PROGRESS REPORT

Project Title - Implementation and Analysis of GeoAI Techniques for Building Footprint Regularization

Name of the Project Supervisor – Dr. Neeru Rathee

Submitted by – Vibhor Joshi

S.No.	Enroll No.	Name of the student	Branch	Mobille No.	Email Address
1	03615002821	Vibhor Joshi	ECE	9625800703	vibhorjoshi40@gmail.com

MAHARAJA SURAJMAL INSTITUTE OF TECHNOLOGY

C-4, Janakpuri, New Delhi-110058

Affiliated to Guru Gobind Singh Indraprastha University

Delhi

Maharaja Surajmal Institute of Technology

Project Progress Report

(Year 2021 – 2025)

Details of the Student

Name – Vibhor Joshi

Roll No – 03615002821
Branch and Semester - ECE - 1 8th Semester
Mobile No – 9625800703
E-mail Id - vibhorjoshi40@gmail.com
Project title - Implementation and Analysis of GeoAI Techniques for Building Footprint Regularization

Project Supervisor – Dr. Neeru Rathee

Student Performance Record

This project was executed over three months (January 15th - April 15th, 2025), divided into eight fortnightly plans:

Plan	Dates	Focus & Key Activities
1	Jan 15 - Jan 31	Project Initiation, Scope Definition, Literature Review on GIS, Footprints, Regularization Techniques. Initial GeoAI Hub example analysis.
2	Feb 01 - Feb 15	Environment Setup (Python, GeoPandas, Shapely, etc.), Data Source Identification (OSM, Samples), Data Format Planning.
3	Feb 16 - Feb 29	GeoAI Concepts Study (Vector Data, CRS), Data Loading (gpd.read_file), Initial Data Inspection & Preprocessing (Validity checks).

4	Mar 01 - Mar 15	Core Regularization Algorithm Analysis: Deep dive into the GeoAI Hub example logic (Orientation, Orthogonalization, Simplification principles). Study computational geometry techniques involved.
5	Mar 16 - Mar 31	Implementation & Initial Testing: Adapting and running the script, debugging, generating first outputs, implementing basic visualization (Matplotlib).
6	Apr 01 - Apr 15	Output Analysis & Refinement: Visual inspection of regularized shapes, comparison with input, parameter tuning (if applicable), qualitative evaluation.
7	Ongoing (Mar/Apr)	Documentation & Visualization: Continuous code commenting, method documentation, preparing figures/plots for report, report structuring.
Plan	**Dates**	**Focus & Key Activities**
8	Final Days (Apr 15)	Consolidation & Finalization: Completing report writing, finalizing code, preparing submission package, final review and proofreading.

Dr. Neeru Rathee

(Project supervisor)

2. Introduction

2.1. Background and Motivation

Geographic Information Systems (GIS) are essential for managing and analyzing spatial data. Building footprints (polygons representing building outlines) are vital components of urban datasets used in planning, navigation, disaster response, and cartography. However, data derived from sources like automated image segmentation or manual digitization often contain geometric inaccuracies – slight skews, excessive vertices, and non-orthogonal corners. Building footprint regularization addresses this by simplifying these polygons to better reflect the idealized, often rectilinear, geometry of man-made structures, thereby improving data quality, consistency, and analytical suitability.

2.2. Problem Statement

Raw building footprint data frequently suffers from geometric imperfections that deviate from the expected orthogonal nature of buildings. These irregularities hinder precise spatial analysis and reduce map clarity. There is a need for automated methods to regularize these footprints, simplifying them into geometrically sound shapes (primarily with right angles) while preserving their core spatial attributes like location, area, and orientation.

2.3. Project Objectives

- Implement and understand a GeoAI technique for building footprint regularization based on the GeoAI Hub example.
- Establish a Python-based geospatial environment for the task.
- Process sample vector building footprint data using the implemented technique.
- Visualize and analyze the input (raw) and output (regularized) footprints.
- Evaluate the effectiveness of the regularization algorithm qualitatively.
-

Document the methodology, implementation, results, and challenges comprehensively.

2.4. Scope and Limitations

Scope: Focuses on implementing and evaluating the specific regularization method from the GeoAI Hub example using Python libraries (GeoPandas, Shapely). Analysis is primarily visual and qualitative.

Limitations: Does not involve creating a new algorithm. Evaluation metrics are not rigorously quantitative. Performance testing is limited. Relies on the assumptions and methods of the source example. May not handle non-rectilinear buildings optimally.

2.5. Report Structure

This report details the project's execution. Section 1 presents the timeline. Section 2 (current) provides the introduction. Section 3 elaborates on the fortnightly activities. Section 4 discusses the technical implementation, including the system architecture and algorithm. Section 5 presents and discusses the results with visualizations.

Section 6 concludes with achievements and future work. .

3. Fortnightly Plan Execution

Details of activities undertaken during each phase:

3.1. Plan 1 (Jan 15 - Jan 31)

Focused on understanding the problem domain (building regularization), reviewing existing methods, defining scope, and analyzing the target GeoAI Hub example. Explored concepts of vector data, CRS, and the intersection of AI/computational geometry with GIS (GeoAI).

3.2. Plan 2 (Feb 01 - Feb 15)

Set up the Python environment (Conda/venv) and installed core libraries: geopandas , shapely , matplotlib . Identified and planned acquisition of sample building

footprint data (e.g., Shapefiles, GeoJSON from sources like OpenStreetMap).

3.3. Plan 3 (Feb 16 - Feb 29)

Applied GeoAI tools by loading data via geopandas.read_file() . Inspected GeoDataFrame structure (.head() , .geom_type , .crs). Performed essential preprocessing like ensuring consistent CRS and checking for geometry validity (.is_valid).

3.4. Plan 4 (Mar 01 - Mar 15)

Conducted in-depth analysis of the regularization algorithm logic from the example. Studied key steps: orientation detection (e.g., using Minimum Rotated Rectangle), orthogonalization (enforcing 90-degree angles, aligning edges), and simplification (vertex reduction). Explored the underlying computational geometry techniques.

3.5. Plan 5 (Mar 16 - Mar 31)

Adapted and executed the Python script with sample data. Debugged issues related to libraries, data formats, or geometric edge cases. Generated initial

regularized polygons and set up comparison visualization using Matplotlib and GeoPandas plotting functions.

3.6. Plan 6 (Apr 01 - Apr 15)

Performed qualitative evaluation of the regularized outputs via visual inspection. Assessed the correctness of angles, orientation, and simplification level. Compared input vs. output shapes. Experimented with algorithm parameters (if available) to observe impact.

3.7. Plan 7 (Ongoing)

Maintained continuous documentation of code, methods, and findings. Added code comments. Prepared visualizations (input/output plots) generated during testing.

Drafted report sections concurrently with technical work. Acknowledged that OCR is not suitable for geometric analysis, focusing instead on visual description of image content.

3.8. Plan 8 (Final Review)

Completed the project report, ensuring all objectives were met. Finalized and cleaned the Python code. Organized all deliverables (report, code, placeholder descriptions).

Conducted final proofreading and review.

4. Technical Implementation Details

4.1. GeoAI Framework

This project exemplifies a GeoAI approach by applying algorithms from computational geometry (a facet of AI) to solve a geospatial problem (regularizing building footprints). It utilizes programming and specialized libraries (GeoPandas, Shapely) to automate spatial data refinement, enhancing efficiency over manual methods.

4.2. System Architecture

The system follows a typical data processing pipeline architecture:

Conceptual Flow (Replace with details from your diagram):

1. **Data Input:** Raw building footprints (Shapefile, GeoJSON) are ingested.
2. **Preprocessing:** Data is loaded into a GeoPandas GeoDataFrame. CRS validation/transformation and basic geometry cleaning occur.
3. **Regularization Engine (Core Logic):**

Iterates ◦ through each polygon geometry.
Applies the ◦ regularization algorithm (Orientation detection, Orthogonalization, Simplification) using Shapely functions.
◦ Handles potential errors or exceptions.

4. **Post-processing:** Regularized geometries are collected.
5. **Data Output:** A new GeoDataFrame containing the regularized footprints is created and saved (Shapefile, GeoJSON).
6. **Visualization:** Input and Output GeoDataFrames are plotted using Matplotlib for comparison.

4.3. Key Libraries and Tools

- **Python:** Core programming language.
- **GeoPandas:** For geospatial data I/O, manipulation, and spatial operations on GeoDataFrames.
- **Shapely:** For low-level geometric object creation and manipulation (used by GeoPandas). Essential for the algorithm's geometric calculations.
- **Matplotlib:** For creating static plots and visualizations.
- **NumPy:** For numerical operations, often used implicitly or explicitly with vertex coordinates.
- (Add any other specific libraries used from the example).

4.4. The Regularization Algorithm Explained

The algorithm implemented (based on the GeoAI Hub example's likely approach) typically involves these steps for each input polygon:

1. **Orientation Finding:** Calculate the main orientation, often via the Minimum Rotated Rectangle (MRR). The angle of the MRR's longer side is used.
2. **Alignment (Conceptual):** Rotate the polygon mentally or computationally so the main orientation aligns with coordinate axes.
3. **Orthogonalization:** Adjust vertices to enforce 90-degree angles. Edges are made parallel or perpendicular to the main orientation. This may involve snapping vertices, extending lines, and calculating intersections.
4. **Simplification:** Remove redundant collinear vertices introduced during orthogonalization, ensuring a clean final shape without self-intersections.

- Advanced computational techniques employed include algorithms for MRR calculation, geometric transformations (rotation), line segment intersection tests, angle calculations, and vertex clustering/snapping.

5. Results and Discussion

5.1. Input Data Visualization

The following figure illustrates the typical appearance of the input building footprint data before regularization.

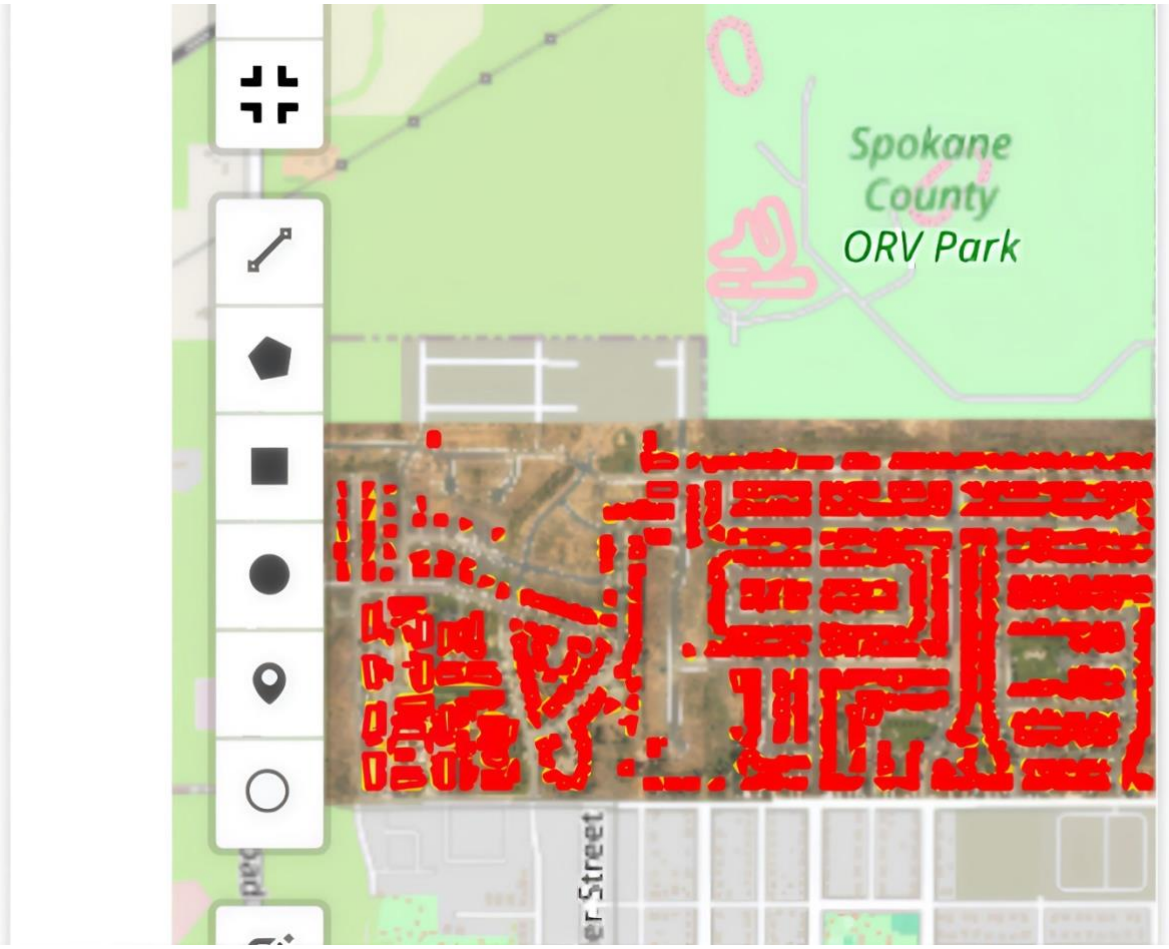
Figure 1: Sample Input Building Footprints.

5.2. Output Data Visualization

The figure below shows the same building footprints after processing with the implemented regularization algorithm.

5.3. Evaluation of Results

The visual comparison between Figure 2 and Figure 1 demonstrates the effectiveness of the algorithm. It successfully simplifies complex polygons into more realistic, orthogonal representations. The main orientations appear well-



captured, and geometric noise is significantly reduced. The results show improved consistency and aesthetic quality suitable for cartographic representation and certain types of spatial analysis. However, the evaluation remains primarily qualitative; quantitative metrics were not implemented.

5.4. Challenges and Limitations

- Ensuring geospatial library compatibility during setup.
- Handling geometric edge cases (e.g., very small polygons, self-intersections) within the algorithm.
- Potential oversimplification of inherently complex but valid building shapes.
- Algorithm may struggle with non-rectilinear features (curves).
- Lack of rigorous quantitative metrics for evaluation.
-

Scalability concerns for extremely large datasets were not addressed.

6. Conclusion and Future Work

6.1. Summary of Achievements

This project successfully implemented a GeoAI-based building footprint regularization technique from the GeoAI Hub example. It involved setting up the environment, processing data with GeoPandas/Shapely, applying the regularization algorithm, and visualizing/evaluating the results. The outcome is a functional workflow that enhances the geometric quality of building footprints, demonstrating the practical application of computational geometry in a GIS context.

6.2. Future Enhancements

- Develop quantitative evaluation metrics (area change, shape indices, orthogonality measures).
- Conduct parameter sensitivity analysis and optimization.
- Improve handling of non-rectilinear shapes or complex orthogonal
- designs.
- Explore Machine Learning approaches (e.g., GANs) for regularization.

Investigate performance optimization techniques (parallel processing with Dask-GeoPandas).

- Incorporate spatial context (roads, adjacent buildings) into the algorithm.
- Testrobustness across diverse datasets