SNAP Processing of Sentinel-1 Phase Coherence for Living England

Dr. Chris Moore, Lead Advisor Earth Observation, Natural England
14 December 2021

This document sets out the step-by-step processing undertaken to generate average phase coherence images using the European Space Agency (ESA) developed open-source SentiNel Applications Platform (SNAP) and Python. The Python script "snappy_bgz.py" has been developed to automate this processing.

1. Sentinel-1 Data

Sentinel-1 interferometric wide-swath (IW) scenes acquired by the ESA Copernicus programme are pre-processed to Level-1 Single Look Complex (SLC) format with vertical-vertical (VV) and vertical-horizontal (VH) polarisations. Open data (Copernicus Sentinel data 2020, processed by ESA) are sourced from the ESA Scihub or the Alaska Satellite Facility (ASF supports command line batch downloads with .csv table and .py Python script). Downloaded data should be quality checked to account for errors such as missing and mis-processed bursts and large perpendicular baselines (the orbit "offset" between 2 satellite passes).

Selected scenes should be from tracks that cover the whole area of interest (e.g., Living England Biogeographic Zone (BGZ)). To account for any temporal variation in coherence, 5 consecutive dates with temporal baselines of 6-days should be used in order to construct a network of 10 coherence images (see Figure 1). If the scenes cover a winter period, there should not be significant snow cover on any of the acquisition dates. Note that if processing 10 interferograms (as shown in Figure 1) with 2 IWs per scene, the total data size of the files created will be approximately 500 GB.

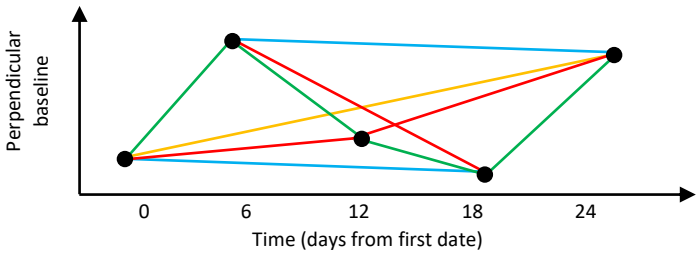

Figure 1. Example plot of an interferogram network consisting of 5 acquisition dates (black circles), and 10 interferograms (coloured lines) with 6 (green), 12 (red), 18 (blue), and 24-day (orange) temporal baselines.

2. Automated Processing

The Graph Builder tool in SNAP (see Figure 2) may be used to partially automate processing, although this will require a lot of memory as SNAP does not clear its cache between steps, and can be slower than running the steps individually.

The Python script snappy_bgz.py uses the SNAP interface with python (Snappy) to run steps in batch without crashing due to memory limitations. This processing chain is fully automated, and can produce average ascending or descending coherence (VV and VH) maps for a specified BGZ in GeoTiff format.

Information on how to use snappy_bgz.py, how to prepare for automated processing, and what each processing step does and outputs can be found below in Sections 3-5.

Note that up to 32 GB of RAM may be required for stacking (step 12) if processing 2 IW sub-swaths covering a BGZ. In order to run snappy_bgz.py, SNAP must be installed with snappy – this is best done using a python virtual environment (https://towardsdatascience.com/getting-started-with-snap-toolbox-in-python-89e33594fa04). The snappy_bgz.py script may then be edited to define the correct processing directory (where the .zip files are stored, see File Structure below), the season (e.g., sumYYYY/winYYYY-YY), the orbit direction (A/D), and the start and end steps of the processing.

3. Pre-processing Preparation

All raw data .zip files should be retained in a single archive folder as shown in Section 5. This helps reduce the disk space used as well as increases the efficiency of downloading data in batch. In this folder there must be a text file for each BGZ, season, and each look direction of the format "bgz01_sum2021_A_zips.txt". This should list on new lines the 4-digit ID of each S1 scene to be processed (e.g., "694D" for "S1A_IW_SLC__1SDV_20210601T180536_20210601T180603_038150_0480AE_694D.zip").

BGZ boundaries should be buffered by approx. 10 km (to allow account for inaccuracies in burst selection (step 2)) and be stored as shapefiles with just the zone ID number (e.g., "1.shp"). BGZ shapefiles must also be converted into well-known-text (WKT) polygon format and stored within text files of the format "BGZ01_wkt.txt". From ArcPro, you can extract the polygon vertices by adding x/y geometry, then exporting the attribute table to .csv format. The x/y values can then be copied into a text file and formatted into a WKT polygon: "POLYGON((X1 Y1, X2 Y2, X3 Y3, X1 Y1))" or "MULTIPOLYGON(((X1 Y1, X2 Y2, X3 Y3, X1 Y1)), ((X4 Y4, X5 Y5, X6 Y6, X4 Y4)))".

4. Using the Processing Chain

Within snappy_bgz.py (lines 9-19), you can define the BGZ to process ("01"-"14"), which season the data for (e.g., "sum2021"), the orbit direction ("A"/"D"), the start and end processing steps (0-14), and whether or not to clean the processing directory once finished ("y"/"n"). You should define the master processing and WKT/ESRI shapefile directories in lines 22-26.

Each of the processing steps shown below runs from snappy_bgz.py which calls the relevant sub-processing script (e.g., snappy_bgz_01_asm.py for Step 1). Each step can also be run manually in SNAP following the descriptions provided. During processing all inputs should be left at the default values unless stated.

As the output to each step will be written, step END (snappy_bgz_END_clean.py) is used to delete the temporary files once processing is complete. It is good practice to check the outputs of the processing chain in SNAP once each step is finished to ensure that there are no errors in the data. This is particularly important for step 2 (SLC) to ensure that the correct bursts are being processed, and step 12 (Stack) to ensure that the coherence images have stacked correctly before averaging.

If there is not enough RAM to complete step 12 (Stack), then you can use the additional step 11 ("snappy_bgz_11-2_sub.py") to subset the BGZ into 2 sections and continue the processing chain. Note that using this option requires additional pre-processing preparation as detailed within "snappy_bgz.py" under Step 11.

Once the processing chain is finished, you may mosaic average coherence maps together using the SNAP tool "SAR-Mosaic". Do not use the "normalize" option as this may distort the coherence values; leave the other variables at the default settings. This is useful to combine different tracks to completely cover a BGZ or to average the ascending and descending coherence maps for a BGZ (the script "snappy_bgz_mosaic.py" can be used to run this from the command line).

Processing steps:

0. **(Zip).** Copy raw data .zip files from the local data archive to the processing directory.
1. **(Asm) Slice Assembly.** Merge along-track scenes before processing if required. If not required, open the .zip file and save in .dim format.
2. **(SLC) TOPS Split.** Use the split TOPS function to process only the sub-swaths and bursts that cover the area of interest (WKT format polygon). For each sub-swath (IW*), the selected bursts must be spatially consistent between acquisition dates.
3. **(Orb) Apply Orbit File.** Apply orbital corrections using precise orbits. Tick "Do not fail…" box if precise orbits cannot be found for the scene.
4. **(RSLC) Back Geocoding.** For every image pair, co-register the secondary image to the primary image with the S1 back geocoding function - use the SRTM 1Sec HGT digital elevation model with default interpolation.
5. **(Esd) Enhanced Spectral Diversity.** Improve the accuracy of the co-registration.
6. **(cc) Coherence Estimation.** For each image pair stack, calculate the phase coherence between the 2 images – subtract the flat-earth phase component, and use an independent window size of 10 looks in range and 2 in azimuth.
7. **(Deb) TOPS De-burst.** Merge along-track bursts.
8. **(Mrg) TOPS Merge.** Merge IW sub-swaths to generate a single image.
9. **(ML) Multi-looking.** Average values in the range direction to create square pixels and reduce noise. Tick "GR Square Pixel" box, and set number of azimuth looks to 1.
10. **(TC) Range-Doppler Terrain Correction.** Apply a terrain correction and convert from range-azimuth to geographical coordinates - use the SRTM 1Sec HGT digital elevation model with default interpolation, and set the output coordinate system to British National Grid.
11. **(Msk) Land-Sea-Mask.** Set pixels outside of the BGZ boundary to 0. First import the relevant ESRI shapefile to the target layer, then use select the vector polygon as the mask.
11. **(Msk) Subset.** Clip the image to the BGZ boundary (WKT format polygon).
12. **(Stack) Create Stack.** Collate all coherence images together using the create stack function. Set coherence image with the earliest acquisition date to be the new primary image - use nearest neighbour resampling.
13. **(BM) Band Maths.** Calculate the mean VV and VH average of the 6-day, 12-day, 18-day, and 24-day coherence images in the stack.
14. **(Exp) Band Maths.** Calculate a single output for VV and VH by taking the mean of the values from the 6-day, 12-day, 18-day, and 24-day average images.
14. **(Exp) Band Extractor.** Export the resulting coherence image(s) in "GeoTIFF / BigTIFF" format (in snappy_bgz_14_exp.py this is combined with the previous Band Maths step).

END. **(Clean).** Delete all processing outputs apart from the final GeoTiff files to clear disk space, before running for the other look direction (ascending/descending) or another BGZ.

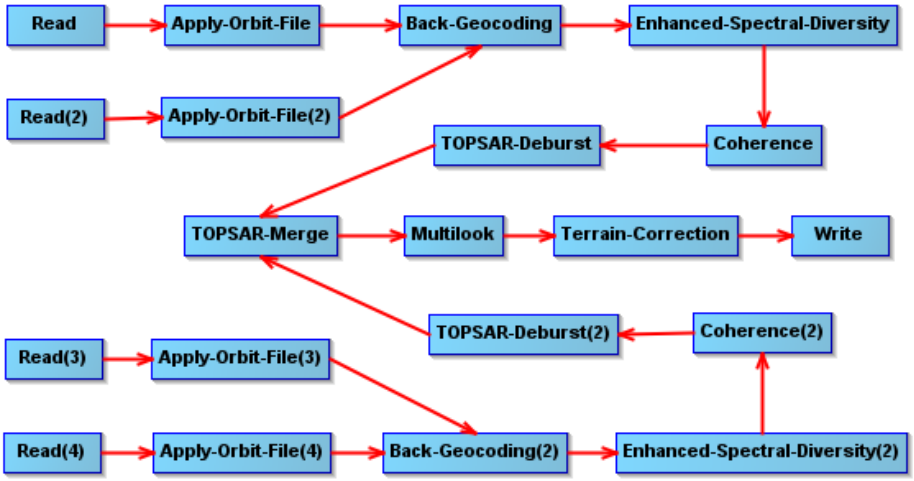

Figure 2. Batch processing XML graph in SNAP showing steps 3-10 for 2 acquisition dates with 2 IW sub-swaths each. Input "Read" files of the format: 20200526_IW2_SLC, 20200601_IW2_SLC, 20200526_IW3_SLC, 20200601_IW3_SLC. Output "Write" file of the format: 20200526_20200601_cc_ML_TC.

5. File Structure

The required folder structure within the master directory is shown below with file outputs and the processing step indicated for each output file. All dates are in the form YYYYMMDD. Outputs of steps 1-13 are in SNAP standard .dim and .data format.

ASF_download		S1[A/B]_IW_SLC__1SDV_*.zip	
		bgz[##]_sum2021_A_zips.txt	
		bgz[##]_sum2021_D_zips.txt	
bgz[##]	**sum2021_A**	S1[A/B]_IW_SLC__1SDV_*.zip	0
		[date]_SLC	1
		[date]_IW[#]_SLC	2
		[date]_IW[#]_SLC_Orb	3
		[date1]_[date2]_IW[#]_RSLC	4
		[date1]_[date2]_IW[#]_RSLC_esd	5
		[date1]_[date2]_IW[#]_cc	6
		[date1]_[date2]_IW[#]_cc_deb	7
		[date1]_[date2]_cc	8
		[date1]_[date2]_cc_ML	9
		[date1]_[date2]_cc_ML_TC	10
		[date1]_[date2]_cc_ML_TC_msk	11
		bgz[##]_sum2021_A_stack	12
		bgz[##]_sum2021_A_stack_VH_BM	13
		bgz[##]_sum2021_A_stack_VV_BM	13
		bgz[##]_sum2021_A_VH_cc.tif	14
		bgz[##]_sum2021_A_VV_cc.tif	14
	sum2021_D	…	…
	bgz[##]_sum2021_VH_cc.tif		
	bgz[##]_sum2021_VV_cc.tif		

6. Developer Support

By using Snappy, the full range of geo-processing operators and parameters are available from the SNAP Sentinel toolboxes. Within SNAP, help pages are given for each operator, providing technical processing details and further information on the required inputs. While there is no formal documentation for Snappy, many online documents show examples of processing workflows, and ESA support the Science Toolbox Exploitation Platform (STEP) Forum where ESA and community experts provide help and examples of SNAP and Snappy (categorised under Development/Python) workflows.

Important Note: The parameter names shown within SNAP are the aliases of the parameter names required by Snappy operators. Not all Snappy operators support the use of aliases and many require the abbreviated parameter names. If an incorrect name or unsupported alias is used as the parameter, then the default value is used. The python script “snappy_op_params.py” may be used to identify the required abbreviated parameter names, alias parameter names (“None” if unsupported), and default values for any SNAP operator.