AlignSAR Tutorial

Version 1.0 (2024-01-02)

University of Twente, University of Leeds, RHEA System S.p.A.,
AGH University of Krakow

Contents

List of Figures

Chapter 1

Introduction

This document provides a tutorial of using AlignsAR package to create SAR benchmark datasets, and describes the concept of some relevant methods. This docuemnt is complied by Ling Chang.

1.1 Main contributors

This package is accomplished by all AlignSAR members. The main contributors for script development are Ling Chang, Anurag Kulshrestha, Xu Zhang, José Manuel Delgado Blasco, Angie Catalina Carrillo Chappe, Milan Lazecky, and Serkan Girgin.

1.2 Installation

All scripts for the AlignSAR package can be found on the GitHub web page:

https://github.com/AlignSAR/alignSAR.

The user can directly download them to his/her local computer.

SAR data preprocessing requires using either ESA SNAP or Doris-5. The user can download SNAP via ESA SNAP, and Doris-5 via GitHub AlignSAR and manually install them on his/her computer. Note that if the user uses Doris-5 via GitHub TUD-Doris, the trouble shooting when installing it refers to Appendix A.

1.2.1 Docker creation

By creating a docker that contains the pre-installed necessary software tools, i.e. SNAP 9.0, Doris and python 3 environment and other customised scripts, the end users could directly follow the procedure to create SAR benchmark datasets without installing the required software tools.

The instruction of Docker software installation on e.g. Ubuntu refers to DigitalOcean, see the attachment in Appendix B.

To build the AlignSAR docker image, user should install docker and download Dockerfile provided in the github repository (https://github.com/AlignSAR/alignSAR) to a dedicated directory. Inside the directory, user can build the image using:

```
docker build -t alignsar .
```

and test the image by running an interactive terminal session using:

```
docker run -it alignsar
```

Note that one needs to run these commands in terminal, and in the same folder as the 'docker' file stored. Here 'alignsar' is the name of the container going to be mounted, and can be customized by the enduser. To unmount (alignsar) container, one can directly close the terminal.

1.2.2 SAR data processing with Doris in Docker

When the Docker installation is done, follow the instructions below to process raw SAR data using Doris:

1. Download the DockerFile. It can be found via: 'https://github.com/AlignSAR/alignSAR /blob/7c5966de6256ec1cd266b4794b225efae7be1a8a/DockerFile'. Visit this page and download it to the local path, e.g., /home/docker;

2. Compile the 'DockFile'. In the terminal under path '/home/docker', type the command 'sudo docker build -t your_docker_image_name .'. Variable 'your_docker_image_name' can be defined as preference;

3. Run the docker image. Run the command 'sudo docker run -it -v /your_local_path:/path

_you_want_set_in_docker your_docker_image_name' can open a docker interface. Use the keyword '-it' will run the docker image using interface mode in the terminal, while '-v' will mount the local disk to docker;

4. Set a path in Doris-5 before using it. The command in the docker interface 'vim /root/DorisITCupdate/doris/doris_stack/main_code/doris_main.py' will set the Doris package path. Line 5 'sys.path.append('/home/username/software/')' should be changed to 'sys.path.append('/root/Doris5ITCupdate/')'. Also, make sure when using the command 'vim /root/DorisITCupdate/doris/install/doris_config.xml' to open the XML file, Line 2 should be '<source_path>/root/Doris5ITCupdate/doris</source_path>'.

Then Doris is prepared to process data.

Chapter 2

Definition of SAR benchmark datasets and data sources

2.1 SAR signatures

Here we recap the identified representative SAR signatures which are classified into three categories [1].

1. Category 1:

   Single polarimetric signatures, like amplitude, intensity, backscatter coefficient, (interferometric) phase and coherence

2. Category 2:

   Multi polarimetric signatures, such as co-, dual polarization cross product, and summation, difference and ratio of the co-, dual, quad polarization intensities, and entropy, scattering mechanisms, SAR vegetation index (RVI) , SAR soil moisture

3. Category 3:

   Inherited attributes from additional geospatial observations, e.g. land use land cover type, cadastral features, temperature, atmospheric phase, and geological information.

Based on the quality level, we extract the pixel with high quality, namely SAR benchmark dataset (SARbd) [4]. We provide SARbd in radar coordinates (as intermediate products), and SARbd in geo coordinates. We consider cateograize SARbd into three different levels in terms of the quality level of SARbd. Basically, Level 1, namely **SARbd-L1**, is merely based on SAR statistics, Level 2, namely **SARbd-L2**, is based on external geospatial reference data, and Level 3, namely **SARbd-L3**, is based on both SAR statistics and external geospatial reference data.

2.2 Useful data sources

Many SAR data and other geospatial (reference) data can be collected freely, for instance,

- Sentinel-1a&b SAR data can be downloaded from ASFvertex
- ERS and Envisat can be downloaed via eogateway
- GPS dataset is stored in surfdrive. More GPS observations can be found on NGLStation
- AHN (actual height of Netherlands) can be collected via pdok
- Topographic map (TOPNL) can be found on brt
- SRTM can be downloaded via earthexplorer
- Copernicus DEM can be downloaded via panda
- DTM (Digital Terrain Model of Poland) can be downloaded via DTMPoland
- Land Cover Map - HRL - High Resolution Layers can be found via POLSA
- Geological data – Boreholes and Mining areas are stored on dmpgi and more detailed information for boreholes can be found via pgi

Chapter 3

Procedure description

3.1 Use case: Case in the Netherlands

We have three use cases, in the Netherlands, Poland and India. Here we take the first use case in the Netherlands as a demo. The site is in the northern part of the country with a Groningen gas field and the Wadden Sea, see Fig. 3.1. This region is active due to both human activities like oil/gas extraction and geo-processes like coastal erosion. The Groningen region due to human activity is affected by land subsidence and earthquakes.

Seven C-band Sentinel-1A acquisitions (Path 15, Frame 169) in ascending orbit were collected. They have VV and VH polarization channels, and were acquired separately on 09 Jan, 21 Jan, 02 Feb, 14 Feb, 26 Feb, 10 Mar and 22 Mar 2022. The python script to automatically download these acquisitions is attached in DownloadCode, see data list in Fig. 3.2. The area of interest is outlined by purple in Fig. 3.1, and its shape file can be obtained via aoi. The orbital data can be collected from POEORB.

For instance, using a denoising method, multi-temporal filtering (Eq. (C.1)) on these 7 Sentinel-1 amplitude images in VV, the speckle noise can be reduced, see Fig. 3.3. The zoomed-in images (line $200-400$, pixel $2000-2200$) for the original and filtered SAR images are indicated in red and green separately. The spatial window size was 3×3 for this example.

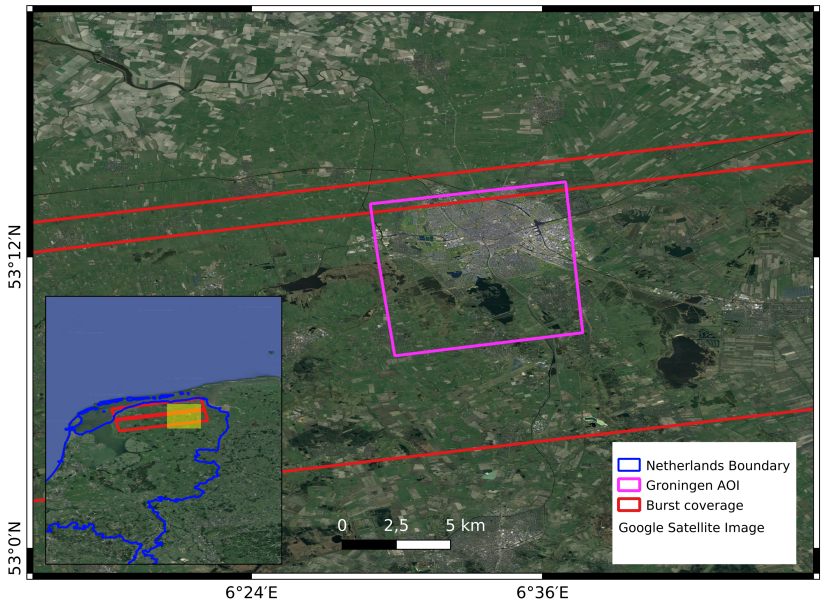


Figure 3.1: Study area for Use case 1: Groningen, The Netherlands. The city is shown within the pink polygon and corresponds to the amplitude image in Fig. 3.3. The yellow area in the inset map shows the overview of the outer map.

S1A_IW_SLC__1SDV_20220109T171712_20220109T171740_041387_04EBB7_25EB.zip	4,6 GB
S1A_IW_SLC__1SDV_20220121T171712_20220121T171739_041562_04F172_DA9D.zip	4,6 GB
S1A_IW_SLC__1SDV_20220202T171711_20220202T171738_041737_04F770_A0F7.zip	4,5 GB
S1A_IW_SLC__1SDV_20220214T171711_20220214T171738_041912_04FD8C_24E7.zip	4,5 GB
S1A_IW_SLC__1SDV_20220226T171711_20220226T171738_042087_050391_55E5.zip	4,5 GB
S1A_IW_SLC__1SDV_20220310T171711_20220310T171738_042262_05097D_3DA1.zip	4,5 GB
S1A_IW_SLC__1SDV_20220322T171711_20220322T171738_042437_050F75_7D83.zip	4,4 GB

Figure 3.2: List of Sentinel-1 SLC data used

3.1.1 Creation of SAR signatures: Category-1 and -2

To generate Category-1 and -2 SAR signatures, Doris-5 is used. Here we show some basic procedures and parameter configuration. After Doris-5 installation (e.g. in the folder of /software/doris/), downloading the seven Sentinel-1 SAR images via DownloadCode, and the corresponding orbital data via Sentinel-1OrbitatASF, save Sentinel-1 SAR data and

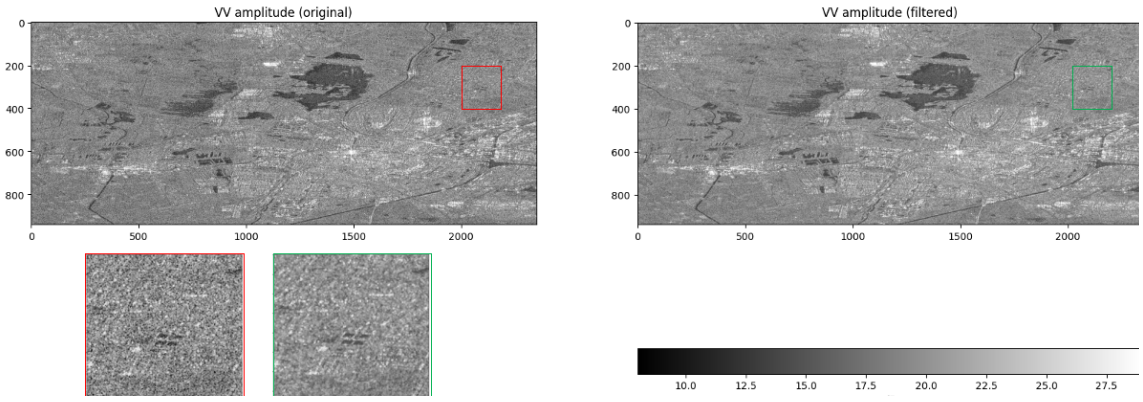


Figure 3.3: Multi-temporal filter example

its orbital data in separate folders, then first go to /software/doris/prepare_stack, and run the python scripts: python2 prepare_datastack_main.py. This script will ask one to define the different folders that were created before. Here is an example:

Enter the path to the archive data folder: /home/ling/d2/groningendata

Which polarisation do you want to use (vv,hh,vh,hv): vv

Which SAR data track/path do you want to work with? (explore on https://scihub.copernicus.eu/dhus/) : 15

Is this track ascending or descending? (asc/dsc) : asc

Enter the path to the folder of new datastack: /home/ling/d2/groningen/process

Enter full path to the shapefile: /home/ling/d2/groningen/aoi/gro_aoi.shp

Enter the path to the folder of the orbit files: /home/ling/d2/groningen_orbit

Do you want to generate the DEM file automatically (Yes/No): Yes

Enter path to the dem folder: /home/ling/d2/groningen/DEM

Do you want to use parallel computing (Yes/No): Yes

How many cores do you want to use: 2

What is the start date of your stack in yyyy-mm-dd (can be changed later): 2022-01-09

What is the end date of your stack in yyyy-mm-dd (can be changed later): 2022-03-22

What is the master date of your stack in yyyy-mm-dd (can be changed later): 2022-02-14

After providing answers to these questions, DEM will be automatically downloaded,

and the terminal will show the processing status, like

https://e4ftl01.cr.usgs.gov/MEASURES/SRTMGL1.003/2000.02.11/SRTMGL1_page_1.html status200 received ok

https://e4ftl01.cr.usgs.gov/MEASURES/SRTMGL1.003/2000.02.11/SRTMGL1_page_2.html status200 received ok

https://e4ftl01.cr.usgs.gov/MEASURES/SRTMGL1.003/2000.02.11/SRTMGL1_page_3.html status200 received ok

https://e4ftl01.cr.usgs.gov/MEASURES/SRTMGL1.003/2000.02.11/SRTMGL1_page_4.html status200 received ok

https://e4ftl01.cr.usgs.gov/MEASURES/SRTMGL1.003/2000.02.11/SRTMGL1_page_5.html status200 received ok

https://e4ftl01.cr.usgs.gov/MEASURES/SRTMGL1.003/2000.02.11/SRTMGL1_page_6.html status200 received ok

https://e4ftl01.cr.usgs.gov/MEASURES/SRTMGL3.003/2000.02.11/ status200 received ok

https://e4ftl01.cr.usgs.gov/MEASURES/SRTMGL30.002/2000.02.11/ status200 received ok

([], [], [])

Bounding box is:

from 51.6 latitude to 54.800000000000004

from 4.9 longitude to 8.200000000000001

total file size is 3841 in latitude and 3961 in longitude

total file size is 3841 in latitude and 3961 in longitude

Save data to geotiff

Calculate geoid correction for SRTM data

Correct DEM for geoid

–2023-10-01 15:06:06– https://github.com/anurag-kulshrestha/geoinformatics/raw/master/WW15MGH.DAC

Resolving github.com (github.com)... 140.82.121.3

Connecting to github.com (github.com)—140.82.121.3—:443... connected.

HTTP request sent, awaiting response... 302 Found

Location: https://raw.githubusercontent.com/anurag-kulshrestha/geoinformatics/master/WW15MGH.DAC [following]

–2023-10-01 15:06:07– https://raw.githubusercontent.com/anurag-kulshrestha/geoinformatics/master/WW15MGH.DAC

Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 2606:50c0:8001::154, 2606:50c0:8003::154, 2606:50c0:8002::154, ... Connecting to raw.githubusercontent.com (raw.githubusercontent.com)—2606:50c0:8001::154—:443... connected. HTTP request sent, awaiting response... 200 OK Length: 2076480 (2,0M) [application/octet-stream] Saving to: '/home/ling/d2/groningen/DEM/EGM96_15min.dat'

/home/ling/d2/groning 100

2023-10-01 15:06:07 (58,9 MB/s) - '/home/ling/d2/groningen/DEM/EGM96_15min.dat' saved [2076480/2076480]

0 .. 10 .. 20 .. 30 .. 40 .. 50 .. 60 .. 70 .. 80 .. 90 .. 100 - Done Input file size is 3961, 3841 0...10...20...30...40...50...60...70...80...90...100 - done.

The stack folder will be created in /home/ling/d2/groningen/process, containing create_dem.sh, dem doris_input.xml, doris_stack.sh, download_sentinel.sh, input_files, and stack. The 'dem' folder includes dem.raw, dem.raw.doris_inputfile, dem.raw.q, dem.raw.var, and dem.tiff (SRTM1 for this case). The 'stack' folder is empty for the time being. One can still modify 'doris_input.xml' to decide to run/cancel any processing steps by defining 'Yes' or 'No'. For instance, if one doesn't want to run coherence processing, in the 'doris_input.xml' file, one can define <do_coherence>No</do_coherence>. After the 'doris_input.xml' modification, run the command bash doris_stack.sh

To showcase the extracted SAR signatures, a demo 'signature_extraction_demo.ipynb' on juypter notebook is created, as well as an illustration shown in Fig. 3.4. The python script, namely 'Jupyter_input_preapre.py' can be used to convert e.g. .raw data to .npy

file that can be directly and interactively processed on jupyter notebook.

3.1.2 Creation of SAR signatures: Category-3

To create Category-3 SAR signature, Radarcoding shall be implemented for the additional geospatial data.

3.1.3 Radarcoding concept

Radarcoding is a step to convert all available data to radar coordinates, and link radar scatterers in SAR images with the counterparts in the reference datasets registered in geographic coordinate systems. Radarcoding is used to associate Category-3 signatures to radar scatterers. The Radarcoding script, namely rdrCode_main.py is python based, [2].

The Radarcoding procedure is illustrated in Fig. 3.5. A stack of SAR images (≥ 2) is used and coregistered, resampled and aligned to the common master grids using differential SAR Interferometry (DInSAR). The mid-line number, PRF (pulse repetition frequency), RSR (range sampling frequency), first-line timing and first-pixel timing information are taken from the SAR image metadata. The SAR coverage boundary is extracted using the extreme extent coordinates from the satellite image metadata.

The additional reference observations may contain point-line-polygon, like topographic maps including land use land cover clarification product. For such observations, we rasterize them class-wisely using the GDAL rasterize function. Here, the choice of the raster image resolution is dependent upon the size of reference polygons, the spatial resolution of the SAR data and the processing limitations of the computer. In case the reference datasets are available as rasters, e.g. optical images, Generic Atmospheric Correction Online Service for InSAR (GACOS) atmospheric phase delay maps [7], the datasets can be used directly for radarcoding. They can be optionally resampled to a required spatial resolution, especially for datasets with ultra-fine resolutions.

Rasterized reference data is then cropped based on SAR coverage boundary information, and later assigned radar coordinates (with range and azimuth indices), namely

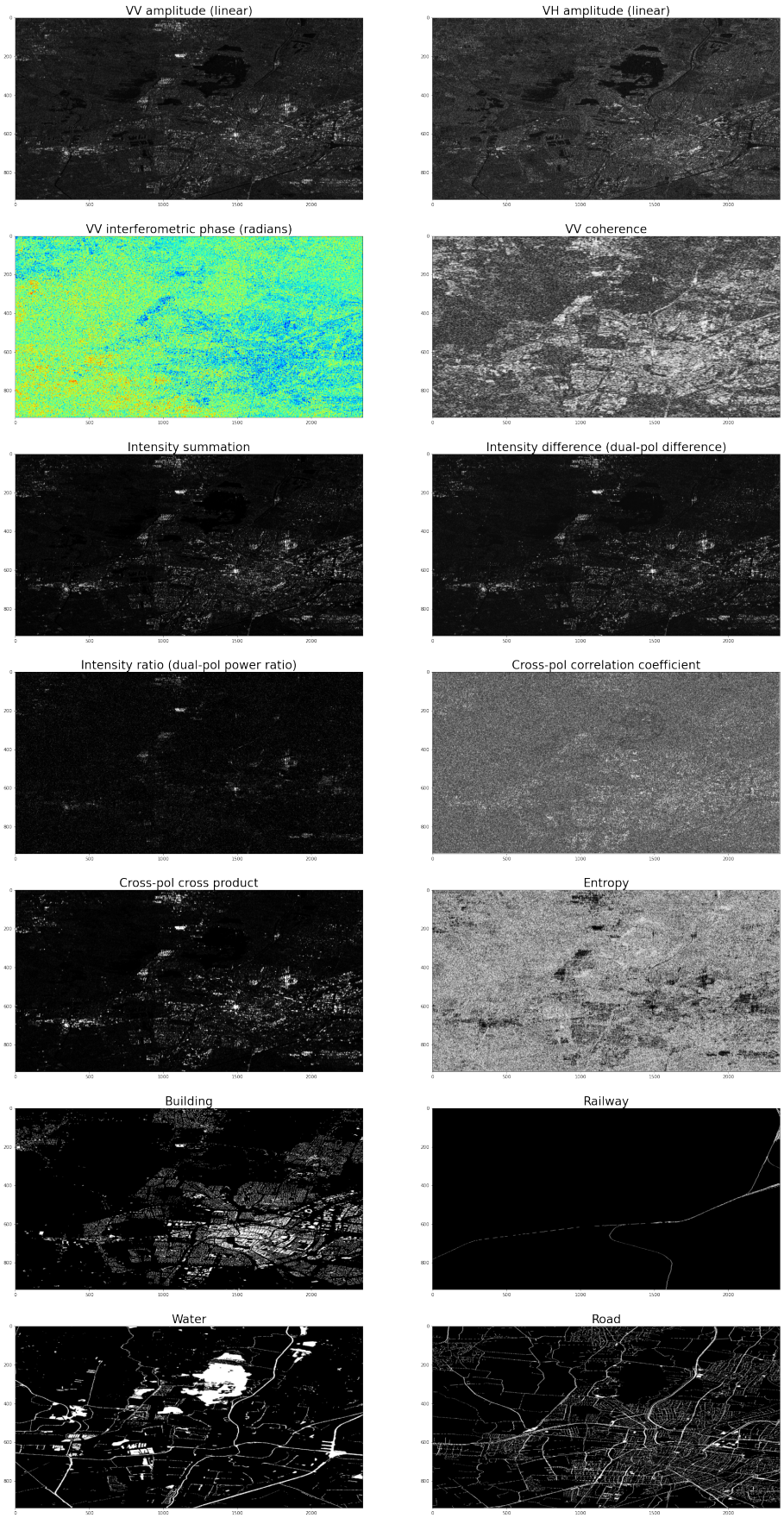


Figure 3.4: 14 SAR signatures for the acquistion on 09 Jan 2022

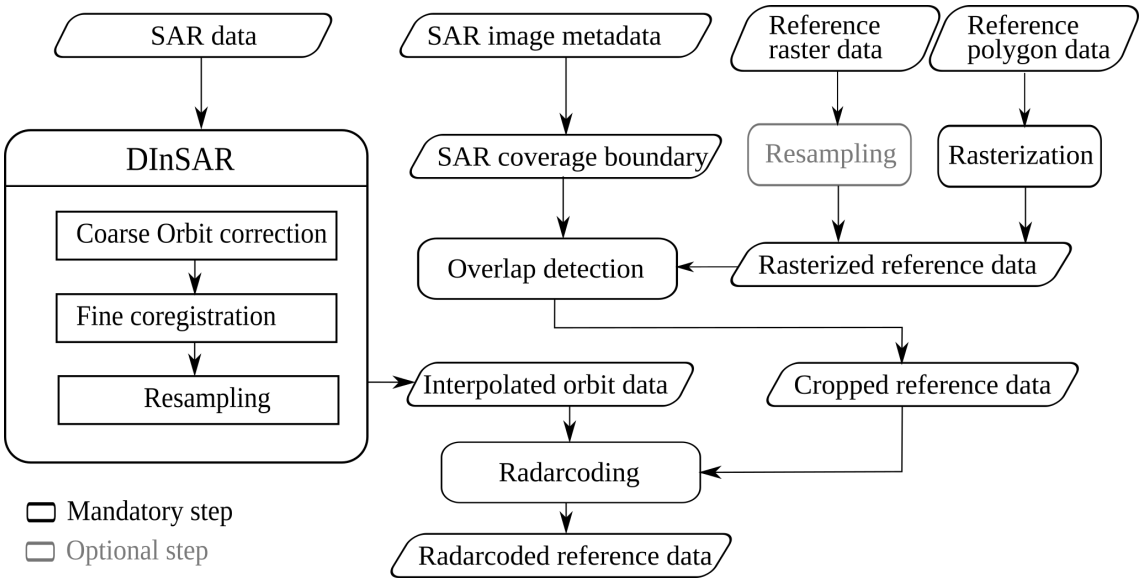


Figure 3.5: Radarcoding procedure [2]

radarcoded reference data.

To run Rdr-Code, the user needs to define the name of the file that needs to radarcoded. If it is multilayered file, i.e. with multiple bands in raster data, or multiple layers in vector, then the name of layers needs to be mentioned.

The alternative option for radarcoding is a function called snap_rdrcode.py, allowing radarcoding from data prepared by SNAP and developed based on GMTSAR radarcoding routines.

In the end, all SAR signatures can be directly converted to a geographic (e.g. WGS-84) coordinate system using Doris-5 (when setting '<do_calc_coordinates>Yes</do_calc_coordinates>' in 'doris_input.xml') or geocoding tables generated by SNAP, see Fig. 3.6. For the later option, we generate geocoding look-up tables based on the reference acquisition that will be later used to transform signatures generated in radar coordinates from this and other precisely coregistered acquisitions in the data stack.

The geocoding script `geocode_ifg_snap.sh` calls a SNAP processing graph file `graph_geocoding_s1_geotiff.xml` that contains two operations: Multilooking and Terrain Correction (using Copernicus DEM automatically downloaded by SNAP) leading to geocoding the data to a 0.00027 deg (30 m) resolution GeoTIFF grid in WGS-84 system

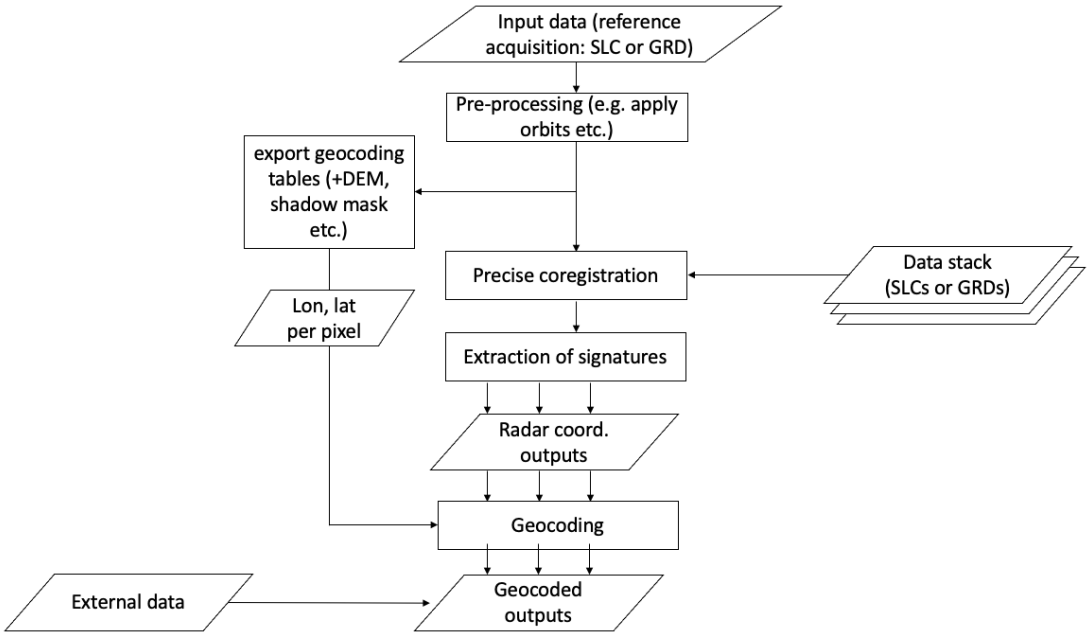


Figure 3.6: Procedure of extracting geocoded SAR signature

that can be further converted to another coordinate system (e.g. UTM).

Usage:

`geocode_ifg_snap.sh XX.dim /output/path [bandname]`

(if bandname is not provided, the script would geocode all bands found inside the SNAP dim file structure)

3.2 Geospatial data TOP10NL collection and coversion

We take the geographic base map TOP10NL data as an example and it can be obtained through https://service.pdok.nl /brt/topnl/atom/top10nl.xml, see Fig. 3.7. For TOP10NL Geopackage block, three versions of TOP10NL data are included which were updated in different years. Here, we select the 2022 version '$top10nl\_Compleet-2022.gpkg$', which aligns with the SAR dataset.

The TOP10NL data in 'gpkg' format can be viewed by using QGIS. When importing the TOP10NL data into the QGIS, a pop-up window shows all the attribute layers as shown in Fig. 3.8. Here, 'top10nl_gebouw_vlak' (buildings), 'top10nl_spoorbaandeel_lijn'

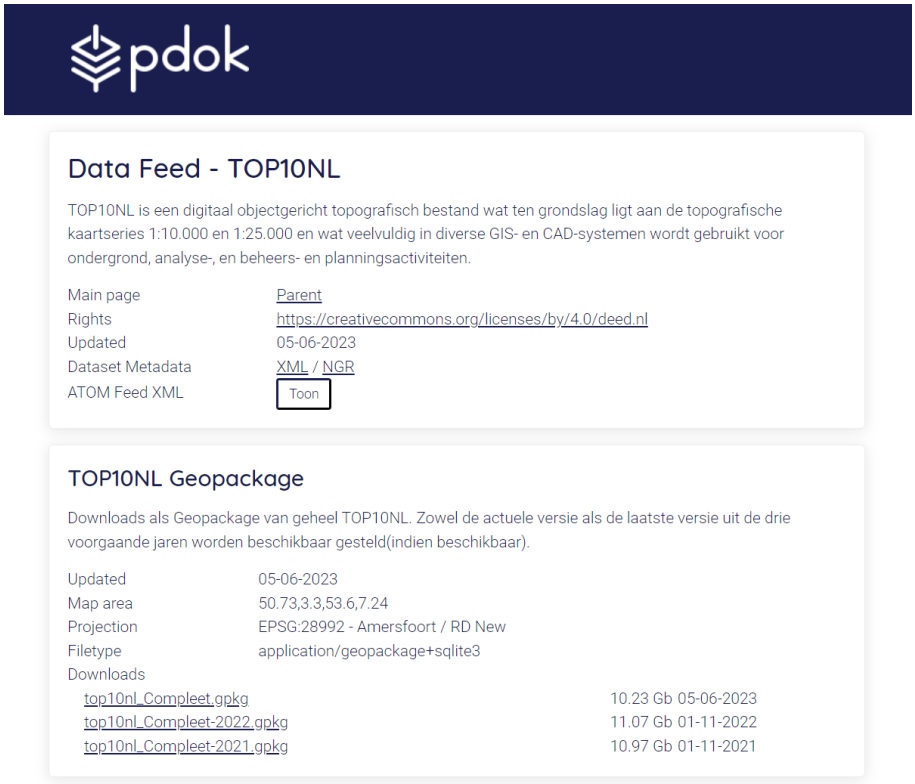


Figure 3.7: pdok website where to download TOP10NL

(railways), ‘top10nl_waterdeel_vlak‘ (water bodies), ‘top10nl_wegdeel_vlak‘ (roads) are selected to obtain additional SAR signatures by radarcoding, which are highlighted in blue, see Fig. 3.8, and later visualized by QGIS as the screenshot illustrated in Fig. 3.9.

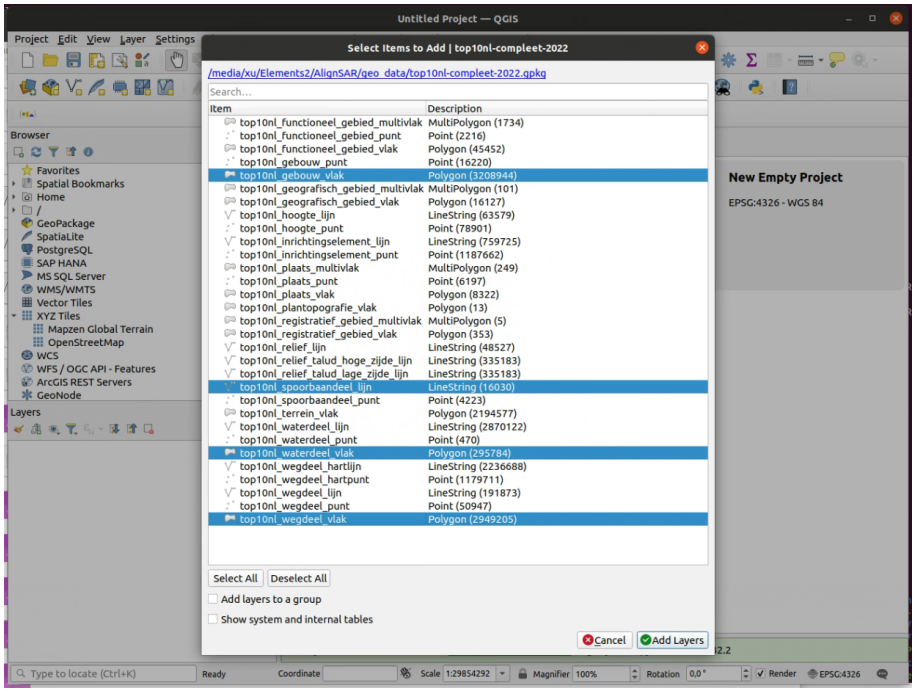


Figure 3.8: TOP10NL imported onto QGIS

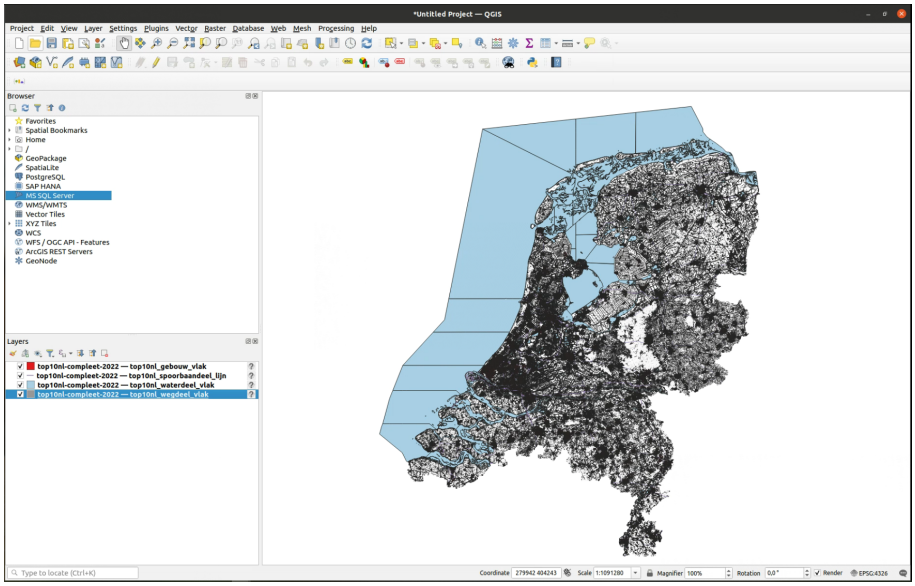


Figure 3.9: Overview of the four selected TOP10NL layers on QGIS

The radarcoding scripts can be found via ‘AlignSAR/alignSAR/rdrcode‘ on Github, as shown in Fig. 3.10. The user only needs to set the parameters in ‘Input_card.txt‘ file as shown in Fig. 3.11. In Line 8, the project ID can be defined here. From Line 14 to Line 21, these parameters are part of the parameters in Doris preprocessing. Note that in Line 17, for ’SensorName’, the user can keep it as e.g. ’s1’, if older SAR data is used like from Envisat. Such setting will not affect the radarcoding result. From Line 27 to Line 32, these parameters are prepared for the TOP10NL data for radarcoding. Line 27 is the TOP10NL data downloaded from PDOK. Line 28 can set the value to be ‘True‘ or ‘False‘ to crop the datasets or not. Line 29 is the data type of reference data. The data can be in polygon or raster format. Line 30 shows the layer name that we want to extract from the reference data. The ‘resolution‘ in Line 31 is suggested to be larger than 0.0001 decimal degrees. The ‘burn_val‘ in Line 32 is the space distance used to separate the adjacent layers. Line 38 can set the intermediate folder path to store the intermediate radarcoding products as shown in Fig. 3.12. Line 39 is prepared for the interferogram generation. To do it or not, the user can set ‘True‘ or ‘False.‘

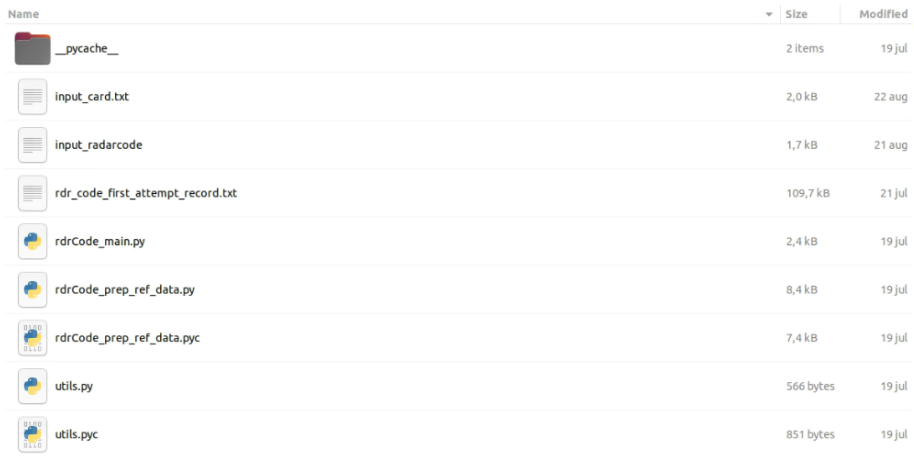


Figure 3.10: Radarcoding script files

After this parameter setup, the user can open the terminal and go to the folder where the ‘python rdrCode_main.py‘ file exists. Then, run ‘python rdrCode_main.py –inputFile input_card.txt‘, the radarcoding process will run automatically. Finally, the radarcoding products, e.g., ‘top10nl_gebouw_vlak_radarcoded.raw‘ can be found in the master image

```
% ----------------------------------------------------------------------
% Radarcoding project parameters
% ----------------------------------------------------------------------

%*************************************
% Project parameters
%*************************************
project_id = 'rdr_test'


%*************************************
% SAR data parameters
%*************************************
SARDataDir  =  '/media/xu/Elements2/AlignSAR/Doris_Processing/Doris_Processing_36_Groningen/sar_data_2022' %Path to the SAR data directory
dorisProcessDir  = '/media/xu/Elements2/AlignSAR/Doris_Processing/Doris_Processing_36_Groningen/new_datastack/stack_vv % 'Full path to the processing directory')
cropAoi     = '/media/xu/Elements2/AlignSAR/Doris_Processing/Doris_Processing_36_Groningen/AOI' % 'Full path to shapefile used to crop the data'
sensorName  = 's1'%  's1 | paz | tsx'
orbit_dir   = '/media/xu/Elements2/AlignSAR/Doris_Processing/Doris_Processing_36_Groningen/orbit_files'
startDate   = '20220109'
stopDate    = '20220322'
masterDate  = '20220214'

%*************************************
% Ref data params
%*************************************

refDataFile = '/media/xu/Elements2/AlignSAR/geo_data/top10nl-compleet-2022.gpkg'% 'Full path to shapefile/or geotiff raster for the reference data to be radarcoded'
cropSwitch  = 'True'% True | False % Whether the datasets need to be cropped to the cropAoi file boundary
refDataType = 'polygon' %polygon/raster
layerNames  = ['top10nl_wegdeel_vlak', 'top10nl_waterdeel_vlak', 'top10nl_gebouw_vlak', 'top10nl_spoorbaandeel_lijn'] %List of layer names in the shapefile to be
  radarcoded
resolution  = 0.00015%decimal degrees
burn_val    = 1 %m

%*************************************
% processing options
%*************************************

int_process_dir = '/media/xu/Elements/AlignSAR/radarcoding/groningen_processing_test %intermediate processing dir
doInSAR     = 'False'% Whether to do InSAR interferogram generation or not
```

Figure 3.11: Radarcoding inputcard

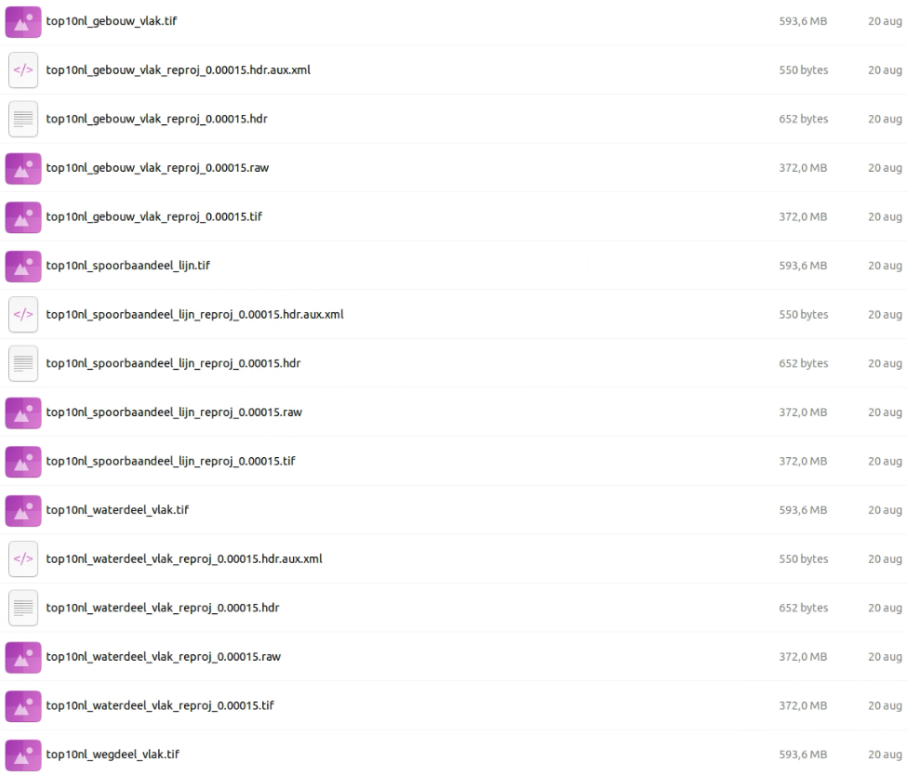


Figure 3.12: Radarcoding intermediate output

folder (../new_datastack/stack_vv/20220214). Note that in input_radarcode, on line 10 the default setting 'Height = 0.0 // average WGS84 height' is not used for radarcoding. This parameter HEIGHT (if specified) is only used for the image cropping calculation of the crop, using geo-coordinates information.

Alternatively, radarcoding of a GeoTIFF file in WGS-84 can be performed using functions from `snap_rdrcode.py` based on the outputs from SNAP containing orthorectified coordinates per pixel (e.g. outputs of SNAP2STAMPS) (function `snap_geo2rdc`), or corresponding outputs e.g. by GEOCODE step of doris (function `geo2rdc`).

A demonstration is shown below to illustrate how radarcoding works using SNAP. First of all, the SNAP radarcoding script can be downloaded in alignSAR Github (https://github.com/AlignSAR/alignSAR/blob/main/rdrcode/snap_rdrcode.py). Besides, `'bin'` and `'snap_graphs'` folders should be downloaded to the local computer. They are prepared for script installation. Then, there are two required inputs. One is the preprocessed `.dim` file generated by using SNAP, as shown in Fig. 3.13. The 'orthorectifiedLat' and 'orthorectifiedLon' must be provided for the input. Another one is the image that needs to be radarcoded. Finally, following the scripts in `'snap_radarcode.py'`, from Line 7 to 25, the radarcoding process can be done using jupyter notebook.

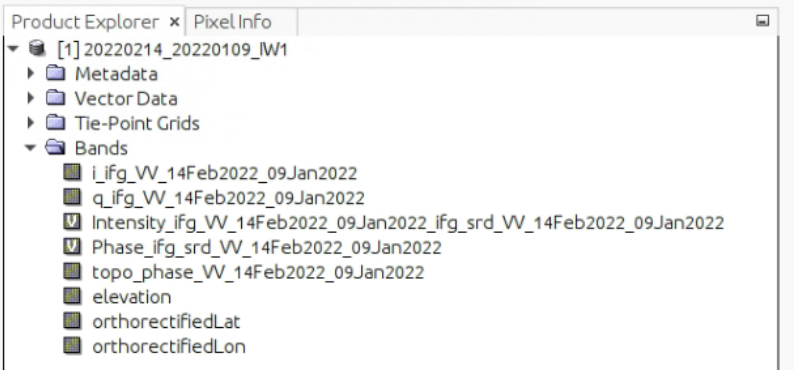


Figure 3.13: SNAP's output '.dim' file demonstration.

3.3 SAR signature extraction demonstration

As we have done in Sec. 3.1.1, to obtain SAR benchmark datesets first in radar coordinates (ultimately in geo-coordinates) using Doris, we apply a standard interferogram generation procedure using Doris for these seven Sentinel-1 SAR images in VV and VH. All coregistered images can be found on https://alignsar.stargazer-cod.ts.net/ITC_data /demo_data.zip. The standard output of these coregistered images covers the folders listed in Fig. 3.14, and in each 'date' folder, it has raw images (.raw) and their preview images in .ras and log/dumpy files, see Fig. 3.15. For instance, 'cint_srd.raw' is the resultant interferogram after removing reference and topographic phase, 'coherence.raw' is the coherence maps, and 'slave_rsmp.raw' is the coregistered and resampled slave image. Please note that there are no 'cint_srd.raw' and 'coherence.raw' for the master 'date folder' – '20220214'.

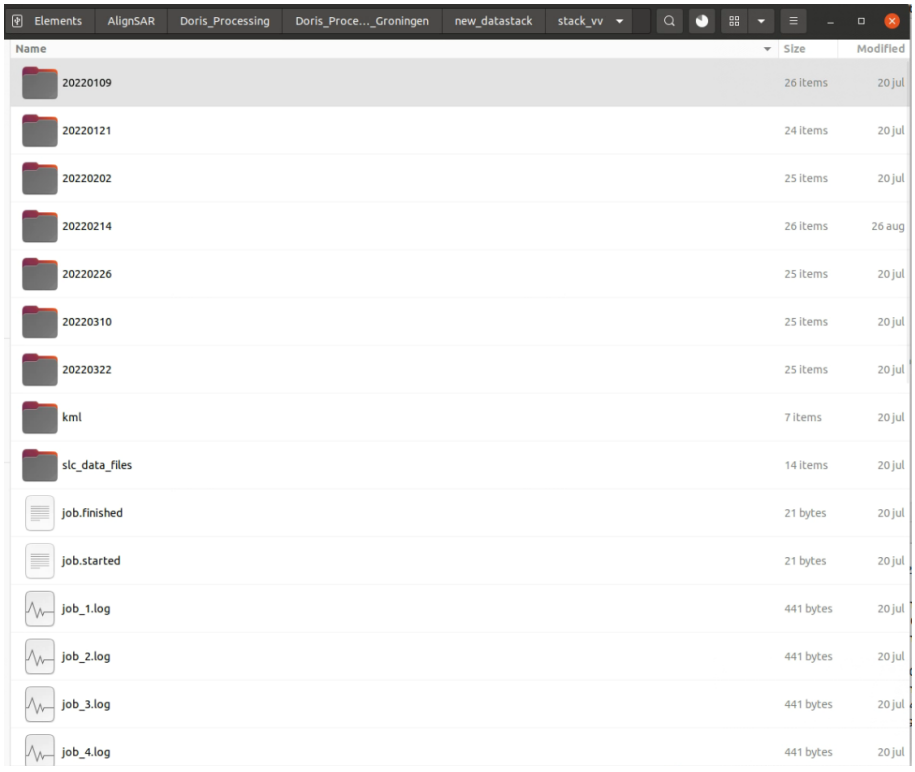


Figure 3.14: Overview of Doris preprocessed output.

The SAR signatures contain

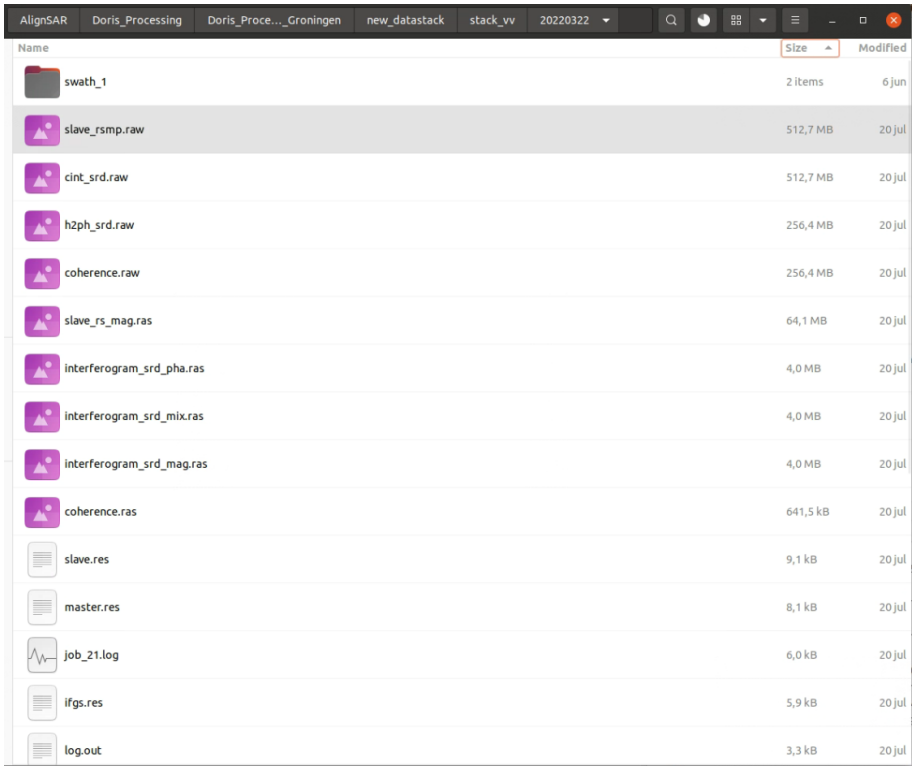


Figure 3.15: Output from Doris-based interferogram generation in a slave 'date' folder

1. VV amplitude (linear),
2. VH amplitude (linear),
3. VV interferometric phase [in radians],
4. VV coherence,
5. Intensity summation $|S_{VV}|^2 + |S_{VH}|^2$, (S_{VV} and S_{VH} represent complex number in VV and VH)
6. Intensity difference (dual-pol difference) $|S_{VV}|^2 - |S_{VH}|^2$,
7. Intensity ratio (dual-pol power ratio) $|S_{VV}|^2/|S_{VH}|^2$,
8. cross-pol correlation coefficient,
9. cross-pol cross product,

10. entropy,
11. buildings (TOP10NL),
12. roads (TOP10NL),
13. water (TOP10NL),
14. railways (TOP10NL).

These SAR signatures are extracted and calculated based on 'cint_srd.raw', 'coherence.raw', and 'slave_rsmp.raw' mentioned above. A NetCDF data format is prepared to contain the signatures along with their global and local attributes. Particularly, global and local attributes are defined separately as shown in Table 3.1 and Table 3.2. The global attributes describe the basic information for all the signatures. For instance, 'sar_date_time' describes the acquisition time of the current SLC image where all the signatures are extracted, as shown in Fig. 3.16. The local attributes belong to the signatures that are stored in 'variables' as shown in Fig. 3.16. As an example, 'VV interferometric phase', its local attributes: unit is radians, the format is float32, the range is between $(-\pi,\ \pi]$, and description is the phase difference between master and slave acquisition.

To build up the NetCDF data format with global and local attributes, the user should give the path of the unzipped raw SLC folder to parameter 'sar_folder_path' in 'signature_extraction.py' Line 311, e.g., 'sar_folder_path = /(your doris processing path)/Doris_Processing_36_Groningen/sar_data_2022/'. The next folder level should contain 7 raw SLC folders. Then, in the 'Meta_info_extraction_global_local.py' file, the user can define the global attributes themselves from Line 61 to 69. The attributes before 'sar_date_time' can be added or deleted manually, while the later attributes will be extracted from 'xml' files of unzipped raw SLC folder automatically, as shown in Fig. 3.18. Lines 80 to 198 have prepared local attributes for 14 signatures. The local attributes for each signature, e.g., 'VV_amplitude_attr', are defined in a dictionary format with keywords and their value, e.g., keywords 'Format' and its value 'float32", as shown in Fig. 3.19. The user is free to add or delete the keywords and values.

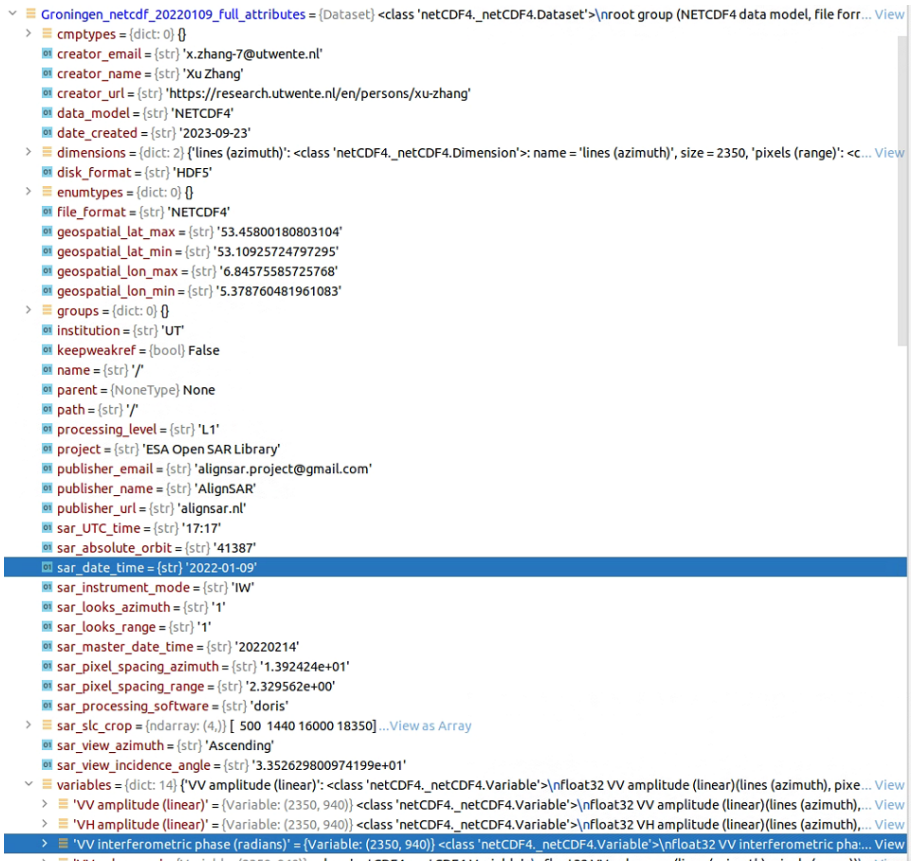


Figure 3.16: Global attributes example of NetCDF data format.

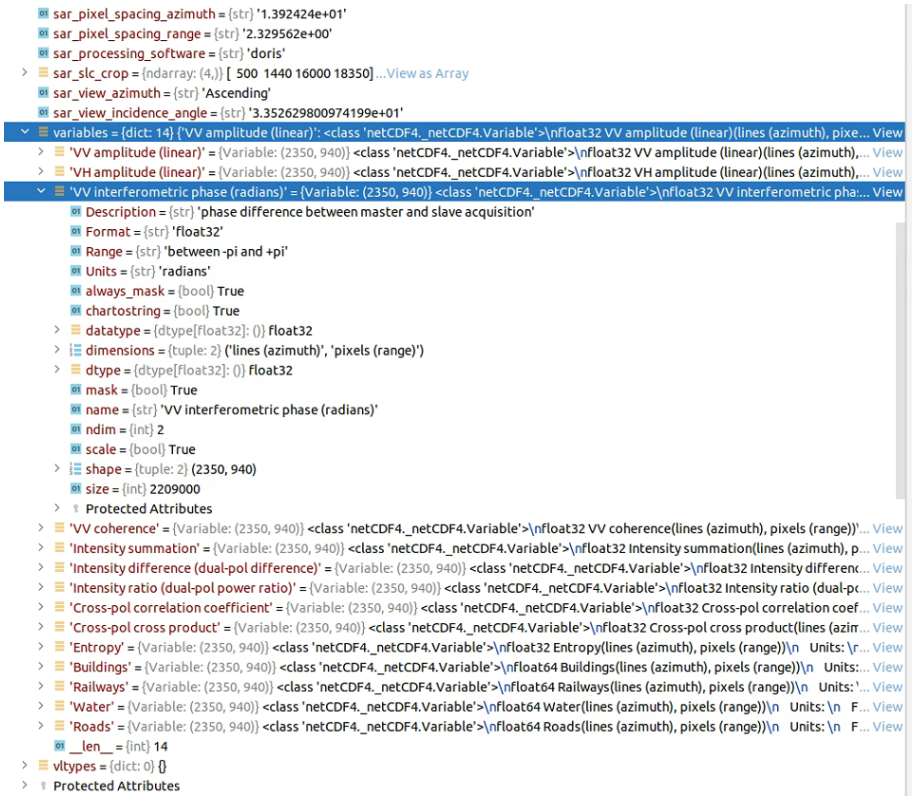


Figure 3.17: Local attributes example of NetCDF data format.

```
61      Global_attr = {'processing_level':'L1', 'date_created': '2023-09-23', 'creator_name': 'Xu Zhang', 'creator_email': 'x.zhang-7@utwente.nl', \
62                'creator_url': 'https://research.utwente.nl/en/persons/xu-zhang', 'institution': 'UT', 'project': 'ESA Open SAR Library', 'publisher_name': 'Alig
63                'publisher_email': 'alignsar.project@gmail.com', 'publisher_url': 'alignsar.nl', 'geospatial_lat_min': '53.10925724797295', \
64                'geospatial_lat_max': '53.45800180803104', 'geospatial_lon_min': '5.378760481961083', 'geospatial_lon_max': '6.84575585725768', \
65                'sar_date_time':sar_date_time, 'sar_master_date_time':sar_master_date_time, 'sar_UTC_time':sar_UTC_time, 'sar_instrument_mode':sar_instrument_mode
66                'sar_looks_range':sar_looks_range, 'sar_looks_azimuth':sar_looks_azimuth, 'sar_pixel_spacing_range':sar_pixel_spacing_range, \
67                'sar_pixel_spacing_azimuth':sar_pixel_spacing_azimuth, 'sar_processing_software':sar_processing_software, \
68                'sar_absolute_orbit':sar_absolute_orbit, 'sar_view_azimuth':sar_view_azimuth, 'sar_view_incidence_angle':sar_view_incidence_angle, \
69                'sar_slc_crop':sar_slc_crop}
70      return Global_attr
```

Figure 3.18: Global attributes in Python file.

```
80    VV_amplitude_attr={
81          'Units': 'voltage [linear]',\
82          'Format': 'float32',\
83          'Description': 'the absolute value of every complex number in VV channel'
84      }
85
```

Figure 3.19: Local attributes in Python file.

Having the prepared global and local attributes, we will continue with the signatures. In ‘signature_extraction.py‘, as shown in Fig. 3.20, Lines 249 and 250 should be given the paths to the VV and VH preprocessed folder, e.g., ‘(your data folder)/new_data stack/stack_vv/.‘ It is similar to the VH mode in Line 250. In Line 252, the user should set the master date to avoid reading master coherence or interferogram maps. In Line 253, the user can choose whether to crop the images. If setting ‘True‘ in the following Line 254, the user can define the crop region. The parameters of ‘CRP_LIST‘ are the first line, last line, first pixel, and last pixel, assuming the coordinate starts from the top left. Line 255 shows the maximum number of SAR images. Line 257 indicates the type of reading maps. Three map types are defined here, ‘cpx‘ represents complex SAR images, ‘ifg‘ means interferogram, and ‘coh‘ is the coherence map. They are prepared to generate SAR signatures 1-10. As we have seven SAR SLC acquisitions, each acquisition has signatures 1-10. Then, Line 266 is the folder path to the TOP10NL radarcoding signatures. From Line 268 to Line 271, the user can give the file name of the radarcoding files. These four variables are prepared for signatures 11-14. Finally, using the scripts from Line 273 to 301, we have extracted and calculated 14 signatures stored in 2-dimensional numpy arrays, which will be stored in the ‘variables‘ along with their local attributes, as shown in Fig. 3.17.

At this stage, the global and local attributes, and the 14 signatures are prepared. A loop operation is described from Line 312 to 343 to assemble them into NetCDF data format. Line 312 defines the number of acquisitions, which also means the number of NetCDF data files. In each NetCDF data file, we name it using its acquisition date in Line 321. The signature size and global attributes are defined from Line 325 to Line 329. Line 331 gives the number of signatures. Later on, another loop builds up each signature along with its local attributes into the NetCDF data file. Please note that from Lines 334 to 337, conditional lines are used to define the signature format, since the four TOP10NL layers are ’float64’ and the others are ’float32’.

```python
if __name__=='__main__':
    doris_stack_dir_VV = '/media/xu/Elements2/AlignSAR/Doris_Processing/Doris_Processing_36_Groningen/new_datastack/stack_vv/'
    doris_stack_dir_VH = '/media/xu/Elements2/AlignSAR/Doris_Processing/Doris_Processing_36_Groningen/new_datastack/stack_vh/'
    #paz_doris_stack = '/media/anurag/AK_WD/PAZ_Processing/stack'
    master_date = 20220214 #'20220214'#'20170117'
    CROPPING = True
    CRP_LIST = [500, 1440, 16000, 18350]#[2000, 3500, 8500, 10000]# [first line, last line, first pixel, last pixel]
    MAX_IMAGES = 30
    # SP_AVG_WIN_SIYE = 3
    map_type = 'cpx' # 'cpx', 'ifg', 'coh'

    #Get the dates
    dates = get_dates(doris_stack_dir_VV, master_date)[:MAX_IMAGES]
    print(dates)
    #Extract the stack array
    vv_arr_stack = get_stack(dates, master_date, doris_stack_dir_VV, map_type, crop_switch=CRP_LIST, crop_list=CRP_LIST, sensor='s1')
    vh_arr_stack = get_stack(dates, master_date, doris_stack_dir_VH, map_type, crop_switch=CRP_LIST, crop_list=CRP_LIST, sensor='s1')

    top10_path = '/media/xu/Elements2/AlignSAR/Doris_Processing/Doris_Processing_36_Groningen/new_datastack/stack_vv/20220214/'
    # top10NL layers' file name
    top10_building_file_name = top10_path + 'top10nl_gebouw_vlak_radarcoded.raw'
    top10_railway_file_name = top10_path + 'top10nl_spoorbaandeel_lijn_radarcoded.raw'
    top10_water_file_name = top10_path + 'top10nl_waterdeel_vlak_radarcoded.raw'
    top10_road_file_name = top10_path + 'top10nl_wegdeel_vlak_radarcoded.raw'
```

Figure 3.20: Parameters need to be changed in '*singnature_extraction.py*'.

Field name	SAR usage: Global attributes
processing_level	SAR data processing level.
date_created	The creation date of SAR data.
creator_name	Creator's name.
creator_email	Creator's contact information.
creator_url	Creator's webpage.
institution	Creator's affiliation.
project	Project's name.
publisher_name	Publisher's name.
publisher_email	Publisher's contact information.
publisher_url	Publisher's official website.
geospatial_lat_min	data spatial coverage, minimum latitude value [decimal degrees].
geospatial_lat_max	data spatial coverage, maximum latitude value [decimal degrees].
geospatial_lon_min	data spatial coverage, minimum longitude value [decimal degrees].
geospatial_lon_max	data spatial coverage, maximum longitude value [decimal degrees].
sar_date_time	Center date time of the product, in UTC.
sar_reference_date_time	Reference image acquisition time, in UTC.
sar_instrument_mode	The name of the sensor acquisition mode that is used.
sar_looks_range	Number of range looks, which is the number of groups of signal samples (looks) perpendicular to the flight path.
sar_looks_azimuth	Number of azimuth looks, which is the number of groups of signal samples (looks) along the flight path.
sar_pixel_spacing_range	The range pixel spacing is the distance between adjacent pixels perpendicular to the flight path in meters (m).
sar_pixel_spacing_azimuth	The azimuth pixel spacing is the distance between adjacent pixels parallel to the flight path in meters (m).
sar_processing_software	Software used for InSAR processing. For instance "ESA SNAP Toolbox": "8.0", "SNAPHU": "1.4.2".
sar_absolute_orbit	Absolute orbit (track) of the input datasets.
sar_relative_orbit	Relative orbit (track) of the input datasets.
sar_view_azimuth	The azimuth angle (heading) of the center of the product.
sar_view_incidence_angle	The incidence angle of the center of the product.
sar_slc_crop	The crop region based on AOI. From left to right are the first line, last line, first pixel, and last pixel.

Table 3.1: Metadata/attributes information

Field name	SAR usage: Local attributes
unit	The unit of the SAR signature.
format	The format of the SAR signature.
range	The value range of the SAR signature.
description	The detailed description of the SAR signature.

Table 3.2: Cont. Metadata/attributes information

3.4 Machine learning scripts for LULC classification

After obtaining 14 SAR signatures, an ANN (artificial neural network) is implemented for land use land cover (LULC) classification, see the scripts in the folder 'MLscripts/Netherlands-LULC'.

Appendix A

Doris installation, implementation and trouble shooting

In this appendix, we introduce the installation of Doris-5, and offer solutions to some installation and implementation errors.

We made updates upon the Doris-5 published by the radar group at the Delft University of Technology on TUDelftGeodesy github. The updated Doris-5 scripts can be found via AlignSAR on github, or Doris-Surfdrive. Note that if you download Doris-5 package via TUDelftGeodesy, be sure that you rename 'Doris-master' to 'doris'. Because having the hyphen '-' in a software directory will introduce errors when invoking Doris-5 python scripts that are under the folder 'Doris-master'. Doris-Surfdrive also provides some Doris-5-required software packages under the folder 'doris5_required_software'. As Doris-5 is built upon python2 and only compatible with the old version of the required software packages, we recommend using the packages in 'doris5_required_software'.

Here we give an installation demonstration, based on the installation descriptions in INSTALL.txt on TUDelftGeodesy. The operating system we recommend is Ubuntu:18.04.

- Installation of FFTW library (fftw-3.2.1 is offline)

  In the terminal, create a folder where to install software, e.g. /home/username/software, then run

wget -c http://www.fftw.org/pub/fftw/fftw-3.2.2.tar.gz

gunzip fftw-3.2.2.tar.gz

tar xvf fftw-3.2.2.tar

cd fftw-3.2.2/

./configure –prefix=‘pwd‘ –enable-float

make

make install

- Compilation of the doris core run

  cd ../doris_core

  ./configure

  Edit 'Makefile', here is an example:

  ———————

  compiler: g++

  fftw: y

  FFTW LIB DIR: /home/username/software/fftw-3.2.2/lib

  FFTW INCLUDE DIR: /home/username/software/fftw-3.2.2/include

  veclib: n

  lapack: n

  Little endian: y

  DEBUG version: n

  Install in dir: /usr/local/bin

  ———————

  (Note that g++ has the version of 7.5.0.)

  when running 'make' in the terminal, you may encounter an error like

---

bk_messages.hh:214:26: error: invalid conversion from ‘char’ to ‘const char*’ [-fpermissive]

strcat(name_,’\0’); ;// terminate id

The solution is to edit ’bk_messages.hh’, change its Line 214 strcat(name_,’\0’); to name_[9] = ’\0’;

Then run ’sudo make install’. If ’construct_dem.sh’ does not exist in doris/bin, then copy one to ’bin’ folder.

- Compilation of Doris utilities

  cd ../sartools

  In ’Makefile’, edit ’INSTALL_DIR’, like INSTALL_DIR = /usr/local/bin and edit cpxfiddle.cc, change if (argv[optind]==’\0’) to if (*argv[optind]==’\0’)

  then run ’make’ and ’sudo make install’

  cd ../envisat_tools

  In ’Makefile’, change INSTALL_DIR = /usr/local/bin, and CC = g++ //[it was gcc]

  then run ’make’ and ’sudo make install’

- Installation of useful external software

  gfortran: version is 7.5.0

  gunzip getorb_doris.tar.gz

  tar xvf getorb_doris.tar

  In ’Makefile’: change

  FC = gfortran

- Installation of Doris - Python part

  Python has the version of 2.7. use pip2 to install Python packages, for instance:

  sudo pip2 install requests

  pip2 install GDAL==2.2.2

  pip2 install shapely==1.7.1

  sudo pip2 install pygeoif

  sudo pip2 install lxml

  sudo pip install -U pyopenssl

  sudo pip install numpy scipy matplotlib requests fiona pyproj fastkml osr

  In case you encounter error during GDAL installation, try to install libgdal first and then set up appropriate paths:

  sudo apt-get install libgdal-dev

  export CPLUS_INCLUDE_PATH=/usr/include/gdal

  export C_INCLUDE_PATH=/usr/include/gdal

Here we showcase the Doris-5 implementation.

- Create an account

  for the downloading of SRTM DEMs at https://urs.earthdata.nasa.gov/users/new,
  and create an account for downloading Sentinel data at https://urs.earthdata.nasa.gov/users/new/

- Move to the install directory

  cd ../install

  python init_cfg.py and fill in the different paths and user accounts. Here is an example:

  Enter the path to doris: /usr/local/bin/doris

  Enter the path to cpxfiddle: /usr/local/bin/cpxfiddle

Enter the path to snaphu: /usr/local/bin/snaphu

Enter your username for scihub (https://scihub.copernicus.eu/dhus/#/self-registration)lchang?

Enter your password for scihub ?

Enter your username for srtm download (https://urs.earthdata.nasa.gov/users/new/)Chang?

Enter your password for srtm download ?

Doris is initialized. If you want to make changes later, you can change the doris_config.xml file in 'install' folder or run this script again.

- Run the stack preparation script (Move to the prepare_stack directory)

  cd prepare_stack

  Run the python script:

  python prepare_datastack_main.py

  Here is an example:

  Enter the path to the archive data folder: /home/ling/d2/groningendata

  Which polarisation do you want to use (vv,hh,vh,hv): vv

  Which track do you want to work with? (explore on https://scihub.copernicus.eu/dhus/) : 15

  Is this track ascending or descending? (asc/dsc) : asc

  Enter the path to the folder of new datastack: /home/ling/d2/groningen/processing

  Enter full path to the shapefile: /home/ling/d2/groningen/aoi/G1.shp

  Enter the path to the folder of the orbit files: /home/ling/d2/groningen_orbit

  Do you want to generate the DEM file automatically (Yes/No): Yes

  Enter path to the dem folder: /home/ling/d2/groningen/DEM

  Do you want to use parallel computing (Yes/No): Yes

  How many cores do you want to use: 2

What is the start date of your stack in yyyy-mm-dd (can be changed later): 2022-01-09

What is the end date of your stack in yyyy-mm-dd (can be changed later): 2022-03-22

What is the master date of your stack in yyyy-mm-dd (can be changed later): 2022-02-14

Note that track number is the path number of Sentinel-1, and edit 'create_dem.py' and change 'server = http://e4ftl01.cr.usgs.gov' to 'server = https://e4ftl01.cr.usgs.gov'. The doris directory can be edited in 'doris_config.xml' under the folder 'software/doris/install', like <source_path>/home/username/software/doris</source_path>

In case you encounter error like ImportError: No module named html.parser, simply change 'from html.parser import HTMLParser' to 'from HTMLParser import HTMLParser' in create_dem.py

- run bash doris_stack.sh

  be sure that you edit 'doris_main.py' in 'software/doris/doris_stack/main_code' folder, add edit sys.path.append('/home/username/software/') and provide the absolute directory of Doris-5 software.

Appendix B

Docker installation instruction reference

The instruction of Docker software installation on e.g. Ubuntu refers to DigitalOcean and can be found on the next pages.

It may take 40mins to complete Docker software installation by following this instruction.

// Tutorial //

How To Install and Use Docker on Ubuntu 20.04

By Brian Hogan
https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04

Introduction

Docker is an application that simplifies the process of managing application processes in *containers*. Containers let you run your applications in resource-isolated processes. They're similar to virtual machines, but containers are more portable, more resource-friendly, and more dependent on the host operating system. For a detailed introduction to the different components of a Docker container, check out The Docker Ecosystem: An Introduction to Common Components.

In this tutorial, you'll install and use Docker Community Edition (CE) on Ubuntu 20.04. You'll install Docker itself, work with containers and images, and push an image to a Docker Repository.

Prerequisites

To follow this tutorial, you will need the following:

- One Ubuntu 20.04 server set up by following the Ubuntu 20.04 initial server setup guide, including a sudo non-root user and a firewall.
- An account on Docker Hub if you wish to create your own images and push them to Docker Hub, as shown in Steps 7 and 8.

Step 1 — Installing Docker

The Docker installation package available in the official Ubuntu repository may not be the latest version. To ensure we get the latest version, we'll install Docker from the official Docker repository. To do that, we'll add a new package source, add the GPG key from Docker to ensure the downloads are valid, and then install the package.

First, update your existing list of packages:

```
1. sudo apt update
2.
```

Next, install a few prerequisite packages which let `apt` use packages over HTTPS:

```
1. sudo apt install apt-transport-https ca-certificates curl
   software-properties-common
2.
```

Then add the GPG key for the official Docker repository to your system:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
apt-key add -
```

Add the Docker repository to APT sources:

```
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu focal stable"
```

This will also update our package database with the Docker packages from the newly added repo.

Make sure you are about to install from the Docker repo instead of the default Ubuntu repo:

```
apt-cache policy docker-ce
```

You'll see output like this, although the version number for Docker may be different:

Output of apt-cache policy docker-ce

```
docker-ce:
  Installed: (none)
  Candidate: 5:19.03.9~3-0~ubuntu-focal
  Version table:
     5:19.03.9~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64
Packages
```

Notice that `docker-ce` is not installed, but the candidate for installation is from the Docker repository for Ubuntu 20.04 (`focal`).

Finally, install Docker:

```
sudo apt install docker-ce
```

Docker should now be installed, the daemon started, and the process enabled to start on boot. Check that it's running:

```
sudo systemctl status docker
```

The output should be similar to the following, showing that the service is active and running:

```
Output
```

```
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled;
vendor preset: enabled)
     Active: active (running) since Tue 2020-05-19 17:00:41 UTC; 17s
ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 24321 (dockerd)
      Tasks: 8
     Memory: 46.4M
     CGroup: /system.slice/docker.service
             └─24321 /usr/bin/dockerd -H fd:// --
containerd=/run/containerd/containerd.sock
```

Installing Docker now gives you not just the Docker service (daemon) but also the `docker`command line utility, or the Docker client. We'll explore how to use the `docker` command later in this tutorial.

Step 2 — Executing the Docker Command Without Sudo (Optional)

By default, the `docker` command can only be run the **root** user or by a user in the **docker** group, which is automatically created during Docker's installation process. If you attempt to run the `docker` command without prefixing it with `sudo` or without being in the **docker** group, you'll get an output like this:

```
Output
docker: Cannot connect to the Docker daemon. Is the docker daemon
running on this host?.
See 'docker run --help'.
```

If you want to avoid typing `sudo` whenever you run the `docker` command, add your username to the `docker` group:

```
1. sudo usermod -aG docker ${USER}
2.
```

To apply the new group membership, log out of the server and back in, or type the following:

```
1. su - ${USER}
2.
```

You will be prompted to enter your user's password to continue.

Confirm that your user is now added to the **docker** group by typing:

```
1. groups
2.
```

```
Output
sammy sudo docker
```

If you need to add a user to the `docker` group that you're not logged in as, declare that username explicitly using:

```
1. sudo usermod -aG docker username
2.
```

The rest of this article assumes you are running the `docker` command as a user in the **docker**group. If you choose not to, please prepend the commands with `sudo`. Let's explore the `docker` command next.

Step 3 — Using the Docker Command

Using `docker` consists of passing it a chain of options and commands followed by arguments. The syntax takes this form:

```
1. docker [option] [command] [arguments]
2.
```

To view all available subcommands, type:

```
1. docker
2.
```

As of Docker 19, the complete list of available subcommands includes:

```
Output
  attach      Attach local standard input, output, and error streams to
a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local
filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's
filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem
image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
```

```
  logs          Fetch the logs of a container
  pause         Pause all processes within one or more containers
  port          List port mappings or a specific mapping for the
container
  ps            List containers
  pull          Pull an image or a repository from a registry
  push          Push an image or a repository to a registry
  rename        Rename a container
  restart       Restart one or more containers
  rm            Remove one or more containers
  rmi           Remove one or more images
  run           Run a command in a new container
  save          Save one or more images to a tar archive (streamed to
STDOUT by default)
  search        Search the Docker Hub for images
  start         Start one or more stopped containers
  stats         Display a live stream of container(s) resource usage
statistics
  stop          Stop one or more running containers
  tag           Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top           Display the running processes of a container
  unpause       Unpause all processes within one or more containers
  update        Update configuration of one or more containers
  version       Show the Docker version information
  wait          Block until one or more containers stop, then print their
exit codes
```

To view the options available to a specific command, type:

```
1. docker docker-subcommand --help
2.
```

To view system-wide information about Docker, use:

```
1. docker info
2.
```

Let’s explore some of these commands. We’ll start by working with images.

Step 4 — Working with Docker Images

Docker containers are built from Docker images. By default, Docker pulls these images from Docker Hub, a Docker registry managed by Docker, the company

behind the Docker project. Anyone can host their Docker images on Docker Hub, so most applications and Linux distributions you'll need will have images hosted there.

To check whether you can access and download images from Docker Hub, type:

```
1. docker run hello-world
2.
```

The output will indicate that Docker in working correctly:

```
Output
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest:
sha256:6a65f928fb91fcfbc963f7aa6d57c8eeb426ad9a20c7ee045538ef34847f44f1
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working
correctly.

...
```

Docker was initially unable to find the `hello-world` image locally, so it downloaded the image from Docker Hub, which is the default repository. Once the image downloaded, Docker created a container from the image and the application within the container executed, displaying the message.
You can search for images available on Docker Hub by using the `docker` command with the `search`subcommand. For example, to search for the Ubuntu image, type:

```
1. docker search ubuntu
2.
```

The script will crawl Docker Hub and return a listing of all images whose name match the search string. In this case, the output will be similar to this:

```
Output
NAME                                                    DESCRIPTION
STARS               OFFICIAL            AUTOMATED
ubuntu                                                  Ubuntu is a
Debian-based Linux operating sys…   10908               [OK]
dorowu/ubuntu-desktop-lxde-vnc                          Docker image
to provide HTML5 VNC interface …   428
[OK]
```

```
rastasheep/ubuntu-sshd                                          Dockerized
SSH service, built on top of offi…   244
[OK]
consol/ubuntu-xfce-vnc                                          Ubuntu
container with "headless" VNC session…   218
[OK]
ubuntu-upstart                                                  Upstart is an
event-based replacement for th…   108                  [OK]
ansible/ubuntu14.04-ansible                                     Ubuntu 14.04
LTS with
...
```

In the **OFFICIAL** column, **OK** indicates an image built and supported by the company behind the project. Once you've identified the image that you would like to use, you can download it to your computer using the `pull` subcommand.
Execute the following command to download the official `ubuntu` image to your computer:

```
1. docker pull ubuntu
2.
```

You'll see the following output:

```
Output
Using default tag: latest
latest: Pulling from library/ubuntu
d51af753c3d3: Pull complete
fc878cd0a91c: Pull complete
6154df8ff988: Pull complete
fee5db0ff82f: Pull complete
Digest:
sha256:747d2dbbaaee995098c9792d99bd333c6783ce56150d1b11e333bbceed5c54d7
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

After an image has been downloaded, you can then run a container using the downloaded image with the `run` subcommand. As you saw with the `hello-world` example, if an image has not been downloaded when `docker` is executed with the `run` subcommand, the Docker client will first download the image, then run a container using it.

To see the images that have been downloaded to your computer, type:

```
1. docker images
2.
```

The output will look similar to the following:

```
Output
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
ubuntu              latest              1d622ef86b13        3 weeks ago
73.9MB
hello-world         latest              bf756fb1ae65        4 months
ago         13.3kB
```

As you'll see later in this tutorial, images that you use to run containers can be modified and used to generate new images, which may then be uploaded (*pushed* is the technical term) to Docker Hub or other Docker registries.

Let's look at how to run containers in more detail.

Step 5 — Running a Docker Container

The `hello-world` container you ran in the previous step is an example of a container that runs and exits after emitting a test message. Containers can be much more useful than that, and they can be interactive. After all, they are similar to virtual machines, only more resource-friendly.

As an example, let's run a container using the latest image of Ubuntu. The combination of the **-i** and **-t** switches gives you interactive shell access into the container:

```
1. docker run -it ubuntu
2.
```

Your command prompt should change to reflect the fact that you're now working inside the container and should take this form:

```
Output
root@d9b100f2f636:/#
```

Note the container id in the command prompt. In this example, it is `d9b100f2f636`. You'll need that container ID later to identify the container when you want to remove it.

Now you can run any command inside the container. For example, let's update the package database inside the container. You don't need to prefix any command with `sudo`, because you're operating inside the container as the **root** user:

```
1. apt update
2.
```

Then install any application in it. Let's install Node.js:

```
1. apt install nodejs
2.
```

This installs Node.js in the container from the official Ubuntu repository. When the installation finishes, verify that Node.js is installed:

```
1. node -v
```

```
2.
```

You'll see the version number displayed in your terminal:

```
Output
v10.19.0
```

Any changes you make inside the container only apply to that container.

To exit the container, type `exit` at the prompt.

Let's look at managing the containers on our system next.

Step 6 — Managing Docker Containers

After using Docker for a while, you'll have many active (running) and inactive containers on your computer. To view the **active ones**, use:

```
1. docker ps
2.
```

You will see output similar to the following:

```
Output
CONTAINER ID        IMAGE               COMMAND             CREATED
```

In this tutorial, you started two containers; one from the `hello-world` image and another from the `ubuntu` image. Both containers are no longer running, but they still exist on your system.
To view all containers — active and inactive, run `docker ps` with the `-a` switch:

```
1. docker ps -a
2.
```

You'll see output similar to this:

```
1c08a7a0d0e4        ubuntu              "/bin/bash"         2 minutes
ago         Exited (0) 8 seconds ago
quizzical_mcnulty
a707221a5f6c        hello-world         "/hello"            6 minutes
ago         Exited (0) 6 minutes ago                       youthful_curie
```

To view the latest container you created, pass it the `-l` switch:

```
1. docker ps -l
2.
1. CONTAINER ID        IMAGE               COMMAND
   CREATED             STATUS                      PORTS
   NAMES
2.
```

```
3. 1c08a7a0d0e4          ubuntu                "/bin/bash"           2
   minutes ago         Exited (0) 40 seconds ago
   quizzical_mcnulty
4.
5.
6.
```

To start a stopped container, use `docker start`, followed by the container ID or the container's name. Let's start the Ubuntu-based container with the ID of `1c08a7a0d0e4`:

```
1. docker start 1c08a7a0d0e4
2.
```

The container will start, and you can use `docker ps` to see its status:

```
Output
CONTAINER ID         IMAGE                COMMAND               CREATED
STATUS               PORTS                NAMES
1c08a7a0d0e4         ubuntu               "/bin/bash"           3 minutes
ago         Up 5 seconds                              quizzical_mcnulty
```

To stop a running container, use `docker stop`, followed by the container ID or name. This time, we'll use the name that Docker assigned the container, which is `quizzical_mcnulty`:

```
1. docker stop quizzical_mcnulty
2.
```

Once you've decided you no longer need a container anymore, remove it with the `docker rm`command, again using either the container ID or the name. Use the `docker ps -a` command to find the container ID or name for the container associated with the `hello-world` image and remove it.

```
1. docker rm youthful_curie
2.
```

You can start a new container and give it a name using the `--name` switch. You can also use the `--rm` switch to create a container that removes itself when it's stopped. See the `docker run help`command for more information on these options and others.

Containers can be turned into images which you can use to build new containers. Let's look at how that works.

Step 7 — Committing Changes in a Container to a Docker Image

When you start up a Docker image, you can create, modify, and delete files just like you can with a virtual machine. The changes that you make will only apply to that container. You can start and stop it, but once you destroy it with the `docker rm` command, the changes will be lost for good.

This section shows you how to save the state of a container as a new Docker image.

After installing Node.js inside the Ubuntu container, you now have a container running off an image, but the container is different from the image you used to create it. But you might want to reuse this Node.js container as the basis for new images later.

Then commit the changes to a new Docker image instance using the following command.

```
1. docker commit -m "What you did to the image" -a "Author Name"
   container_id repository/new_image_name
2.
```

The **-m** switch is for the commit message that helps you and others know what changes you made, while **-a** is used to specify the author. The `container_id` is the one you noted earlier in the tutorial when you started the interactive Docker session. Unless you created additional repositories on Docker Hub, the `repository` is usually your Docker Hub username.
For example, for the user **sammy**, with the container ID of `d9b100f2f636`, the command would be:

```
1. docker commit -m "added Node.js" -a "sammy" d9b100f2f636
   sammy/ubuntu-nodejs
2.
```

When you *commit* an image, the new image is saved locally on your computer. Later in this tutorial, you'll learn how to push an image to a Docker registry like Docker Hub so others can access it.

Listing the Docker images again will show the new image, as well as the old one that it was derived from:

```
1. docker images
2.
```

You'll see output like this:

```
Output
REPOSITORY                TAG                 IMAGE ID
CREATED              SIZE
sammy/ubuntu-nodejs   latest              7c1f35226ca6          7 seconds
ago         179MB
...
```

In this example, `ubuntu-nodejs` is the new image, which was derived from the existing `ubuntu`image from Docker Hub. The size difference reflects the changes that were made. And in this example, the change was that NodeJS was installed. So next time you need to run a container using Ubuntu with NodeJS pre-installed, you can just use the new image.

You can also build Images from a `Dockerfile`, which lets you automate the installation of software in a new image. However, that's outside the scope of this tutorial.

Now let's share the new image with others so they can create containers from it.

Step 8 — Pushing Docker Images to a Docker Repository

The next logical step after creating a new image from an existing image is to share it with a select few of your friends, the whole world on Docker Hub, or other Docker registry that you have access to. To push an image to Docker Hub or any other Docker registry, you must have an account there.

This section shows you how to push a Docker image to Docker Hub. To learn how to create your own private Docker registry, check out How To Set Up a Private Docker Registry on Ubuntu 18.04.

To push your image, first log into Docker Hub.

```
1. docker login -u docker-registry-username
2.
```

You'll be prompted to authenticate using your Docker Hub password. If you specified the correct password, authentication should succeed.

Note: If your Docker registry username is different from the local username you used to create the image, you will have to tag your image with your registry username. For the example given in the last step, you would type:

```
1. docker tag sammy/ubuntu-nodejs docker-registry-username/ubuntu-
   nodejs
2.
```

Then you may push your own image using:

```
1. docker push docker-registry-username/docker-image-name
2.
```

To push the `ubuntu-nodejs` image to the **sammy** repository, the command would be:

```
1. docker push sammy/ubuntu-nodejs
2.
```

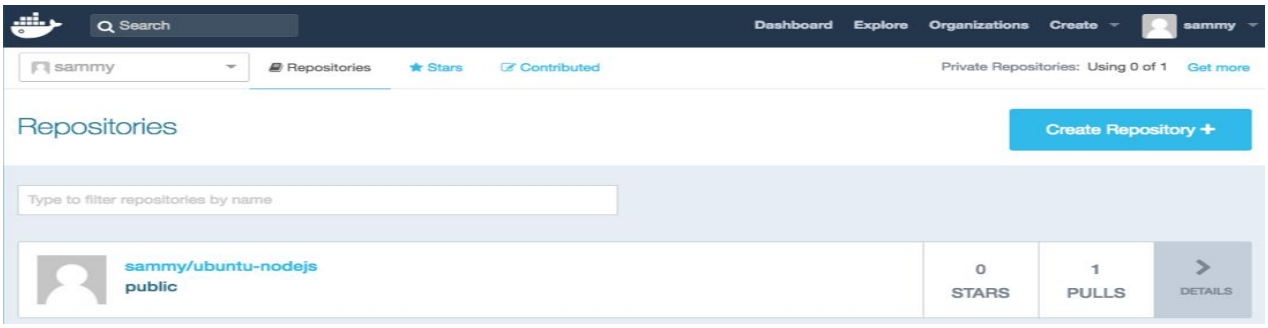
The process may take some time to complete as it uploads the images, but when completed, the output will look like this:

```
Output
The push refers to a repository [docker.io/sammy/ubuntu-nodejs]
e3fbbfb44187: Pushed
```

```
5f70bf18a086: Pushed
a3b5c80a4eba: Pushed
7f18b442972b: Pushed
3ce512daaf78: Pushed
7aae4540b42d: Pushed

...
```

After pushing an image to a registry, it should be listed on your account's dashboard, like that show in the image below.



If a push attempt results in an error of this sort, then you likely did not log in:

```
Output
The push refers to a repository [docker.io/sammy/ubuntu-nodejs]
e3fbbfb44187: Preparing
5f70bf18a086: Preparing
a3b5c80a4eba: Preparing
7f18b442972b: Preparing
3ce512daaf78: Preparing
7aae4540b42d: Waiting
unauthorized: authentication required
```

Log in with `docker login` and repeat the push attempt. Then verify that it exists on your Docker Hub repository page.
You can now use `docker pull sammy/ubuntu-nodejs` to pull the image to a new machine and use it to run a new container.

Conclusion

In this tutorial you installed Docker, worked with images and containers, and pushed a modified image to Docker Hub. Now that you know the basics, explore the other Docker tutorials in the DigitalOcean Community.

Appendix C

Denoising example

To reduce noise in SAR images, Denoising step is provided. For this step, a number of spatial filtering methods, e.g. filtering methods embedded in SNAP, AI based filtering methods and multi-temporal speckle filtering methods, is included. For instance, the multi-temporal speckle filtering is proposed by [5]. As reviewed by [1], in case of applying this filter on SAR amplitude, for a pixel at position (rg, az), with the amplitude value $A^k(rg, az)$, at the kth acquisition, its resultant amplitude value after the filter, denoted by $\tilde{A}^k(rg, az)$, can be expressed as,

$$\tilde{A}^k(rg, az) = [[A^k(rg, az)]_{filter\_space}]_{filter\_time} = S^k(rg, az)T(rg, az), \tag{C.1}$$

where the spatial filter output is $S^k(rg, az) = [A^k(rg, az)]_{filter\_space}$. The Boxcar filter, Lee-sigma filter [3] and IDAN (Intensity Driven Adaptive Neighborhood) filter [6] are examples of the spatial filter methods. The subscript $filter\_space$ and $filter\_time$ indicate the spatial and temporal filter process respectively. The temporal filter output $T(rg, az)$ is defined as

$$T(rg, az) = \frac{1}{m+1} \sum_{i=1}^{m+1} \frac{A^i(rg, az)}{[A^i(rg, az)]_{filter\_space}}, \tag{C.2}$$

where $m + 1$ is the total number of SAR image acquisitions, $A^i(rg, az)$ represents SAR amplitude at position (rg, az) of ith acquisition, $i \in [1, m + 1]$. Having the spatial filter

output as a divisor in Eq. (C.2), we can normalize the amplitude values. The multi-temporal speckle filtering method is scripted with python, namely 'speckle_filt.py'.

Bibliography

[1] Ling Chang et al. "Extraction and Analysis of Radar Scatterer Attributes for PAZ SAR by Combining Time Series InSAR, PolSAR, and Land Use Measurements". In: *Remote Sensing* 15.6 (2023), p. 1571.

[2] Anurag Kulshrestha, Ling Chang, and Alfred Stein. "Radarcoding Reference Data for SAR Training Data Creation in Radar Coordinates". In: *IEEE Geoscience and Remote Sensing Letters* (2023). under review.

[3] Jong-Sen Lee et al. "Improved sigma filter for speckle filtering of SAR imagery". In: *IEEE Transactions on Geoscience and Remote Sensing* 47.1 (2008), pp. 202–213.

[4] Chang Ling and et al. "AlignSAR: An open-source package of SAR benchmark dataset creation for machine learning applications". In: *2024 IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2024, submitted*. 2024, pp. 1–5.

[5] Shaun Quegan and Jiong Jiong Yu. "Filtering of multichannel SAR images". In: *IEEE Transactions on Geoscience and Remote Sensing* 39.11 (2001), pp. 2373–2379. ISSN: 01962892. DOI: 10.1109/36.964973.

[6] Gabriel Vasile et al. "Intensity-driven adaptive-neighborhood technique for polarimetric and interferometric SAR parameters estimation". In: *IEEE Transactions on Geoscience and Remote Sensing* 44.6 (2006), pp. 1609–1621.

[7] Chen Yu et al. "Generic Atmospheric Correction Model for Interferometric Synthetic Aperture Radar Observations". In: *Journal of Geophysical Research: Solid Earth* 123.10 (2018), pp. 9202–9222. DOI: https://doi.org/10.1029/2017JB015305.

eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2017JB015305.

URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2017JB015305.