Problem Statement 1:

Improving Spatial Accuracy of POIs and Point Addresses through Visual Re-localization

Objective

Massive open Point of Interest (POI) datasets—such as those from HERE Maps—are rich with information but often suffer from geographic inaccuracies due to limitations in traditional geocoding techniques. This leads to POIs and Point Addresses (PAs) being incorrectly placed on the map, reducing the effectiveness of navigation services, harming user experience, and creating serious issues for precision-based applications like AR and smart delivery systems.

This project aims to develop a **scalable, image-assisted POI and PA re-localization system** that improves spatial accuracy by correcting misplacements through visual triangulation, spatial analysis, and intelligent automation.

Why This Problem Matters

POIs and PAs end up in the wrong locations for several reasons:

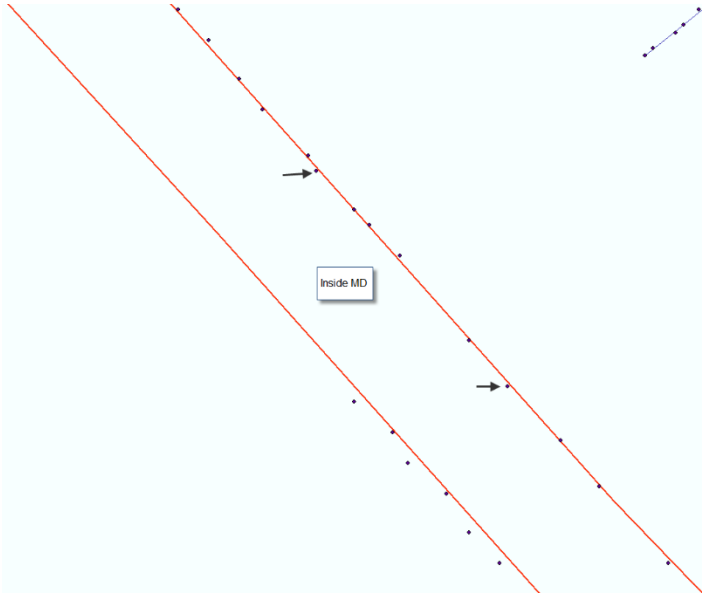
1. **Imprecise Geocoding**: Geocoders often interpolate points along streets without precise knowledge of entrances or building outlines.
2. **Outdated or Missing Data**: Geospatial data may lag behind real-world changes.
3. **Shared or Vague Addresses**: Malls, complexes, and campuses often share a single address for many POIs.
4. **Map Misalignment**: Building footprints and road geometry may not reflect real-world positions.
5. **Digitized Road Complexity**: In multi-digitized road setups, addresses can end up between lanes or on the wrong side.

Sample Scenario: Misplaced Point Addresses in Multi-Digitized Roads

A common real-world manifestation of this problem occurs with **Point Addresses (PAs)** on **multi-digitized roads**, where each direction of travel is represented by its own parallel line. In such cases, PAs often appear incorrectly placed between the two lanes or directly on the roadway instead of near the actual building or entrance.

For example:

- A user looking for "45 Maple Street" is directed to a spot between the northbound and southbound lanes of a dual carriageway—nowhere near the actual building entrance.
- This misplacement stems from the PA being snapped to the road centerline instead of being correctly positioned on the side associated with the house or building.



This scenario mirrors the broader challenge of POI accuracy—where **location errors are not just geocoding issues, but perception problems**, stemming from a digital map's inability to correctly interpret and represent the physical world.

How Accurate Do POIs Need to Be?

- **Good enough** for traditional 2D mapping and navigation apps.
- **Crucial** for AR/XR experiences (e.g. smart glasses), where precision matters at the building or entrance level.

Proposed Solution: Visual and Spatial Re-localization

Inspired by visual positioning systems used in AR and user localization, we propose a system that:

- **Uses images (with known camera position and orientation)** to triangulate and correct the true location of a POI or PA.

- **Employs mobile apps** where users capture intentional photos of nearby POIs or addresses, allowing visual-based re-localization.
- **Integrates spatial rules and GIS analysis** to detect and auto-correct misplaced addresses in known error-prone contexts like multi-digitized roads.
- **Incorporates machine learning models** to automate detection and relocation of misplaced points using surrounding map features, road geometry, and historical patterns.

Scalability Challenge & Future Work

While the visual method is effective in controlled cases, it currently depends on intentional image capture. Scaling this to millions of POIs and PAs will require:

- **Crowdsourced imagery** from sources like Mapillary or OpenStreetCam.
- **Gamification and user incentives** to encourage voluntary correction contributions.
- **Semi-supervised learning techniques** to generalize corrections from a limited set of verified samples.

Takeaway

Improving the spatial accuracy of POIs and PAs isn't just about better geocoding—it's about better understanding and representing the real world. By combining visual cues, user interaction, and automated spatial intelligence, we can dramatically improve map quality at scale—enabling better navigation, more precise AR applications, and smarter urban systems.

Problem statement 2:

Detecting Missing Roundabouts for Database Integration

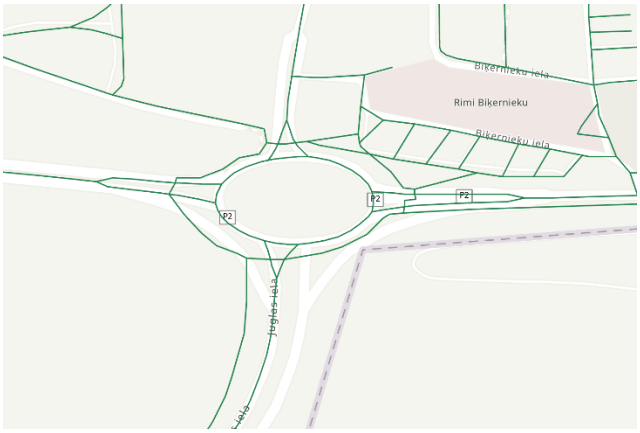
Objective

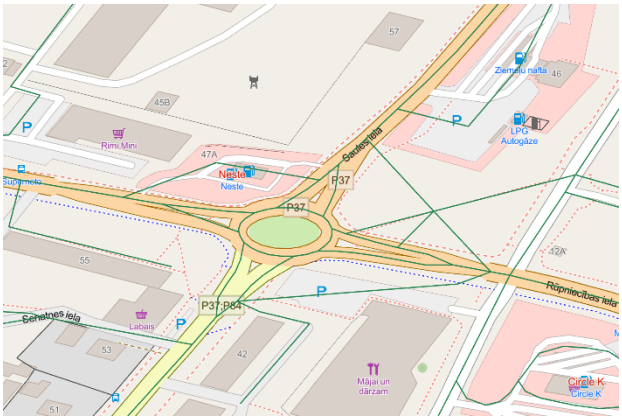
Our database includes **roundabouts**, which are circular intersections allowing traffic to flow continuously in one direction around a central island. However, some roundabouts are missing in our database, and a provided source contains roundabout geometries but lacks the attributes necessary for proper identification and integration.

Challenge

Develop a method to automatically identify roundabouts from the provided source data based on their geometric and spatial characteristics.

Illustrative Examples





Problem Statement 3:

Smart Road Merge in Java in HERE Map

Objective

Develop a robust **Java-based algorithm** to intelligently merge two road segments into one, preserving **geometric continuity**, **topological correctness**, and **semantic attributes**. Your implementation should handle **all real-world edge cases** (e.g., at HERE Technologies).

Map data often comes from multiple sources or partitions. Two road segments may represent the same real-world road but differ in shape, metadata, or connections. A well-merged road ensures better routing, display, and data consistency.

Input Format – Develop java interface which should take input as geojson road object.

Output – The output should be merged road object.

Technology

- Java 11+ preferred not restricted

Suggested Libraries

- Geometry:
  - JTS Topology Suite – for spatial operations (union, distance, snapping, etc.)
- JSON Serialization:
  - Jackson or Gson
- Logging:
  - SLF4J with Logback
- Testing:
  - JUnit 5

Optional (Bonus)

- Use GeoTools for extended GIS functions.
- Use Spring Boot for RESTful visualization/debug API.
- Can use QGIS, mapcreator.here.com

Sample Geojson input


```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "id": "road1",
        "name": "Main St.",
        "speed_limit": "50",
        "lanes": "2",
        "country": "DE"
      },
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [8.682127, 50.110924],
          [8.682557, 50.111224],
          [8.682987, 50.111524]
        ]
      }
    },
    {
```

```
      "type": "Feature",
      "properties": {
        "id": "road2",
        "name": "Main Street",
        "speed_limit": "60",
        "lanes": "2",
        "country": "DE"
      },
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [8.683007, 50.111525],
          [8.683437, 50.111825],
          [8.683867, 50.112125]
        ]
      }
    }
  ]
}
```

Problem statement 4:

Develop a desktop-based GIS tool where you could upload different GIS object perform GIS operation and visualize it.

Objective

Develop a desktop GIS tool using Java that allows users to:

1. Upload and manage GIS objects (e.g., roads, buildings, boundaries in GeoJSON, Shapefile, or WKT).
2. Perform core GIS operations such as union, intersection, buffer, difference, spatial join.
3. Visualize results interactively on a map canvas.

Core Features

1. File Upload Support

- Upload one or more GIS datasets (support .geojson, .shp, or .wkt).
- Parse files and display metadata.

2. GIS Operations

Use spatial algorithms on selected layers:

- **Union**
- **Intersection**
- **Difference**
- **Buffer**
- **Point-in-Polygon / Spatial Join**

Operations should support:

- Layer-on-layer operations (e.g., roads vs zones).
- Attribute-based filtering (optional).

3. Visualization

- Render uploaded layers on a simple 2D canvas.
- Visual differentiation (colors, layer names).
- Zoom & pan.
- Highlight results of GIS operations.
- Optionally export results to GeoJSON/WKT.

Suggested UI Layout

Panel	Description
Left Panel	Layer list + operations menu (add/remove/select layer)
Top Toolbar	File upload, Export, Operation buttons
Main Canvas	Map display with zoom, pan, highlight
Bottom Panel	Log output / operation results

Sample GIS Scenarios

Scenario	Operation
Merge adjacent land parcels	Union
Find roads crossing flood zones	Intersection
Show hospitals outside city bounds	Difference
Buffer around rivers	Buffer
Count POIs inside city polygons	Spatial Join

Problem Statement 5:

AI-Driven Map Making from Raw Data to Finished Map

Objective

Modern digital maps rely on a complex pipeline that transforms raw geospatial data into structured, routable, and semantically rich map products. Traditionally, this involves multiple steps such as data ingestion, validation, enrichment, feature extraction, topology building, and map rendering. Manual curation and rule-based systems dominate this workflow, which limits scalability and adaptability.

With the rise of **Generative AI**, there is a transformative opportunity to rethink map making—automating and optimizing each step using foundation models, computer vision, natural language processing, and synthetic data generation.

The Challenge:

Design and prototype an AI-first pipeline that takes **raw map data** (e.g., satellite imagery, Lidar, GPS traces, traffic camera logs, or open-source data) and transforms it into **high-quality, navigable maps** using **Generative AI** techniques.

Your solution should address **some or all of these core map-making steps**:

1. **Data Ingestion & Preprocessing**
    - Handle raw data from heterogeneous sources (imagery, logs, traces, sensors).
    - Clean, align, and normalize datasets for downstream use.
2. **Feature Detection & Extraction**
    - Use generative models (e.g., image-to-text, vision transformers) to extract map features like roads, buildings, signs, and POIs from imagery.
3. **Semantic Enrichment & Classification**
    - Use NLP or graph AI to add meaningful labels, tags, and classifications to map features (e.g., type of road, traffic rules).
4. **Topology Construction**
    - Generate road and lane graphs, connectivity, and turn restrictions using learned spatial reasoning models.
5. **Validation & Repair**
    - Propose techniques to automatically detect inconsistencies and suggest repairs using AI (e.g., hallucination detection, anomaly scoring, few-shot learning).
6. **Map Generation & Visualization**
    - Convert vectorized data into renderable map tiles or simulate a real-world navigation experience with synthetic data generation.
7. **Feedback Loops & Continuous Learning**
    - Design self-improving feedback mechanisms using human-in-the-loop or reinforcement learning from user reports and telemetry.

Deliverables:

- A working prototype, notebook, or pipeline that demonstrates the use of Generative AI in at least one or more of the above steps.
- Clear explanation of architecture, models used, and design choices.
- A demo video (optional) and a short presentation/pitch explaining how your solution could scale in a real-world map-making platform.

Tools & Datasets Suggestions:

Any tech stack, No code should be from internet of chat-gpt.
