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

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

GitHub repository

1 Introduction

Our project aims to reconstruct 3D building models from 2D architectural floor plans using classical algorithms implemented in C. The overall goal is to establish a pipeline that extracts structural information from floor plan images and generates dimensioned 3D geometries suitable for energy modeling and visualization. In this checkpoint, we present the current progress and development plan of our project.

In this early stage, our primary focus has been implementing and testing the image segmentation module. We first prototyped segmentation algorithms in Python. Following this proof of concept, we developed C libraries using fundamental data structures and algorithm knowledge, which will later be utilized throughout the project.

2 Show of Progress

2.1 Algorithm Overview

The floor plan segmentation program identifies architectural elements (rooms, windows, doors) from scanned floor plans using image processing techniques. The core algorithm consists of three stages: binary segmentation via Otsu thresholding to separate walls from free space, connected component analysis to identify rooms and windows based on geometric properties, and RANSAC-based circle fitting to detect curved door arcs in wall edges.

2.2 Python Implement

2.2.1 Environment Python 3.11 with NumPy 1.24 and Pillow 10.0 in Jupyter Notebook environment.

2.2.2 Code Analysis The Python prototype leverages NumPy's vectorized operations for efficient image processing. Key functions include ***label_components()*** using BFS with deque for connected component labeling, ***otsu_thresh()*** for automatic threshold selection, and the RANSAC implementation performing 4000 iterations to robustly fit circles to edge pixels. The code processes a 800×600 floor plan in approximately 1.2 seconds.

2.3 C Implement

2.3.1 Environments C17 standard compiled with MSVC on Windows, using only standard library without external dependencies.

2.3.2 Analysis The C version implements all algorithms from scratch, including PGM/PPM file I/O, manual memory management with malloc/free, and custom queue implementation using arrays for BFS. The RANSAC algorithm uses deterministic random sampling with fixed seed for reproducibility. Critical optimizations include single-pass erosion and reusable memory buffers for component labeling.

2.4 Comparison

Both two scripts are developed to execute the same core task: the identification and enumeration of facility settings from a 2D floor plan image. They share an identical algorithmic foundation—employing Otsu's thresholding for segmentation, connected component labeling (via BFS) for region identification, and a RANSAC-based circle-fitting algorithm for door arc detection.

2.4.1 Advantages The Python script's advantages lie in rapid prototyping, high-level readability, and flexible image I/O via libraries like NumPy and Pillow. The C implementation provides the project's core benefits: exceptional computational efficiency and true portability as a standalone executable, achieving with zero external dependencies, relying only on the C standard library as required by the project scope.

2.4.2 Disadvantages The primary disadvantages from Python are the slower, interpreter-based performance and its reliance on external libraries like NumPy and Pillow; while C implementation is significantly more complex, requiring manual memory management and more difficult debugging. This C version is also less flexible, as its hard-coded I/O is restricted to PGM/PPM formats to maintain its zero-dependency status.

2.5 Difficulties

The main difficulties that have been identified in our project, involving to realize our ideas into programming with C and Python. These tasks will be spearheaded by I. Aziz and M. Zhang. This task will be completed within approximately two weeks. After the code has been implemented, our team will work on data structures, image segmentation, algorithm development, and 3D extrusion. We expect this process to be the most difficult, as it has the most potential for bugs and other issues. Once our code is working, image processing will begin. Finally, the team will determine the division of labor for the final paper and presentation and then complete it.

3 Literature Reviews

This paper[1] develops a pipeline for separating texts from graphical content in scanned floor plans. The key innovation is to first remove external walls to prevent thick lines from being misclassified as text. It then uses connected component analysis to distinguish text elements (even when characters overlap with lines). Finally, it recovers missing or touching characters and cleans up large non-text labels using statistical thresholds. The method used in the paper prevents OCR labels from contaminating the polygon segmentation and wall tracing process.

The system[2] starts with preprocessed floor plans (isolate lines and regions), then builds a graph where each node represents a background-connected component and edges represent adjacency. Each node is described using shape descriptors and objects are recognized by matching against prototypes. The method might be relevant to the connectivity phase, as it treats rooms or symbols as white connected components and builds a graph of their relationships.

This paper by Zhang and Zhang[6] investigates the efficiency of various modeling approaches. Rather than utilizing existing machine learning-based approaches, they aimed to implement a solution that is more efficient in rendering 3D models from 2D raster floor plans. Their approach utilized similarity maps and clustering models to prompt the "Segment Anything Model." This approach demonstrated success and was effective in producing models with minimal data requirements. While their approach is different than ours, their research goal is similar. Our project aims to create another rendering model that will help create 3D models in a more efficient way. Their paper also makes the case for why this research is important, further supporting our own novelty and necessity claims.

By using an Application Domain Extension (ADE)[4], these researchers aim to help preserve otherwise forgotten data from building information models (BIM). Konde et al. then use newly recovered data to inform development of CityGML 3.0. In doing so, they were able to provide one of the first datasets that is consistent with ongoing development. This paper also supports our notion that more accessible data is important. Our 3D models will help make building information more available than previously. Additionally, our models can also be used in other settings, such as CityGML 3.0.

The paper[3] introduces a method to convert scanned 2D floor plans into semantic, vectorized 3D building models. It emphasizes dataset generalizability and automated extraction of room shapes and relationships. The pipeline achieves strong results on public benchmarks and demonstrates scalability for varied house layouts. Semantic vectorization and robust segmentation techniques help automate floor plan understanding.

The work[5] proposes automated conversion of 3D laser scanner data into 2D floor plans through geometric extraction. It identifies walls, floors, ceilings, and columns from point clouds, minimizing manual modeling and error. The paper details each processing stage and discusses practical deployment for building documentation and renovation. Emphasis on geometric accuracy promotes better architectural model reliability.

4 Division of work

The project responsibilities were divided across both C and Python components.

C Program Component

- **Data Structures and Core Logic:** Mengping will lead the design and implementation of data structures and program flow.
- **Image Segmentation:** Mengping and Ryan will jointly develop algorithms to distinguish individual spaces from input floor plans.
- **Dimension Extraction:** Hsu-Chieh will implement the piercing algorithm for geometric and dimensional construction.
- **3D Extrusion and Openings:** Tianxiang and Hsu-Chieh will collaborate on constructing the extrusion pipeline that transforms processed polygons and metadata into complete 3D geometries.
- **Parallelization:** Imran will implement OpenMP parallelism to enhance computational performance.

Python Program Component

- **Methodology Research:** Imran and Ryan will conduct background research to support the design of image processing and stable diffusion models.
- **Image Preprocessing and Edge Detection:** Tianxiang will implement the preprocessing pipeline to generate image input for downstream geometry modeling.
- **Stable Diffusion Visualization:** Tianxiang, Hsu-Chieh, and Mengping will implement the model.

References

[1] S. Ahmed, M. Weber, M. Liwicki, and A. Dengel. 2011. Text/graphics segmentation in architectural floor plans. In *2011 International Conference on Document Analysis and Recognition*. IEEE, 734–738. https://doi.org/10.1109/ICDAR.2011.153

[2] A. Barducci and S. Marinai. 2012. Object recognition in floor plans by graphs of white connected components. In *2012 21st International Conference on Pattern Recognition (ICPR)*. IEEE, 298–301. https://doi.org/10.1109/ICPR.2012.89

[3] A. C. Barreiro, M. Trzeciakiewicz, A. Hilsmann, and P. Eisert. 2023. Automatic reconstruction of semantic 3D models from 2D floor plans. In *2023 IEEE Conference*. https://doi.org/10.1109/ICCV51070.2023.00215746

[4] A. Konde, H. Tauscher, F. Biljecki, and J. Crawford. 2018. Floor plans in CityGML. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* IV-4/W6 (2018), 25–32. https://doi.org/10.5194/isprs-annals-IV-4-W6-25-2018

[5] B. Okorn, X. Xiong, B. Akinci, and D. Huber. 2010. Toward automated modeling of floor plans. In *3D Data Processing, Visualization, and Transmission*. Carnegie Mellon University, 33–40. https://www.ri.cmu.edu/pub_files/2010/5/2009%203DPVT%20plan%20view%20modeling%20v13%20(resubmitted).pdf

[6] Haolan Zhang and Ruichuan Zhang. 2025. Efficient accessibility analysis in floor plans using semantic segmentation. *CIB Conferences* 1, 202 (2025). https://doi.org/10.7771/3067-4883.1876