Sensors and Materials, Vol. 36, No. 12 (2024) 5535–5557
MYU Tokyo

S & M 3884

Research on the Technology of 3D Model Reconstruction of Irregular Buildings Based on Point Clouds

Yong Wang,[1] Chao Tang,[1*] Ming Huang,[2] Haipeng Zhu,[3] and Yuan Gao[2]

[1]Beijing Urban Construction Survey and Design Institute Co., Ltd., Chao Yang 100020, China
[2]School of Mapping and Urban Spatial Information, Beijing University of Civil Engineering and Architecture, Beijing 102616, China
[3]Zhejiang Xinnuorui Marine Technology Co., Ltd., Zhe Jiang 315336, China

(Received September 20, 2024; accepted December 5, 2024)

Keywords: point cloud, planar element extraction, 3D modeling, irregular buildings, 3D polyhedron

Addressing the limitations of existing reconstruction technologies, which are constrained by the Manhattan model assumption and exhibit insufficient applicability to irregular building shapes, as well as issues related to low model accuracy, dense triangular mesh faces, and weak robustness to chaotic and incomplete point clouds, we propose a 3D reconstruction technology specifically designed for irregular buildings, free from the constraints of the Manhattan model assumption. The proposed method includes a grid outlier removal technique based on eight-connected domains, a one-point Random Sample Consensus (RANSAC) method utilizing single points and their normal vectors for random plane segmentation to extract building planar structural elements, and a technique for 3D reconstruction of irregular buildings based on the selection of 3D polyhedral mesh faces. Validation with various datasets demonstrates that this method offers significant advantages over other reconstruction approaches in terms of model triangulation lightweightness and topological structure correctness.

1. Introduction

With the introduction of the "Realistic 3D China" strategy by the Ministry of Natural Resources, the demand for digital city construction has become increasingly urgent. In addition to comprehensive 3D map information, the creation of 3D landscape models for entire cities has also become a significant requirement. As fundamental elements of urban landscapes, building models have attracted widespread attention from both academia and industry in fields such as urban planning, natural disaster management, virtual reality, and real-time emergency response.(1) The automated construction of building models has been a research hotspot both domestically and internationally, primarily focusing on the fields of surveying and computer vision. Research in the field of surveying often concentrates on the 3D reconstruction of buildings themselves, excluding elements such as terrain, trees, and other features within the scene.(2) This study is similarly focused on this area.

*Corresponding author: e-mail: 15801186470@163.com
https://doi.org/10.18494/SAM5371

ISSN 0914-4935
https://myukk.org/

In recent decades, the rapid development of data acquisition technologies such as Light Detection and Ranging (LiDAR), RGB-D cameras, Structure from Motion (SfM), and Multi-View Stereo (MVS) has made the 3D point cloud capture of urban scenes and buildings more efficient and precise. With the continuous acquisition of high-density and high-precision point clouds, it has become possible to reconstruct more detailed and complex building models.[3] Nevertheless, owing to the inherent characteristics of point clouds and the spatial complexity of building structures, the geometric diversity and occlusion issues present significant challenges to the reconstruction of building surfaces in complex real-world scenarios. Thus, the automatic reconstruction of 3D building models from point clouds remains an urgent problem to be solved.

The Manhattan-world assumption is a common model for building scenes, originating from the box-like modernist architectural style of New York's Manhattan area. This model effectively expresses the structural characteristics of man-made buildings in the form of a 3D grid, describing the orthogonal structure of urban and interior environments. In the Manhattan-world model, the building model is aligned with an orthogonal coordinate system, where the corresponding X, Y, and Z axes are mutually perpendicular. This assumption applies to structurally regular building forms. The irregular buildings evaluated in this study primarily refer to those whose facades and roof sections consist of irregular planar surfaces, excluding curved surfaces and other arcuate structures.

Traditional building model construction methods typically rely on manual modeling using commercial software such as Rhino, Revit, and SketchUp. These methods generally involve first constructing basic building units, such as planes, cylinders, and rings, within the software, followed by the setting of parameters for angles, sizes, and orientations to complete the model reconstruction. This approach is highly dependent on the experience of the practitioner, and the manual modeling process is often tedious, time-consuming, and resource-intensive, resulting in high costs. Therefore, the demand for automated building modeling has become increasingly urgent.[4]

Current building model construction methods can be broadly categorized into two main types: image-based 3D reconstruction and LiDAR point cloud-based 3D reconstruction. Image-based 3D reconstruction typically involves capturing the target object from multiple angles using drones, satellites, or cameras, extracting the building's geometric relationships and 3D information from the images, and reconstructing its surface structure to complete the model. This method essentially converts images into point clouds, which are then processed through reconstruction techniques to generate a mesh that represents the building's surface model.[5]

However, whether the point clouds are generated from images or acquired via LiDAR scanning, they inevitably contain noise, missing information, and non-ideal structures. Most of the popular reconstruction algorithms today are limited to building model reconstruction under the Manhattan-world assumption and show weak robustness to cluttered point clouds. Although some algorithms can reconstruct irregular building models, they often suffer from low model accuracy, inaccurate topological structures, and dense triangular meshes. Consequently, the challenge of constructing irregular building models with lightweight triangular meshes and accurate topological structures from point clouds remains substantial.

To address the issues in current 3D reconstruction methods, such as dense triangular mesh patches, erroneous model topology, poor robustness of point clouds, and the failure to account for irregular building shapes on roofs, in this paper, we propose a 3D model reconstruction method for irregular buildings based on point clouds. The primary research focuses are as follows:

(1) In this paper, we propose an improved eight-connected domain mesh outlier removal method. This method is based on the eight-neighbor algorithm in digital image processing and has been modified to suit 3D point cloud processing.
(2) Building structures typically comprise complex planar elements that form the adjacency relationships and geometric constraints of the architecture. Here, we introduce a random planar segmentation method based on a single point and its normal vector—one-point Random Sample Consensus (RANSAC). Compared with the traditional RANSAC segmentation method, this approach demonstrates better robustness and efficiency when processing point cloud data containing outliers and noise, especially in small local regions, thereby laying a solid foundation for subsequent model reconstruction.
(3) Building on the extraction of planar structural elements, in this paper, we first construct a superset of the target object by extracting the intersecting relationships between planes, treating the object's surface as a 3D polyhedron composed of a series of candidate patches. Then, by constructing a binary linear programming equation and using the SCIP solver to optimize the objective function, redundant patches are eliminated, resulting in an approximate polygonal model of the building.

2 Planar Element Extraction of Buildings Based on Point Clouds

2.1 Point cloud filtering processing

Buildings possess complex shapes and varying surface reflectivity, and the point cloud data obtained using LiDAR scanners often contain a large number of outliers. These outliers are unavoidable byproducts of 3D scanning, typically manifesting as disordered, noisy data with inconsistent local density, and surface shapes with geometric discontinuities and sharp features.[6] Therefore, effectively removing outliers from point clouds is the first problem that must be addressed in point cloud processing.

Common point cloud filtering and denoising algorithms include bilateral filtering,[7] statistical outlier removal,[8] and radius outlier removal.[9] Bilateral filtering is primarily used to smooth small-scale noise in point clouds, effectively reducing noise on the 3D model's surface while preserving geometric features, but it is only suitable for denoising ordered point clouds. In statistical outlier removal, the average distance of a point to its neighboring points is calculated, assuming this distribution follows a Gaussian distribution, and then points with distances greater than a threshold are removed on the basis of the calculated mean and standard deviation. In radius outlier deletion, it is assumed that there are at least enough points within a certain radius; otherwise, the point is deleted. Although these three filtering methods are simple, straightforward, and easy to understand, their effectiveness in removing outliers is limited.

In contrast, DBSCAN[10] is a density-based clustering algorithm that performs well in removing outliers but has a higher algorithmic complexity. To more effectively extract point cloud data of building structures and filter out outliers outside the building, in this paper, we propose a grid-based outlier removal method using an eight-connected domain approach to more efficiently remove anomalies in point clouds.

In this method, we employ the eight-connected domain algorithm for outlier removal in point clouds. First, the acquired building point cloud is partitioned into a 2D grid on the basis of the X and Y coordinates, with the grid cell size determined by the resolution of the point cloud data. In Fig. 1(a), the red box highlights the outliers in the original building point cloud. In the experiment, this parameter is set according to the voxel size during the downsampling process. If the number of points in a grid cell is greater than 0, the grid cell is labeled 1. All grid cells are then traversed to create a binary grid map, and outlier filtering is performed using the eight-neighbor algorithm. As shown in Fig. 1(b), grid cells labeled 1 (indicated by red arrows) are selected as seed points, and grid cells are searched for in eight surrounding directions. Each detected grid cell is grouped into the same region until all grid cells are processed. The final binary grid map is converted into a grid region map, as shown in Fig. 1(c). “o” represents the

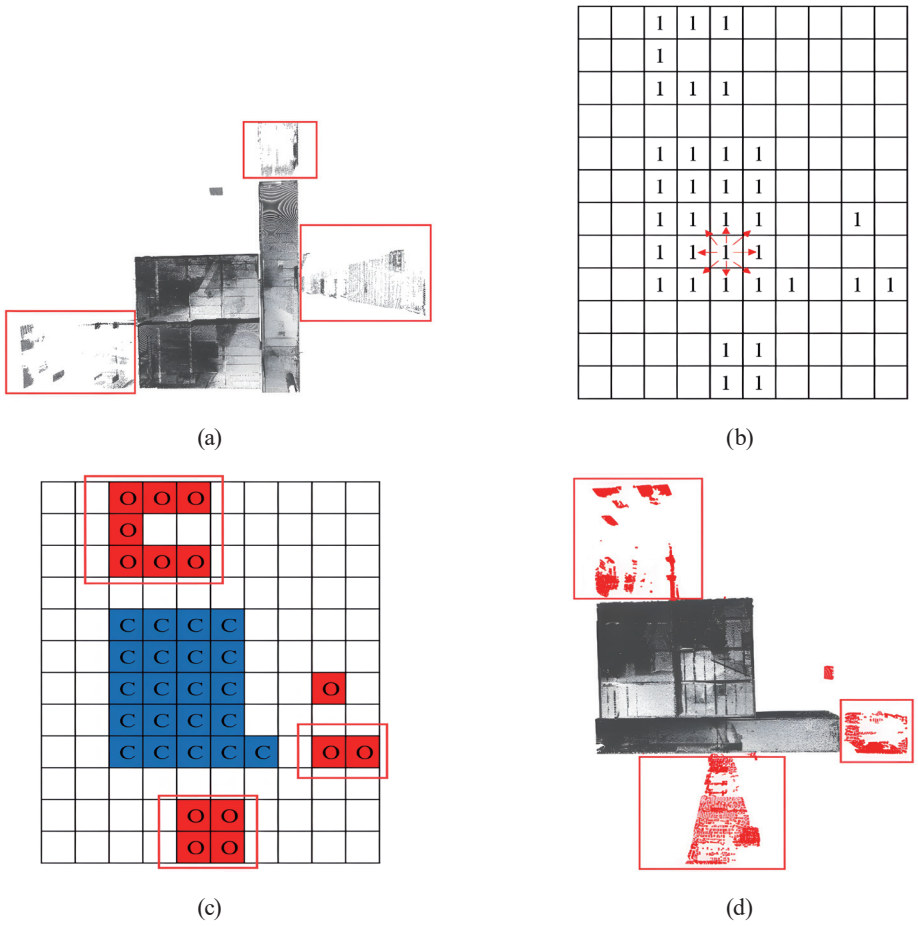


Fig. 1. (Color online) Outlier removal algorithm flow for point cloud. (a) Raw point cloud visualization, (b) binary grid map, (c) grid area map, and (d) building point cloud after outlier removal.

filtered outlier grid cells, and "c" represents the retained building structure grid cells. An adaptive threshold is calculated on the basis of the maximum difference in the number of sorted grid cells in each region between outlier and building structure point cloud areas. Since the number of grid cells in the building structure point cloud exceeds the threshold, the outlier grid regions are removed, as shown in Fig. 1(d). This approach retains the building point cloud with anomalies removed.

2.2 Point cloud simplification and resampling

Point cloud resampling is divided into upsampling and downsampling. Upsampling is conducted to increase the number of points in the point cloud through interpolation when the data volume is low to restore missing surface parts. Downsampling simplifies the point cloud by constructing a 3D voxel grid and replacing other points within the grid with the centroid of the grid, thus reducing the data volume. In this study, we choose downsampling for point cloud simplification, as the input point clouds are typically unevenly distributed and have missing information. Downsampling can help with point cloud repair to some extent. Although point cloud simplification algorithms often face the issues of long runtime and accuracy loss, the octree-based downsampling method effectively addresses these issues.

The octree-based point cloud downsampling method recursively divides a 3D point cloud, as shown in Fig. 2. The algorithm for constructing an octree for point cloud data is as follows:

(1) Construct a bounding box around the point cloud data and use this box as the root node.
(2) Add the known point cloud data to the root node. When the number of points within the root node exceeds a set threshold, the root node is subdivided into eight child nodes, and the root node's attribute is set to False (initially True). Additionally, determine which child node the added point cloud data belongs to.
(3) Continue adding point clouds to the root node and determine which child node the new data points belong to. When the number of data points in a child node exceeds the threshold, that

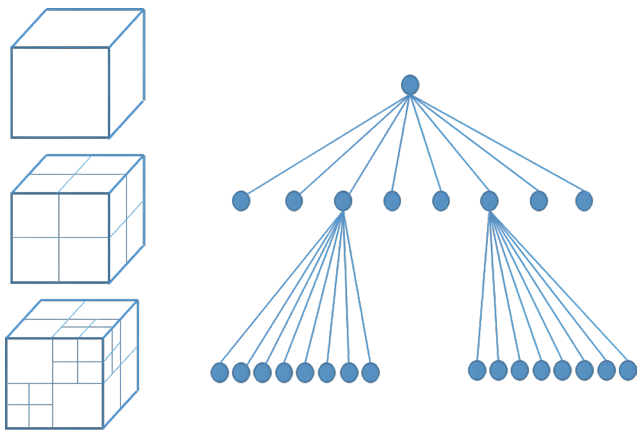


Fig. 2. (Color online) Schematic of point cloud octree segmentation.

child node is further subdivided into eight child nodes, its attribute is set to False, and the point cloud's location in these child nodes is computed.

(4) Repeat the process in step 3 until all points have been traversed, completing the tree construction.

The general process for simplifying 3D point clouds is as follows: (1) Use the octree to establish the topological structure of spatially scattered point clouds and search for neighborhoods. (2) Set the octree levels and filter all nodes. (3) Calculate the centroid of each intermediate node to replace other points within the node, thus completing the point cloud simplification and compression process.

2.3 One-point RANSAC plane segmentation method

Plane segmentation is the process of grouping point cloud data into multiple homogeneous regions with similar attributes, enabling the identification and construction of 3D building scene models by dividing complex building point clouds into simpler geometric primitives.[11] Unlike images or triangulated irregular network (TIN) models, point clouds do not directly represent topological information. Existing image and TIN segmentation methods face two main challenges: firstly, converting irregular 3D point clouds into other models often leads to information loss, particularly evident in distance-based image algorithms; secondly, the conversion of large volumes of point data incurs high computational costs, which is especially prominent in large-scale LiDAR applications. Therefore, effectively extracting planar features from building point cloud data is a fundamental step in 3D reconstruction.[12] To effectively extract planar features from building point clouds, in this section, we introduce an improved one-point RANSAC plane segmentation method based on sampled point clouds. The input for this method is high-density scattered point cloud data generated by LiDAR scanning or imagery.

By sampling the input point cloud set $p = \{p_1, p_2, p_3, ..., p_N\}$, $p_i \in R^3$, where N represents the total number of points in the cloud, we obtain a sampled subset of point cloud data. Normals are then computed using this subset of point cloud data. Given the high cost of accurately estimating normals for all points, a hypothesis plane can be defined by a single point and its normal vector. Therefore, only a subset of the point cloud data is used for normal estimation, and the hypothesis is validated using all point cloud data.

With a given sampling rate α_s, a sampled subset of point cloud data $p_s = \{p_1^s, p_2^s, p_3^s, ..., p_{N_s}^s\}$, $p_i^s \in R^3$ is obtained, where N_s is the cardinality of p_s. The point cloud normals are estimated by performing SVD on the neighborhood of the sampling points. The selected sampling point is represented as $p_i = \{p_{i,x}, p_{i,y}, p_{i,z}\}$, and its corresponding normal vector is denoted as $n_i = \{n_{i,x}, n_{i,y}, n_{i,z}\}$. The K-nearest neighbors of the sampling point p_i are found by a K-D tree search, represented as $Q_i = \{q_{i1}, q_{i2}, q_{i3}, ..., q_{ik}\}$ $q_{ik} \in p$ and $q_{ik} \neq p_i$. The choice of parameter K is crucial for normal estimation. For each sampling point, the matrix M_i is computed as shown in Eq. (1), and the eigenvector corresponding to the smallest eigenvalue is used as the normal vector for that point.[13]

$$M_i = \sum_{q_{ij} \in Q_i} e^{\frac{-\left\|q_{ij}-p_i\right\|_2^2}{2\sigma^2}} \frac{\left(q_{ij}-p_i\right)\left(q_{ij}-p_i\right)^T}{\left\|q_{ij}-p_i\right\|_2^2} \tag{1}$$

The weight coefficient $e^{\frac{-\left\|q_{ij}-p_i\right\|_2^2}{2\sigma^2}}$ helps reduce the effect of the neighboring points p_i that are far from the sampling point. The normals for all sampling points are normalized.

After computing the normal vectors of the point cloud, in this paper, we use a point and its corresponding normal vector to define a plane. For a given set of sampling points p_s, a plane is defined by randomly selecting a sampling point with a normal vector and calculating the distance of all other sampling points to this plane. Points with distances less than the threshold θ_h are classified as belonging to the same plane, and the threshold θ_N is set to determine the minimum number of points required to define the plane. This process is iterated to find the plane model with the maximum number of inliers. The number of iterations N_{iter} is initially set to the total number of points in the point cloud. p is the probability that at least one randomly sampled point cloud is not an outlier, whereas e is the probability that a point is an outlier among all points; this value is updated on the basis of the number of inliers found in each iteration.

$$N_{iter} = \frac{\log\left(1-p\right)}{\log\left(1-\left(1-e\right)\right)} \tag{2}$$

Since only one true inlier needs to be selected, the number of iterations for plane detection is certainly less than the number required for randomly choosing three inlier sampling points. Once the largest plane is detected, the plane's normal vector is re-estimated for all inliers using SVD.[14]

To detect all planes, the one-point RANSAC method is applied multiple times. After detecting a plane, it is removed from the total point cloud, and the remaining point cloud is further analyzed until there are not enough points (below a threshold) to form a plane. To extract basic semantic scenes, the detected planes are categorized into three types: vertical planes, horizontal planes, and other planes. The sampled point clouds are divided into these three groups on the basis of the estimated normal vectors, and the one-point RANSAC plane detection method is applied to each group separately. The detection results may include some small plane segments, which will be merged in subsequent processing. By performing coplanarity tests on each pair of planes, we can merge these small plane segments.

3 Building Reconstruction Based on 3D Polyhedron Face Selection

3.1 3D polyhedron face selection algorithm

The algorithm presented in this paper aims to convert the input point cloud data into a lightweight polygonal 3D building model. Figure 3 illustrates the steps for transforming point clouds into a 3D polyhedron surface model, divided into two main phases:

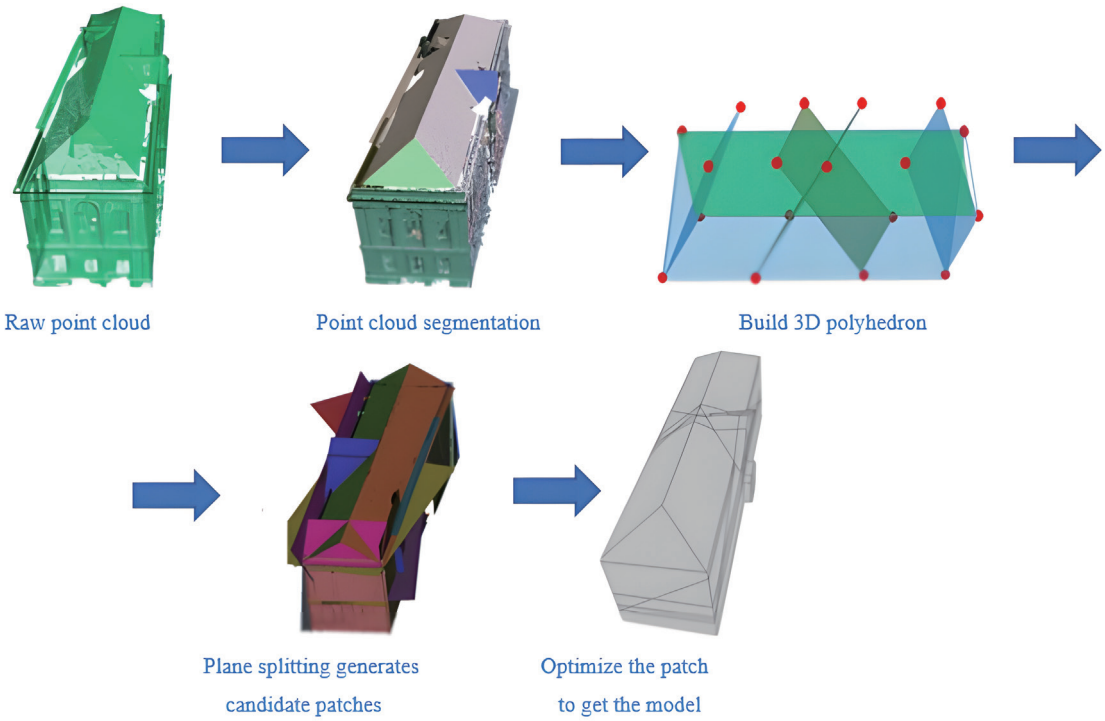


Fig. 3. (Color online) Steps for transforming point clouds into a 3D polyhedron surface model.

(1) Generation of 3D Polyhedron: Use the one-point RANSAC plane segmentation method to extract a collection of planar objects from the point cloud. Since the detected planes may contain outliers, noise, or missing data, the plane set is refined through iterative merging and fitting of new planes. The planes are then clipped by extending the bounding box of the point cloud, and the intersection points of the clipped planes are calculated to construct the 3D polyhedron.
(2) 3D Polyhedron Optimization: Select an optimal subset of the polyhedron faces to approximate a manifold, sealed polygonal surface model. The face selection problem is formulated as a binary linear programming problem, where the objective function includes point cloud data fitting, point cloud coverage, and model complexity, with constraints to ensure the tightness of the final model. A global optimization method is employed to simultaneously optimize plane segmentation and face selection, ensuring correct topology, lightweight triangular mesh faces, and improved robustness to noisy point clouds.

3.2 Iterative plane refinement

In this paper, we utilize an improved iterative plane refinement algorithm proposed by Li *et al.*(2016)[15] to refine the initially segmented plane set. Specifically, in this method, we calculate the angle between the supporting planes of any pair of planes. Starting from the plane pair with the smallest angle, denoted as the combination (π_0, π_1), the method tests whether the following

conditions are met: 1. The angle between the two planes is smaller than a preset threshold θ_t, denoted as $angle(\pi_0, \pi_1) < \theta_t$. 2. The distance between the centroids of the point clouds of the two planes is less than a given threshold d_t. 3. There must be more than a certain number N_t of point clouds that simultaneously lie on both supporting planes. Then, a new plane element is fitted from the merged point clouds using the least squares method.[16] This process is repeated until no further plane pairs meet the merging criteria. The final result is a refined set of segmented planes, with the number of segmented planes significantly reduced.

3.3 Construction of 3D polyhedrons with planar segment constraints

A 3D polyhedron is composed of vertices V, edges E, and face patches F, along with the mapping relationships among them, ensuring the integrity of their combinations. As illustrated in Fig. 4, the surface categories of the 3D polyhedron can be restricted to oriented 2D manifolds—either with or without boundaries. Each edge E consists of two half-edges with opposite orientations. The vertices V represent points in 3D space, edges E are straight line segments connecting two endpoints, and face patches F are plane polygons without holes, defined as closed loops formed by a sequence of half-edges along their boundaries. A 3D polyhedron can have holes but must be enclosed by at least two face patches, as a single face patch cannot form a hole. The half-edges around each vertex are typically arranged in a clockwise direction. In this paper, we constrain face patches to be simple plane polygons without holes and define the boundaries of the polyhedron as oriented 2D manifolds to ensure that each face patch is continuous and oriented. Additionally, each edge must be associated with exactly two face patches, each edge must have two distinct endpoints, and each vertex must be connected to three face patches. These conditions are crucial for ensuring the tightness and consistency of the model.[17]

3.3.1 Construction of the minimum bounding box of the point cloud

The first step in constructing the 3D polyhedron is to compute the bounding box of the original input building point cloud. In this paper, the axis-aligned bounding box (AABB) is

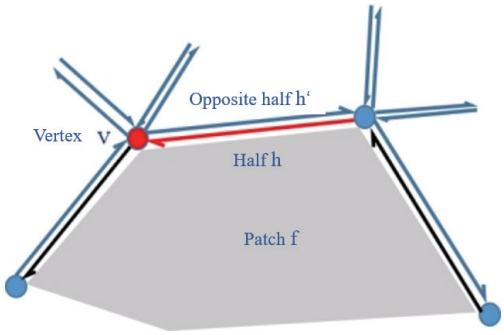


Fig. 4. (Color online) Constituent elements of a 3D polyhedron, points, edges, and patches.

adopted. This type of bounding box is defined as the smallest six-faced polyhedron whose edges are aligned with the coordinate axes. The advantage of AABB lies in its simplicity, requiring only six variables to describe it, which leads to minimal memory usage while achieving a balance between efficiency and spatial coverage. Compared with bounding spheres and oriented bounding boxes, AABB finds extensive application across a variety of computational tasks.

For the input point cloud $p = \{p_1, p_2, p_3, ..., p_N\}$, $p_i \in R^3$, the important property of the AABB is to satisfy the condition that any point coordinates $p_i(x, y, z)$ are contained within $x\_min \le x \le x\_max$, $y\_min \le y \le y\_max$, $z\_min \le z \le z\_max$. Therefore, it is necessary to identify two key vertices, $p_{min}(x\_min, y\_min, z\_min)$ and $p_{max}(x\_max, y\_max, z\_max)$. Additionally, owing to potential point cloud errors, to better ensure that the point cloud is encompassed within the bounding box, in this paper, we expand the edge lengths of the bounding box by the radius r, thus:

$$r = \sqrt{(x\_max - x\_min)^2 + (y\_max - y\_min)^2 + (z\_max - z\_min)^2}. \tag{3}$$

The boundaries of the expanded minimum bounding box are defined by the points p_{min} and p_{max} as follows.

$$\begin{aligned} p_{max}.x\_max &= x\_max + r * 0.5f \\ p_{max}.y\_max &= y\_max + r * 0.5f \\ p_{max}.z\_max &= z\_max + r * 0.5f \end{aligned} \tag{4}$$

The minimum bounding box of the original point cloud is shown in Fig. 5.

3.3.2 Spatial organization of 3D polyhedrons

The spatial organization of a 3D polyhedron can be regarded as a mesh in a graph data structure, consisting of vertices, edges, and face patches. Each edge is made up of two half-

Fig. 5. (Color online) Minimum external bounding box of original point cloud.

edges oriented in opposite directions. Figure 6 illustrates the composition and relationships within the graph data structure. For the bounding box of the original point cloud, its graph structure is constructed by sequentially adding the eight vertices a_1, a_2, a_3, a_4, b_4, b_3, b_2, and b_1 in a counterclockwise direction, along with six face patches. The face patches are constructed in the following order: $f_1(a_1, a_2, b_2, b_1)$, $f_2(a_2, a_3, b_3, b_2)$, $f_3(a_3, a_4, b_4, b_3)$, $f_4(a_4, a_1, b_1, b_4)$, $f_5(a_1, a_2, a_3, a_4)$, and $f_6(b_1, b_2, b_3, b_4)$. This construction ensures that each vertex is associated with three face patches and each edge is shared by two face patches, thereby guaranteeing that each edge in the mesh is shared by no more than two face patches. This ultimately ensures that the surface of the reconstructed model possesses manifold characteristics.

3.3.3 Construction of 3D polyhedrons by clipping support planes with bounding boxes

In the process of constructing a 3D polyhedron, we first clip the point cloud data to obtain a superset of the original building point cloud surface. The specific steps are as follows:

(1) Utilize the bounding box of the point cloud to clip all support planes obtained from the iterative refinement of the point cloud. Calculate the intersection points of each support plane with the edges of the point cloud's bounding box, and save these intersection points, edges, and face patches along with their relationships. Using these intersection points and face patches, form a 3D polyhedron that includes the extended patches of the point cloud.

(2) As shown in Fig. 7, the support planes of the face patches are infinite. The actual support plane of a face patch is considered a superset of the face patch. For example, the intersection points where the support plane of the point cloud patch $f_1(a_1, a_4, s)$ intersects with the bounding box are used to construct extended face patches. If the number of intersection points exceeds three, connect these points in the convex hull to form the extended face patches. If an intersection point is in the middle of the bounding box's edge, assign the corresponding two face patches of the bounding box and the extended face patch itself as the associated patches of that intersection point. If the intersection point is at the edge of the

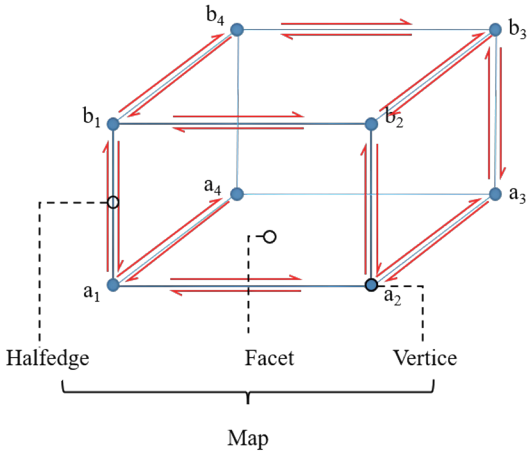


Fig. 6. (Color online) Spatial organization of 3D polyhedron.

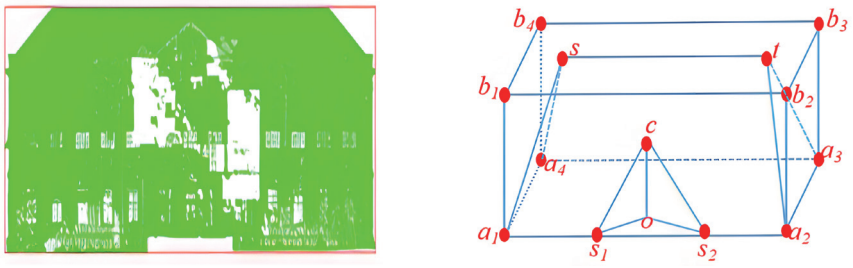


Fig. 7. (Color online) Simplified representation of original point cloud model.

bounding box, directly assign the face patch corresponding to the bounding box's endpoint to that intersection point. Each intersection point is assigned an index to avoid redundant calculations.

(3) For each newly generated extended face patch, first associate it with its corresponding face patch. Then, calculate the distances from the two intersection points p_1 and p_2 of the clipping edge to the six planes of the bounding box. If the distance is less than the set threshold ($1e^{-6}$), the intersection point is associated with the corresponding face patch. If the number of face patches associated with the clipping edge is not equal to 2, it is considered a clipping error.

Ultimately, we obtain the intersection points, edges, and face patch associations between the support planes of the point cloud and the bounding box, preparing for the next step of mutual segmentation of the extended point cloud face patches.

To optimize the 3D polyhedron and obtain a more accurate building structure model, we need to process and further optimize the extended point cloud face patches. The specific process is as follows:

(1) Calculate the intersection of all other extended face patches with the support plane of any given face patch. To better represent the segmentation process, use the two endpoints of the edges of the extended face patches to compute the distances to the support plane. The equation of the support plane can be expressed as $ax + by + cz + d = 0$, and the distance formula can be expressed as

$$d_t = \sqrt{\frac{ax + by + cz + d}{a^2 + b^2 + c^2}}\,. \tag{5}$$

(2) Next, the extension faces are divided by intersecting extension faces, generating new candidate faces. Note that after each cut, the original extension face no longer exists; it is split into multiple candidate face parts, each of which is further divided by another extension face until the splitting condition is no longer met.

(3) By calculating edge and face splitting, we can obtain the intersection points of the splits, as well as the relationships between new half-edges and new faces. Each candidate face (with 1 associated plane), intersection points (with 3 associated planes), and edges (with 2 associated planes) are generated from the mutual splitting of planes. Thus, by splitting all faces, the final 3D polyhedron is composed of individual candidate faces.

3.4 Construction of binary linear programming equations for face optimization

In the previous section, candidate faces were generated, and in this section, we focuse on selecting the subset of candidate faces that best represent the actual structure of the building while ensuring that the selected faces form a manifold and closed polygonal surface model. The optimization of 3D polyhedron faces can be formulated as a binary linear programming problem.[18] Our objective function consists of three energy coefficients: point cloud fitting, face coverage, and model complexity.

3.4.1 Point cloud fitting

Point cloud fitting is used to evaluate the matching quality between the faces and the point cloud, incorporating a confidence concept. Mathematically, point cloud fitting is defined as the percentage of confidence-weighted points not involved in the final model reconstruction, i.e., the percentage of point clouds not used in the reconstruction.

$$E_f = 1 - \frac{1}{|p|}\sum_{i=1}^{N} x_i \cdot s\left(f_i\right) \tag{6}$$

Here, $|p|$ is the total number of point clouds in the input point cloud set p, and $s(f_i)$ represents the quality of the point cloud, defined by considering the effect of each point cloud's local neighborhood. It is determined by the number of point clouds that compose the face.

$$s\left(f\right) = \sum\nolimits_{p,f|dist\left(p,f\right)<\varepsilon}\left(1 - \frac{dist\left(p,f\right)}{\varepsilon}\right)\cdot conf\left(p\right) \tag{7}$$

The confidence coefficient $conf(p)$ represents the local quality of the point cloud at point p, calculated using the local covariance matrix defined at p.[19] In this study, the covariance matrix at point p is computed at three different scales (i.e., different neighborhood ranges denoted as i = 6, 16, 32). Therefore, $conf(p)$ is defined by the following formula.

$$conf\left(p\right) = \frac{1}{3}\sum_{i=1}^{3}\left(1 - \frac{3\lambda_i^1}{\lambda_i^1 + \lambda_i^2 + \lambda_i^3}\right)\cdot\frac{\lambda_i^2}{\lambda_i^3} \tag{8}$$

3.4.2 Model complexity

Owing to occlusions causing incomplete point cloud data and the strict requirement for incomplete surfaces as defined by the point cloud fitting degree in Eqs. (3) and (4), the final reconstructed model may have gaps. Additionally, noise and outliers can introduce gaps and protrusions, resulting in an uneven surface and affecting subsequent applications. Therefore, in

this paper, we define the model complexity coefficient as the ratio of the number of nonplanar edges to the total number of edges in the model:

$$E_m = \frac{1}{|E|}\sum_{i=1}^{|E|} c(e_i), \tag{9}$$

where $|E|$ represents the total number of edges obtained from the pairwise segmentation of the faces, and $c(e_i)$ is the indicator function whose value is determined by the structure of the two candidate faces associated with the edge e_i. If the two faces are coplanar, the function value is 0; if the two faces are not coplanar, the function value is 1.

3.4.3 Point cloud coverage

To address the issue of missing data due to occlusion, the area of the model without point cloud support should be minimized. To measure the point cloud coverage of a face f, the process involves first projecting the point clouds with distances smaller than ε onto the face f. Then, a 2D α-shape mesh is constructed on the basis of the projected point clouds that meet the criteria.

By setting an appropriate threshold, we can obtain the area of the α-shape mesh surface that provides a measure of the coverage of the face by the input point cloud. Therefore, the point cloud coverage coefficient can be defined as the ratio of the point clouds not covered by the model.[20]

$$E_c = \frac{1}{B(M)}\sum_{i=1}^{N} x_i \cdot \left(B(f_i) - B\left(M_i^{\alpha}\right)\right) \tag{10}$$

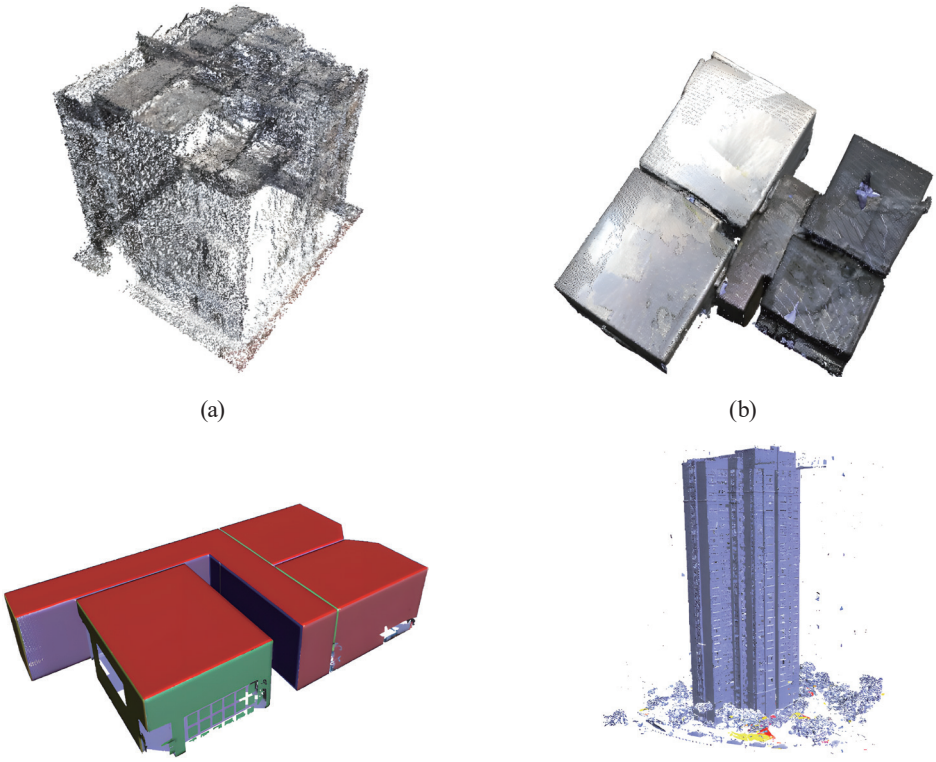
Here, $B(M)$ represents the surface area of the final model, $B(f_i)$ denotes the area of the candidate face f_i, and $B(M_i^{\alpha})$ is the surface area of the α-shape mesh f_i constructed from the point clouds projected onto the face M_i^{α}.

4 Experiment and Analysis of Results

4.1 Experimental environment and dataset

The experimental environment for this algorithm was set up on a PC with the following configuration: Windows 10 64-bit operating system, Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz, and 16.0 GB of memory. The programming language used was C++, and the geometric algorithm library employed was CGAL.[21] The graphical visualization tool used was the open-source software CloudCompare.[22]

The experimental data are shown in Fig. 8. Figure 8(a) shows point cloud data obtained from images using MVS technology. These data are of poor quality, with numerous noise and outlier points. Figure 8(b) shows point cloud data of a room collected using a Faro Focus 130 total



(a) (b)

(c) (d)

Fig. 8. (Color online) Experimental data. (a) Building, (b) Room, (c) Synthetic Data, and (d) High building.

station ground laser scanner, which appears cluttered owing to occlusions from tables and chairs. Figure 8(c) illustrates a campus point cloud dataset acquired by the Visualization and Multimedia Laboratory at the University of Zurich using a Faro Focus 3D scanner. These data are relatively complete and not significantly cluttered. Figure 8(d) shows a high-rise building point cloud dataset provided by the Visual Computing Research Center of Shenzhen University, obtained via drone aerial photography, with average point cloud quality.

4.2 Building point cloud 3D reconstruction results

To reconstruct the 3D model of the input point cloud data, the following steps were performed: first, the original point cloud was segmented into planes; second, the segmented faces were used to construct a 3D polyhedron; and finally, face selection optimization was conducted to obtain a polygonal surface model. In this experiment, the three energy coefficients of the objective function were set to λ_m = 0.27, λ_f = 0.46, and λ_c = 0.27. Through multiple experiments, it was verified that these weights produced consistent results over a wide range of variations.

Figures 9(a)–9(d) sequentially show the input Building raw point cloud data, the one-point RANSAC segmented plane point cloud (with different colors representing different segmented faces), the reconstructed model, and the overlay of the reconstructed model with the original

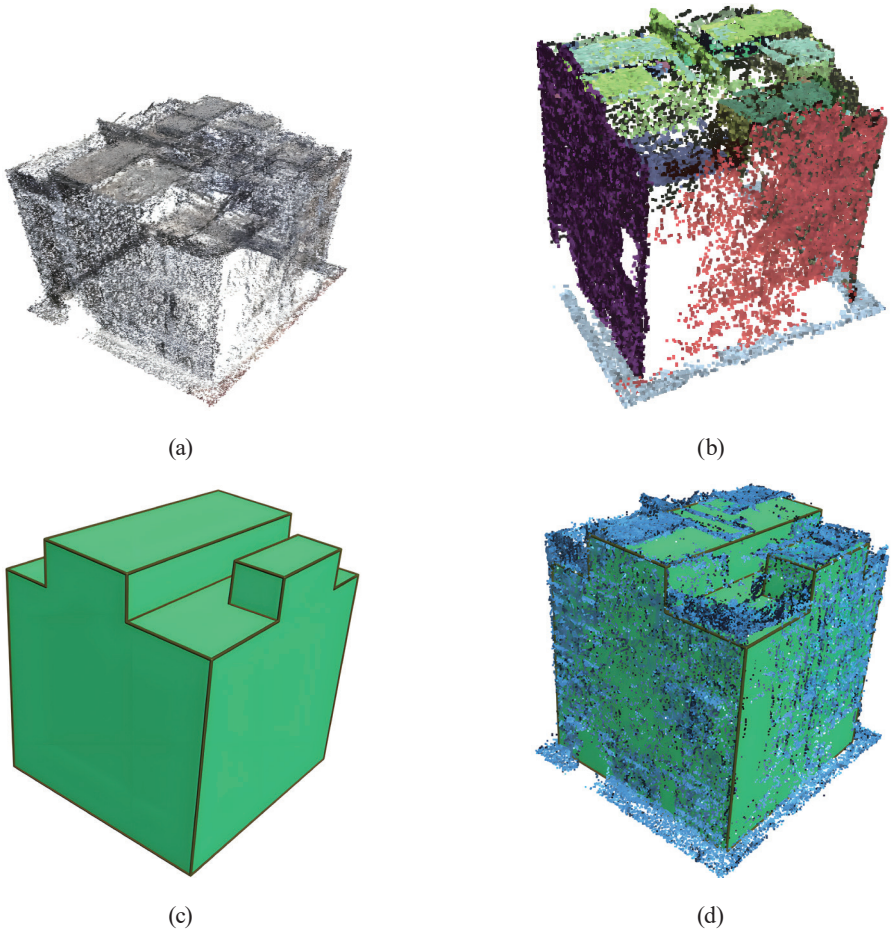
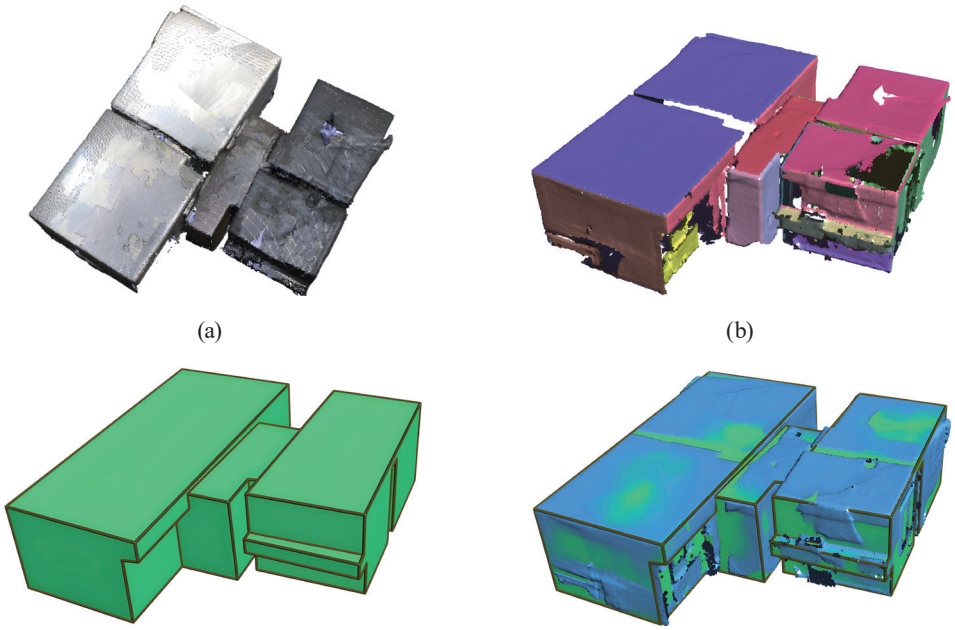


Fig. 9. (Color online) Reconstruction process from building point cloud to model. (a) Input building raw point cloud data, (b) segmentation plane, (c) reconstruction model, and (d) superimposition of the reconstructed model and the raw point cloud.

point cloud. For the Building point cloud data, the initial segmentation extracted 19 planes. After iterative refinement, 13 planes were retained. Plane segmentation resulted in 348 faces forming a 3D polyhedron. After the final optimization, 14 faces were kept, with the entire modeling process taking 2.47 s.

Figures 10(a)–10(d) sequentially show the input Room raw point cloud data, the one-point RANSAC segmented plane point cloud (with different colors representing different segmented faces), the reconstructed model, and the overlay of the reconstructed model with the original point cloud. For the Room point cloud data, the initial segmentation extracted 28 planes. After iterative refinement, 23 planes were retained. Plane segmentation resulted in 1,825 faces forming a 3D polyhedron. After the final optimization, 26 faces were kept, with the entire modeling process taking 45.65 s.

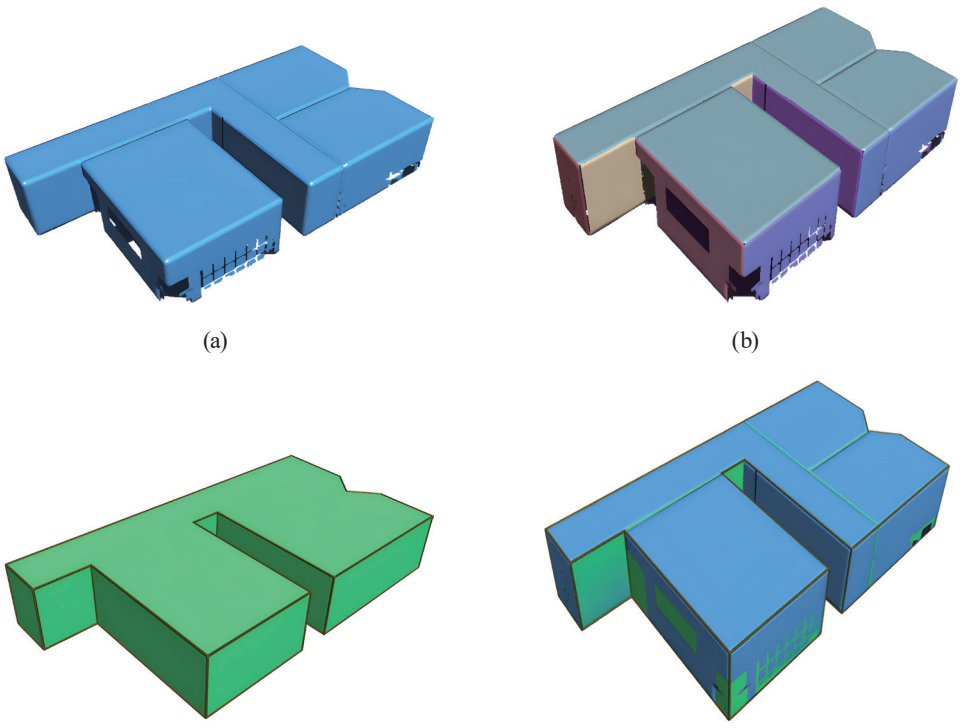
Figures 11(a)–11(d) sequentially show the input Synthetic Data raw point cloud data, the one-point RANSAC segmented plane point cloud (with different colors representing different segmented faces), the reconstructed model, and the overlay of the reconstructed model with the original point cloud. For the Synthetic Data point cloud, the initial segmentation extracted 19



(a) (b)

(c) (d)

Fig. 10. (Color online) Reconstruction process from Room point cloud to model. (a) Input Room raw point cloud data, (b) segmentation plane, (c) reconstruction model, (d) superimposition of the reconstructed model and the raw point cloud.



(a) (b)

(c) (d)

Fig. 11. (Color online) Reconstruction process from Synthetic Data point cloud to model. (a) Input Synthetic Data raw point cloud data, (b) segmentation plane, (c) reconstruction model, (d) superimposition of the reconstructed model and the raw point cloud.

planes. After iterative refinement, 17 planes were retained. Plane segmentation resulted in 515 faces forming a 3D polyhedron. After the final optimization, 15 faces were kept, with the entire modeling process taking 37.53 s.

Figures 12(a) to 12(d) sequentially show the input High building raw point cloud data, the one-point RANSAC segmented plane point cloud (with different colors representing different segmented faces), the reconstructed model, and the overlay of the reconstructed model with the original point cloud. For the High building UAV imagery point cloud, the initial segmentation extracted 43 planes. After iterative refinement, 40 planes were retained. Plane segmentation resulted in 3,251 faces forming a 3D polyhedron. After the final optimization, 52 faces were kept, with the entire modeling process taking 62.33 s.

4.3 Experimental analysis

The experimental results demonstrate that the proposed algorithm exhibits excellent adaptability to various types of building point cloud data, effectively completing the 3D reconstruction of different architectural models. Despite the presence of significant noise, outliers, and missing data in the point cloud data, the reconstructed models accurately restore the

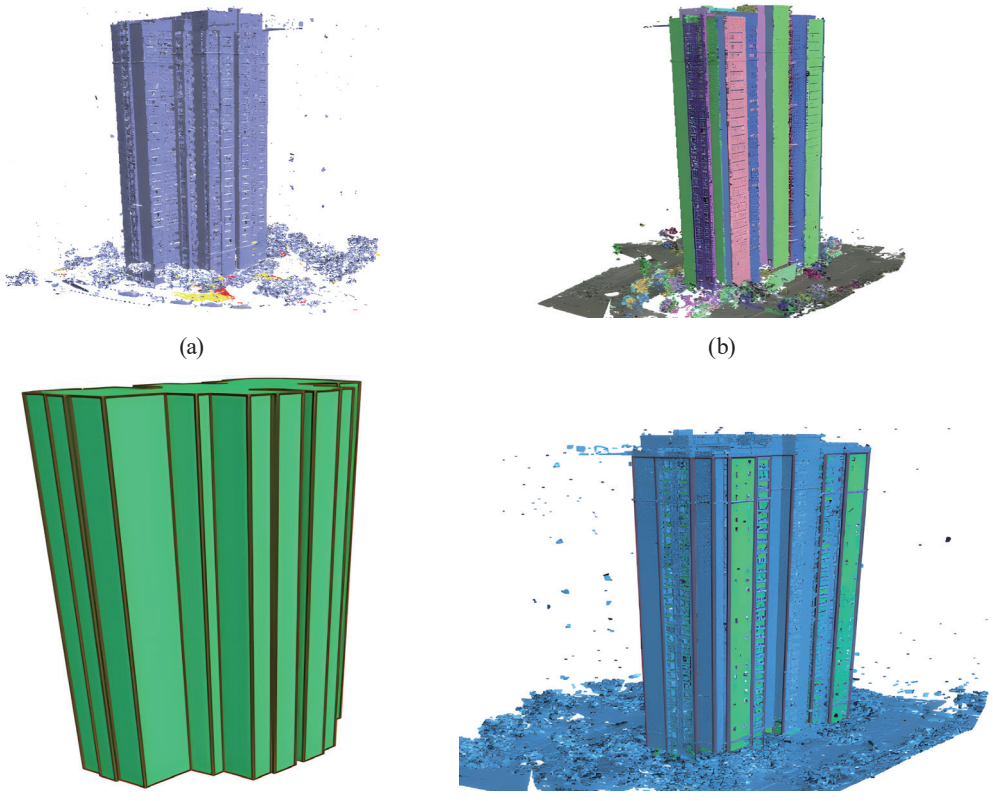


Fig. 12. (Color online) Reconstruction process from High building point cloud to model. (a) Input High building raw point cloud data, (b) segmentation plane, (c) reconstruction model, and (d) superimposition of the reconstructed model and the raw point cloud.

buildings, ensuring the correctness and compactness of the topological structure. For indoor scenes with chaotic characteristics, despite the presence of voids in the data, particularly the loss of data at wall edges, the algorithm effectively fills these missing regions and reconstructs permanent structures in these scenes, such as walls and roofs. Table 1 presents the statistical analysis of the modeling results for various experimental datasets.

In this paper, we used two different types of point cloud data for our experimental tests. Figure 13 shows the reconstruction results of the proposed method compared with three other methods on the same Manhattan-rule building point cloud data. The results show that the screened Poisson[23] algorithm produces a dense surface model with many uneven surfaces, being highly sensitive to noise and outliers, and resulting in a mesh count of 282,176, which consumes a large amount of memory. The structured indoor modeling (SIM)[24] algorithm generates a more detailed dense surface model with a total of 104 mesh patches, but the model has protrusions and gaps and is only suitable for Manhattan-rule models. In contrast, both RAPTER[25] and the proposed method produce clean and compact surface models, with similar numbers of mesh patches: 22 for RAPTER and 25 for the proposed method, indicating that both methods can produce lightweight triangular mesh models.

Figure 14 shows the reconstruction results of the proposed method compared with those of two other methods on the same irregular indoor building point cloud data. The results indicate that the screened Poisson reconstruction algorithm fits the point cloud data well and restores architectural details effectively, but it has poor robustness to noise and outliers and generates a large number of mesh patches, which may not be suitable for subsequent applications. The SIM algorithm, on the other hand, cannot handle irregular indoor building models and is only applicable to Manhattan-rule buildings. The RAPTER algorithm can reconstruct the main structure of the building but exhibits inaccuracies in representing the topological relationships between planes, as indicated within the red box. In contrast, the proposed method accurately represents the model's topological structure, although it lags behind the screened Poisson algorithm in detail representation, requiring further optimization and improvement in future work.

Table 1
Experimental data modeling and statistical results

Type	Building	Room	Synthetic Data	High building
Input point cloud	118K	186K	1833K	3431K
Initial number of segmented surfaces	19	28	19	43
Number of generated candidate surfaces	348	1825	515	3251
Number of final model surfaces	14	26	15	52
Modeling time consumption/s	2.47	45.65	37.53	62.33
Root mean square error (RMSE)/m	0.28	0.11	0.07	0.73

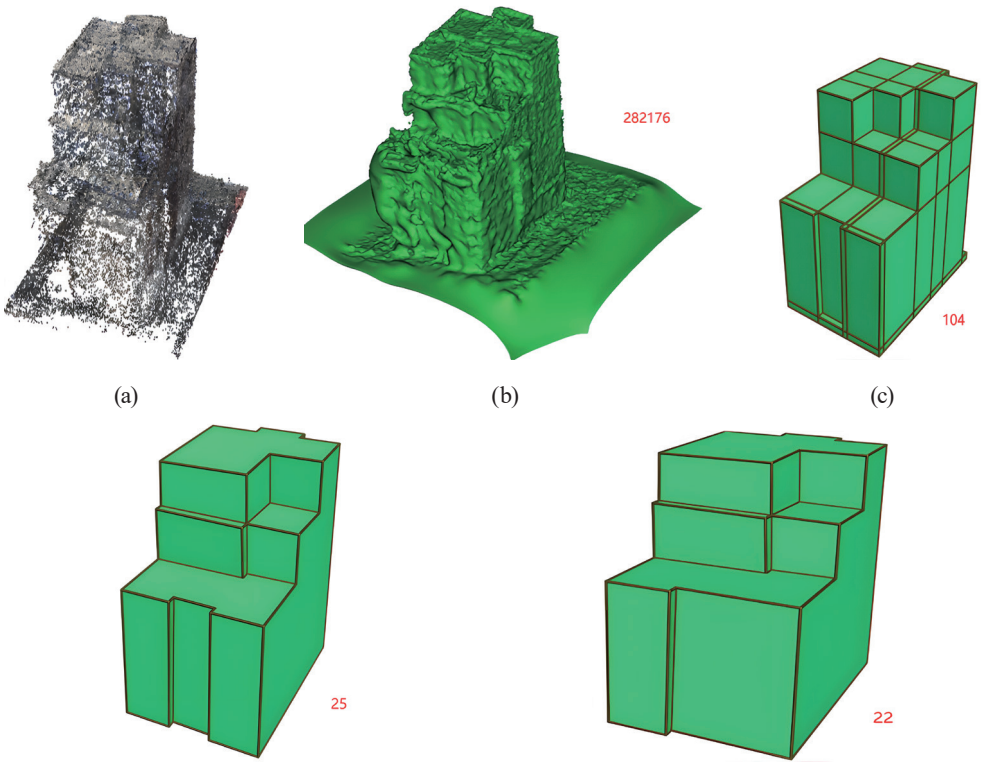


Fig. 13. (Color online) Comparison of our method with the other three popular modeling methods. The numbers in red are the numbers of patches of the generated models. (a) input building point cloud, (b) model of the Screened Poisson reconstruction algorithm, (c) Screened Poisson, (d) SIM, and (e) our algorithm.

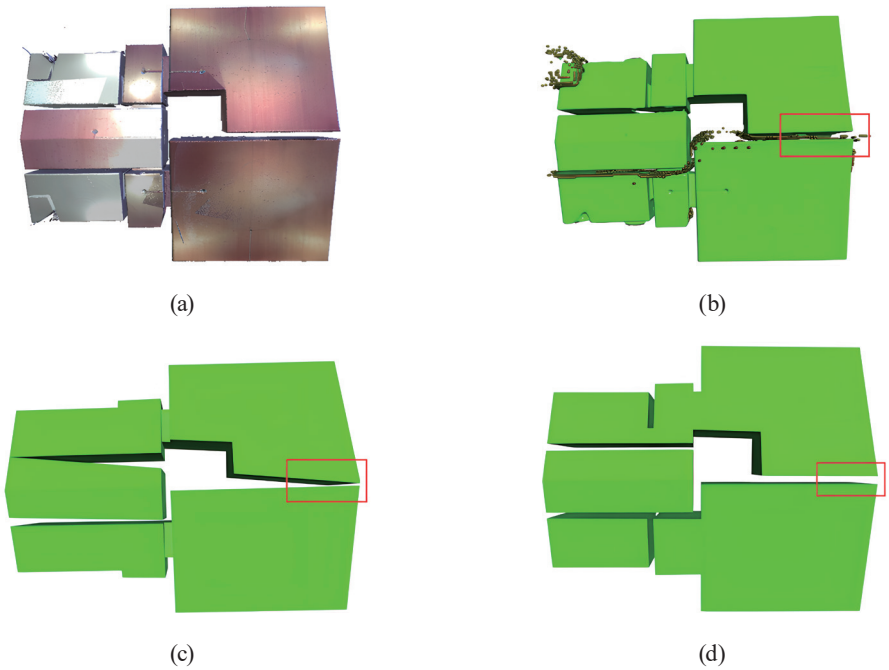


Fig. 14. (Color online) Our method is compared with the other two popular modeling methods. (a) Input building point cloud, (b) model of screened Poisson reconstruction algorithm, (c) model of rapid reconstruction algorithm, and (d) results of our algorithm.

5 Conclusion

In this paper, we explore an automated approach to constructing 3D building models from raw point cloud data, presenting a comprehensive technical solution that encompasses point cloud preprocessing, plane segmentation, and model reconstruction. The method effectively constructs irregular 3D building models with lightweight triangular meshes and correct structural topology. The key contributions of this research include the following:

(1) An improved eight-connectivity algorithm is proposed for point cloud processing. This algorithm initially partitions and labels the point cloud into a 2D grid, followed by outlier removal using eight connectivity domains. Compared with traditional statistical filtering and radius filtering methods, this approach shows superior performance in outlier removal.
(2) A plane segmentation method based on a single point and its normal vector is introduced for extracting geometric elements from 3D building point clouds. By downsampling the point cloud and calculating normal vectors, we can define a plane by this method with just one sample point containing a normal, reducing the number of iterations compared with conventional RANSAC methods and enhancing the efficiency and quality of plane detection.
(3) To address noise and information loss in point clouds, a 3D building model reconstruction method based on 3D polyhedron face selection optimization is proposed. After outlier removal and plane segmentation, the method generates a 3D polyhedron through iterative refinement, and binary linear programming is used to optimize candidate faces, selecting those that best represent the actual building structure to construct the 3D model.

Experimental results demonstrate that the proposed method is robust to noisy point clouds and missing information, accurately represents the building's topological structure, and produces fewer triangular mesh patches, making it suitable for various industrial applications.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (grant number 52378385).

References

1 Y. Xu and U. Stilla: IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. **14** (2021) 2857. https://doi.org/10.1109/JSTARS.2021.3060568
2 D. Li, Z. Shao, and X. Yang: Geo-spatial Inf. Sci. **9** (2011) 1.
3 J. L. Du, D. Chen, Z. X. Zhang, and L. Q. Zhang: J. Remote Sens. **23** (2019) 374. https://doi.org/10.11834/jrs.20188199
4 P. Zhao: Ph.D. Thesis, Wuhan University (2020) p. 141. https://link.cnki.net/doi/10.27379/d.cnki.gwhdu.2020.000642
5 D. Skarlatos and S. Kiparissi: ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci. (2012). https://doi.org/10.5194/isprsannals-I-3-299-2012
6 Y. Wang: University of British Columbia (2014). https://dx.doi.org/10.14288/1.0167059
7 C. Tomasi and R. Manduchi: Proc. 6th Int. Conf. Comput. Vis. (IEEE Cat. No. 98CH36271) (1998) 839.
8 H. Balta, J. Velagic, W. Bosschaerts, G. De Cubber, and B. Siciliano: IFAC-PapersOnLine **51** (2018) 348. https://doi.org/10.1016/j.ifacol.2018.11.566

9 X. Ning, F. Li, G. Tian, and Y. Wang : PLoS One **13** (2018) e0201280. https://doi.org/10.1371/journal.pone.0201280
10 D. Birant and A. Kut: Data Knowl. Eng. **60** (2007) 208. https://doi.org/10.1016/j.datak.2006.01.013
11 Q. Li, X. Hua, B. Zhao, W. Tao, and C. Li.: Chin. J. Lasers **48** (2021) 1604002. https://doi.org/10.3788/CJL202148.1604002
12 P. Hu, Y. Liu, M. Tian, and M. Hou: ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci. **2** (2020) 221. https://doi.org/10.5194/isprs-annals-V-2-2020-221-2020
13 K. Jordan and P. Mordohai: Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (2014) 4220. https://doi.org/10.1109/IROS.2014.6943157
14 R. Hartley: Multiple view geometry in computer vision, and A. Zisserman, Eds. (Australian National University, Canberra , 2003) 2nd ed., Chap. 2.
15 M. Li, L. Nan, N. Smith, and P. Wonka: Comput. Graph. **54** (2016) 84. https://doi.org/10.1016/j.cag.2015.07.004
16 C. Giannelli, S. Imperatore, L. M. Kreusser, E. Loayza-Romero, F. Mohammadi, and N. Villamizar: Math. Comput. Simul. **225** (2024) 52. https://doi.org/10.1016/j.matcom.2024.04.029
17 E. Fogel: CGAL Arrangements and Their Applications: A Step-by-Step, D. Halperin, and R. Wein, Eds. (Tel Aviv, Israel, 2012) 7th ed., Chap. 19–42. https://doi.org/10.1007/978-3-642-17283-0?nosfx=y
18 G. Pond: Interfaces **40** (2010) 3. https://doi.org/10.1007/978-0-387-92280-5
19 M. Pauly, N. J. Mitra, J. Giesen, M. Gross, and L. J. Guibas: Proceedings of the third Eurographics symposium on Geometry processing (SGP,2005) 23.
20 F. Bernardini and C. L. Bajaj: Sampling and Reconstructing Manifolds Using Alpha-Shapes (CCCG,1997).
21 Rooms UZH Irchel Dataset: http://www.ifi.uzh.ch/en/vmml/research/datasets.html (2016).
22 Q. Zheng, A. Sharf, and G. Wan: ACM Trans. Graph. **29** (2010) 1. https://doi.org/10.1145/1778765.1778831
23 M. Kazhdan and H. Hoppe: ACM Trans. Graph. **32** (2013) 1. https://doi.org/10.1145/2487228.2487237
24 S. Ikehata, H. Yang, and Y. Furukawa: Proc. IEEE Int. Conf. Comput. Vis. (2015) 1323. https://doi.org/10.1109/ICCV.2015.156
25 A. Monszpart, N. Mellado, and G. J. Brostow: ACM Trans. Graph. **34** (2015) 1. https://doi.org/10.1145/276699

About the Authors

Yong Wang is from Henan, China. In 2018, he received his master's degree in surveying and mapping engineering from Beijing University of Civil Engineering and Architecture. Since 2018, he has been working at Beijing Urban Construction Survey and Design Institute, mainly engaged in rail transit safety monitoring, laser technology, sensors, and other aspects of research. (wangy170@163.com)

Chao Tang is a professorial senior engineer. He received his Ph.D. degree from China University of Geosciences in 2014. Since 2016, he has been the chief engineer of the Intelligent Engineering Institute of Beijing Urban Construction Survey and Research Institute Co. His research interests are in the intelligent detection of tunnel health, laser technology, and sensors. (15801186470@163.com)

Ming Huang is a professor and Ph.D. supervisor at Beijing University of Civil Engineering and Architecture. His research interests are in point cloud and image hyperfine 3D modeling and visualization, and mobile vehicle laser point cloud ground feature intelligent recognition. (huangming@bucea.edu.cn)

Haipeng Zhu received his B.S. degree from Nanchang Institute of Technology, China, in 2015 and his M.S. degree from Beijing University of Civil Engineering and Architecture, China, in 2022. From 2016 to 2018, he was a computer programmer at Hangzhou Xiruan Information Technology Co., Ltd., China. Since 2022, he has been a software engineer at Piesat Information Technology Co., Ltd. His research interests are in GIS and point cloud processing. (zhuhaipeng79@gmail.com)

Yuan Gao is a graduate student of Beijing University of Civil Engineering and Architecture. His research interests include indoor 3D reconstruction (15306386022@163.com)