Line-based 3D Building Abstraction and Polygonal Surface Reconstruction from Images

Jianwei Guo, Yanchao Liu, Xin Song, Haoyu Liu, Xiaopeng Zhang, Zhanglin Cheng

Abstract—Textureless objects, repetitive patterns and limited computational resources pose significant challenges to man-made structure reconstruction from images, because feature-points-based reconstruction methods usually fail due to the lack of distinct texture or ambiguous point matches. Meanwhile multi-view stereo approaches also suffer from high computational complexity. In this paper, we present a new framework to reconstruct 3D surfaces for buildings from multi-view images by leveraging another fundamental geometric primitive: line segments. To this end, we first propose a new multi-resolution line segment detector to extract 2D line segments from each image. Then, we construct a 3D line cloud by introducing an improved Line3D++ algorithm to match 2D line segments from different images. Finally, we reconstruct a complete and manifold surface mesh from 3D line segments by formulating a *Bayesian probabilistic modeling problem*, which accurately generates a set of underlying planes. This output model is simple and has low performance requirements for hardware devices. Experimental results demonstrate the validity of the proposed approach and its ability to generate abstract and compact surface meshes from the 3D line cloud with low computational costs.

Index Terms—3D reconstruction, 3D Line cloud, Scene abstraction, Polygonal mesh model

1 Introduction

Image-based 3D urban scene reconstruction is an important problem in 3D vision and computer graphics. It is still an active field of research because the 3D urban scene, particularly buildings, is essential for a variety of real-world applications, such as city planning, visualization, navigation, and simulations, etc.

A widely proven successful pipeline for automatic urban building reconstruction consists of two steps: the *Structure from Motion* (SFM) [1], [2], [3] first robustly obtains camera poses and sparse 3D point cloud, while the subsequent *Multiple View Stereo* [4], [5], [6] (MVS) generates a dense point cloud for recovering fine shapes. Based on the standard SFM+MVS pipeline, many open-source packages (*e.g.*, Bundler [7], VisualSFM [8], COLMAP [9], [10], MVE [11], PMVS [4], [12]) and commercial software (*e.g.*, Agisoft photoscan, Pix4D) have been developed to make significant advances in this domain; see [13] for a detailed performance evaluation of different methods. Nonetheless, existing classic reconstruction techniques have several serious limitations. First, this feature-points-based reconstruction pipeline often fails when dealing with man-made environments which typically consist of large textureless surfaces or repetitive parts. As a result, the point matching process is difficult and less reliable, thus leading to sparse and low-quality surface meshes with holes and inaccuracies. Second, the MVS step is computationally expensive. Even top-performing MVS approaches would not cope in terms of memory and computation time, especially for large-scale urban scene reconstruction. In addition, the dense triangular mesh reconstructed from the MVS point cloud may be redundant, *i.e.*, a simple planar surface is easily presented by hundreds of thousands of faces. It requires a large memory size to store or visualize such meshes.

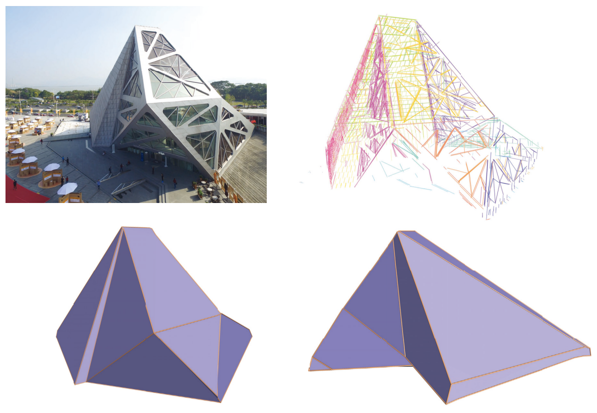


Fig. 1. From multi-view images, we propose an algorithm to generate a 3D line cloud (top right, where the color indicates different candidate planes) to abstract the scene, from which we can further reconstruct a polygonal surface model (bottom shows the mesh from two viewpoints).

Recently, several research works have demonstrated that straight lines can be successfully used in camera pose estimation [14], [15], structure from motion [16], [17], [18], line-based SLAM [19], [20], [21], and 3D reconstruction [22], [23], [24], [25]. As man-made objects primarily comprise planar faces, line segments preserve more important structural

- *Jianwei Guo and Xiaopeng Zhang are with NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China. (Jianwei Guo and Yanchao Liu are joint first authors with equal contribution.)*
- *Yanchao Liu and Haoyu Liu are with School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China, and Institute of Automation, Chinese Academy of Sciences.*
- *Xin Song and Zhanglin Cheng are with the Shenzhen Key Laboratory of Visual Computing and Analytics (VisuCA), Shenzhen Institute of Advanced Technology (SIAT), Chinese Academy of Sciences, Shenzhen 518055, China. (Zhanglin Cheng is the corresponding author. E-mail: zl.cheng@siat.ac.cn)*

and semantic information of a scene than point features. Moreover, line features are more robust to poorly-textured objects, illumination change, and wide baselines, because the position and orientation of line segments can be rather accurately determined [18]. Therefore, man-made environments (indoor and outdoor scenes where linear structures frequently occur) can be well described and abstracted by the network of 3D line segments.

Inspired by this observation, in this paper we present a new framework for 3D line cloud generation and lightweight surface model reconstruction from those line segments. We first propose a new 2D line segment detector, which combines the advantages of line segment detection on both low and high resolution images. A clustering approach is also conducted to merge similar and redundant line segments. As a result, in terms of line segment continuity and integrity, the proposed approach has evident advantages compared with the commonly used line segment detectors. Second, we introduce an efficient 3D line cloud reconstruction method by matching 2D line segments across neighboring views. The resultant 3D line cloud abstractions provide a compact representation of the scene, which reveals high-level structures hidden in the raw data. Finally, to obtain a mesh representation further, we generate a set of candidate planes from the 3D line cloud, where we cast the 3D line segments grouping as a Bayesian probabilistic modeling problem. This step is the core of our method. Then we construct the lightweight and manifold 3D mesh model using PolyFit [26] based on binary linear programming. Benefiting from our approach, we can obtain a memory-efficient representation of 3D surfaces to meet the performance requirements of various applications. In summary, the main contributions of this work include the following:

- A novel framework for efficient 3D building abstraction and polygonal surface reconstruction from a set of images by taking advantage of line segments.
- An improved 3D line cloud generation approach with a robust multi-resolution 2D line segment detector (MR-LSD). The more complete and continuous 2D line segments can be aggregated into significant 3D lines by constructing an undirected graph.
- A novel Bayesian plane prediction formulation, as the core of our algorithm, which is able to generate candidate planes for the automatic 3D reconstruction of building structure accurately.

2 Related work

We restrict the discussion to the most related works that focus on 3D line cloud reconstruction and surface mesh generation from line segments or primitive shapes. For comprehensive 3D reconstruction approaches, we refer the reader to the survey papers [6], [27].

2.1 Line-based 3D reconstruction

In traditional line-based 3D reconstruction methods, line segments have been used in the procedure of structure from motion by utilizing explicit line segment matching. For example, Bartoli and Sturm [17] first propose a new SFM pipeline from line correspondences across multiple views by introducing an orthonormal representation for 3D lines. Later Schindler *et al.* [16] incorporate the Manhattan-world assumption into 3D line-based reconstruction for urban scenes, which could improve the reconstruction performance. Bay *et al.* [28] use line segments [29] in two images of poorly-textured indoor scenes to address camera self-calibration, bundle adjustment and 3D reconstruction. Zhang and Koch [30] propose another full line-based SfM pipeline, which introduces the Cayley representation of 3D lines and their projections to reconstruct the scene structure and estimate the camera motion. Salaün *et al.* [15] present a robust approach for camera pose estimation without Manhattan-world assumptions by combining both line and point information. Micusik and Wildenauer [18] introduce a SLAM-like line-based SfM system, where they consider unstable endpoints by utilizing relaxed constraints on their positions for line bundle matching and adjustment stages. Although these methods using only line segments for camera pose estimation achieved impressive results for some special scenes, they are rarely used in real-world scenarios because explicit line matching may not put enough constraints on the epipolar geometry for general cases.

More robust approaches in 3D line reconstruction relying on given camera poses have been presented. Researchers argue that accurate camera pose estimation alone is much easier than obtaining dense 3D models, and it also works well around complex and weakly textured objects [31]. In the work of Jain *et al.* [32], a 3D line model is generated by imposing global topological constraints given by connections between neighboring lines. They formulate the reconstruction procedure as an optimization problem and use a sweeping-based approach to avoid explicit 2D line matching between views. However, a disadvantage of this sweeping approach is that it is computationally more expensive than previous methods. To improve the efficiency, Hofer *et al.*propose a series of methods [23], [31] that use weak epipolar-guided line segment matching and limit its potential 3D locations to a discrete set coinciding with these matches. Based on these techniques, they present a robust and publicly available line-based 3D reconstruction tool denoted as Line3D++ [24]. Ramalingam and Brand [33] propose an optimization-based method for reconstructing the 3D arrangement of lines extracted from a single image using Manhattan-world assumption. However, all of above methods only generate a meaningful 3D line cloud rather than surface meshes.

2.2 Surface mesh generation from line segments

Early works focusing on surface reconstruction from line segments usually use 3D lines as constraints in addition to points. Baillard and Zisserman [34] reconstruct piecewise planar models of urban buildings from multiple aerial images. The novelty of this approach is to generate a plane hypothesis from a 3D line with a neighboring point set that is detected in texture images. Zebedin *et al.* [35] present an information fusion strategy that exploits the advantages of several information sources (*e.g.,* height field, building mask and sparse 3D lines) to reconstruct buildings automatically. The 3D line segments are used for obtaining a line-based

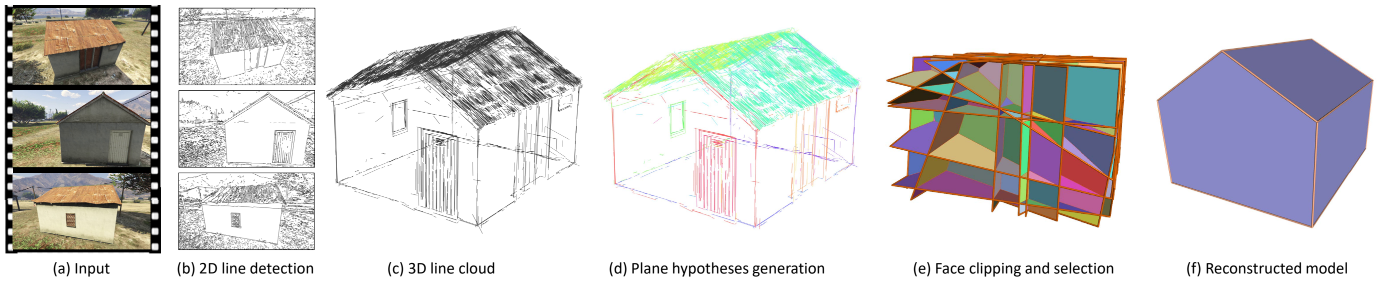


Fig. 2. Overview of the proposed method. Starting from multi-view images (a), we detect 2D line segments in each view by proposing an improved LSD method (b). Then the 2D line segments in neighboring views are matched and triangulated to reconstruct a 3D line could (c). As the core of our system, we generate a set of plane hypotheses from the line cloud by introducing a clustering approach, see (d) where the same group of 3D line segments are coded with the same color. Finally, the candidate faces are clipped (e) and selected to obtain a manifold surface mesh (f).

segmentation, thus helping to generate accurate geometric primitives. Sinha *et al.* [36] compute a set of plane candidates by fitting planes to the sparse point cloud and sparse 3D line segments. Then the piecewise planar depth maps are inferred to recover planar polygonal patches. Similarly, [25] extend the original tetrahedra-carving method to extract 3D surfaces using both 3D points and line cloud. They integrate the 3D line segments with sparse point cloud under a global optimization framework to produce a 3D surface that can preserve sharp edges and flat planes.

Since the *Line Segment Detector* (LSD) [37] was developed, the study of surface mesh generation from pure line segments has gained momentum as the 2D line segments extraction task was fairly well solved. Aiming at applications of robotic mapping and image-based rendering, [38] present a maximally informative surface representation, where they generate plane hypotheses from non-collinear coplanar 3D line segments. Then an efficient plane intersection scheme based on line segment relations is defined to build surface faces that also satisfy the visibility constraint. However, this approach does not scale well to many lines and the output surface mesh is not watertight. Wang *et al.* [39] construct a surface scaffold structure to enhance structural characteristics and suppress irregularities in the building models. They leverage a set of automatic or user-drawn structural scaffold lines to improve the building regularization. Starting from a point cloud, Boulch *et al.* [40] introduce new regularization terms that minimize the length of edges and number of corners in a reconstructed surface, which can be formulated as a sparse mixed-integer linear programming problem. Recently, Langlois *et al.* [41] extend this framework by generalizing data fidelity and visibility from points to 3D line segments. While they could generate watertight piecewise-planar surfaces with impressive quality, this method consumes a lot of memory and the running time is unacceptable for complex shapes due to the cubic complexity in the number of detected planes. In our approach, we formulate a Bayesian probabilistic modeling problem to directly generate candidate planes from 3D line segments, which can be efficiently solved by the *Expectation Maximization* (EM) algorithm.

2.3 Primitive-based scene abstraction/reconstruction

The geometric primitives has been shown to be particularly beneficial for representing and abstracting man-made scenes. Li *et al.* [42] introduce the GlobFit to recover consistently fitted primitives along with their global mutual relations for man-made engineering objects. Limberger and Oliveira [43] propose a real-time plane detection algorithm based on a Hough-transform voting scheme. They use a robust segmentation strategy to identify clusters of approximately coplanar samples, then cast votes for each cluster by using a trivariate Gaussian kernel. To improve the robustness to noise, Araujo and Oliveira [44] further present a new detection method independent of parameter tunning based on a novel planarity test drawn from robust statistics and on a split and merge strategy. Recently, [45] and [46] fit shape primitives from 3D CAD or architectural models to recover structures. Aiming at indoor reconstruction, [47] present a system that automatically recognizes different rooms as separate components. They are capable of coping with heavy occlusions and missing data. Monszpart *et al.* [48] explore the regular arrangements of planes to provide compact and simplified representations of urban scenes. Li *et al.* [49] use a set of well-aligned boxes to approximate the geometry of the buildings. Nan and Wonka [26] generalize this idea to propose the PolyFit that reconstructs lightweight polygonal surfaces. We refer to the survey paper [50] for a more comprehensive discussion. Our work also falls in this type of using a compact set of planes to reconstruct urban buildings. Instead of inputting a 3D point cloud, our technical contribution is to recover better 3D line cloud from multi-view images, and extract candidate planes from the line cloud by a novel clustering algorithm. Then we apply the PolyFit [26] to generate the final polygonal surface mesh.

3 Problem Statement and Overview

Our goal is to generate a complete and compact urban building model from images. Given a set of unordered images $\mathcal{I} = \{I_i\}_{i=1}^{n}$, the output of our method is a lightweight polygonal surface model of buildings, $\mathcal{S} = \{f_j\}_{j=1}^{m}$.

As illustrated in Fig. 2, our algorithm undergoes three main stages: 2D line segment extraction, 3D line cloud reconstruction, and polygonal surface mesh reconstruction. The algorithm starts with extracting 2D line segments in every image. We propose an improved multi-resolution line segment detector (MR-LSD) based on [37]. Specifically, we combine the advantages of line segment detection on low- and high-resolution images, and we also use a clustering

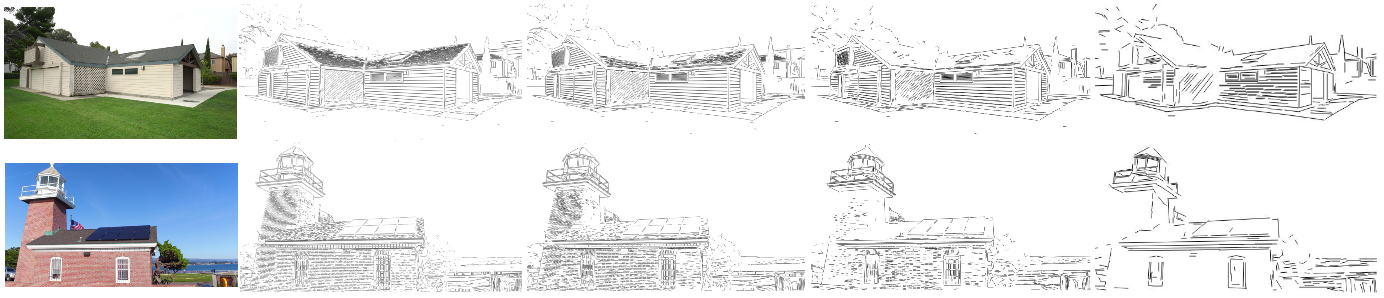


Fig. 3. Illustrating the effect of 2D line segment detection under different image resolutions. The detected line segments on high resolution images are more precise but lack completeness and continuity, while the line segments on low resolution images better capture global structures but lose precision.

approach to merge line segments extracted from multi-resolution images. In this way, we ensure the continuity and completeness of the line segments, thus improving the quality of 2D line segments extraction.

In the second step, we first recover camera parameters from the input image sequence by an arbitrary SFM system (COLMAP [9] or Visual-SFM [8] are used in our experiments). Then a 2D line-to-line matching is performed to establish line segment correspondence using the corresponding camera poses. A new 3D line cloud reconstruction method is also presented to obtain clean and complete 3D lines, where we introduce an efficient clustering and merging operator to filter out mismatched and redundant 3D line hypotheses.

Next, to build a polygonal surface model, we generate a set of candidate planes from 3D line segments, where we formulate the plane hypotheses generation as a Bayesian probabilistic modeling problem. Finally, we utilize the method of PolyFit [26] to make selections from plane candidates to construct the polygonal surface model.

4 Methodology

4.1 Multi-resolution Line Segment Detection

To generate a triangulated 3D line cloud, we require to first extract a set of 2D line segments in each image. Line segment detection itself is still a challenging problem in computer vision due to its simple structure but uncertain length and width in discrete pixels. The LSD algorithm [37] has been widely used to detect 2D line segments. LSD is quite robust, efficient, and produces accurate results without parameter tuning. However, this algorithm computes the image gradient quite locally, thus failing to reconstruct the global structure. Especially in the case of uneven illumination or occlusion, the detected line segments are incomplete and discontinuous. Thus, they cannot effectively meet the needs of stereo line matching and 3D surface reconstruction. By contrast, the complete and continuous line segments can ensure the effect of stereo matching and reduce the number of false matching.

MR-LSD. To extract more complete line segments, we propose a multi-resolution line segment detector (MR-LSD) by extending the original LSD to focus on the robust detection of general line segments. Our approach exploits the observation that the quality of extracted line segment features heavily depends on the image resolution. Generally speaking, the detection results under high image resolution are more precise but lack completeness and continuity because of high frequency noise and textures. In contrast, the detected line segments on low resolution images will lose precision but can better capture global structures.

We introduce the MR-LSD to possess the advantages of line segment detection on both low- and high-resolution images. We observe that the detection error will be larger if the resolution is too low (*e.g.*, the sub-sampling factor is smaller than 0.5), and the computational complexity will be increased greatly if the resolution is too high while the detection quality would not increase. Given the input image, it will be appropriate to set three different scales of the resolution, ranging from half to twice the raw resolution. In most of our used examples, we specify the sampling factor as 0.5, 1.0, 2.0, and we do not need to tune them too much. In our implementation, we generate the high-resolution images by using an upsampling method which is interpolated by the cubic spline interpolation. Low resolution images are obtained by performing Gaussian filtering and downsampling on the original images. Then for an image with a specific resolution, we apply the LSD algorithm to extract line segments reliably. Each detected line segment $L_i^m = \mathbf{s}_i\mathbf{e}_i$ in image I_m is represented as a vector that consists of two endpoints $\mathbf{s}_i, \mathbf{e}_i \in \mathbb{R}^2$. Fig. 3 shows the effect of line segment detection under different image resolutions (the scaling factor is s). As is shown, in some instances, we cannot obtain line segments from the original image but they can be well extracted from the upsampled image. Meanwhile, the completeness of the line segment increases as the resolution decreases, but the accuracy also decreases.

Clustering line segments. After detecting a set of 2D line segments in images with different resolutions, we propose an efficient clustering method to combine the corresponding line segments, aiming at generating continuous and complete line segments. The clustering algorithm groups similar line segments together into a cluster. As such, we can find a representative line segment for each cluster. To do so, we first need to define the distance function to measure the similarity between line segments. We apply similarity measures that are adapted from the line segment Hausdorff distance used in the area of pattern recognition [51].

Our distance function contains three components: angle distance (d_θ), vertical distance (d_v) and parallel distance (d_p). Given two line segments $L_i = \mathbf{s}_i\mathbf{e}_i$ and $L_j = \mathbf{s}_j\mathbf{e}_j$, we

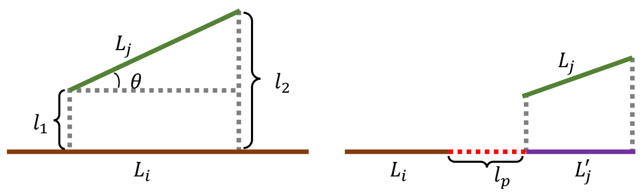


Fig. 4. Definition of distance functions for 2D line segments clustering. The left shows the angle distance and vertical distance, while the right shows the parallel distance.

assume that the line segment L_i is longer than L_j without loss of generality, as illustrated in Fig. 4. The angle distance, d_θ, between two lines is borrowed from [51] and defined as:

$$d_\theta = \|L_j\| sin\theta \,, \tag{1}$$

where θ is the angle between L_i and L_j. $||L_j||$ is the length of L_j, which transforms the angular difference into distance. As a result, the value of angle distance can be directly compared with the vertical distance and parallel distance. Note our angle distance is designed for line segments without directions, in other words, $\mathbf{e}_i\mathbf{s}_i$ represents exactly the same line segment as $\mathbf{s}_i\mathbf{e}_i$.

To compute the vertical distance d_v, we denote l_1 and l_2 as the projection distance of the endpoints $\mathbf{s}_j$ and $\mathbf{e}_j$ onto the underlying line of L_i, respectively. Then the vertical distance can be defined by the Lehmer mean [52] with the order of filter 2:

$$d_v = \frac{l_1^2 + l_2^2}{l_1 + l_2} \,. \tag{2}$$

The parallel distance d_p can be understood as the interval distance between two collinear line segments, while the collinearity may be not a general situation (see Fig. 4). We compute the projected line segments L_j' of L_j onto the underlying line of L_i. Naturally, the length l_p should be the Euclidean distance between two closest end points of L_j' and L_i. While L_i and L_j' have an intersection part, d_p should be zero. Therefore, d_p is defined as:

$$d_p = \begin{cases} 0, & \text{if } L_i \text{ intersects with } L_j' \\ l_p, & \text{else} \end{cases} \,. \tag{3}$$

Finally, the distance function between two line segments for clustering is defined as the weighted sum of these three items:

$$d(L_i, L_j) = \alpha d_\theta + \beta d_v + \gamma d_p \,, \tag{4}$$

which has two constraints:

$$\begin{aligned} 0 < \alpha, \beta, \gamma < 1 \,, \\ \alpha + \beta + \gamma = 1 \,. \end{aligned} \tag{5}$$

After defining the distance function, we next group the line segments into different clusters. As we have no prior knowledge about the scene, the appropriate number of clusters is difficult to determine. Moreover, the extracted line segments are usually noisy, incomplete, and with outliers. Thus we should not only consider the distance function between line segments, but also consider their distributions especially the transitivity between line segments.

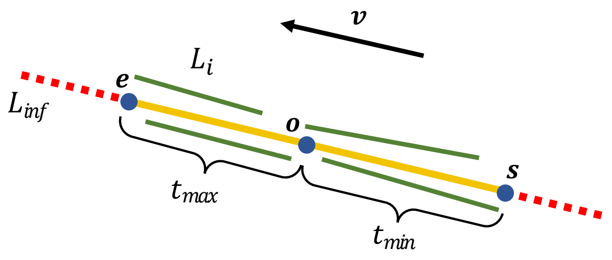
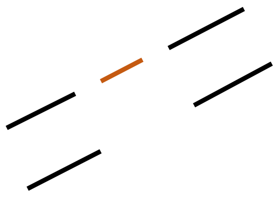


Fig. 5. Illustrating the process of merging 2D line segments. The green line segments represent the inputs, while the yellow one is the final merged result.



As shown in the inset figure, if a red line segment runs between the two black line segments, the three line segments very likely belong to the same line segment because the broken line segments may be caused by image noise; otherwise, the probability that the two black line segments belong to the same cluster is low. In our implementation, we find that the *density-based spatial clustering of applications with noise* (DBSCAN) [53] is suitable for solving our problem. The DBSCAN algorithm is robust to noise and does not require specifying the number of clusters. In addition, such a density-based clustering approach is useful to find distinctive patterns and identify the transitivity between line segments. DBSCAN contains two main parameters: the density-reachable radius ϵ and the local density threshold $minPts$. As the line segments are usually noisy and contain many duplicates, we set $\epsilon = 0.6$ and and a relatively large $minPts = 5$ in default. This default value generally works well in our used data sets.

Merging line segments. The final step of our line segment detector is to merge the items belonging to the same cluster to determine a representative line segment for each cluster. Given a cluster containing segments $C = \{L_1, L_2, \cdots, L_n\}$ to be merged, we first compute their geometric center $\mathbf{o}$ and average direction $\mathbf{v}$. When calculating the average direction, we need to keep the orientation of all line segments consistent, *i.e.*, the inner product of two arbitrary line segments L_i and L_j should be non-negative. As illustrated in Fig. 5, the geometric center $\mathbf{o}$ and average direction $\mathbf{v}$ jointly determine the line L_{inf} where the merged line segment should be located. Then we project the endpoints of each line segment in set C onto l_{inf}, thus the projection points will distribute on both sides of point $\mathbf{o}$ along $\mathbf{v}$. We identify the two farthest projected endpoints in the opposite direction of $\mathbf{o}$ as point $\mathbf{s}$ and $\mathbf{e}$, namely the starting point and the endpoint of the line segment after merging. The yellow line segment L in Fig. 5 is the final merged line segment. More details on this merging step are given in Algorithm 1.

4.2 3D Line Cloud Generation

As mentioned above, we have extracted a set of more continuous and complete 2D line segments. Our next goal is to generate a corresponding set of candidate 3D line segments by using the associated camera poses. This process contains two steps: establishing correspondences between matched

Algorithm 1 Merging 2D line segments in one cluster

Input: A line segments cluster $C = \{L_1, L_2, \cdots, L_n\}$ where $L_i = \mathbf{s}_i\mathbf{e}_i$

Output: A merged line $L = \mathbf{se}$

```
1: $\mathbf{o} \leftarrow \mathbf{s}_1 + \mathbf{e}_1$
2: $\boldsymbol{v} \leftarrow \overrightarrow{\mathbf{s}_1\mathbf{e}_1}$
3: for $i \leftarrow 2, n$ do
4:     $\mathbf{o} \leftarrow \mathbf{o} + \mathbf{s}_1 + \mathbf{e}_1$
5:     if $\boldsymbol{v} \cdot \overrightarrow{\mathbf{s}_i\mathbf{e}_i} < 0$ then
6:         $\boldsymbol{v} \leftarrow \boldsymbol{v} - \overrightarrow{\mathbf{s}_i\mathbf{e}_i}$
7:     else
8:         $\boldsymbol{v} \leftarrow \boldsymbol{v} + \overrightarrow{\mathbf{s}_i\mathbf{e}_i}$
9:     end if
10: end for
11: $\mathbf{o} \leftarrow \mathbf{o}/2n$
12: $\boldsymbol{v} \leftarrow \boldsymbol{v}/\|\boldsymbol{v}\|$
13: $t_{max} \leftarrow -\infty$
14: $t_{min} \leftarrow +\infty$
15: for $i \leftarrow 1, n$ do
16:     $t_{\mathbf{s}} \leftarrow \overrightarrow{\mathbf{os}_i} \cdot \boldsymbol{v}$
17:     $t_{\mathbf{e}} \leftarrow \overrightarrow{\mathbf{oe}_i} \cdot \boldsymbol{v}$
18:     $t_{max} \leftarrow \max(t_{max}, t_{\mathbf{s}}, t_{\mathbf{e}})$
19:     $t_{min} \leftarrow \min(t_{min}, t_{\mathbf{s}}, t_{\mathbf{e}})$
20: end for
21: $s \leftarrow \mathbf{o} + t_{min}\boldsymbol{v}$
22: $e \leftarrow \mathbf{o} + t_{max}\boldsymbol{v}$
23: $L \leftarrow \mathbf{se}$
24: return L
```

2D line segments, and then generating and grouping 3D line segments to obtain a 3D line cloud.

3D segment hypotheses generation by 2D line-to-line matching. 2D line-to-line matching is performed on pairs of images. However, to reconstruct an urban scene, hundreds of images are usually required, even much more for a complex scene. Matching among all images is unnecessary and infeasible, because most pairs of images have no common parts between them. For efficient matching, we select a set of neighboring views that should satisfy the following three conditions:

- The neighboring views should share a large field of view overlap, which can be evaluated by counting the number of matched SIFT feature points.
- Appropriate distance between the cameras corresponding to the images is essential. The reason is that two images taken from the same viewpoint will never rebuild the 3D line segments even if they have a lot of overlaps.
- Each image can only be matched with a certain number of neighboring images, which can reduce the number of unnecessary matches caused by too densely captured images when the camera positions may not be evenly distributed.

For each pair of selected neighboring images, we then match all line segments in the reference image I_r to all line segments in the target image I_t. To accomplish this goal, we establish the correspondences by satisfying the epipolar constraint. Specifically, for each line segment $L_i = \mathbf{s}_i\mathbf{e}_i$ in I_r, we compute the corresponding epipolar lines, $e^i_{\mathbf{s}}$ and $e^i_{\mathbf{e}}$, in I_t for the two endpoints of L_i. According to geometric

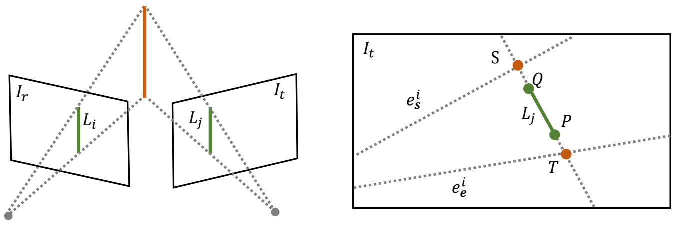


Fig. 6. An example for the epipolar-based matching procedure. Line Segments Matching. The left subfigure shows two-view geometry of two lines in respectively two matched image for match. The right subfigure shows the projection of polar line of l_1 in I_2.

relationship, if line segment $L_j \in I_t$ is matched to L_i, then the endpoints of L_j should be located on $e^i_{\mathbf{s}}$ and $e^i_{\mathbf{e}}$, respectively. For an example in Fig. 6, I_r and I_t are two neighboring images, while L_i and L_j are two line segments located in the corresponding images. $e^i_{\mathbf{s}}$ and $e^i_{\mathbf{e}}$ are two epipolar lines of L_i projected in I_t. P and Q are two end points of L_j, S and T are the intersection points of $e^i_{\mathbf{s}}$ and $e^i_{\mathbf{e}}$, and the straight line where L_j is located. The directional matching score between L_i and L_j can be evaluated in terms of *Intersection over Union* (IOU):

$$s(L_i, L_j) = 1 - \frac{\|PT\| + \|QS\|}{\|ST\|}. \tag{6}$$

We also compute $s(L_j, L_i)$ in the same way, and define the final matching score between L_i and L_j as the average value of $s(L_i, L_j)$ and $s(L_j, L_i)$. L_i and L_j are potentially matched if the final matching score is bigger than a threshold α ($\alpha = 0.35$ in default).

After 2D matching, we obtain pairwise matched lines, our next goal is to find the set of 2D line segments that correspond to one 3D line. To this end, we use an undirected graph to organize the matching relationships, in which a node represents a line segment, and an edge between two nodes indicates that the line segments are potentially matched. By utilizing the graph structure, we can extract a set of connected components from the graph. Ideally, if all of the matches are correct, we can reconstruct an exact 3D line segment for each connected component. In practice, some false matches always occur due to imprecise line detection or occlusions. Finding an optimal graph partitioning solution is NP-complete. Although some standard graph partitioning algorithms (*e.g.,* spectral partitioning, multilevel methods) can be used, an appropriate clustering evaluation metric should be found. Although the matching score defined on the graph edges can be used, it may lead to inaccurate partitioning results due to the false matches. Indeed, our goal is to minimize the re-projection error of a generated 3D line segment with respect to each view. Thus the 3D line segments should be taken into account. To solve this issue, we apply a greedy algorithm based on the breadth-first search. We first find the two nodes that are the best match. Then we generate an initial and accurate 3D line segment by triangulating the two corresponding endpoint pairs from the corresponding images. Then starting with these two nodes, we visit other nodes in the connected component in a breadth-first-search manner. For each visited node representing a line segment L_i in view I_m, we project the initial 3D line segment to I_m to obtain a projected 2D line segment L_p. If the distance between L_i and L_p is small, we think L_i is a correct match and use it to optimize the initial 3D line segment. Otherwise, the node representing L_i is deleted from the current connected component. Finally, after visiting all of the nodes, an initially connected component in the graph may be partitioned into several components, from each of which we can reconstruct a 3D line segment.

3D line grouping. To accelerate the matching process in the above stage, instead of performing pairwise line matching among all images, we only match line segments in neighboring views. Such local matching would generate redundant 3D line segments. For instance, the same 3D line may be generated in different sets of neighboring views. Therefore, we apply a 3D line clustering algorithm to remove the redundant 3D line segments and the possible outliers. Given that the number of reconstructed 3D line segments in a building scene is very large (usually hundreds of thousands), directly performing clustering on all 3D line segments will lead to a huge time cost because each pair of line segments should be compared. However, we find that many pairs of line segments are impossible to be grouped together, *e.g.*, the line segments that are very far apart.

To efficiently group those 3D line segments, we utilize the *Bounding Volume Hierarchy* (BVH) [54] tree to accelerate the clustering process. Specifically, we propose a two-step clustering approach where the position and direction of line segments are considered separately. Firstly, each 3D line segment is regarded as a point in the 3D space, and the distance between two 3D line segments is measured by the Euclidean distance between their midpoints. We construct the BVH tree by wrapping the 3D line segments in bounding volumes based on a top-down partitioning approach. We then perform DBSCAN algorithm (with parameters $\epsilon = 0.016, minPts = 4$), in which we can efficiently traverse the nearest 3D line segments in the BVH tree. In the second step, we further cluster the segments based on their directions. We convert the direction vector of each line segment into spherical coordinates, thus the direction can be regarded as a point on a sphere. Then, we parameterize the spherical surface into a rectangle. Next, we construct a BVH tree again and conduct the DBSCAN clustering procedure (with parameters $\epsilon = \pi/36, minPts = 4$) with BVH traversal acceleration. After clustering, we merge the 3D line segments in each cluster by using the merging operator described in Sec. 4.1 to obtain the final 3D line model.

4.3 Surface mesh reconstruction

Once we have constructed a 3D line cloud $\mathcal{L} = \{l_i | i = 1, 2, \cdots, n\}$, where each line segment contains two endpoints $\mathbf{p}_1(x_1, y_1, z_1)$ and $\mathbf{p}_2(x_2, y_2, z_2)$ in 3D space, our next goal is to obtain a compact and manifold surface model. To accomplish this goal, we first generate plane hypotheses from 3D line segments, then make a face selection from plane candidates to construct the final polygonal mesh.

Plane hypotheses generation. In this step, we formulate the candidate planes generation as a Bayesian probabilistic modeling problem, where we aim to minimize the distance from all 3D line segments to the corresponding candidate

Algorithm 2 Estimating parameters of candidate planes

```
Input: A 3D line cloud $\mathcal{L} = \{l_i | i = 1, \cdots, n\}$
Output: Probabilistic model's parameters: $\{\mathrm{P}(f_j), \mathbf{v}_j, \mathbf{n}_j, \sigma_j\}$
1: Set initial values for $\mathrm{P}(f_j), \mathbf{v}_j, \mathbf{n}_j, \sigma_j$, and set the maximum iteration number $N$
2: while $N > 0$ do
3:     E step: calculate the plane response:
           $\hat{\gamma}_{ij} = \frac{\mathrm{P}(f_j)\mathrm{P}(l_i|f_j)}{\sum_{j=1}^{k} \mathrm{P}(f_j)\mathrm{P}(l_i|f_j)}$
4:     M step: update the probabilistic model's parameters
       (1) compute the priori probability:
           $\hat{\mathrm{P}}(f_j) = \frac{\sum_{i=1}^{n} \hat{\gamma}_{ij}}{n}$
       (2) update the standard deviation:
           $\hat{\sigma}_j^2 = \frac{\sum_{i=1}^{n} \hat{\gamma}_{ij}\{[(\mathbf{p}_{i1} - \mathbf{v}_j) \cdot \mathbf{n}_j]^2 + [(\mathbf{p}_{i2} - \mathbf{v}_j) \cdot \mathbf{n}_j]^2\}}{\sum_{i=1}^{n} \hat{\gamma}_{ij}}$
       (3) update the plane parameters:
           $\hat{\mathbf{v}}_j = \frac{\sum_{i=1}^{n} \hat{\gamma}_{ij}(\mathbf{p}_{i1} + \mathbf{p}_{i2})}{2\sum_{i=1}^{n} \hat{\gamma}_{ij}},$
           $A\hat{\mathbf{n}}_j = \lambda_{min}\hat{\mathbf{n}}_j,$
       where the matrix is computed as:
           $A = \sum_{i=1}^{n} \hat{\gamma}_{ij}[(\mathbf{p}_{i1} - \mathbf{v}_j)(\mathbf{p}_{i1} - \mathbf{v}_j)^T + (\mathbf{p}_{i2} - \mathbf{v}_j)(\mathbf{p}_{i2} - \mathbf{v}_j)^T].$
5:     $N = N - 1$
6: end while
```

planes. We suppose the input 3D lines are sampled from k different planes (k can be a reasonably large number) and each plane is represented by a vertex $\mathbf{v} = (v_x, v_y, v_z)$ and a normal vector $\mathbf{n} = (n_x, n_y, n_z)$ with $n_x^2 + n_y^2 + n_z^2 = 1$. To determine the initial planes, we compute the bounding box of the 3D line cloud and denote its diagonal length as d_{scene}. Then we uniformly sample m nodes with an interval distance of τ, thus having $m = d_{scene}/\tau + 1$ (we set $\tau = 0.02d_{scene}$ in default). At the position of each node, we generate three orthogonal planes along the axes, and we can obtain a set of dense planes with $k = 3m$. Next, we iteratively perform two steps until reaching the convergence criteria: (1) assigning each 3D line segment to its nearest plane; (2) updating the plane parameters ($\mathbf{v}$ and $\mathbf{n}$) according to the assignment.

In detail, we assume that the probability of a 3D line segment $l_i = (\mathbf{p}_{i1}, \mathbf{p}_{i2})$ belonging to a plane f_j is represented as a Gaussian distribution:

$$\mathrm{P}(l_i|f_j) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left\{-\frac{[(\mathbf{p}_{i1} - \mathbf{v_j}) \cdot \mathbf{n_j}]^2 + [(\mathbf{p}_{i2} - \mathbf{v_j}) \cdot \mathbf{n_j}]^2}{2\sigma_j^2}\right\}. \tag{7}$$

According to the Bayes' theorem, if we have the priori probability $\mathrm{P}(f)$ of the planes, the joint probability distribution $\mathrm{P}(l_i, f_j)$ can be written as:

$$\mathrm{P}(l_i, f_j) = \mathrm{P}(f_j)\mathrm{P}(l_i|f_j), \tag{8}$$

where $\sum_{i=1}^{k} \mathrm{P}(f_j) = 1$, and initially we set $\mathrm{P}(f_j) = 1/k$. We then obtain the marginal probability of observing a line segment l_i:

$$\mathrm{P}(l_i) = \sum_{j=1}^{k} \mathrm{P}(l_i \,|f_j)\mathrm{P}(f_j)\,. \tag{9}$$

Therefore, given an observed 3D line cloud $\mathcal{L}$ in which each line segment l_i is independent, we can maximize a likelihood function $\mathrm{P}(\mathcal{L}|\theta)$ to obtain the parameters of each plane f_j:

$$\max \prod_{i=1}^{n} \prod_{j=1}^{k} [P(l_i|f_j)P(f_j)]^{\gamma_{ij}}\,, \tag{10}$$

where we use θ to denote all parameters in the parameterized model. γ_{ij} is the hidden variable indicating the plane to which the line segments belong:

$$\gamma_{ij} = \begin{cases} 1, & l_i \in f_j, \\ 0, & otherwise \end{cases} \tag{11}$$

For each l_i, the hidden variable is γ_{ij} unknown. Hence, we leverage the EM algorithm to solve our optimization problem iteratively. The algorithm is terminated when the maximal iteration number N is reached or the priori probability $\mathrm{P}(f)$ does not change between two iterations, *i.e.*, $|\mathrm{P}^i(f) - \mathrm{P}^{i-1}(f)| < \delta$, where we set $N = 180, \delta = 10^{-5}$ by default. We implement the algorithm by using matrix operations to further speed up the computation. The numerical algorithm for estimating plane parameters is given in Algorithm 2, and the detailed formula derivation and calculation process can be found in the supplementary materials.

Surface mesh reconstruction. After acquiring the parameters of all candidate planes, we make face selections for constructing our surface models. We modify the PolyFit [26] framework to directly process our extracted planar primitives. It solves the integer optimization problem to seek an optimal combination of them to obtain the final manifold polygonal surface meshes.

4.4 Computational complexity

Our approach includes three main stages: multi-resolution 2D line segment detection (MR-LSD), 3D line cloud generation, and surface mesh reconstruction. The MR-LSD is built on the original linear-time LSD algorithm [37], thus its time complexity is $O(sP)$, where $s = 3$ is the number of scales for multi-resolution, and P is the number of pixels in the image. To cluster 2D line segments, we apply the DBSCAN whose computational time is $O(mlogm)$, where m is the number of detected line segments in an image.

In the 3D line cloud generation phase, we first perform a 2D line-to-line matching between neighboring images to extract a set of potential 3D lines. In the worst case scenario, this step costs $O(vm^2)$ with v is the number of neighboring views. Then a graph is constructed to refine potential 3D lines, where we extract the connected components in a breadth-first-search manner, which runs in $O(vm + e)$ (e is the number of edges in the graph). Finally, 3D line grouping is based on two iterations of DBSCAN with the complexity of $O(nlogn)$, where n is the number of generated 3D line segments.

TABLE 1
Quantitative comparison of 2D line segments detectors. $|L|$ is the number of detected segments, l_{avg} is average length of all line segments in one image. The best result of each measurement is marked in **bold** font.

Scene	Image No.	CannyLine [55]		LSD [37]		Ours	
		$\|L\|$	l_{avg}	$\|L\|$	l_{avg}	$\|L\|$	l_{avg}
Indoor	1	641	41.40	1165	30.20	**350**	**54.59**
	2	393	50.62	721	32.89	**284**	**55.12**
	3	442	39.93	630	29.15	**277**	**48.65**
	4	201	71.21	263	48.33	**102**	**76.34**
	5	486	37.11	1047	22.36	**396**	**38.45**
	6	297	77.92	617	44.12	**151**	**84.07**
	7	399	54.81	709	34.64	**247**	**62.73**
	8	527	34.27	1005	20.95	**372**	**38.77**
	9	456	45.74	1086	25.61	**351**	**47.26**
	10	229	56.38	361	37.11	**102**	**74.73**
Outdoor	1	608	32.24	1544	22.37	**332**	**36.86**
	2	456	32.00	806	20.64	**317**	**34.88**
	3	252	34.68	427	23.37	**151**	**43.83**
	4	316	37.16	972	18.56	**301**	**39.67**
	5	301	40.22	512	27.87	**201**	**43.11**
	6	405	34.90	907	18.49	**346**	**37.99**
	7	629	41.65	1243	26.80	**346**	**52.39**
	8	643	38.91	1393	25.41	**407**	**42.84**
	9	402	41.41	691	26.39	**296**	**42.02**
	10	458	35.52	876	20.92	**321**	**41.46**

For surface reconstruction, 3D lines are used to predict 3D planes in a Bayesian probabilistic modelling formulation solved under standard expectation maximization (EM). The computational time of this step takes $O(ikn)$ where n is the number of input 3D lines, k is the number of assumed candidate planes, and i is the number of EM iterations. We set the maximum iteration number to 100, but we find that our algorithm usually only needs a few dozen iterations.

5 Experimental Results

In this section, we demonstrate the effectiveness of our approach in several scenes with different styles and shapes. We also evaluate the proposed algorithm qualitatively and quantitatively through the visual inspection of our results and a comparison with previous point-based and line-based reconstruction methods. Our algorithm is implemented in C++ and Python. All results presented in this paper are obtained on a desktop computer equipped with an Intel i7-7700k processor with 4.2 GHz and 16 GB RAM.

5.1 Comparison on 2D line detection and 3D line cloud generation

Compared with other 2D line segment extraction methods, the main feature of our method is a line detector under multi-resolution images, and a line segment clustering and merging operation, both of which improve the integrity and continuity of the detected line segments. Here we compare two popular robust detectors, CannyLines [55] and LSD [37], in terms of the quality of extracted line segments for scene abstraction and reconstruction. We employ the *YorkUrbanDB* dataset [59] which consists of 102 images that are captured from real-world urban scenes. Each image in the database has been hand-labeled to identify the set of major line segments, which can be regarded as ground-truth lines. We first select 20 images from the database for testing,

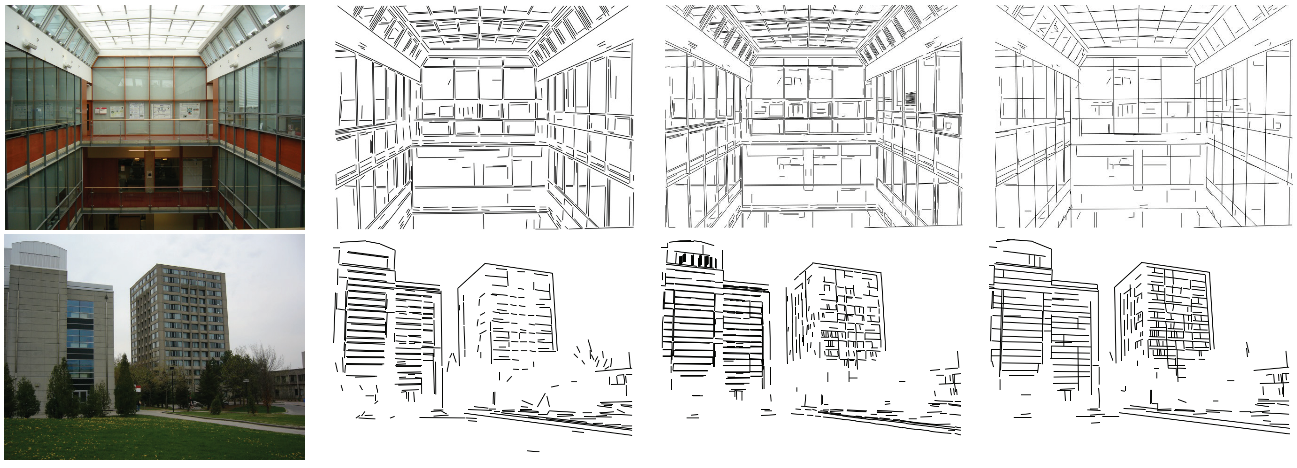


Fig. 7. Visual comparison of different line segment extraction methods using indoor and outdoor scenes.

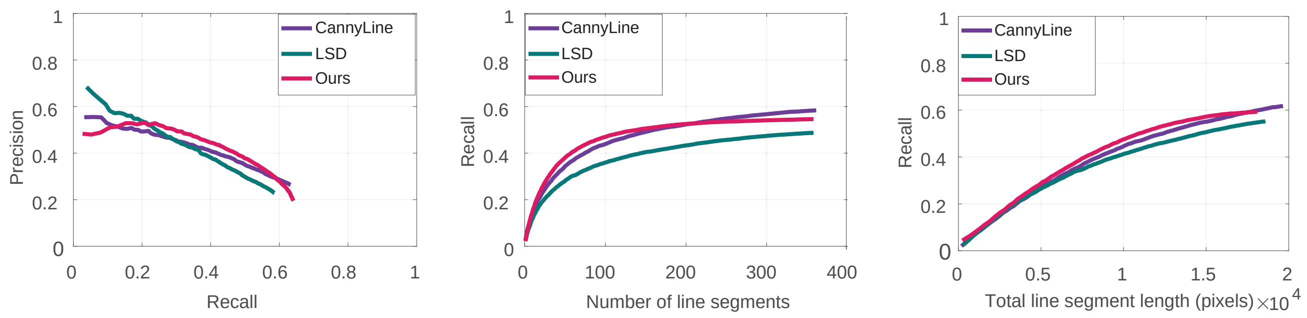


Fig. 8. Performance of the proposed MR-LSD compared with previous 2D line segment detectors. From left to right are precision-recall, recall as a function of the number of segments returned, and recall as a function of the total length of segments returned.

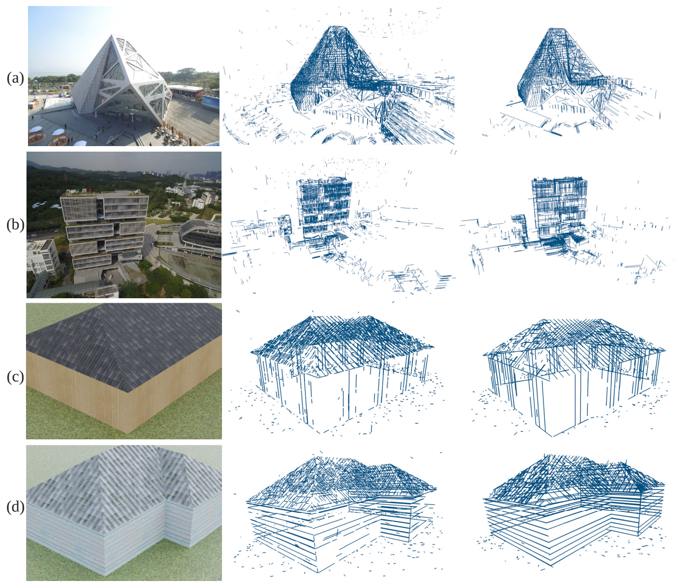


Fig. 9. Comparing our approach (right column) to Line3D++ [56] (middle column) on 3D scene abstraction and wireframe model reconstruction.

of which 10 are indoor scenes and 10 are outdoor scenes. Fig. 7 shows the line segment extraction results of two representative examples. Through the visual comparison, we can observe that our results are much cleaner. Moreover, the completeness and continuity of the line segments extracted by our method are the best because we have the least number of broken line segments. For quantitative evaluation, Table 1 reports the number of detected line segments and also calculates the average length of all line segments. To measure the distance between the ground-truth and detected line segments further, we adopt three evaluative measures proposed by [60], including precision-recall, recall as a function of the number of segments, and recall as a function of total segment length. From Table 1 and Fig. 8, it can be seen that we use fewer line segments to represent the scenes. The average length of our line segments is much longer than the comparison methods while not reducing detection accuracy. It indicates that in general our proposed approach extracts more meaningful line segments.

Next, we evaluate the results of 3D line cloud generation. We select Line3D++ [56] as a competitor since to the best of our knowledge it is still the state-of-the-art method, and it has been widely used in several line-based 3D scene abstraction and reconstruction approaches. Fig. 9 gives four visual comparison examples by showing the reconstructed 3D wire-frame models. The first two examples are our captured real-world scenes, and the last two synthetic scenes are from [61], which introduces a synthetic dataset by rendering multi-view images of the buildings in Blender with synthetic textures. Based on the synthetic data, we can also conduct a quantitative comparison. Given that a

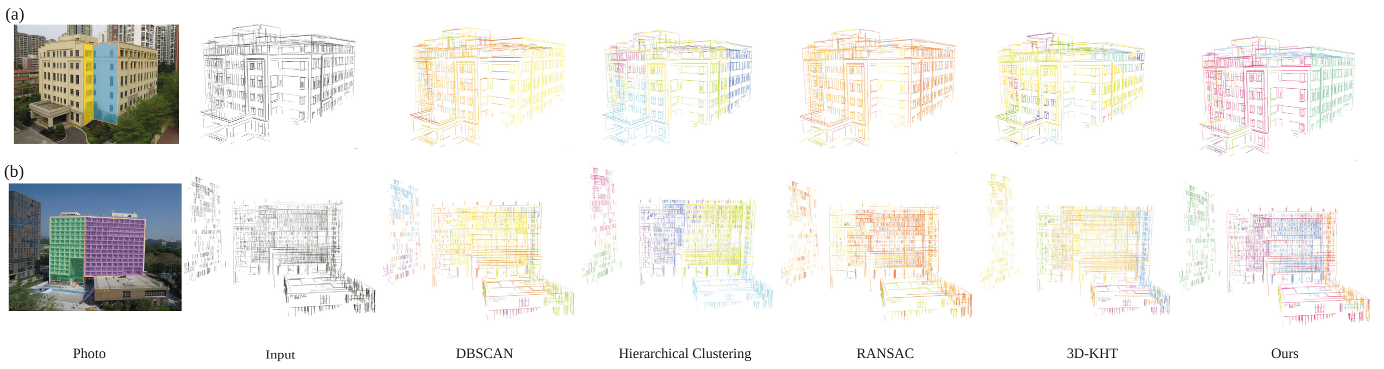


Fig. 10. Visual comparison of different clustering methods using real-world data. From left to right: the reference photo, input 3D line clouds, the clustering results of DBSCAN [53], hierarchical clustering [57], RANSAC [58], 3D-KHT [43] and ours. A full comparison to all of previous methods can be found in the supplementary materials.

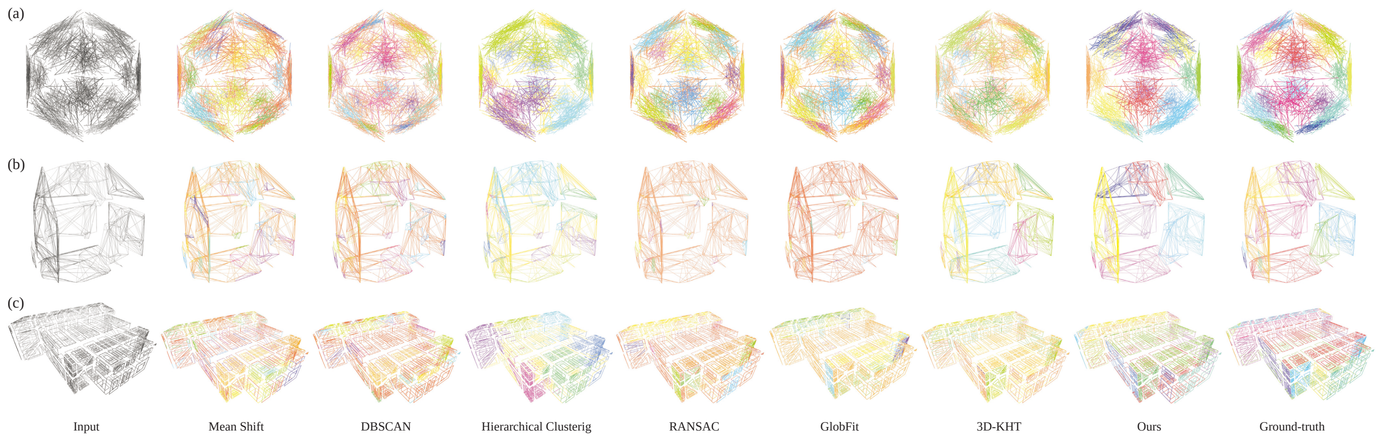


Fig. 11. Visual comparison of different clustering methods using synthetic data. The ground-truth labeling is provided in the last column. Please zoom in to compare the clustering details.

TABLE 2
Quantitative comparison to Line3D++ using the synthetic data shown in Fig. 9. E_{mean} and E_{RMS} represent mean and root mean square (RMS) of the reconstruction error measuring the per-point distance between the reconstructed 3D line segments and the ground-truth mesh.

Scene	Line3D++		Ours	
	E_{mean}	E_{RMS}	E_{mean}	E_{RMS}
Fig. 9 (c)	**3.74**	17.18	5.03	**14.36**
Fig. 9 (d)	3.01	12.14	**2.23**	**8.79**

set of ground-truth 3D lines are very difficult to synthesize, we instead compute the reconstruction error which measures the distance between the generated 3D lines of our approach (or Line3D++) and the ground-truth surface mesh. As shown in Fig. 9, the 3D line clouds generated by Line3D++ are quite noisy. In contrast, the proposed method outperforms Line3D++ in terms of noise control, line segment consistency and completeness, and visual effects. In addition, Table 2 shows that we still have comparable reconstruction fidelity with Line3D++. Therefore, our approach is more suitable for plane detection and surface mesh reconstruction.

5.2 Comparison on plane hypotheses generation

One of the critical steps for the successful reconstruction of our algorithm is the Bayesian plane prediction formulation (see Sec. 4.3) which has two functions: (1) finding the group of 3D line segments belonging to the same plane, and (2) returning the plane parameters. To demonstrate its efficiency, we compare our approach against two kinds of methods: direct clustering methods (*e.g.*, mean shift [62], DBSCAN [53] and hierarchical clustering [57]) and plane detection methods (*e.g.*, RANSAC [58], GlobFit [42] and 3D-KHT [43]). For the clustering methods, we feed the same 3D line segments to all clustering methods to judge the performance of the clustering. For plane detection methods, we input the endpoints of 3D line segments to them, and evaluate their clustering performance by converting their plane detection results into the clustering of 3D line segments. It can judge whether they are able to accurately generate candidate planes.

In Fig. 10, we use two real-world data to show the clustering comparison, and the visual results illustrate the superiority of our algorithm. In Fig. 10 (a), the planes indicated by yellow and cyan boxes in the reference photo are two small planes, which are difficult to be accurately detected by other methods, while our approach can correctly recover such small planes. Furthermore, our approach is also capable of distinguishing planes with similar normal

TABLE 3
Quantitative comparison of different clustering methods. *#l* represents the number of input 3D line segments, *#GC* is the ground-truth number of clusters in each example, and *#C* is the final number of clusters obtained by each method.

Input	Methods	#C	*RI*	*NMI*
Fig. 11 (a) *#l=1800* *#GC=20*	Mean Shift	34	0.8150	0.6411
	RANSAC	23	0.9384	0.8056
	DBSCAN	24	0.7940	0.6280
	Hierarchical Clustering	7	0.8694	0.5943
	Globfit	24	0.9418	0.8121
	3D-KHT	17	0.7945	0.4976
	Ours	20	**0.9701**	**0.8920**
Fig. 11 (b) *#l=2428* *#GC=12*	Mean Shift	18	0.7814	0.4282
	RANSAC	23	0.5073	0.3622
	DBSCAN	27	0.7891	0.4749
	Hierarchical Clustering	7	0.7871	0.3688
	Globfit	39	0.5396	0.3786
	3D-KHT	10	0.9299	0.7810
	Ours	12	**1.0000**	**1.0000**
Fig. 11 (c) *#l=4878* *#GC=16*	Mean Shift	45	0.8483	0.4440
	RANSAC	33	0.8719	0.6629
	DBSCAN	41	0.7855	0.3812
	Hierarchical Clustering	7	0.7979	0.3096
	Globfit	13	0.8382	0.6419
	3D-KHT	20	0.9392	0.7893
	Ours	16	**1.0000**	**1.0000**

directions. For example, the green and purple planes labeled in the reference photo in Fig. 10 (b) have a small difference the in normal direction. Compared with previous methods, we can separate these two planes better.

For a quantitative comparison, we randomly generate 3D line segments on three synthetic models from [26], as a result the ground-truth clusters can be provided (see Fig. 11). We adopt two commonly used measures, *Rand Index* (RI) and *Normalized Mutual Information* (NMI), to determine the quality of clustering. RI computes the similarity between two clusterings by considering all pairs of samples and counting pairs assigned to the same or different clusters in the predicted and true clusterings:

$$RI = \frac{TP + TN}{TP + FP + FN + TN}, \tag{12}$$

where TP, TN, FP, FN are the number of true positives, true negatives, false positives, and false negatives, respectively. This index gives a value between zero and one, for $RI = 1$ the two clusterings are identical. The value of NMI depends on the mutual information $I(\cdot\,;\cdot)$ and the entropy $H(\cdot)$ of the labeled classes Ω and clusters C:

$$NMI = \frac{2 \times I(\Omega\,; C)}{H(\Omega) + H(C)}. \tag{13}$$

$NMI = 0$ indicates no mutual information, and 1 means perfect correlation. Fig. 11 shows the qualitative results of different methods, while Table 3 reports the numerical statistics of the clustering performance. Both qualitative and quantitative comparison show that our approach outperforms the other methods in terms of RI and NMI, and we are able to extract candidate planes accurately.

5.3 Comparison on surface reconstruction

Finally, we demonstrate the effectiveness of our proposed pipeline by comparing it against both point-based and line-based surface reconstruction methods. Given the input set of images for each dataset, we follow the typical photogrammetry pipeline using SfM to obtain the camera poses as well as a sparse point cloud. Then we run MVS to generate a dense point cloud. For a thorough evaluation, we compare with previous methods by assembling different compatible configurations. First, we apply the screened Poisson reconstruction [63] on the sparse and dense point clouds respectively to reconstruct corresponding surface meshes. Then the PolyFit method [26] is utilized on the sparse point cloud to obtain a lightweight polygonal surface. We have also implemented another baseline approach, where we sample a point cloud from our reconstructed 3D line cloud, then we detect planes from this point cloud by performing RANSAC, and input the planes to PolyFit to generate the final mesh. Finally, we feed our generated 3D line cloud into SRLS (Langlois *et al.* [41]), which can directly reconstruct the surface from 3D line segments.

Fig. 12 and Fig. 13 display the visual comparison results on several selected real-world and synthetic datasets, respectively. For quantitative comparison, we first use the real-world data to inspect the computation time of each method as well as the face number in the reconstructed mesh, as shown in Table 4. We then use synthetic data to compute the reconstruction error between the mesh generated by each method and the ground-truth surface. Table 5 reports corresponding quantitative comparison using Hausdorff distance. As shown in Figures 12 and 13, due to missing data, the 3D surfaces reconstructed from sparse point clouds lack structural details and contain holes and inaccuracies. Although Poisson reconstruction on a dense point cloud obtains models with fine shapes, the mesh of a single building has hundreds of thousands or even millions of faces (see Tables 4 and 5) which will lead to a large storage burden for large-scale urban reconstruction. Moreover, MVS+Poisson reconstruction often represents planar regions with a large number of bumps, as noise exists in plane normal directions due to the inference error in depth computation. Compared with purely point-based methods, PolyFit is able to reconstruct piecewise planar models. However, this method fails to reconstruct faithful models due to missing data and unreliable plane detection from a point cloud, because it heavily relies on plane detection results. SRLS [41] detects planes from a line cloud with visibility information, then reconstructs a surface mesh by labeling each 3D cell of the plane arrangement as full or empty. Therefore, although this method can capture more building details, it is sensitive to the quality of input 3D line segments, where the noisy or outlier line segments would cause a large number of invalid cells. In addition, Table 4 shows that this method is very time consuming (usually takes over 30 minutes) because of a cubic complexity in the number of planes. By comparison, we could reconstruct a good approximation of the buildings with compact and clean representations. Furthermore, our approach generates structured surface meshes that achieve a good trade-off between the face number and the running time.

5.4 Limitations

While producing convincing results, our framework still has several limitations. First, our algorithm may fail to extract

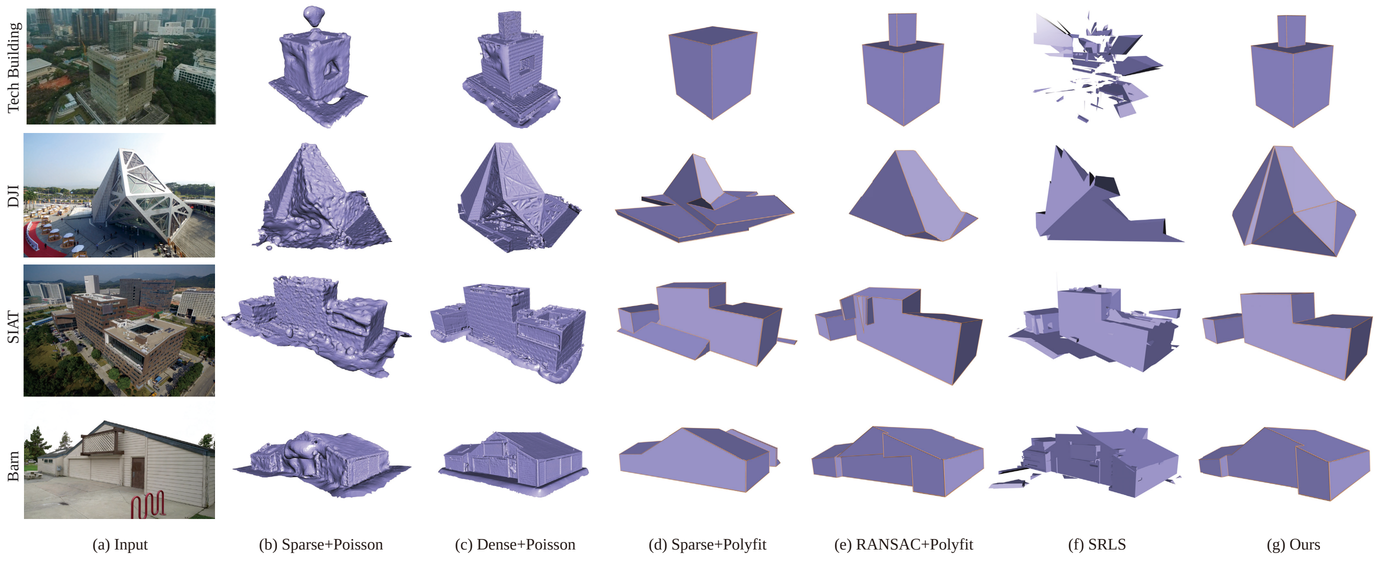


Fig. 12. Comparison to point-based or lightweight reconstruction methods using real-world data. From left to right are: (a) the reference photos, (b) sparse point cloud + Poisson [63], (c) dense point cloud + Poisson [63], (d) sparse point cloud + PolyFit [26], (e) RANSAC [58] + PolyFit [26], (f) SRLS [41], (g) our reconstruction results.

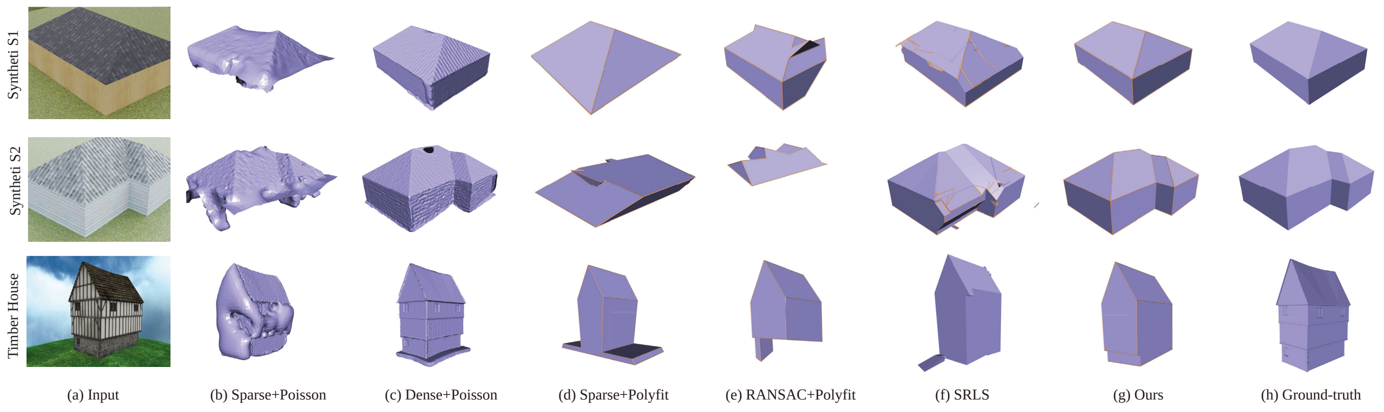


Fig. 13. Comparison to point-based or lightweight reconstruction methods using synthetic data where the ground-truth surface meshes are provided.

TABLE 4
Quantitative comparison to different reconstruction methods using real-world data (Fig. 12) in terms of computation time (t_* in seconds) and the face number ($\#f$) in the output mesh. $\#img$ is the number of input images for each scene. For our approach, t_{3DLine}, t_{Plane} and t_{Mesh} represent the running time of extracting a 3D line cloud, 3D plane detection and surface mesh reconstruction by PolyFit, respectively.

Scene	$\#img$	Sparse+Poisson				Dense+Poisson				Sparse+PolyFit			RANSAC+PolyFit			SRLS			Ours				
		t_{SFM}	$t_{Poiss.}$	t_{total}	$\#f$	t_{MVS}	$t_{Poiss.}$	t_{total}	$\#f$	$t_{Recons.}$	t_{total}	$\#f$	$t_{Recons.}$	t_{total}	$\#f$	$t_{Recons.}$	t_{total}	$\#f$	t_{3DLine}	t_{Plane}	t_{Mesh}	t_{total}	$\#f$
Tech Building	120	317	3.9	320.9	61070	5400	25.8	5742.8	1217139	4.48	321.48	12	8.46	325.46	472	4969	5286	5232	52	14.23	6.89	390.12	490
DJI	218	1539	12.8	1551.8	83457	10582	31.7	12152.7	773706	10.68	1549.68	268	5.19	1544.19	216	9181	10720	4187	98	24.85	39.09	1700.94	389
SIAT	206	1016	8.4	1024.4	186164	9529	27.6	10572.6	650920	13.84	1029.84	452	19.05	1035.05	472	7847	8863	9133	88	13.31	6.39	1123.7	466
Barn	410	902	8.2	910.2	128061	15609	29.5	16540.5	696741	5.55	907.55	104	16.32	918.32	272	12035	12937	39910	70	29.15	32.84	1033.99	417

TABLE 5
Quantitative comparison to different reconstruction methods using synthetic data (Fig. 13) in terms of Hausdorff distance and the face number ($\#f$) in the output mesh. H_{mean} and H_{RMS} represent mean and root mean square (RMS) of the Hausdorff distance, respectively, with respect to the bounding box diagonal of the ground-truth.

Scene	$\#img$	Sparse+Poisson			Dense+Poisson			Sparse+PolyFit			RANSAC+PolyFit			SRLS			Ours		
		$\#f$	H_{mean}	H_{RMS}	$\#f$	H_{mean}	H_{RMS}	$\#f$	H_{mean}	H_{RMS}	$\#f$	H_{mean}	H_{RMS}	$\#f$	H_{mean}	H_{RMS}	$\#f$	H_{mean}	H_{RMS}
Synthetic S1	35	30556	5.3213	8.1652	244867	0.0726	0.1242	46	7.5983	10.3291	108	4.8246	8.5199	6518	1.4128	2.8233	114	0.0266	0.0480
Synthetic S2	35	40330	3.0717	5.4218	270312	0.1127	0.2415	322	9.4290	12.9280	112	3.6546	4.4746	20492	0.6639	1.5575	212	0.0401	0.0462
Timber House	240	128766	1.0594	1.2549	549202	0.1427	0.5316	66	4.6438	8.0946	384	1.7177	2.8251	6016	1.2525	1.5646	596	0.8656	0.9195

faithful candidate planes if the plane has no or only a few line segments. Taking Fig. 14 as an example, line segments are difficult to extract from the hollow structure near pillars. Thus, the building shape can not be maintained well.

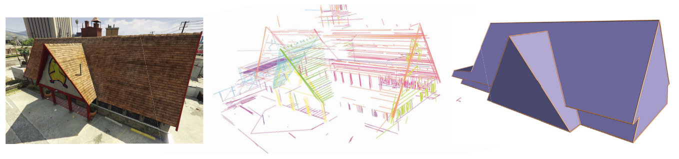


Fig. 14. We can not reconstruct the hollow structure near the pillars because it is hard to extract valid support lines from these regions.

Second, we use a modified PolyFit to obtain the final surface mesh from candidate planar primitives. However, according to our experiments, the running time of PolyFit would be a bottleneck if a large number of candidate planes for complex buildings exist. Finally, as our goal is to reconstruct coarse polygonal surfaces quickly, the geometric details (*e.g.*, doors, windows) of the buildings are missing. A possible solution for future study would be to perform instance segmentation on the input images, then add the details to our coarse model using a template assembly approach [64].

6 Conclusion and Future Work

We have presented a new approach for line-based 3D scene abstraction and modeling from images with rich line segments texture. Our algorithm can extract more complete and continuous 2D line segments, which are then matched to obtain a high-quality 3D line cloud. Moreover, we propose an optimization approach to find candidate planes for 3D line segments accurately. It allows us to create a manifold and compact surface model. We demonstrate the effectiveness and advantages of our approach by comparing it with the state-of-the-art methods on synthetic and real data.

Future work. Besides addressing the limitations above, we would like to extend our approach to support 3D curves [65] or other types of geometric primitives (*e.g.*, cylinders and spheres). This approach would allow us to generate polygonal surfaces for more complex and general scenes.

In addition, we plan to perform semantic analysis on 2D line segments using deep learning to distinguish the semantic and textural line segments [66]. A semantic line represents the edge of the outline of an object, and it is the intersection of different planes; the textural line is the boundary due to the change of texture color, and it is only located on a single plane. Such semantic information can be used to verify the matching relationship between line segments and filter false matches, *i.e.*, only the line segments with the same semantics can be a possible correct match.

Acknowledgments

The authors thank anonymous reviewers for their valuable comments. They also would like to thank Yuzhi Chen and Zhenbang He for preparing the results of some comparison methods. This work is partially funded by the National Natural Science Foundation of China (62172416, U22B2034, U21A20515, 61972388), Shenzhen Science and Technology Program (JCYJ20180507182222355, GJHZ20210705141402008), and Youth Innovation Promotion Association of the Chinese Academy of Sciences (2022131).

References

[1] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, p. 133, 1981.

[2] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from internet photo collections," *Int. Journal of Computer Vision*, vol. 80, no. 2, pp. 189–210, 2008.

[3] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski, "Building rome in a day," *Communications of the ACM*, vol. 54, no. 10, pp. 105–112, 2011.

[4] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multiview stereopsis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 8, pp. 1362–1376, 2010.

[5] H.-H. Vu, P. Labatut, J.-P. Pons, and R. Keriven, "High accuracy and visibility-consistent dense multiview stereo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 5, pp. 889–901, 2011.

[6] Y. Furukawa and C. Hernández, "Multi-view stereo: A tutorial," *Foundations and Trends® in Computer Graphics and Vision*, vol. 9, no. 1-2, pp. 1–148, 2015.

[7] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 835–846, Jul. 2006.

[8] C. Wu, "Visualsfm: A visual structure from motion system," *http://ccwu.me/vsfm/*, 2011.

[9] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.

[11] S. Fuhrmann, F. Langguth, N. Moehrle, M. Waechter, and M. Goesele, "Mve—an image-based reconstruction environment," *Computers & Graphics*, vol. 53, pp. 44–53, 2015.

[12] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Towards internet-scale multi-view stereo," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 1434–1441.

[13] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–13, 2017.

[14] A. Elqursh and A. Elgammal, "Line-based relative pose estimation," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 3049–3056.

[15] Y. Salaün, R. Marlet, and P. Monasse, "Robust and accurate line- and/or point-based pose estimation without manhattan assumptions," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 801–818.

[16] G. Schindler, P. Krishnamurthy, and F. Dellaert, "Line-based structure from motion for urban environments," in *3DPVT*, 2006, pp. 846–853.

[17] A. Bartoli and P. Sturm, "Structure-from-motion using lines: Representation, triangulation, and bundle adjustment," *Comput. Vis. Image Underst.*, vol. 100, no. 3, pp. 416–441, 2005.

[18] B. Micusik and H. Wildenauer, "Structure from motion with line segments under relaxed endpoint constraints," *Int. Journal of Computer Vision*, vol. 124, no. 1, pp. 65–79, 2017.

[19] P. Smith, I. Reid, and A. J. Davison, "Real-time monocular slam with straight lines," in *BMVC*, 2006.

[20] G. Zhang, J. H. Lee, J. Lim, and I. H. Suh, "Building a 3-d line-based map using stereo slam," *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1364–1377, 2015.

[21] B. Micusik and H. Wildenauer, "Descriptor free visual indoor localization with line segments," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3165–3173.

[22] C. Baillard, C. Schmid, A. Zisserman, and A. Fitzgibbon, "Automatic line matching and 3d reconstruction of buildings from multiple views," in *ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagery*, vol. 32, 1999, pp. 69–80.

[23] M. Hofer, A. Wendel, and H. Bischof, "Incremental line-based 3d reconstruction using geometric constraints." in *BMVC*, 2013.

[24] M. Hofer, M. Maurer, and H. Bischof, "Efficient 3d scene abstraction using line segments," *Comput. Vis. Image Underst.*, vol. 157, pp. 167–178, 2017.

[25] T. Sugiura, A. Torii, and M. Okutomi, "3d surface reconstruction from point-and-line cloud," in *International Conference on 3D Vision (3DV)*, 2015, pp. 264–272.

[26] L. Nan and P. Wonka, "Polyfit: Polygonal surface reconstruction from point clouds," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2353–2361.

[27] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, and C. T. Silva, "A survey of surface reconstruction from point clouds," in *Comput. Graph. Forum*, vol. 36, no. 1, 2017, pp. 301–329.

[28] H. Bay, A. Ess, A. Neubeck, and L. Van Gool, "3d from line segments in two poorly-textured, uncalibrated images," in *3D Data Processing, Visualization, and Transmission*, 2006, pp. 496–503.

[29] H. Bay, V. Ferraris, and L. Van Gool, "Wide-baseline stereo matching with line segments," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 329–336.

[30] L. Zhang and R. Koch, "Structure and motion from line correspondences: Representation, projection, initialization and sparse bundle adjustment," *Journal of Visual Communication and Image Representation*, vol. 25, no. 5, pp. 904–915, 2014.

[31] M. Hofer, M. Maurer, and H. Bischof, "Improving sparse 3d models for man-made environments using line-based 3d reconstruction," in *International Conference on 3D Vision (3DV)*, 2014, pp. 535–542.

[32] A. Jain, C. Kurz, T. Thormählen, and H.-P. Seidel, "Exploiting global connectivity constraints for reconstruction of 3d line segments from images," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 1586–1593.

[33] S. Ramalingam and M. Brand, "Lifting 3d manhattan lines from a single image," in *IEEE International Conference on Computer Vision (ICCV)*, 2013, pp. 497–504.

[34] C. Baillard and A. Zisserman, "Automatic reconstruction of piecewise planar models from multiple views," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 1999, pp. 559–565.

[35] L. Zebedin, J. Bauer, K. Karner, and H. Bischof, "Fusion of feature- and area-based information for urban buildings modeling from aerial imagery," in *European Conference on Computer Vision (ECCV)*, 2008, pp. 873–886.

[36] S. Sinha, D. Steedly, and R. Szeliski, "Piecewise planar stereo for image-based rendering," in *IEEE International Conference on Computer Vision (ICCV)*, 2009, pp. 1881–1888.

[37] R. Grompone von Gioi, J. Jakubowicz, J. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 722–732, 2010.

[38] J. Witt and G. Mentges, "Maximally informative surface reconstruction from lines," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 2029–2036.

[39] J. Wang, T. Fang, Q. Su, S. Zhu, J. Liu, S. Cai, C.-L. Tai, and L. Quan, "Image-based building regularization using structural linear features," *IEEE Trans. on Vis. and Comput. Graph.*, vol. 22, no. 6, pp. 1760–1772, 2016.

[40] A. Boulch, M. de La Gorce, and R. Marlet, "Piecewise-planar 3d reconstruction with edge and corner regularization," *Comput. Graph. Forum*, vol. 33, no. 5, pp. 55–64, 2014.

[41] P.-A. Langlois, A. Boulch, and R. Marlet, "Surface reconstruction from 3d line segments," in *International Conference on 3D Vision (3DV)*, 2019, pp. 553–563.

[42] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra, "Globfit: Consistently fitting primitives by discovering global relations," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 1–12, 2011.

[43] F. A. Limberger and M. M. Oliveira, "Real-time detection of planar regions in unorganized point clouds," *Pattern Recognition*, vol. 48, no. 6, pp. 2043–2053, 2015.

[44] A. M. C. Araujo and M. M. Oliveira, "A robust statistics approach for plane detection in unorganized point clouds," *Pattern Recognition*, vol. 100, pp. 1–12, 2020, article 107115.

[45] F. Lafarge, R. Keriven, and M. Brédif, "Insertion of 3-d-primitives in mesh-based representations: Towards compact models preserving the details," *IEEE Trans. Image Process.*, vol. 19, pp. 1683 – 1694, 08 2010.

[46] L. Zhang, J. Guo, J. Xiao, X. Zhang, and D.-M. Yan, "Blending surface segmentation and editing for 3d models," *IEEE Trans. on Vis. and Comput. Graph.*, vol. 28, no. 8, pp. 2879–2894, 2022.

[47] C. Mura, O. Mattausch, A. Jaspe Villanueva, E. Gobbetti, and R. Pajarola, "Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts," *Computers & Graphics*, vol. 44, pp. 20–32, 2014.

[48] A. Monszpart, N. Mellado, G. J. Brostow, and N. J. Mitra, "Rapter: rebuilding man-made scenes with regular arrangements of planes." *ACM Trans. Graph.*, vol. 34, no. 4, pp. 103–1, 2015.

[49] M. Li, P. Wonka, and L. Nan, "Manhattan-world urban reconstruction from point clouds," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 54–69.

[50] A. Kaiser, J. A. Ybanez Zepeda, and T. Boubekeur, "A survey of simple geometric primitives detection methods for captured 3d data," *Comput. Graph. Forum*, vol. 38, no. 1, pp. 167–196, 2019.

[51] J. Chen, M. K. Leung, and Y. Gao, "Noisy logo recognition using line segment hausdorff distance," *Pattern recognition*, vol. 36, no. 4, pp. 943–955, 2003.

[52] P. S. Bullen, *Handbook of means and their inequalities*. Springer Science & Business Media, 2013, vol. 560.

[53] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*, 1996, pp. 226–231.

[54] Y. Gu, Y. He, K. Fatahalian, and G. Blelloch, "Efficient bvh construction via approximate agglomerative clustering," in *Proceedings of High-Performance Graphics Conference*, 2013, pp. 81–88.

[55] X. Lu, J. Yao, K. Li, and L. Li, "Cannylines: A parameter-free line segment detector," in *IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 507–511.

[56] M. Hofer, M. Maurer, and H. Bischof, "Line3d: Efficient 3d scene abstraction for the built environment," in *German Conference on Pattern Recognition*, 2015, pp. 237–248.

[57] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," *arXiv preprint arXiv:1109.2378*, 2011.

[58] R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," in *Comput. Graph. Forum*, vol. 26, no. 2, 2007, pp. 214–226.

[59] P. Denis, J. H. Elder, and F. J. Estrada, "Efficient edge-based methods for estimating manhattan frames in urban imagery," in *European Conference on Computer Vision (ECCV)*, 2008, pp. 197–210.

[60] E. J. Almazàn, R. Tal, Y. Qian, and J. H. Elder, "Mcmlsd: A dynamic programming approach to line segment detection," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5854–5862.

[61] Y. Luo, J. Ren, X. Zhe, D. Kang, Y. Xu, P. Wonka, and L. Bao, "Learning to construct 3d building wireframes from 3d line clouds," *arXiv preprint arXiv:2208.11948*, 2022.

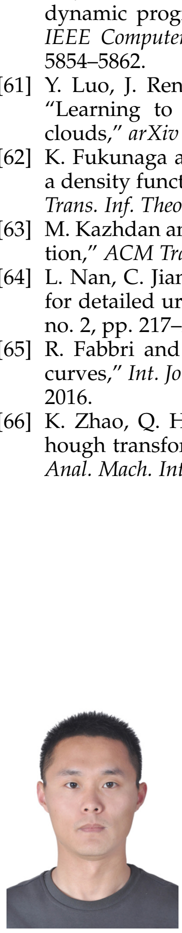
[62] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Trans. Inf. Theory*, vol. 21, no. 1, pp. 32–40, 1975.

[63] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 3, pp. 1–13, 2013.

[64] L. Nan, C. Jiang, B. Ghanem, and P. Wonka, "Template assembly for detailed urban reconstruction," *Comput. Graph. Forum*, vol. 34, no. 2, pp. 217–228, 2015.

[65] R. Fabbri and B. B. Kimia, "Multiview differential geometry of curves," *Int. Journal of Computer Vision*, vol. 120, no. 3, pp. 324–346, 2016.

[66] K. Zhao, Q. Han, C.-B. Zhang, J. Xu, and M.-M. Cheng, "Deep hough transform for semantic line detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021.



Jianwei Guo is an associate professor in National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences (CASIA). He received his Ph.D. degree in computer science from CASIA in 2016, and bachelor degree from Shandong University in 2011. His research interests include computer graphics and 3D vision.

Yanchao Liu received his B.S. degree from Sun Yet-sen University, China, in 2016. He is currently working towards the Ph.D. in computer engineering at the School of Artificial Intelligence, University of Chinese Academy of Sciences and NLPR, Institute of Automation, Chinese Academy of Sciences. His research interests include geometric learning, 3D reconstruction and computer vision.

Xin Song is a graphic engineer in TiMi Studio Group, a subsidiary of Tencent Games. He got his master's degree in computer engineering at Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, in 2018. His research interests include 3D reconstruction, rendering and image processing.

Haoyu Liu received his B.S. degree from Harbin Institute of Technology, Harbin, China in 2019. He is currently working towards the M.S. degree in computer engineering at the School of Artificial Intelligence, University of Chinese Academy of Sciences and Institute of Automation, Chinese Academy of Sciences. His research interests include 3D reconstruction and point cloud processing.

Xiaopeng Zhang received the PhD degree in computer science from Institute of Software, Chinese Academic of Sciences in 1999. He is a professor in National Laboratory of Pattern Recognition at Institute of Automation, Chinese Academy of Sciences. He received the National Scientific and Technological Progress Prize (second class) in 2004 and the Chinese Award of Excellent Patents in 2012. His main research interests include image processing, computer graphics and computer vision.

Zhanglin Cheng is a professor in the Shenzhen Key Laboratory of Visual Computing and Analytics (VisuCA), Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. He received the Ph.D. degree from Institute of Automation, Chinese Academy of Sciences in 2008. His research interests include computer graphics and visualization.