3D Building Reconstruction from Floor Plans Using C and Classical Algorithms

Hsu Chieh Ma, Imran Aziz, Mengping Zhang
Ryan Travis Nation, Tianxiang Gao

October 3, 2025

GitHub Repository: `https://github.com/MengpingZhang/CS6010-Project-Group4`

Abstract

This project proposes a C-based program to automatically convert a 2D floor plan image into a labeled 3D building model, with the long-term goal of enabling large-scale campus energy demand modeling in EnergyPlus. Starting from a single floor plan diagram with drawn walls, room labels, windows, and doors but without explicit dimensions, the program will be designed to provide geometric measurements and construct a simplified 3D representation suitable for EnergyPlus preprocessing. The broader motivation comes from the Georgia Tech campus, which has more than 150 buildings that require 3D layouts defined in terms of enclosures, zones, and surfaces to generate IDF files and simulate heating and cooling demand under different weather conditions.

Key technical challenges to be addressed include wall length and room size estimation from images, determining scale from a known total area, and detecting symbols for doors, windows, and text labels. The proposed methodology will combine image processing, computational geometry, and graph algorithms in C to segment the floor plan into rooms and walls, compute dimensions via polygonal shape fitting, and extrude the layout into a 3D mesh with proper proportions. Each room's volume will be labeled with its name, and window and door openings will be placed according to the plan. A post-processing step will explore Stable Diffusion based visualization through standard Python libraries to apply realistic textures or context to the raw 3D geometry.

In summary, the project will demonstrate how classical algorithms and data structures in C (with optional use of parallel processing) can be applied to automate the floor plan to 3D model conversion. This approach will provide the basis for a scalable workflow to support campus wide energy modeling and architectural simulation, reducing the current reliance on labor intensive CAD model extraction.

1 Description of a System Being Studied

Background

The Georgia Tech campus contains more than 150 buildings, and to perform energy demand modeling of these buildings in EnergyPlus, a 3D representation of each building is required. Figure 1 shows the Global Information System (GIS) map of the campus with the building footprints and their 2D layouts. While detailed 3D CAD models exist for some buildings, EnergyPlus cannot directly import such models. Instead, the software requires simplified 3D layouts defined in terms of enclosures, zones, and surfaces, along with their internal connections. These serve as the preprocessing step for generating the IDF file, which is then run under specified boundary conditions to compute heating and cooling demand for different weather scenarios.

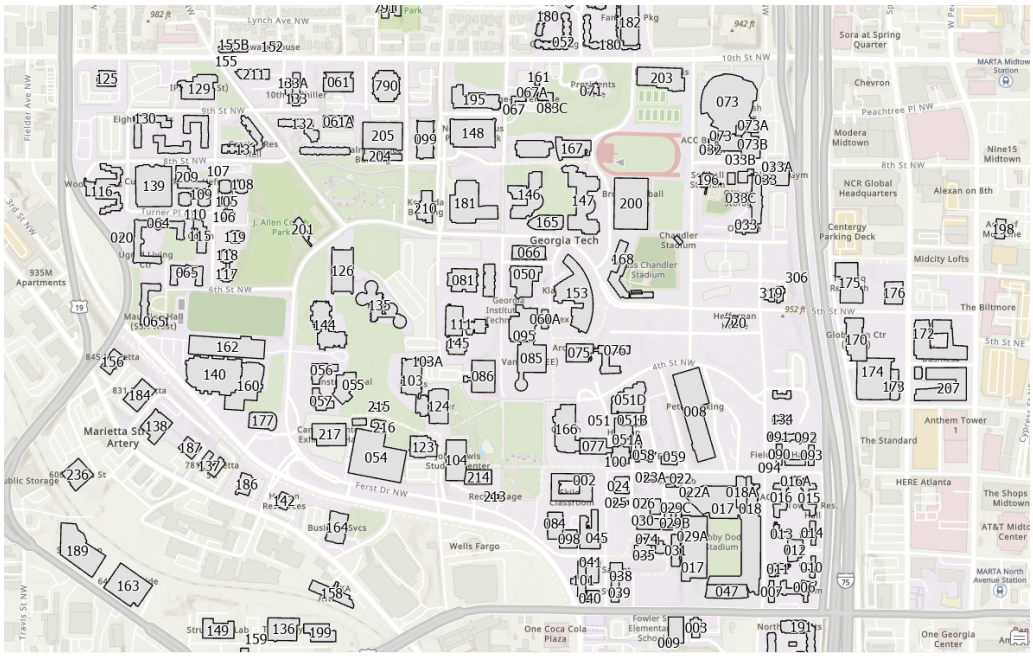

Figure 1: Georgia Tech campus map on ArcGIS showing building outlines

Currently, extracting the required information from CAD files involves manually opening each model and recording all necessary dimensions. This process is highly labor-intensive, and in many cases, especially for older buildings, detailed CAD models do not exist at all. This creates a clear need for a computational tool that can perform a form of reverse engineering, starting from available GIS maps or simple floorplan data as shown in Figure 2. The program should infer detailed inner 2D layouts with dimensions and then construct a simplified 3D model suitable for EnergyPlus preprocessing. The future goal is to be able to extract simple 3D models for complicated floor plans such as shown in Figure 3.

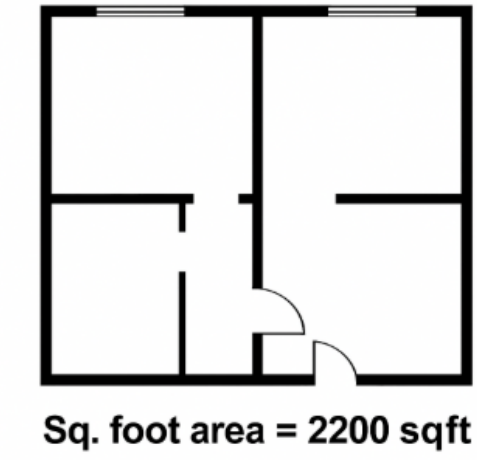

Figure 2: Floor plan of a 2200 sqft house with no labels

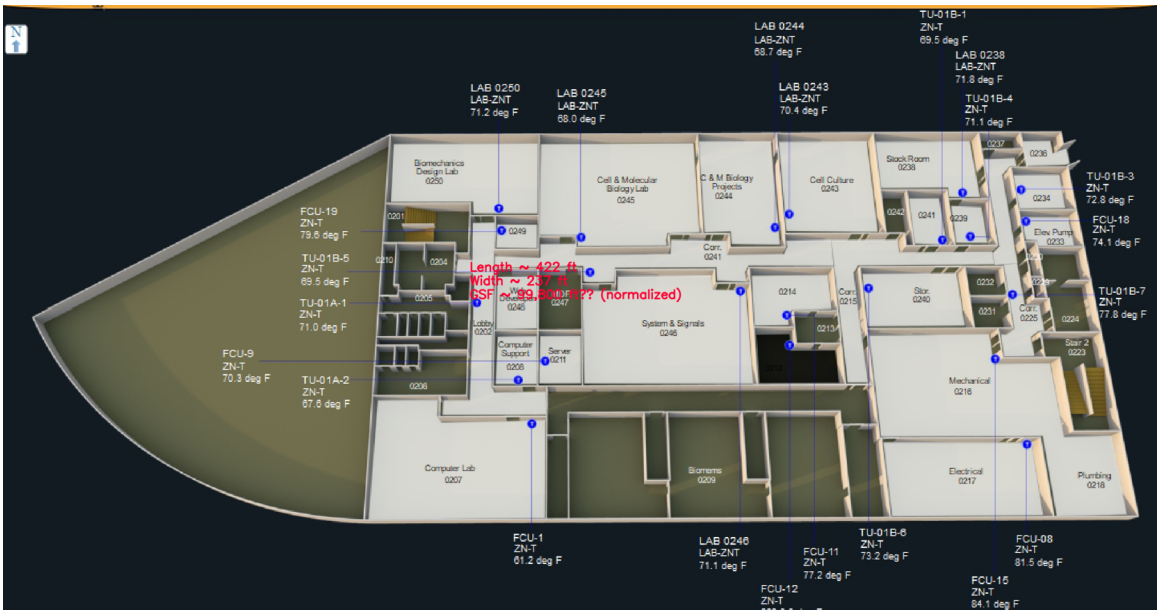

Figure 3: Floor Plan of high energy intensive Georgia Tech research building

Our project, as introduced in the abstract, represents an initial step toward developing such a C-based program that can automate this workflow and support large scale campus energy modeling.

The system under study will process a single top-down floor plan image and reconstruct a labeled 3D building model. The floor plan is assumed to contain standard architectural linework (walls as parallel lines, enclosed spaces for rooms, graphical symbols for doors and windows, and text labels for room names or codes) but may not include explicit numeric dimensions for individual rooms. External metadata provides the total interior floor area (e.g., 2200 sq ft), which will serve as the calibration basis for converting pixel distances to real-world dimensions.

Inputs

- A single image file of a 2D architectural floor plan (PNG or PGM).
- The known total interior area of the building.
- Optional metadata such as outer dimensions if available.

Unknowns and Requirements

From this input, the system will deduce geometric and semantic information that an architect could infer manually with a scale ruler. Specifically, the software will:

- Extract room polygons and compute their boundaries.
- Infer wall lengths, wall thickness, and room dimensions by calibrating pixel distances against the known total area.
- Identify adjacencies between rooms, i.e., which walls are shared.
- Detect and interpret openings, doors and windows from architectural symbols.
- Associate text labels with the correct room polygons via optical character recognition (OCR) and centroid in polygon matching.

Outputs

The system will produce a 3D model by extruding the 2D floor plan into vertical geometry. Each room will be represented as a labeled prism with walls, floors, and ceilings. Openings will appear as voids or objects in walls. The model will be exportable to:

- Standard geometry format (OBJ) for visualization.
- Geometry will be enhanced through Stable Diffusion in image-to-image mode, adding textures, lighting, and aesthetic realism for presentation purposes.

Scope and Assumptions

- The work focuses on single-floor buildings. Multi-story extensions are left for future development.
- Input images are assumed to be relatively clean digital floor plans or clear scans (no perspective distortion).
- All relevant architectural features (walls, doors, windows, labels) are present and recognizable.
- The total interior area provided is accurate and corresponds to the sum of enclosed rooms.

This description establishes the problem space for building a pipeline that can take a raw floor plan image, infer geometric and semantic structure, and deliver both simulation-ready models and presentation-quality visualizations.

2 Conceptual Model of the System

The proposed system will consist of a pipeline that transforms 2D architectural floor plan images into 3D building models suitable for energy simulation and visualization. The conceptual workflow is outlined below:

1. **Preprocessing and Edge Extraction**
   Floor-plan drawings will be converted to a clean black-and-white representation (thresholding and light morphology). From this, the system will derive an edge/line image for downstream steps. When needed for visuals, a depth image will be rendered from the reconstructed geometry (not assumed in the input) to condition Stable Diffusion.
2. **Segmentation and Region Identification**
   Image segmentation methods (e.g., connected component labeling) will be used to distinguish individual spaces such as rooms and circulation areas. Each region will be analyzed to infer spatial boundaries.
3. **Scale Calibration**
   The system will estimate a global or local scale factor based on known dimensions (e.g., annotated room areas or reference markers) so that generated geometry aligns with real-world units.
4. **Polygon Fitting for Geometry Reconstruction**
   Each segmented region will be traced to a polygon and then will be regularized: snapping nearly horizontal or vertical edges, merging nearly collinear segments, and removing small artifacts. Where helpful, simple shapes such as rectangles or other orthogonalized polygon will be used to approximate rooms while preserving total area within tolerance.
5. **Data Structures and Metadata Integration**
   Geometry and semantics information will be stored in explicit C data structures such as arrays/lists for vertices and faces, queues for BFS, and a graph (adjacency lists) for room-to-room connections. Room labels will be associated via Optical Character Recognition (OCR) and centroid-in-polygon checks.
6. **Openings and Connectivity**
   Windows and doors will be identified as gaps along wall segments using shape/edge heuristics and integrated into wall geometries. Connectivity between adjacent spaces will be established by detecting shared boundaries.
7. **3D Model Generation**
   The resulting polygons, openings, and metadata will be extruded into 3D geometry. The 3D model will be written as Wavefront OBJ (OBJ) with groups that preserve room identity and surface types.
8. **Visualization and Aesthetic Enhancement**
   For user-friendly visualization, the edge/line image derived from the floor plan will be processed with generative AI tools to add textures, colorization, and design aesthetics.

Key Challenges to be Addressed

- Handling noisy or low-resolution floor plan drawings. Inferring missing dimensional information when explicit labels are absent.
- Choosing optimal geometric simplifications (rectangles vs. hexagons) to balance accuracy and computational efficiency.

- Robust OCR for handwritten or ambiguous annotations.
- Image scans and region passes will be optimized in C, with optional Open Multi-Processing (OpenMP) for safe loops.
- Stable diffusion implementation via PyTorch.

Expected Demonstrations

- A proof-of-concept pipeline that can process sample floor plans into 3D models.
- Example visualizations showing floor plans converted into colored, aesthetic renderings using generative AI.

3 Platforms / Modalities of Development

Our goal is to develop a reliable, cross-platform workflow that transforms a single 2D floor-plan image into a labeled 3D model with high-quality visualizations. The system will be built using standard, maintainable tools so the team can iterate quickly and extend the platform in the future.

The implementation will be written in C, compiled with the GNU Compiler Collection (GCC) following the C11 standard, and built cross-platform using CMake. Input formats will include Portable Graymap (PGM) and PNG images, while the 3D output will be exported as Wavefront OBJ (OBJ) files. For visualization, we will generate edge maps, line drawings, or depth images from the reconstructed geometry, which will then be processed with Stable Diffusion in image-to-image mode. By conditioning on these structural cues, Stable Diffusion can synthesize realistic textures, materials, and lighting while preserving the geometric layout. This will provide colorization, aesthetic enhancement, and contextual realism, making the raw 3D output visually consistent with architectural renderings.

Collaboration will be managed with Git, and every change will trigger Continuous Integration (CI) to compile and run a minimal end-to-end test. We will maintain a clean single-threaded baseline, with optional use of OpenMP for loop-level parallelism where performance gains are safe and measurable.

We propose a six-to-eight-week project plan using this development setup:

1. Set up repository and CI: establish minimal I/O and automated build/test checks.
2. Floor-plan preprocessing: clean the input image and separate interior rooms from exterior boundaries using Breadth-First Search (BFS).
3. Geometry extraction: trace contours into polygons, snap edges, and compute areas.
4. Scaling and extrusion: derive a global scale from known total area, apply it, extrude walls, floors, and ceilings, and export grouped OBJ files by room.
5. Doors, windows, and adjacency: detect gaps along shared walls, place openings, compute room-to-room adjacency, tighten tolerances, and remove duplicate vertices.
6. Labeling and visualization: attach room labels, generate visuals from edges/lines or depth cues, and apply Stable Diffusion with PyTorch for texture synthesis and rendering.
7. Performance and documentation: enable optional OpenMP parallelism, write clear documentation, and prepare a short demonstration of the end-to-end pipeline.

This workflow will allow our five-person team to focus on the central goal: turning real-world floor plans into dimensioned, labeled 3D models with visually compelling renderings. The resulting platform will not only support the current proof-of-concept but will also provide a solid foundation for future extensions such as energy demand modeling and enhanced visualization capabilities.

4 Literature Review

This project builds on two main types of previous work. First, there are classical "geometry-first" methods for understanding floor plans. These methods clean up scanned plans, separate text from graphics, detect walls and symbols, and then form closed room shapes that can be turned into 3D models. They are easy to understand and fast, but they can fail if the lines, scan quality, or drawing style changes, which can lead to broken walls or missing symbols that need fixing afterwards [1, 2, 6]. Second, there are learning-based methods

that use neural networks to segment walls and detect doors and windows more reliably across different styles. Some of these methods also combine AI outputs with rules or optimization to make usable 3D geometry. They usually work better across different floor plans, but they still need extra steps like tracing room polygons, fixing topology, and placing openings before the models are ready for simulation. They also require labeled data and training [3, 4, 6, 7].

In both approaches, the key step to go from 2D to 3D is creating solid room shapes that match the real layout. Once rooms are watertight and doors and windows are correctly placed, walls, floors, and ceilings can be generated. Doors connect exactly two rooms, windows connect a room to the outside, and shared walls line up properly. This gives a clean 3D model that can be used for visualization or energy analysis [1, 6]. Recently, AI methods like diffusion models can generate very realistic images from floor plans. However, these models can make up details that aren't really there, so they are best used for visuals rather than for the actual 3D geometry [5]. In short, the research shows that creating reliable 3D models from floor plans still depends on having correct 2D layouts and consistent scaling. Generative AI is mainly useful for making visuals, not for building accurate geometry. This is the approach our project will follow.

Novelty and Fit of Our Approach

Our pipeline introduces a student-friendly, C-based implementation that can infer real-world scale without needing any per room measurements. We do this by matching the sum of room areas in pixels to a single known total interior area, then applying that global scale throughout the reconstruction. We build on classical data structures and algorithms. First, we clean the floor plan using binarization and light morphology on 2D arrays. Next, we use Breadth-First Search (BFS) to label interior regions and trace contours, snapping them together to form watertight polygons. The shoelace formula ensures that the room areas remain consistent, and we represent room relationships with an explicit adjacency graph. Once the rooms are defined, we extrude them into a grouped Wavefront OBJ file, inserting doors and windows as context-aware gaps, i.e., doors connect two rooms, and windows connect a room to the outside. This approach gives us control over measurement accuracy while still producing visually compelling models. We can render edge/line or depth images from the reconstructed geometry and feed them into Stable Diffusion for textures and lighting, without altering the underlying metric 3D model.

Our two main points of novelty are:

1. A simple, auditable global area-based scaling method.
2. A fully transparent, end-to-end C implementation that emphasizes queues, lists, graphs, and optional OpenMP, making the pipeline easy to understand, test, and adapt for future tasks like energy-modeling exports.

References

[1] Ahmed, S., H. Li, and F. Deng. 2018. "Automatic Reconstruction of 3D BIM Models from 2D Indoor Floor Plans." *Automation in Construction* 88: 48–59.

[2] De las Heras, L.P., E. Valveny, et al. 2014. "Statistical Segmentation and Structural Recognition for Floor Plan Interpretation." *International Journal on Document Analysis and Recognition* 17(3): 221–237.

[3] Dodge, S., J. Xu, and B. Stenger. 2017. "Parsing Floor Plan Images." In *Proceedings of the 15th IAPR International Conference on Machine Vision Applications (MVA).*

[4] Lv, X., S. Zhao, X. Yu, and B. Zhao. 2021. "Residential Floor Plan Recognition and Reconstruction." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

[5] Nguyen, H.T., et al. 2024. "HouseCrafter: Lifting Floorplans to 3D Scenes with 2D Diffusion Model." *arXiv preprint* arXiv:2406.20077.

[6] Pizarro, P.N., N. Hitschfeld-Kahler, I. Sipiran, and J.M. Saavedra. 2022. "Automatic Floor Plan Analysis and Recognition: A Survey." *Automation in Construction* 140: 104348.

[7] Zhang, Y., et al. 2022. "A High-Precision Method for Segmentation and Recognition of Shopping Mall Floor Plans." *Sensors* 22(7): 2510. (MDPI)