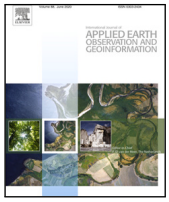
International Journal of Applied Earth Observation and Geoinformation 128 (2024) 103728

Contents lists available at ScienceDirect

International Journal of Applied Earth Observation and Geoinformation

journal homepage: www.elsevier.com/locate/jag



Accurate and complete line segment extraction for large-scale point clouds

Xiaopeng Xin [a,b,1], Wei Huang [c,1], Saishang Zhong [d], Ming Zhang [e,f], Zheng Liu [a,f,*], Zhong Xie [a]

[a] School of Computer Science, China University of Geosciences (Wuhan), Wuhan, 430074, China
[b] National Engineering Research Center of Geographic Information System, China University of Geosciences (Wuhan), Wuhan, 430074, China
[c] School of Geography and Information Engineering, China University of Geosciences (Wuhan), Wuhan, 430074, China
[d] School of Computer Science and Artificial Intelligence, Wuhan Textile University, Wuhan, 430200, China
[e] Guangzhou Urban Planning and Design Survey Research Institute, Guangzhou, 510060, China
[f] Guangdong Enterprise Key Laboratory for Urban Sensing, Monitoring and Early Warning, Guangzhou, 510060, China

ARTICLE INFO

Keywords:
Line segment extraction
Large-scale point clouds
Feature point detection

ABSTRACT

Line segment extraction from point clouds is critical for analyzing and understanding large-scale scenes. The main challenge is to generate line segments accurately as well as completely. However, state-of-the-art approaches continue to struggle with this issue. In this paper, we propose a novel method for effectively generating line segments from large-scale point clouds. To this end, we design a weighted centroid displacement scheme for identifying comprehensive feature points. Then, we employ an L_1-median optimization to refine the identified feature points to perceive geometric edges on the underlying surface accurately. Following that, we generate complete and concise line segments from the refined feature points by designing three geometric operators: clustering, exclusion, and assimilation. The clustering operator generates the initial line segments based on optimized feature points, and the exclusion operator and the assimilation operator ensure the completeness and continuity of these line segments. We evaluate our approach on various scene point clouds, such as TLS, MLS, and ALS data. Extensive experimental results show that our method can outperform the competing approaches in terms of both accuracy and efficiency.

1. Introduction

In recent years, with the rapid development of 3D laser scanning technologies, it has become more and more convenient for acquiring scene-level point clouds containing rich 3D geospatial information, which has been widely used in many fields, such as segmentation (Li et al., 2023a; Adam et al., 2023; Lei et al., 2022), 3D mapping (Li et al., 2023b; Tang et al., 2016; Ma et al., 2023), digital twins (Wu et al., 2022; Lehtola et al., 2022), autonomous driving (Li and Zhuang, 2023; Chen et al., 2018), reconstruction (Cui et al., 2019; Wang et al., 2023), etc. However, using scene-level point clouds directly in practice is usually hard because of their huge volume. To this end, line segment extraction has already been a fundamental step for analyzing and understanding large-scale scenes. The main technical challenge of extracting line segments from point clouds is to completely and accurately generate line segments from structural contours to shadow details of the underlying surface.

The proposed line segment extraction method is divided into two stages, i.e., feature point detection followed by line segment generation. For the former, the existing feature point detection methods can be broadly classified into two categories: geometric-based methods (Xia and Wang, 2017; Chen and Yu, 2019; Liu et al., 2020, 2021) and learning-based methods (Yu et al., 2018; Himeur et al., 2021; Matveev et al., 2022). The geometric-based methods detect feature points based on intrinsic geometric properties of the underlying surface. Although these methods effectively detect the feature points in geometric structures, they frequently misidentify shallow features on the underlying surface. The learning-based methods are also able to detect feature points effectively. However, these methods are time-consuming, and their performance depends on the completeness of training datasets.

For the second stage, researchers typically use point-based methods to generate line segments. Although existing point-based methods are effective in extracting the structural contours of scenes, they potentially ignore shallow details to some extent. Thus, effective methods for extracting line segments at various scales are still challenging. For instance, the method proposed by Xia and Wang (2017) performs well at extracting model contours but requires high-quality feature points as input. The method developed by Lin et al. (2017) maximizes the utility of feature points detected in large-scale scenes to generate line segments.

* Corresponding author at: School of Computer Science, China University of Geosciences (Wuhan), Wuhan, 430074, China.
E-mail address: liu.zheng.jojo@gmail.com (Z. Liu).
[1] Contribute equally to this work.

https://doi.org/10.1016/j.jag.2024.103728
Received 15 December 2023; Received in revised form 1 February 2024; Accepted 18 February 2024

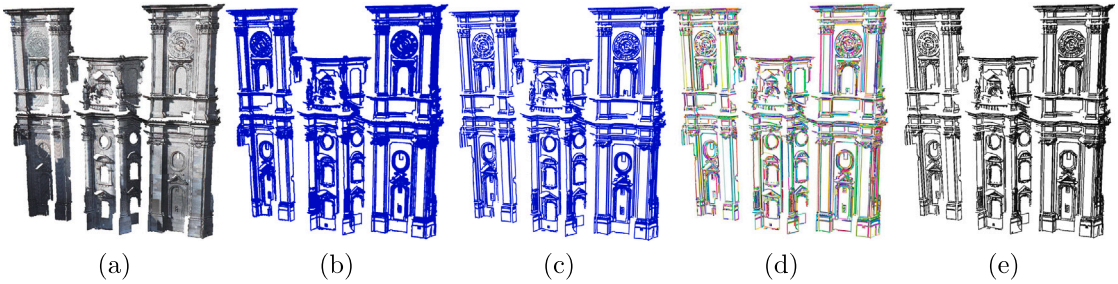


Fig. 1. Overview of the proposed method. (a) The raw input point cloud. (b) Identified feature points by weighted centroid displacement. (c) Thinned feature points via L_1-median optimization. (d) Extracted line segments from groups of refined points using the clustering operator. (e) The final complete line segments via the assimilation operator.

However, because of limitations in the feature point detection method, it struggles to capture small-scale features effectively. Tian et al. (2022) proposed a RANSAC-based line segment extraction algorithm. Nonetheless, high-quality feature points derived from point cloud segmentation results are required for the RANSAC-based algorithm. In addition to point-based line segment extraction, existing methods encompass image-based methods (Lin et al., 2015; Hofer et al., 2017; Lu et al., 2019) and learning-based methods (Wang et al., 2020; Zhang et al., 2020; Hu et al., 2022). Image-based methods can accurately extract line segment features by using object edges as prior information. Nonetheless, these methods require high-quality images with clear edge information. Furthermore, the 2D–3D transformation may result in losing important geometric features. On the other hand, learning-based detection methods must prepare annotated training datasets when processing LiDAR point clouds, which is time-consuming and labor-intensive.

To address the aforementioned issues, we propose a two-stage line segment extraction method. In the first stage, we introduce a novel metric for identifying feature points based on a weighted centroid displacement scheme, as shown in Fig. 1(b). Following that, we employ the L_1-median optimization to refine the identified coarse feature points, as illustrated in Fig. 1(c). The refined feature points can be used to fit the geometric edges of the underlying surface accurately. In the second stage, we design three geometric operators (i.e., clustering operator, exclusion operator, and assimilation operator) to generate line segments from point clouds. Specifically, the cluster operator divides the refined feature points into distinct clusters, and each can be used for generating a corresponding line segment; see Fig. 1(d). The exclusion operator filters out the outliers line segments. The assimilation operator ensures the completeness and continuity of the extracted line segments; see Fig. 1(e). We verify the effectiveness of our method on a variety of laser-scanning datasets. In summary, the main contributions of our work are as follows:

- We propose a weighted centroid displacement scheme to identify feature points on the input point cloud and then employ an L_1-median optimization to refine the identified features. Our method can accurately perceive feature points on the underlying surface that range from structural to finer characteristics.
- We design three geometric operators to generate line segments based on optimized feature points and ensure their completeness and continuity.
- We conduct extensive experiments to demonstrate that our method outperforms the state-of-the-art approaches on a variety of large-scale point clouds, such as TLS, MLS, and ALS data.

2. Related work

In this section, we review the noticeable progress of line segment extraction methods for point clouds from two aspects: Feature point detection and line segment generation. Since there are many various line segment extraction methods, we only review those that are highly related to ours.

2.1. Feature point detection

Feature point detection is crucial for processing large-scale point clouds in various applications, such as point cloud scene segmentation and reconstruction. Existing techniques for feature point detection can be broadly classified into geometric-based and learning-based methods. The geometric-based methods usually detect feature points from point clouds based on the intrinsic geometric properties of their corresponding underlying surfaces.

For example, Xia and Wang (2017) first defined gradients in unorganized 3D point clouds with a proposed edge index based on geometric centers and then utilized an eigenvalue analysis approach and graph snapping algorithm to detect edge points. Chen and Yu (2019) employed a vector distribution and clustering algorithm to detect feature points and used an improved cubic B-spline curve algorithm to fit the feature lines on point clouds. By analyzing the tensors of point normals and locations, Liu et al. (2020) proposed a bi-tensor voting method to recognize feature points. Besides, Liu et al. (2021) further introduced a neighbor reweighted local centroid scheme for robustly detecting feature points from point clouds. Hackel et al. (2017) designed a joint classification and contour extraction method for point clouds, enabling the definition of an expressive feature set and the extraction of topological meaningful object contours. Recently, a growing trend has been utilizing learning-based methods for feature point detection from point clouds. Yu et al. (2018) introduced EC-Net, the first neural network to detect feature points from point clouds. Himeur et al. (2021) proposed PCEDNet, a lightweight neural network that leverages neural networks to detect edges in point clouds. Matveev et al. (2022) proposed DEF, a deep learning framework for extracting sharp features in CAD models.

Though the above methods can efficiently detect feature points in small-scale point clouds, e.g., CAD models, they frequently misidentify shallow features on the underlying surface. Besides, the performances of some methods rely on additional geometric information (e.g., normal vectors), which may be unreliable. Last but not least, they are time-consuming when handling large-scale point clouds.

2.2. Line segment generation

Existing techniques for extracting line segments from point clouds can be roughly divided into three types of methods: (1) point-based methods: Moghadam et al. (2013) proposed a feature line extraction algorithm based on region growth. This method first detects boundary points and the intersection points between planes and then fits these two types of points into feature line segments. Ni et al. (2016) utilized the RANSAC scheme and angular gap metric algorithms to detect feature line segments. Lin et al. (2017) proposed a method extracting line segments from point clouds based on linear features in the local planar region, which is provided by a collection of facets. Xia and Wang (2017) designed an edge-linking algorithm on point clouds. This method first builds the graph structure using feature points and generates line segments using approximate orientation. Tian et al. (2022) presented an approach that extracts line segments from point

clouds by fitting boundary points of the detected plane structures; (2) image-based methods: Lin et al. (2015) presented a segmentation-based line segment algorithm for point clouds, which first projects the segmented regions on point clouds onto the corresponding 2D images, and extracts 2D line segments using the contours of the projected regions, and then projects these line segments back into the 3D space to obtain line segments on point clouds. Hofer et al. (2017) first employed the LSD algorithm to extract 2D line segments from images and then used graph clustering to obtain the final 3D line segments. Lu et al. (2019) proposed a method that extracts line segments from point clouds by using 2D planes and the structural information of the scene; (3) learning-based methods: Wang et al. (2020) introduced PIE-NET, an end-to-end learnable approach for detecting feature edges using a region proposal strategy. Zhang et al. (2020) proposed a parametric space-based framework and a guided learning approach to address the challenges of extracting features from large-scale point clouds with huge, unstructured point space. Hu et al. (2022) used a pre-trained neural networks to detect 3D line segments from point clouds. Zhao et al. (2022) developed the first learning-based model for segmenting and describing 3D lines in LiDAR point clouds, as well as performing robust point cloud registration with the extracted 3D lines as feature descriptors.

Although existing methods for extracting line segments from point clouds can extract them on significant structures well, they still have the following limitations. First, existing approaches frequently miss line segments on fine details. Second, line segments extracted by previous methods are often inaccurate and incomplete. Finally, the performances of image-based methods rely on the quality of 2D images, and the performances of learning-based methods depend on the completeness of the training dataset.

3. Methodology

In this section, we present a novel two-stage method for extracting high-quality line segments from large-scale point clouds. In the first stage, we can accurately detect feature points that precisely locate the geometric edges of the underlying surface. Using three geometric operators, we reconstruct complete and concise line segments from the detected feature points in the second stage. Details of both are depicted as follows.

3.1. Detecting feature points

Since large-scale point clouds frequently contain widely geometric characteristics, ranging from structural to finer edges, it is a challenge to directly identify accurate feature points from large-scale point clouds. To this end, we first introduce a weighted centroid displacement scheme for detecting coarse feature points, followed by using an L_1-median optimization to refine the detected feature points so that these refined points can accurately match underlying geometric edges.

3.1.1. Coarse feature points detection by a weighted centroid displacement scheme

First, we denote the input point cloud as P. Given a point $p_i \in P$, its traditional centroid displacement vector L_i can be calculated as $L_i = c_i - p_i$, where c_i is the centroid computed by averaging the positions of the k-nearest neighbors of p_i. Roughly, $\|L_i\|$, the length of vector L_i, can be used as a metric for identifying coarse feature points. In most cases, a large value of $\|L_i\|$ indicates that point p_i is more likely to be a feature point. However, the traditional centroid displacement cannot perceive minor surface variations effectively. This may result in feature points being misidentified as non-feature points in regions with minor surface variations, as shown in Fig. 3(b). The reason is that L_i is essentially a uniform Laplacian vector, and its weight strategy is not sensitive to the surface variation. As can be observed from Figs. 3(a) and 3(b), the lengths of traditional displacement vectors of points in the flat region and shallow feature are both small, which cannot be used for distinguishing shallow feature points and non-feature points clearly.

To tackle these challenges, we introduce a novel weighted centroid displacement scheme. The key is to estimate a new centroid for each point to enlarge the differences between feature points and non-feature points, especially those in regions with shallow features. As illustrated in Fig. 2, if point p_i in the flat region or shallow feature, its corresponding traditional centroid c_i is very close to the underlying surface; see purple points in Figs. 2(a) and 2(b). Thus, we are difficult to identify feature points from shallow features using the traditional centroid displacement vector L_i. For this reason, we estimate a new centroid for each point p_i. To this end, for one neighbor point p_j, we first generate a temp point m_j, the green point in Fig. 2(c), to construct a vector $\overrightarrow{p_i m_j} = \overrightarrow{p_i p_j} + \overrightarrow{p_i c_i}$. After constructing all the vectors, for point p_i, we can estimate a new centroid d_i for computing the proposed weighted centroid displacement vector $\delta_i = d_i - p_i$, which can be rewrite as

$$\delta_i = \frac{1}{\sum_{p_j \in \Omega(p_i)} \omega(i,j)} \sum_{p_j \in \Omega(p_i)} \omega(i,j) \overrightarrow{p_i m_j}, \tag{1}$$

where $\omega(i,j) = \exp(-\left\|p_i - p_j\right\|^2 / r^2)$, $\Omega(p_i)$ denotes the $k-$nearest neighbor points of p_i in P, and r represents the average distance between point p_i and its neighbor points. As shown in Fig. 2, the weighted centroid displacement vectors of points in sharp features and shallow features are larger than those of points in flat regions. The comparison between Figs. 2(a) and 2(b) shows that the weighted centroid displacement scheme amplifies the differences between points in shallow features and points in flat regions. Inspired by previous Laplace-based methods SSI (Nie, 2016) and NRLC (Liu et al., 2021), our scheme aims to push the centroid far from the local surface for identifying the potential feature point by adding a vector $\overrightarrow{p_i p_j}$ to the traditional centroid displacement vector $\overrightarrow{p_i c_i}$; see Fig. 2 for example. Doing this allows us to enlarge the differences between feature and non-feature points.

Fig. 4 illustrates the point weights computed by using two metrics including $\|L_i\|$ and $\|\delta_i\|$. From Fig. 4(b), we can observe that the colors of feature points in shallow edges are similar to those in smooth regions when using metric $\|L_i\|$. Being different from that, we can see from Fig. 4(c) that metric $\|\delta_i\|$ greatly enlarges the differences between feature points and non-feature points, especially those in shallow edges; see the zoomed regions. The reason is that the proposed weighted centroid displacement scheme would be more suitable for the task of feature point detection when compared to the traditional centroid displacement scheme.

Based on the above analysis, we define a new metric for detecting feature points from point clouds as

$$f_i = \kappa_i \cdot \|\delta_i\|, \forall p_i, \tag{2}$$

where $\kappa_i > 0$ is a weighted term aiming to further increase the differences between feature and non-feature points, especially in the regions with finer edges. As we know, feature points generally appear in regions with large geometric variations. Therefore, with this prior knowledge, we use flatness to estimate the weight κ_i based on the weighted centroid displacement vector δ_i. Specifically, the flatness of the local surface is measured along xyz axes, which is calculated as

$$\kappa_i = \omega_i^x + \omega_i^y + \omega_i^z, \tag{3}$$

where ω_i^x, ω_i^y, and ω_i^z are calculated as

$$\begin{aligned} \omega_i^x &= \alpha + (1-\alpha) \cdot \frac{(\delta_i^x - \theta_{min}^x)}{\theta_{max}^x - \theta_{min}^x}, \\ \omega_i^y &= \alpha + (1-\alpha) \cdot \frac{(\delta_i^y - \theta_{min}^y)}{\theta_{max}^y - \theta_{min}^y}, \\ \omega_i^z &= \alpha + (1-\alpha) \cdot \frac{(\delta_i^z - \theta_{min}^z)}{\theta_{max}^z - \theta_{min}^z}. \end{aligned} \tag{4}$$

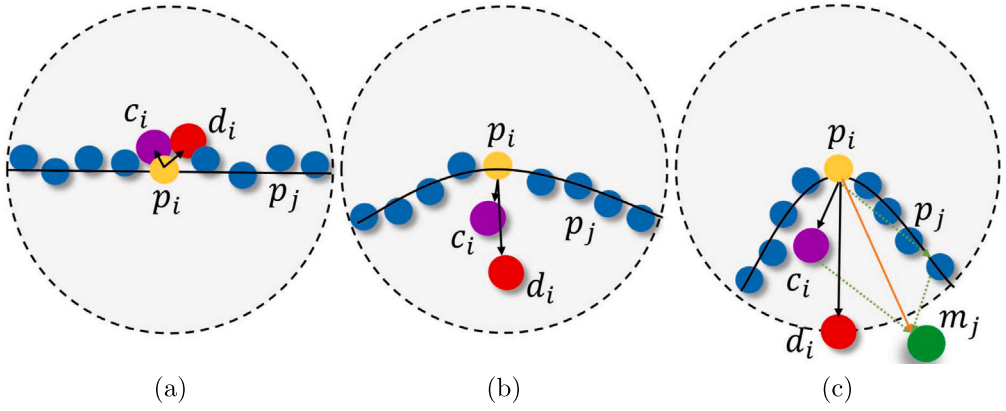


Fig. 2. Illustrations of the traditional centroid c_i (colored in purple) and our weighted centroid d_i (colored in red) of a point p_i located in (a) flat region, (b) shallow feature, and (c) sharp edge, respectively. Note that point p_j (colored in blue) is one of the k-nearest neighbor points of point p_i, and the point m_j (colored in green) in the rightest column is a temp point. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 3. Feature point identification on raw input using different metrics. From left to right: raw input, feature detection results using $\|L_i\|$, $\|\delta_i\|$, and f_i, respectively.

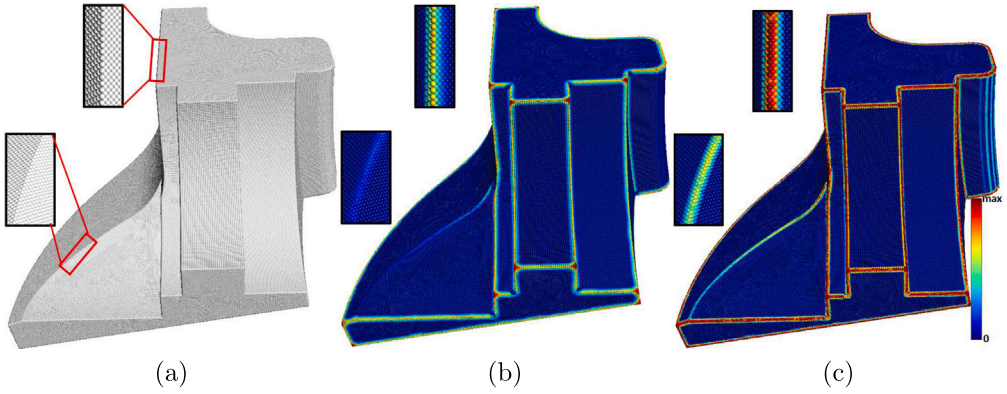


Fig. 4. Visualization of point weights computed by different metrics. From left to right: (a) input, (b) point weights computed by metric $\|L_i\|$, (c) point weights computed by metric $\|\delta_i\|$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Here, $\alpha \in [0, 1]$ is a balance parameter, $\delta_i^x, \delta_i^y, \delta_i^z$ are three components of vector δ_i, $\theta_{min}^x, \theta_{min}^y, \theta_{min}^z$ are minimum component values of the weighted centroid displacement vector of point p_i and those vectors of its $k-$neighbors. Similarly, $\theta_{max}^x, \theta_{max}^y, \theta_{max}^z$ are maximum component values. Empirically, we set $\alpha = 0.5$ by default. The formula (3) allows us to capture the independent surface variations along different axes, enabling measuring the flatness of local regions precisely. As a result, the value of κ_i of feature points tends to be larger than non-feature points.

Now, we can easily obtain feature points $\mathbf{F}$ from input point clouds via

$$\mathbf{F} = \{p_i \mid f_i \geq \epsilon_f\}, \tag{5}$$

where $\epsilon_f \geq 0$ is a threshold for determining whether a point is a feature point. We conduct a series of comparison experiments on a large-scale point cloud scene to testify the proposed method, as shown in Fig. 3. Specifically, Fig. 3(b) illustrates the feature point detection result using the metric $\|L_i\|$, where feature points in fine edges fail to

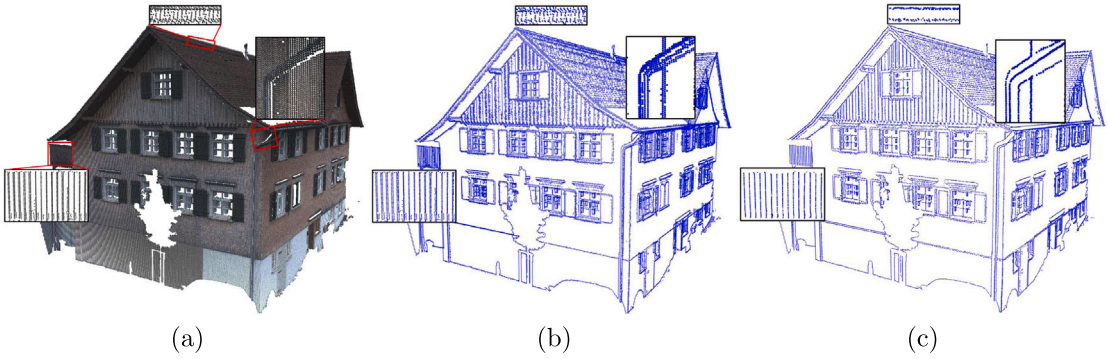


Fig. 5. From left to right: raw input, coarse feature points, and feature point optimization result.

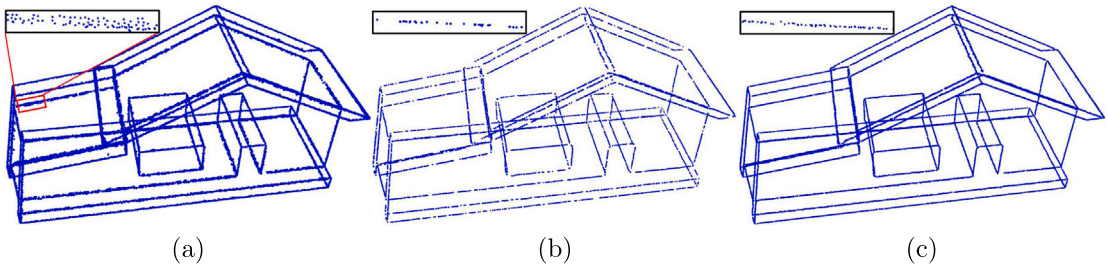


Fig. 6. From left to right: coarse feature points, the feature point optimization result by the first term of model FPO (6), and the optimization result by full model FPO (6).

be identified. Fig. 3(c) shows the feature points detection result using $\|\delta_i\|$ as the metric. As we can see, more feature points on finer edges are detected but still incomplete. In contrast, with the proposed metric f_i, our method is capable of detecting complete feature points from widely geometric characteristics, as demonstrated in Fig. 3(d).

3.1.2. Coarse feature point optimization

Due to the irregular sampling rates of point clouds (especially those large-scale point cloud scenes), the detected feature points may be redundant. Such an example can be seen in Fig. 5(b). Thus, it is necessary to optimize them further.

In general, the detected feature points are assumed to lie in the geometric edges of the underlying surface. Thus, we aim to reposition these points to make their distribution thin while not changing the shape of the input point clouds. To this end, we utilize the well-known L_1-median optimization (Lipman et al., 2007; Chen et al., 2022), designed for whole point cloud optimization. First, we denote the optimized feature points as $\mathbf{X}$. Then, we apply L_1-median optimization to formulate a feature point optimization model, abbreviated as FPO, which is written as

$$\min_{x_i} \sum_{p_k \in N^P(x_i)} e^{-\frac{\|p_k - x_i\|^2}{r^2}} \|x_i - p_k\|_2 + \mu \sum_{x_k \in N^{\mathbf{X}}(x_i)} \frac{1}{\sigma_i \|x_i - x_k\|_2}, \tag{6}$$

where $x_i \in \mathbf{X}$, $N^P(x_i)$ denotes the k-neighbor points of x_i in the input point cloud P, and $N^{\mathbf{X}}(x_i)$ denotes the k-neighbor points of x_i in the feature points $\mathbf{X}$, μ is the balance parameter, $\sigma_i = t_i^2/(t_i^0 + t_i^1 + t_i^2)$, where $t_i^0 < t_i^1 < t_i^2$ are the eigenvalues computed by applying PCA to the local neighborhood of point x_i. A larger value of σ_i indicates that more neighbor points are distributed in the principal direction. As a result, the first term can drive the optimized feature points $\mathbf{X}$ to approximate the geometry of roughly detected feature points $\mathbf{F}$, and the second term can keep the distribution of the optimized feature points $\mathbf{X}$ fair.

We apply the gradient descent method to minimize model (6). Note that we initialize $\mathbf{X} = \mathbf{F}$, and empirically set $\mu = 0.07$ and set $r = 3\bar{dis}$, where $\bar{dis}$ is the average distance between points. Moreover, we apply it to feature points, and the optimized result is demonstrated in Fig. 6. As shown in Fig. 6(b), ignoring the second term of model (6) leads to discontinuities. Conversely, our model is capable of generating a complete and thin result.

3.2. Line segment extraction

With detected feature points, line segment generation is used to accurately extract complete line segments from large-scale point clouds. To achieve this goal, we propose a coarse-to-fine generation algorithm, which first uses feature points to produce coarse line segments and then refine them from the aspects of completeness and simplicity. The whole algorithm is outlined in Algorithm 1, where the key is a series of geometric operators, including the clustering operator, exclusion operator, and assimilation operator. First, the clustering operator classifies feature points into multiple groups such that each will be used to generate a line segment. Second, the exclusion operator removes outlier line segments from the generated line segment set by recognizing inclusion relationships between line segments. Finally, the assimilation operator merges line segments to ensure their completeness and simplicity. Details of these operators are depicted as follows.

Algorithm 1: Line segment extraction

Input: detected feature points set $\mathbf{X}$, threshold ϵ_l
Output: set of final line segments **FLS**
Initialization: **FLS** $\leftarrow \emptyset$
(1) Clustering operator :
  R = RegionGrowingbyPCA(**X**);
  LS = LineSegmentGeneration(**R**);
(2) Exclusion operator :
  FilterAndUpdate(**LS**);
(3) Assimilation operator:
while (New line segment generation) **do**
  SortLineSegmentsbyLength(**LS**);
  AssimilateLineSegmentsAndUpdate(**LS**);
end
FLS = **LS**;
return FLS;

Clustering operator. The clustering algorithm aims to group the detected feature points into different clusters based on their geometric properties concerning the underlying surface of the point clouds. Here, we utilize a region-growing strategy (Haghighatgou et al., 2022) to

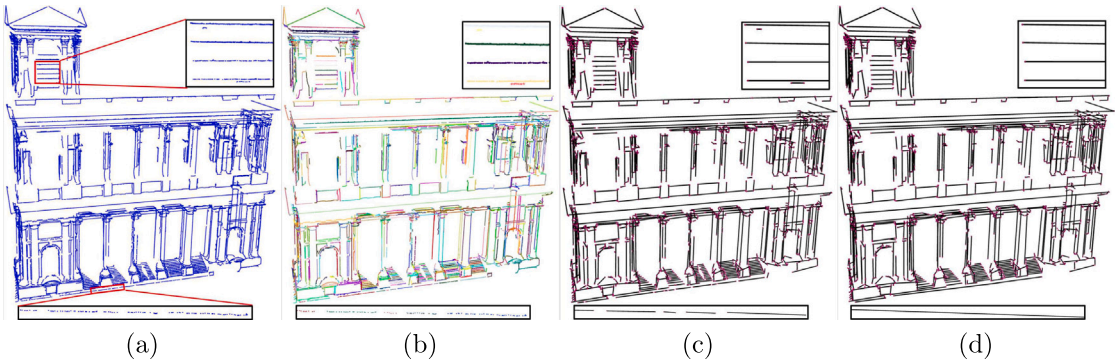


Fig. 7. Line segment generation on a large-scale point cloud. From left to right: optimized feature points, point clustering result, coarse line segments, final line segments.

group feature points into different clusters, where points within each cluster tend to lie on the same edge. Specifically, this algorithm can be detailed as: (1) For each feature point $x_i \in \mathbf{X}$, we calculate the eigenvector e_i corresponding to the largest eigenvalue of the covariance matrix of x_i as its principal direction using the PCA algorithm. (2) we use the breadth-first search strategy to obtain all the points that belong to the same cluster with point x_i. Let this cluster be denoted as v_i. Specifically, we determine whether point p_j belongs to cluster v_i or not by checking the condition: $e_j \cdot \bar{e}_i \geq 0.85$, where $\bar{e}_i = \frac{1}{|v_i|}\sum_{x_m \in v_i} e_m$ is the average principal direction of the cluster v_i. (3) When all the feature points are grouped, we can get the final clustering result $\mathbf{R} = \{v_1, v_2, v_3, \ldots, v_n\}$; see Fig. 7(b). Moreover, we can further obtain a collection of coarse line segments $\mathbf{LS} = \{l_1, l_2, l_3, \ldots, l_n\}$ by performing a least-squares fitting method on each group of points; see Fig. 7(c)

Exclusion operator. Outlier line segments are inevitably generated when processing large-scale point clouds with rich details. These outlier line segments not only waste computational resources but also cause errors in downstream applications, such as shape analysis, measurement, etc. Thus, removing them from the line segment set **LS** is necessary. The key is to recognize outlier segments correctly. Here, we propose a recognition strategy using the neighborhood relations between line segments. Specifically, we first use the distance of two midpoints to measure the distance between the corresponding line segments. Then, the outlier line segments can be recognized by

$$\mathrm{L}_{\text{outlier}} = \left\{ l_o \mid l_j \in nei(l_o), l_o \notin nei(l_j) \right\}, \tag{7}$$

where $nei(\cdot)$ denotes the k-nearest neighbor line segments ($k = 8$ by default). This means that once a neighbor line segment l_j of l_o satisfies Eq. (7), then l_o is judged as an outlier. The method checks each line segment in the neighbor set $nei(l_o)$ of l_o according to Eq. (7). Even though some line segments of $nei(l_o)$ are outliers, the result is unchanged. We show a line segment set in Fig. 8. If a line segment is an outlier, the neighborhood relations between the line segment and its k-nearest neighbor line segments are not dual.

Assimilation operator. After filtering outlier line segments, the remaining line segments still have issues in completeness and simplicity, as illustrated in Fig. 7(c). To tackle this problem, the assimilation operator merges line segments with similar geometric attributes as much as possible. For this, by taking location, length, and direction into account, we first define a metric for measuring the similarity between two adjacent line segments as

$$S(l_i, l_j) = S_\theta(l_i, l_j) + S_h(l_i, l_j) + S_d(l_i, l_j), \tag{8}$$

where l_i, l_j are two adjacent line segments, and

$$S_\theta(l_i, l_j) = |l_j| \sin\theta,$$

$$S_h(l_i, l_j) = \frac{h_1^2 + h_2^2}{h_1 + h_2},$$

$$S_d(l_i, l_j) = \begin{cases} 0, & \text{if } l_i \text{ intersects with } l_{j'} \\ d_p, & \text{else} \end{cases}.$$

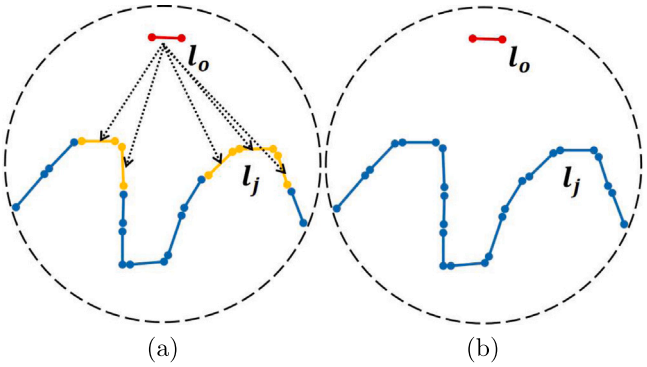


Fig. 8. A line segment set for demonstrating the outlier recognition strategy. The red line segment is the outlier, and the yellow line segments denote the neighbor line segments of the outlier by k−nearest searching. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Similar to the work proposed by Du et al. (2019), $S_\theta(l_i, l_j)$ converts angular differences into length representation; see in Fig. 9(a), θ is the angle between l_i and l_j, $|l_j|$ denotes the length of l_j. $S_h(l_i, l_j)$ is the vertical distance (Bullen, 2013) between two line segments, which is a more comprehensive way to synthesize distances and reduce the effect of extreme values. Here, h_1 and h_2 denote two projection distances of the endpoints of line segment l_j when projecting it onto line segment l_i. Inspired by Guo et al. (2022), $S_d(l_i, l_j)$ quantifies the distance between two line segments by using their projected endpoints, where d_p denotes the minimum distance between the endpoints of two line segments. In general, if two line segments l_i, l_j are similar, the metrics $S_\theta(l_i, l_j)$, $S_h(l_i, l_j)$, and $S_d(l_i, l_j)$ have small values. In other words, a smaller value of $S(l_i, l_j)$ means line segments l_i and l_j tend to be more similar.

With the proposed similarity metric (8), we present a two-stage assimilation operator based on an expansion scheme to merge similar line segments. In the first stage, for line segment l_i, we create a neighbor line segment set $\mathbf{S_i}$ by searching line segments within a specified region defined by longitudinally expanding the line segment with a radius $R = 4\bar{dis}$ and expand the line segment along its direction by $4\bar{dis}$; see Fig. 10. After that, we can obtain an assimilation set $\mathbf{S_i}$ for l_i by checking the condition $S(l_j, l_j) < \epsilon_l$. Specifically, if a line segment l_j in the neighbor line segment set satisfies this condition, we add it to $\mathbf{S_i}$. In the second stage, we generate a merged line segment l_a for assimilation set $\mathbf{S_i}$. To this end, for each assimilation set, we first compute a geometric central point o by averaging all endpoints of line segments and a direction vector by averaging the directions of all line segments. Then, we can produce a new line by using the central point and the direction vector. Finally, based on orthogonal projection, we use the two endpoints with the largest distance to clip the line to generate the merged line segment.

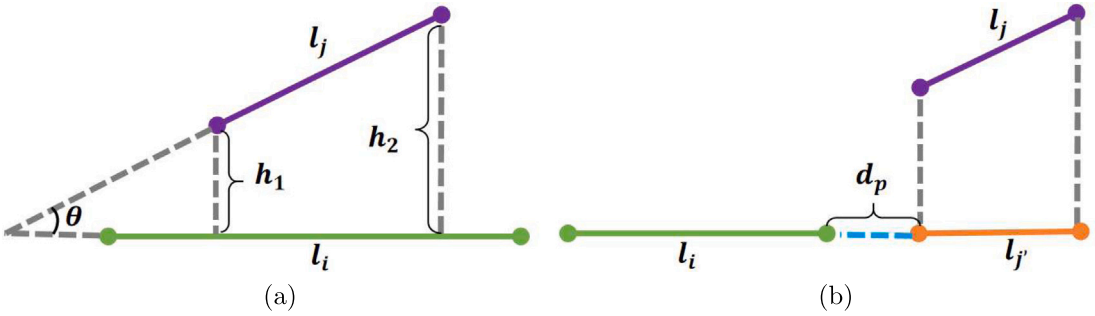


Fig. 9. The illustrations of (a) $S_\theta(l_i, l_j), S_h(l_i, l_j)$, and (b) $S_d(l_i, l_j)$.

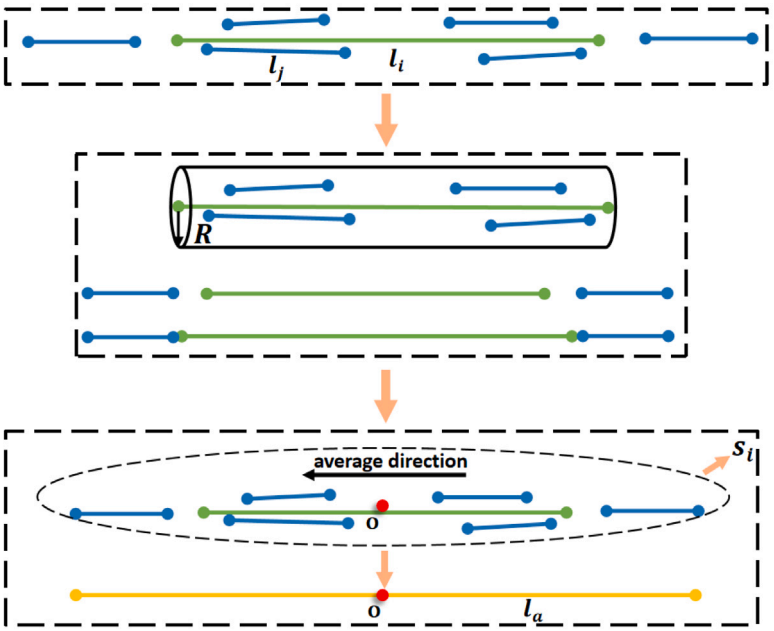


Fig. 10. From top to bottom: scatter line segment distribution, similar line segment searching, line segment assimilation. $\mathbf{S_i}$ represents the line segments set to be assimilated. o is the central point of set $\mathbf{S_i}$. l_a is the merged line segment for assimilation set $\mathbf{S_i}$.

4. Experiments

To validate the performance of our method, we have conducted a series of experiments on various datasets, including TLS, MLS, ALS, CAD, and RGB-D point clouds. TLS, MLS, and ALS point clouds generally capture large-scale outdoor scenes, while RGB-D point clouds depict indoor scenes. Besides, CAD point clouds usually represent single objects. Here, we give the details about the testing data in Table 1. All the algorithms involved are implemented in C++, and experiments are conducted on a computer with an AMD Ryzen 7 5800H CPU and 16 GB RAM.

4.1. Parameter setting

Our method involves four parameters, i.e., α, μ, K, and R. The parameter α contained in formula (4) is used for detecting feature points. $\alpha = 0.5$ is sufficient to handle most point cloud data. The parameter μ contained in formula (6) is used for optimizing the distribution of coarse feature points. We empirically set $\mu = 0.07$ for all the tested experiments.

The value of parameter K denotes the number of closest neighbor points, which impacts the accuracy of feature point detection. As shown in Fig. 11, the detected shallow features on the roof gradually disappear where K increases. Figs. 11(b) and 11(c) demonstrate that our strategy can produce satisfactory results for K in the range of $[25, 40]$.

The parameter R denotes the search range for constructing line segments, which impacts the completeness of line segment generation. In Fig. 12, we present line segment extraction results for varying R. As we can see, the detected line segments become sparser with increasing values of R. Specifically, the number of detected line segments in Figs. 12(b), 12(c) and 12(d) are 621, 303, and 226, respectively. Besides, too small R leads to the production of discontinuous line segments; see Fig. 12(b). On the contrary, too large R leads to missing some important line segments; see Fig. 12(d). Empirically, R is suggested to be set in the range of $[3\bar{dis}, 5\bar{dis}]$ for generating satisfactory results.

4.2. Performance of feature point detection

Qualitative comparison. To demonstrate the effectiveness of our feature point detection method, we conduct a series of experiments on large-scale point cloud scenes. We also compare our method with existing methods, including NRLC (Liu et al., 2021) and ED (Xia and Wang, 2017), and the results are illustrated in Figs. 13 and 14. In Fig. 13, we show the feature point detection results on TLS point clouds with sharp and shallow features. As can be seen from the zoomed views, ED and NRLC methods tend to misidentify lots of points in shallow features as non-feature points; see Figs. 13(b) and 13(c). Compared to these two methods, our method can detect most points in shallow regions as feature points. Fig. 14 shows feature point detection

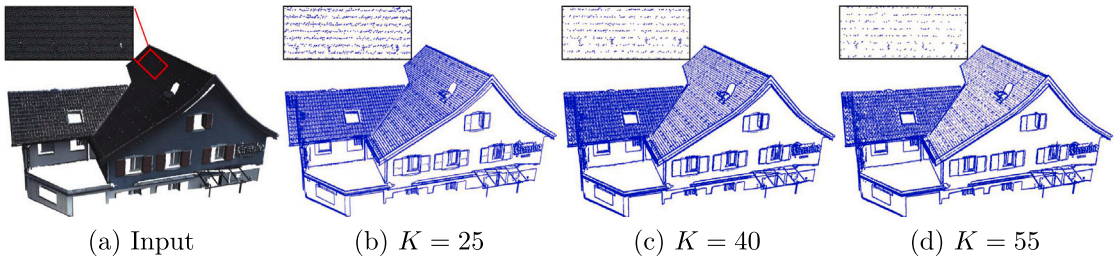


Fig. 11. Feature point detection results for varying K. We keep the same number of feature points in each result. From left to right: input point cloud and results with increasing K.

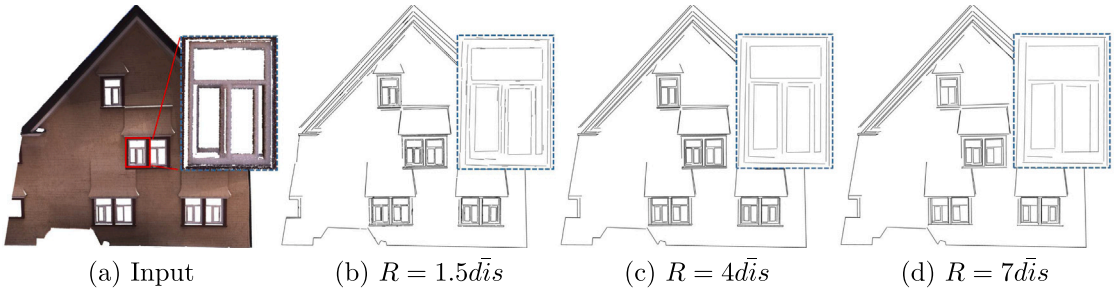


Fig. 12. Line segment extraction results for varying R. From left to right: input point cloud and results with increasing R.

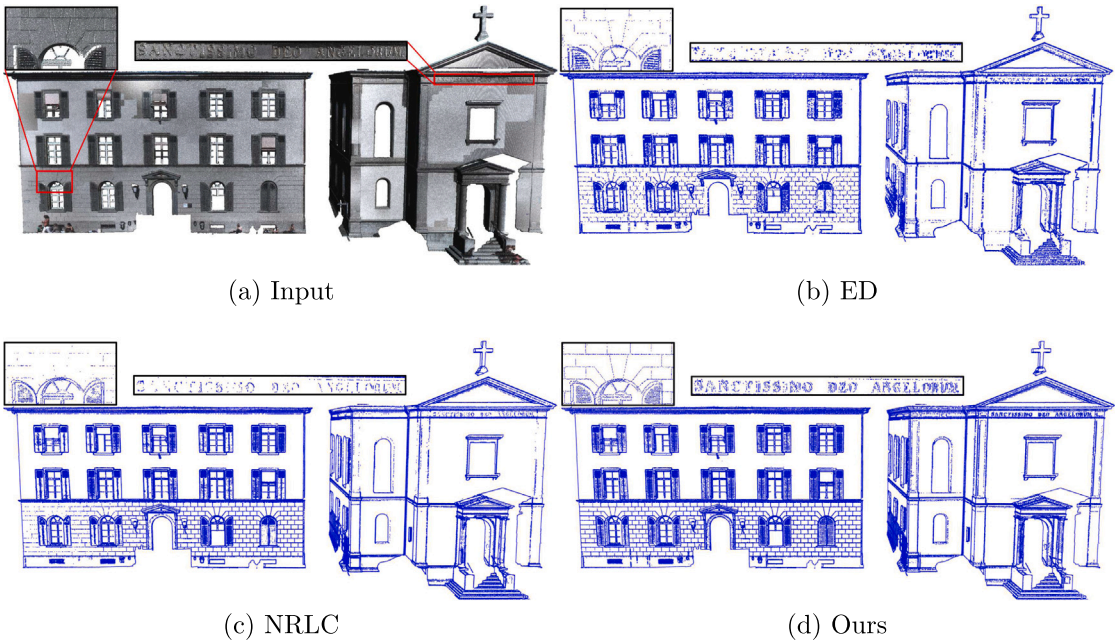


Fig. 13. Feature point detection results on TLS point clouds. Zoomed views highlight that our method can detect most points in shallow features as feature points.

results on MLS point clouds with low sampling rates. Method ED misses feature points in some geometric structures (e.g., windows) and misidentifies numerous non-feature points in smooth regions as feature points; see Fig. 14(b). Besides, although method NRLC can identify feature points in shallow features well, it misses many feature points in salient geometric features, leading to discontinuous detection results; see Fig. 14(c). In contrast, our method also can accurately detect the most complete feature points. Thus, the above results demonstrate that our method outperforms the compared methods in handling large-scale point clouds.

Evaluation of feature detection. To further evaluate the performance of our method and those competing methods, we employ three metrics (Liu et al., 2021), including precision (Pre), recall (Rec), and the F1-score ($F1$). Specifically, Pre and Rec are defined as $Pre = \frac{TP}{TP+FP}$ and $Rec = \frac{TP}{TP+FN}$, respectively. Here, TP, FP, and FN are numbers of correctly identified, incorrectly identified, and erroneously rejected points. Theoretically, Pre measures the accuracy of feature point detection, Rec quantifies the completeness of detected feature points, and $F1 = \frac{2Pre \cdot Rec}{Pre+Rec}$ denotes the trade-off between precision and recall rate. In general, the larger values of these metrics mean the better performance of the corresponding method.

Table 2 lists the quantitative evaluation results. As we can see, all the metric values of method ED are the smallest ones, while all the metric values of our method are the largest ones. The above shows that our method performs best on these point clouds. Moreover, we record the run time of these methods in the last column of the table. The cost

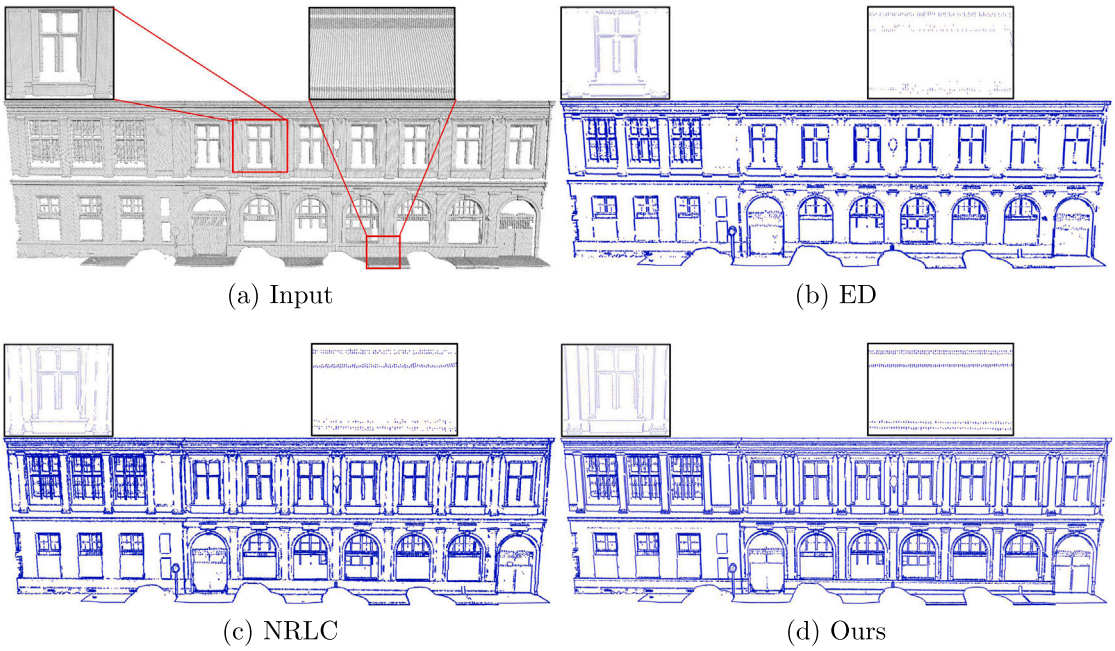


Fig. 14. Feature point detection results on MLS point clouds. Zoomed views highlight that our method can detect more complete feature points from point clouds with low sampling rates.

Table 1
Details of the testing data, including type, number of points, station, acquisition equipment, and reference links.

Figure	Data type	Number of points	Station	Acquisition equipment
13[a]	TLS	2153.7K	Multi-station	Leica P40
14[b]	MLS	437.5K	Moving	Stereopolis II
15[a]	TLS	2748.2K	Multi-station	Leica P40
16[c]	TLS	1114.5K	Multi-station	Trimble TX8
17[a]	TLS	212.2K	Multi-station	Leica P40
19[d]	CAD	2123.3K	–	OnShape CAD
20(a)[e]	RGB-D	901.2K	–	Synthetic
20(b)[f]	ALS	1675.6K	Moving	Riegl LMS-Q680i

[a] https://www.semantic3d.net/.
[b] http://data.ign.fr/benchmarks/UrbanAnalysis/.
[c] http://www.libe57.org/data.html.
[d] https://deep-geometry.github.io/abc-dataset/.
[e] http://redwood-data.org/indoor/dataset.html.
[f] https://geo.nyu.edu/.

Table 2
Quantitative comparison and running time of our method with ED (Xia and Wang, 2017) and NRLC (Liu et al., 2021) on results in Figs. 13, 14.

Figure	Method	Metrics			Time (s)
		$P-Pre\uparrow$	$P-Rec\uparrow$	$P-F1\uparrow$	
13	ED	0.68	0.63	0.65	41.57
	NRLC	0.75	0.69	0.72	22.53
	Ours	**0.86**	**0.80**	**0.83**	23.92
14	ED	0.71	0.66	0.68	8.71
	NRLC	0.79	0.75	0.77	4.85
	Ours	**0.88**	**0.83**	**0.85**	4.96

of our method is closest to method NRLC, which is the fastest one. This verifies the high efficiency of our method.

4.3. Performance of line segment generation

Qualitative comparison. We evaluate the performance of our line segment generation method on various real-world point cloud scenes and compare it with methods ED (Xia and Wang, 2017) and F3D (Lu et al., 2019). The comparison results are shown in Figs. 15, 16 and 17. In Fig. 15, we show line segment generation results on a point cloud with rich details. As we can see, method F3D fails to extract complement line segments due to its inadequate point cloud segmentation when handling surfaces with rich details; see Fig. 15(b). Although ED generates more line segments, the continuity of many line segments is not preserved. Conversely, our method generates the most line segments with the continuity kept. In Fig. 16, we show line segment generation results on a point cloud with multi-scale features. As we can see, though both ED and F3D can produce line segments at large-scale geometric features, they fail to extract line segments at small-scale geometric features; see Figs. 16(b) and 16(c). This problem is more serious for ED. Compared to these two methods, our method can successfully generate line segments at various scales; see Fig. 16(d). In Fig. 17, we show line segment generation results on a point cloud with low sampling rates. As we can see, though methods F3D and ED can extract the contour edges of the model, both of them omit many line segments at small-scale geometric features; see Figs. 17(b) and 17(c). Conversely, our method is not only able to extract the contour edges but also generate line segments at small-scale geometric features; see Fig. 17(d). Thus, the above results demonstrate that our method outperforms the compared methods in handling large-scale point clouds, especially those with low sampling rates, rich details, and multi-scale geometric features.

Quantitative evaluation. To further evaluate the performance of our method, we adopt two metrics, including precision (Pre) and recall (Rec), which are described in the work proposed by Lin et al. (2017). Specifically, we first manually extract feature points from the original point cloud and use them as ground-truth points. Then, Pre represents the proportion of points on the line segment that are ground-truth points, while Rec represents the proportion of ground-truth points located on the line segment. We use a distance threshold $t = 3\bar{dis}$ to determine whether a point lies on a line segment. Using these metrics, we compared the performance of our method with methods ED and F3D.

The quantitative evaluation results are listed in Table 3. As we can see, both F3D and ED achieve higher Pre values than Rec, which indicates that they produce fewer false positive line segments but more false negative ones. For example, as illustrated in Fig. 17, although they

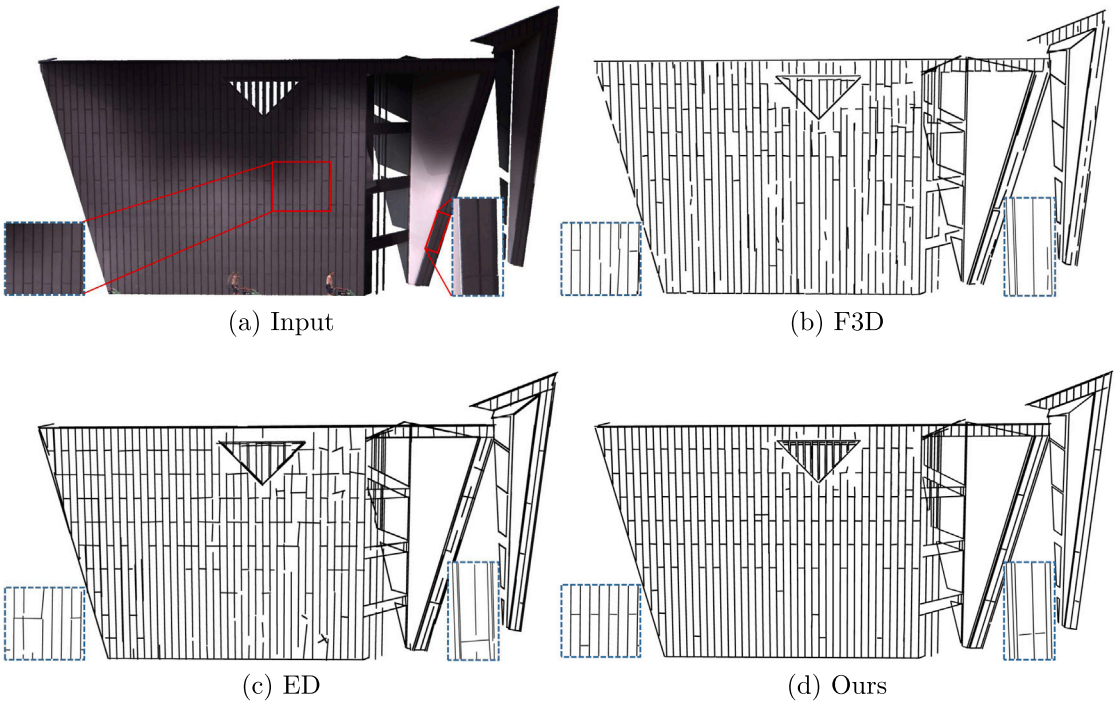


Fig. 15. Line segment generation results on a point cloud scene with rich details. Zoomed views highlight that our method can generate more complete line segments.

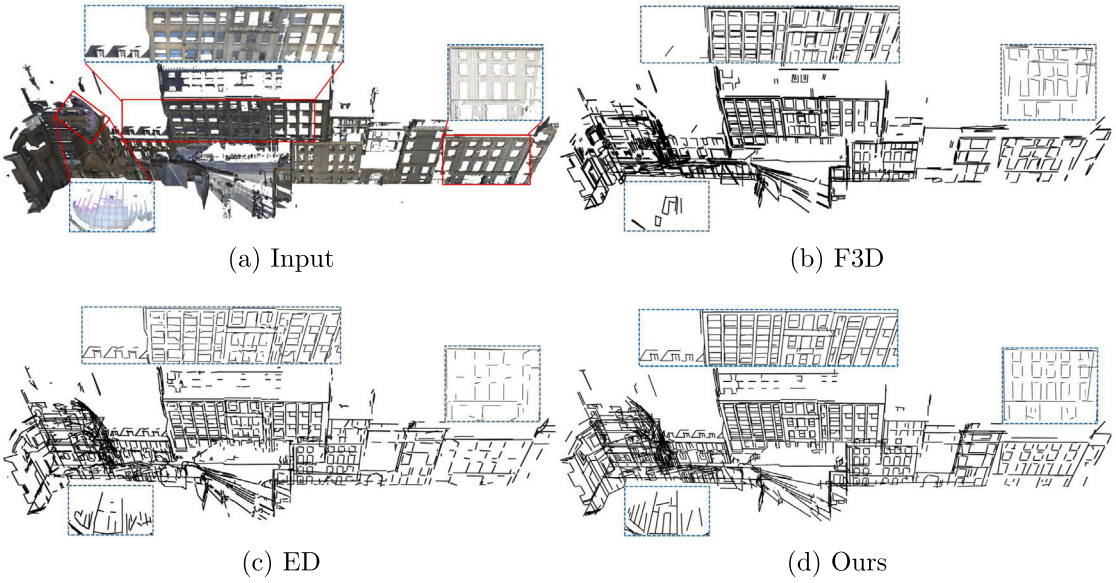


Fig. 16. Line segment generation results on a point cloud with multi-scale features. Zoomed views highlight that our method can extract line segments at varying scales.

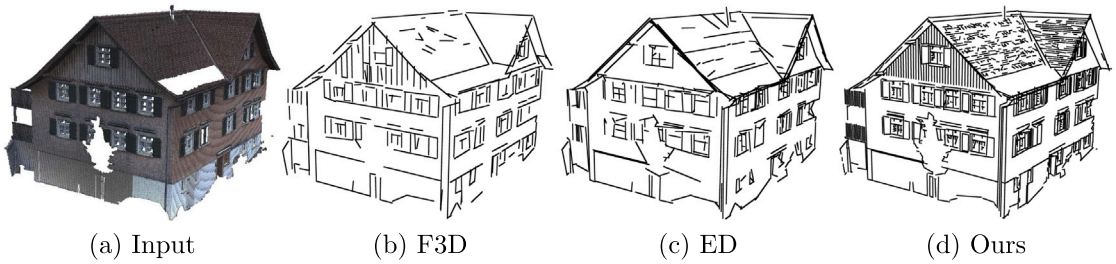


Fig. 17. Line segment generation results on a point cloud with low sampling rates, highlighting that our method can extract more complete line segments.

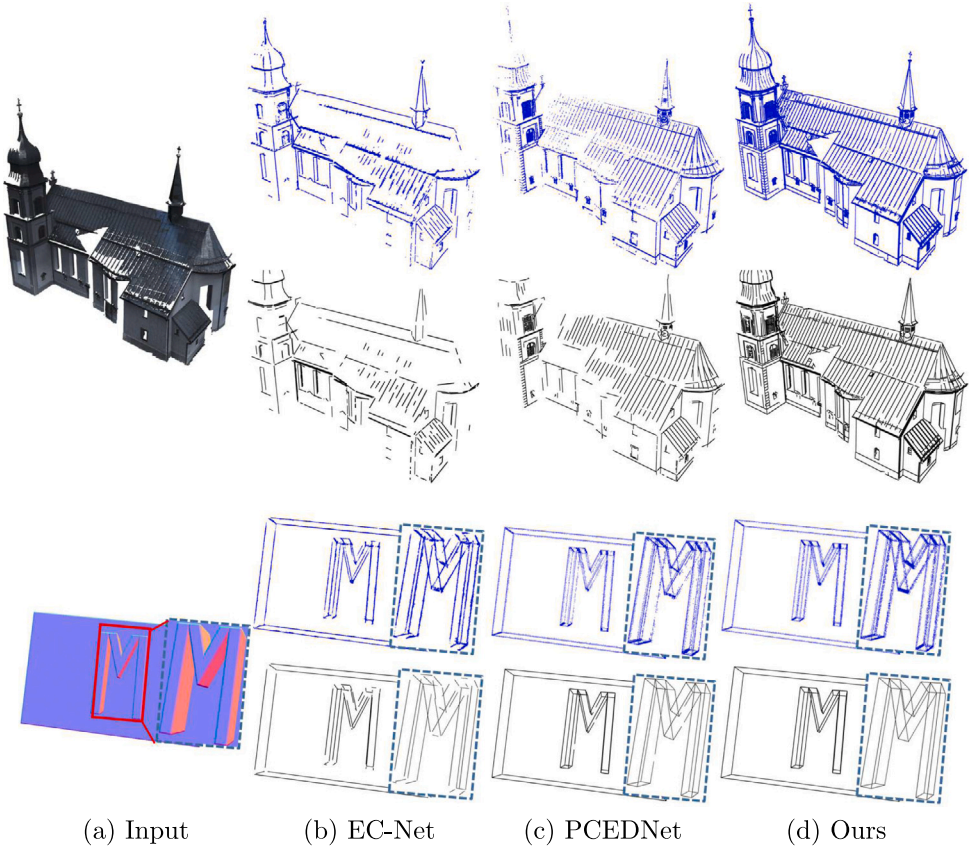


Fig. 18. Visual comparison of deep-learning methods (EC-Net and PCEDNet) with our method on Bildstein and M-cube, demonstrating that our method can produce more accurate feature points (colored in blue) and complete line segments (colored in black) on both small-scale and structure features. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Quantitative evaluation of line segment generation methods including F3D (Lu et al., 2019), ED (Xia and Wang, 2017), and ours on results in Figs. 15, 16, and 17.

Figure	Method	Metrics		Number of lines	Time (s)
		$L-Pre\uparrow$	$L-Rec\uparrow$		
15	F3D	0.73	0.71	598	85.05
	ED	0.83	0.78	498	90.13
	Ours	**0.89**	**0.87**	411	55.73
16	F3D	0.72	0.67	1979	35.89
	ED	0.83	0.73	2198	38.95
	Ours	**0.87**	**0.84**	1762	26.69
17	F3D	0.70	0.36	282	7.61
	ED	0.69	0.43	543	8.46
	Ours	**0.87**	**0.83**	866	5.57

Table 4
Quantitative comparison of our method with deep-learning methods (EC-Net and PCEDNet) on results in Fig. 18. *Pre*: precision; *Rec*: recall; ↑: the higher score, the better.

Model	Method	Metrics				Time (s)
		$P-Pre\uparrow$	$P-Rec\uparrow$	$L-Pre\uparrow$	$L-Rec\uparrow$	
Bildstein	EC-Net	0.77	0.49	0.82	0.55	429.37
	PCEDNet	0.83	0.63	0.86	0.68	78.51
	Ours	**0.84**	**0.81**	**0.86**	**0.84**	35.63
M-cube	EC-Net	0.87	0.79	0.91	0.82	37.34
	PCEDNet	0.90	0.89	0.95	0.91	9.10
	Ours	**0.93**	**0.94**	**0.97**	**0.95**	2.07

omit many line segments, they can ensure that the existing detected line segments are as correct as possible. On the contrary, our method achieves the largest values of *Pre* and *Rec*, demonstrating its superiority when handling large-scale point clouds, especially those with low sampling rates, rich details, and multi-scale geometric features. Besides, we also record the run time in the last column of the table. As we can see, the run time of method ED is slightly larger than that of method F3D, but our method is much faster than these two methods.

4.4. Comparison with deep-learning methods

We compare our method's performance to current deep-learning approaches, such as EC-Net (Yu et al., 2018) and PCEDNet (Himeur et al., 2021). For fairness, we employ the proposed Algorithm 1 to extract line segments from the detected point sets produced by EC-Net and PCEDNet. Fig. 18 shows the comparison results. EC-Net and PCEDNet cannot properly recognize feature points at small-scale features and boundaries, resulting in incomplete line segment extraction, as shown in Figs. 18(b) and 18(c). Contrarily, benefiting from the proposed weighted centroid displacement scheme (1) and FPO (6), our method can detect more accurate feature points and generate complete line segments, especially in shallow features. In summary, our method produces results with significantly higher visual quality and numerical measures, demonstrating that it outperforms the deep-learning methods (see Table 4).

4.5. Generalization tests

Aside from the cases tested above (e.g., buildings), we also evaluate our method's performance on a broader variety of point clouds, including object- and scene-level data. In Fig. 19, we exhibit line segment

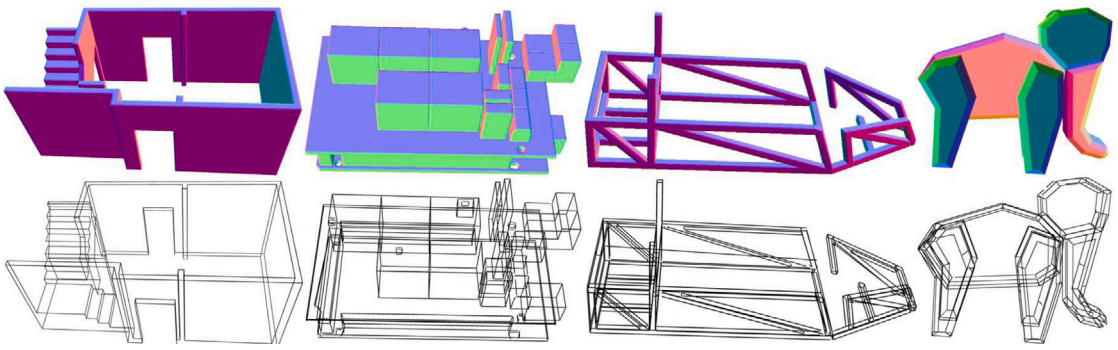


Fig. 19. Line segment extraction results on CAD models.

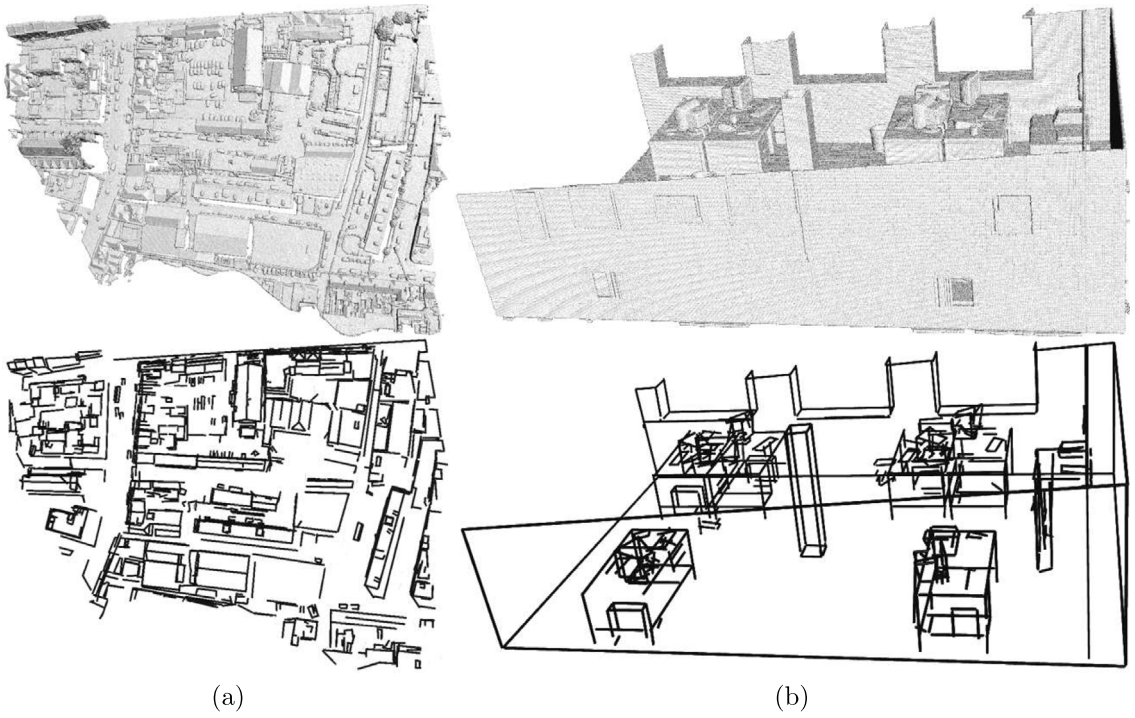


Fig. 20. Line segment extraction results from scene point clouds.

extraction results on CAD models obtained from ABC dataset (Choi et al., 2015). We can see that our method can successfully extract line segments from object-level point clouds with rich geometric information at various sizes. In Fig. 20, we show line segment extraction results on point cloud scenes from Augmented ICL-NUIM (Choi et al., 2015) dataset and DublinCity dataset (Laefer et al., 2017). We can observe that our method is similarly effective in extracting line segments from scene-level point clouds with rich geometric characteristics and complex contours. These findings show that our method has excellent generality.

5. Conclusion

In this work, we present a two-stage method to extract high-quality line segments from large-scale point clouds based on a weighted centroid displacement scheme and three geometric operators. The proposed weighted centroid scheme can be used to efficiently recognize feature points from point clouds, especially in those regions with shallow features. The proposed geometric operators, namely the cluster, exclusion, and assimilation operators, are utilized to ensure the simplicity, accuracy, and completeness of the generated line segments. As demonstrated in extensive experimental results, our method outperforms the existing state-of-the-art approaches in accuracy and efficiency. For future study, we may extend our work from two aspects. First, we intend to extract curved lines from point clouds to broaden the applicability of the proposed method. Second, we plan to investigate the possibility of extending our approach by using deep learning techniques.

CRediT authorship contribution statement

Xiaopeng Xin: Conceptualization, Methodology. **Wei Huang:** Methodology. **Saishang Zhong:** Writing – original draft. **Ming Zhang:** Funding acquisition. **Zheng Liu:** Project administration, Writing – review & editing. **Zhong Xie:** Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was supported by National Key R&D Program of China (2022YFB3904100), National Natural Science Foundation of China (62302350), Guangdong Enterprise Key Laboratory for Urban Sensing, Monitoring and Early Warning (2020B121202019), and Open Research Project of the Hubei Key Laboratory of Intelligent Geo-Information Processing (KLIGIP-2023-B10).

References

Adam, J.M., Liu, W., Zang, Y., Afzal, M.K., Bello, S.A., Muhammad, A.U., Wang, C., Li, J., 2023. Deep learning-based semantic segmentation of urban-scale 3D meshes in remote sensing: A survey. Int. J. Appl. Earth Obs. Geoinf. 121, 103365. http://dx.doi.org/10.1016/j.jag.2023.103365.

Bullen, P., 2013. Handbook of Means and their Inequalities, vol. 560, Springer Science & Business Media, http://dx.doi.org/10.1007/978-94-017-0399-4.

Chen, H., Huang, Y., Xie, Q., Liu, Y., Zhang, Y., Wei, M., Wang, J., 2022. Multiscale feature line extraction from raw point clouds based on local surface variation and anisotropic contraction. IEEE Trans. Autom. Sci. Eng. 19, 1003–1016. http://dx.doi.org/10.1109/TASE.2021.3053006.

Chen, Y., Wang, J., Li, J., Lu, C., Luo, Z., Xue, H., Wang, C., 2018. LiDAR-video driving dataset: Learning driving policies effectively. In: IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 5870–5878. http://dx.doi.org/10.1109/CVPR.2018.00615.

Chen, X., Yu, K., 2019. Feature line generation and regularization from point clouds. IEEE Trans. Geosci. Remote Sens. 57, 9779–9790. http://dx.doi.org/10.1109/TGRS.2019.2929138.

Choi, S., Zhou, Q.-Y., Koltun, V., 2015. Robust reconstruction of indoor scenes. In: IEEE Conference on Computer Vision and Pattern Recognition. CVPR, http://dx.doi.org/10.1109/CVPR.2015.7299195.

Cui, Y., Li, Q., Yang, B., Xiao, W., Chen, C., Dong, Z., 2019. Automatic 3D reconstruction of indoor environment with mobile laser scanning point clouds. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 12 (8), 3117–3130. http://dx.doi.org/10.1109/JSTARS.2019.2918937.

Du, J., Chen, D., Wang, R., Peethambaran, J., Mathiopoulos, P.T., Xie, L., Yun, T., 2019. A novel framework for 2.5-D building contouring from large-scale residential scenes. IEEE Trans. Geosci. Remote Sens. 57 (6), 4121–4145. http://dx.doi.org/10.1109/TGRS.2019.2901539.

Guo, J., Liu, Y., Song, X., Liu, H., Zhang, X., Cheng, Z., 2022. Line-based 3D building abstraction and polygonal surface reconstruction from images. IEEE Trans. Vis. Comput. Graphics 1–15. http://dx.doi.org/10.1109/TVCG.2022.3230369.

Hackel, T., Wegner, J.D., Schindler, K., 2017. Joint classification and contour extraction of large 3D point clouds. ISPRS J. Photogramm. Remote Sens. (ISSN: 0924-2716) 130, 231–245. http://dx.doi.org/10.1016/j.isprsjprs.2017.05.012.

Haghighatgou, N., Daniel, S., Badard, T., 2022. A method for automatic identification of openings in buildings facades based on mobile LiDAR point clouds for assessing impacts of floodings. Int. J. Appl. Earth Obs. Geoinf. (ISSN: 1569-8432) 108, 102757. http://dx.doi.org/10.1016/j.jag.2022.102757.

Himeur, C.-E., Lejemble, T., Pellegrini, T., Paulin, M., Barthe, L., Mellado, N., 2021. PCEDNet: A lightweight neural network for fast and interactive edge detection in 3D point clouds. ACM Trans. Graph. 41, 1–21. http://dx.doi.org/10.1145/3481804.

Hofer, M., Maurer, M., Bischof, H., 2017. Efficient 3D scene abstraction using line segments. Comput. Vis. Image Underst. 157, 167–178. http://dx.doi.org/10.1016/j.cviu.2016.03.017.

Hu, Z., Chen, C., Yang, B., Wang, Z., Ma, R., Wu, W., Sun, W., 2022. Geometric feature enhanced line segment extraction from large-scale point clouds with hierarchical topological optimization. Int. J. Appl. Earth Obs. Geoinf. 112, 102858. http://dx.doi.org/10.1016/j.jag.2022.102858.

Laefer, D.F., Abuwarda, S., Vo, A.-V., Truong-Hong, L., Gharibi, H., 2017. 2015 Aerial laser and photogrammetry survey of dublin city collection record. http://dx.doi.org/10.17609/N8MQ0N.

Lehtola, V.V., Koeva, M., Elberink, S.O., Raposo, P., Virtanen, J.-P., Vahdatikhaki, F., Borsci, S., 2022. Digital twin of a city: Review of technology serving city needs. Int. J. Appl. Earth Obs. Geoinf. 102915. http://dx.doi.org/10.1016/j.jag.2022.102915.

Lei, X., Guan, H., Ma, L., Yu, Y., Dong, Z., Gao, K., Delavar, M.R., Li, J., 2022. WSPointnet: A multi-branch weakly supervised learning network for semantic segmentation of large-scale mobile laser scanning point clouds. Int. J. Appl. Earth Obs. Geoinf. 115, 103129. http://dx.doi.org/10.1016/j.jag.2022.103129.

Li, H., Guan, H., Ma, L., Lei, X., Yu, Y., Wang, H., Delavar, M.R., Li, J., 2023a. MVPNet: A multi-scale voxel-point adaptive fusion network for point cloud semantic segmentation in urban scenes. Int. J. Appl. Earth Obs. Geoinf. 122, 103391. http://dx.doi.org/10.1016/j.jag.2023.103391.

Li, J., Wu, W., Yang, B., Zou, X., Yang, Y., Zhao, X., Dong, Z., 2023b. WHU-Helmet: A helmet-based multi-sensor SLAM dataset for the evaluation of real-time 3D mapping in large-scale GNSS-denied environments. IEEE Trans. Geosci. Remote Sens. http://dx.doi.org/10.1109/TGRS.2023.3275307.

Li, Q., Zhuang, Y., 2023. An efficient image-guided-based 3D point cloud moving object segmentation with transformer-attention in autonomous driving. Int. J. Appl. Earth Obs. Geoinf. 123, 103488. http://dx.doi.org/10.1016/j.jag.2023.103488.

Lin, Y., Wang, C., Chen, B., Zai, D., Li, J., 2017. Facet segmentation-based line segment extraction for large-scale point clouds. IEEE Trans. Geosci. Remote Sens. 55, 4839–4854. http://dx.doi.org/10.1109/TGRS.2016.2639025.

Lin, Y., Wang, C., Cheng, J., Chen, B., Jia, F., Chen, Z., Li, J., 2015. Line segment extraction for large scale unorganized point clouds. ISPRS J. Photogramm. Remote Sens. 102, 172–183. http://dx.doi.org/10.1016/j.isprsjprs.2014.12.027.

Lipman, Y., Cohen-Or, D., Levin, D., Tal-Ezer, H., 2007. Parameterization-free projection for geometry reconstruction. ACM Trans. Graph. 26 (3), 22. http://dx.doi.org/10.1145/1276377.1276405.

Liu, Z., Xiao, X., Zhong, S., Wang, W., Li, Y., Zhang, L., Xie, Z., 2020. A feature-preserving framework for point cloud denoising. Comput.-Aided Des. 127, 102857. http://dx.doi.org/10.1016/j.cad.2020.102857.

Liu, T., Yang, Z., Hu, S., Zhang, Z., Xiao, C., Guo, X., Yang, L., 2021. Neighbor reweighted local centroid for geometric feature identification. IEEE Trans. Vis. Comput. Graphics 29, 1545–1558. http://dx.doi.org/10.1109/TVCG.2021.3124911.

Lu, X., Liu, Y., Li, K., 2019. Fast 3D line segment detection from unorganized point cloud. http://dx.doi.org/10.48550/arXiv.1901.02532, arXiv preprint arXiv:1901.02532.

Ma, Y., Zhao, X., Li, H., Gu, Y., Lang, X., Liu, Y., 2023. RoLM: Radar on LiDAR map localization. In: Proceedings of the International Conference on Robotics and Automation. ICRA, pp. 3976–3982. http://dx.doi.org/10.1109/ICRA48891.2023.10161203.

Matveev, A., Rakhimov, R., Artemov, A., Bobrovskikh, G., Egiazarian, V., Bogomolov, E., Panozzo, D., Zorin, D., Burnaev, E., 2022. DEF: Deep estimation of sharp geometric features in 3D shapes. ACM Trans. Graph. 41 (4), 1–22. http://dx.doi.org/10.1145/3528223.3530140.

Moghadam, P., Bosse, M., Zlot, R., 2013. Line-based extrinsic calibration of range and image sensors. In: Proceedings of the International Conference on Robotics and Automation. ICRA, pp. 3685–3691. http://dx.doi.org/10.1109/ICRA.2013.6631095.

Ni, H., Lin, X., Ning, X., Zhang, J., 2016. Edge detection and feature line tracing in 3D-point clouds by analyzing geometric properties of neighborhoods. Remote Sens. 8 (9), 710. http://dx.doi.org/10.3390/rs8090710.

Nie, J., 2016. Extracting feature lines from point clouds based on smooth shrink and iterative thinning. Graph. Models 84, 38–49. http://dx.doi.org/10.1016/j.gmod.2016.04.001.

Tang, S., Zhu, Q., Chen, W., Darwish, W., Wu, B., Hu, H., Chen, M., 2016. Enhanced RGB-D mapping method for detailed 3D modeling of large indoor environments. ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci. 1, 151–158. http://dx.doi.org/10.5194/isprs-annals-III-1-151-2016.

Tian, P., Hua, X., Tao, W., Zhang, M., 2022. Robust extraction of 3D line segment features from unorganized building point clouds. Remote Sens. 14, 3279. http://dx.doi.org/10.3390/rs14143279.

Wang, S., Liu, X., Zhang, Y., Li, J., Zou, S., Wu, J., Tao, C., Liu, Q., Cai, G., 2023. Semantic-guided 3D building reconstruction from triangle meshes. Int. J. Appl. Earth Obs. Geoinf. (ISSN: 1569-8432) 119, 103324. http://dx.doi.org/10.1016/j.jag.2023.103324.

Wang, X., Xu, Y., Xu, K., Tagliasacchi, A., Zhou, B., Mahdavi-Amiri, A., Zhang, H., 2020. PIE-net: Parametric inference of point cloud edges. In: Proceedings of the Advances in Neural Information Processing Systems. NeurIPS, Vol. 33, pp. 20167–20178. http://dx.doi.org/10.5555/3495724.3497417.

Wu, H., Zhu, Q., Guo, Y., Zheng, W., Zhang, L., Wang, Q., Zhou, R., Ding, Y., Wang, W., Pirasteh, S., et al., 2022. Multi-level voxel representations for digital twin models of tunnel geological environment. Int. J. Appl. Earth Obs. Geoinf. 112, 102887. http://dx.doi.org/10.1016/j.jag.2022.102887.

Xia, S., Wang, R., 2017. A fast edge extraction method for mobile lidar point clouds. IEEE Geosci. Remote Sens. Lett. 14, 1288–1292. http://dx.doi.org/10.1109/LGRS.2017.2707467.

Yu, L., Li, X., Fu, C.-W., Cohen-Or, D., Heng, P.-A., 2018. EC-net: an edge-aware point set consolidation network. In: Proceedings of the European Conference on Computer Vision. ECCV, http://dx.doi.org/10.1007/978-3-030-01234-2_24.

Zhang, W., Chen, L., Xiong, Z., Zang, Y., Li, J., Zhao, L., 2020. Large-scale point cloud contour extraction via 3D guided multi-conditional generative adversarial network. ISPRS J. Photogramm. Remote Sens. http://dx.doi.org/10.1016/j.isprsjprs.2020.04.003.

Zhao, X., Yang, S., Huang, T., Chen, J., Ma, T., Li, M., Liu, Y., 2022. SuperLine3D: Self-supervised line segmentation and description for LiDAR point cloud. In: Proceedings of the European Conference on Computer Vision. ECCV, pp. 263–279. http://dx.doi.org/10.1007/978-3-031-20077-9_16.