To reduce the artifacts, we advise using a reference image with the same resolution as the raw images or a reference image that comes from the same sensor as the raw images.

2. Structure of the example

Correlation_ 3D_example contains two files:

- example_izmit: contains the code and the inputs parameters,
- results.

3. Code operation

izmit_end_to_end_workflow.py is the code that enables us to get the 3D correlation map. The code is organized into different parts, (Figure 2):

- Import classes and functions
- Parameters
- Assignment of parameters
- Workflow

```python
# Import classes and functions
import ...

# Parameters
folder = os.path.dirname(__file__)
dataset_dir = os.path.join(folder, 'Spot_Data')
raw_img_list = ExtractSubfiles(dataset_dir, fileExtension=[".TIF"])
config_file = os.path.join(folder, 'geo_3DDA_config.yaml')
dem_path = os.path.join(folder, "REF_DATA/SRTM_DEM.tif")
ref_ortho = os.path.join(folder, "REF_DATA/rOrtho_1999-07-Spot4.tif")
workspace_dir = os.path.join(SOFTWARE.WKDIR, "3DDA_WS_IZMIT")
sensor = SENSOR.SPOT1_5
event_date = "1999-08-17"  # YYYY/MM/DD""
ortho_gsd = 10

# Assignment of parameters
izmit = GeoCosiCorr3DPipeline(img_list=raw_img_list,
                              sensor=sensor,
                              event_date=event_date,
                              dem_file=dem_path,
                              ref_ortho=ref_ortho,
                              config_file=config_file,
                              workspace_dir=workspace_dir)

# Workflow
# data_file = izmit.data_file
data_file = izmit.build_rsm_data_file()
izmit.compute_footprint(data_file)
izmit.feature_detection(data_file=data_file)
izmit.gcp_generation(data_file=data_file)
izmit.rsm_refinement(data_file)
izmit.orthorectify(data_file, ortho_gsd=ortho_gsd)
prePostFile = izmit.compute_pre_post_pairs(data_file, pre_post_overlap_th=80)
izmit.correlate() # optional
izmit.generate_3DDA_sets(data_file)
izmit.correlate(corr_mode='set')
izmit.compute_3DD(data_file)
```

Figure 2: Code to get the 3D correlation map

We will explain these parts below.

3.1 Import classes and functions

The first part of the code consists of importing classes and functions that enable running izmit_end_to_end_workflow.py (Figure 3).

```python
# Import classes and functions
from geoCosiCorr3D.georoutines.file_cmd_routines import ExtractSubfiles
from geoCosiCorr3D.geoCore.constants import *
from geoCosiCorr3D.geoCosiCorr3D_scripts.geoCosiCorr3D_end_2_end_pipeline import GeoCosiCorr3DPipeline
```

Figure 3: Import of classes and functions to run the correlation code

3.2 Parameters

The second part of the code consists of writing the input parameters to get the 3D correlation map (Figure 4). All the input parameters are mandatory to run the code.

1. **folder = os.path.dirname(__file__):** assigns a file to "folder", where there are all the input parameters. In this example, it assigns to "folder" the "example_izmit" file.

2. **dataset_dir = os.path.join(folder, 'Spot_Data')**: assigns "Spot_Data" file to "dataset_dir". "Spot_Data" contains four images with their metadata that will be correlated: the "raw images". "Spot_Data" must be in "example_izmit" file. If the name of the file that contains the raw images is different, modify the writing in quotation marks.

3. **raw_img_list = ExtractSubfiles(dataset_dir, fileExtension=[".TIF"]:** assigns all the files in "Spot.Data" with the TIF format to "raw_img_list". The goal is to create a list with the four raw images. If the raw image format is different, modify the writing in quotation marks.

4. **config_file = os.path.join(folder, 'geo_3DDA_config.yaml'):** assigns the document 'geo_3DDA_config.yaml' to "config_file". In this document, is written all the specific input parameters of each function used in the workflow. We can modify these input parameters. 'geo_3DDA_config.yaml' must be in "example_izmit" file.

5. **dem_path = os.path.join(folder, "REF_DATA/SRTM_DEM.tif"):** assigns the DEM, called "SRTM_DEM.tif" here, to "dem_path". Here the DEM is in REF_DATA file which is in "example_izmit" file. If the path or the names of the files are different, modify the writing in quotation marks.

6. **ref_ortho = os.path.join(folder, "REF_DATA/rOrtho_1999-07-Spot4.tif"):** assigns the reference image, called "rOrtho_1999-07-Spot4.tif" here, to "ref_ortho". Here the reference image is in "REF_DATA" file which is in "example_izmit" file. If the path or the names of the files are different, modify the writing in quotation marks.

7. **workspace_dir = os.path.join(SOFTWARE.WKDIR, "3DDA_WS_IZMIT"):** creates the folder "3DDA_WS_IZMIT" in the folder "GEO_COSI_CORR_3D_WD". All the results of the code will be stocked in "3DDA_WS_IZMIT". We can change the name of the folder by modifying the writing in quotation marks.

8. **sensor = SENSOR.SPOT1_5:** assigns the sensor "SPOT1_5" to "sensor", because, for this example, we use three SPOT 4 images and one SPOT 2 image. If you use images that come from another sensor, modify the writing in capital letters after the point. To know what kind of sensor you can use, refer to the code constant.py (geoCosiCorr3D/geoCore/constants).

9. **event_date = "1999-08-17":** assigns the date of the event in the format "YYYY-MM-DD" to "event_date". For this example, Izmit Earthquake occurred on 17/08/1999. If you treat another example, change the date in quotation marks.

10. **ortho_gsd = 10:** affects the Ground Sampling Distance (GSD) to orthorectify the images. It is possible to choose a GSD inferior to the resolution of the raw images but the orthorectified images got, will have more artifacts. In this example, Spot 2 and 4 images have a resolution of

10m so we can choose at least the GSD of 10 to get orthorectified images without adding artifacts.

```python
# Parameters
folder = os.path.dirname(__file__)
dataset_dir = os.path.join(folder, 'Spot_Data')
raw_img_list = ExtractSubfiles(dataset_dir, fileExtension=[".TIF"])
config_file = os.path.join(folder, 'geo_3DDA_config.yaml')
dem_path = os.path.join(folder, "REF_DATA/SRTM_DEM.tif")
ref_ortho = os.path.join(folder, "REF_DATA/rOrtho_1999-07-Spot4.tif")
workspace_dir = os.path.join(SOFTWARE.WKDIR, "3DDA_WS_IZMIT")
sensor = SENSOR.SPOT1_5
event_date = "1999-08-17"  # YYYY/MM/DD""
ortho_gsd = 10
```

Figure 4: Parameters for the correlation

3.3 Assignment of parameters

The third part consists of assigning the input parameters to GeoCosiCorr3DPipeline class (Figure 5). This class is in GeoCosiCorr3D/geoCosiCorr_3D_scripts/geoCosiCorr3D_end_2_end_pipeline and it is composed of all the functions used for the workflow. Don't modify the names of the input parameters. In this example, we have called GeoCosiCorr3Dpipeline class, "izmit". If you treat another event, you can change this name.

```python
# Assignment of parameters
izmit = GeoCosiCorr3DPipeline(img_list=raw_img_list,
                              sensor=sensor,
                              event_date=event_date,
                              dem_file=dem_path,
                              ref_ortho=ref_ortho,
                              config_file=config_file,
                              workspace_dir=workspace_dir)

```

Figure 5: Assignment of the input parameters to GeoCosiCorr3Dpipeline class

3.4 Workflow

The fourth part consists of running the functions presented in GeoCosiCorr3DPipeline (Figure 6) class by following the workflow (Aati et al., 2022), (Figure 7). If you have called GeoCosiCorr3DPipeline class with another name than "izmit", don't forget to replace "izmit" with the new name everywhere.

```
# Workflow
# data_file = izmit.data_file
data_file = izmit.build_rsm_data_file()
izmit.compute_footprint(data_file)
izmit.feature_detection(data_file=data_file)
izmit.gcp_generation(data_file=data_file)
izmit.rsm_refinement(data_file)
izmit.orthorectify(data_file, ortho_gsd=ortho_gsd)
prePostFile = izmit.compute_pre_post_pairs(data_file, pre_post_overlap_th=80)
izmit.correlate() # optional
izmit.generate_3DDA_sets(data_file)
izmit.correlate(corr_mode='set')
izmit.compute_3DD(data_file)
```

Figure 6: Workflow to get the 3D correlation map

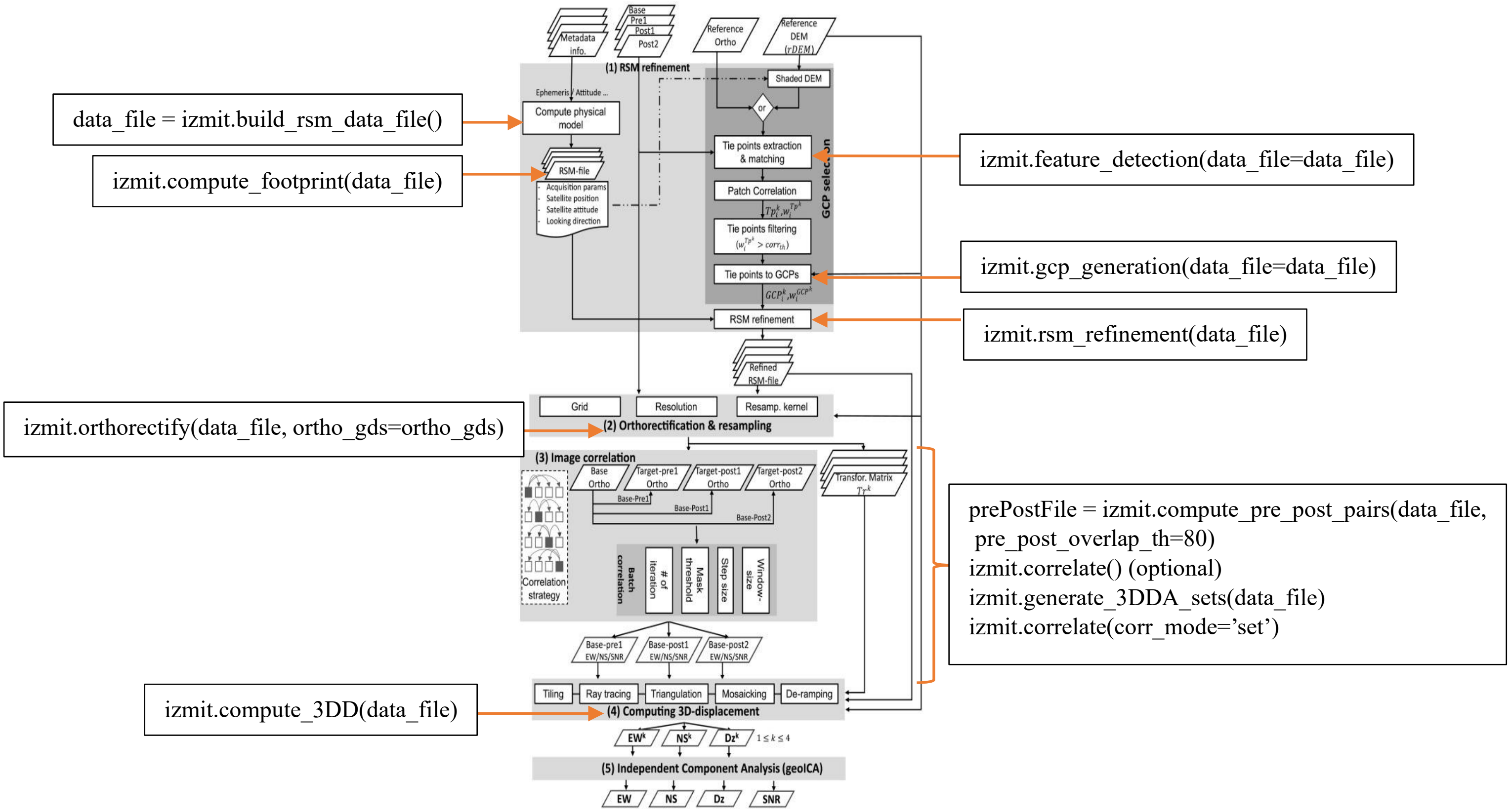

Figure 7: Code functions in relation to the workflow

4. Results

All the results for each function that follows the workflow are stocked in GEO_COSI_CORR_3D_WD/3DDA_WS_IZMIT. We will explain the result for each function:

1. **data_file = izmit.build_rsm_data_file():** creates a "DataFile.csv" file and a "RSMs" folder:
- "RSMs" folder contains files with information about the four images and the imaging systems of the four images.

- "DataFile.csv" file contains information on the imaging systems (column Name, Date, Time, Platform, GSD), the paths of the four images (column ImgPath), the paths of metadata (column DIM) which have information about the images and the imaging systems and the documents in RSMs file (RSM), (Figure 8).

	A	B	C	D	E	F	G	H
1	Name	Date	Time	Platform	GSD	ImgPath	DIM	RSM
2	1999-10-05-09-04-57-Spot-4-HRVIR-1-M-10	1999-10-05	09:04:57	Spot-4-HR	10	/home/mcadou	/home/mca	/home/mcadoux/Pychar
3	2000-07-28-08-55-02-Spot-4-HRVIR-1-M-10	2000-07-28	08:55:02	Spot-4-HR	10	/home/mcadou	/home/mca	/home/mcadoux/Pychar
4	1999-07-25-08-49-39-Spot-4-HRVIR-2-M-10	1999-07-25	08:49:39	Spot-4-HR	10	/home/mcadou	/home/mca	/home/mcadoux/Pychar
5	1998-07-26-09-17-10-Spot-2-HRV-1-P-10	1998-07-26	09:17:10	Spot-2-HR	10	/home/mcadou	/home/mca	/home/mcadoux/Pychar

Figure 8: DataFile.csv

2. **izmit.compute_footprint(data_file):** creates a "Footprints" folder. In this folder, the code creates four files for the four images (Figure 9). In each file, there are the coordinates (longitude, latitude, altitude) of the image footprint in the geographic coordinate system of the image (CRS84 here). The code adds the path of the four files in "DataFile.csv" in the column "Fp".


```
{
"type": "FeatureCollection",
"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" } },
"features": [
{ "type": "Feature", "properties": { }, "geometry": { "type": "Polygon", "coordinates": [ [ [ 30.019794181969687, 41.108396309959282,
104.98307925183326 ], [ 30.882222154815491, 40.933950274835532, 704.513012899085879 ], [ 30.66316199745642, 40.418955767908834,
787.531059409491718 ], [ 29.802895662381236, 40.592537696199123, 944.295931684784591 ], [ 30.019794181969687, 41.108396309959282,
104.98307925183326 ] ] ] } }
]
}
```


Figure 9: 1998-07-26-09-17-10-Spot-2-HRV-1-P-10.geojson in "Footprints" file

3. **izmit.feature_detection(data_file=data_file):** creates a "Matches" folder. In this folder, the code creates four files for the four images (Figure 10). In each file, there is a list of the coordinates (X, Y) of tie points (common points) between the reference image (called "Base Image" in the file) and the raw images (called "Wrap Image" in the file). Each line corresponds to a tie point. The first column is the list of the X coordinates of the Base Image, the second column is the list of the Y coordinates of the Base Image, the third column is the list of the X coordinates of the Wrap Image and the fourth column is the list of the Y coordinates of the Wrap image. The Base Image coordinates and the Wrap Image coordinates are the position of the pixel with as origin in the upper left part of the image. COSI-Corr uses MicMac software to select automatically the tie points between the reference image and the raw images. The code adds the number of tie points for each raw image in "DataFile.csv" in the column "Tp" and the path of the four files in the column "MatchFile".

```
; COSI-Corr tie points file (from Micmac)
; base file:rOrtho_1999-07-Spot4.tif
; warp file:rOrtho_1999-07-Spot4.tif
; Base Image (x,y), Warp Image (x,y)
;
5463.200000     2724.760000     18.077560       2545.508000
5687.680000     2347.252000     136.379200      2128.400000
5865.400000     2044.240000     224.328800      1796.876000
5730.480000     2932.364000     314.940400      2694.808000
6184.120000     1512.896000     387.452000      1210.164000
6090.080000     2220.416000     474.100000      1922.984000
5483.160000     4987.840000     567.088000      4756.920000
6614.000000     987.252000      654.436000      606.872000
6424.360000     2114.260000     755.392000      1749.852000
6147.360000     3551.260000     846.084000      3213.816000
6795.880000     1469.176000     938.392000      1040.888000
5736.640000     5928.160000     1035.952000     5625.920000
5806.680000     6008.160000     1124.396000     5689.040000
```

Figure 10: rOrtho_1999-07-Spot4_VS_1999-07-25-08-49-39-Spot-4-HRVIR-2-M-10_matches.pts in "Matches" file

4. **izmit.gcp_generation(data_file=data_file):** transforms the tie points in GCPs (Ground Control Points). The difference between the tie points and GCPs is the coordinates. The GCPs have (X, Y, Z) projected in a geographic system. The code creates two documents for each raw image in the "Matches" file:

- Document with the csv format (Figure 11): contains information about each GCP,
- Document in png format: graph (Figure 12) where we can locate each GCP in relation to the DEM and the reference image. The elements are projected in EPSG: 4326 (WGS).

The code adds the path of the four documents in format csv, in "DataFile.csv", in the column "GCPs". Be careful, this code works if MicMac finds at least 20 tie points. You can modify these parameters into the function gcp_generation in GeoCosiCorr3Dpipeline (GeoCosiCorr3D/geoCosiCorr_3D_scripts/geoCosiCorr3D_end_2_end_pipeline).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		lon	lat	alt	xPix	yPix	weight	opti	dE	dN	dA	x_map	y_map	epsg	ref_img	dem	raw_img	gcp_id
2	0	30.117	40.855	303.6591	18.07756	2545.508	1	1	0	0	0	256972.33	4526628.341	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	ba742d90-d4e2-4003-ac29-cc357e3929cc
3	1	30.142	40.889	210.6034	136.3792	2128.4	1	1	0	0	0	259217.3	4530403.712	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	14c5218e-8192-4655-8884-cb3fe0d024aa
4	2	30.162	40.917	204.9172	224.3288	1796.876	1	1	0	0	0	260994.64	4533434.065	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	15851cf5-536b-496d-ae3a-377e8d8c0a6b
5	3	30.149	40.837	260.7916	314.9404	2694.808	1	1	0	0	0	259645.33	4524552.141	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	7eb6e35d-ed07-4e5b-8216-026fcd657e02
6	4	30.198	40.966	177.0825	387.452	1210.164	1	1	0	0	0	264182.08	4538747.914	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	70b7523c-328b-4b43-970e-cf6d3a909d27
7	5	30.189	40.902	130.6593	474.1	1922.984	1	1	0	0	0	263241.61	4531672.169	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	5933e19a-c126-40a1-a48d-495e687cd216
8	6	30.128	40.651	1192.457	567.088	4756.92	1	1	0	0	0	257171.94	4503995.798	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	b64061f9-c207-4c23-a2d8-9ec7587226b2
9	7	30.247	41.014	124.572	654.436	606.872	1	1	0	0	0	268481.21	4544004.759	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	65eee054-2788-4620-be58-452ef07647c7
10	8	30.228	40.912	122.7167	755.392	1749.852	1	1	0	0	0	266584.67	4532733.811	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	bdce1b9a-664a-48d6-a09e-23f46f923a84
11	9	30.201	40.782	347.66	846.084	3213.816	1	1	0	0	0	263814.45	4518362.704	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	48070e18-5e14-446c-a466-e715b0cea09a
12	10	30.27	40.972	132.6285	938.392	1040.888	1	1	0	0	0	270300.15	4539185.148	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	d88479ae-3258-4297-8d51-af88666fd53d
13	11	30.161	40.567	980.1533	1035.952	5625.92	1	1	0	0	0	259706.94	4494591.874	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	e38895ff-76eb-4ded-ae3f-bf4efad0c2e3
14	12	30.17	40.56	737.7395	1124.396	5689.04	1	1	0	0	0	260407.39	4493791.812	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	74fb602a-3b38-41c3-83d1-25af7fdae28d
15	13	30.285	40.899	122.2053	1226.732	1810.032	1	1	0	0	0	271292.63	4531127.007	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	162e6725-e3e3-40a7-9d81-96e0cca1820d
16	14	30.32	40.973	98.66479	1323.8	947.572	1	1	0	0	0	274492.48	4539256.553	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	8e64e56d-5c51-4b3e-8943-1846103c69ea
17	15	30.344	41.016	135.8037	1410.992	449.34	1	1	0	0	0	276676.24	4543882.109	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	e51fa6ef-d1f5-4e49-83c0-698e11bff831
18	16	30.306	40.854	75.71506	1501.624	2274.16	1	1	0	0	0	272907.15	4526064.817	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	931c7722-f965-4e6b-8556-576cbcd621b8
19	17	30.293	40.773	242.9199	1593.244	3177.924	1	1	0	0	0	271569.05	4517091.527	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	d523d756-4589-410b-a171-d6f672785afd
20	18	30.325	40.836	39.80498	1692	2440.008	1	1	0	0	0	274434.87	4524028.501	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	e23d1866-fa87-4ce9-b159-3f88f58e4dc5
21	19	30.321	40.78	75.36533	1797.936	3056.916	1	1	0	0	0	273942.43	4517834.424	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	c6c164e7-9252-49dd-ae93-d6cdcdc61728
22	20	30.388	40.953	286.8953	1899.096	1069.756	1	1	0	0	0	280122.91	4536826.886	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	0090b58d-ae43-4287-bde5-08f80e070fb2
23	21	30.348	40.786	23.99842	1998.212	2958.688	1	1	0	0	0	276239.01	4518350.623	32636	/home/mcadou	/home/mcado	/home/mcadoux/P	fbbfc66b-ad5f-4c28-96fa-3fee0e3f8c0d

Figure 11: rOrtho_1999-07-Spot4_VS_1999-07-25-08-49-39-Spot-4-HRVIR-2-M-10_matches_GCP.csv in" Matches" file

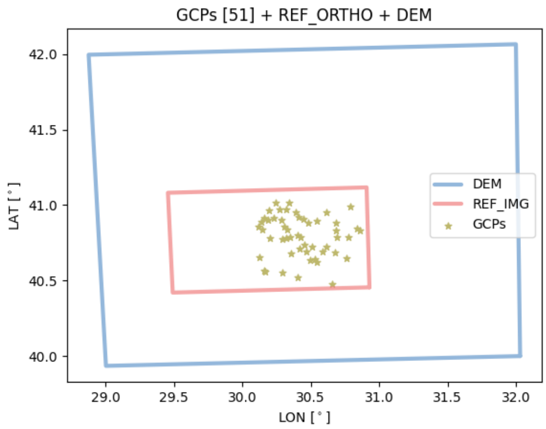

Figure 12: Ortho_1999-07-Spot4_VS_1999-07-25-08-49-39-Spot-4-HRVIR-2-M-10_matches_GCP.csv.png in "Matches" file

5. **izmit.rsm_refinement(data_file):** optimizes information about the imaging systems of the four raw images to reduce the error location of each GCP. By default, the code does three loops to optimize. It is possible to change the number of loops in geo_ortho_config.yaml or in geoCosiCorr3D/geoOptimization/gcpOptimization l74:
self.nb_loops = self.opt_params.get('nb_loops', 3)

The code creates an "RSM_Refinement" folder. In this folder, COSI-Corr creates four folders for the four raw images. In each folder, there are:

- rOrtho_1999-07-Spot4_VS_namerawimage_matches_GCP_opt.opt_report.csv: file containing the new coordinates corrected of each GCP, by loop with the error distribution. The coordinates are corrected thanks to the RSM or RFM refinement (Figure 13),
- rOrtho_1999-07-Spot4_VS_ namerawimage_matches _GCP_opt_loop_#.csv: files presenting the new coordinates corrected of each GCP and information used by the RSM or RFM refinement, by loop,
- rOrtho_1999-07-Spot4_VS_ namerawimage_matches_GCP_optloop_#_correction.txt: files containing the correction for each loop,
- RSM_gcp_patches: folder presenting all the patches for each GCP for each loop in the png and tif format,
- SRTM_DEM_32636.vrt and SRTM_DEM_32636_32635.vrt: files containing information on the DEM projected in a UTM grid.

	A	B	C	D	E	F	G	H
1	GCP_ID	Lon	Lat	Alt	nbLoop	dxPix	dyPix	SNR
2	9f6962c1-7f33-4f9d-827a-5120f4a9e9d9	29.9831892049	41.0088661288	117.286	0	1.531	0.76	0.986
3	c7adf851-e11a-44ba-8996-5abf115f94d3	29.9120554	40.8123680602	375.879	0	1.134	1.156	0.987
4	8556ea1a-3fd3-46de-a621-f7949d25e279	29.9291574769	40.8244768007	333.293	0	1.159	0.911	0.987
5	73753061-22a9-4a5e-9c69-91e6b8c1b3d	29.927564113	40.8074854527	194.183	0	0.982	0.965	0.985
6	a7889d4e-f0d8-47ba-9db4-089eb5efc4e8	29.9222541805	40.7776213951	200.505	0	1.168	1.12	0.983
7	fd63678f-3191-4e7b-af4b-ac96f87eaef9	29.9734741321	40.8741577704	324.918	0	1.178	0.884	0.984
8	220de7f1-5f18-4a23-a7ca-33ca9991a05a	29.9637919808	40.824603477	193.048	0	1.21	0.98	0.985
9	d179098d-3e08-431d-b722-e3ad7b5da67	29.9834270542	40.8419815162	300.934	0	1.206	0.999	0.987
10	4ffca92a-6247-45a7-8906-9705d4302a62	30.0282863366	40.9354444968	304.601	0	1.398	0.724	0.988

Figure 13: rOrtho_1999-07-Spot4_VS_1998-07-26-09-17-10-Spot-2-HRV-1-P-10_matches_GCP_opt.opt_report.csv in "RSM_Refinement/1998-07-26-09-17-10-Spot-2-HRV-1-P-10_METADATA" file

6. **izmit.orthorectify(data_file, ortho_gds=ortho_gds):** orthorectifies the four raw images in UTM grid. The code creates an "Orthos" folder and a "Trxs" folder.

- "Orthos" folder contains the four orthorectified images in the tif format (open with QGIS for example), (Figure 14) and "SRTM_DEM_32636.vrt" file which presents information about the DEM projected in a UTM grid,
- "Trxs" folder contains the transformation matrice used for the orthorectification for the four raw images. This matrice contains Band-1 and Band-2 which correspond respectively to the 2D matrices of $\boldsymbol{X}(x_{pix}, y_{pix})$ and $\boldsymbol{Y}(x_{pix}, y_{pix})$ coordinates of the pixel $p(x_{pix}, y_{pix})$ in the raw image to be projected (Aati et al., 2022).

  The code adds the path of the four orthorectified images in "DataFile.csv", in the column "Orthos" and the path of the four transformation matrice in the column "Trxs".

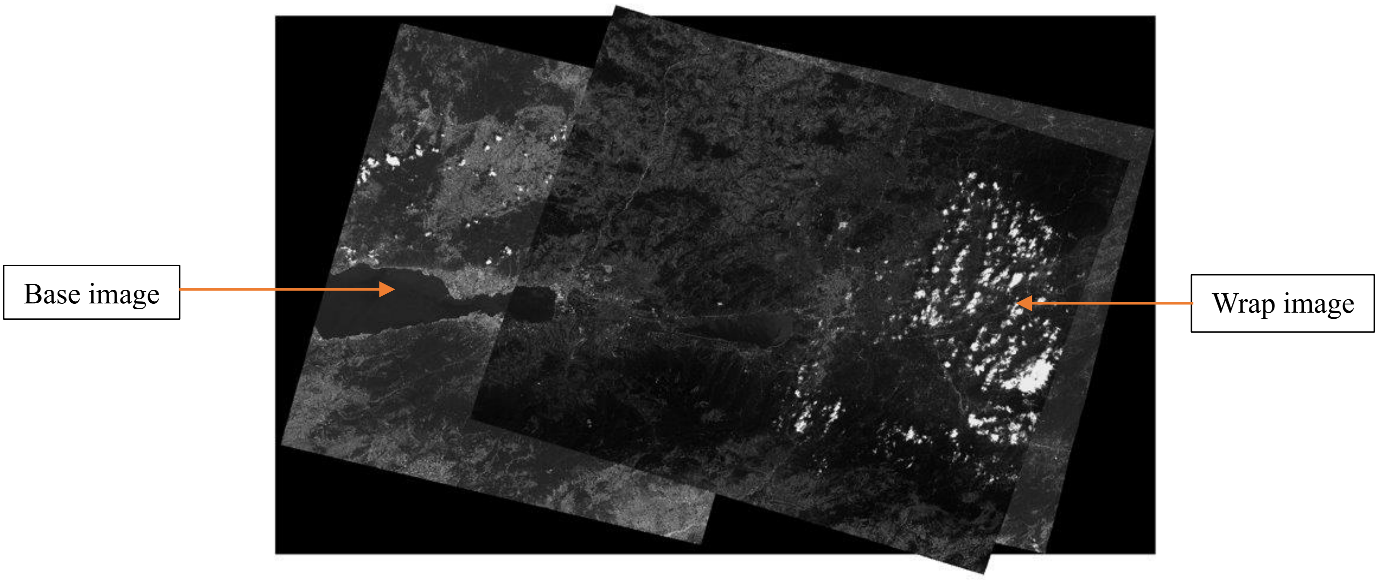

Figure 14: 1998-07-26-09-17-10-Spot-2-HRV-1-P-10_ORTHO_10.tif (Wrap Image) and rOrtho_1999-07-Spot4.tif (Base Image) opened with QGIS software

7. **prePostFile = izmit.compute_pre_post_pairs(data_file, pre_post_overlap_th=80):** computes the overlap between the images before (pre_i) and after (post_j) the event. The code creates a "PrePost_Pairs.csv" file, (Figure 15). Be careful, the code works if the overlap is over 80 (pre_post_overlap_th input parameter), but you can modify it.

	A	B	C	D
1	pre_i	post_j	Intersection	Overlap
2	/home/mcadoux/PycharmProjects	/home/mcadoux/PycharmProjects/GEO_COSI	POLYGON ((248060 4480360, 248060 4551300, 321580 4551	97.755387886277
3	/home/mcadoux/PycharmProjects	/home/mcadoux/PycharmProjects/GEO_COSI	POLYGON ((248060 4480360, 248060 4551300, 322660 4551	97.7335254814621
4	/home/mcadoux/PycharmProjects	/home/mcadoux/PycharmProjects/GEO_COSI	POLYGON ((246520 4480280, 246520 4551360, 321580 4551	100
5	/home/mcadoux/PycharmProjects	/home/mcadoux/PycharmProjects/GEO_COSI	POLYGON ((321780 4479330, 244020 4479330, 244020 4552	98.880976602238
6				

Figure 15: PrePost_Pairs_overlap.csv

8. **izmit.correlate() (optional):** creates the 2D correlation maps between images before and after the event. The code makes a "Correlation" folder (called "Correlation_optional" in this example not to be overwritten by the other folder). In "Correlation_optional", there are eight maps and each map is presented in the png (Figure 16) and tif (Figure 17) format. The tif format includes three bands: Band 1: East/West, Band 2: North/South (horizontal offset) and Band 3: SNR (Signal Noise Ratio). The png format includes two bands: Band 1: East/West, Band 2: North/South (horizontal offset). To know how to visualize the different bands on QGIS with the tif format and get information about the SNR, refer to the "correlation 2D" documentation.

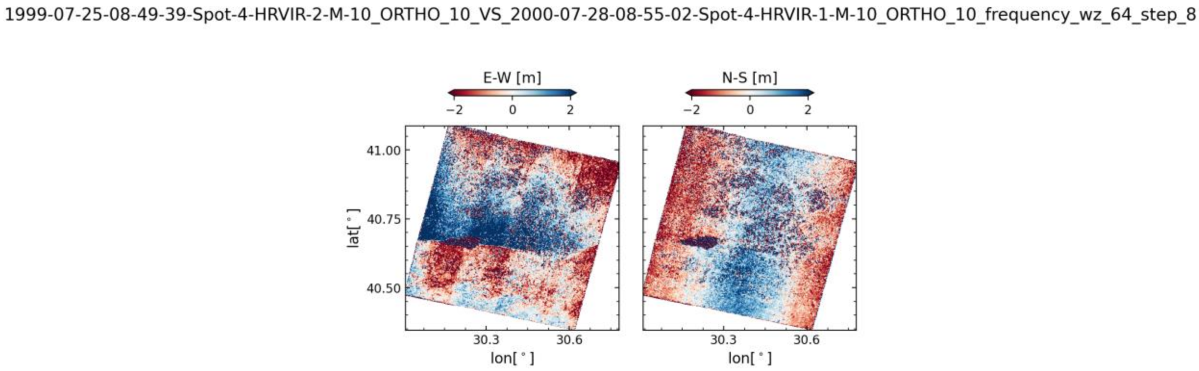

Figure 16 : 1999-07-25-08-49-39-Spot-4-HRVIR-2-M-10_ORTHO_10_VS_2000-07-28-08-55-02-Spot-4-HRVIR-1-M-10_ORTHO_10_frequency_wz_64_step_8.png with the Band 1: East/West (left), Band 2: North/South (right), in "Correlation_optional" file

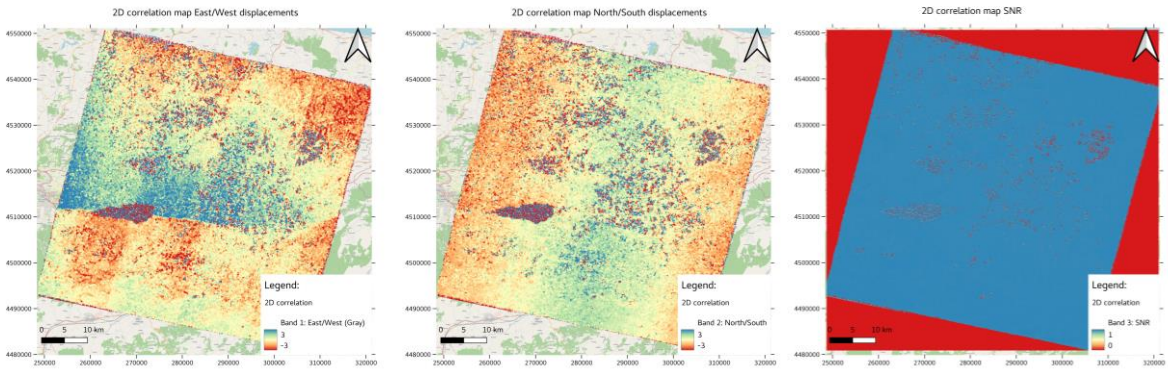

Figure 17: 1999-07-25-08-49-39-Spot-4-HRVIR-2-M-10 ORTHO 10 VS 2000-07-28-08-55-02-Spot-4-HRVIR-1-M-10_ORTHO_10_frequency_wz_64_step_8.tif with the Band 1: East/West (left), Band 2: North/South (center), Band 3: SNR (right),in " Correlation_optional" file

9. **izmit.generate_3DDA_sets(data_file):** computes the overlaps between the two images before the event, the two images after the event, and the four images. The code creates three documents:
   - "Pre_Pairs_overlaps.csv": overlap of the two images before the event (Figure 18),
   - "Post_Pairs_overlaps.csv": overlap of the two images after the event (Figure 19),
   - "Sets_3DDA.csv": overlap of the four images (Figure 20).

	A	B	C	D	E	F	G
1	img_i	img_j	fp_i	fp_j	Intersection	Overlap	crs
2	/home/mcadoux/	/home/mcadoux/Pyc	POLYGON ((3243	POLYGON ((321780 447	POLYGON ((248060 4480360,	96.5806367090266	EPSG:32636
3							

Figure 18 : Pre_Pairs_overlap.csv

	A	B	C	D	E	F	G
1	img_i	img_j	fp_i	fp_j	Intersection	Overlap	crs
2	/home/mcadoux/Pyc	/home/mcadoux/Pycharm	POLYGON ((321580 448	POLYGON ((3226	POLYGON ((246520 448	100	EPSG:32636
3							

Figure 19 : Post_Pairs_overlap.csv

	A	B	C	D	E	F
1	pre_i	pre_j	post_i	post_j	Intersection	Overlap
2	/home/mcadoux/Py	/home/mcadoux	/home/mcadoux	/home/mcadou	POLYGON ((248060 4551300, 321580 45	99.7287032013022
3						

Figure 20 : Sets_3DDA.csv

10. **izmit.correlate(corr_mode='set'):** creates the 2D correlation maps between images before and after the event but also between the two images before (Figure 22) the event and the two images after (Figure 21) the event in the "Correlation" folder. Twelve maps are available and each map is presented in the png and tif format. The tif format includes three bands: Band 1: East/West, Band 2: North/South (horizontal offset) and Band 3: SNR. The png format includes two bands: Band 1: East/West, Band 2: North/South (horizontal offset).

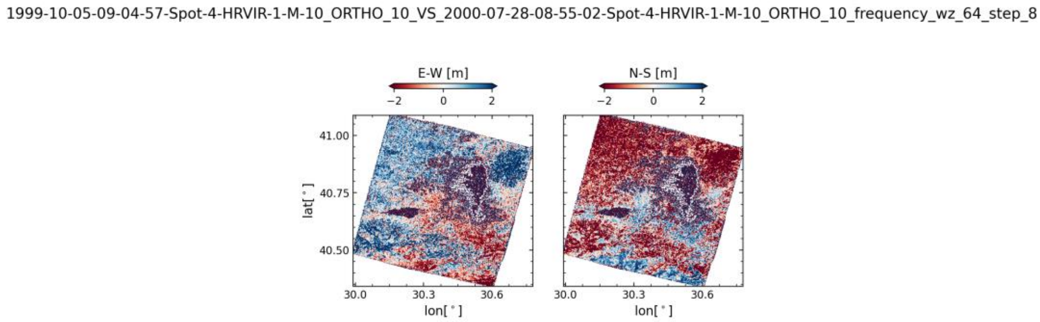

Figure 21: 1999-10-05-09-04-57-Spot-4-HRVIR-1-M-10_ORTHO_10_VS_2000-07-28-08-55-02-Spot-4-HRVIR-1-M-10_ORTHO_10_frequency_wz_64_step_8.png in "Correlation" folder, correlation between the two images after the event

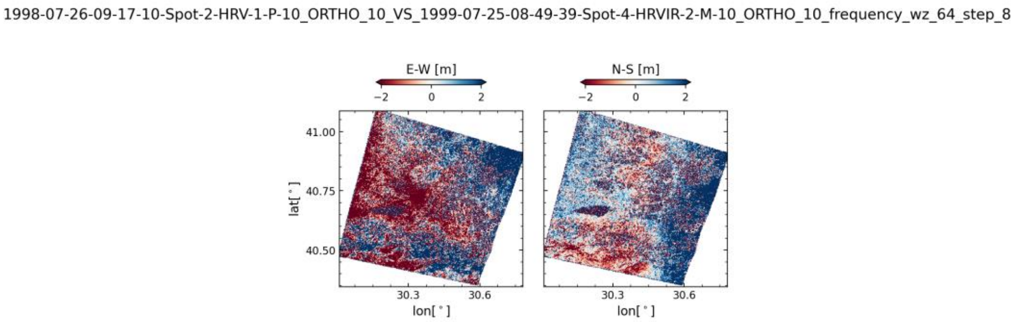

Figure 22: 1998-07-26-09-17-10-Spot-2-HRV-1-P-10_ORTHO_10_VS_1999-07-25-08-49-39-Spot-4-HRVIR-2-M-10_ORTHO_10_frequency_wz_64_step_8.png in "Correlation" folder, correlation between the two images before the event

11. **izmit.compute_ 3DD(data_file):** creates the 3D correlation maps. The code makes an "o3DDA" folder.
In this folder, the code creates another folder "3DDA_Set_1" where inside it will create the 3D correlation maps for the four raw images. If you would have selected more raw images in the input parameters (8, 12…), several folders would be created.
In "3DDA_Set_1", the code makes four folders "Set_1_Comb#", with the results inside. To compute the 3D displacement COSI-Corr uses an orthorectified image (comes from the raw images in the input parameters), called "Base Image", and it will compute the 3D displacement with the three other orthorectified images (comes from the raw images in the input parameters), called "Target Images". Because there are four raw images in the input parameters, four different

"Base Image" are possible. Each "Set_1_Comb#" uses a different "Base Image" to get all the possibilities of the 3D correlation map.
In "Set_1_Comb#", the code makes "Set_1_Comb#_3DDA" folder.
In "Set_1_Comb#_3DDA", different folders present the result:
- "3DDTiles" folder: contains all the patches in the tif format to create the 3D correlation map,
- "Corr" folder: contains three vrt format files called baseimage_VS_targetimage_ORTHO_#_correlatorname_wz_#_step_#.crop.vtr, with for each file, information about the correlation between the "Base Image" and the "Target Image",
- "rDEM" folder: contains the cropped DEM with the same footprint as the 3D correlation map, called "SRTM_DEM_32636.crop.tif" here,
- "RSM_files" folder: contains the four files, for the four raw images, with information about the images and the imaging system of each image after the optimization,
- "Set_1_Comb#_3DDA.tif" file: contains the 3D correlation map (Figure 23),

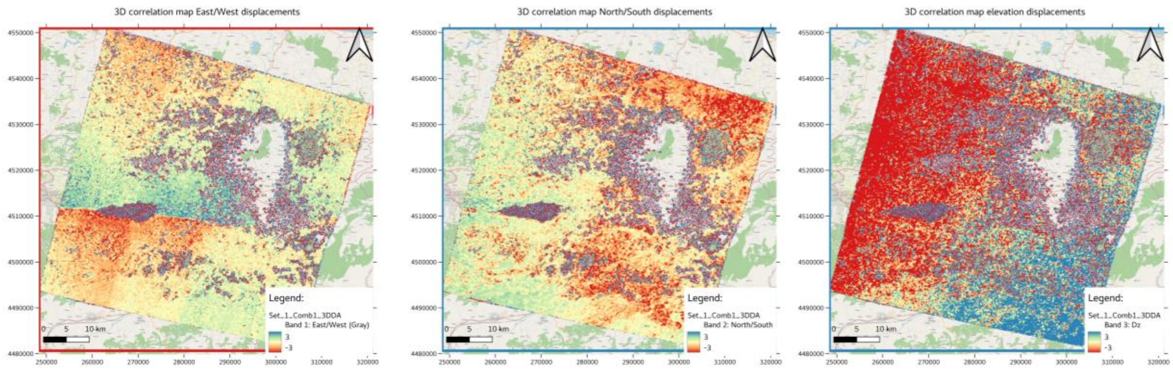

Figure 23 : Set_1_Comb1_3DDA.tif 3D correlation map with the Band 1 : East/West (left), Band 2: North/South (center), Band 3: Dz (right)

- "Tiles" folder: contains several folders with inside all the vrt format files for each patch of the four cropped raw images, the cropped DEM and the three cropped correlations between the "Base Image" and the "Target images",
- "Trx" folder: contain four vrt format files, for each raw image, with information about the transformation matrice used for the orthorectification and cropped to have the same footprint as the 3D correlation map.

Moreover, each time we run the code, all the workflow, several or one functions, a GeoCosiCorr3DPipeline_mth_d_yyyy_h_min_s.log is created. This document stocks all the information about the coding process.

5. References

Aati, S., Milliner, C., Avouac, J.-P., 2022. A new approach for 2-D and 3-D precise measurements of ground deformation from optimized registration and correlation of optical images and ICA-ased filtering of image geometry artifacts. California Institute of Technology, Pasadena, CA.