Flood / Water Change Detection Report of GURDASPUR district of Punjab year 2025 (Start → End)

Why I have chosen this project?

I chose this problem statement because it improved **my real-world problem-solving ability** and **strengthened my practical industry skills**. Flood/water detection is a real operational need, and by completing this task end-to-end (**data download → preprocessing → change detection → GIS output**), I practiced the same type of workflow that industries use for **monitoring, reporting, and decision-making.**

1. Project goal

The goal of this project was to detect water/flood-related changes inside the ROI (Gurdaspur district, Punjab, India) by comparing two time periods:

T1 (Dry season baseline): 2025-04-01 to 2025-04-30

T2 (Monsoon/flood season): 2025-08-15 to 2025-09-15

The output should be usable for mapping and interpretation, and finally shown in ArcMap.

2. Study area (ROI)

I prepared an ROI polygon for Gurdaspur (Punjab) and stored it as:
data/roi_1.geojson

I ensured the ROI coordinates are valid in EPSG:4326 (WGS84 lat/long).

3. Software & setup I have used

I used the following platforms across the workflow:

- Visual Studio Code: used to run the automated download pipeline and scripts.
- Jupyter Lab: used to run and debug the processing notebooks (coregistration + change detection).
- ArcGIS Desktop (ArcMap): used for final visualization, checking raster values, and preparing the final map output.

Python environment with required libraries (example: geopandas, rasterio etc.).

I stored Copernicus Data Space Sentinel Hub credentials in a .env file:
SH_CLIENT_ID
SH_CLIENT_SECRET

I ran notebooks/scripts from the project structure (data / output folders).

I did automated download from Visual Studio Code using Copernicus Data Space (Copernicus website platform) through Sentinel Hub APIs.

4. Satellite datasets used (with bands / channels)

A) Optical data
Satellite: Sentinel-2

Product: Sentinel-2 L2A (surface reflectance) via Copernicus Data Space / Sentinel Hub
Bands I extracted:

B03 (Green)

B08 (NIR / Near Infrared)

SCL (Scene Classification Layer) (used for cloud masking)

B) SAR data
Satellite: Sentinel-1
Product: Sentinel-1 GRD via Copernicus Data Space / Sentinel Hub
Channels I extracted:

VV polarization

VH polarization

5. Download method (important part of your pipeline)

Because the ROI can be large, I used an automated approach that:

Reads my ROI bounds from the GeoJSON.

Plans a tile grid so each request stays under Sentinel Hub limits (MAX_DIM = 2500 pixels).

Downloads each tile for:

S2 (Sentinel 2) at T1 (dry season)

S2 (Sentinel 2) at T2 (Monsoon season)

S1 (Sentinel 1) at T1 (dry season)

S1(Sentinel 1) at T2 (Monsoon season)

Mosaics (merges) all tiles into one single GeoTIFF per product.

Deletes temporary tiles and temporary folders, leaving only final outputs.

So even if many tiles are created internally, the final saved results are only the main products.

6. Output spatial resolution (what you actually produced)

In my code settings I used:

DESIRED_RES_M = 10.0 meters per pixel

The tiling planner keeps it near this resolution unless it must increase resolution (coarser) to reduce tiles.

So my delivered resolution is:

Sentinel-2 outputs: ~10 m/pixel (even though native B03/B08 are 10m, my requested output grid was ~10m)

Sentinel-1 outputs: ~10 m/pixel (because I downloaded and processed it on the same planned grid)

7. Cloud handling you applied (Optical only)

I applied cloud handling in two ways:

A) Cloud filter during mosaicking (data selection)
For Sentinel-2 requests I used:

mosaicking_order = LEAST_CC (pick least cloudy scenes inside the date window)

maxcc = 0.2 meaning use scenes with $\leq$ 20% cloud cover for selection (when available)

9. What final GeoTIFFs I created (Step 3 output)

After the download + mosaic + clip pipeline, I produced only 4 GeoTIFF files:

Optical (Sentinel-2)

s2_t1.tif (bands: B03, B08, SCL, cloudMask)

s2_t2.tif (bands: B03, B08, SCL, cloudMask)

SAR (Sentinel-1)

s1_t1.tif (bands: VV, VH)

s1_t2.tif (bands: VV, VH)

These are the main inputs for change detection.

10. Coregistration (Step 4 result)

I completed coregistration so that:

All T1 and T2 rasters align on the same pixel grid (same CRS, same resolution, same transform).

This step is necessary so each pixel represents the same ground location in every raster.

After Step 4, I worked with these aligned inputs:

s2_t1_coreg.tif, s2_t2_coreg.tif

s1_t1_coreg.tif, s1_t2_coreg.tif

11. Change detection (Step 5)

I performed change detection separately for optical and SAR.

A) Optical change (Sentinel-2) using NDWI
I used:

Green = B03

NIR = B08

I computed NDWI:
NDWI=(Green-NIR)/(Green+NIR)

I also applied masking:

Pixels where cloudMask < 0.5 or nodata were set to NaN (ignored).

Then I created water maps for each time:

WATER_1 from NDWI in T1

WATER_2 from NDWI in T2

Threshold selection:

I used Otsu thresholding (and because skimage was missing, I corrected it by using a NumPy-based Otsu function).

Finally, the optical change result:

OPT_NEW_WATER = pixels that are water in T2 but not in T1
(This is the "new water / flood-like" mask from optical data.)

B) SAR change (Sentinel-1) using VV backscatter change
I used SAR channels:

VV (and VH also loaded)

I converted to dB and computed:

VV_change = VV_T2(dB) - VV_T1(dB)

Then I used Otsu thresholding on VV change and created:

SAR_FLOOD = areas where VV change indicates flood-like behavior (VV decrease side).

12. Step 5 outputs saved

I saved the main change outputs into:

output/change/
and created figures into:

output/figures/

I also fixed plotting errors where variables were not in memory by loading rasters again when needed.

13. ArcMap final mapping and why min/max showed only 1

When I opened the mask raster in ArcMap, I observed:

Many pixels showed NoData

Some pixels showed 1

So raster statistics showed min=max=1 (because the only real value stored was 1; everything else was nodata)

This means my raster was effectively:

1 = change/flood

NoData = not marked (instead of 0)

To make it usable as a standard binary mask (0/1), I corrected it in ArcMap by converting NoData to 0:

NoData → 0

1 stays 1

After that, the raster becomes a proper:

0 = no flood

1 = flood

14. Final deliverables you prepared in Step 6

In Step 6, I used ArcMap to create a final usable output map layer and export it for reporting.

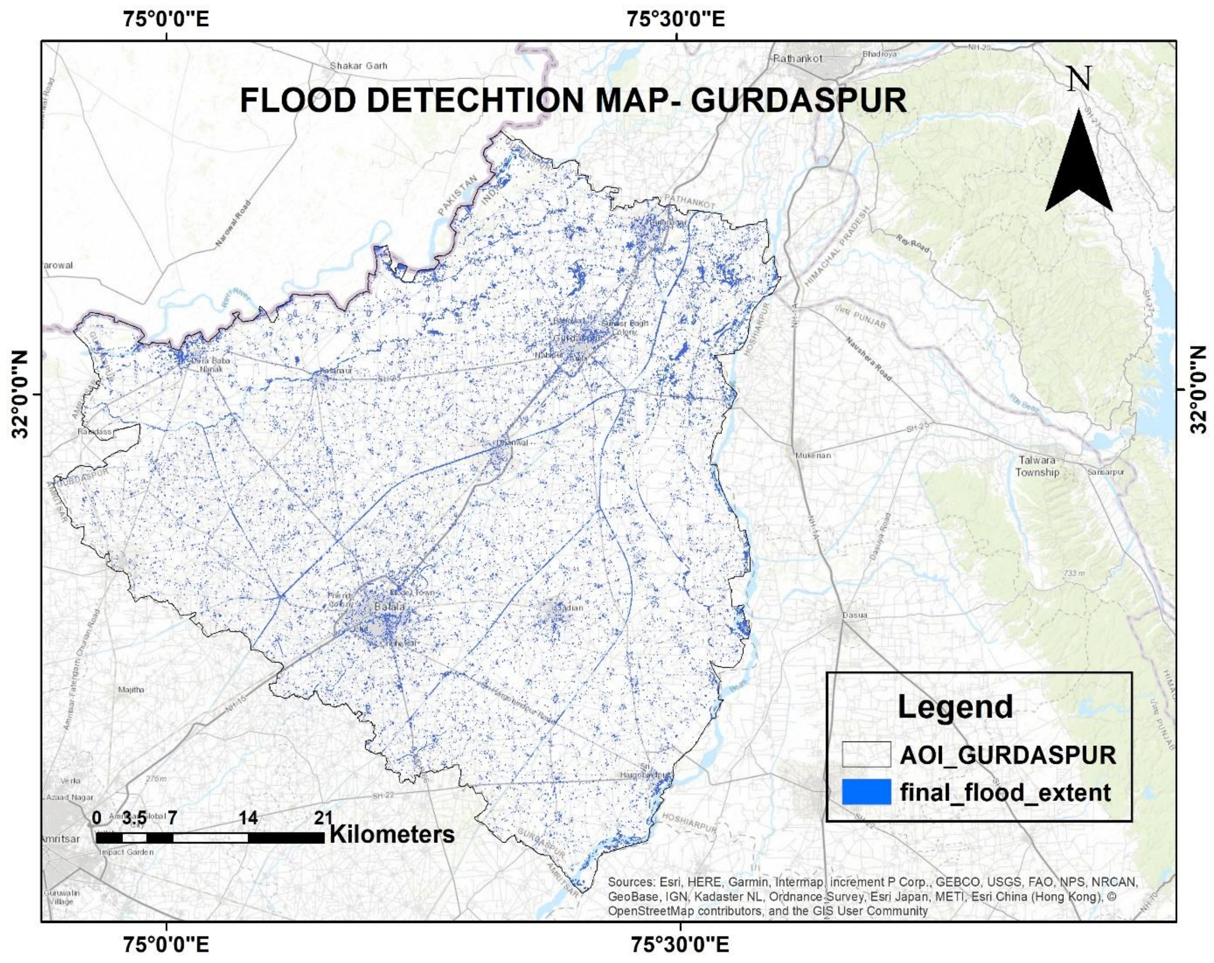
My final deliverables are:

Final flood/change extent layer (single final output produced from the mask workflow)

Final map export (PNG/PDF from ArcMap Layout view)

Short report (this document) describing datasets + methods + output

Final output map


75°0'0"E
75°30'0"E
32°0'0"N
FLOOD DETECHTION MAP- GURDASPUR
N
Legend
AOI_GURDASPUR
final_flood_extent
0 3.5 7 14 21
Kilometers
Sources: Esri, HERE, Garmin, Intermap, increment P Corp., GEBCO, USGS, FAO, NPS, NRCAN, GeoBase, IGN, Kadaster NL, Ordnance Survey, Esri Japan, METI, Esri China (Hong Kong), © OpenStreetMap contributors, and the GIS User Community