SURGE Project Report

Estimating optimal location of Virtual Stations using Satellite Radar Altimetry and QGIS.

Anandita Kaushal
Surge Intern (S230039)
NIT Hamirpur | 20bce034@nith.ac.in

Under the guidance of

Dr. Balaji Devaraju
Assistant Professor
Department of Civil Engineering
IIT Kanpur, UP

12th July, 2023

Table of Contents

Abstract

Satellite altimetry is among the widely used ocean remote sensing techniques. It measures sea surface topography and offers valuable insights on Earth's spatial & temporal changes, river dynamics and soil's moisture contents. As a means of estimating these changes, we need to first define virtual stations, these being the points of intersection between a water body surface and a satellite ground track.

This project report aims to precisely locate optimal virtual stations, state the procedure on how to get there, then briefly discuss the results we get.

We utilise two primary datasets: Shuttle Radar Topography Mission's digital elevation model (SRTM-DEM) data obtained from Google Earth Engine and the United States Geological Survey (USGS) site, along with ground track waveform data acquired from Copernicus Open Access Hub. The DEM dataset is initially loaded, and its elevation band information is extracted and organised into a data frame. Various plotting techniques are employed to visualise the dataset, and the gradient of a smoothed elevation matrix is computed. Potential channel locations are identified, and their geographical representation is visualised using a folium map. Subsequently, the estimation of river width as a function of each channel location is performed. The dataset is further utilised to compute flow accumulation using QGIS software for the selected river basin, which serves as the area of interest. The drainage network is extracted by applying a threshold value to the resulting flow accumulation map. Finally, the intersection between this extracted drainage network and the ground track data yields the precise coordinates of the corresponding virtual stations.

Accurate virtual station estimation is important because it provides crucial reference points for understanding Earth's topography, bridging spatial data gaps, and enhancing radar altimetry data processing. These locations supplement models and algorithms, improving accuracy and enabling comprehensive analysis of climate patterns, sea level rise, and hydrological processes. Moreover, virtual station data informs future satellite mission planning and supports scientific research and environmental management.

Keywords: Satellite altimetry, digital elevation model (DEM), google earth engine (GEE), quantum geographic information system (QGIS) software.

Acknowledgement

I would like to express my sincere gratitude to all those who have contributed to the successful completion of this project. Their support, guidance, and encouragement have been invaluable throughout this journey and made this endeavour possible in the first place.

I want to start by saying how grateful I am to my supervisor, Dr. Balaji Devaraju, for his counsel and expertise. He has been instrumental in helping me understand concepts better and providing valuable feedback that has significantly improved this work.

I would extend my appreciation to Abhilasha mam, a PHD scholar and Shubhi Kant mam, an MTech student who both study here at IIT Kanpur, for always lending a helping hand whenever I needed it or had a doubt or question.

I am also obliged to IIT Kanpur institute for selecting me through their Surge Program and giving me this opportunity to learn and explore.

Lastly, I would love to mention my parents, my brother and all my friends here at IIT Kanpur who stayed with me throughout and kept my spirits high whenever things got rough. Their understanding, patience, and encouragement have been the pillars that propelled me forward.

And in closing, I would like to acknowledge the trailblazers and visionaries whose groundbreaking work has paved the way for this study. Their tireless pursuit of knowledge and passion for discovery continue to inspire and fuel our collective progress.

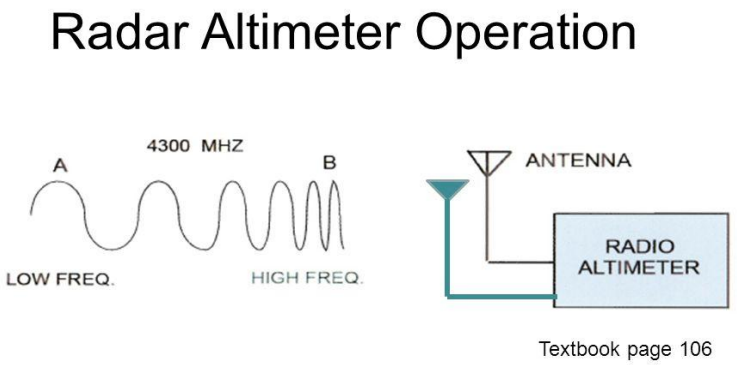
Thank you all for your contributions.

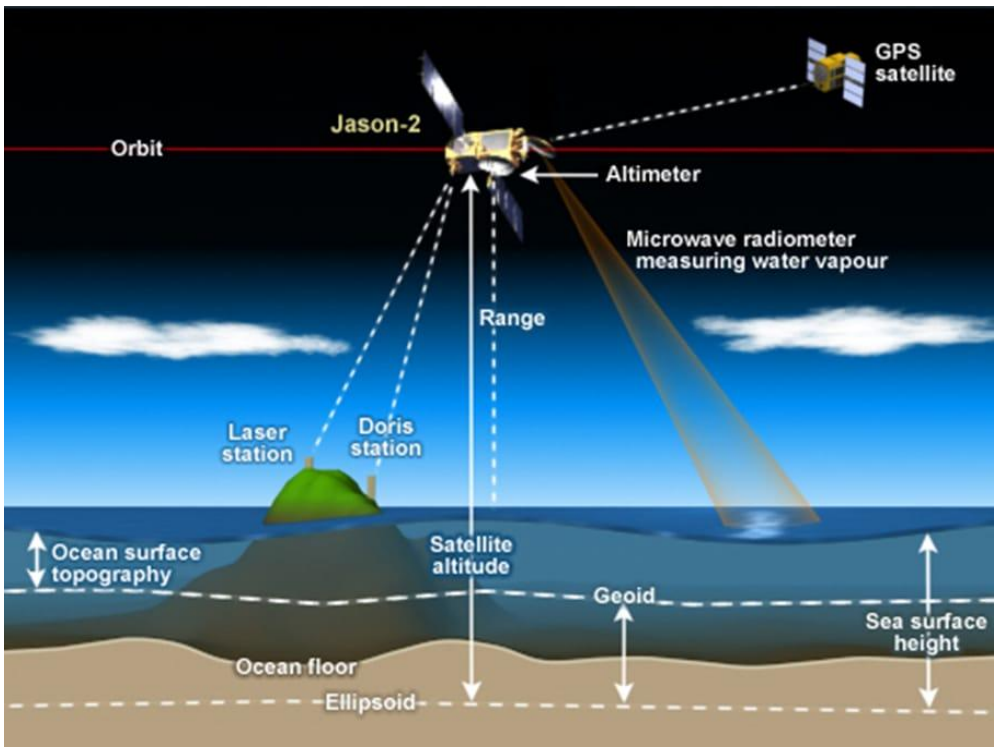
Principle of Satellite Radar Altimetry

The altimeter emits a radar pulse towards the nadir direction (directly below the observer) and measures the time it takes back for the pulse to reflect off the Earth's surface and return to the altimeter.

By knowing the speed of wave propagation through the traversed medium, the travel time enables the calculation of the distance (R) between the altimeter and the reflecting surface. Given the satellite's orbit (and consequently its height, H, relative to an ellipsoid), the height of the reflective surface (h) can be derived as h = H - R.

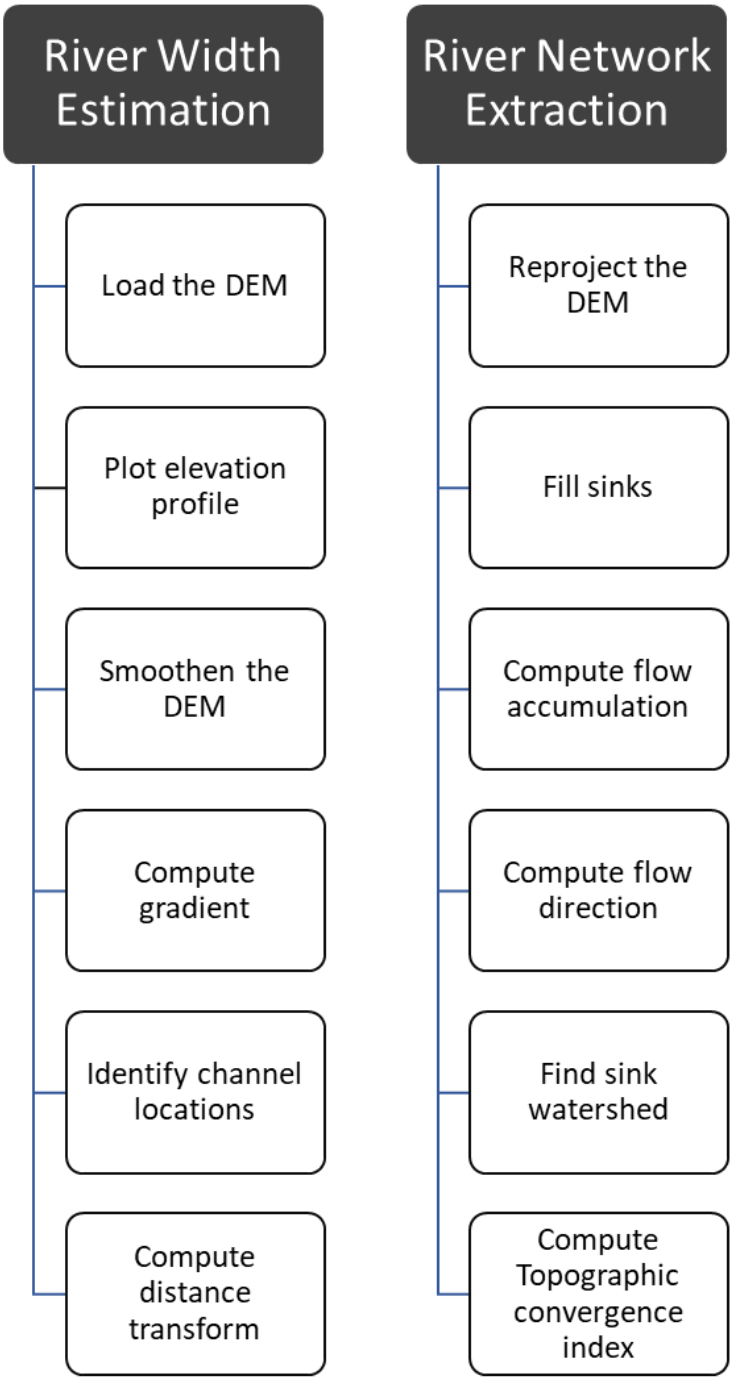
However, practical measurements require various corrections to account for atmospheric disturbances, ionosphere effects, and solid and liquid earth tides.





Methodology

Let's break down the process with the help of a flowchart:



River Width Estimation:

1. Import the necessary libraries, packages and functions.
   Some of the main ones being used are:

```
import numpy as np
import pandas as pd

import scipy.ndimage
import scipy.signal
import matplotlib.pyplot as plt

from scipy.ndimage import sobel
from skimage.feature import peak_local_max
from scipy.ndimage import distance_transform_edt
```

2. Authenticate and initialise Google Earth Engine to import its functionalities.

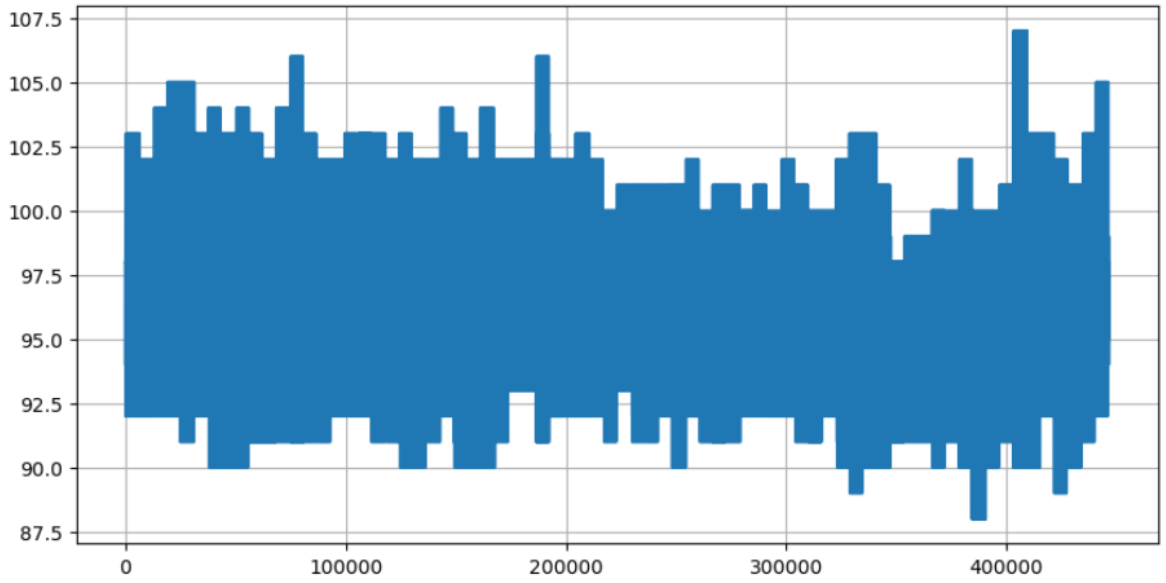
```
import ee
ee.Authenticate()
ee.Initialize()
```

3. Dataset 1: Google Earth Engine's DEM: 'CGIAR/SRTM90_V4'

   3.1. Get the Earth Engine's elevation band information in a pandas dataframe.

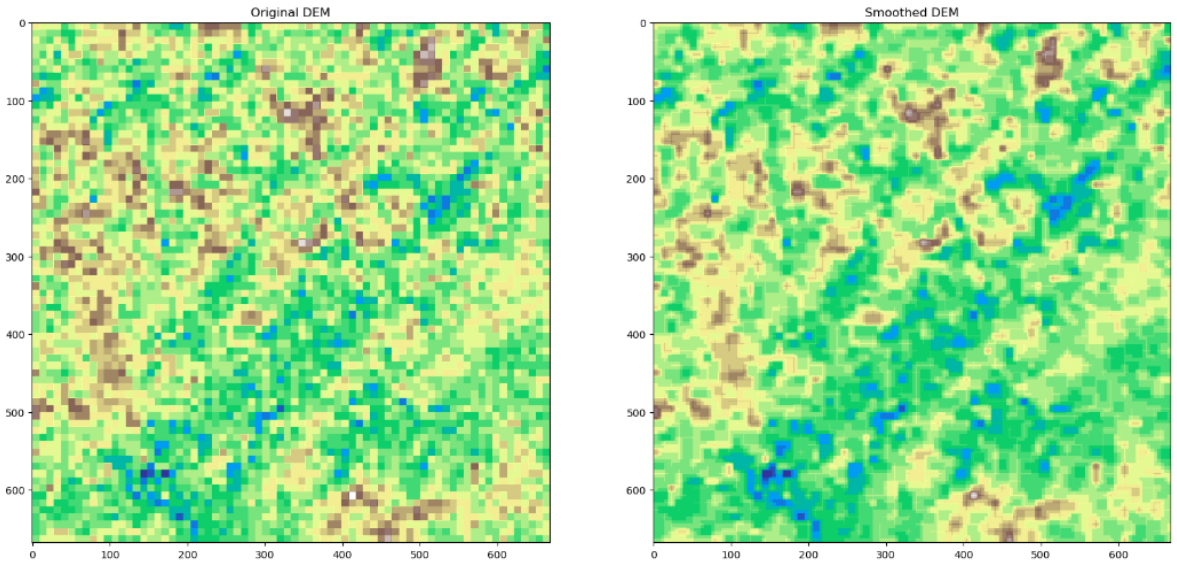
	Key	Value
0	type	Image
1	bands	[{'id': 'elevation', 'data_type': {'type': 'PixelType', 'precision': 'int', 'min': -32768, 'max': 32767}, 'dimensions': [432000, 144000], 'crs': 'EPSG:4326', 'crs_transform': [0.000833333333333, 0, -180, 0, -0.000833333333333, 60]}]
2	id	CGIAR/SRTM90_V4
3	version	1641990053291277
4	properties_system:visualization_0_min	-100.0
5	properties_type_name	Image
6	properties_keywords	[cgiar, dem, elevation, geophysical, srtm, topography]
7	properties_thumb	https://mw1.google.com/ges/dd/images/SRTM90_V4_thumb.png
8	properties_description	The Shuttle Radar Topography Mission (SRTM) digital\nelevation dataset was originally produced to provide consistent,\nhigh-quality elevation data at near global scope. This version\nof the SRTM digital elevation data has been processed to fill data\nvoids, and to facilitate its ease of use. Provider: NASA/CGIAR Bands<table class="eecat"><tr><th scope="col">Name</th><th scope="col">Description</th></tr><tr><td>elevation</td><td> Elevation </td></tr></table> Terms of Use DISTRIBUTION. Users are prohibited from any commercial, non-free resale, or\nredistribution without explicit written permission from CIAT. Users should\nacknowledge CIAT as the source used in the creation of any reports,\npublications, new datasets, derived products, or services resulting from the\nuse of this dataset. CIAT also request reprints of any publications and\nnotification of any redistributing efforts. For commercial access to\nthe data, send requests to Andy Jarvis. NO WARRANTY OR LIABILITY. CIAT provides these data without any warranty of\nany kind whatsoever, either express or implied, including warranties of\nmerchantability and fitness for a particular purpose. CIAT shall not be\nliable for incidental, consequential, or special damages arising out of\nthe use of any data. ACKNOWLEDGMENT AND CITATION. Any users are kindly asked to cite this data\nin any published material produced using this data, and if possible link\nweb pages to the CIAT-CSI SRTM website. Suggested citation(s) Jarvis, A., H.I. Reuter, A. Nelson, E. Guevara. 2008. Hole-filled\nSRTM for the globe Version 4, available from the CGIAR-CSI SRTM\n90m Database: https://srtm.csi.cgiar.org. <style>\n table.eecat {\n border: 1px solid black;\n border-collapse: collapse;\n font-size: 13px;\n }\n table.eecat td, tr, th {\n text-align: left; vertical-align: top;\n border: 1px solid gray; padding: 3px;\n }\n td.nobreak { white-space: nowrap; }\n</style>
9	properties_source_tags	[cgiar]
10	properties_visualization_0_max	8000.0
11	properties_title	SRTM Digital Elevation Data Version 4
12	properties_product_tags	[srtm, elevation, topography, dem, geophysical]

   3.2. Plot the Elevation Profile of the Earth Engine's elevation dataset and get the statistics of the DEM. Define the bounding box coordinates [minX, minY, maxX, maxY] for Ganga River in Ayodhya, India as: bounding_box = [82.12, 25.72, 82.18, 25.78].

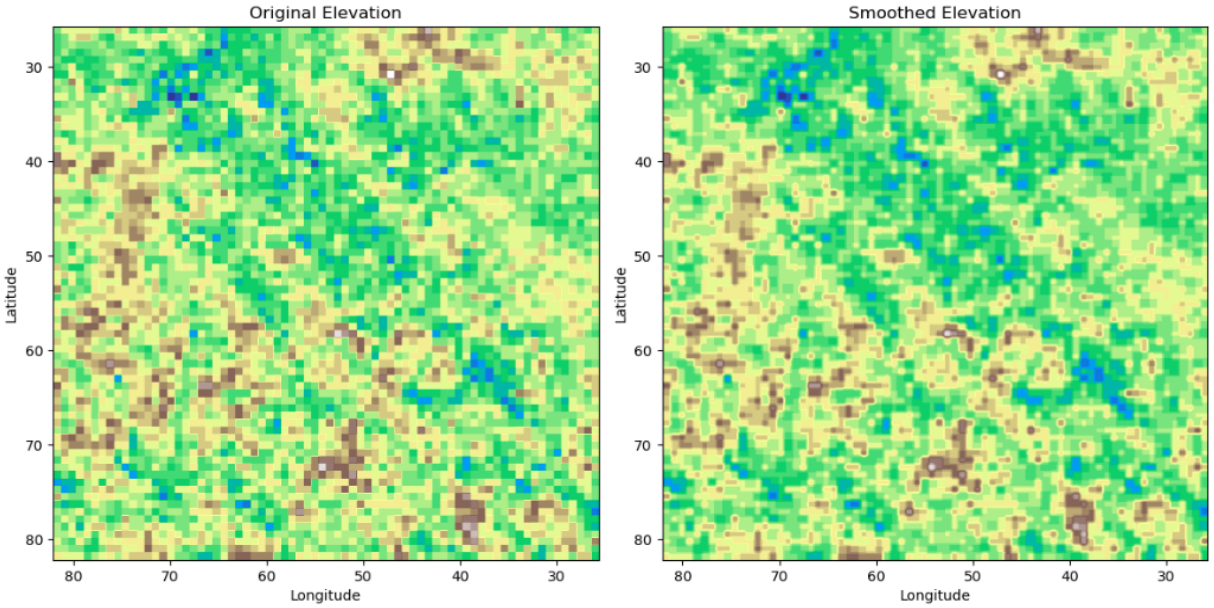



```
Minimum Elevation: 88
Maximum Elevation: 107
Mean Elevation: 96.17562479830758
Median Elevation: 96.0
Standard Deviation: 2.3676893157611203
```

3.3. Apply Gaussian filter to smoothen the DEM and visualise the results. First, reshape the elevation array to a 2D grid, then smoothen the DEM as: smoothed_elevation = scipy.ndimage.gaussian_filter(elevation_grid, sigma=1).

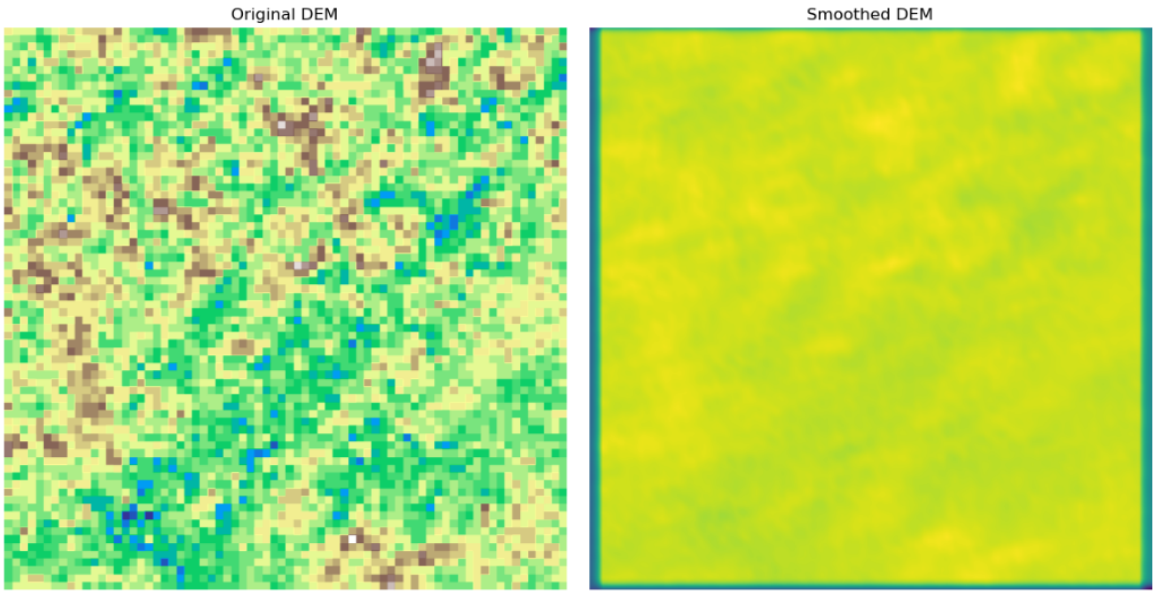


3.4. Apply Mean filter to smoothen the DEM and visualise the results. First, reshape the elevation array into a matrix, then smoothen the DEM as: smoothed_elevation = uniform_filter(elevation_matrix, size=2*radius+1).



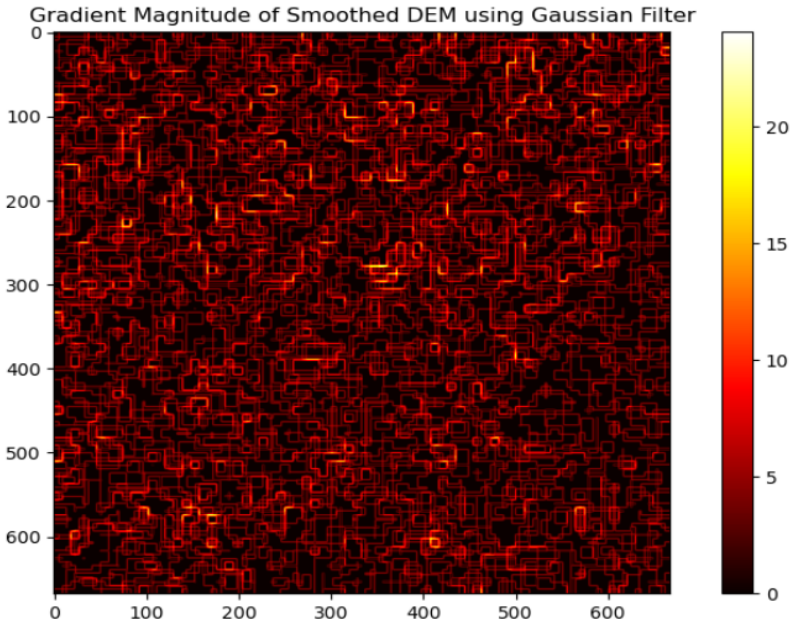
3.4. Apply Bilateral filter to smoothen the DEM and visualise the results. First, reshape the elevation array into a matrix, convert the elevation matrix to float between 0 and 1, then smoothen the DEM as:

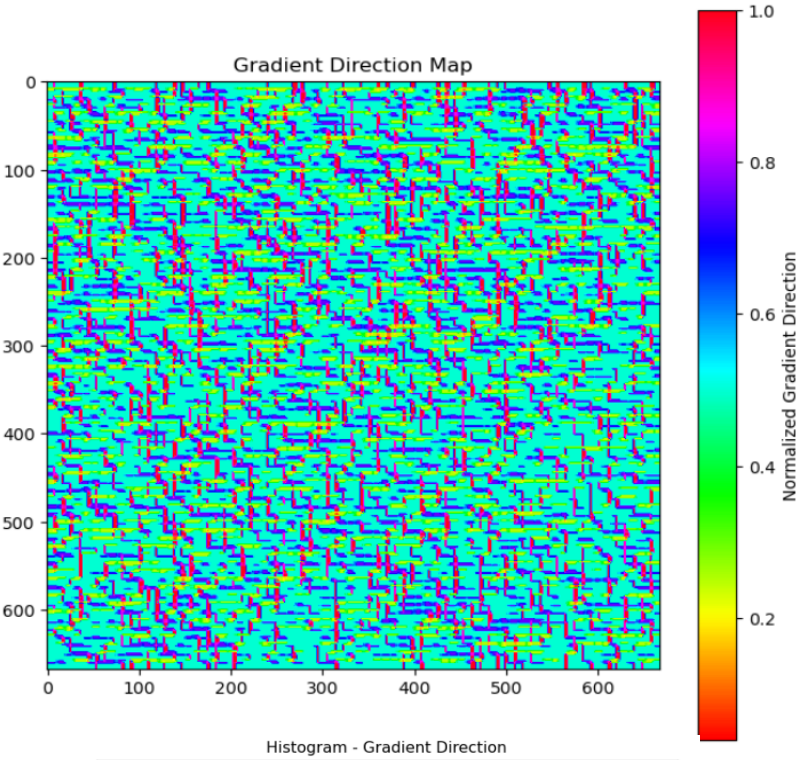
```
sigma_color = 0.1                    # Controls the range similarity
sigma_spatial = 5                    # Controls the spatial smoothing
smoothed_elevation = denoise_bilateral (elevation_float,
sigma_color=sigma_color, sigma_spatial=sigma_spatial)
```

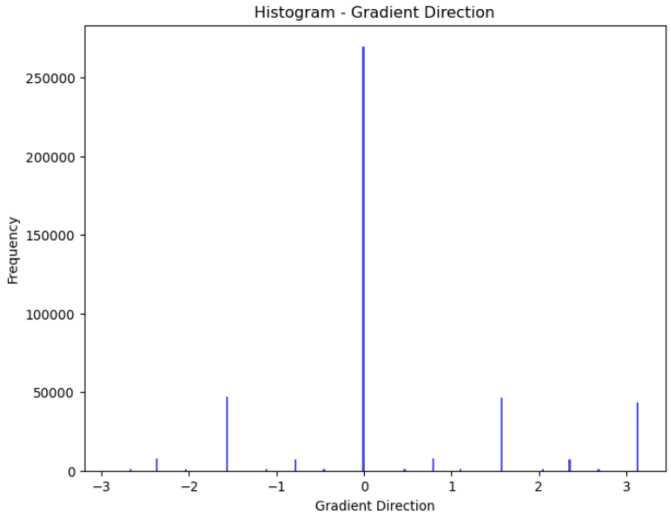


3.5. Compute and visualise the gradient magnitude, gradient direction and their respective histograms of the DEM using:

```
gradient_magnitude = np.sqrt(dx**2 + dy**2)
gradient_direction = np.arctan2(dy, dx)
```


Gradient Magnitude of Smoothed DEM using Gaussian Filter
0
100
200
300
400
500
600
0
100
200
300
400
500
600
20
15
10
5
0


Gradient Direction Map
0
100
200
300
400
500
600
0
100
200
300
400
500
600
1.0
0.8
0.6
0.4
0.2
Normalized Gradient Direction


Histogram - Gradient Direction
250000
200000
150000
100000
50000
0
Frequency
−3
−2
−1
0
1
2
3
Gradient Direction

4. Dataset 1: USGS SRTM DEM:
"C:/Users/HP/Downloads/n25_e000_1arc_v3.tif"

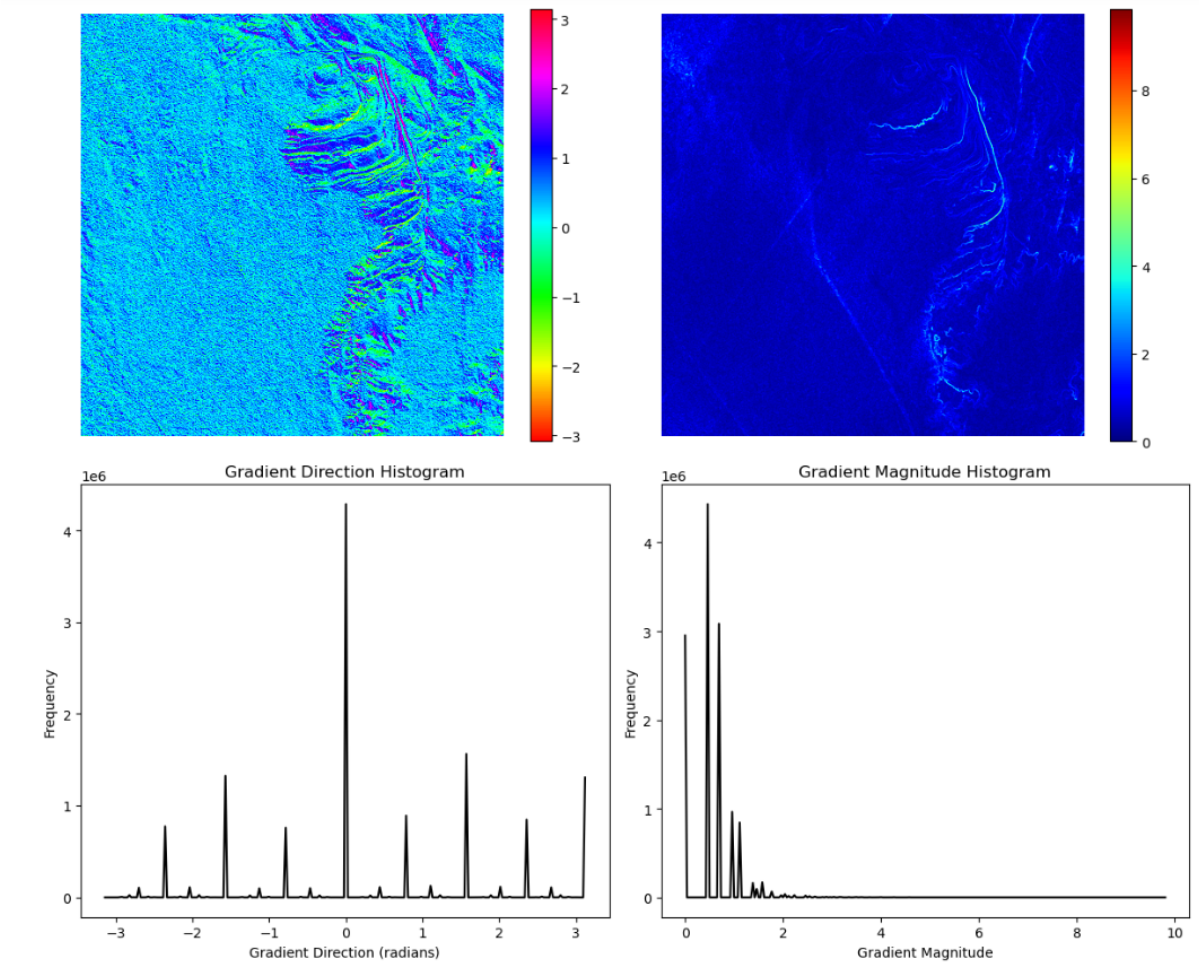
4.1. Get the USGS DEM's elevation band information in a pandas dataframe.

	Key	Value
0	driver	GTiff
1	dtype	int16
2	nodata	-32767.0
3	width	3601
4	height	3601
5	count	1
6	crs	(init)
7	transform	(0.0002777777777777778, 0.0, -0.0001388888888888889, 0.0, -0.0002777777777777778, 26.00013888888889, 0.0, 0.0, 1.0)

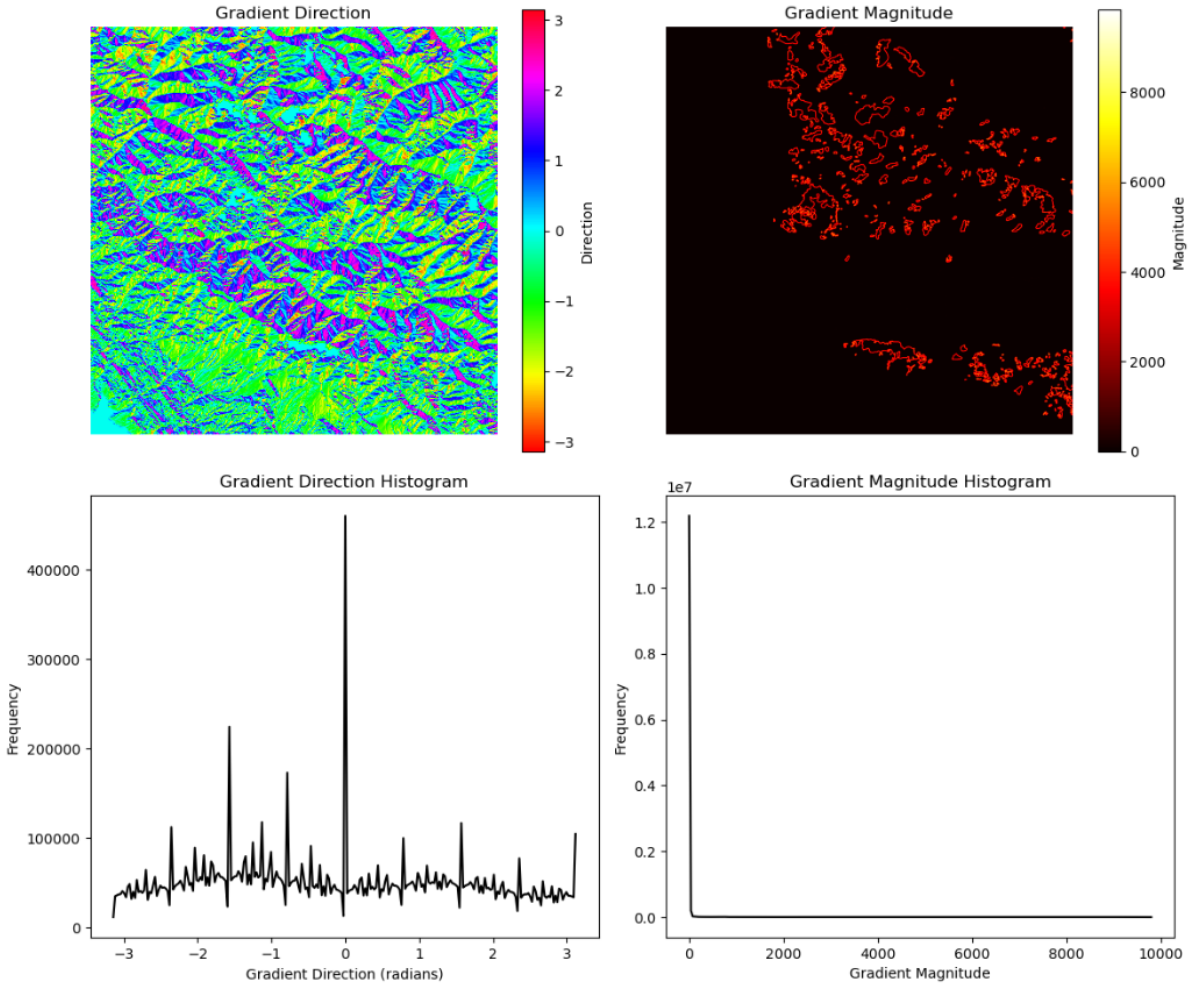
4.2. Compute and visualise the gradient magnitude, gradient direction and their respective histograms of the DEM using:

gradient_magnitude = np.sqrt(dx**2 + dy**2)

gradient_direction = np.arctan2(dy, dx)

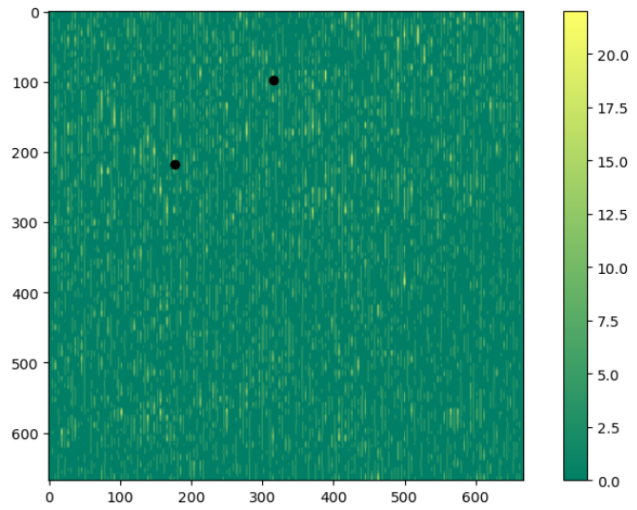


4.3. Compute and visualise the gradient magnitude, gradient direction and their respective histograms of a random hilly terrain DEM for reference.



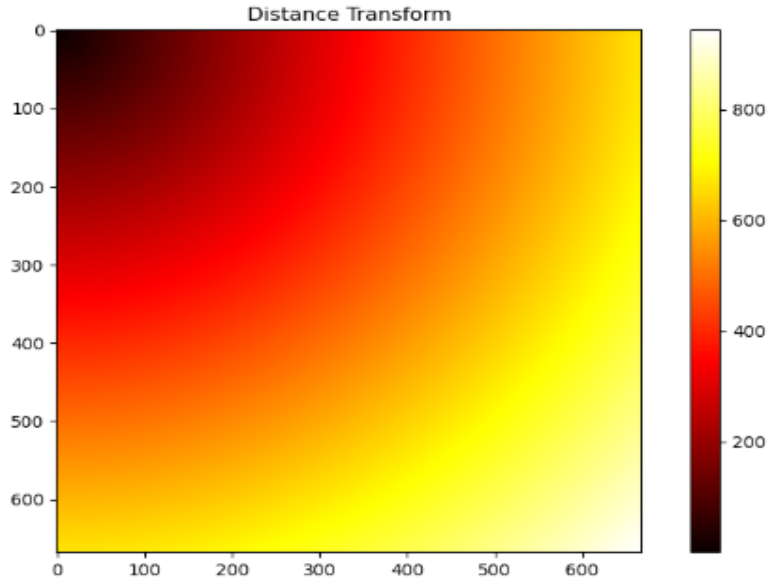
5. Visualise the gradient magnitude image with the identified channel locations marked as black dots using peak_local_max function. peak_local_max function is called with parameters min_distance=10 and num_peaks=2. These parameters control the minimum distance between identified peaks and the maximum number of peaks to identify using:

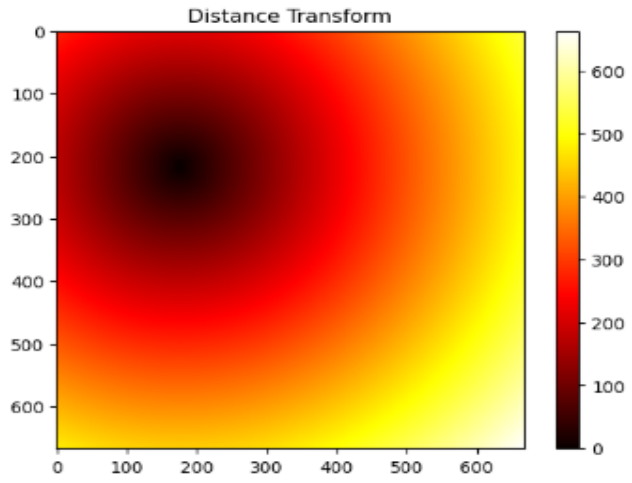
channel_locations = peak_local_max(gradient_magnitude1, min_distance=10, num_peaks=2)



6. Compute and visualise the distance transform for each channel location for Earth Engine's DEM

- Iterate over each channel location (loc) in the channel_locations array.
- Extract the coordinates of the channel location (y and x).
- Create a binary mask by thresholding the gradient magnitude image based on the current channel location. The thresholding operation (gradient_magnitude > gradient_magnitude[y, x]) creates a binary image where True values represent pixels with gradient magnitudes higher than the magnitude at the current channel location.
- Compute the distance transform using the distance_transform_edt function from the scipy.ndimage module. The distance transform calculates the Euclidean distance from each pixel to the nearest False (zero) pixel in the binary mask. In this case, it calculates the distance to the nearest pixel outside the region of interest around the river channel.
- Append the distance transform image (distance_transform) to the distance_transforms list.

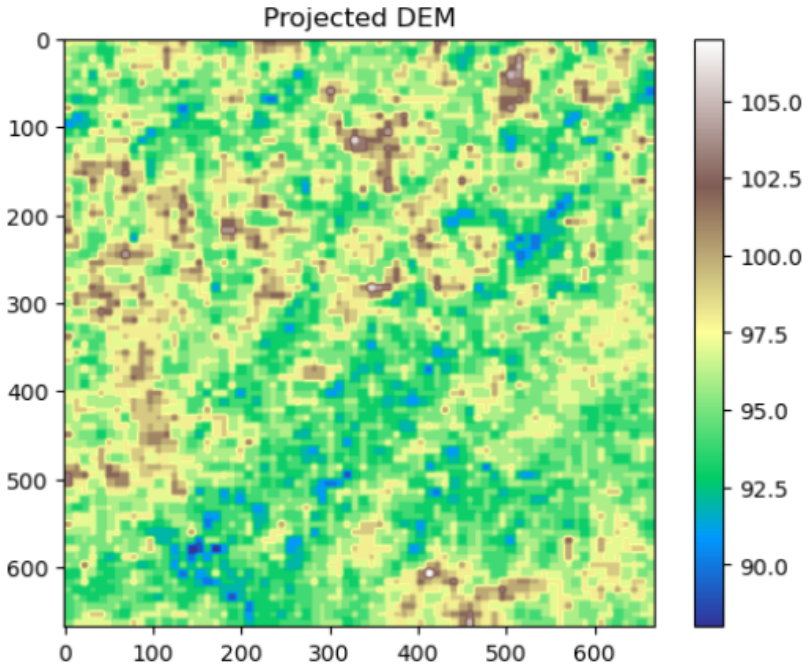




7. Estimate the river width as the mean of maximum distance from each channel location using:
river_width = np.mean([2 * np.max(dt) for dt in distance_transforms])
The estimated width is around 1606.5706427293383 for the Ganga river dataset.

8. Project the smoothened DEM onto the river section plane

- Compute the gradient in x, y directions of the smoothed DEM using the sobel function from the scipy.ndimage module.
- Compute the orientation angle at each pixel by applying the arctan2 function to the gradient_y and gradient_x arrays. This will give you the angle of the steepest ascent/descent at each pixel.
- Compute the perpendicular orientation to the river channel by adding π/2 radians to each orientation value and taking the modulo π. This effectively rotates the orientations by 90 degrees counter-clockwise.
- Determine the pixel coordinates of the cross-section location. We assume the cross-section is in the middle of the river width and the middle row of the grid.
- Compute the distances of each pixel from the cross-section location along the perpendicular orientation using the distance_transform_edt function from scipy.ndimage. This function calculates the Euclidean distance transform.
- Identify the pixels within the river section by creating a mask based on the distances and the river width. Pixels with distances less than or equal to the river width are considered part of the river section.
- Convert the data type of the smoothed DEM to floating-point using the astype method. This is done to ensure compatibility for assigning NaN values in the next step.
- Project the DEM onto the river section plane by assigning NaN values to pixels outside the river section.

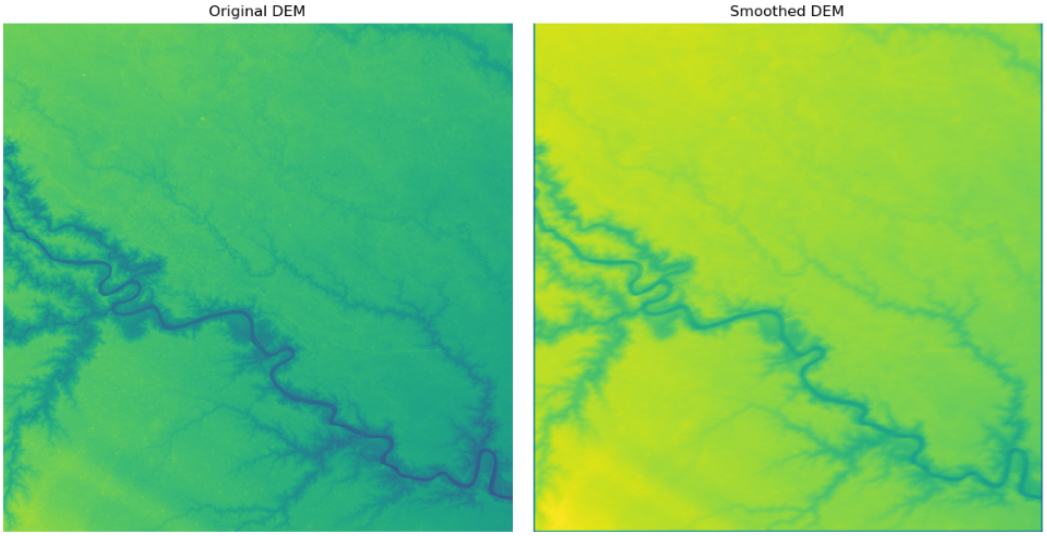


9. Compute and print the radar nadir coordinates by filtering the Sentinel-1 GRD collection based on the defined geometry (bounding box) of the Ganga region and date range using 'COPERNICUS/S1_GRD' dataset.

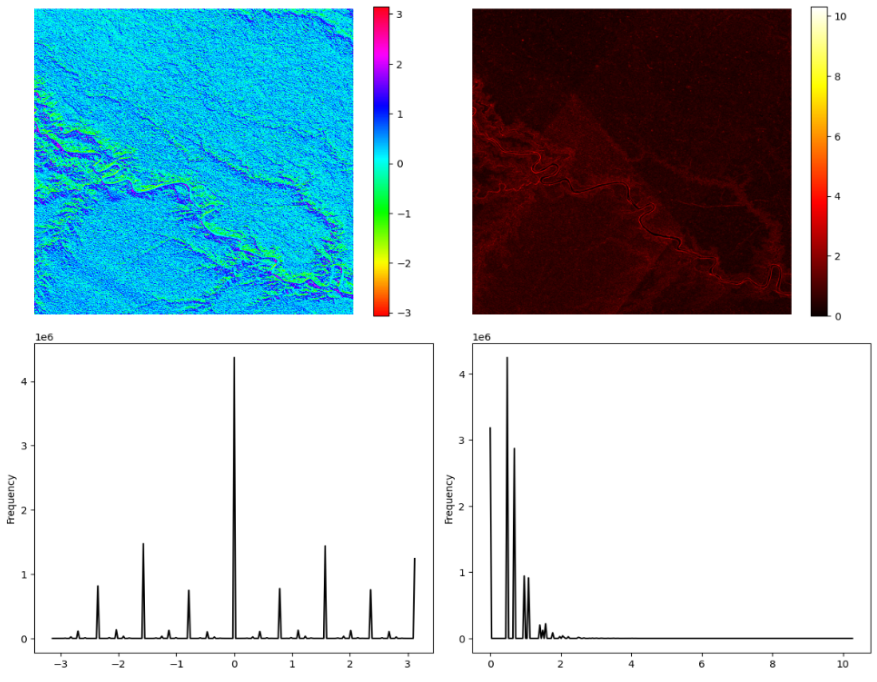
```
Nadir Locations (Longitude, Latitude):
82.12 25.72
82.18 25.72
82.18 25.78
82.12 25.78
82.12 25.72
```

10. Taking the second dataset for river Yamuna near Kalpi region, UP and re-doing all of the above steps using the same algorithms.

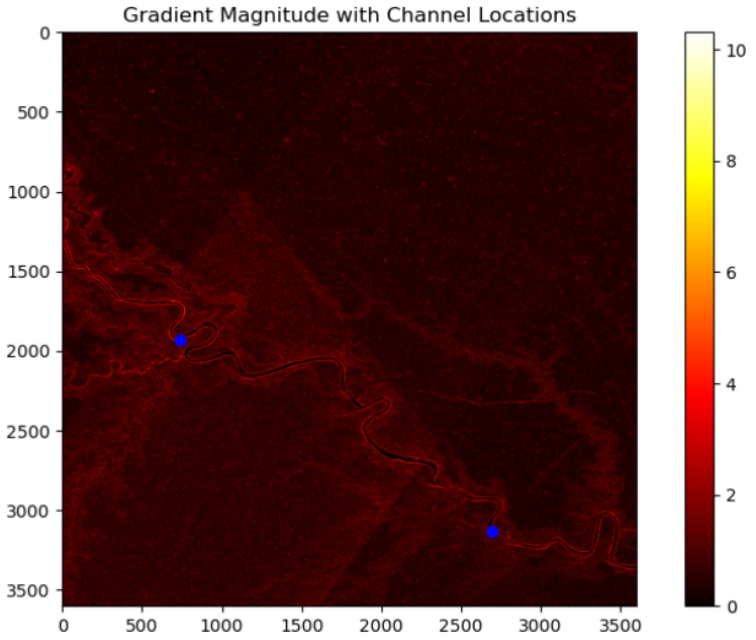
10.1. Smoothen the DEM using a Bilateral filter.



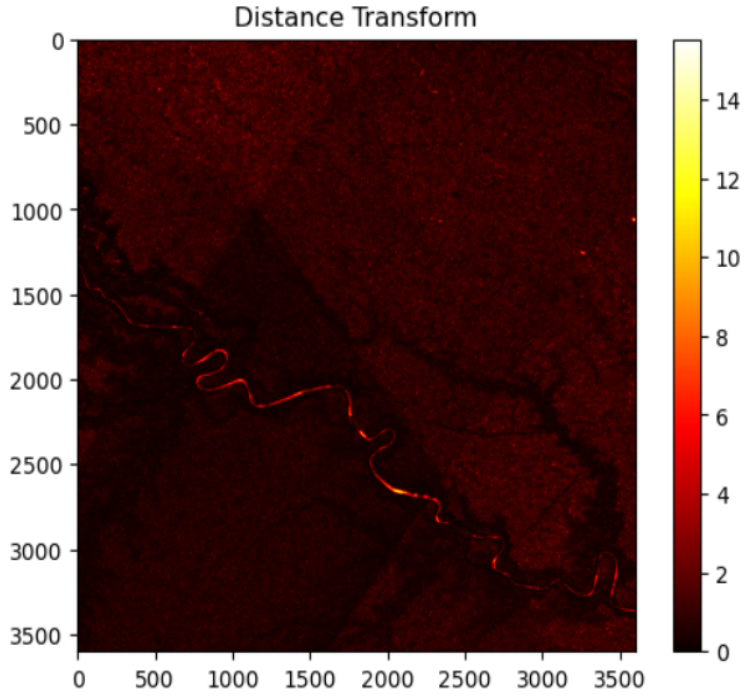
10.2. Compute and visualise the gradient magnitude, gradient direction and the respective histograms of the DEM.



10.3. Visualise the gradient magnitude image with the identified channel locations marked as blue dots using peak_local_max function.



10.4. Compute and visualise the distance transform for each channel location.



10.5. Estimate the river width of Yamuna river as the mean of maximum distance from each channel location.

```
# Estimate river width
river_width = np.mean([2 * np.max(dt) for dt in distance_transforms])

# Print the estimated river width
print("Estimated river width:", river_width)
```

```
Estimated river width: 31.04834939252005
```

River Network Estimation:

We use QGIS (quantum geographic information system) software to extract the river network from the DEM. QGIS is a spatial analysis software used to work with raster and vector data, allowing users to analyse and edit spatial information, in addition to composing and exporting graphical maps.

Here I will highlight the general steps using the Yamuna river as my region of interest. Same steps will work for the Ganga river.

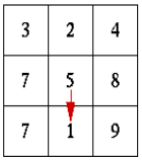
1. Specifications:
   Function used: GRASS r.terraflow, it performs flow computation for massive grids.

DESCRIPTION

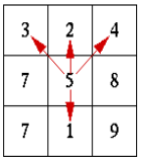
r.terraflow takes as input a raster digital elevation model (DEM) and computes the flow direction raster and the flow accumulation raster, as well as the flooded elevation raster, sink-watershed raster (partition into watersheds around sinks) and TCI (topographic convergence index) raster maps.

r.terraflow computes these rasters using well-known approaches, with the difference that its emphasis is on the computational complexity of the algorithms, rather than on modeling realistic flow. *r.terraflow* emerged from the necessity of having scalable software able to process efficiently very large terrains. It is based on theoretically optimal algorithms developed in the framework of I/O-efficient algorithms. *r.terraflow* was designed and optimized especially for massive grids and is able to process terrains which were impractical with similar functions existing in other GIS systems.

Flow directions are computed using either the MFD (Multiple Flow Direction) model or the SFD (Single Flow Direction, or D8) model, illustrated below. Both methods compute downslope flow directions by inspecting the 3-by-3 window around the current cell. The SFD method assigns a unique flow direction towards the steepest downslope neighbor. The MFD method assigns multiple flow directions towards all downslope neighbors.



Flow direction to steepest downslope neighbor (SFD).



Flow direction to all downslope neighbors (MFD).

The SFD and the MFD method cannot compute flow directions for cells which have the same height as all their neighbors (flat areas) or cells which do not have downslope neighbors (one-cell pits).

- On plateaus (flat areas that spill out) *r.terraflow* routes flow so that globally the flow goes towards the spill cells of the plateaus.
- On sinks (flat areas that do not spill out, including one-cell pits) *r.terraflow* assigns flow by flooding the terrain until all the sinks are filled and assigning flow directions on the filled terrain.

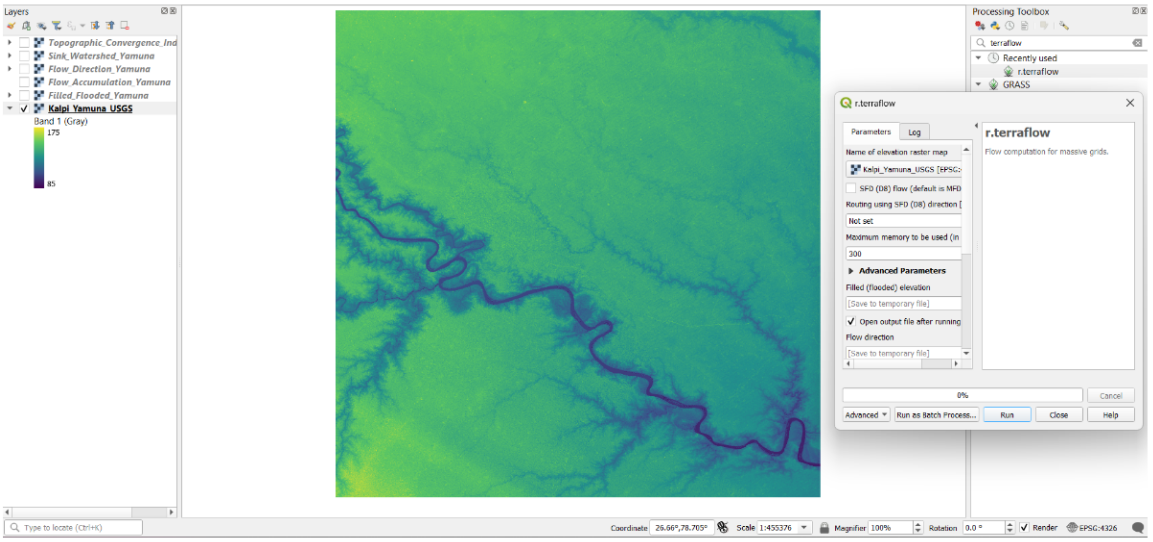
In order to flood the terrain, *r.terraflow* identifies all sinks and partitions the terrain into sink-watersheds (a sink-watershed contains all the cells that flow into that sink), builds a graph representing the adjacency information of the sink-watersheds, and uses this sink-watershed graph to merge watersheds into each other along their lowest common boundary until all watersheds have a flow path outside the terrain. Flooding produces a sink-less terrain in which every cell has a downslope flow path leading outside the terrain and therefore every cell in the terrain can be assigned SFD/MFD flow directions as above.

Once flow directions are computed for every cell in the terrain, *r.terraflow* computes flow accumulation by routing water using the flow directions and keeping track of how much water flows through each cell.

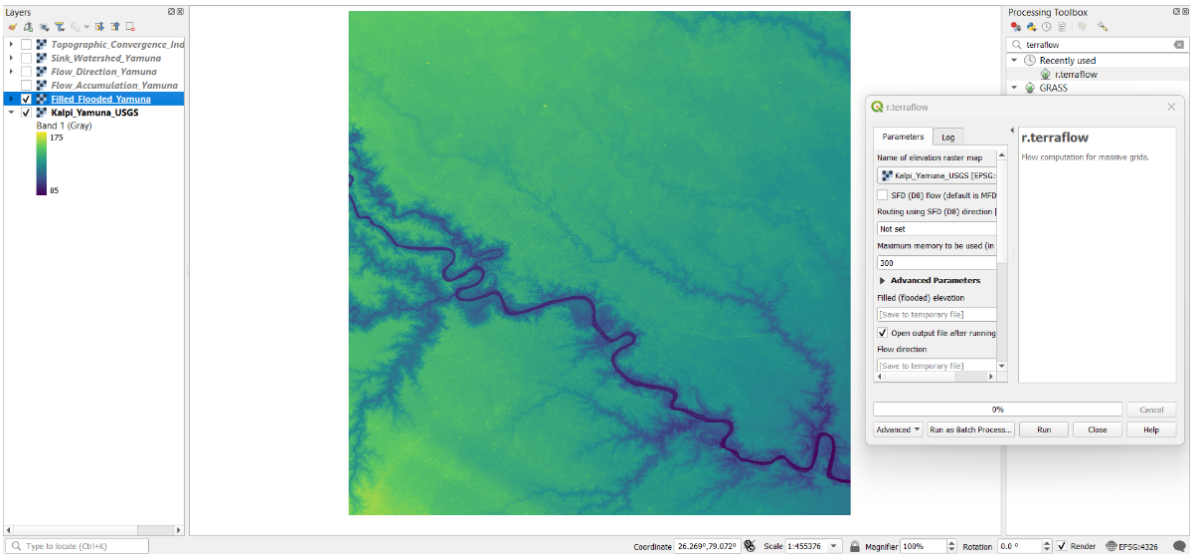
If flow accumulation of a cell is larger than the value given by the **d8cut** option, then the flow of this cell is routed to its neighbors using the SFD (D8) model. This option affects only the flow accumulation raster and is meaningful only for MFD flow (i.e. if the **-s** flag is not used); If this option is used for SFD flow it is ignored. The default value of **d8cut** is *infinity*.

r.terraflow also computes the **tci** raster (topographic convergence index, defined as the logarithm of the ratio of flow accumulation and local slope).

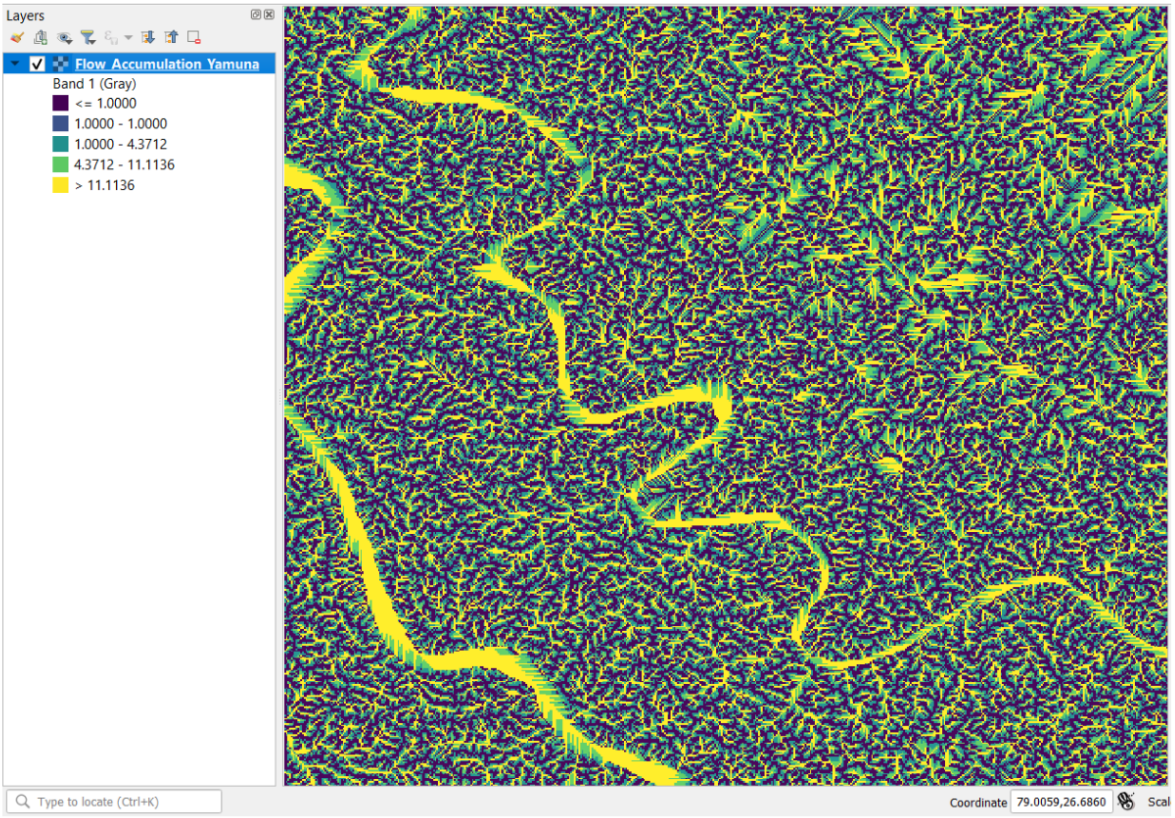
2. Open the QGIS software (version: 3.22.14), load and visualise the DEM. Style the layer to give a single band pseudocolor to its elevation band.¶



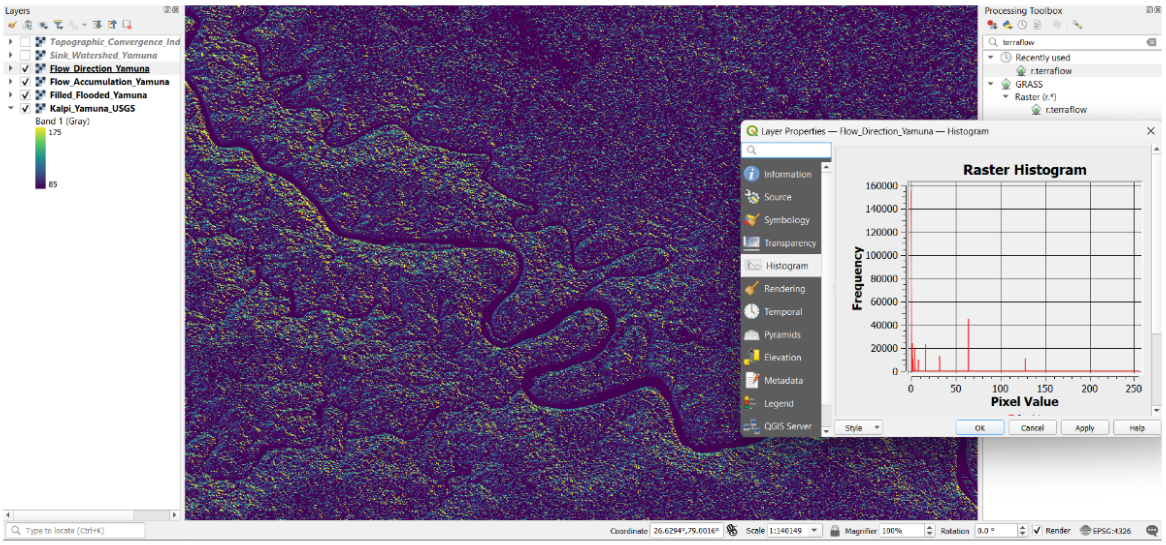
3. Fill and flood the DEM. You may reproject the DEM to give it warp Reprojection using Nearest Neighbour resampling method.



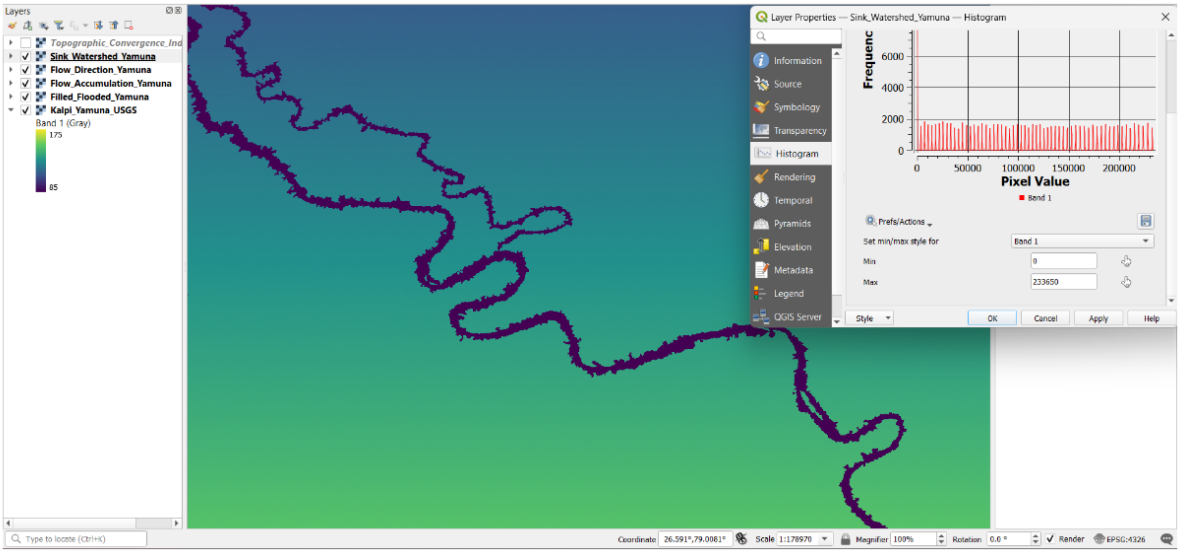
4. Compute Flow Accumulation of the reprojected elevation layer. The result of Flow Accumulation is a raster of accumulated flow to each cell, as determined by accumulating the weight for all cells that flow into each downslope cell.
    - We will use the MFD modelling algorithm (may also use D8).



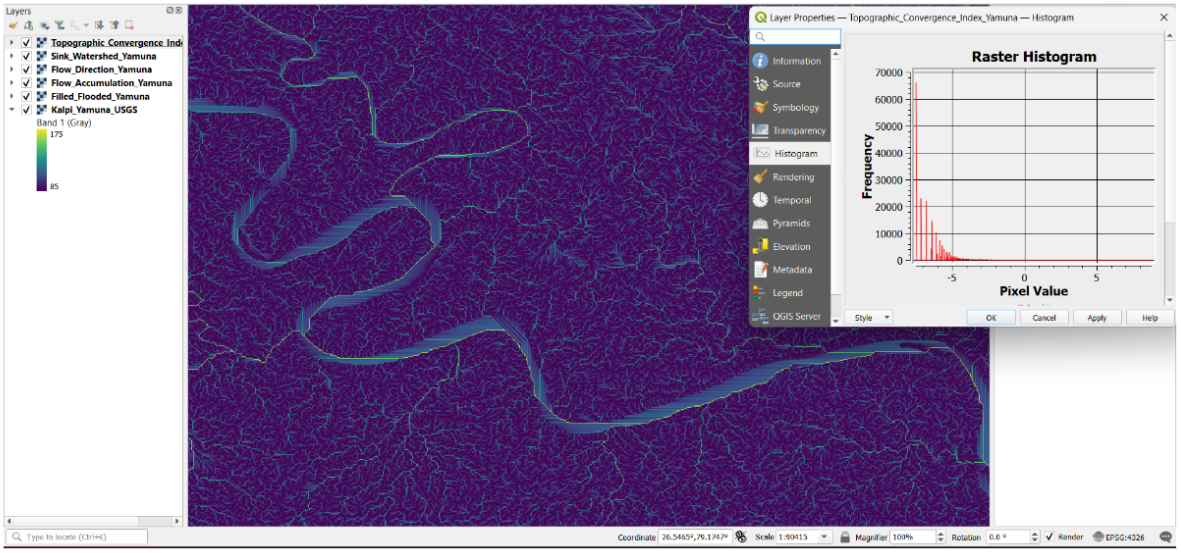
5. Compute Flow Direction of the reprojected elevation layer also showing the raster histogram.



6. Compute Sink Watershed of the reprojected elevation layer also showing the raster histogram. A sink is a cell or set of spatially connected cells whose flow direction cannot be assigned one of the eight valid values in a flow direction raster.

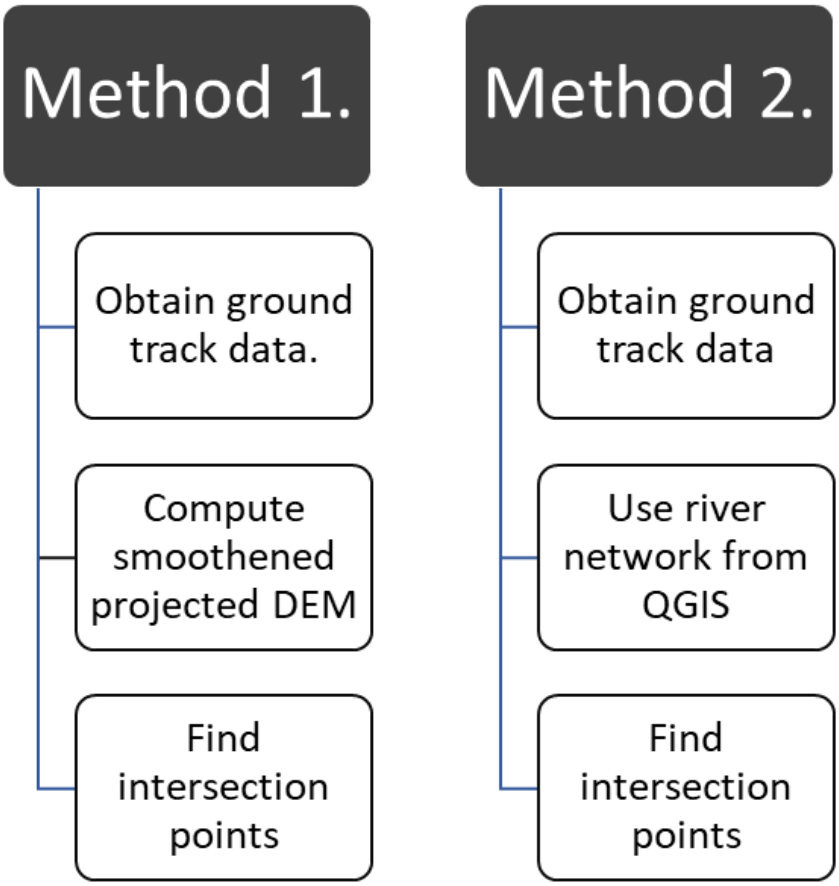


7. Compute Topographic Convergence Index of the reprojected elevation layer also showing the raster histogram. The index is obtained by averaging the bias of the slope directions of the adjacent cells from the direction of the central cell and subtracting 90 degrees. The possible values of the index range from -90° to +90°.



Coming to the final part of the procedure, we find the intersection points aka the crossings where our potential virtual stations will lie. There are 2 methods that are used to find these points.

Here's a flowchart briefly mentioning the steps:

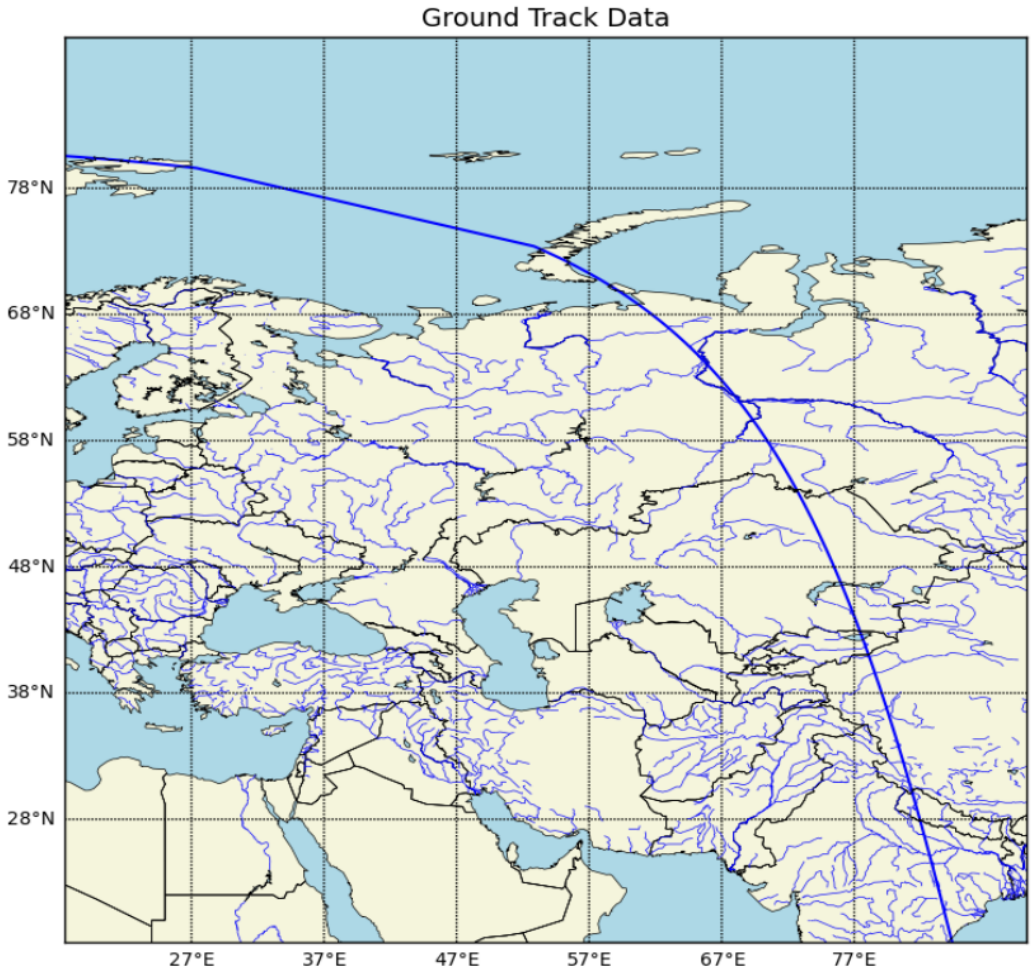


Method 1:

1. Obtaining ground track data from the Copernicus Open Access Hub site. It will be in netCDF file format and we will work with its KU band.

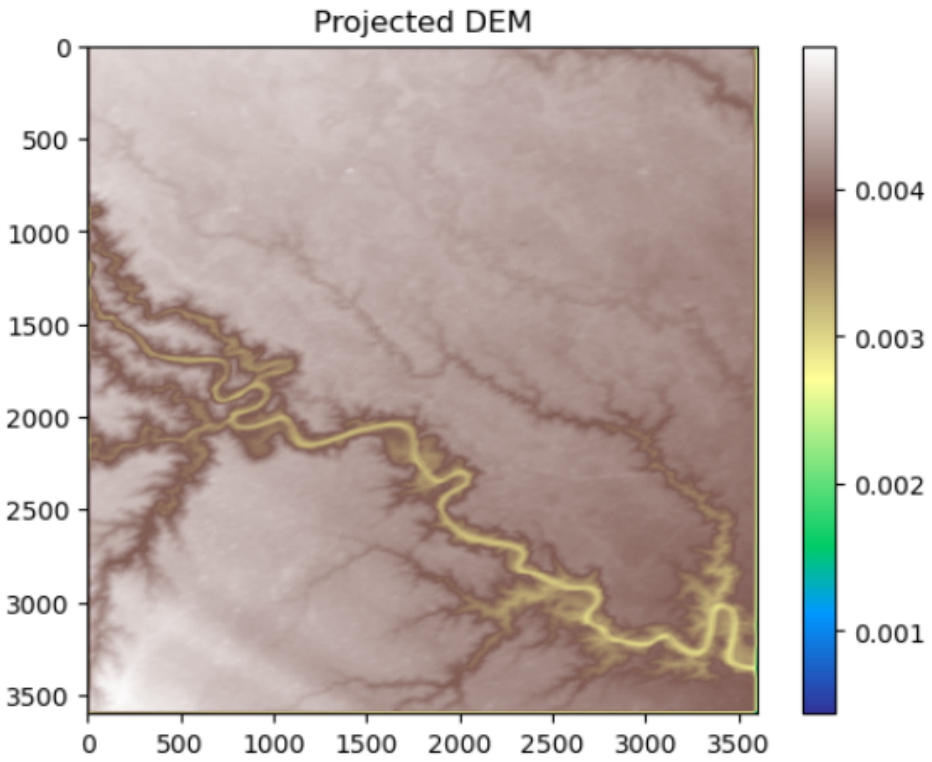
```
NetCDF Header Contents:
-----------------------
<class 'netCDF4._netCDF4.Dataset'>
root group (NETCDF4 data model, file format HDF5):
    Conventions: CF-1.6
    title: IPF SRAL/MWR Level 2 Measurement
    mission_name: Sentinel 3A
    altimeter_sensor_name: SRAL
    radiometer_sensor_name: MWR
    gnss_sensor_name: GNSS
    doris_sensor_name: DORIS
    netcdf_version: 4.2 of Mar 13 2018 10:14:33 $
    product_name: S3A_SR_2_LAN_HY_20230220T161116_20230220T163013_20230407T024613_1136_095_354_____LN3_O_NT_005.SEN3
    institution: LN3
    source: IPF-SM-2
    history:
    contact: eosupport@copernicus.esa.int
    creation_time: 2023-04-07T02:47:24Z
    references: S3IPF PDS 003.2 - i3r2 - Product Data Format Specification - SRAL-MWR Level 2 Land
    processing_baseline: SM_L2HY.005.03.00
    acq_station_name: CGS
    phase_number: 1
    cycle_number: 95
    absolute_rev_number: 36516
    pass_number: 707
    absolute_pass_number: 73033
    equator_time: 2023-02-20T14:25:09.971084Z
    equator_longitude: 113.76792214340986
    semi_major_ellipsoid_axis: 6378137.0
    ellipsoid_flattening: 0.00335281066474748
    first_meas_time: 2023-02-20T16:11:16.082560Z
    last_meas_time: 2023-02-20T16:30:12.666494Z
    first_meas_lat: 18.260717
    last_meas_lat: 80.559799
    first_meas_lon: 84.38189
    last_meas_lon: 17.437625
```

```
NetCDF Attributes:
--------------------
Conventions: CF-1.6
title: IPF SRAL/MWR Level 2 Measurement
mission_name: Sentinel 3A
altimeter_sensor_name: SRAL
radiometer_sensor_name: MWR
gnss_sensor_name: GNSS
doris_sensor_name: DORIS
netcdf_version: 4.2 of Mar 13 2018 10:14:33 $
product_name: S3A_SR_2_LAN_HY_20230220T161116_20230220T163013_20230407T024613_1136_095_354______LN3_O_NT_005.SEN3
institution: LN3
source: IPF-SM-2
history:
contact: eosupport@copernicus.esa.int
creation_time: 2023-04-07T02:47:24Z
references: S3IPF PDS 003.2 - i3r2 - Product Data Format Specification - SRAL-MWR Level 2 Land
processing_baseline: SM_L2HY.005.03.00
acq_station_name: CGS
phase_number: 1
cycle_number: 95
absolute_rev_number: 36516
pass_number: 707
absolute_pass_number: 73033
equator_time: 2023-02-20T14:25:09.971084Z
equator_longitude: 113.76792214340986
semi_major_ellipsoid_axis: 6378137.0
ellipsoid_flattening: 0.00335281066474748
first_meas_time: 2023-02-20T16:11:16.082560Z
last_meas_time: 2023-02-20T16:30:12.666494Z
first_meas_lat: 18.260717
last_meas_lat: 80.559799
first_meas_lon: 84.38189
last_meas_lon: 17.437625
xref_altimeter_level1: S3A_SR_1_LAN_RD_20230220T145023_20230220T154052_20230321T002555_3029_095_353______LN3_O_NT_005.SEN3,S
3A_SR_1_LAN_RD_20230220T154052_20230220T163122_20230321T012643_3029_095_353______LN3_O_NT_005.SEN3,S3A_SR_1_LAN_RD_20230220T
163122_20230220T172150_20230321T022257_3028_095_354______LN3_O_NT_005.SEN3
xref_radiometer_level1: S3A_MW_1_MWR____20230220T144701_20230220T162640_20230320T230718_5979_095_353______LN3_O_NT_001.SEN3,
S3A_MW_1_MWR____20230220T162641_20230220T180633_20230321T010315_5991_095_354______LN3_O_NT_001.SEN3
xref_orbit_data: S3A_SR___POESAX_20230219T215523_20230221T002323_20230307T160503_____________________CNE_O_NT_001.SEN3
xref_pf_data: S3A_SR_2_PCPPAX_20230219T215942_20230220T235942_20230317T071221_____________________POD_O_NT_001.SEN3
xref_altimeter_characterisation: S3A_SR___CHDNAX_20160216T000000_20991231T235959_20200312T120000_____________________MPC_O_AL_
006.SEN3
xref_radiometer_characterisation: S3A_MW___CHDNAX_20160216T000000_20991231T235959_20210929T120000_____________________MPC_O_AL
_005.SEN3
```



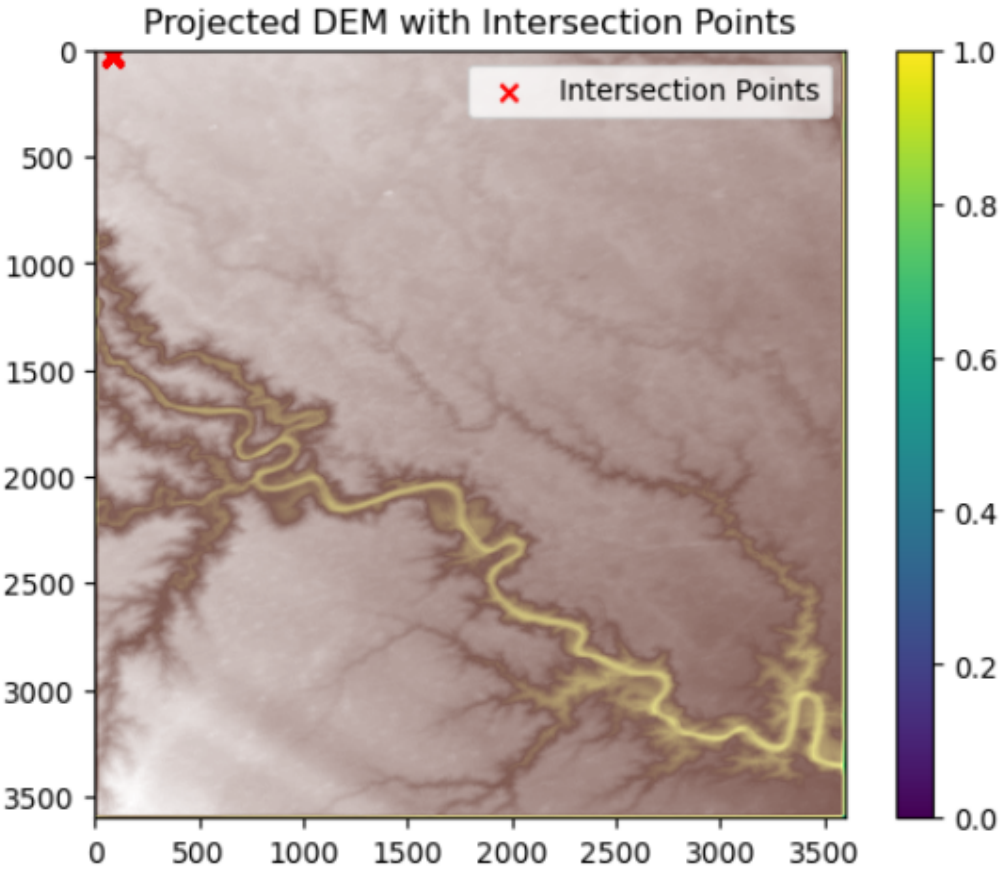
2. Project the DEM onto the river section plane by assigning NaN values to pixels outside the river section.

```
Data type of projected DEM:
<class 'numpy.ndarray'>
```



3. Compute the intersection points .
   3.1. Use algorithm:

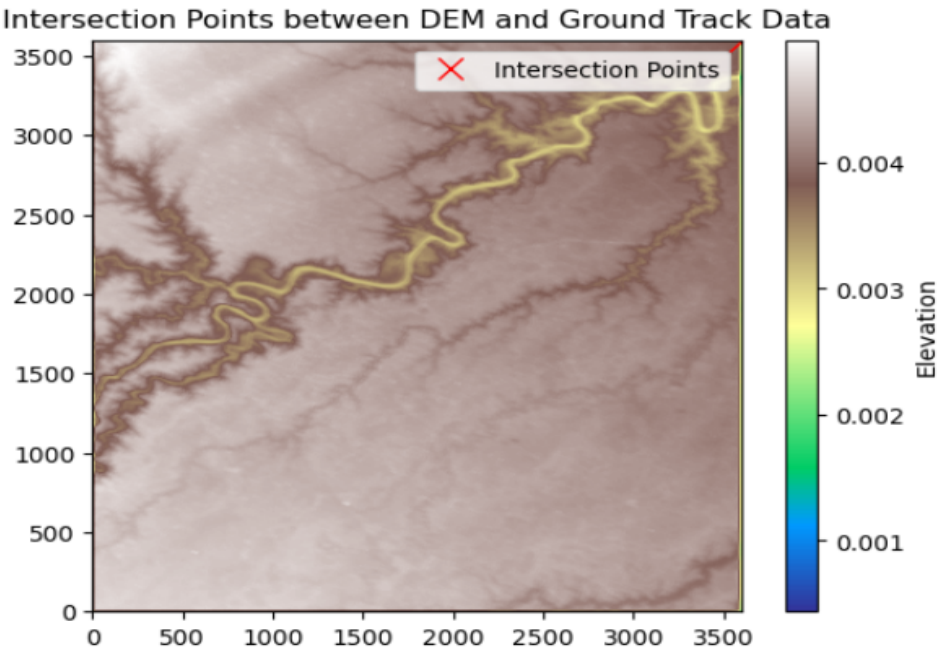
```python
# Flatten the latitude and longitude arrays
lat_lon_points = np.column_stack((latitude.flatten(), longitude.flatten()))

# Create a KDTree from the valid points in the projected DEM
valid_indices = np.argwhere(~np.isnan(dem_projected))
valid_points = np.column_stack((longitude[valid_indices[:, 1]], latitude[valid_indices[:, 0]]))
tree = cKDTree(valid_points)

# Find the nearest neighbors for each ground track point
ground_track_points = np.column_stack((longitude, latitude))
distances, indices = tree.query(ground_track_points)

# Identify the intersection points
intersection_indices = np.where(distances == 0)[0]
intersection_points = ground_track_points[intersection_indices]

# Plot the projected DEM
plt.imshow(dem_projected, cmap='terrain', vmin=np.nanmin(smoothed_elevation), vmax=np.nanmax(smoothed_elevation))

# Plot the intersection points on top of the DEM
plt.scatter(intersection_points[:, 0], intersection_points[:, 1], color='red', marker='x', label='Intersection Points')
```




```
Shape of projected DEM: (3601, 3601)
```

3.2. Use algorithm:

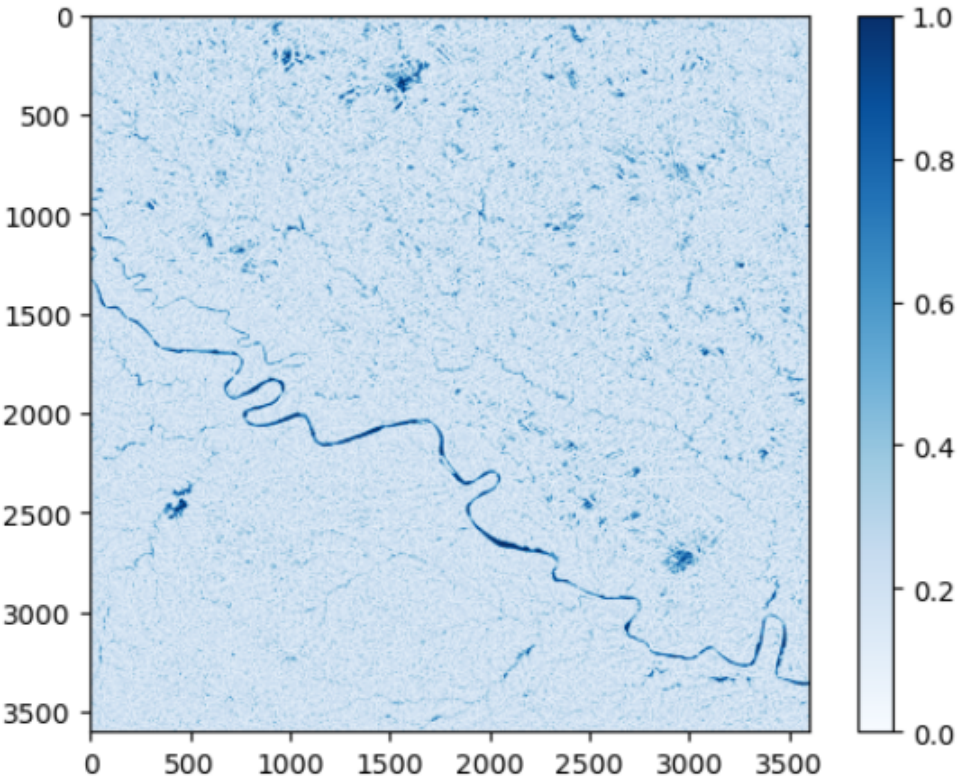
```python
# Compute the projected grid coordinates for ground track data
lon = ground_track_data[:, 0]
lat = ground_track_data[:, 1]
lon_min, lon_max = np.min(lon), np.max(lon)
lat_min, lat_max = np.min(lat), np.max(lat)
x = np.interp(lon, (lon_min, lon_max), (0, dem_projected.shape[1] - 1)).astype(int)
y = np.interp(lat, (lat_min, lat_max), (0, dem_projected.shape[0] - 1)).astype(int)

# Find intersection points within the river section
intersecting_points = np.column_stack((x, y))[river_section_mask[y, x]]
```

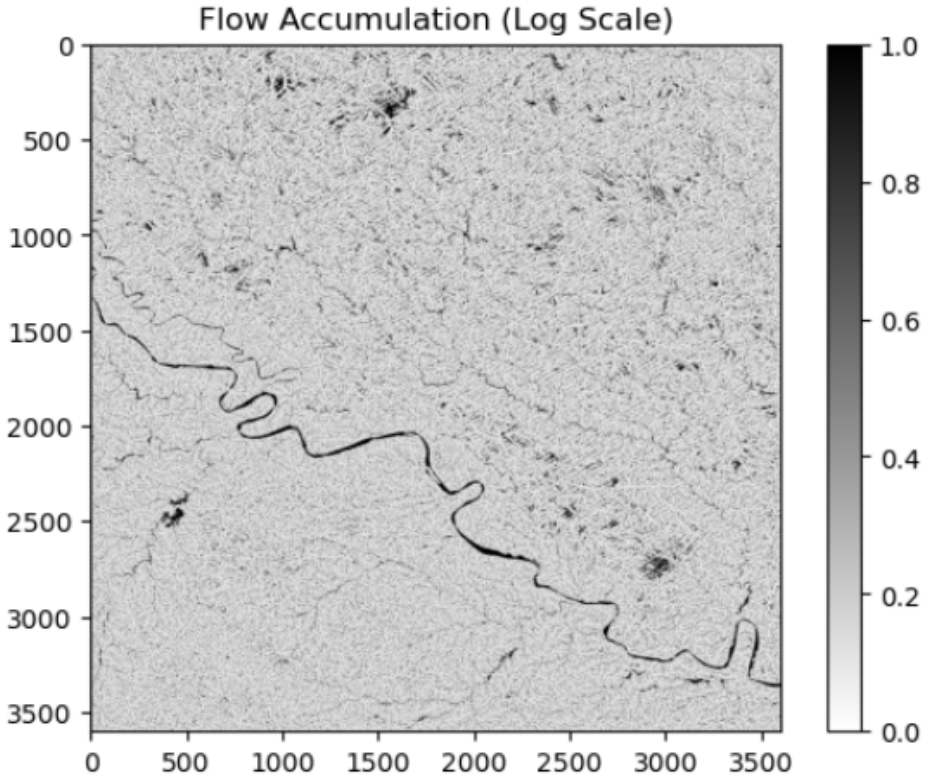


Method 2:

1. Use the same ground track data as used in method 1.

2. Extract river drainage network from flow accumulation map obtained from QGIS by applying a threshold value.



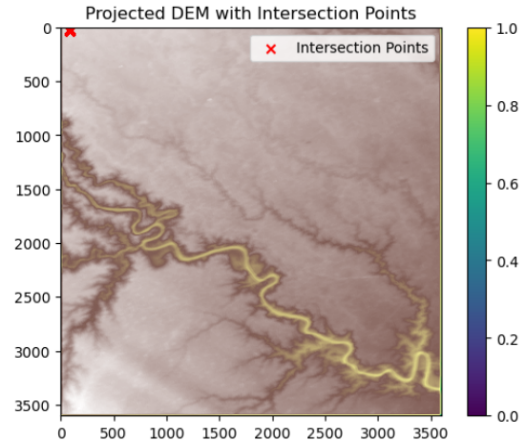


3. Compute intersection points.

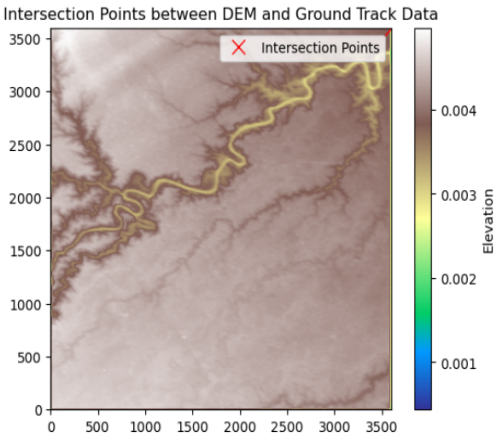
No map displayed due to errors in the calculation.

Results

The final intersection points by method 1 were not accurate. The 1st graph gave me one intersection point that does not lie on the river surface and the 2nd graph gave me those points which were out of bounds.



Shape of projected DEM: (3601, 3601)



The final intersection points by method 2 had too many errors to be displayed.

References

- Berry, P. A. M. (2006) Two decades of inland water monitoring using satellite radar altimetry In: 15 Years of Progress in Radar Altimetry (Proc. Symp., Venice Lido, Italy, 13–18 March 2006). European Space Agency Special Publ. ESA SP614. ESA, Noordwijk, The Netherlands.
- Chelton, D. B., Ries, J. C., Haines, B. J., Fu, L. & Callahan, P. S. (2001) Satellite altimetry. In: Satellite Altimetry and Earth Sciences: A Handbook of Techniques and Applications (ed. by J. Fu & A. Cazenave). Academic Press, San Diego, California, USA.
- Roux, E., Santos da Silva, J., Vieira Getirana, A. C., Bonnet, M.-P., Calmant, S., Martinez, J.-M. & Seyler, F. (2010) Producing time series of river water height by means of satellite radar altimetry—comparative study. Hydrol. Sci. J. 55(1), 104–120.
- Flow computation on massive grids. Lars Arge, Jeffrey S. Chase, Patrick N. Halpin, Laura Toma, Jeffrey S. Vitter, Dean Urban and Rajiv Wickremesinghe. In *Proc. ACM Symposium on Advances in Geographic Information Systems*, 2001.
- GRASS Development Team (2006) Geographic Resources Analysis Support System (GRASS) Software. ITC-irst, Trento, Italy. http://grass.itc.it.
- Berry, P. A. M., Garlick, J. D., Freeman, J. A. & Mathers, E. L. (2005) Global inland water monitoring from multi-mission altimetry. Geophys. Res. Lett. 32(L16401), 1–4.
- Saunders, W. (1999) Preparation of DEMs for use in environmental modeling analysis. ESRI User Conference (24–30 July 1999, San Diego, California). Available at: http://proceedings.esri.com/library/userconf/proc99/navigate/proceed.htm