Project Documentation

Name: Ayush Tiwari | Roll no. 21MI31010 | Department of Mining Engineering | Indian Institute of Technology Kharagpur

Title: Mine Subsidence Monitoring using Sentinel-1 SAR Data

Prepared by: Ayush Tiwari (Roll No. 21MI31010)
Department of Mining Engineering, IIT Kharagpur

1. Introduction

Mine subsidence monitoring is critical to assess ground deformation caused by underground mining.
We use Sentinel-1 Synthetic Aperture Radar (SAR) data because it provides:
- Millimeter-scale deformation detection using InSAR
- Frequent revisits (6–12 days)
- All-weather, day/night monitoring

In this project:
- We fetch Sentinel-1 SAR data from Google Earth Engine (GEE).
- Process it to obtain backscatter time series (proxy for surface change).
- Finally, display the Time vs Subsidence Graph on a website.

2. Prerequisites & Installation

Tools Needed:
1. Google Earth Engine (GEE) account → https://earthengine.google.com/
   - Free signup using Gmail.
   - Verify and activate your account.
   - Create a new project and note down its project id.

2. Python environment (Anaconda or venv recommended)
   Install required libraries:
   pip install earthengine-api pandas

3. Web Framework – we'll use Chart.js to serve charts on a webpage.

4. Google Maps API needed

   - Create an account in Google Earth Maps API and authenticate.

   - An API will be assigned to you, copy the API key.

   - Replace 'APIKey' by your actual API Key in the code:

```
src="https://maps.googleapis.com/maps/api/js?key=APIKeycallback=initMap"
```

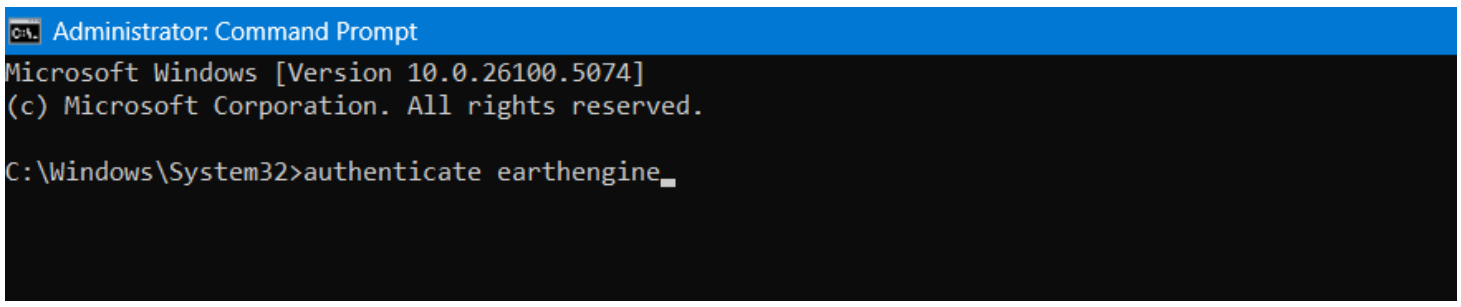
3. Workflow

Step-by-step process:
1. Authenticate Google Earth Engine API.
2. Define Area of Interest (AOI) – Polygon of Singareni mines and specify the date range.
3. Load Sentinel-1 GRD Collection (backscatter data) and filter by AOI, Date.
4. Reduce the image to the mine region.
6. Export / Fetch Time Series into Python.
7. Convert values to csv and json (date vs backscatter).
8. Plot Graph using Chart.js in HTML.
9. interactive chart on website.

4. Code Walkthrough

The Python code includes the following steps:

- Authenticate Earth Engine via terminal



- Import required libraries

```
# ------------------------------
# 1. Import Libraries
# ------------------------------
import ee
import pandas as pd
import json
```

- Authenticate Earth Engine and initialize project with project name created on earthengine account

```
# 1. Trigger the authentication flow.
ee.Authenticate()

ee.Initialize(project="healthy-saga-471512-a4")
```

- Define Area of Interest (AOI)

```
# ------------------------------
# 2. Define AOI and Date Range
# ------------------------------
# 500m x 500m rectangle over Singareni Mines, Yellandu
aoi = ee.Geometry.Polygon([
    [[80.31646, 17.58814],
     [80.31646, 17.59314],
     [80.32146, 17.59314],
     [80.32146, 17.58814]]
])
```

- Set the date range

```
# Date range
start_date = '2014-01-01'
end_date = '2025-06-30'
```

- Load Sentinel-1 Image Collection
    - filter by selected region of interest
    - filter by date
    - specify parameters to select the backscatter data

```
# ------------------------------
# 3. Load Sentinel-1 Collection
# ------------------------------
s1 = ee.ImageCollection('COPERNICUS/S1_GRD') \
       .filterBounds(aoi) \
       .filterDate(start_date, end_date) \
       .filter(ee.Filter.eq('instrumentMode', 'IW')) \
       .select('VV')
```

- Reduce the image of Sentinel-1 collected to the specific region of mine

```
# ------------------------------
# 4. Reduce Each Image Over AOI
# ------------------------------
def reduce_image(image):
    mean_dict = image.reduceRegion(
        reducer=ee.Reducer.mean(),
        geometry=aoi,
        scale=10
    )
    # Handle null values
    vv = ee.Algorithms.If(mean_dict.get('VV'), mean_dict.get('VV'), -9999)

    return ee.Feature(None, {
        'date': image.date().format('YYYY-MM-dd'),
        'VV': vv
    })

fc = s1.map(reduce_image)

# Filter out nulls
fc_filtered = fc.filter(ee.Filter.neq('VV', -9999))
```

- Convert time-series to Pandas DataFrame

```
# ------------------------------
# 5. Convert FeatureCollection to Pandas DataFrame
# ------------------------------
features = fc_filtered.getInfo()['features']
data = [(f['properties']['date'], f['properties']['VV']) for f in features]
df = pd.DataFrame(data, columns=['date', 'VV'])
df.sort_values('date', inplace=True)
df.reset_index(drop=True, inplace=True)
```

- Convert time-series to csv and Json compatible for web viewing

```
# Save CSV
df.to_csv('singareni.csv', index=False)

# ------------------------------
# 6. Convert to JSON for Website
# ------------------------------
chart_json = {'labels': df['date'].tolist(), 'data': df['VV'].tolist()}
with open('singareni.json', 'w') as f:
    json.dump(chart_json, f)

print("CSV and JSON for chart generated successfully!")
```

The HTML code includes the following steps:

- HTML initialization with Chart.js script and styling for the body

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Singareni surface displacement Time Series</title>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 30px;
      background: #f8f9fa;
    }
    #chartContainer {
      width: 80%;
      margin: auto;
    }
    #map {
      height: 500px;
      width: 700px;
      border: 2px solid #333;
      border-radius: 10px;
    }

    #info {
      width: 300px;
      padding: 15px;
      border: 2px solid #333;
      border-radius: 10px;
      background: #f9f9f9;
```

```
    }

    #info h2 {
      margin-top: 0;
    }
  </style>
</head>
```

- Show the map of Singareni mine location in 3D

```
<body>
  <div id="map"></div>
  <div id="info">
    <h2>Elevation Info</h2>
    <p>Click on the map to see elevation.</p>
    <p id="output"></p>
  </div>
  <!-- Load Google Maps API -->
  <script
    src="https://maps.googleapis.com/maps/api/js?key=APIKey&callback=initMap"
    async defer>
  </script>

  <script>
    let map, elevator;

    function initMap() {
      // Example location: Times Square, NYC
      var location = { lat: 17.62750523465081, lng: 80.30424332483172 };

      // Initialize map with 3D view
      map = new google.maps.Map(document.getElementById('map'), {
        center: location,
        zoom: 18,
        mapTypeId: 'satellite',
        tilt: 45,
        heading: 90,
        gestureHandling: "greedy"
      });

      new google.maps.Marker({
        position: location,
        map: map,
        title: "Click to get elevation"
      });
```

- Show the elevation for selected point on map

```
    // Create Elevation Service
    elevator = new google.maps.ElevationService();

    // Add click listener to map
    map.addListener("click", (event) => {
      getElevation(event.latLng);
    });
  }

  function getElevation(latLng) {
    elevator.getElevationForLocations(
      { locations: [latLng] },
      (results, status) => {
        if (status === "OK" && results[0]) {
          document.getElementById("output").innerHTML =
            `<b>Latitude:</b> ${latLng.lat().toFixed(5)}<br>
             <b>Longitude:</b> ${latLng.lng().toFixed(5)}<br>
             <b>Elevation:</b> ${results[0].elevation.toFixed(2)} meters`;
        } else {
          document.getElementById("output").innerHTML =
            "Elevation data not available.";
        }
      }
    );
  }
</script>
```

- Create chart
    - Put the heading of map
    - Initialize the chart
    - Load the data of chart from Json file
    - Define the chart, its type, colour, responsiveness, labels, etc.

```
<h2>Singareni Mines (Yellandu) – Sentinel-1</h2>
<div id="chartContainer">
  <canvas id="Chart"></canvas>
</div>

<script>
  async function loadData() {
    const response = await fetch("singareni.json");
    const chartData = await response.json();

    const chartsubsidence =
document.getElementById("Chart").getContext("2d");
    new Chart(chartsubsidence, {
      type: "line",
```

```
      data: {
        labels: chartData.labels,
        datasets: [{
          label: "Backscatter (dB)",
          data: chartData.data,
          borderColor: "green",
          backgroundColor: "rgba(38, 255, 157, 0.43)",
          fill: true,
          tension: 0.5,
          pointRadius: 3
        }]
      },
      options: {
        responsive: true,
        plugins: {
          legend: { display: true },
          tooltip: { mode: "index", intersect: false }
        },
        scales: {
          x: { title: { display: true, text: "Date" } },
          y: { title: { display: true, text: "Backscatter (dB)" } }
        }
      }
    });
  }
  loadData();
</script>
</body>
</html>
```

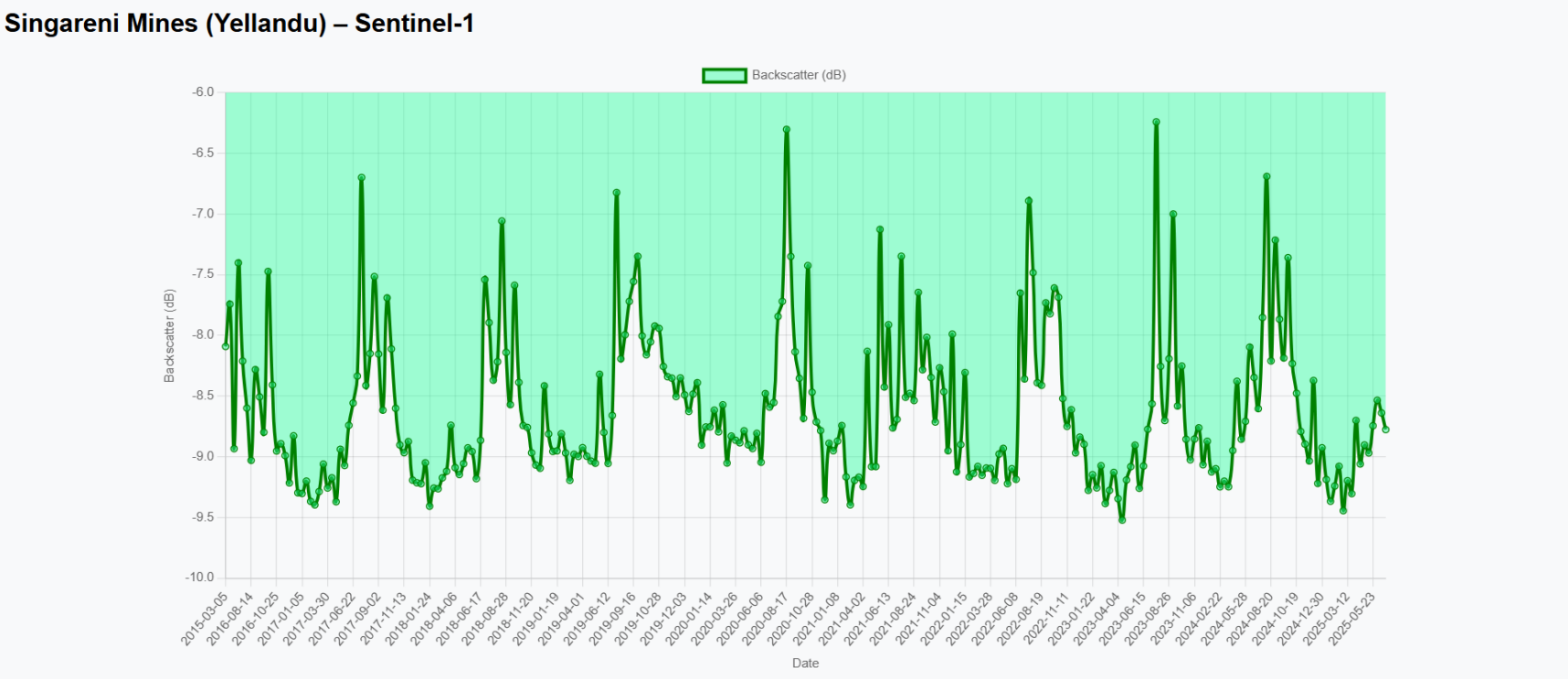
5. How to Run the Code

1. Open Terminal
2. Run earthengine authenticate and follow onscreen prompts
2. Run: python subsidence.py
    - Two new files will get generated singareni.csv and singareni.json.
3. Navigate to the folder where the all the files are located and run in terminal to spin up the server:
    python -m http.server 8000
4. In browser (chrome preferred) browse: http://localhost:8000/index.html
5. The script fetches data and displays the graph of Date vs Backscatter.

6. Output & Interpretation

- Shows map which can be clicked to fetch the real time elevation using google maps API to show the latitude, longitude and elevation.

- Graph shows Date (X-axis) vs Backscatter (Y-axis, in dB).



- The graph shows very small changes in the elevation profile of the order of 10^{-8} mm range which is quite small.
- The reason for the fluctuations is the effect of measurement, weather conditions occurring such as rain, clouds, etc. and vegetation present in the region.

With this workflow, you can:
- Download and process data automatically.
- Display results on your website.