





Google Earth Engine

Hands on Experience

Project 1 Flood Mapping and Damage Assessment



Introduction

Flooding

A flood is an overflow of water onto land that is normally dry. There are a few types of floods – river flood, coastal flood and flash flood. Flooding is a significant and recurring natural disaster with far-reaching consequences, affecting communities, infrastructure and ecosystems worldwide.

With global warming, the frequency and severity of flooding have increased. Having close to real time data on the flood extent is crucial for humanitarian efforts to be relevant, timely and effective.

Leveraging on the power of Google Earth Engine, this project focuses on utilising Synthetic Aperture Radar (SAR) images from Sentinel-1 and geospatial analytics to map flood extents and assess the resulting damages.



Flood in Zhuozhou, in northern China's Hebei Province
on 02 August 2023

Sentinel-1 SAR GRD

Mapping of flooded areas is done by comparing images taken before and after a flood event. Since most flood events are caused by heavy rainfall, optical imagery has limited applications due to cloud cover. Synthetic Aperture Radar (SAR) imagery is not affected by weather and provides a robust and reliable way to detect flood regions. In this project, we will be using Sentinel-1 SAR imagery.

Sentinel-1 Synthetic Aperture Radar Ground Range Detected

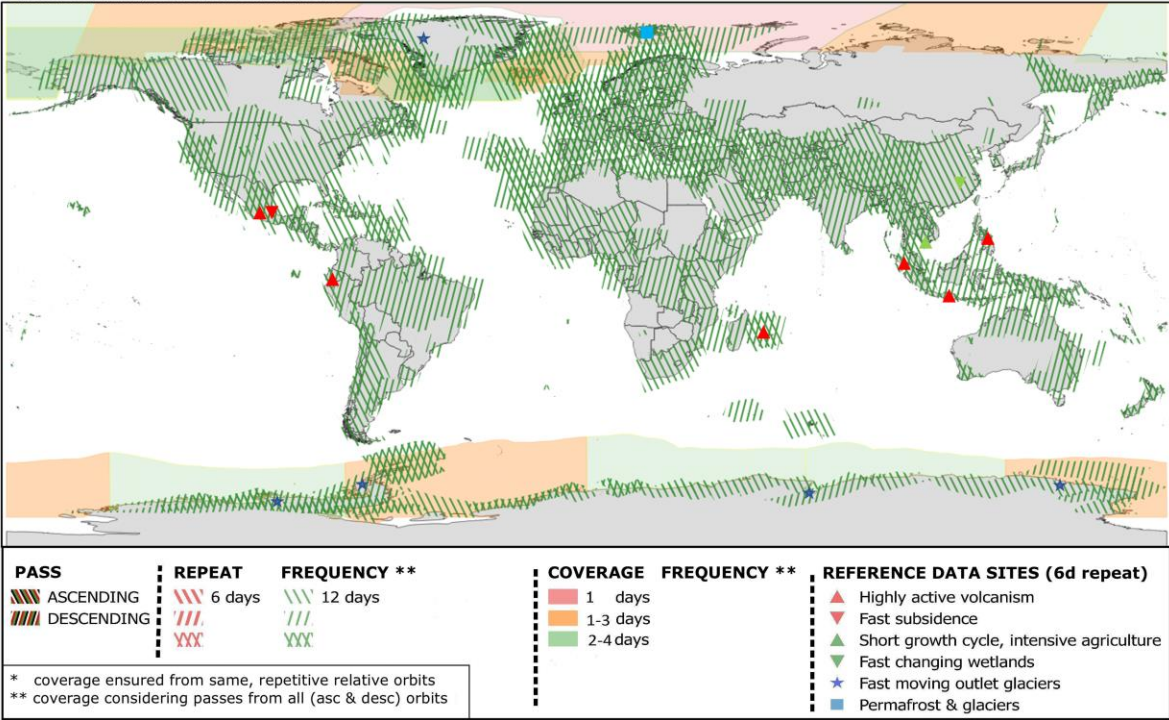
- Missions consists of 2 satellites - Sentinel-1A and Sentinel-1B
- C-band Synthetic Aperture Radar (SAR) sensor
- Spatial Resolution: 10m
- Temporal Resolution: 6-12 days

Sentinel-1A Mission Observation Scenario: Repeat & Coverage Frequency



validity start: 09/2022

Note: Seasonal campaigns not represented
Note: Wave mode systematically operated over open oceans not represented



Guided Lab

1. Set Time Frame

```
// Set the parameters for BEFORE the flood
var before_start= '2019-03-01';
var before_end='2019-03-10';

// Now set the same parameters for AFTER the flood.
var after_start='2019-03-10';
var after_end='2019-03-23';
```

2. Set SAR Parameters

```
var polarization = "VH";
var pass_direction = "DESCENDING";
var difference_threshold = 1.25;
```

Guided Lab

3. Data Selection & Preprocessing

```
// rename selected geometry feature
var aoI = ee.FeatureCollection(geometry);

// Load and filter Sentinel-1 GRD data by predefined parameters
var collection= ee.ImageCollection('COPERNICUS/S1_GRD')
  .filter(ee.Filter.eq('instrumentMode','IW'))
  .filter(ee.Filter.listContains('transmitterReceiverPolarisation',
polarization))
  .filter(ee.Filter.eq('orbitProperties_pass',pass_direction))
  .filter(ee.Filter.eq('resolution_meters',10))
  // .filter(ee.Filter.eq('relativeOrbitNumber_start',relative_orbit ))
  .filterBounds(aoI)
  .select(polarization);

// Select images by predefined dates
var before_collection = collection.filterDate(before_start, before_end);
var after_collection = collection.filterDate(after_start,after_end);

// Print selected tiles to the console

// Extract date from meta data
function dates(imgcol){
  var range = imgcol.reduceColumns(ee.Reducer.minMax(),
["system:time_start"]);
  var printed = ee.String('from ')
    .cat(ee.Date(range.get('min')).format('YYYY-MM-dd'))
    .cat(' to ')
    .cat(ee.Date(range.get('max')).format('YYYY-MM-dd'));
  return printed;
}
// print dates of before images to console
var before_count = before_collection.size();
print(ee.String('Tiles selected: Before Flood
').cat('(').cat(before_count).cat(')'),
  dates(before_collection), before_collection);

// print dates of after images to console
var after_count = after_collection.size();
print(ee.String('Tiles selected: After Flood
').cat('(').cat(after_count).cat(')'),
  dates(after_collection), after_collection);

// Create a mosaic of selected tiles and clip to study area
var before = before_collection.mosaic().clip(aoI);
var after = after_collection.mosaic().clip(aoI);

// Apply reduce the radar speckle by smoothing
var smoothing_radius = 50;
var before_filtered = before.focal_mean(smoothing_radius, 'circle', 'meters');
var after_filtered = after.focal_mean(smoothing_radius, 'circle', 'meters');
```

Guided Lab

4. Flood Extent Calculation

```
// Calculate the difference between the before and after images
var difference = after_filtered.divide(before_filtered);

// Apply the predefined difference-threshold and create the flood extent mask
var threshold = difference_threshold;
var difference_binary = difference.gt(threshold);

// Refine flood result using additional datasets

// Include JRC layer on surface water seasonality to mask flood pixels from areas
// of "permanent" water (where there is water > 10 months of the year)
var swater = ee.Image('JRC/GSW1_0/GlobalSurfaceWater').select('seasonality');
var swater_mask = swater.gte(10).updateMask(swater.gte(10));

// Flooded layer where perennial water bodies (water > 10 mo/yr) is assigned a 0 value
var flooded_mask = difference_binary.where(swater_mask, 0);
// final flooded area without pixels in perennial waterbodies
var flooded = flooded_mask.updateMask(flooded_mask);

// Compute connectivity of pixels to eliminate those connected to 8 or fewer
// neighbours
// This operation reduces noise of the flood extent product
var connections = flooded.connectedPixelCount();
var flooded = flooded.updateMask(connections.gte(8));

// Mask out areas with more than 5 percent slope using a Digital Elevation Model
var DEM = ee.Image('WWF/HydroSHEDS/03VDEM');
var terrain = ee.Algorithms.Terrain(DEM);
var slope = terrain.select('slope');
var flooded = flooded.updateMask(slope.lt(5));

// Calculate flood extent area
// Create a raster layer containing the area information of each pixel
var flood_pixelarea = flooded.select('polarization')
    .multiply(ee.Image.pixelArea());

// Sum the areas of flooded pixels
// default is set to 'bestEffort: true' in order to reduce computation time, for a more
// accurate result set bestEffort to false and increase 'maxPixels'.
var flood_stats = flood_pixelarea.reduceRegion({
    reducer: ee.Reducer.sum(),
    geometry: aoi,
    scale: 10, // native resolution
    //maxPixels: 1e9,
    bestEffort: true
});

// Convert the flood extent to hectares (area calculations are originally given in meters)
var flood_area_ha = flood_stats
    .getNumber('polarization')
    .divide(10000)
    .round();
```

Guided Lab

5. Exposed Population Density

```
// Load JRC Global Human Settlement Population Density layer
// Resolution: 250. Number of people per cell is given.
var population_count =
ee.Image('JRC/GHSL/P2016/POP_GPW_GLOBE_V1/2015').clip(aoi);

// Calculate the amount of exposed population
// get GHSL projection
var GHSLprojection = population_count.projection();

// Reproject flood layer to GHSL scale
var flooded_res1 = flooded
  .reproject({
    crs: GHSLprojection
  });

// Create a raster showing exposed population only using the resampled flood
layer
var population_exposed = population_count
  .updateMask(flooded_res1)
  .updateMask(population_count);

//Sum pixel values of exposed population raster
var stats = population_exposed.reduceRegion({
  reducer: ee.Reducer.sum(),
  geometry: aoi,
  scale: 250,
  maxPixels:1e9
});

// get number of exposed people as integer
var number_pp_exposed = stats.getNumber('population_count').round();
```

Guided Lab

6. Affected Agricultural Land

```
// using MODIS Land Cover Type Yearly Global 500m
// filter image collection by the most up-to-date MODIS Land Cover product
var LC = ee.ImageCollection('MODIS/006/MCD12Q1')
  .filterDate('2014-01-01',after_end)
  .sort('system:index',false)
  .select("LC_Type1")
  .first()
  .clip(aoi);

// Extract only cropland pixels using the classes cropland (>60%) and
// Cropland/Natural
// Vegetation Mosaics: mosaics of small-scale cultivation 40-60% incl. natural
// vegetation
var cropmask = LC
  .eq(12)
  .or(LC.eq(14))
var cropland = LC
  .updateMask(cropmask)

// get MODIS projection
var MODISprojection = LC.projection();

// Reproject flood layer to MODIS scale
var flooded_res = flooded
  .reproject({
    crs: MODISprojection
  });

// Calculate affected cropland using the resampled flood layer
var cropland_affected = flooded_res
  .updateMask(cropland)

// get pixel area of affected cropland layer
var crop_pixelarea = cropland_affected
  .multiply(ee.Image.pixelArea()); //calculate the area of each pixel

// sum pixels of affected cropland layer
var crop_stats = crop_pixelarea.reduceRegion({
  reducer: ee.Reducer.sum(), //sum all pixels with area
  information:
    geometry: aoi,
    scale: 500,
    maxPixels: 1e9
  });

// convert area to hectares
var crop_area_ha = crop_stats
  .getNumber('polarization')
  .divide(10000)
  .round();
```


Guided Lab

6. Affected Urban Area

```
// Using the same MODIS Land Cover Product
// Filter urban areas
var urbanmask = LC.eq(13)
var urban = LC
  .updateMask(urbanmask)

//Calculate affected urban areas using the resampled flood layer
var urban_affected = urban
  .mask(flooded_res)
  .updateMask(urban);

// get pixel area of affected urban layer
var urban_pixelarea = urban_affected
  .multiply(ee.Image.pixelArea()); //calculate the area of each pixel

// sum pixels of affected cropland layer
var urban_stats = urban_pixelarea.reduceRegion({
  reducer: ee.Reducer.sum(), //sum all pixels with area
  information
    geometry: aoi,
    scale: 500,
    bestEffort: true,
  });

// convert area to hectares
var urban_area_ha = urban_stats
  .getNumber('LC_Type1')
  .divide(10000)
  .round();
```

Guided Lab

7. Displaying products on the map

```
// Before and after flood SAR mosaic
Map.centerObject(aoi,8);
Map.addLayer(before_filtered, {min:-25,max:0}, 'Before Flood',0);
Map.addLayer(after_filtered, {min:-25,max:0}, 'After Flood',1);

// Difference layer
Map.addLayer(difference,{min:0,max:2},"Difference Layer",0);

// Flooded areas
Map.addLayer(flooded,{palette:"0000FF"},"Flooded areas");

// Population Density
var populationCountVis = {
  min: 0,
  max: 200.0,
  palette: ['060606','337663','337663','ffffff'],
};
Map.addLayer(population_count, populationCountVis, 'Population Density',0);

// Exposed Population
var populationExposedVis = {
  min: 0,
  max: 200.0,
  palette: ['yellow', 'orange', 'red'],
};
Map.addLayer(population_exposed, populationExposedVis, 'Exposed Population');

// MODIS Land Cover
var LCVis = {
  min: 1.0,
  max: 17.0,
  palette: [
    '05450a', '086a10', '54a708', '78d203', '009900', 'c6b044', 'dcd159',
    'dade48', 'fbff13', 'b6ff05', '27ff87', 'c24f44', 'a5a5a5', 'ff6d4c',
    '69fff8', 'f9ffa4', '1c0dff'
  ],
};
Map.addLayer(LC, LCVis, 'Land Cover',0);

// Cropland
var croplandVis = {
  min: 0,
  max: 14.0,
  palette: ['30b21c'],
};
Map.addLayer(cropland, croplandVis, 'Cropland',0)

// Affected cropland
Map.addLayer(cropland_affected, croplandVis, 'Affected Cropland');

// Urban
var urbanVis = {
  min: 0,
  max: 13.0,
  palette: ['grey'],
};
Map.addLayer(urban, urbanVis, 'Urban',0)

// Affected urban
Map.addLayer(urban_affected, urbanVis, 'Affected Urban');
```