Universiteit Leiden

Master Computer Science

Region-Adaptive Vertex Reduction for Lightweight Mesh Reconstruction from Point Clouds

Name: Shiran Wu
Student ID: s3809366

Date: 15/08/2025

Specialisation: Artificial Intelligence

1st supervisor: Hazel Doughty
2nd supervisor: Daan Pelt

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Digital-twin applications require lightweight yet geometrically faithful 3D building models reconstructed from large, noisy, open-boundary point clouds. We propose a region-adaptive pipeline that (i) performs 3D surface reconstruction with multi-scale Ball-Pivoting; (ii) carries out unsupervised region growing for geometric segmentation using curvature and normal-consistency criteria; and (iii) applies a two-stage, boundary-aware QEM (Quadric Error Metrics) edge-collapse procedure to differentially decimate vertices in planar versus high-curvature regions. Compared with Marching Cubes and Poisson reconstruction on real Dutch residential point clouds, Ball-Pivoting avoids out-of-bounds faces and better preserves open boundaries. For segmentation, we compare an intensity-augmented RandLA-Net baseline with region growing and adopt the latter for its stability in extracting windows, doors, and frame regions. Region growing yields a geometrically consistent partition of building geometry—sketching window and door frames as well as regions with distinct curvature. Although it does not assign semantic labels, it has proven highly robust in our pipeline, remaining stable under measurement noise, irregular sampling, and missing data. Our boundary-aware QEM further outperforms other mesh-simplification strategies (e.g., standard QEM and vertex clustering): under the same face budget ($\approx$10,685 faces), it preserves critical edges (e.g., window/door frames), whereas standard QEM exhibits pronounced artifacts. Overall, the method achieves substantial mesh compression while maintaining architectural fidelity, facilitating efficient storage, transmission, and rendering. Limitations include hand-tuned thresholds and sensitivity to classical geometric-difference metrics; future work will investigate automatic parameter adaptation and multi-resolution mesh generation.

Contents

1 Introduction

With the rapid development of smart cities and digital twin technologies, high-precision and efficient 3D building reconstruction has become a critical foundation for urban planning, disaster management, energy simulation, and related fields. However, the structural complexity and detail diversity of urban architecture (e.g., Gothic gables and ornate windows/doors characteristic of Dutch buildings) make it challenging for traditional reconstruction methods to balance geometric accuracy with computational efficiency. Achieving lightweight 3D reconstruction while preserving architectural details (such as LOD3-level window and door ornamentation) remains an unresolved challenge.

In 3D reconstruction tasks, there are two different reconstruction approaches: one is meshing model, and the other is modeling. In research on meshing models, researchers use methods such as triangulation like ball-pivoting [1] or Poisson reconstruction [2] to reconstruct building surfaces from point cloud data, which can preserve a great deal of architectural detail. However, this reconstruction process uses a large number of data points, increasing the size of the model and thereby adding to the burden on computer memory and GPU when users browse the model or conduct research. Another type of 3D reconstruction focuses on modeling objects using as few points as possible. The goal is not to reproduce all the fine details, but to extract and represent the core structural and geometric features in a simplified and abstract manner. However, this simplification comes at a cost: it may overly smooth the surfaces of buildings or objects, thereby overlooking important architectural details such as window frames, decorative edges, or local structural variations. As a result, while this type of 3D reconstruction offers the advantage of lightweight performance, it may compromise visual fidelity and the preservation of fine details.

In the process of architectural design and construction, a commonly used digital version of traditional drawings is the BIM (Building Information Modeling) model, which includes a data-rich representation of structural details, appearance, and utilities such as electrical and plumbing systems. A key concept in this context is the Level of Detail (LOD), which categorizes the degree of detail in a building's reconstruction. The base architectural model differentiates between five consecutive levels of detail (LoD0 to LoD4), with higher numbers indicating increasingly detailed depictions of the building's contours and internal structures. The variations in architectural details corresponding to levels LOD0 to LOD4 are illustrated in Figure 1.

Figure 1: Variations in architectural details corresponding to levels LOD0 to LOD4.

From the image, it is evident that LOD0 and LOD1 are rough representations of buildings, only capable of conveying basic architectural information such as length, width, height, and footprint in city maps. In the advanced version, LOD2, the basic contours of the building are more clearly expressed, including the basic shape of the roof. At the LOD3 level, details such as windows and doors are represented in the model, providing more visual information compared to earlier levels. At the highest level, LOD4, the model includes internal structural details such as furniture and stairs; this level of detail is suitable for precise simulation analyses like light studies. Dutch urban architecture predominantly exhibits national characteristics, such as pointed gable roofs, preserving styles that span from the medieval era to modern times. In 3D reconstruction of Dutch urban buildings, the primary challenge is that Dutch architecture often features windows decorated with complex patterns and unique curved designs. The 3D reconstruction technology must be highly accurate to faithfully replicate these intricate decorative features in the digital models.

Currently, most research technologies are capable of converting urban point cloud data directly into LOD2 level 3D models. For instance, the research by Ravi Peters [3] and colleagues transformed a large-scale architectural point cloud from the Netherlands into high-quality LOD2 level 3D models. However, further extensive research is needed to create models of objects such as windows and doors in order to achieve LOD3 3D models. Therefore, creating accurate yet lightweight 3D models for applications such as digital twins remains a significant research challenge, requiring better strategies to balance geometric detail preservation with model complexity control.

As urban digitalization accelerates, there is a growing need for efficient storage, transmission and real time rendering of large scale architectural point cloud data. However, real world radar acquired building point clouds are not only enormous but also affected by noise and open boundaries, which complicate mesh reconstruction and simplification. Most existing simplification algorithms face a fundamental limitation: when applied to an entire mesh model, they often aim to reduce the number of vertices while preserving the overall geometry by evaluating the impact of each simplification step. However, under aggressive simplification, these algorithms—even with geometric awareness—tend to significantly degrade regions with pronounced geometric transitions, such as the edges between wall segments, door frames, and window boundaries. Figure 2 illustrates the extent to which excessive simplification degrades the object's geometric structure.

This paper proposes a region adaptive simplification strategy that applies aggressive reduction to planar regular surfaces to eliminate redundant vertices, while using a more conservative approach for curved and high curvature areas to preserve critical structures and details. By tailoring the level of simplification to each region's geometric characteristics, we aim to greatly reduce model size while maintaining the structural integrity and visual fidelity of the architectural mesh.

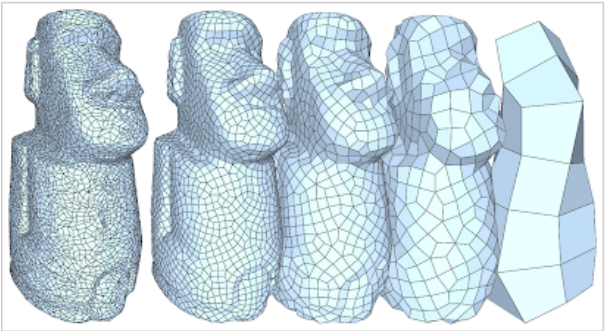


Figure 2: Impact of excessive simplification on the geometric integrity of the object, illustrating how over-aggressive decimation introduces distortions and erases critical structural details

- **To achieve efficient simplification and reconstruction of architectural point cloud models,** this study first draws on prior research in deep learning for point cloud segmentation. Based on its robust performance in outdoor architectural scenarios, RandLA-Net was selected as the primary segmentation model. Considering the characteristics of this model, we manually annotated the original architectural point cloud data to construct a dedicated dataset, and fine-tuned the parameters of RandLA-Net to improve segmentation accuracy across various architectural components.

- **After completing the initial segmentation experiments,** we conducted a systematic comparison between deep learning methods and unsupervised learning approaches for point cloud segmentation tasks. Taking into account segmentation accuracy, computational resource requirements, and the specific needs of our application scenario, an unsupervised learning method was ultimately adopted as the core algorithm for the segmentation module in this study.

- **For mesh model reconstruction,** we compared several mainstream mesh generation algorithms on outdoor architectural point cloud data and ultimately selected the Ball-Pivoting algorithm as the base method for constructing architectural mesh models. During the mesh simplification phase, in order to minimize the number of vertices and improve storage and rendering efficiency while preserving geometric accuracy, we employed a method that combines Edge Collapse and Quadric Error Metrics. Furthermore, we proposed a region-adaptive mesh vertex simplification strategy, which applies more aggressive simplification to flat areas while preserving geometric and structural details in high-curvature regions. This approach enables significant compression of mesh data while maintaining the model's geometric integrity and visual fidelity.

2 Literature Review

In this research, two technological aspects are addressed: semantic segmentation of point clouds and 3D model reconstruction of the segmented point clouds. Subsequent sections will detail the prior research pertaining to each of these tasks.

2.1 Mesh model Reconstruction

There are also many commonly used modeling methods in the field of meshing, such as Ball-Pivoting, Poisson reconstruction, Marching Cubes, and NURBS surface. Marching cubes uses a divide-and-conquer approach to locate the surface in a logical cube created from eight pixels; four each from two adjacent slices [4] . This algorithm extracts isosurfaces from 3D medical data and generates triangle meshes. It achieves high-quality modeling through linear interpolation and gradient information, and can be applied to various imaging modalities such as CT and MRI. Although Marching Cubes is effective at extracting isosurfaces with good visual accuracy, the algorithm tends to generate a large number of redundant vertices and faces during the mesh construction process. As a result, in scenarios where geometric precision and compactness are critical—such as when preserving the original structure of point cloud data—it may not perform well.

Poisson-based surface reconstruction method from oriented points, which formulates the problem globally to enhance noise robustness and avoid heuristic partitioning.[5] . Its core idea is to formulate the reconstruction problem as the solution of a Poisson equation, extracting an isosurface by solving for an implicit function. This method is well-suited for generating closed surfaces and demonstrates strong robustness against noisy or incomplete point clouds. However, it can lead to severe distortions when dealing with open surfaces and heavily relies on accurate normal information. If the normals are inconsistent or incorrectly estimated, the resulting surface may flip or the reconstruction may fail.

NURBS surfaces are composed of control points, knot vectors, and weights. They can accurately represent common geometric shapes such as circles, ellipses, and cylinders, as well as arbitrarily complex free-form surfaces. By adjusting the positions of control points and their associated weights, the shape of the surface can be flexibly modified. One major limitation of NURBS surface reconstruction is its unsuitability for outdoor point cloud environments. Due to the inherently smooth and continuous nature of NURBS representations, the method tends to over-smooth surfaces and struggles to accurately represent sharp edges or angular geometric features commonly found in architectural structures. Additionally, the lack of a well-defined parameterization and the presence of noise in outdoor point clouds further hinder its effectiveness in such scenarios.

The BPA follows the advancing-front paradigm to incrementally build an interpolating triangulation[1] . This algorithm constructs a triangle mesh by "rolling" a ball of a specified radius over a point cloud. When the ball touches three points without enclosing any others, a triangle is formed. The ball then pivots around the edges of existing triangles to find new points and gradually builds the mesh. To handle uneven point sampling, the process can be repeated with balls of different radii, resulting in an efficient and high-quality triangulated surface.

2.2 Semantic segmentation of point clouds

In the field of point cloud segmentation, the PointNet [6] , developed by Qi et al., represents the first methodology to directly learn features from raw point cloud data. The network model comprises a sequence of multiple Multi-Layer Perceptrons (MLPs), which process each point individually to learn local features. These features are then aggregated by a global pooling layer to form a comprehensive global feature representation. Building on the foundation of PointNet, Charles R. Qi et al. subsequently introduced PointNet++ [7] , which employs a multi-scale sampling strategy to enhance the model's capability to capture more detailed and precise local and global feature representations.

Unlike the scenarios where point clouds are used as direct inputs, SAMPart3D [8] , proposed by Yunhan Yang et al., achieves multi-granularity 3D part segmentation by distilling features from 2D vision models like DINOv2 into a 3D backbone network and integrating multi-view 2D segmentation masks. Similarly, in MaskClustering [9] proposed by Mi Yan et al., the framework leverages View Consensus to first capture scene information from multi-view 2D images and then aggregates the classified instances into 3D instances, thereby avoiding the need for complex segmentation tasks directly on point clouds.

In addition, RandLA-Net [10] , proposed by Qingyong Hu et al., combines KNN embedded coordinates, attention pooling, and dilated residual blocks into a novel local feature aggregation module to capture complex local structures within progressively shrinking point sets. Further more, Stratified Transformer [11] , proposed by Xin Lai et al., introduces a novel approach to learning both local and global features of point clouds by hierarchically partitioning the point cloud into grids. Then, the transformer is used to predict the category of each point for classification.

In the latest research, transformer models have also been extensively applied in point cloud segmentation techniques. Among them, representative model is the Point Cloud Transformer [12] , introduced by Meng-Hao Guo et al. in 2021, begins by capturing initial spatial features of the point cloud and dynamically focuses on key features using optimized attention and local neighborhood structures. Semantic segmentation of architectural point clouds is particularly challenging due to the need to classify together areas such as window frames and door frames,

which tend to adhere to the recognized objects. In the ASGFormer [13] released by Ting Han and colleagues, the transformer learns the similarity of features and attributes of points, and dynamically updates the weights of points in the graph. The locally learned structural information is incorporated into the attention mechanism to obtain feature neighbors and the weight of the edges. This allows parts that are structurally adherent to the object to be classified together. The techniques used in point cloud segmentation in this paper are primarily based on and reference experiments conducted with the ASGFormer.

2.3 Mesh model simplification

For mesh modeling, it is not only necessary to reconstruct an accurate 3D model from the input point cloud, but also important to minimize the number of vertices used in the mesh reconstruction process. The algorithm for reducing the number of vertices in a mesh model are edge collapse, Vertex Clustering, Quadric Error Metrics, etc.

Edge Collapse is one of the most common methods for mesh simplification. It is an algorithm that reduces the number of vertices and faces by simplifying the mesh structure. In each operation, it selects an edge and merges its two endpoints into a new vertex, while removing the triangles connected to that edge, thereby simplifying the local mesh structure. The algorithm is a form of progressive simplification, with the advantage of allowing precise control over the number of vertices after simplification. However, its drawback lies in the lack of geometric awareness during the process, which may lead to the loss of important features such as sharp edges and corner points when collapsing edges.

Vertex Clustering is a simple and efficient mesh simplification algorithm. Its core idea is to divide the 3D space into regular grids and merge all vertices that fall within the same grid cell into a single representative vertex, thereby reducing the number of vertices and faces in the mesh. This method is widely used as a fast preprocessing step for large-scale mesh or point cloud data. Its main advantage lies in its high computational speed. However, a major drawback arises when dealing with outdoor scenes, which often contain significant noise and incomplete (hole-filled) point clouds. In such cases, this algorithm can easily introduce non-manifold or invalid topologies, which can severely compromise the quality of the reconstructed 3D mesh.

Quadric Edge Collapse Decimation (QEM), proposed by Garland and Heckbert, uses the Quadric Error Metric to measure the geometric error introduced by edge collapses[14] . It is one of the most widely used mesh simplification methods today. Using quadric error metrics to track geometric error, the method allows contraction of arbitrary vertex pairs—even across unconnected regions—and supports non-manifold surfaces, enabling flexible and accurate model simplification for varying levels of detail. The advantage of this method is that it reduces the number of vertices while taking into account the impact of vertex removal on the overall geometric structure.

However, when applied to large-scale mesh simplification, it can still result in varying degrees of 3D structural loss.

The proposed simplification strategy in this paper is characterized by its region-adaptive segmentation of geometric areas and the application of differentiated simplification levels based on the type of surface. The method is built upon Quadric Edge Collapse Decimation (QEM), further optimized by incorporating regional geometric features. Compared to traditional algorithms, its main advantage lies in its ability to significantly reduce the number of vertices in the mesh model while effectively preserving prominent sharp edges and boundaries, thus avoiding the loss of critical structural elements.

3 Methodology

The core objective of this study is to reduce the number of vertices in mesh models reconstructed from raw point cloud data, while preserving geometric fidelity in areas with structural complexity. This problem can be formally defined as follows: given a point cloud $P = \{p_1, p_2, ..., p_N\}$, the goal is to generate a simplified mesh M' with significantly fewer vertices than a full-resolution reconstruction, such that flat or smooth regions are highly compressed, while regions with high curvature or structural importance retain sufficient detail.

To achieve this objective, our pipeline first applies a mesh reconstruction algorithm to convert the raw point cloud into an initial triangular mesh. This mesh then serves as the input for a segmentation stage, in which vertices are partitioned according to their geometric characteristics. We adopt a traditional unsupervised segmentation strategy based on the Region Growing algorithm, which segments the point cloud into geometrically similar regions by iteratively expanding from seed points according to local curvature and normal similarity.

By explicitly segmenting the geometry before simplification, we enable an adaptive simplification process—allocating more vertices to complex regions and fewer to simpler ones—thereby overcoming the limitations of traditional mesh simplification techniques, which apply uniform decimation across all regions and often severely degrade critical structural details.

Furthermore, to improve the overall simplification quality, we enhance the classical *edge collapse* strategy by distinguishing between boundary and interior vertices, prioritizing the collapse of interior points. This allows us to preserve edges and contour structures as much as possible, achieving a better balance between vertex reduction and geometric fidelity.
In the following sections, we provide a detailed explanation of each component of our method.

3.1 Mesh model reconstruction

In this study, we evaluate and compare three widely used mesh reconstruction methods: Poisson Reconstruction, Marching Cubes, and the Ball Pivoting Algorithm (BPA). These methods are applied to the same real-world architectural point cloud dataset to assess their performance under conditions such as structural complexity, open boundaries, and surface noise.

Based on experimental results, we selected Ball Pivoting as the primary mesh reconstruction method in our pipeline. Compared to the other two methods, Ball Pivoting demonstrates several significant advantages. First, it offers higher geometric fidelity by directly generating triangular faces from the original point samples, without requiring interpolation or grid-based resampling. Second, it is more robust when dealing with real-world data, which often contains boundary discontinuities, irregular surfaces, and scanning noise. Third, the mesh produced by Ball Pivoting preserves structural details more accurately, making it more suitable for downstream tasks such as segmentation and adaptive simplification.

3.1.1 Poisson Reconstruction

Poisson surface reconstruction seeks an implicit *indicator function* $\chi(\mathbf{x})$ that equals 1 inside the object and 0 outside, such that its gradient $\nabla\chi(\mathbf{x})$ matches a vector field $\mathbf{V}(\mathbf{x})$ derived from the input point cloud's oriented normals.
The optimization problem can be formulated as:

$$\min_{\chi} \int_{\Omega} \|\nabla\chi(\mathbf{x}) - \mathbf{V}(\mathbf{x})\|^2 \, d\mathbf{x} \tag{1}$$

where:

- $\Omega \subset \mathbb{R}^3$ — the reconstruction domain.
- $\mathbf{x} \in \mathbb{R}^3$ — a spatial location.
- $\chi(\mathbf{x})$ — implicit indicator function (1 inside, 0 outside).
- $\mathbf{V}(\mathbf{x})$ — vector field constructed from oriented point normals.

Taking the Euler–Lagrange equation yields the Poisson equation:

$$\Delta\chi = \nabla \cdot \mathbf{V} \tag{2}$$

where:

- Δ — Laplace operator.
- $\nabla\cdot$ — divergence operator.

The equation is solved using an octree-based finite element method. The final surface is extracted as an iso-surface:

$$\mathscr{S} = \{\mathbf{x} \mid \chi(\mathbf{x}) = \tau\}, \quad \tau \approx 0.5 \tag{3}$$

where τ is the chosen threshold (typically 0.5).

3.1.2 Marching Cubes

Marching Cubes discretizes Ω into a regular grid of cubes. Each cube vertex $\mathbf{x}_i$ is assigned a scalar field value $f(\mathbf{x}_i)$, where the target surface is defined as the zero level set:

$$\mathscr{S} = \{\mathbf{x} \mid f(\mathbf{x}) = 0\} \tag{4}$$

where:

- $f(\mathbf{x})$ — scalar field value at position $\mathbf{x}$ (positive inside, negative outside, or vice versa).

The sign of $f(\mathbf{x}_i)$ at the cube's eight vertices is encoded as:

$$b_i = \begin{cases} 1, & f(\mathbf{x}_i) > 0, \\ 0, & f(\mathbf{x}_i) \leq 0, \end{cases} \tag{5}$$

yielding an 8-bit index to a precomputed triangulation lookup table.
Intersection points $\mathbf{p}_{\text{edge}}$ on cube edges are computed by linear interpolation:

$$\mathbf{p}_{\text{edge}} = \mathbf{p}_1 + \frac{0 - f(\mathbf{p}_1)}{f(\mathbf{p}_2) - f(\mathbf{p}_1)} \cdot (\mathbf{p}_2 - \mathbf{p}_1) \tag{6}$$

where $\mathbf{p}_1$ and $\mathbf{p}_2$ are the edge endpoints.

3.1.3 Ball Pivoting Algorithm (BPA)

Among the methods evaluated, the Ball Pivoting Algorithm (BPA) stands out for its simplicity and effectiveness in handling real-world architectural point clouds. The intuition behind BPA is straightforward: imagine placing a ball of radius r on the surface of the point cloud and rolling it around. Whenever the ball touches three points without enclosing any other point inside, those three points are considered to form a valid triangle.
Formally, given a point cloud $P = \{p_1, p_2, ..., p_N\}$, BPA looks for triplets of points (p_i, p_j, p_k) such that:

$$\exists c \in \mathbb{R}^3 \text{ such that } \begin{cases} \|p_i - c\| = \|p_j - c\| = \|p_k - c\| = r \\ \forall p \in P \setminus \{p_i, p_j, p_k\},\ \|p - c\| \geq r \end{cases} \tag{7}$$

Here, c is the center of the ball and r is the pivoting radius. If such a center exists, it means the ball can "rest" on those three points without overlapping any others, and thus the triangle $\triangle(p_i, p_j, p_k)$ is added to the mesh. The algorithm then continues by pivoting the ball around the existing triangle edges to search for adjacent triangles.
Unlike Poisson reconstruction, which often generates redundant fragments in areas with holes or open boundaries, or Marching Cubes, which suffers from boundary inaccuracies due to voxelization, Ball Pivoting operates directly on the raw point cloud and maintains a clean and topologically faithful mesh. These advantages make it a more robust and effective choice for high-fidelity reconstruction of real-world architectural scenes.

3.2 Region Growing-Based Segmentation: A Unsupervised Clustering Approach

Because deep-learning–based point cloud segmentation remains relatively unstable, it lacks sufficient reliability in practical applications. As a result, it does not meet our requirements—especially when it comes to preserving regions with significant geometric variation and key architectural elements such as windows and doors during mesh reconstruction. Most regions in architectural point clouds are dominated by planar or gently curved surfaces, except for areas with sharp geometric changes. Based on this observation, we shifted our focus to traditional unsupervised segmentation methods. These methods are generally more stable and easier to interpret.

Region Growing is a classical point cloud segmentation method based on local neighborhood similarity. Its fundamental idea is to start from one or more seed points and iteratively incorporate neighboring points that satisfy predefined similarity criteria into the same cluster. Specifically, the algorithm first selects an initial set of seed points, searches for neighboring points that meet the similarity conditions. A candidate point p_j is added to the current region only if it satisfies both spatial proximity and surface normal similarity conditions, equation 8 defined as:

$$\|p_i - p_j\| < \delta_d \quad \text{and} \quad \arccos(\mathbf{n}_i \cdot \mathbf{n}_j) < \delta_\theta \tag{8}$$

where $\|p_i - p_j\|$ denotes the Euclidean distance between points p_i and p_j, $\mathbf{n}_i$ and $\mathbf{n}_j$ are their respective normal vectors, and δ_d and δ_θ are the predefined thresholds for distance and normal vector angle deviation, respectively.

The newly added points then serve as new growth centers to further expand the region. This process is repeated until no additional points satisfy the criteria, ultimately partitioning the point cloud into multiple coherent sub-regions with similar geometric properties.

In this study, we aim to accurately distinguish important architectural elements such as window frames, door edges, and local geometric protrusions. Therefore, we segment point cloud regions based on the curvature characteristics of local fragments. Experimental results show that, when applying region growing to the convex and edge areas of architectural surfaces, these regions are segmented into a series of continuous small-scale patches. This behavior provides new insights into how Region Growing can be leveraged to more precisely extract and segment geometrically significant structures. A series of continuous small-scale fragments can be seen in Figure 3. Based on the aforementioned observations, we propose an unsupervised point cloud aggregation framework. We observed that non-planar regions generated by Region Growing exhibit spatial continuity between adjacent point cloud fragments. Therefore, if these small-scale fragments can be aggregated based on their spatial proximity, regions such as window frames or architectural corner features are expected to be merged into larger coherent clusters.

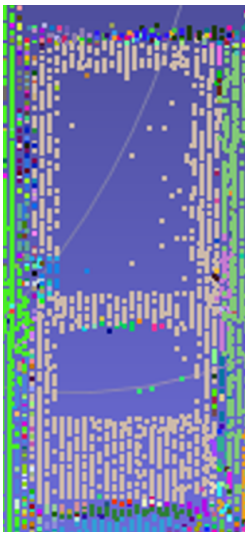


Figure 3: Points in different colors represent distinct clusters resulting from the region growing segmentation; most edge points belong to separate clusters.

In the specific implementation, we employ a spherical Kd-tree search strategy. Starting from a point cloud cluster generated by Region Growing with fewer than 500 points, we search for its neighboring points within the complete point cloud. The cluster IDs associated with these neighboring points are recorded. Subsequently, the newly identified clusters are treated as new search centers, and the process is iteratively repeated until all contiguous small-scale fragments are aggregated into a unified region. This clustring process can be easily understood by Figure 4. Using this approach, window and door regions can be progressively separated from the walls. The flowchart of the proposed method is shown in Figure 5.

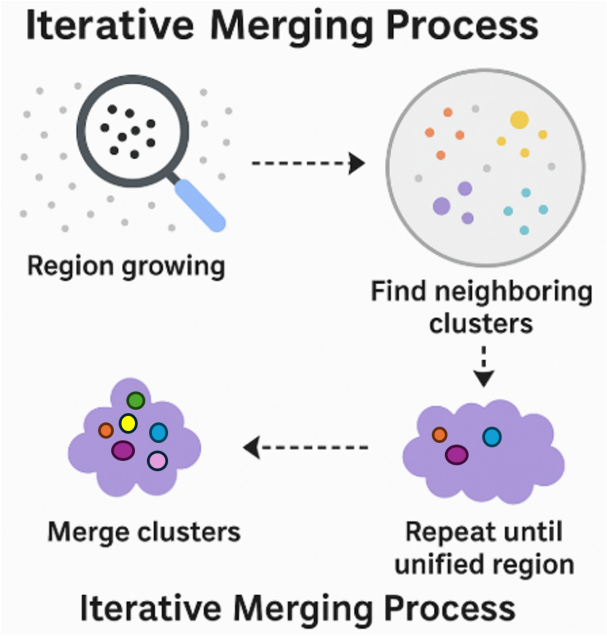


Figure 4: Iterative Aggregation of Small Clusters into Unified Regions Using Neighborhood Search

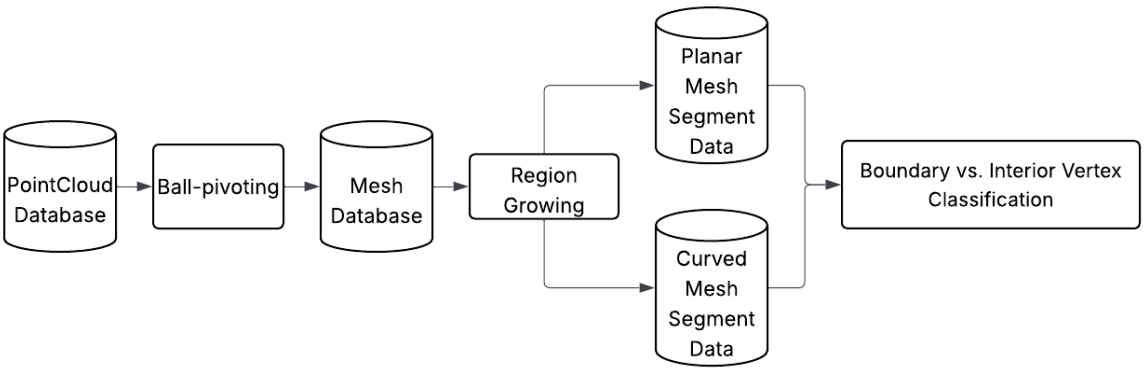


Figure 5: Proposed Method Workflow of unsupervised segmentation

3.3 Mesh Simplification

After extracting the point cloud vertices corresponding to the mesh structures we intend to preserve, we further consider how to maximize vertex reduction across different types of mesh fragments characterized by varying curvatures. Our optimization objective is to minimize the number of mesh vertices as much as possible without compromising essential geometric features, adapting the degree of simplification based on the fragment type. For instance, a flat wall surface without any geometric protrusions can be sufficiently represented using only 4–5 vertices, without the need to redundantly employ hundreds or thousands of points and triangles to describe a simple planar region.

Inspired by the Quadric Error Metrics (QEM) based Edge Collapse approach, we aim to consider the impact of vertex removal on local geometry during mesh simplification, so as to minimize the geometric degradation of the original mesh model. On this basis, we propose a boundary-aware QEM edge collapse method. After partitioning the mesh into regions, the edges between segments often correspond to geometric feature areas. For these areas, we apply different optimization thresholds to boundary vertices and interior vertices, thereby further reducing the geometric degradation in feature regions.

3.3.1 Quadric Error Metrics based Edge Collapse

Therefore, after a comprehensive comparison of experimental results, we found that the Edge Collapse algorithm performs significantly better in preserving critical geometric features. Unlike the spatially-based aggregation in Vertex Clustering, Edge Collapse evaluates the impact of collapsing each edge on the overall mesh structure and selects the optimal edge to contract based on Quadric Error Metrics (QEM). This not only effectively removes redundant edges and vertices but also better preserves geometrically significant regions, especially structural edges and sharp corners, during the simplification process.

Quadric Error Metrics (QEM) is a widely used approach for mesh simplification that quantitatively measures the geometric error introduced when a vertex is removed or collapsed. The core idea is to evaluate the squared distance from a point to the set of planes it belongs to and to minimize the accumulated error.

Plane Definition. Given a plane defined by the equation:

$$ax + by + cz + d = 0,$$

we can represent it compactly as:

$$\mathbf{n}^T \mathbf{v} + d = 0,$$

where $\mathbf{n} = (a, b, c)^T$ is the normal vector of the plane and $\mathbf{v} = (x, y, z)^T$ is a point in 3D space.

Squared Distance to a Plane. The squared distance from a point $\mathbf{v}$ to the plane is given by:

$$E(\mathbf{v}) = (\mathbf{n}^T \mathbf{v} + d)^2.$$

Quadric Form. This can be rewritten in quadratic form:

$$E(\mathbf{v}) = \mathbf{v}^T \mathbf{A} \mathbf{v} + 2\mathbf{b}^T \mathbf{v} + c,$$

where the coefficients are aggregated from all planes incident to the vertex:

$$\mathbf{A} = \sum_i \mathbf{n}_i \mathbf{n}_i^T, \quad \mathbf{b} = \sum_i d_i \mathbf{n}_i, \quad c = \sum_i d_i^2.$$

Quadric Matrix Representation. Each plane can be encoded using a 4×4 symmetric matrix:

$$Q = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \begin{bmatrix} a & b & c & d \end{bmatrix},$$

and the total error matrix Q for a vertex is the sum over all incident planes.

Error of a Vertex Collapse. When collapsing two vertices v_1 and v_2, we compute the total quadric:

$$Q = Q_{v_1} + Q_{v_2},$$

and the error of placing a new vertex v at a candidate position is:

$$E(v) = v^T Q v.$$

We selected Quadric Edge Collapse Decimation [14] as the foundational strategy for mesh vertex simplification. Prior to this, our approach involved simplifying different types of mesh

fragments individually, and then reassembling the simplified fragments into a complete mesh model. However, this segmented simplification strategy introduced two major challenges.

First, during aggressive simplification, the fragment boundaries—which themselves consist of a large number of vertices and edges—are prone to over-simplification, resulting in the loss of critical geometric detail along the edges. Figure 6 illustrates a scenario in which aggressive simplification leads to the loss of important geometric fragments. Second, even when applying the same simplification parameters across all fragments, the resulting boundary geometries may differ significantly, making it difficult to accurately stitch the simplified fragments back together. This leads to discontinuities and misalignments in the reconstructed global mesh structure. Figure 7 Illustrate Geometric Fragment Loss and Boundary Misalignment Caused by Aggressive Simplification. Secondly, applying the same simplification parameters to both boundary vertices

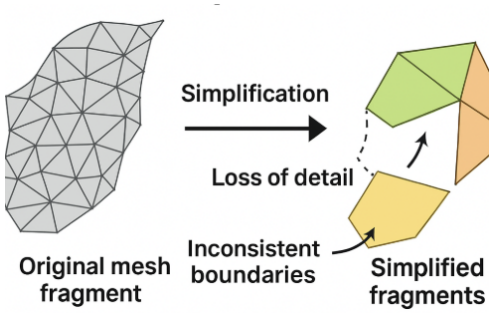


Figure 6: Geometric Errors Caused by Aggressive Mesh Fragment Simplification

and interior planar vertices can inevitably lead to the degradation of geometric structure. In our study, since the mesh fragments are segmented by Region Growing, their boundaries inherently correspond to areas of sharp geometric transitions. To address this, we adopt a differentiated simplification strategy by assigning separate parameter tables for planar regions and boundary regions. This allows us to better control the simplification process, effectively reducing the number of vertices while preserving critical structural features. In summary, our final simplification strategy explicitly distinguishes between boundary vertices/edges and interior ones within each mesh fragment. The process begins by applying Quadric Edge Collapse Decimation to all internal edges of each fragment, effectively reducing redundant vertices within the planar regions and curved regions. After assembling the simplified fragments back into a complete mesh, a second-stage simplification is performed specifically on the boundary vertices. This stage utilizes a separate parameter table tailored for edge regions, ensuring that the boundaries are not overly simplified. As a result, the overall vertex count is significantly reduced while preserving the geometric integrity and continuity of critical boundary structures. For the workflow of our proposed simplification algorithm, refer to Figure 8.

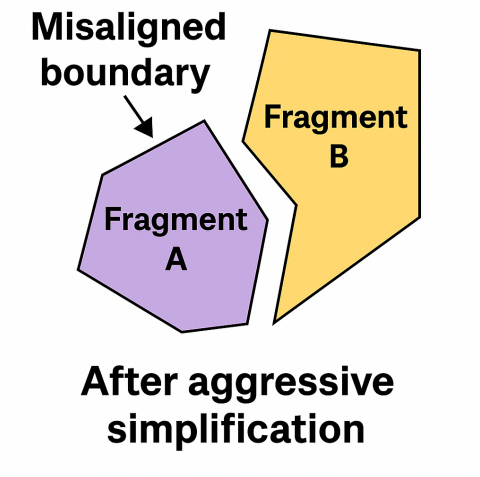


Figure 7: Geometric Misalignment Caused by Aggressive Mesh Simplification

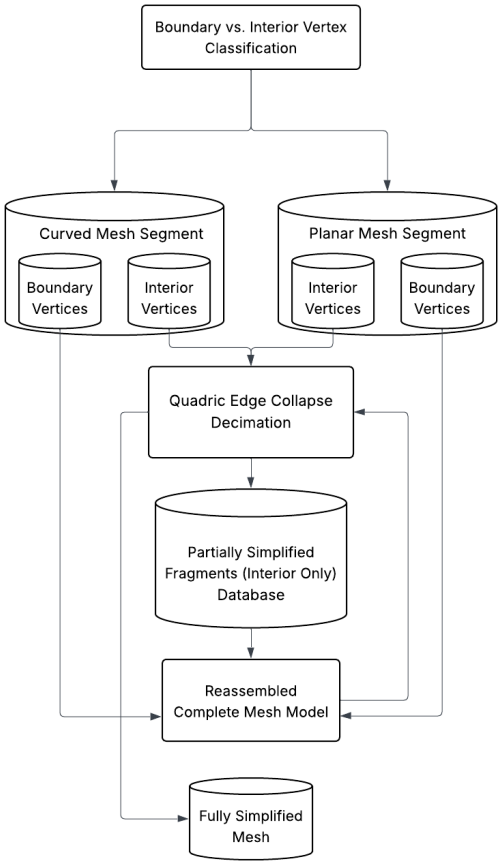


Figure 8: Proposed simplification algorithm

3.3.2 Vertex Clustering

Vertex Clustering is one of the most widely used and conceptually simple mesh simplification techniques. The method reduces mesh complexity from a spatial perspective by overlaying the model with a uniform voxel grid. All vertices that fall within the same voxel cell are merged into a single representative vertex, typically chosen as the centroid of the points in that cell. Formally, given a grid cell C containing vertices $\{v_1, v_2, \ldots, v_n\}$, the representative vertex $\hat{v}$ is computed as

$$\hat{v} = \frac{1}{n} \sum_{i=1}^{n} v_i. \tag{9}$$

This yields a coarse but efficient simplification, as the number of vertices is reduced in proportion to the number of occupied grid cells. However, since the algorithm does not explicitly account for the geometric importance of vertices, it often results in noticeable detail loss, particularly along sharp edges and high-curvature regions.

4 Data Preparation

In this study, we constructed a custom radar-scanned point cloud dataset, representing a semi-panoramic map of a residential district in the Netherlands. The dataset captures a variety of urban features including buildings, streets, and vegetation. Each building point cloud contains approximately 50,000–250,000 points, depending on occlusion and scan coverage. Each point in the dataset contains rich attributes: 3D coordinates (x, y, z), color information (RGB), depth (D), and reflection intensity.

In the methodology section, we introduced two distinct point cloud segmentation strategies: a deep learning–based approach using RandLA-Net and an unsupervised method based on Region Growing. These two approaches require different preprocessing pipelines.

Dataset Preparation for Deep Learning (RandLA-Net) To train the RandLA-Net model, we first extracted building point clouds from the full scene using CloudCompare. Our objective was to train the model to accurately identify and segment structural components such as windows, window frames, doors, and door frames from the overall building geometry. We manually annotated the dataset into six semantic categories: *door frame*, *door*, *window*, *window frame*, *wall*, and *roof*. The dataset includes point clouds from 26 buildings for training and 12 buildings for testing.

Dataset Preparation for Region Growing (Unsupervised) Although the Region Growing algorithm does not require annotated training labels, it still benefits from noise reduction and point distribution smoothing. To this end, we applied grid sampling with a voxel size of 0.3 m—preserving only one representative point per voxel—to ensure a more uniform point

distribution. In addition, we applied *Moving Least Squares (MLS)* filtering to eliminate outlier points and reduce local surface noise. This preprocessing step helps mitigate errors such as QHULL mesh distortion in subsequent surface reconstruction tasks.

5 Experiment and Result

In this chapter, we conduct a systematic experimental evaluation and comparison of the three key stages involved in this study.

First, in the mesh reconstruction stage, we examine the performance of three typical methods—Marching Cubes (using RIMLS for volumetric reconstruction), Poisson Surface Reconstruction, and Ball-Pivoting—in terms of their reconstruction accuracy when applied to real-world architectural point clouds. The accuracy is evaluated by observing the spatial deviation between the reconstructed mesh vertices and the original point cloud data.

Second, in the region segmentation stage, we compare the performance of a supervised learning method, RandLA-Net, with an unsupervised geometric clustering approach, Region Growing. Although the RandLA-Net model incorporates an intensity-enhanced feature module tailored for radar data, experiments using our manually labeled architectural point cloud dataset show that the Region Growing method exhibits higher stability and accuracy in identifying key regions with prominent geometric structures.

Finally, we propose and test an improved mesh simplification strategy based on the traditional edge collapse method. In the comparative experiments between the two approaches, we use indicators such as Principal Component Analysis (PCA), surface normal deviation, and geometric boundary range to evaluate simplification performance. The results demonstrate that our improved strategy outperforms the traditional uniform simplification approach in preserving critical structural features.

The following sections will introduce the specific methods and experimental results for each of these stages in detail.

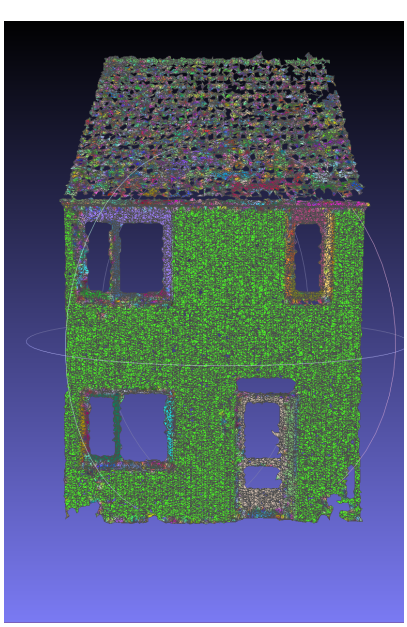
5.1 Mesh Model Reconstruction

To evaluate the performance of different mesh reconstruction algorithms on architectural point clouds with open boundaries, we conducted comparative experiments using three representative methods: **Marching Cubes**, **Poisson Surface Reconstruction**, and **Ball-Pivoting**. The comparison focuses on geometric fidelity, boundary preservation, and reconstruction accuracy.

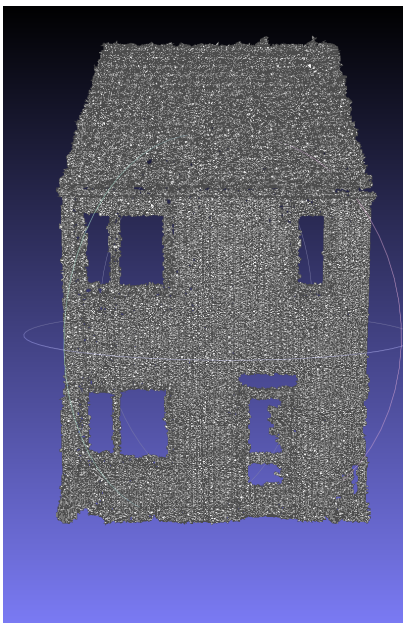
Before performing 3D mesh reconstruction, we uniformly estimated the normals of the original point cloud. The normal estimation was based on the **10 nearest neighbors** for each point.

Table 1: Parameter settings for mesh reconstruction algorithms

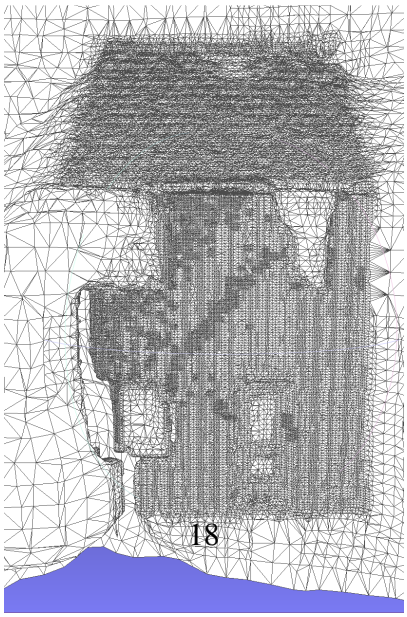
Method	**Parameter Settings**
Marching Cubes (RIMLS)	MLS filter scale: 2 Projection accuracy: 0.0001 Max projection iterations: 15 MLS sharpness: 0.75 Max fitting iterations: 3 Grid resolution: 500
Poisson Surface Reconstruction	Octree depth: 12 Solver Divide: 6 Samples per node: 1.5 Surface offsetting: 1
Ball-Pivoting	Ball radius: 0.03, 0.06, 0.09 (multi-scale) Normal consistency check: enabled



(a) Ball Pivoting



(b) Marching Cubes (RIMLS)



(c) Poisson Surface Reconstruction

The point cloud data of the same building was reconstructed into 3D mesh models using these three methods. The reconstruction results are shown in Figure 9. From the figure, it is evident that the Marching Cubes algorithm generates a large number of redundant mesh faces when dealing with point cloud fragments that have open boundaries, performing significantly worse than the other two methods. As shown in Figure 11, the top-left image displays the mesh vertices generated by the Marching Cubes method (white points). It is clearly visible that many vertices extend beyond the boundaries of the original colored point cloud. Especially around edge regions, the algorithm connects distant points across boundaries, resulting in distorted and unrealistic mesh faces (see bottom-left image). These artifacts significantly degrade the fidelity of the reconstructed geometry and distort the architectural boundary structure. The top-right image shows the vertex distribution generated by the Poisson reconstruction algorithm. Compared with the original point cloud, it is evident that Poisson reconstruction, which assumes a watertight input surface, produces a large number of redundant vertices when applied to architectural structures with open boundaries. Most of these extraneous vertices are located outside the actual building outline, leading to a final reconstruction (see bottom-right image) that suffers from significant structural deviation and excessive surface smoothing. As a result, it fails to accurately represent the geometric features of the original architecture. The specific parameter settings are provided in Table 1.

When applying the Ball-Pivoting algorithm for mesh reconstruction, the method does not introduce additional redundant vertices beyond those in the original point cloud. Although not every point is used in the final mesh, the algorithm's point-based surface generation ensures that the reconstructed geometry remains faithful and accurate. Furthermore, the use of normal consistency in the algorithm further enhances the smoothness and precision of reconstructed edges and surfaces. The Figure 10 shows reconstructed result of using ball-pivoting.

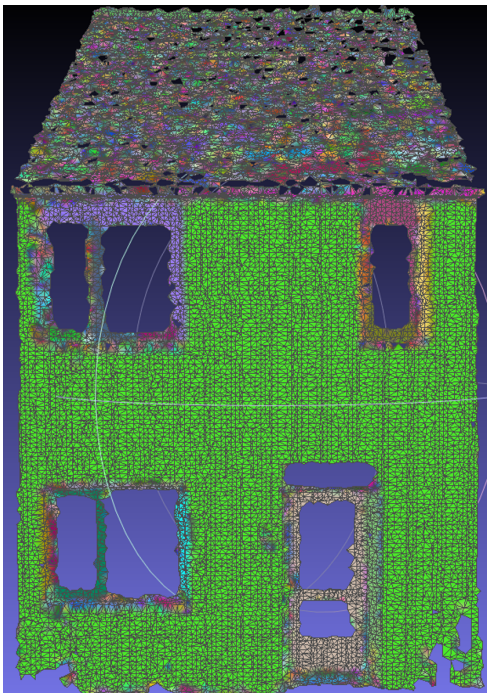


Figure 10: Reconstruction result of the building point cloud using the Ball Pivoting algorithm, demonstrating preserved hole boundaries and faithful retention of the original geometric structure.

Overall, both reconstruction methods have limitations when applied to architectural point clouds with open boundaries. Marching Cubes is somewhat capable of preserving sharp boundary structures, but tends to generate excessive out-of-bounds mesh faces, especially near edges. In contrast, Poisson reconstruction produces smoother surfaces but performs poorly on non-watertight data, often deviating from the true geometry and introducing substantial topological artifacts. Ball-Pivoting, on the other hand, avoids introducing redundant data and better preserves the original geometry, making it more suitable for architectural point clouds with open boundaries.

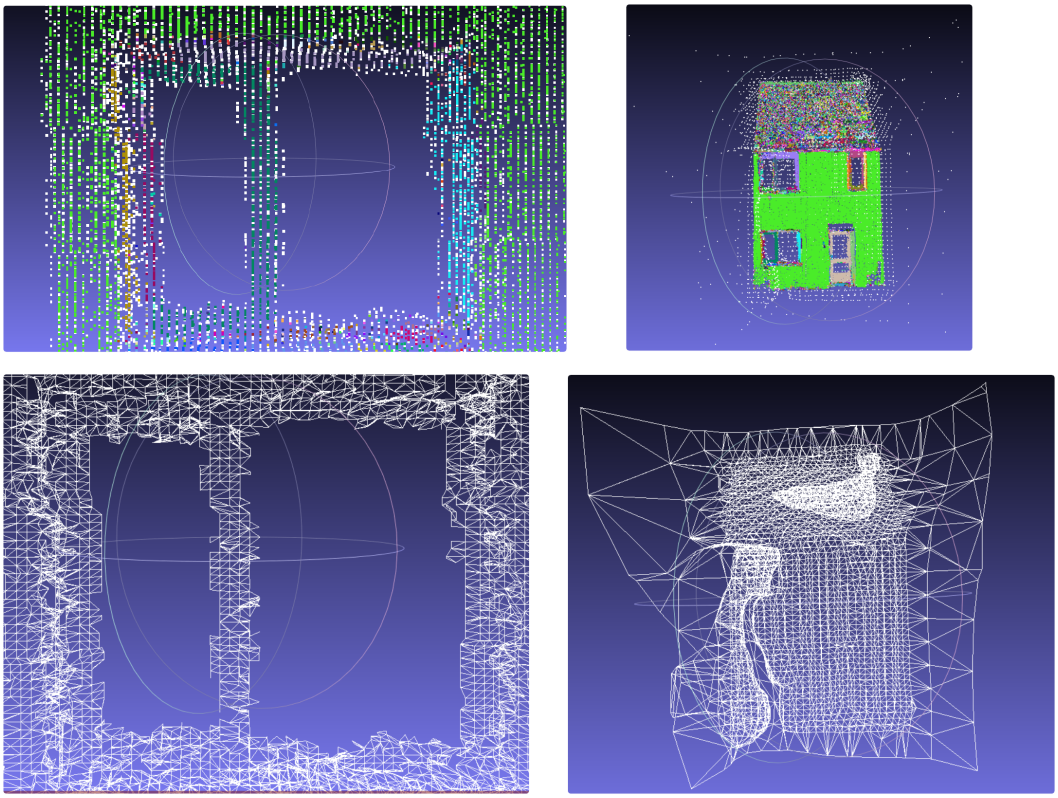


Figure 11: In the top-left image, white points indicate the mesh vertices generated by the Marching Cubes algorithm. These vertices extend beyond the boundaries of the original colored point cloud, resulting in reconstructed faces that distort the true edge structure, as shown in the bottom-left image. The top-right image compares the Poisson reconstruction vertices (in white) with the original point cloud; because the architectural surfaces have open boundaries, many redundant vertices appear outside the building outline. The bottom-right image presents the final Poisson reconstruction, which clearly disrupts the building's original geometric structure.

5.2 Point Cloud Segmentation

In the early stages of this study, we explored deep learning–based point cloud semantic segmentation methods to extract fine-grained architectural elements such as windows, door frames, and other decorative components from building point clouds. To this end, we conducted experiments on two representative deep learning architectures: PCT and RandLA-Net.

The dataset used in these experiments was a non-public outdoor architectural point cloud dataset provided by the Dutch company GeoSignum. In this dataset, key architectural components—such as window and door frames, which typically correspond to geometrically complex regions—were manually annotated using CloudCompare software to serve as ground truth semantic labels.

We first experimented with the Point Cloud Transformer (PCT)[12] , a Transformer-based model that employs self-attention mechanisms for global feature modeling. However, due to its large GPU memory requirements and the need for massive amounts of data, we attempted to reduce

the number of feature points used for training. As a result, the model exhibited very low accuracy, and therefore we abandoned this approach.

We then turned to RandLA-Net for further exploration. This model features a lightweight architecture and strong local feature aggregation capability, making it applicable to large-scale point cloud processing tasks. In our experiments, we incorporated radar-specific intensity information to enhance feature extraction, and evaluated its segmentation performance.

Since this project is oriented toward real-world engineering applications, we also experimented with traditional unsupervised segmentation strategies. Among them, we applied the Region Growing algorithm to perform region extraction on architectural point cloud data.

The following subsections will describe each part of this process in detail.

5.2.1 RandLA-Net Baseline vs. Enhanced Version

Given the nature of our radar-based point cloud data, we modified the input to RandLA-Net by adding reflection intensity to the original geometric and color features. This enhancement enables the model to better perceive differences between materials, thereby improving segmentation performance on fine-grained structural elements. The intensity-colored point cloud visualization shown in Figure 12, the contours and structures of doors and windows—including both their main bodies and frames—are more intuitively highlighted.

For the RandLA-Net architecture, we tested two configurations: one using only RGB-D and geometric features (x, y, z, R, G, B, D), and another enhanced version including radar reflection intensity (x, y, z, R, G, B, D, I). The experimental results indicate that the baseline model performs poorly in segmenting fine architectural components. Specifically, it achieved only 3.2% IoU for door frames and 7.7% for window frames. This performance is attributed to the difficulty in distinguishing geometrically similar regions where RGB-D differences are subtle.

After incorporating intensity features, the segmentation accuracy improved significantly, especially for components with distinct material properties. Metallic structures such as window and door frames, which typically exhibit higher radar reflectivity, became more distinguishable from adjacent wall surfaces. As shown in Table 2, the IoU for door frames increased to 68.9%, and for doors to 42.1%, while the mean IoU improved from 28.58% to 53.43%.

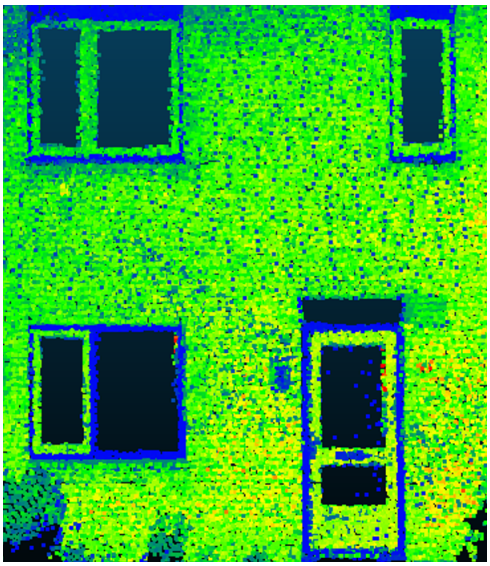


Figure 12: Windows, doors, and their respective frames are clearly separated based on intensity information, showing distinct regional boundaries.

Table 2: Segmentation performance metrics for different components

Method	**Door Frame**	**Door**	**Window**	**Window Frame**	**Roof**	**Wall**	**Mean IoU**
Baseline (RGB-D)	3.2%	5.6%	12.0%	7.7%	68%	75%	28.58%
With Intensity (Ours)	68.9%	42.1%	22.5%	24.1%	78%	85%	53.43%

While the addition of intensity data significantly enhanced segmentation performance, especially in distinguishing material-based structures, challenges remain. Due to the inherent limitations of deep learning models—such as overfitting to training distributions and limited generalization in unseen environments—achieving segmentation accuracy above 90% remains difficult even with feature enhancement and data augmentation.

Moreover, segmentation errors directly impact the reliability of subsequent 3D modeling processes. For applications with strict structural integrity requirements, such as digital twin simulations or automated retrofitting analysis, even small segmentation mistakes can lead to substantial downstream errors. Therefore, despite the improvements brought by intensity features, deep learning–based segmentation methods still face reliability constraints in practical industrial scenarios.

5.2.2 Region Growing–Based Segmentation for Architectural Point Clouds

To segment point cloud regions using Region Growing, we first estimate normals and curvature on the architectural point cloud, then configure the unsupervised Region Growing algorithm with a normal deviation threshold $\theta = 2^\circ$ and a spatial distance threshold $d = 0.3\,\mathrm{m}$ to ensure that only points with similar normals and close proximity are grouped into the same cluster. Segmentation results show that planar or smoothly curved areas form large clusters, while regions with pronounced geometric variations—such as corners and window frames—are divided into numerous small clusters. As illustrated in Figure 13, walls, being nearly flat, are consolidated into a single large cluster, whereas windows and door frames are segmented into multiple small clusters.

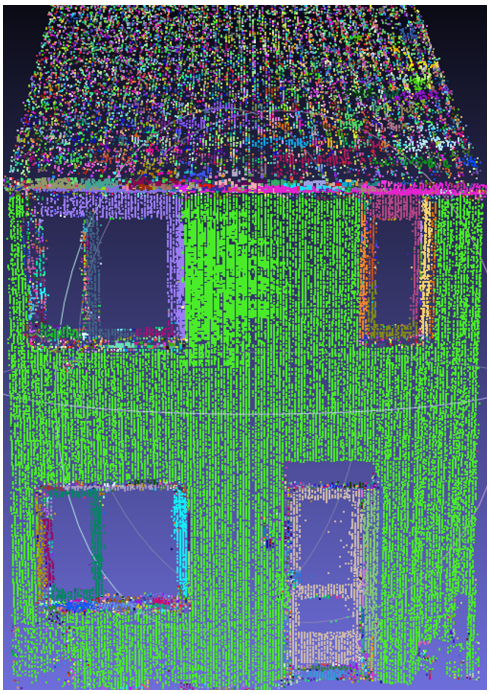
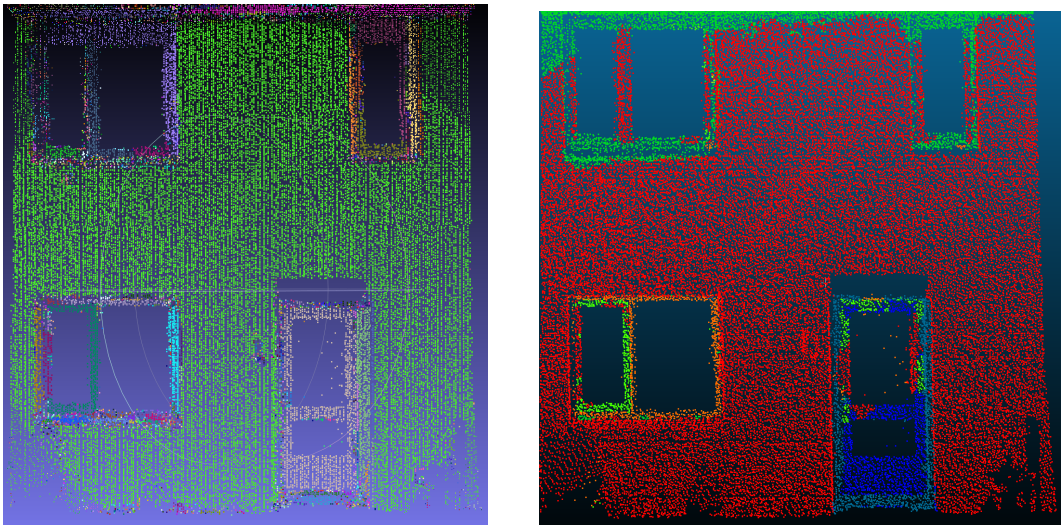


Figure 13: Result of applying the Region Growing algorithm to the building point cloud. Colors denote distinct clusters, with planar surfaces forming large clusters and high-curvature regions (e.g., window and door frames) segmented into numerous small clusters.

Subsequently, we use cluster size as a discriminative feature and leverage Kd tree neighborhood searches to expand and merge all small clusters, thereby precisely extracting regions of high curvature. KD-Tree (k-d tree) is a data structure for organizing and retrieving points in k-dimensional space. It recursively partitions the space by alternately selecting splitting hyperplanes along different dimensions, forming a balanced binary tree that enables efficient management of point sets. In other words, aside from the large green wall cluster shown in Figure 13, all other small clusters are identified and classified as geometric transition regions by our aggregation method. This accomplishes automatic segmentation of smooth surfaces and high curvature areas,

laying the groundwork for subsequent region specific vertex simplification. Figure 15 illustrates the point cloud corresponding to the extracted high curvature regions.



(a) Region Growing (b) RandLA-Net

Figure 14: Comparison of segmentation results using Region Growing and RandLA-Net. In (a), colorful blocks represent regions where points are grouped together as long as they can be connected into a continuous area. In (b), colors represent semantic labels: light blue for door frame, dark blue for door, light green for windows, orange-red for window frame, dark red for wall, and dark green for roof.

This Figure 14 illustrates the segmentation results of the improved RandLA-Net model and the Region Growing method when applied to the same building point cloud. A major drawback observed in RandLA-Net during the experiments is that, while it performs well in segmenting doors with relatively uniform shapes, its segmentation accuracy drops significantly when dealing with window categories that exhibit greater variability in the dataset. In contrast, the Region Growing approach tends to merge adjacent small clustered regions into complete parts during the subsequent simplification process, often representing entire structures such as doors and windows. Although the segmentation is not perfectly accurate, it demonstrates strong stability and effectively distinguishes between planar and curved surfaces.

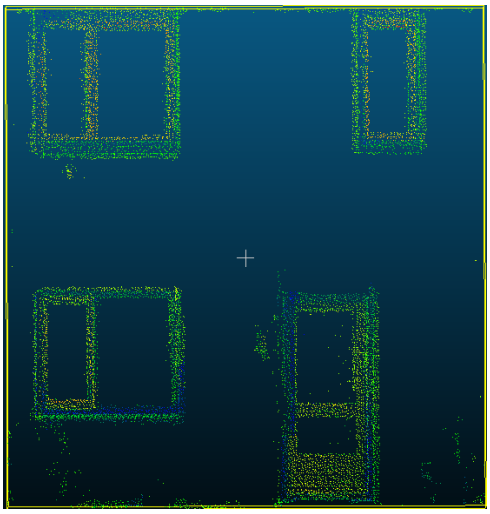


Figure 15: Segmented point clouds of door and window regions obtained using our proposed unsupervised aggregation framework.

5.3 Mesh Vertices Simplification

5.3.1 Structural Changes in Internal Vertices Across Different Surface Types

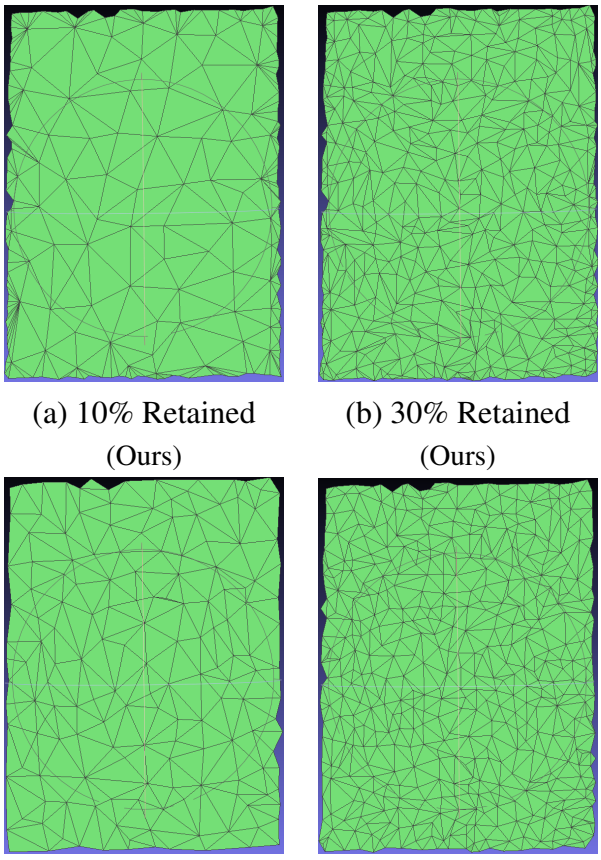
In this section, we present a comparative study between our proposed region-aware Quadric Edge Collapse (QEC) mesh simplification strategy and the standard QEC approach. By combining quantitative metrics with visual evaluation, we assess their respective abilities to preserve mesh fidelity after simplification and identify the appropriate simplification thresholds for different types of regions.

we designed dedicated experiments targeting different mesh patch types and curvature characteristics, aiming to determine the optimal simplification levels for each region. For planar areas, our goal was to represent flat surfaces with as few vertices as possible while preserving boundary structures. We tested two simplification thresholds—retaining 30%, 15% of the original vertices—to assess the trade-off between compression and geometric fidelity.

In Table 3, the columns **X Range**, **Y Range (Thickness)**, and **Z Range** represent the spatial extent of each mesh segment along the three axes. Specifically, **Y Range** corresponds to the thickness direction of the surface patch, which is often used to evaluate geometric distortion introduced during simplification. Table 3 demonstrates that, under our method, the geometry of the mesh segment remains stable when simplified to 30% of its original vertices, but begins to change noticeably at the 15% level. This suggests that the optimal simplification threshold for planar regions lies between 30% and 15%. In contrast, using the standard QEM-based edge collapse method, structural deformation occurs as early as the 30% simplification level.

This highlights the superior geometric fidelity of our approach during aggressive mesh simplification. The experimental mesh images are shown in Figure 16, which compares mesh simplification

under different retention ratios and methods. Meanwhile, Figure 17 visually illustrates how planar regions behave under different simplification thresholds within the context of the entire building point cloud scene.



(a) 10% Retained (Ours) (b) 30% Retained (Ours)

(c) 10% Retained (Original Edge Collapse) (d) 30% Retained (Original Edge Collapse)

Figure 16: Comparison of mesh simplification results under different retention ratios and methods.

Table 3: Comparison of geometric ranges under different simplification methods

Simplification Method	**X Range**	**Y Range (Thickness)**	**Z Range**
Original Mesh (100%)	1.61504117	0.02682999	2.14988475
Ours (30% Points Retained)	1.61504117	0.02682999	2.14988475
Ours (15% Points Retained)	1.61504117	0.02682999	2.14994150
Non-Optimized (30% Retained)	1.61575900	0.02585200	2.14989000
Non-Optimized (15% Retained)	1.59318600	0.02400800	2.14406000

In our experiments, we conducted Principal Component Analysis (PCA) on mesh fragments under different simplification thresholds to evaluate the geometric changes introduced by the simplification process. In particular, PCA computes the eigenvalues and eigenvectors of the

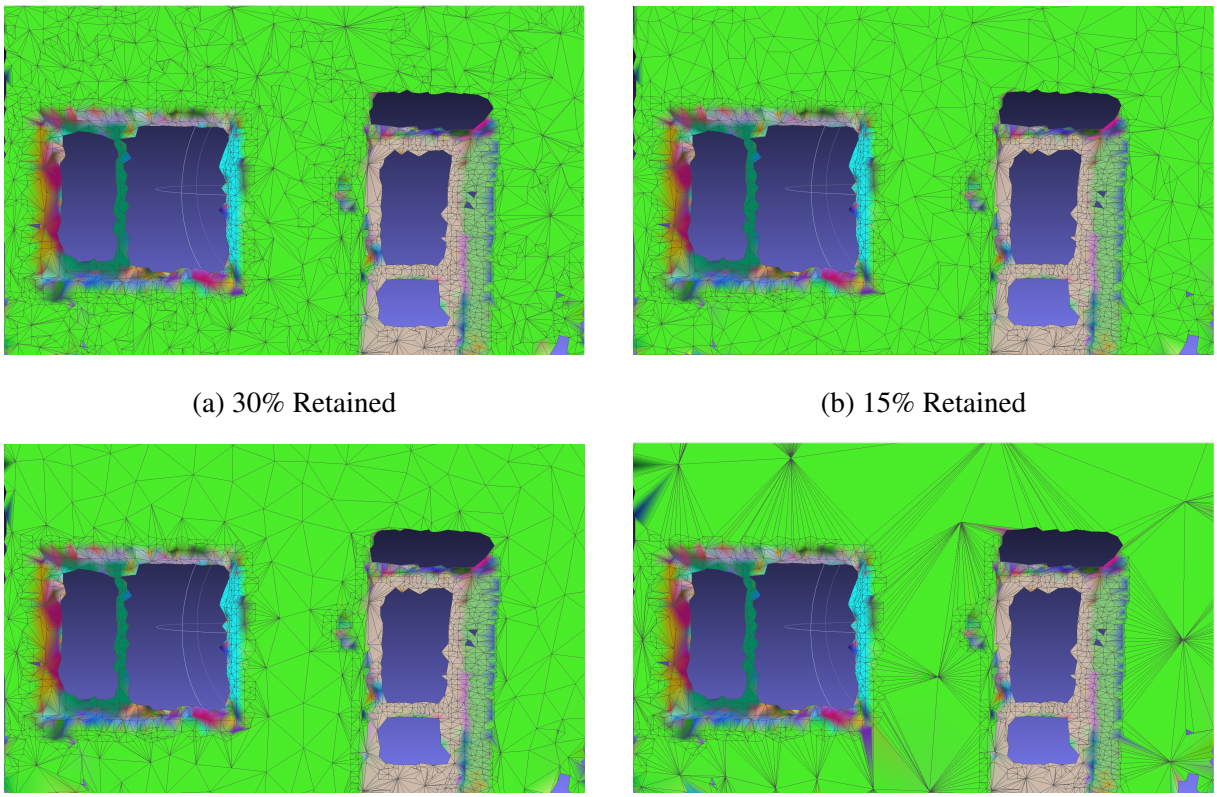
covariance matrix of the vertex positions within each mesh fragment. Given a set of n 3D points $\{\mathbf{x}_i \in \mathbb{R}^3\}_{i=1}^n$, we first center the data by subtracting the mean:

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$$

The covariance matrix is then computed as:

$$\mathbf{C} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

By performing eigen decomposition on the covariance matrix $\mathbf{C}$, we obtain the principal components, which indicate the primary directions of geometric variation in the mesh fragment. These principal directions can be used to analyze how well the local geometric structure is preserved or distorted during the simplification process.



(a) 30% Retained

(b) 15% Retained

(c) 10% Retained

(d) 5% Retained

Figure 17: Comparison of mesh simplification results under different vertex retention rates.

In our experiments, we applied Principal Component Analysis (PCA) to point cloud models at varying levels of simplification to quantitatively evaluate the variance contributions along the primary directions (PC1 and PC2) as well as the thickness direction (PC3). Notably, the third principal component (PC3) typically corresponds to the normal or thickness direction of

Table 4: Comparison of PC3 Variance Ratios and Deviation from Reference

Result of Optimized Method	**PC3 Variance Ratio**	**Deviation from Reference**	**#Points**
Full-resolution (Reference)	2.9808e-05	0.0000e+00	2700
30% Retained (Ours)	1.4329e-05	1.5478e-05	496
15% Retained (Ours)	4.1144e-05	1.1336e-05	243
10% Retained (Ours)	4.8213e-05	1.8406e-05	164
Result of Non-Optimized Method	**PC3 Variance Ratio**	**Deviation from Reference**	**#Points**
30% Retained (Non-optimized)	1.4799e-05	1.5009e-05	496
15% Retained (Non-optimized)	1.2928e-05	1.6880e-05	243
10% Retained (Non-optimized)	9.6660e-06	2.0142e-05	164

a mesh fragment, and changes in its variance ratio can reflect potential structural degradation, compression, or information loss during simplification.

Since this analysis is performed on nearly planar mesh fragments, we had previously demonstrated the superiority of our method in preserving boundary integrity by analyzing the changes in the XYZ spatial extent of each fragment under different simplification levels. However, perfect planar surfaces rarely exist in real-world data. Given that our segmentation process allows a tolerance angle of 2 degrees, the extracted planar patches inevitably contain slight curvature.

Therefore, this experiment further investigates how such subtle internal curvature evolves under different simplification levels. Theoretically, the closer the PC3 variance value remains to that of the reference mesh, the less internal curvature distortion occurs, indicating that the simplified fragment better preserves the original surface curvature and flatness.

We take the original, un-simplified mesh as the reference, with its third principal component (PC3) having an explained variance ratio of 2.9808e-05, representing structural features along the thickness direction. After simplifying the model with various retention rates, we compared the performance of our optimized method (retaining 30%, 15%, and 10% of the points) against the non-optimized method in preserving the variance along PC3. The results show that our method better retains the original thickness-related variance. For instance, when retaining 15% of the points, our method shows a deviation of only 1.1336e-05, while the non-optimized method reaches a deviation of 1.6880e-05 when retaining 10% of the points. This demonstrates the superior stability of our simplification approach in preserving geometric thickness structure for planar mesh surface. The specific figure is shown in Table 4. The conclusion is that both our method and the baseline approach are capable of preserving the 2D-dominant nature of planar segments without severe geometric collapse. However, our method exhibits smaller deviations

from the original mesh, particularly in preserving the structure along the thickness direction, demonstrating greater robustness and fidelity during simplification. For detailed PCA-based analysis results of meshes simplified using different methods and at varying simplification levels, please refer to Table 5.

Table 5: Complete PCA Analysis Results with Full Precision

Type	**Pts**	**Level of simplification**	**PCA Components**			
			Exp.Var.	**Axis 1**	**Axis 2**	**Axis 3**
Planar Surfaces (Ours)						
Planar	2700	Baseline Mesh	5.14812135e-01	3.95573495e-01	-4.86220600e-01	7.79173369e-01
			4.85158058e-01	-4.92546223e-01	6.03746463e-01	6.26808125e-01
			2.98076454e-05	7.75190188e-01	6.31727581e-01	6.60021644e-04
Planar	496	30% Retained (Ours)	6.25759260e-01	1.67005601e-02	9.99860482e-01	-3.26900785e-04
			3.74226411e-01	-2.18479209e-03	3.63437940e-04	9.99997547e-01
			1.43296513e-05	9.99858149e-01	-1.66998049e-02	2.19055689e-03
Planar	243	15% Retained (Ours)	5.16440977e-01	2.94578193e-02	6.26171169e-01	7.79128939e-01
			4.83517879e-01	-3.78568145e-02	-7.78209082e-01	6.26863212e-01
			4.11441126e-05	9.98848887e-01	-4.79613629e-02	7.80503271e-04
Planar	164	10% Retained (Ours)	5.19583625e-01	-2.39175122e-01	5.79463008e-01	7.79113524e-01
			4.80368162e-01	2.96110656e-01	-7.20650377e-01	6.26882376e-01
			4.82133817e-05	9.24723602e-01	3.80638486e-01	7.76471026e-04
Planar Surfaces (Original Edge Collapse)						
Planar	496	30% Retained (Original)	6.41305132e-01	3.66561489e-02	9.99327874e-01	-3.56750556e-04
			3.58680069e-01	-2.29713909e-03	4.41250429e-04	9.99997264e-01
			1.47990179e-05	9.99325297e-01	-3.66552291e-02	2.31176967e-03
Planar	243	15% Retained (Original)	6.48830995e-01	7.31727259e-03	9.99973077e-01	-5.50448517e-04
			3.51156078e-01	2.15383774e-03	5.34701340e-04	9.99997538e-01
			1.29275989e-05	-9.99970909e-01	7.31844015e-03	2.14986720e-03
Planar	164	10% Retained (Original)	6.48644533e-01	-9.95058805e-03	9.99950424e-01	3.68453326e-04
			3.51345801e-01	1.79847220e-03	3.86367713e-04	-9.99998308e-01
			9.66604235e-06	-9.99948874e-01	-9.94990856e-03	-1.80222762e-03

To further validate the geometric fidelity of our simplification algorithm on curved surfaces, we conducted a series of experiments using normal deviation analysis. For each vertex, we computed the angle between its normal vector and the normals of its neighboring vertices. These deviation angles were aggregated and plotted as cumulative distribution functions (CDFs) to assess surface smoothness and structural consistency after simplification.

This Figure 18 illustrates the vertex distribution of the architectural surface mesh model used in our normal deviation analysis experiment. Each point represents a mesh vertex, and its color indicates the average deviation angle between the vertex normal and the normals of its neighboring vertices. The color transitions from blue (low deviation) to red (high deviation), providing an intuitive representation of the geometric smoothness or structural variation across different regions.

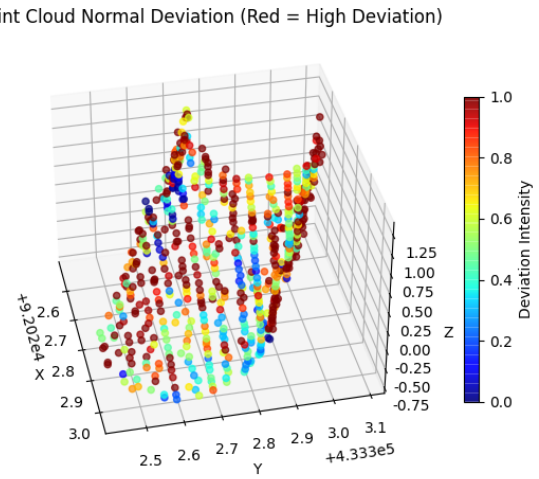
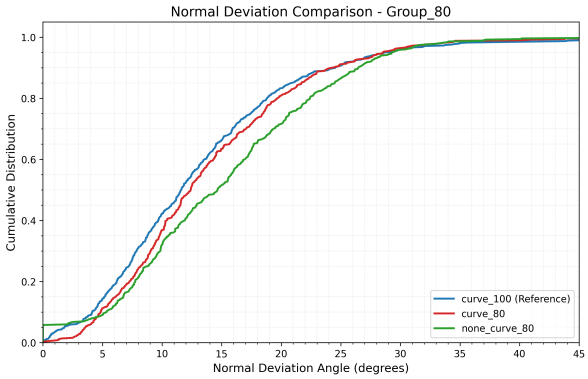


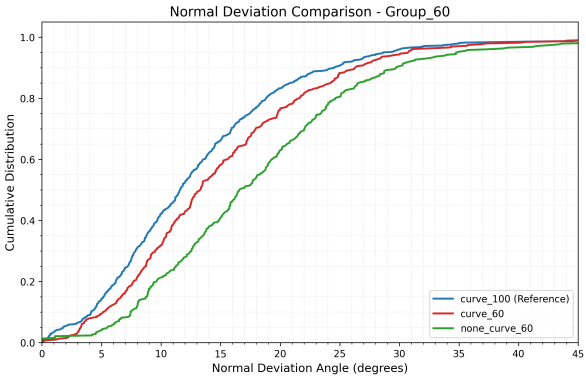
Figure 18: Visualization of Vertex Normal Deviation on Architectural Curved Surface Mesh

Using the unsimplified mesh patch as the reference, we performed grouped comparisons at four different simplification levels: 80%, 60%, 40%, and 20% retained vertices. The Figure 19 clearly show that our method achieves superior normal consistency, especially at the 80% and 60% levels, where the CDF curves closely match that of the original mesh. In contrast, the standard QEM-based simplification significantly degrades the surface geometry at all tested thresholds.

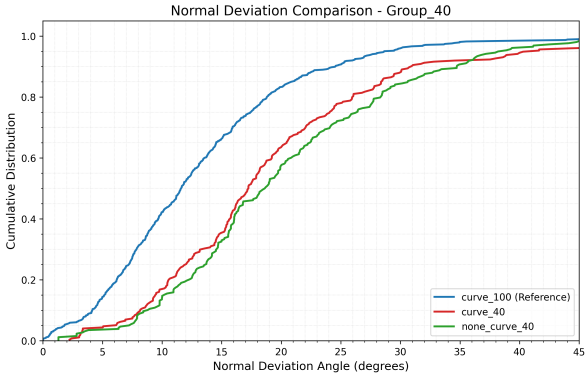
Moreover, the progressive shift of the CDF curves indicates that, as the number of retained vertices decreases, the mesh patch gradually becomes smoother and flatter. This trend suggests that the optimal simplification threshold for curved surfaces using our method lies between 80% and 60%, where the curvature is best preserved without introducing structural artifacts. Meanwhile, Figure 20 illustrates the visual outcomes of applying different simplification levels to doorframe edge regions within the architectural mesh, further highlighting the impact of simplification on high-curvature structures.



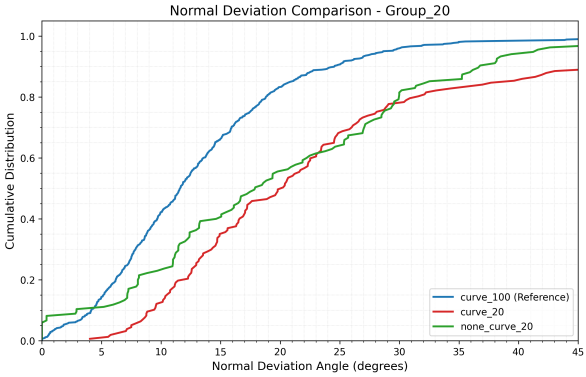
(a) 80% Retained



(b) 60% Retained



(c) 40% Retained



(d) 20% Retained

Figure 19: Normal deviation comparison under different simplification thresholds. The blue curve represents the normal distribution of the original unsimplified surface patch and serves as the reference baseline. The red curve shows the normal distribution after applying our proposed simplification algorithm, while the green curve corresponds to the distribution obtained using the standard QEC simplification method.

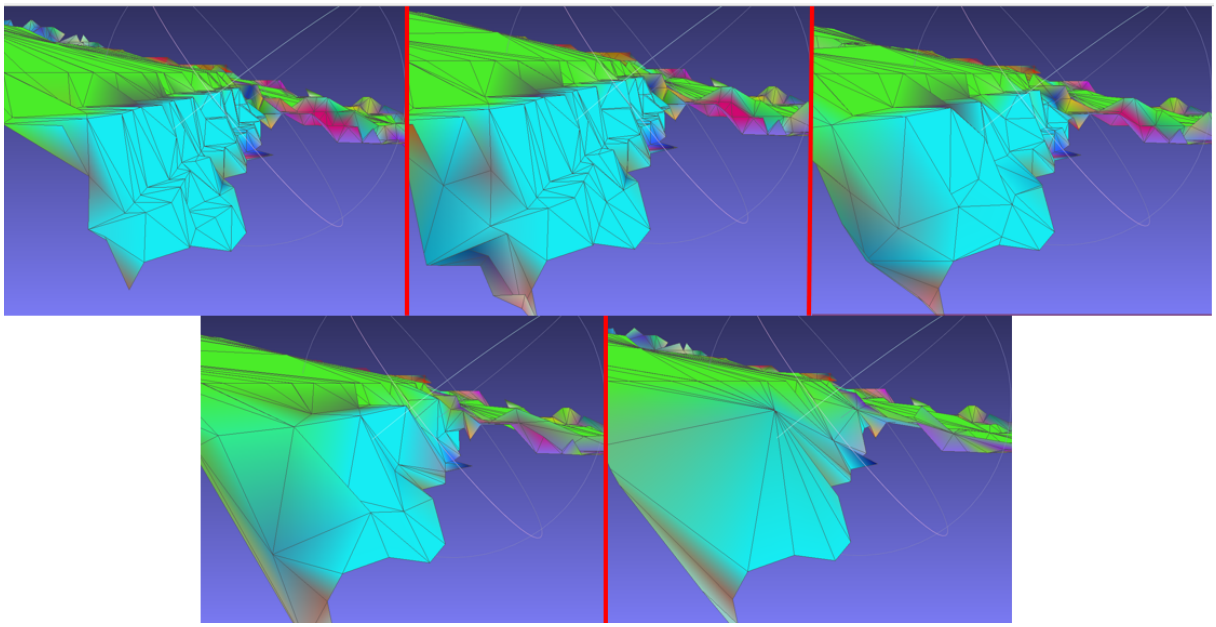


Figure 20: Visualization of curved surface simplification under different vertex retention thresholds (from top-left to bottom-right: 100%, 80%, 60%, 40%, and 20%)

5.3.2 Global Evaluation on Full-Scale Architectural Meshes

In this chapter, we take the complete building point cloud model as input and apply both our proposed region-adaptive vertex simplification algorithm and the traditional Quadric Edge Collapse (QEC) algorithm to simplify the reconstructed architectural mesh. The final results are compressed to the same number of mesh faces for a fair comparison. Based on this setup, we systematically compare the geometric fidelity of the two methods across various structural regions—such as walls, door frames, and windows—to comprehensively evaluate their ability to preserve architectural details under extreme simplification conditions.

Figure 21 (left) presents the building mesh obtained via our proposed extreme vertex simplification scheme, the center shows the original unsimplified mesh, and the right displays the mesh simplified to the same face count using the standard edge collapse algorithm. From this figure results show that our proposed vertex simplification algorithm exhibits superior structural preservation under extreme simplification conditions. For planar regions such as walls, the algorithm retains only a minimal number of internal vertices, effectively achieving the goal of vertex count reduction. In contrast, regions with significant geometric features—such as doors, windows, and their frames—undergo only light simplification to preserve critical structural details as much as possible. Our improved QEC algorithm prioritizes the removal of redundant vertices that have minimal impact on the overall geometry, thereby reducing the number of mesh faces while preserving the original structure to the greatest extent.

However, when applying the traditional QEC algorithm with a uniform simplification strategy to reduce the mesh to the same 10,685 faces as our method, critical geometric features—such

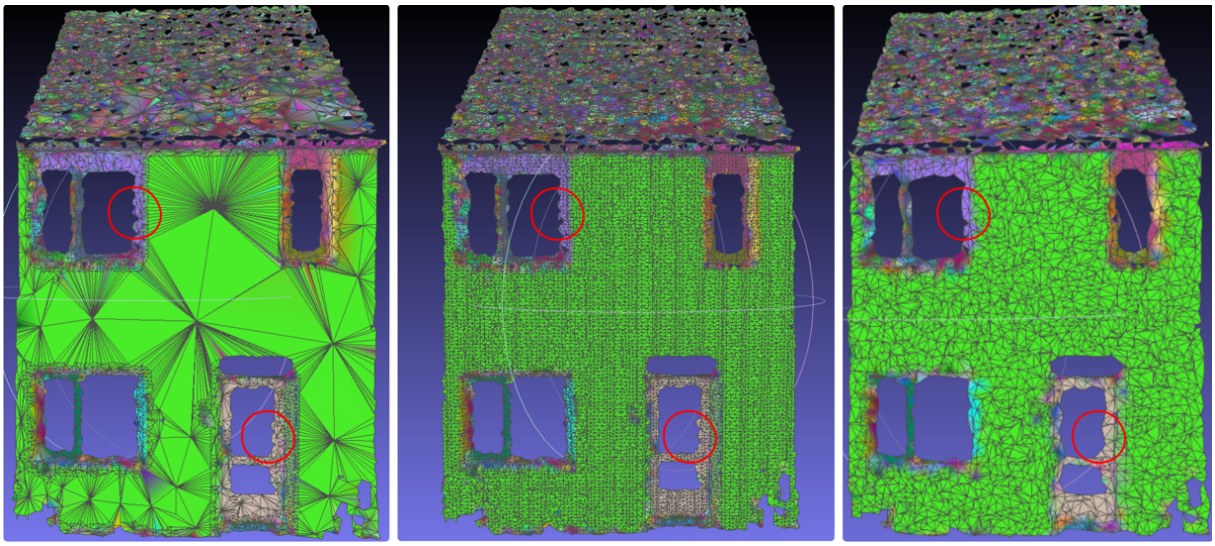


Figure 21: Comparison of mesh simplification results. *Left:* Mesh simplified to 10,685 faces using our improved vertex simplification algorithm, effectively preserving critical geometric details. *Middle:* Original unsimplified mesh model. *Right:* Mesh simplified to 10,685 faces using the traditional Quadric Edge Collapse algorithm, where coarse triangular facets distort the geometries of windows and door frames. Red circles highlight the differences in geometry preservation between the conventional algorithm and our method.

as wall corners, door frames, and window frames—suffer from severe deformation, resulting in a simplified mesh that differs significantly from the original model. This demonstrates that traditional simplification methods tend to cause notable damage to geometric integrity when subjected to extreme vertex reduction. In contrast, our vertex simplification strategy is capable of preserving the original mesh geometry even in highly detailed regions, exhibiting greater fidelity and robustness.

In the two additional comparison figures, Figure 22 detailed differences in the window frame region demonstrate our vertex simplification strategy's superior performance in preserving geometric edges, as it more accurately retains critical structural features and prevents noticeable edge distortion compared to traditional methods. The Figure 23 compares the internal surface geometry at the recessed junction between the door frame and wall, further demonstrating that our strategy not only precisely preserves sharp edges but also maintains fine surface details, avoiding the unnatural flattening or distortion that can result from aggressive vertex reduction.

6 Conclusion

This paper presents an end to end lightweight mesh reconstruction and simplification framework for large scale real world architectural point clouds, with the following key innovations and contributions. First, we estimate normals and curvature and configure an unsupervised Region

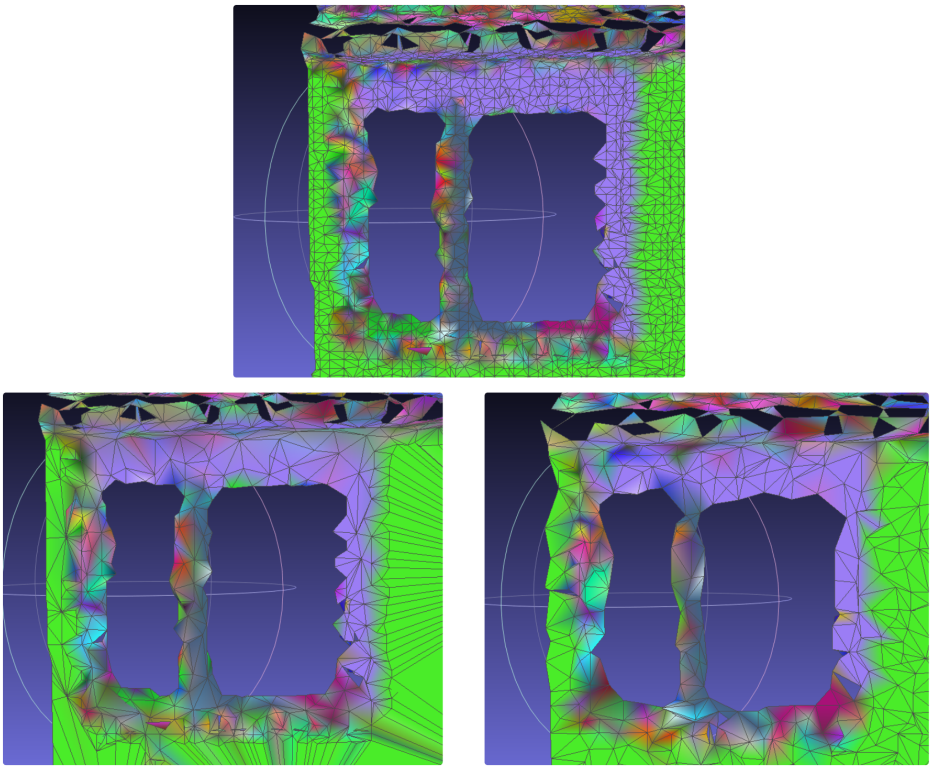


Figure 22: Additional validation of our advanced vertex simplification strategy. *Top:* Original mesh. *Bottom-left:* Mesh simplified to the same face count using our proposed method, preserving fine geometric details. *Bottom-right:* Mesh simplified to the same face count using the traditional Quadric Edge Collapse algorithm, showing pronounced distortion in critical regions.

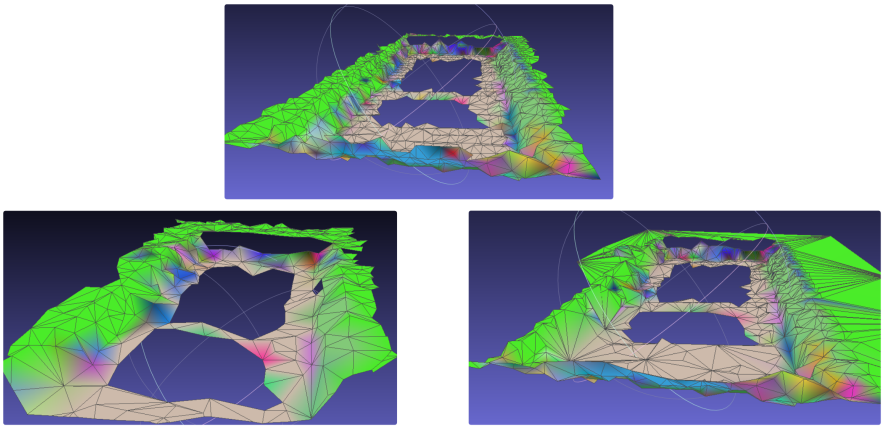


Figure 23: Comparison of mesh simplification results (layout as before). Top: original mesh; bottom-left: mesh simplified to the same face count using our proposed vertex simplification strategy, preserving both edge structures and surface geometric details; bottom-right: mesh simplified to the same face count using the traditional Quadric Edge Collapse algorithm, where critical edge and surface features are distorted.

Growing algorithm with a normal deviation threshold and a spatial distance threshold. This automatically segments planar or smoothly curved surfaces into large clusters while grouping high curvature sharp feature regions into small clusters. Cluster size discrimination combined with Kd tree neighborhood searches then accurately extracts key regions such as window frames, door frames, and corners to guide subsequent simplification. Second, to robustly process radar point clouds that contain noise and open boundaries, we adopt the Ball Pivoting algorithm for mesh reconstruction, ensuring topological coherence and fidelity to the original data. Our core innovation is a two stage differentiated Quadric Edge Collapse decimation strategy. In the first stage, internal vertices of planar and high curvature regions are aggressively simplified to ten percent and seventy percent of their original counts, respectively. In the second stage, the preserved boundary vertices in the reassembled mesh undergo a mild ninety percent threshold simplification. This approach achieves extreme vertex and face reduction while effectively preserving critical structural details. Experimental comparisons with traditional simplification schemes show that our method significantly outperforms them in geometric fidelity and visual detail retention at the same reduction levels, providing an efficient and reliable solution for lightweight storage, transmission, and rendering of large scale architectural point clouds. Future work will explore automatic parameter adaptation and multi resolution mesh generation to further enhance the framework's generalization and real time performance.

7 Future Work

Although the proposed method delivers impressive practical results, it has notable limitations. First, the simplification thresholds are not derived from rigorous numerical optimization or automated processes but are instead chosen by visually inspecting reconstruction results across different architectural point clouds—an approach that lacks scientific grounding and repeatability. Second, although metrics such as the Hausdorff distance are widely used in theory to quantify geometric deviation, they prove unreliable for mesh fragments generated from real radar point clouds: even nominally flat regions display slight undulations in their triangular facets, causing these metrics to be overly sensitive to noise and unable to accurately reflect the true geometric changes before and after simplification.

Future work will focus on integrating automated parameter adaptation and developing more robust geometric-difference metrics to overcome these challenges. We also aim to achieve more precise segmentation of surface patches with varying curvature. Instead of simply classifying surfaces into planar and curved categories, we seek to assign a specific semantic label to each individual surface patch. This would enable targeted extraction and further optimization operations for different surface types, thereby supporting more advanced geometric processing and structural analysis.

References

[1] Fausto Bernardini, J. Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *Visualization and Computer Graphics, IEEE Transactions on*, 5:349 – 359, 11 1999.

[2] Michael M. Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In Alla Sheffer and Konrad Polthier, editors, *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, volume 256 of *SGP '06*, pages 61–70, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[3] Ravi Peters, Balázs Dukai, Stelios Vitalis, Jordi van Liempt, and Jantien Stoter. Automated 3d reconstruction of lod2 and lod1 models for all 10 million buildings of the netherlands, 2021.

[4] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 163–169, New York, NY, USA, 1987. Association for Computing Machinery.

[5] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, page 61–70, Goslar, DEU, 2006. Eurographics Association.

[6] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017.

[7] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017.

[8] Yunhan Yang, Yukun Huang, Yuan-Chen Guo, Liangjun Lu, Xiaoyang Wu, Edmund Y. Lam, Yan-Pei Cao, and Xihui Liu. Sampart3d: Segment any part in 3d objects, 2024.

[9] Mi Yan, Jiazhao Zhang, Yan Zhu, and He Wang. Maskclustering: View consensus based mask graph clustering for open-vocabulary 3d instance segmentation, 2024.

[10] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds, 2020.

[11] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation, 2022.

[12] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, April 2021.

[13] Ting Han, Yiping Chen, Jin Ma, Xiaoxue Liu, Wuming Zhang, Xinchang Zhang, and Huajuan Wang. Point cloud semantic segmentation with adaptive spatial structure graph transformer. *International Journal of Applied Earth Observation and Geoinformation*, 133:104105, 2024.

[14] Michael Garland and Paul S. Heckbert. *Surface Simplification Using Quadric Error Metrics*. Association for Computing Machinery, New York, NY, USA, 1 edition, 2023.