A Real-Time Multi-Model Parametric Representation of Point Clouds

Anonymous author

***Abstract*— In recent years, parametric representations of point clouds have been widely applied in tasks such as memory-efficient mapping and multi-robot collaboration. Highly adaptive models, like spline surfaces or quadrics, are computationally expensive in detection or fitting. In contrast, real-time methods, such as Gaussian mixture models or planes, have low degrees of freedom, making high accuracy with few primitives difficult. To tackle this problem, a multi-model parametric representation with real-time surface detection and fitting is proposed. Specifically, the Gaussian mixture model is first employed to segment the point cloud into multiple clusters. Then, flat clusters are selected and merged into planes or curved surfaces. Planes can be easily fitted and delimited by a 2D voxel-based boundary description method. Surfaces with curvature are fitted by B-spline surfaces and the same boundary description method is employed. Through evaluations on multiple public datasets, the proposed surface detection exhibits greater robustness than the state-of-the-art approach, with 3.78 times improvement in efficiency. Meanwhile, this representation achieves a 2-fold gain in accuracy over Gaussian mixture models, operating at 36.4 fps on a low-power onboard computer.**

I. Introduction

Parametric representation of point clouds plays an important role in robotic perception. For simultaneous localization and mapping (SLAM), point cloud parametric representation can enable the system to apply line, plane or surface features in the environment, thus decreasing the memory consumption and improving efficiency [1]–[5]. In multi-robot systems, compressing point clouds with parametric models sharply decreases the requirement of bandwidth for exchanging map information [6]–[8]. Moreover, by extracting specific environmental feature information, robots can perform tasks, such as docking and grasping, automatically [9], [10].

Due to the limited computational resources of robotic systems, achieving high-precision point cloud parameterization in real-time remains challenging. Recently, parameterization methods based on 3D Gaussian splatting (3DGS) have attracted significant attention [11]–[13]. While effective for accurate 3D reconstruction, these methods are unsuitable for real-time robotic applications. Surface-based parameterization techniques, such as the B-spline surface [14]–[18] and other models [19], can accurately fit complex geometries and curvature in point clouds. They typically rely on region-growing [17], [20] or supervoxel segmentation [21] for curved surface detection, which needs careful parameter tuning. These approaches require multiple passes over the data, making real-time operations on low-power onboard computers difficult. In addition to curvature fitting, B-spline surfaces demand boundary description, involving either manual selection of control points [17] or computationally expensive automatic refinement [18].

Fig. 1. Demonstration of multi-model representation. (a), (b), and (c) show the original frames, GMM-based segmentations and multi-model representations, respectively. Flat regions are represented by planes, curved regions by B-spline surfaces, and unstructured regions by Gaussian distributions.

In contrast to computationally expensive high-precision models, some real-time parametric approaches provide accuracy that is sufficient for robotic tasks. Voxel maps and octree maps [22], [23] are widely used for obstacle avoidance, representing point clouds as collections of cubes. However, they are essentially a form of point cloud down-sampling, making it difficult to extract geometric features from the environment. Parametric representations based on planes and triangular meshes [24]–[26] are well-suited for modeling planar regions in the environment but fail to approximate curved surfaces with a small number of geometric primitives. Quadratic surface-based methods [1], [27], [28] can fit both planar and curved surfaces. Unfortunately, due to their implicit formulation and the presence of multiple solutions,

This work was supported by the AAAA AAAA AAAA AAAA. *(Corresponding author: AAAA.)*

All authors are with the AAAA AAAA AAAA AAAA, AAAA (e-mail: AAAA@AAAA; AAAA@AAAA).

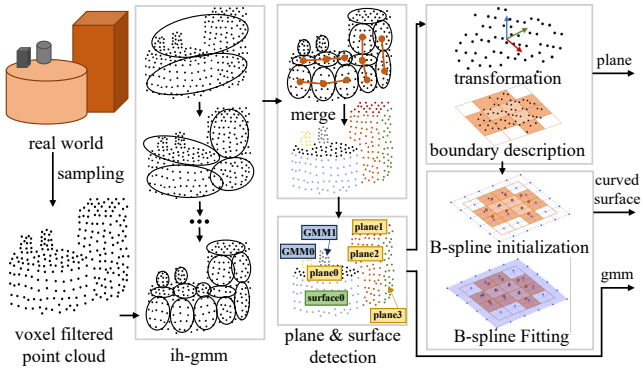

Fig. 2. The overall process of the proposed method. A point cloud sampled from the real world is first voxel-filtered, and then over-segmented using an integrated hierarchical clustering method. Flat clusters are extracted and merged together and three types of clusters are obtained: GMMs, planes, and surfaces. Points belonging to planes and surfaces are transformed into their respective local coordinate systems, and their boundaries are described. For curved surfaces, B-spline surface fitting is applied to model curvature.

Algorithm 1 Multi-model Parametric Representation

Input: Point Matrix $\mathbf{P}$

Output: Model Parameters $\mathbf{\Theta}^{\mathcal{G}}$, $\mathbf{\Theta}^{\mathcal{P}}$, $\mathbf{\Theta}^{\mathcal{S}}$

1: $\mathcal{M} \triangleq \{\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2 \dots\}$
2: **procedure** HIERARCHICAL_MULTIMODEL_REPRESENTATION($\mathbf{P}$)
3: $\mathcal{M} \leftarrow$ IntegratedHierarchicalClustering($\mathbf{P}$) III-A
4: $\{\mathcal{M}^{\mathcal{G}}, \mathcal{M}^{\mathcal{P}}, \mathcal{M}^{\mathcal{S}}\} \leftarrow$ MergeAndDetection($\mathcal{M}$) III-B
5: **for** $i \leftarrow 0; i < \mathcal{M}^{\mathcal{G}}$.size() $- 1; i \leftarrow i + 1$
6: $\mathbf{\Theta}^{\mathcal{G}}$.pushback(GaussianEstimate($\mathbf{P}_i^{\mathcal{G}}$))
7: **end for**
8: **for** $j \leftarrow 0; j < \mathcal{M}^{\mathcal{P}}$.size() $- 1; j \leftarrow j + 1$
9: $\{\theta^{\mathcal{P}}.mean, \theta^{\mathcal{P}}.cov, \tilde{\mathbf{P}}_j^{\mathcal{P}}\} \leftarrow$ Transform($\mathbf{P}_j^{\mathcal{P}}$) IV-A
10: $\theta^{\mathcal{P}}.bdy \leftarrow$ BoundaryCal($\tilde{\mathbf{P}}_j^{\mathcal{P}}$) IV-B
11: $\mathbf{\Theta}^{\mathcal{P}}$.pushback($\theta^{\mathcal{P}}$)
12: **end for**
13: **for** $k \leftarrow 0; k < \mathcal{M}^{\mathcal{S}}$.size() $- 1; k \leftarrow k + 1$
14: $\{\theta^{\mathcal{S}}.mean, \theta^{\mathcal{S}}.cov, \tilde{\mathbf{P}}_k^{\mathcal{S}}\} \leftarrow$ Transform($\mathbf{P}_k^{\mathcal{S}}$) IV-A
15: $\theta^{\mathcal{S}}.bdy \leftarrow$ BoundaryCal($\tilde{\mathbf{P}}_k^{\mathcal{S}}$) IV-B
16: $\theta^{\mathcal{S}}.control_points \leftarrow$ BsplineFitting($\tilde{\mathbf{P}}_k^{\mathcal{S}}, \theta^{\mathcal{S}}.bdy$) V
17: $\mathbf{\Theta}^{\mathcal{S}}$.pushback($\theta^{\mathcal{S}}$)
18: **end for**
19: **return** $\mathbf{\Theta}^{\mathcal{G}}$,$\mathbf{\Theta}^{\mathcal{P}}$,$\mathbf{\Theta}^{\mathcal{S}}$
20: **end procedure**

representing point clouds with these methods makes boundary description cumbersome. Except for voxel maps, plane- and surface-based methods can only fit structured regions in the point cloud. For unstructured point clouds, balancing accuracy and memory consumption is challenging.

To mitigate this issue, highly real-time Gaussian mixture models (GMMs) have been widely proposed in recent years [29]–[33]. The construction of GMM maps can be classified into two main approaches. The first is optimization-based, typically relying on the expectation-maximization (EM) algorithm [29], [30]. The second involves region-partitioning and merging [31]–[33]. Several studies [7], [34] have extended these local mapping approaches to achieve global map construction. Such methods enable real-time model representations on robots with limited computational resources. However, ellipsoids struggle to accurately fit straight boundaries and sharp corners. Therefore, without increasing the number of distributions, such methods suffer from a limited accuracy ceiling.

In summary, GMM-based approaches fit unstructured point clouds well but exhibit lower accuracy for structured regions. Plane-based representations achieve higher accuracy for flat regions but are not memory-efficient for curved regions. Curvature can be better fitted by curved surfaces. Combining the strengths of planes, curved surfaces and GMMs enables a multi-model point cloud representation: GMMs handle unstructured regions, while planes or curved surfaces model highly structured areas. Existing multi-model methods are mainly applied in SLAM systems. [1] represents point clouds using quadratic surfaces and degenerate quadrics, i.e., Gaussian distributions. This approach replaces surface detection with point cloud segmentation [35], achieving real-time performance at the cost of low detection rates. Other multi-model representations, based on lines, planes and cylinders, do not effectively fit coarse regions [2]–[5]. The main challenge is how to represent curved surfaces in a multi-model representation and to develop a real-time and suitable surface detection for such surface models.

To tackle the problem, this paper proposes a multi-model parametric representation of point clouds with real-time surface detection and fitting (Fig. 1). The point cloud is first represented using GMMs. Next, flat point cloud clusters are extracted from GMMs and merged into larger surfaces according to their normal vectors and distances between them. The merged point clouds can be fitted with planes and B-spline surfaces. Finally, the planes and surfaces are projected onto a two-dimensional mesh, with their boundaries determined by constraining each voxel's extents. Our main contributions are as follows:

1) Our approach improves the **robustness** and **efficiency** of curved surface detection. Compared with previous methods for B-spline surface detection, our approach achieves an efficiency improvement of **3.78** times.
2) This work presents a multi-model parametric representation framework that effectively integrates Gaussian distributions, planes, and B-spline surfaces, choosing the most appropriate model based on each region's geometric characteristics. The proposed method achieves a **twofold** improvement in accuracy over Gaussian mixture models with negligible efficiency loss.

II. ALGORITHM OVERVIEW

The system pipeline is shown in Fig. 2 and summarized in Algorithm 1. The algorithm takes point cloud coordinates as input and outputs the parameters of each model. Lines 3-4 cover integrated hierarchical clustering for GMM (Section III-A) and the subsequent merging and detection (Section III-B). In Lines 9-10, each detected plane is transformed into its local coordinate system and its boundary is determined (Sections IV-A and IV-B). Curved surfaces are fitted with B-spline surfaces in Line 16 (Section V).

III. GMM-Based Plane and Surface Detection

In this section, a method for plane and surface detection is described. In Section III-A, we present an integrated hierarchical clustering approach based on GMM for segmenting the point cloud. Then, the merging strategy of clusters for plane and surface detection is introduced in Section III-B.

A. Integrated Hierarchical Clustering Based on GMM

An integrated hierarchical approach is used for fitting the point clouds with planar geometric primitives. In the previous work [29], the point cloud is segmented hierarchically by K-means [36] and EH-GMM [37] algorithms to boost efficiency and the segmentation is terminated by an evaluation index. The evaluation index considers maximum likelihood function, model complexity and reconstruction accuracy. However, it's not suitable for this work because the hierarchical approach may terminate before the point cloud is segmented into clusters shaped like planar geometric primitives. Therefore, a new evaluation strategy for termination is proposed.

The point cloud is filtered with a_{voxel}-size to decrease computational load. A point matrix representing the voxel-filtered points is defined as $\mathbf{P} = [\mathbf{p}_0, \ldots, \mathbf{p}_n, \ldots, \mathbf{p}_{N-1}]^T$, where $\mathbf{p}_n \in \mathbb{R}^{3\times 1}$ is the 3D coordinate vector of nth point. A one-to-two partitioning operation is denoted as

$$\begin{aligned} &\mathcal{C} : \bigcup_{N=4}^{\infty} \mathbb{R}^{N\times 3} \rightarrow \bigcup_{N'=2}^{\infty} \mathbb{R}^{N'\times 3} \times \bigcup_{N''=2}^{\infty} \mathbb{R}^{N''\times 3} \\ &\mathcal{C}(\mathbf{P}) = (\mathbf{P}', \mathbf{P}''), N = N' + N'' \end{aligned} \tag{1}$$

where each $\mathbf{p}_n^T$ is a row of $\mathbf{P}'$ or $\mathbf{P}''$. The assignment of row $\mathbf{p}_n^T$ is determined by a clustering algorithm selected by

$$\mathcal{C}(\mathbf{P}) = \begin{cases} \mathcal{C}_{\text{K-means}}(\mathbf{P}), & N > N_{\text{em}} \\ \mathcal{C}_{\text{EM-GMM}}(\mathbf{P}), & N \le N_{\text{em}} \end{cases} \tag{2}$$

where N_{em} is a parameter set manually. It limits the number of points in clusters segmented by the EM-GMM algorithm to increase efficiency. To represent a segmentation of the point cloud, the I-th set in iteration is defined as

$${}^{I}\mathcal{M} = \left\{{}^{1}\mathbf{P}, {}^{2}\mathbf{P}, \ldots, {}^{I}\mathbf{P}\right\}, {}^{i}\mathbf{P} \in \mathbb{R}^{N_i\times 3},\ i = 1, \ldots, I \tag{3}$$

where ${}^{i}\mathbf{P}$ is the point matrix of i-th cluster in ${}^{I}\mathcal{M}$. The set is initialized to ${}^{1}\mathcal{M} = \{\mathbf{P}\}$, and the iteration proceeds as:

$$\begin{aligned} &\text{if } f\left({}^{i}\mathbf{P}\right) = 0, \mathcal{C}\left({}^{i}\mathbf{P}\right) = \left({}^{i}\mathbf{P}', {}^{i}\mathbf{P}''\right) \\ &{}^{I+1}\mathcal{M} = \left({}^{I}\mathcal{M} \backslash {}^{i}\mathbf{P}\right) \cup \left\{{}^{i}\mathbf{P}', {}^{i}\mathbf{P}''\right\} \end{aligned} \tag{4}$$

where $f : \bigcup_{N=2}^{\infty} \mathbb{R}^{N\times 3} \rightarrow \{0, 1\}$ represents whether the point cluster ${}^{i}\mathbf{P}$ needs to be further segmented.

The definition of f is described in detail as follows. By calculating the mean vector ${}^{i}\bar{\mathbf{p}} = {N_i}^{-1} \sum_{n=0}^{N_i-1} {}^{i}\mathbf{p}_n$ and covariance matrix ${}^{i}\mathbf{\Sigma} = (N_i - 1)^{-1} \sum_{n=0}^{N_i-1} ({}^{i}\mathbf{p}_n - {}^{i}\bar{\mathbf{p}})({}^{i}\mathbf{p}_n - {}^{i}\bar{\mathbf{p}})^T$, the eigenvalues ${}^{i}\lambda_0, {}^{i}\lambda_1, {}^{i}\lambda_2$ and normalized eigenvectors ${}^{i}\mathbf{v}_0, {}^{i}\mathbf{v}_1, {}^{i}\mathbf{v}_2$ of covariance matrix can be obtained by ${}^{i}\mathbf{\Sigma}\,{}^{i}\mathbf{v}_j = {}^{i}\lambda_j\,{}^{i}\mathbf{v}_j,\ j = 0, 1, 2$, where the eigenvalues satisfy $\lambda_0 \ge \lambda_1 \ge \lambda_2$. According to the classic principal component analysis (PCA) [38], the three-sigma boundary of the points forms an ellipsoid with principal axes ${}^{i}\mathbf{x} = 3\sqrt{{}^{i}\lambda_0}\,{}^{i}\mathbf{v}_0, {}^{i}\mathbf{y} = 3\sqrt{{}^{i}\lambda_1}\,{}^{i}\mathbf{v}_1, {}^{i}\mathbf{z} = 3\sqrt{{}^{i}\lambda_2}\,{}^{i}\mathbf{v}_2$. Based on the ellipsoid, f is defined as $f({}^{i}\mathbf{P}) = \min\{1, f_l({}^{i}\mathbf{P}) \cdot f_p({}^{i}\mathbf{P}) \cdot f_c({}^{i}\mathbf{P}) + f_n({}^{i}\mathbf{P})\}$, with each item illustrated in Fig. 3. To ensure complete envelopes of Gaussian distributions, the Mahalanobis distance judgment f_l is denoted as

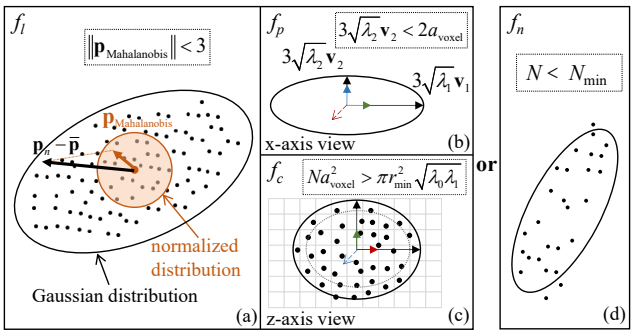

Fig. 3. Termination judgment for integrated hierarchical clustering based on GMM. The Mahalanobis distance is the normalized distance in a Gaussian distribution. If a point lies outside the three-sigma bound, the distribution is considered non-compact and requires further partitioning. In the x-axis view, the principal axis of ellipse along z-axis should be sufficiently short to ensure that the cluster can be fitted as a planar primitive. If this condition is satisfied, the projection of the cluster onto the xy-plane should be dense. If neither of the above conditions is met, only a constraint is imposed to prevent overfitting.

$$\begin{aligned} f_l({}^{i}\mathbf{P}) &= \prod_{n=0}^{N_i-1} s_l({}^{i}\mathbf{p}_n) \\ s_l({}^{i}\mathbf{p}_n) &= \begin{cases} 1, & ({}^{i}\mathbf{p}_n - {}^{i}\bar{\mathbf{p}})^T [{}^{i}\mathbf{\Sigma}]^{-1} ({}^{i}\mathbf{p}_n - {}^{i}\bar{\mathbf{p}}) < 3^2 \\ 0, & \text{else} \end{cases} \end{aligned} \tag{5}$$

where $({}^{i}\mathbf{p}_n - {}^{i}\bar{\mathbf{p}})^T [{}^{i}\mathbf{\Sigma}]^{-1} ({}^{i}\mathbf{p}_n - {}^{i}\bar{\mathbf{p}})$ is the square of the Mahalanobis distance from ${}^{i}\mathbf{p}_n$ to its mean vector ${}^{i}\bar{\mathbf{p}}$. The judgment of flatness f_p is defined as

$$f_p({}^{i}\mathbf{P}) = 1 \text{ if } 6\sqrt{{}^{i}\lambda_2} < a_{\text{voxel}}, \text{ else } 0 \tag{6}$$

The density limitation f_c is described as

$$f_c({}^{i}\mathbf{P}) = 1 \text{ if } N_i a_{\text{voxel}}^2 > \pi r_{\text{min}}^2 \sqrt{{}^{i}\lambda_0\,{}^{i}\lambda_1}, \text{ else } 0 \tag{7}$$

where $r_{\text{min}} \in [2, 3]$ and is set manually. To prevent overfitting, the number of points in a cluster is constrained by

$$f_n({}^{i}\mathbf{P}) = 1 \text{ if } N_i < N_{\text{min}}, \text{ else } 0 \tag{8}$$

where N_{min} is the minimum allowed number of points in a cluster. The integrated hierarchical clustering is completed by repeating (4) until $\forall\ {}^{i}\mathbf{P} \in {}^{I}\mathcal{M}, f\left({}^{i}\mathbf{P}\right) = 1$.

B. Merging of Clusters for Plane and Surface Detection

After the GMM clustering is completed, the flat point clusters can be extracted for further merging. Formally:

$$\begin{aligned} &{}^{I}\mathcal{M} = \mathcal{M}^{\mathcal{G}} \cup \mathcal{M}^{\mathcal{B}},\ \mathcal{M}^{\mathcal{G}} \cap \mathcal{M}^{\mathcal{B}} = \varnothing \\ &\forall\ {}^{i}\mathbf{P}^{\mathcal{G}} \in \mathcal{M}^{\mathcal{G}}, f_l({}^{i}\mathbf{P}^{\mathcal{G}}) \cdot f_p({}^{i}\mathbf{P}^{\mathcal{G}}) \cdot f_c({}^{i}\mathbf{P}^{\mathcal{G}}) = 0 \\ &\forall\ {}^{j}\mathbf{P}^{\mathcal{B}} \in \mathcal{M}^{\mathcal{B}}, f_l({}^{j}\mathbf{P}^{\mathcal{B}}) \cdot f_p({}^{j}\mathbf{P}^{\mathcal{B}}) \cdot f_c({}^{j}\mathbf{P}^{\mathcal{B}}) = 1 \end{aligned} \tag{9}$$

where $\mathcal{M}^{\mathcal{G}}$ represents the set of clusters that should be fitted by Gaussian distributions and clusters in $\mathcal{M}^{\mathcal{B}}$ can be fitted by planar geometric primitives.

To extract planes and surfaces, planar-shaped clusters are merged by concatenating their corresponding point matrices. The judgment function is denoted as

$$d : \bigcup_{N_i=2}^{\infty} \mathbb{R}^{N_i \times 3} \times \bigcup_{N_j=2}^{\infty} \mathbb{R}^{N_j \times 3} \to \{0, 1\} \tag{10}$$

If $d({}^i\mathbf{P}^{\mathcal{B}}, {}^j\mathbf{P}^{\mathcal{B}}) = 1$, the two clusters can be merged into a larger one. d is the product of multiple symbolic functions, i.e., $d = d_a \cdot d_n \cdot d_g \cdot d_p$, and its brief description is shown in Fig. 4. d_a requires that the normal directions of the two clusters are similar, which is expressed as

$$d_a({}^i\mathbf{P}^{\mathcal{B}}, {}^j\mathbf{P}^{\mathcal{B}}) = 1 \text{ if } \left|{}^i\mathbf{v}_2^{\mathcal{B}} \cdot {}^j\mathbf{v}_2^{\mathcal{B}}\right| > \cos(\theta_{\min}), \text{ else } 0 \tag{11}$$

where $\theta_{\min}$ is the smallest angle can be tolerated. The vector from the mean of ${}^i\mathbf{P}^{\mathcal{B}}$ to mean of ${}^j\mathbf{P}^{\mathcal{B}}$ is $\Delta_{i,j}\bar{\mathbf{p}}^{\mathcal{B}} = {}^i\bar{\mathbf{p}}^{\mathcal{B}} - {}^j\bar{\mathbf{p}}^{\mathcal{B}}$, projecting it onto ${}^i\mathbf{v}_2^{\mathcal{B}}$ can obtain the distance between the two means in the normal direction of ${}^i\mathbf{P}^{\mathcal{B}}$, i.e., ${}^i_{i,j}l^{\mathcal{B}} = \|{}^i\mathbf{v}_2^{\mathcal{B}} \cdot \Delta^i_j\bar{\mathbf{p}}^{\mathcal{B}}\|$. According to this, the distance in normal direction judgment d_n is described as

$$d_n({}^i\mathbf{P}^{\mathcal{B}}, {}^j\mathbf{P}^{\mathcal{B}}) = 1 \text{ if } ({}^i_{i,j}l^{\mathcal{B}} + {}^j_{i,j}l^{\mathcal{B}})/2 < l_{\min}, \text{ else } 0 \tag{12}$$

where $l_{\min}$ is set in $[a_{\text{voxel}}, 2a_{\text{voxel}}]$. d_g is denoted as the judgment of Euclidean distance between two clusters. The distance is defined as the center-to-center distance, excluding the segments lying within the three-sigma boundaries of the two distributions. The normalized vector of the connecting line is $\Delta_{i,j}\bar{\mathbf{p}}^{\mathcal{B}}/\|\Delta_{i,j}\bar{\mathbf{p}}^{\mathcal{B}}\|$, rotating it into the space spanned by the three-axis orthogonal basis of i-th distribution, i.e., $\Delta_{i,j}\bar{\mathbf{p}}^{\mathcal{B}}_{\mathbf{v}} = [{}^i\mathbf{Q}^{\mathcal{B}}]^T\Delta_{i,j}\bar{\mathbf{p}}^{\mathcal{B}}/\|\Delta_{i,j}\bar{\mathbf{p}}^{\mathcal{B}}\|$, where ${}^i\mathbf{Q}^{\mathcal{B}} = [{}^i\mathbf{v}_0^{\mathcal{B}}, {}^i\mathbf{v}_1^{\mathcal{B}}, {}^i\mathbf{v}_2^{\mathcal{B}}]$ is an orthogonal matrix. Scaling $\Delta_{i,j}\bar{\mathbf{p}}^{\mathcal{B}}_{\mathbf{v}}$ to the boundary of three-sigma, the length within the boundary of i-th distribution is calculated by ${}^i_{i,j}l^{\mathcal{B}}_{\mathbf{v}} = 3[\sqrt{{}^i\lambda_0^{\mathcal{B}}}, \sqrt{{}^i\lambda_1^{\mathcal{B}}}, \sqrt{{}^i\lambda_2^{\mathcal{B}}}]\Delta_{i,j}\bar{\mathbf{p}}^{\mathcal{B}}_{\mathbf{v}}$. Therefore, the distance between the two boundaries is defined as $\Delta_{i,j} = \|\Delta_{i,j}\bar{\mathbf{p}}^{\mathcal{B}}\| - {}^i_{i,j}l^{\mathcal{B}}_{\mathbf{v}} - {}^j_{i,j}l^{\mathcal{B}}_{\mathbf{v}}$, and d_g can be described as

$$d_g({}^i\mathbf{P}^{\mathcal{B}}, {}^j\mathbf{P}^{\mathcal{B}}) = 1 \text{ if } \Delta_{i,j} < 5a_{\text{voxel}}, \text{ else } 0 \tag{13}$$

d_g can be regarded as a preliminary filter for d_p. The computation of $d_p({}^i\mathbf{P}^{\mathcal{B}}, {}^j\mathbf{P}^{\mathcal{B}})$ traverses the points in the two clusters to calculate the length, i.e.,

$$d_p = 1 \text{ if } \exists m, n, \|{}^i\mathbf{p}_m^{\mathcal{B}} - {}^j\mathbf{p}_n^{\mathcal{B}}\| < 3a_{\text{voxel}}, \text{ else } 0 \tag{14}$$

After computing the merging judgment function between any two clusters, connected components can be identified if such relationships exist. The point matrices of the clusters within each connected component are concatenated by

$$\mathbf{P}^{\text{new}} = [\ldots; {}^i\mathbf{P}^{\mathcal{B}}; {}^j\mathbf{P}^{\mathcal{B}}; \ldots], \forall i, \exists j, d({}^i\mathbf{P}^{\mathcal{B}}, {}^j\mathbf{P}^{\mathcal{B}}) = 1 \tag{15}$$

where $\mathbf{P}^{\text{new}}$ is an example of concatenated matrices. A cluster that cannot be merged with others forms a separate component, i.e., its concatenated matrix is the point matrix itself. By the procedures above, a set of concatenated point matrices, called $\mathcal{M}^{\text{new}}$, is calculated. $\mathcal{M}^{\text{new}}$ is partitioned into merged plane set $\mathcal{M}^{\mathcal{P}}$ and curved surface set $\mathcal{M}^{\mathcal{S}}$, and elements in them satisfy

$$\begin{aligned} &\mathcal{M}^{\mathcal{P}} \cup \mathcal{M}^{\mathcal{P}} = \mathcal{M}^{\text{new}}, \mathcal{M}^{\mathcal{P}} \cap \mathcal{M}^{\mathcal{P}} = \varnothing \\ &\forall {}^k\mathbf{P}^{\mathcal{P}} \in \mathcal{M}^{\mathcal{P}}, f_p({}^k\mathbf{P}^{\mathcal{P}}) = 1 \\ &\forall {}^l\mathbf{P}^{\mathcal{S}} \in \mathcal{M}^{\mathcal{S}}, f_p({}^l\mathbf{P}^{\mathcal{S}}) = 0 \end{aligned} \tag{16}$$

The point cloud is segmented into a set of Gaussian distributions $\mathcal{M}^{\mathcal{G}}$, a plane set $\mathcal{M}^{\mathcal{P}}$, and a surface set $\mathcal{M}^{\mathcal{S}}$. For parameterization, each ${}^i\mathbf{P}^{\mathcal{G}} \in \mathcal{M}^{\mathcal{G}}$ can be represented by its mean and covariance matrix.

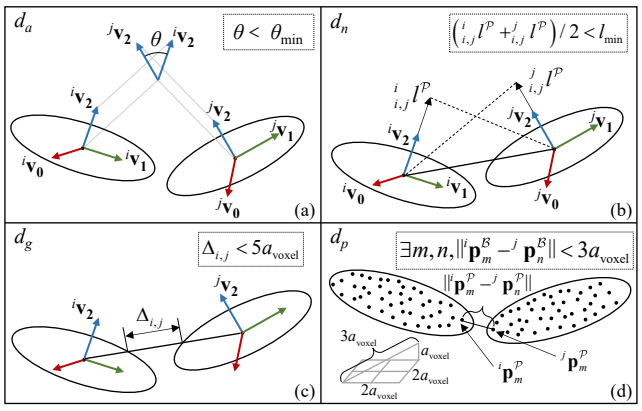

Fig. 4. Merging judgment between two planar geometric primitives described by Gaussian distributions. Firstly, the two planes are required to be parallel. The normal-direction distance between the centroids of two parallel planes should be less than the size of a voxel. Considering possible fitting errors, a distance of less than twice the voxel size may also be accepted. Once these conditions are satisfied, the two planes should be adjacent. A simple geometric distance is computed using the Gaussian distributions to filter out pairs that are far apart. Finally, the point-to-point distances between the two clusters are computed to determine adjacency.

IV. Plane Fitting and Boundary Description

In this section, a method for plane fitting is described. In Section IV-A, we present a coordinate transformation for plane representation. Then, a method for boundary description based on 2D voxels is proposed in Section IV-B.

A. Coordinate Transformation and Plane Representation

In this subsection, only points in ${}^k\mathbf{P}^{\mathcal{P}} \in \mathcal{M}^{\mathcal{P}}$ are considered. For clearer expressions, ${}^k\mathbf{P}^{\mathcal{P}}$ is simply written as $\mathbf{P}$. For each $\mathbf{P}$, a mean vector $\bar{\mathbf{p}}$ and an orthogonal matrix consisting of three orthonormal bases $\mathbf{Q} = [\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2]$ can be obtained. A coordinate transformation is performed on each point $\mathbf{p}_n \in \mathbf{P}$ by

$$\tilde{\mathbf{p}}_n = [\mathbf{Q}]^T(\mathbf{p}_n - \bar{\mathbf{p}}), n = 1, \ldots, N-1 \tag{17}$$

where $\tilde{\mathbf{p}}_n$ is a point in the space spanned by $\mathbf{Q}$. The plane's location and direction can be represented by $\bar{\mathbf{p}}$ and $\mathbf{Q}$.

B. Voxel-Based Boundary Description

An $a^{\mathcal{P}}_{\text{voxel}}$-sized 2D voxel map is calculated as illustrated in Fig. 5. The points in each voxel are described by

$$\tilde{\mathbf{p}}_n \in \mathcal{V}_{i,j}, i = 0, \ldots, N_x - 1, j = 0, \ldots, N_y - 1 \tag{18}$$

where $\mathcal{V}_{i,j}$ is a set of points at location (i, j). To record whether $\mathcal{V}_{i,j} = \varnothing$, $\mathbf{E} \in \{0, 1\}^{N_x \times N_y}$ is defined. If $\mathcal{V}_{i,j} = \varnothing$, $\mathbf{E}_{i,j} = 0$, else $\mathbf{E}_{i,j} = 1$. To determine whether a voxel is at

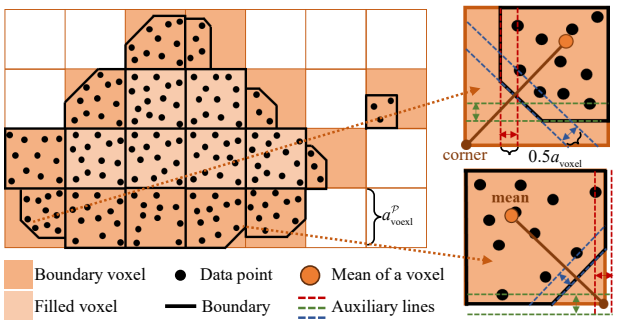

Fig. 5. Boundary description is performed based on a 2D voxel map. The points are first partitioned into different voxels by overlaying a 2D mesh grid. The boundary voxels are then extracted for further boundary characterization. For each voxel, the mean of its points is computed to characterize the point distribution. Empty regions are removed applying line constraints across the grid in three directions: horizontal, 45° diagonal, and vertical. These lines are offset from the data points by a margin of a_{voxel} to avoid direct overlap.

the boundary of a plane, $\mathbf{B} \in \{0,1\}^{N_x \times N_y}$ is defined. For a voxel at the edge of the voxel map (i.e., $i = 0$ or $i = N_x - 1$ or $j = 0$ or $j = N_y - 1$), if $\mathcal{V}_{i,j} \neq \varnothing$, $\mathbf{B}_{i,j} = 1$. For a voxel not at the edge, if $\mathcal{V}_{i,j+1} = \varnothing \vee \mathcal{V}_{i+1,j} = \varnothing \vee \mathcal{V}_{i,j-1} = \varnothing \vee \mathcal{V}_{i-1,j} = \varnothing$ and $\mathcal{V}_{i,j} \neq \varnothing$, $\mathbf{B}_{i,j} = 1$. Otherwise, $\mathbf{B}_{i,j} = 0$.

For each voxel at the boundary, three lines are used to remove empty regions as shown in Fig. 5. The parameters of the lines in voxel (i,j) are defined as $l^X_{i,j}$, $l^Y_{i,j}$ and $l^{XY}_{i,j}$. $l^X_{i,j}$ and $l^Y_{i,j}$ are optimized by

$$\begin{aligned} l^X_{i,j} &= \arg\min \left| x(\tilde{\mathbf{p}}_n) - x'_{i,j} \right| - 0.5a_{\text{voxel}} \\ l^Y_{i,j} &= \arg\min \left| y(\tilde{\mathbf{p}}_n) - y'_{i,j} \right| - 0.5a_{\text{voxel}} \\ s.t.\ &\tilde{\mathbf{p}}_n \in \mathcal{V}_{i,j} \end{aligned} \tag{19}$$

where $(x'_{i,j}, y'_{i,j})$ is the corner point without neighboring points in the voxel (i,j), and $x(\mathbf{p})$ and $y(\mathbf{p})$ denote the respective coordinates of $\mathbf{p}$. $(x'_{i,j}, y'_{i,j})$ is defined by

$$\begin{aligned} (x'_{i,j}, y'_{i,j}) &= \underset{(x'_{i,j}, y'_{i,j}) \in \mathcal{V}^{\text{cor}}_{i,j}}{\arg\max} \| [x'_{i,j}; y'_{i,j}] - \bar{\tilde{\mathbf{p}}}_{i,j} \| \\ \mathcal{V}^{cor}_{i,j} &= \{x^{\min}_{i,j}, x^{\max}_{i,j}\} \times \{y^{\min}_{i,j}, y^{\max}_{i,j}\} \end{aligned}$$

in which $x^{\min}_{i,j}$, etc., represent the coordinates of corner points in voxel (i,j) and $\bar{\tilde{\mathbf{p}}}_{i,j}$ is the mean. Similarly, $\forall \tilde{\mathbf{p}}_n \in \mathcal{V}_{i,j}$

$$l^{XY}_{i,j} = \arg\min \left| x(\tilde{\mathbf{p}}_n) - x'_{i,j} \right| + \left| y(\tilde{\mathbf{p}}_n) - y'_{i,j} \right| - \frac{\sqrt{2}}{2} a_{\text{voxel}} \tag{20}$$

If the point $(x'_{i,j}, y'_{i,j})$ is surrounded by non-empty voxels, the voxel is considered to contain no sparse regions, so $l^X_{i,j}$, $l^Y_{i,j}$ and $l^{XY}_{i,j}$ are all set to 0. To parametrize a plane, several parameters are used, namely $\bar{\mathbf{p}}$, $\mathbf{Q}$, $a^{\mathcal{P}}_{\text{voxel}}$, $x^{\min}_{0,0}$, $y^{\min}_{0,0}$, $\mathbf{E}$ and $\mathbf{B}$ with corresponding $l^X_{i,j}$, $l^Y_{i,j}$ and $l^{XY}_{i,j}$.

V. B-spline Surface Fitting

In this section, we describe a strategy for B-spline surface fitting. The boundary description applies the method proposed in Section IV-B. In Section V-A, the control points of B-spline surfaces are designed and initialized based on the voxels for boundary description. For the fitness of B-spline surface to data points, the control points of B-spline are optimized in Section V-B.

A. Initialization of the Control Points

Control points are applied by B-spline to adjust the curvature of surface. As illustrated in Fig. 6, the x- and y-coordinates of the control points are located at the centers of the voxels in a 2D mesh (blue-gray), with an extra layer surrounding the boundary description mesh (brown). Control points are denoted by $\mathbf{p}^{\text{B}}_{\alpha,\beta}$ ($\alpha = 0, \ldots, N_x + 1$, $\beta = 0, \ldots, N_y + 1$). For the overlap region, i.e., $\alpha = 1, \ldots, N_x$, $\beta = 1, \ldots, N_y$, x and y of control points are

$$\begin{aligned} x(\mathbf{p}^{\text{B}}_{\alpha,\beta}) &= (x^{\min}_{\alpha-1,\beta-1} + x^{\max}_{\alpha-1,\beta-1})/2 \\ y(\mathbf{p}^{\text{B}}_{\alpha,\beta}) &= (y^{\min}_{\alpha-1,\beta-1} + y^{\max}_{\alpha-1,\beta-1})/2 \end{aligned} \tag{21}$$

The z-coordinate of $\mathbf{p}^{\text{B}}_{\alpha,\beta}$ is initialized by $z(\bar{\tilde{\mathbf{p}}}_{\alpha-1,\beta-1})$ and should be further optimized. For the extra layers, e.g., $\alpha = 0$, $\beta = 1, \ldots, N_y$, the coordinates are

$$\begin{aligned} x(\mathbf{p}^{\text{B}}_{0,\beta}) &= x^{\min}_{0,\beta-1} - 0.5a^{\mathcal{S}}_{\text{voxel}},\ z(\mathbf{p}^{\text{B}}_{0,\beta}) = 0 \\ y(\mathbf{p}^{\text{B}}_{0,\beta}) &= (y^{\min}_{0,\beta-1} + y^{\max}_{0,\beta-1})/2 \end{aligned} \tag{22}$$

For other extra layers, $\mathbf{p}^{\text{B}}_{\alpha,\beta}$ are initialized in a similar way as in (22). Referring to [15], the order of the B-spline curve is chosen as $T = 3$ in both x and y directions. The knots vector in x direction is defined as $\mathbf{u}^{\text{x}} \in [0,1]^{N_x+T+3}$ and u^{x}_i is calculated by

$$u^{\text{x}}_i = \begin{cases} 0, & 0 \le i \le T \\ \frac{i-T}{N_x+2-2T}, & T < i < N_x + 2 \\ 1, & N_x + 2 \le i \le N_x + T + 2 \end{cases} \tag{23}$$

$\mathbf{u}^{\text{x}}$ in y direction can be calculated in a same way.

B. Optimization of Z-coordinates of Control Points

The B-spline interpolation is essentially a weighted sum of the control points [15], i.e.,

$$\mathbf{p}^{\text{B}}(u,v) = \sum_{\alpha=0}^{N_x+1} \sum_{\beta=0}^{N_y+1} W_{\alpha,\beta}(u,v) \mathbf{p}^{\text{B}}_{\alpha,\beta}, u, v \in [0,1] \tag{24}$$

where $W_{\alpha,\beta}(u,v)$ is the weight of $\mathbf{p}^{\text{B}}_{\alpha,\beta}$ for calculating $\mathbf{p}^{\text{B}}(u,v)$. In this work, only z-axis has to be considered due

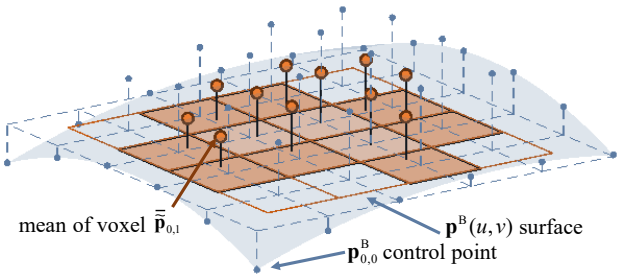

Fig. 6. Computation of B-spline surface control points. Based on a two-dimensional grid, control points are initialized and these points can interpolate a surface. To ensure that the mean of points in each voxel lies on the surface, the coordinates of the control points are further optimized.

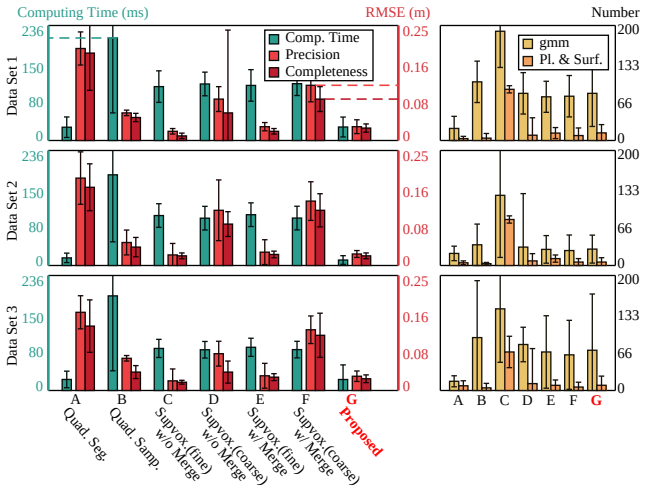

Fig. 7. Results of the comparison between different detection methods. The green bars represent the computation time of each method, while the red bars indicate the RMSEs. In the right-hand figure, the yellow bars denote the number of fitted Gaussian distributions, and the orange bars represent the total number of detected planes and surfaces.

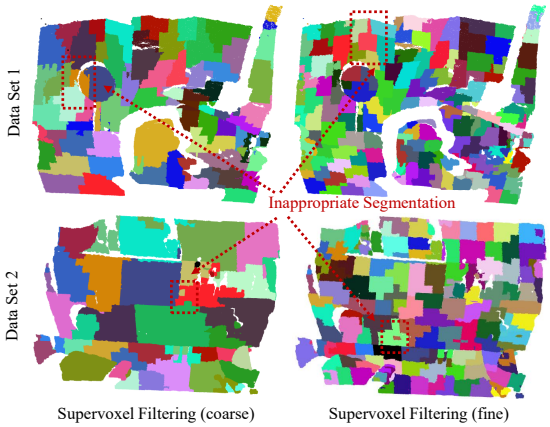

Fig. 8. Demonstration of supervoxel filtering. The upper two figures highlight unreasonable segmentations caused by incorrectly placed supervoxels. The right one includes a corner. The lower two figures illustrate erroneous segmentations resulting from noise.

to the definition of control points. As shown in Fig. 6, by selecting appropriate pairs of (u, v), the B-spline estimation of $z(\bar{\tilde{\mathbf{p}}}_{i,j})$, namely $z^{\mathrm{B}}_{i,j}$, can be obtained by

$$z^{\mathrm{B}}_{i,j} = \sum_{\alpha=0}^{N_x+1} \sum_{\beta=0}^{N_y+1} W_{\alpha,\beta} \left(\frac{i+0.5}{N_x+1} + \frac{x(\bar{\tilde{\mathbf{p}}}_{i,j}) - x^{\min}_{i,j}}{a^{\mathcal{S}}_{\mathrm{voxel}}}, \frac{j+0.5}{N_y+1} + \frac{y(\bar{\tilde{\mathbf{p}}}_{i,j}) - y^{\min}_{i,j}}{a^{\mathcal{S}}_{\mathrm{voxel}}} \right) z(\mathbf{p}_{\alpha,\beta}) \tag{25}$$

For fitting, an unconstrained optimization problem with $z(\mathbf{p}_{\alpha,\beta})$ as optimization variables is formulated as

$$\min \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} \| z(\bar{\tilde{\mathbf{p}}}_{i,j}) - z^{\mathrm{B}}_{i,j} \|^2 \tag{26}$$

Considering the linear relationships between $z^{\mathrm{B}}_{i,j}$ and $z(\mathbf{p}^{\mathrm{B}}_{\alpha,\beta})$, it is evident that this is a convex optimization problem. The cost function can achieve fast convergence through the gradient descent method. In summary, to represent a curved surface, only $z(\mathbf{p}^{\mathrm{B}}_{\alpha,\beta})$ needs to be additionally recorded compared to the parameterization of a plane.

VI. Experiments

In this section, we divide the experiments into two parts: detection of planes and surfaces, and model fitting. To demonstrate the superiority of our plane and surface detection method, we compare it with several existing approaches in Section VI-A. To evaluate the trade-off between efficiency and accuracy, we compare our representation method against commonly used parametric models in Section VI-B. Finally, supplementary experiments on our approach are presented in Section VI-C, and its characteristics are further analyzed.

For the proposed method, its parameters are set by a trial and error approach. N_{em} is set as 200, r_{min} is 2, N_{min} is 40 and θ_{min} is 15°. Three data sets are selected for testing, namely D1: TUM RGB-D freiburg1_teddy, D2: TUM RGB-D freiburg3_long_office_household_validation [39], and D3: ETH RGBD Voxblox [40]. In dataset D1, the size of the voxel filter $a_{\mathrm{voxel}} = 0.04$, and the other two are 0.03. This depends on the scene size and density of the point clouds. The 2D voxel size for boundary description is set to $a^{\mathcal{P}}_{\mathrm{voxel}} = 5a_{\mathrm{voxel}}$ for planes and $a^{\mathcal{S}}_{\mathrm{voxel}} = 3a_{\mathrm{voxel}}$ for surfaces. All experiments are conducted on a commonly used low-power onboard computer, the Jetson Xavier NX (16GB).

A resampled point cloud can be generated from the parametric model representation. The precision of the generated point cloud is evaluated by the root mean squared error (RMSE) of the generated-to-real distance, i.e., the distance from each point in the generated point cloud to its nearest neighbor in the original point cloud. The completeness error is calculated in the reverse direction.

A. Comparison of Surface Detection Methods

For the comparison of surface detection, we apply a point cloud segmentation method for quadratic fitting [1], a sampling-based quadratic detection method [28] and typically used supervoxel filtering methods [14]–[18]. After the sampling method [28] is completed, to ensure that the remaining point clouds can also be fitted, the rest points are processed by the Gaussian clustering method of this work. In supervoxel filtering methods, the point cloud is first over-segmented and then the segmented clusters are merged. Since the approach is parameter-sensitive, the seed resolution of the supervoxel is set to either 0.15 or 0.4. These methods are only applied for detection, while the fitting is performed using the approach proposed in this work.

As shown in Fig. 7, the point cloud segmentation method suffers low accuracy because it focuses more on the Euclidean distance between point cloud clusters rather than the clusters' fitness to the surface. The sampling-based quadratic detection method can detect surfaces with high fitness from point clouds, but repetitive sampling operations are too time-consuming. Both methods have low detection capability.

TABLE I

COMPARISON OF DIFFERENT PARAMETRIC MODELS

Evaluation Index	Only Fitting Time (ms)			Precision RMSE (m)			Completeness RMSE (m)		
Models	D1	D2	D3	D1	D2	D3	D1	D2	D3
B-spline	621	428	591	**0.02**	**0.01**	**0.02**	**0.02**	**0.02**	**0.02**
GMM	**0**	**0**	**0**	0.08	0.07	0.07	0.06	0.06	0.07
Quadric+GMM	**1.02**	**0.36**	**0.71**	–	–	–	0.19	0.17	0.14
Proposed	1.37	0.41	0.60	**0.03**	**0.03**	**0.03**	**0.03**	**0.02**	**0.03**

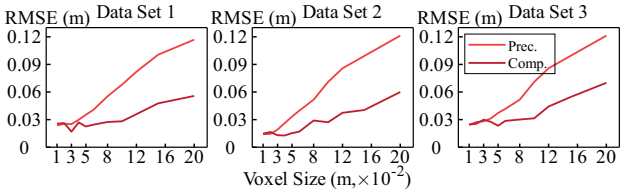

Fig. 9. The sensitivity analysis of voxel size in filtering. It shows that when the voxel size decreases below a certain threshold, the accuracy of the proposed method no longer improves significantly.

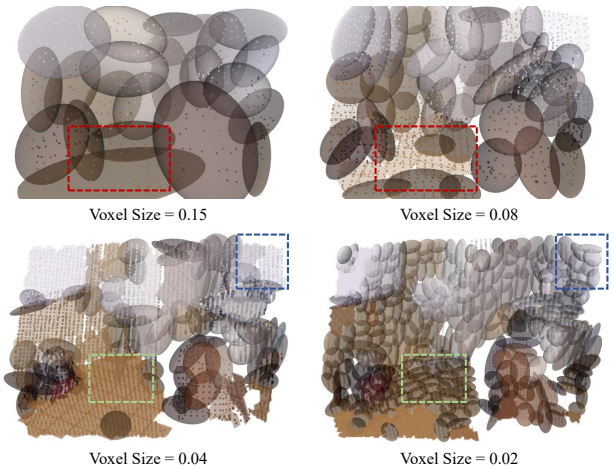

Fig. 10. Representations under different voxel sizes. The red box highlights that finer voxel filtering facilitates plane extraction. The green and blue boxes indicate that overly fine voxel filtering amplifies the impact of errors, thereby hindering plane extraction.

When adjusting the parameters of the supervoxel filter, achieving a balance between compression ratio and accuracy is difficult, as shown in Fig. 7. When the compression ratio is low, the method often produces relatively high errors. If high precision is pursued, this often leads to the excessive use of geometric primitives. Observations from Fig. 8 suggest two main reasons. The position of the voxel may include corners, and supervoxel clustering cannot separate these regions. Corners within such voxels cannot be accurately fitted by Gaussian distributions, planes, or curved surfaces, resulting in high fitting errors. Meanwhile, noise within a voxel may cause incorrect estimation of the normal direction. If the normal of a voxel is not aligned with its neighbors, incomplete merging and disconnected plane detection may occur. By comparison, the proposed method achieves higher accuracy with the same number of primitives. The integrated hierarchical approach of GMM clustering for surface detection is proven to be stable.

In terms of efficiency, our method outperforms the supervoxel approach by factors of **4.21**, **8.86** and **3.78** on the three datasets, respectively. In conclusion, our method can robustly extract the planar and surface features in the point cloud with higher real-time performance.

B. Comparison of Different Geometric Primitives

In this subsection, plane and surface detection is performed by the proposed method. Several baseline methods are selected for comparison. The B-spline fitting method [18] performs both boundary fitting and curvature fitting. Its boundary is also modeled with B-spline curves. Gaussian mixture model (GMM) fitting is carried out following [29], and quadratic surface fitting follows [1].

The RMSE for the quadratic surface resampling cannot be calculated due to the absence of boundary constraints and the existence of multiple solutions. A significant portion of points resampled by quadratic surfaces are located at infinity. All the results are shown in Table I.

It is evident that, although the proposed method is slightly less accurate than the B-spline method, its efficiency is significantly improved. Compared with the slightly faster GMM method, the accuracy is doubled. This improvement comes from representing structured point clouds with a more appropriate parametric model, which provides greater practical significance for robotic applications than the accuracy gain alone.

With regard to the reasons for the proposed method's high efficiency, the time complexity of the boundary description is $O(n)$. Furthermore, due to the well-initialized control points, the convex optimization typically converges rapidly.

C. Supplement

1) Sensitivity to Voxel Filtering: In [29], the stability of hierarchical Gaussian clustering has been demonstrated. Most merging parameters need little to no manual tuning; only the voxel filter size must be chosen based on the environment. However, the scene size provides only a relative reference and cannot determine the absolute voxel size. As shown in Fig. 9, we conducted a sensitivity analysis of voxel size with respect to accuracy. The voxel size selected in our experiments ensures that accuracy remains relatively high.

2) Upper Limit of Precision: As illustrated in Fig. 9, as the voxel size decreases, the accuracy does not improve significantly beyond a certain point. Fig. 10 shows the corresponding representations. Observation suggests two main reasons for this accuracy limit: (1) fitting errors of the Gaussian model on coarse point clouds, and (2) noise in the point cloud. When using GMMs to fit coarse and unstructured point clouds, if the minimum number of points per Gaussian, $N_{\min}$, is not adjusted, the fitting accuracy of their elliptical boundaries cannot approach zero in the hierarchical structure. Reducing the voxel size increases the point cloud density, which shrinks the area represented by each Gaussian. This can obscure the shortest-axis feature

due to noise, potentially causing failures in plane primitive recognition or normal vector estimation.

VII. Conclusions

In this paper, we propose a multi-model parametric point cloud representation method capable of robust and real-time detection of curved surfaces. Structured regions in the point cloud are represented using planes and curved surfaces, while highly noisy regions are modeled with Gaussian distributions. Experiments on various datasets show 3.78 times faster surface detection and 2 times higher accuracy. Parametric representation of space also improves the efficiency and reduces memory usage in robot mapping and localization. However, these benefits remain limited when only individual frames are processed. In future work, we aim to build upon this foundation to integrate robot pose information, with the goal of constructing a global parametric representation.

References

[1] C. Xia et al., "Quadric Representations for LiDAR Odometry, Mapping and Localization," *IEEE Robot. Autom. Lett.*, vol. 8, no. 8, pp. 5023-5030, Aug. 2023.

[2] Zehuan Yu, Zhijian Qiao, Wenyi Liu, Huan Yin, Shaojie Shen, "SLIM: Scalable and Lightweight LiDAR Mapping in Urban Environments," *Tran. Robot.*, vol.41, pp.2569-2588, 2025.

[3] L. Li, M. Yang, L. Weng, and C. Wang, "Robust Localization for Intelligent Vehicles based on Pole-like Features Using the Point Cloud," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 2, pp. 1095-1108, Apr. 2022.

[4] T. Wen et al., "Roadside HD Map Object Reconstruction Using Monocular Camera," *IEEE Robot. Autom. Lett.*, vol.7, no.3, pp. 7722-7729, Jul. 2022.

[5] L. Zhou, G. Huang, Y. Mao, J. Yu, S. Wang, and M. Kaess, "PLC-LiSLAM: Lidar SLAM with planes, lines, and cylinders," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 7163-7170, Jul. 2022.

[6] P. Peng, W. Dong, G. Chen, and X. Zhu, "Obstacle Avoidance of Resilient UAV Swarm Formation with Active Sensing System in the Dense Environment," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Kyoto, Japan, 2022, pp. 10529-10535.

[7] H. Dong et al., "MR-GMMapping: Communication Efficient Multi-Robot Mapping System via Gaussian Mixture Model," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 3294-3301, April 2022.

[8] Y. Wu et al., "MR-GMMExplore: Multi-Robot Exploration System in Unknown Environments based on Gaussian Mixture Model," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, Jinghong, China, 2022, pp. 1198-1203.

[9] K. Hang et al., "Perching and Resting-A Paradigm for UAV Maneuvering with Modularized Landing Gears,", *Sci. Robot.*, vol. 4, no. 28, p. eaau6637, 2019.

[10] K. M. Popek et al., "Autonomous Grasping Robotic Aerial System for Perching (AGRASP)," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Madrid, Spain, 2018, pp. 1-9.

[11] B. Kerbl, G. Kopanas, T. Leimkuehler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *ACM Trans. Graph.*, vol. 42, no. 4, pp. 1-14, Aug. 2023.

[12] X. Lang et al., "Gaussian-LIC: Real-Time Photo-Realistic SLAM with Gaussian Splatting and LiDAR-Inertial-Camera Fusion," in *2025 IEEE Int. Conf. on Robo. and Autom.*, Attend Atlanta, USA, 2025, pp. 1-8.

[13] D. Gao, P. Z. X. Li, V. Sze and S. Karaman, "GEVO: Memory-Efficient Monocular Visual Odometry Using Gaussians," *IEEE Robot. Autom. Lett.*, vol. 10, no. 3, pp. 2774-2781, March 2025.

[14] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillich and M. Vincze, "Segmentation of Unknown Objects in Indoor Environments," in *2012 Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vilamoura-Algarve, Portugal, 2012, pp. 4791-4796.

[15] L. Piegl and W. Tiller, "The NURBS Book," 2nd ed. Berlin, Heidelberg: Springer, 1997.

[16] Y. Kineri, M. Wang, H. Lin and T. Maekawa, "B-spline Surface Fitting by Iterative Geometric Interpolation/approximation Algorithms," *Computer-Aided Design*, vol. 44, no. 7, pp. 697-708, July. 2012.

[17] A. Aldoma, T. Mörwald, J. Prankl and M. Vincze, "Segmentation of Depth Data in Piece-wise Smooth Parametric Surfaces." in *Proc. Comput. Vis. Winter Workshop*, Seggau, Austria, 2015, pp. 1-9.

[18] T. Mörwald, J. Balzer and M. Vincze, "Modeling Connected Regions in Arbitrary Planar Point Clouds by Robust B-spline Approximation," *Robot. Auton. Syst.*, vol. 76, no. 1, pp. 141-151, Dec. 2016.

[19] K. Zhang, S. Xu, Y. Ding, X. Kong and S. Wang, "CURL-SLAM: Continuous and Compact LiDAR Mapping," *Tran. Robot.*, vol. 41, pp. 4538-4556, 2025.

[20] D. Mumford and J. Shah. "Optimal Approximations by Piecewise Smooth Functions and Associated Variational Problems," *Comput.-Aided Des.*, vol. 42, no. 2, pp. 577-685, July. 1989.

[21] J. Papon, A. Abramov, M. Schoeler and F. Wörgötter, "Voxel Cloud Connectivity Segmentation-Supervoxels for Point Clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Portland, OR, USA, 2013, pp. 2027-2034.

[22] A. Elfes, "Using Occupancy Grids for Mobile Robot Perception and Navigation," *Computer*, no. 6, pp. 46-57, 1989.

[23] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An Efficient Probabilistic 3d Mapping Framework Based on Octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189-206, 2013.

[24] J. Poppinga, N. Vaskevicius, A. Birk and K. Pathak, "Fast Plane Detection and Polygonalization in Noisy 3D Range Images," in *2008 Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nice, France, 2008, pp. 3378-3383.

[25] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke, "Real-Time Plane Segmentation Using RGB-D Cameras," in *RoboCup 2011*, Istanbul, Turkey, 2012, pp. 306-317.

[26] R. Hanocka et al., "MeshCNN: a Network with an Edge," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1-12, 2019.

[27] T. Birdal, B. Busam, N. Navab, S. Ilic and P. Sturm, "A Minimalist Approach to Type-Agnostic Detection of Quadrics in Point Clouds," in *2018 CVPR*, Salt Lake City, UT, USA, 2018, pp. 3530-3540.

[28] T. Birdal, B. Busam, N. Navab, S. Ilic and P. Sturm, "Generic Primitive Detection in Point Clouds Using Novel Minimal Quadric Fits," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 6, pp. 1333-1347, June 2020.

[29] Y. Gao and W. Dong, "An Integrated Hierarchical Approach for Real-Time Mapping With Gaussian Mixture Model," *IEEE Robot. Autom. Lett.*, vol. 8, no. 11, pp. 6891-6898, Nov. 2023.

[30] S. Srivastava and N. Michael, "Approximate Continuous Belief Distributions for Precise Autonomous Inspection," in *Proc. IEEE Int. Symp. Saf., Secur.*, Rescue Robot., 2016, pp. 74-80.

[31] P. Z. X. Li, S. Karaman, and V. Sze, "Memory-efficient Gaussian Fitting for Depth Images in Real Time," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 8003-8009.

[32] A. Dhawale and N. Michael, "Efficient Parametric Multi-fidelity Surface Mapping," in *Proc. Robot.: Sci. Syst.*, 2020, vol. 2, pp. 1-9.

[33] J. P. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal, "3D Normal Distributions Transform Occupancy Maps: An efficient representation for mapping in dynamic environments," *Int. J. Robot. Res.*, vol. 32, no. 14, pp. 1627-1644, Dec. 2013.

[34] P. Z. X. Li, S. Karaman and V. Sze, "GMMap: Memory-Efficient Continuous Occupancy Map Using Gaussian Mixture Model," *Tran. Robot.*, vol. 40, pp. 1339-1355, 2024.

[35] I. Bogoslavskyi and C. Stachniss, "Fast Range Image-based Segmentation of Sparse 3D Laser Scans for Online Operation," in *2016 Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Daejeon, Korea (South), 2016, pp. 163-169.

[36] J. B. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," in *Proc. 5th Berkeley Symp. Math. Statist. Prob.*, 2018, vol. 10, no. 3, pp. 281-297.

[37] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Iincomplete Data via the EM Algorithm," *J. Roy. Stat. Soc.: Ser. B. (Methodological)*, vol. 39, no. 1, pp. 1-22, Sep. 1977.

[38] H. Hotelling, "Analysis of a Complex of Statistical Variables into Principal Components," *J. Educ. Psychol.*, vol. 24, no. 1, pp. 498-520, 1933.

[39] J. Sturm, N. Engelhard, F. Endres, W. Burgard and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *2012 Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vilamoura-Algarve, Portugal, 2012, pp. 573-580.

[40] Helen Oleynikova, Zachary Taylor, Marius Fehr, Juan Nieto, and Roland Siegwart, "Voxblox: Building 3D Signed Distance Fields for Planning," in *2017 Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vancouver, Canada, 2017, pp. 1-8.