2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Multi-Resolution Planar Region Extraction for Uneven Terrains

Yinghan Sun[1], Linfang Zheng[1,2], Hua Chen[1] and Wei Zhang[1]

***Abstract*— This paper studies the problem of extracting planar regions in uneven terrains from unordered point cloud measurements. Such a problem is critical in various robotic applications such as robotic perceptive locomotion. While existing approaches have shown promising results in effectively extracting planar regions from the environment, they often suffer from issues such as low computational efficiency or loss of resolution. To address these issues, we propose a multi-resolution planar region extraction strategy in this paper that balances the accuracy in boundaries and computational efficiency. Our method begins with a pointwise classification preprocessing module, which categorizes all sampled points according to their local geometric properties to facilitate multi-resolution segmentation. Subsequently, we arrange the categorized points using an octree, followed by an in-depth analysis of nodes to finish multi-resolution plane segmentation. The efficiency and robustness of the proposed approach are verified via synthetic and real-world experiments, demonstrating our method's ability to generalize effectively across various uneven terrains while maintaining real-time performance, achieving frame rates exceeding 35 FPS.**

I. Introduction

Legged robots are capable of traversing complex and challenging terrains. This unique ability makes them well-suited for missions such as rescue tasks, where wheeled robots of comparable size face substantial difficulties. In recent years, significant progress has been achieved in legged locomotion [1]–[12]. Despite these advancements, it remains challenging to control robots to traverse complex and uneven terrains effectively. One promising pipeline in recent methods [5]–[12] is incorporating geometric visual information into legged locomotion, which segments the input visual data into multiple planes and then uses this information to plan the robot's footholds. However, achieving precise and efficient plane segmentation, particularly in uneven terrains, continues to pose challenges.

In this paper, we introduce a novel approach for efficient, accurate, and robust plane segmentation applicable to uneven terrains. Our method initiates by preprocessing the input point clouds via a pointwise classification module, which categorizes points into a number of groups based on local surface inclinations. We then organize the unordered point clouds via an octree [13], incorporating a specially designed multi-resolution structure. To achieve multi-resolution segmentation, we employ the divide-and-conquer strategy, capitalizing on an in-depth analysis of point distribution within each node. This process is facilitated by a set of proposed criteria that ensure efficiency and noise robustness. In particular, to avoid redundant and time-consuming computations when merging coplanar planes, we propose an incremental implementation for updating the point cluster's covariance matrix. Experimental results demonstrate the robustness of our method across diverse terrains and noise levels, all while achieving real-time performance.

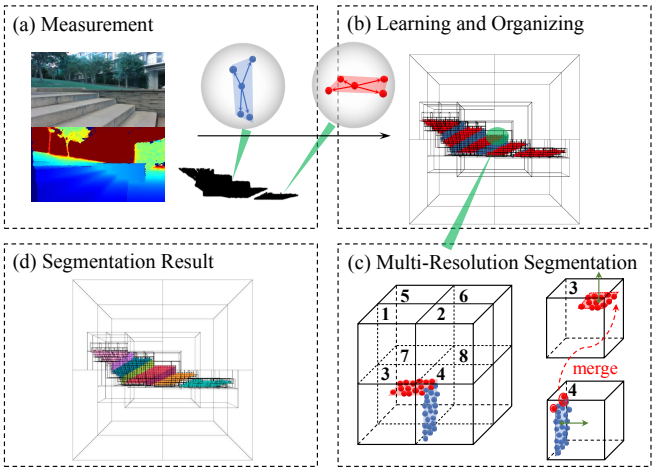

Fig. 1: **The illustration of the proposed method's key idea.** (a) The measured point cloud. (b) Pointwise classification by learning their local geometric properties and organization using an octree. Each square represents an octree's node. (c) Multi-resolution plane segmentation. Children 1 to 2 and 5 to 8 have no points and are therefore not visited. (d) Segmentation results, with points in color belonging to the same plane.

A. Related Work

The research on plane segmentation has attracted considerable attention in recent years. They can be roughly divided into the RANSAC [14] based methods [8], [11], [15]–[17] and the region growing-based methods [18]–[21].

RANSAC-based Methods. RANSAC-based methods usually use RANSAC iteratively on unordered point clouds and remove inliers corresponding to the extracted plane instances. Such a procedure has demonstrated an exceptional ability in handling outliers and successfully segment planes in various legged visual locomotion works [8], [11], [16]. A notable variation is presented in [17], where an octree is employed to structure the unordered point cloud, reducing the need for multiple RANSAC iterations. However, this approach demonstrates speed limitations due to the expensive computation. Moreover, RANSAC exhibits a greedy nature, meaning that any incorrect or missed matches in earlier models cannot be rectified later [22].

This work was supported in part by the National Natural Science Foundation of China (Grant No. 62073159, and Grant No. 62003155), and in part by the Shenzhen Key Laboratory of Control Theory and Intelligent Systems, under Grant No. ZDSYS20220330161800001.

[1]Shenzhen Key Laboratory of Control Theory and Intelligent Systems, School of System Design and Intelligent Manufacturing, Southern University of Science and Technology, Shenzhen, 518055, China.
[2]School of Computer Science, University of Birmingham, UK.

Region-growing-based Methods. These methods operate by initially identifying local planar regions and then iteratively expanding them through merging coplanar regions. Early works [18], [19] primarily focused on pointwise region-growing, demonstrating strong segmentation performance. However, they tend to be time-consuming due to the computational overhead of pointwise searching and merging. Recent advancements in region-growing-based techniques [20], [21], [23] have enhanced processing speed by incorporating connectivity information extracted from 2D image patches and merging coplanar regions to achieve real-time performance. Nevertheless, operating on image patches may sacrifice the plane segmentation resolution, potentially resulting in imprecise plane boundaries [20], [21], [23].

B. Contributions

The main contributions of this paper are summarized as follows: 1) We propose a multi-resolution plane segmentation method that achieves a balance between accuracy and efficiency to meet the requirement of practical applicability. Our method significantly outperforms the RANSAC-based approach, achieving nearly 10 times the processing speed. Moreover, it exhibits a notable reduction of missing points across diverse noise levels compared to region-growing-based methods. 2) We propose a local geometric sensitive pointwise classification module, which reduces the average error on the normal vectors of the extracted planes by approximately tenfold and demonstrates remarkable noise robustness. It also exhibits impressive performance on real-world uneven terrains. 3) We introduce an updating scheme for the covariance matrix estimation that incrementally incorporates new data points during coplanar regions merging. This strategy eliminates redundant matrix multiplication, enhancing the overall effectiveness of the proposed method.

II. Problem Description and Overview

A. Problem Description and Notation Conventions

This paper studies the plane segmentation problem of identifying planes from perceptive measurements. In particular, we consider the input as the 3D point cloud. For depth camera-based applications, the point cloud measurement is obtained by back-projecting the depth image using camera parameters. Specifically, the objective of the plane segmentation problem in this paper is to partition the input points into distinct subsets, each adhering to specific planarity criteria outlined below. The final output is a list of planes, denoted as $L_{\mathcal{P}}$, composed of these subsets.

In this paper, we represent a plane $\mathcal{P}$ using a tuple $(P, N, \boldsymbol{\mu}, \boldsymbol{n}, e_n)$, where P is a set containing all the points in the plane, N and $\boldsymbol{\mu} \in \mathbb{R}^3$ are the point number and centroid of P, $\boldsymbol{n} \in \mathbb{R}^3$ is the plane's normal vector, and $e_n \in \mathbb{R}$ is the mean square error (MSE) in the normal direction.

B. Overview of the Proposed Framework

The schematic diagram of our method is given in Figure 2. Initially, the input point cloud undergoes voxel downsampling to yield a fixed size of N_d points. Subsequently, a pointwise classification step separates points into two categories based on their local surface inclination. These classified points are then organized using an octree, where we assess different point distribution cases within the octree nodes by utilizing point categorical labels. Finally, we implement the multi-resolution plane segmentation using the divide-and-conquer strategy by traversing the octree.

III. Pointwise Classification

To assist our multi-resolution plane segmentation framework, we incorporate a pointwise classification module to pre-process the data. We apply the 3D graph convolution network (3D-GCN) [24] to categorize each point based on its local surface inclinations, thanks to its ability to extract local geometric features from unordered point clouds with shift and scale-invariant properties.

Broadly speaking, points can be classified into distinct categories based on the angle β_i between their local surface normal $\boldsymbol{n}_{l,i}$ and selected baseline vectors $\boldsymbol{b} \in \mathbb{R}^3$:

$$\beta_i = \arccos\left(\frac{|\boldsymbol{b}^{\mathrm{T}}\boldsymbol{n}_{l,i}|}{\|\boldsymbol{b}\|\|\boldsymbol{n}_{l,i}\|}\right). \tag{1}$$

In this work, we exclusively use the gravity vector $\boldsymbol{g}$ as the baseline vector, resulting in a classification into two categories. Specifically, a point is categorized as h if $\beta_i \leq \frac{\pi}{4}$; otherwise, it is assigned to category v.

To train the classification module, we propose to generate a synthetic dataset (as depicted in Figure 2(e)), which contains three types of shapes. Each shape is defined by parameters sampled uniformly from specified ranges: 200 staircases with step dimensions varying from [0.2, 0.4]m in length, [0.2, 1.5]m in width, and [0.08, 0.3]m in height; 200 boxes with dimensions ranging from [0.1, 2.0]m in length and width and [0.08, 0.3]m in height; and 200 planes with dimensions ranging from [0.1, 2.0]m in length and width, and unit normal vectors in the z-component varying from [-1, 1]. To enhance the noise robustness of the trained classifier, we apply the following augmentations to the simulated data: 1) Uniform disturbance of point coordinates ranging from [-0.01, 0.01]m. 2) Random rotation along the z-axis. 3) Removing random clusters from the original point cloud. The network is therefore trained via such a simple dataset to classify each point into two categories.

IV. Plane Segmentation

In this section, we first introduce the design of the nodes' structure in an octree. We then delve into the analysis of different point distribution cases within a node. This analysis will serve as a guide for determining when to divide, when to conquer, and how to conquer a node. Subsequently, we outline the multi-resolution plane segmentation process through octree traversal. Additionally, an incremental implementation for plane merging is proposed to accelerate the process.

A. The Structure of the Octree

An octree [13] is a hierarchical tree structure used to organize 3D data. Each node in the octree has eight children. The octree's spatial resolution is determined by its maximum

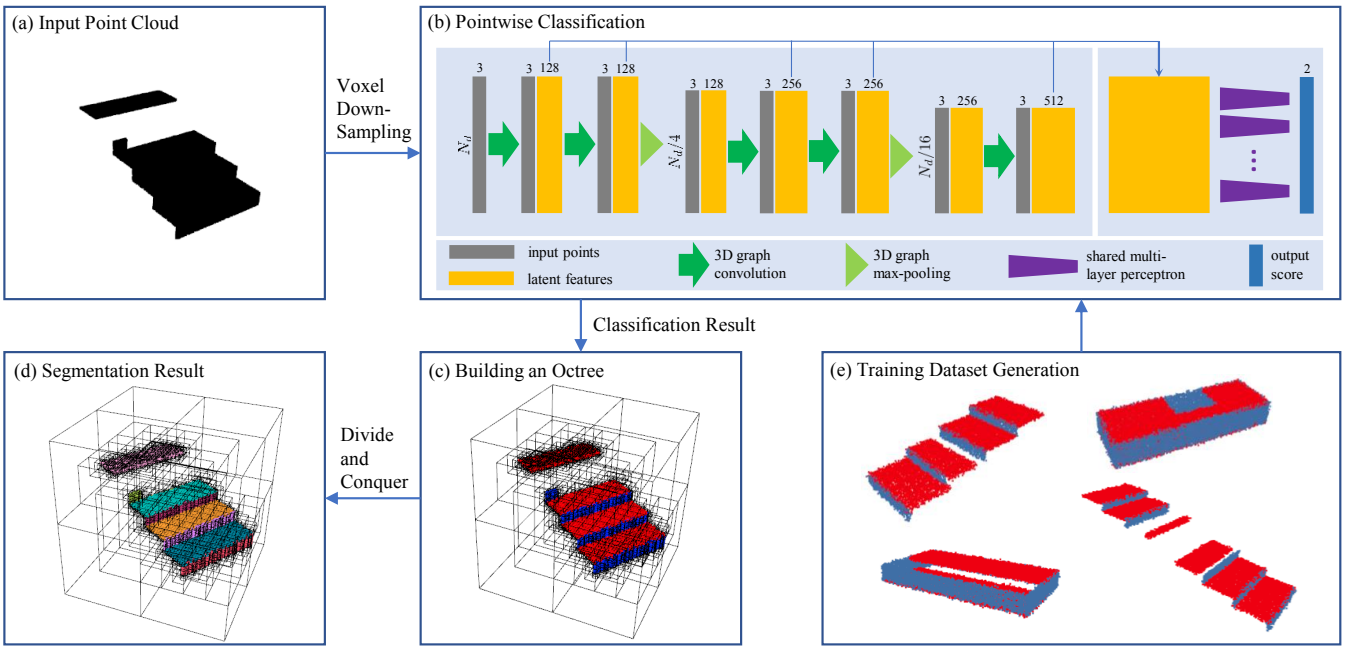

Fig. 2: **The overall framework of the proposed method.** (a) We perform voxel down-sampling on the input point cloud to obtain a point cloud with size N_d for post-processing. (b) The down-sampled points are categorized based on the local surface's inclination around each point. The diagram shows the structure of the 3D-GCN utilized in this step. (c) We organize all points along with their labels using an octree. Each cube represents a node within the octree, with larger cubes representing nodes closer to the root. (d) We apply the divide-and-conquer strategy to achieve multi-resolution plane segmentation. Points with the same color belong to the same plane. (e) To train the neural network, we create a small and simple dataset with proper data augmentations.

depth and the input point cloud's spatial size. To assist the plane segmentation task, each node, denoted as $\mathcal{N}$, stores essential spatial information. This includes its depth in the octree, the spatial bounding box defined by two corner points, pointers to its children and ancestor, counts of points labeled as h and v in the node, and a set P containing all points within the node. Notably, the octree can be efficiently constructed using binary insertion, which effectively manages unordered points.

B. Node Analysis

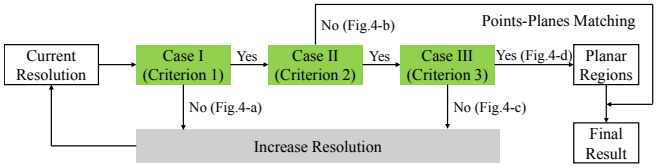

Fig. 3: **Schematic Diagram of the Multi-Resolution Plane Segmentation Process.** The green blocks represent the decision box.

In this subsection, we analyze the different cases that can arise in a node. By applying specific criteria, we aim to determine the optimal moments for node division and conquering and the strategies employed during conquering. This approach enables us to segment planes at multiple resolutions effectively. For clarity, we define the inlier points of a node as the majority of points sharing the same label, while the remaining points are classified as outliers.

1) Case I: We evaluate a node's purity by examining the number of outliers (N_{out}) and the outlier ratio (r_{out}) using Criterion 1. If the criterion is not satisfied, as shown in Figure 4(a), it is highly unlikely for a node to contain only one plane candidate. In such cases, we opt to divide the node.

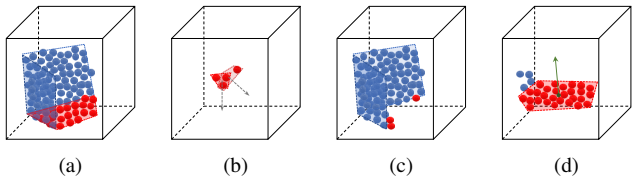

Fig. 4: **The illustration of different cases in a node.** Red color represents the points labeled as h, and blue represents points labeled as v. (a) If a node contains an excessive number of outliers, it is likely to encompass multiple planes. In such cases, we expand the node without applying PCA to prevent unnecessary matrix multiplications. (b) Using too few points to fit a plane can lead to sensitivity to noise. (c) A node may contain multiple planes, even if it meets Criteria 1 and 2. In such cases, the resulting MSE (e_n) of the PCA tends to be large. (d) When all criteria are met, we consider the region constructed by the inlier points as a plane candidate. The green arrow and point indicate the plane's normal vector and centroid, respectively.

Criterion 1 (Purity Criterion)**.** *We say a node satisfies the purity condition if the following conditions hold:*

$$N_{out} \leq \theta_{N_{out}}, \text{ and } r_{out} \leq \theta_{r_{out}}, \tag{2}$$

where $\theta_{N_{out}}$ and $\theta_{r_{out}}$ are two corresponding thresholds.

2) Case II: To determine a plane's equation in space, a minimum of three points is required. However, relying on too few points can lead to undue influence from edges and noise, as demonstrated in Figure 4(b). To enhance robustness against noise, we introduce a hyperparameter that governs the minimum number of points necessary to fit a plane, which leads to the following criterion:

Criterion 2 (Minimum Inlier Points Criterion)**.** *We say a node holds for the minimum inlier points criterion if the number of inlier points N_{in} is larger than a threshold $\theta_{N_{in}}$:*

$$N_{in} \geq \theta_{N_{in}}. \tag{3}$$

3) Case III: For nodes that satisfy Criterion 1 and Criterion 2, we conquer them following [21] and extract their corresponding planes using the principal component analysis (PCA). More precisely, given the inliers' mean position $\boldsymbol{\mu}_{\text{in}} \in \mathbb{R}^3$ and data matrix $\boldsymbol{X}_{\text{in}} = [\boldsymbol{p}_1, \cdots, \boldsymbol{p}_{N_{\text{in}}}] \in \mathbb{R}^{3\times N_{\text{in}}}$, we apply eigenvalue decomposition on the positive semi-definite covariance matrix $\boldsymbol{\Sigma}_{\text{in}} \in \mathbb{R}^{3\times 3}$:

$$\boldsymbol{\Sigma}_{\text{in}} = (\boldsymbol{X}_{\text{in}} - \boldsymbol{\mu}_{\text{in}}\mathbf{1})(\boldsymbol{X}_{\text{in}} - \boldsymbol{\mu}_{\text{in}}\mathbf{1})^{\mathrm{T}}, \tag{4}$$

where $\mathbf{1} \in \mathbb{R}^{1\times N_{\text{in}}}$ is a one matrix. The smallest eigenvalue of $\boldsymbol{\Sigma}_{\text{in}}$ corresponds to the MSE in the normal direction e_n, and its corresponding eigenvector represents the normal vector $\boldsymbol{n}$. We use Criterion 3 to check whether these points belong to a single plane. If the criterion is not met, as illustrated in Figure 4(c), we further divide the node to extract planes at a finer resolution. Otherwise, the plane is directly extracted from this node (see Figure 4(d)). Note that all the operations are conducted on the inlier points. We will show how to deal with the outliers in the next subsection.

Criterion 3 (Plane Candidate Criterion)**.** *We accept a set containing all the inlier points in a node as a plane candidate if the MSE in the normal direction e_n of these points satisfies*

$$e_n \geq \theta_{e_n}. \tag{5}$$

C. Traversing the Octree

After analyzing different cases in a node, we proceed to segment planes by traversing the octree. The key idea is to explore nodes that require division. We establish a queue $Q_{\mathcal{N}}$ to store nodes awaiting exploration. Upon visiting a dequeued node, we evaluate it based on the previously discussed cases. If the node is suitable for conquering, we derive plane parameters from its points. Otherwise, we either enqueue all its children into $Q_{\mathcal{N}}$ or add its points to a list L_p if it's a leaf node, including any identified outliers from Case III. We go through each node until $Q_{\mathcal{N}}$ is empty.

For each extracted plane candidate, we apply Criterion 4 to check whether a coplanar plane already exists in $L_{\mathcal{P}}$. If a coplanar plane is found, we merge the candidate plane with the existing one and update its parameters in $L_{\mathcal{P}}$ accordingly. An incremental implementation of this procedure is discussed in Section IV-D.

Criterion 4 (Coplanarity Criterion)**.** *We say $\mathcal{P}_1$ and $\mathcal{P}_2$ are coplanar if their normal $\boldsymbol{n}$ and mean position $\boldsymbol{\mu}$ satisfy:*

$$\begin{cases} |\boldsymbol{n}_1^{\mathrm{T}}\boldsymbol{n}_2| \geq \theta_{coplanar,n} \\ \left|\boldsymbol{n}_1^{\mathrm{T}}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)/\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|\right| \leq \theta_{coplanar,\mu} \\ \left|\boldsymbol{n}_2^{\mathrm{T}}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)/\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|\right| \leq \theta_{coplanar,\mu} \end{cases} \tag{6}$$

Finally, we address the points in L_p. Note that a plane $\mathcal{P}$ can be defined by its $\boldsymbol{\mu}$ and $\boldsymbol{n}$:

$$n_x(x - \mu_x) + n_y(y - \mu_y) + z(z - \mu_z) = 0. \tag{7}$$

Then, it is convenient to justify whether a point belongs to a plane by computing the distance from points to planes. We use a threshold on this distance that equals 0.005m to handle the noise in the real case. After checking all the points in L_p, we get the final segmentation on the point cloud.

D. Plane Merging

If a newly extracted plane $\mathcal{P}_2$ is coplanar with an existing plane $\mathcal{P}_1$, we merge them and update the plane's properties by applying PCA on the union of P_1 and P_2. However, computing the covariance matrix for this new point set can be computationally demanding, especially if either plane contains a large number of points.

To accelerate this process, we extend the incremental implementation for covariance matrix updates from [18], originally designed for pointwise updates, to handle point clusters. Specifically, we extend the plane representation in Section II by including an additional term $\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}} \in \mathbb{R}^{3\times 3}$. This term contributes to the covariance matrix and can be updated incrementally, as demonstrated below:

$$\boldsymbol{X}_1\boldsymbol{X}_1^{\mathrm{T}} \leftarrow \boldsymbol{X}_1\boldsymbol{X}_1^{\mathrm{T}} + \boldsymbol{X}_2\boldsymbol{X}_2^{\mathrm{T}}. \tag{8}$$

Secondly, $\boldsymbol{\mu}$, N, and P could be updated easily by

$$\boldsymbol{\mu}_1 \leftarrow \frac{N_1}{N_1 + N_2}\boldsymbol{\mu}_1 + \frac{N_2}{N_1 + N_2}\boldsymbol{\mu}_2. \tag{9}$$

$$N_1 \leftarrow N_1 + N_2, \quad P_1 \leftarrow P_1 \cup P_2. \tag{10}$$

Note that the covariance matrix could be expressed as a linear combination of $\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$ and $\boldsymbol{\mu}\boldsymbol{\mu}^{\mathrm{T}}$:

$$\begin{aligned} &(\boldsymbol{X} - \boldsymbol{\mu}\mathbf{1})(\boldsymbol{X} - \boldsymbol{\mu}\mathbf{1})^{\mathrm{T}} \\ =&\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}} - \boldsymbol{X}\mathbf{1}^{\mathrm{T}}\boldsymbol{\mu}^{\mathrm{T}} - \boldsymbol{\mu}\mathbf{1}\boldsymbol{X}^{\mathrm{T}} + \boldsymbol{\mu}\mathbf{1}\mathbf{1}^{\mathrm{T}}\boldsymbol{\mu}^{\mathrm{T}} \\ =&\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}} - N\boldsymbol{\mu}\boldsymbol{\mu}^{\mathrm{T}}. \end{aligned} \tag{11}$$

As $\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$, N, and $\boldsymbol{\mu}$ in the plane representation are updated incrementally, the new covariance matrix can be efficiently computed using Equation (11). Subsequently, $\boldsymbol{n}$ and e_n are updated through the eigenvalue decomposition of the updated 3×3 covariance matrix.

V. Experiments

We perform a series of ablation studies to evaluate the proposed method's precision, efficiency, and robustness qualitatively. We also test our method's generalizability on real-world collected data.

Implementation Details: Our octree implementation and related algorithms were developed in C++. All experiments were conducted on a computer equipped with an Intel(R) Core(TM) i9-10900K CPU, 32GB RAM, and a single NVIDIA GeForce RTX 3090 GPU. We adopt the official code from 3D-GCN [24] with its recommended parameters for the pointwise classification module. The generated synthetic dataset used for evaluation comprises 100 stairs with varying sizes, heights, and numbers of steps. Each stair featured 2 to 6 steps, with step dimensions ranging from 0.2m to 0.4m in length, 0.3m to 1.5m in width, and 0.1m to 0.3m in height. We utilized furthest point sampling in ablation studies to ensure consistent comparison. We also evaluated the generalizability of our method using real-world data captured by an Intel Realsense D435 RGB-D camera at VGA resolution (640 $\times$ 480 pixels). To mitigate depth camera biases, account for robot dimensions, and align with the deployment setup of [1], depth measurements exceeding

1.5m were discarded. Consequently, we choose an octree size of 3.2m $\times$ 3.2m $\times$ 3.2m, representing robot-centric environments with a resolution of 0.02m. The input point cloud was generated by back-projecting depth images using camera intrinsic parameters and downsampled to a voxel size of 0.02m. For all experimental groups, we use the following hyperparameter settings: $\theta_{N_{\text{out}}}$ = 5, $\theta_{r_{\text{out}}}$ = 5%, $\theta_{N_{\text{in}}}$ = 5, θ_{e_n} = 0.005m^2, and $\theta_{\text{coplanar},n}$ = 0.85, and $\theta_{\text{coplanar},\mu}$ = 0.15.

Evaluation Metrics: We evaluate the plane directional error α using the mean angular difference between the estimated plane normal $\hat{\boldsymbol{n}}$ and the ground truth normal $\boldsymbol{n}$. The ratio of the number of segmented planes to the ground truth r_p and the average ratio of missing points r_m are used to further evaluate the segmentation quality.

A. Ablation Study

1) Number of Sampling Points: To assess the impact of point numbers on the performance of our method, we conducted experiments using varying point counts: 2048, 4096, and 8192 points sampled from the generated synthetic dataset. As shown in Table I, the choice of the point number balances the segmentation quality and efficiency. Inadequate sampling, such as using only 2048 points, resulted in detecting only 66% of the correct planes, with more than 25% missed points. Undetected planes primarily occurred for planes of small but feasible size, where insufficient sampling points prevented the satisfaction of Criterion 2 during traversal. Conversely, excessive points reduced the efficiency of pointwise segmentation, resulting in inference times surpassing 23ms for point quantities surpassing 8192, as detailed in Table II. Based on the experimental results, optimal performance was observed with point counts ranging from 4000 to 8000.

TABLE I: **Ablation study on the number of sampling points and effectiveness of the pointwise classification (PC).** $\downarrow$ means the lower value is better. $\rightarrow 100$ means a value closer to 100 is better.

Point Number		2048	4096	8192
with PC	α (°) $\downarrow$	0.03	0.01	0.00
	$r_p \rightarrow 100$	66	97	100
	r_m (%) $\downarrow$	25.2	2.08	0.00
w/o PC	α (°) $\downarrow$	9.17	8.02	12.18
	$r_p \rightarrow 100$	111	122	138
	r_m (%) $\downarrow$	6.81	0.05	0.00

TABLE II: Inference time for pointwise classification.

Point Number	2048	4096	8192
Inference Time (ms)	6	10	23

2) Effectiveness of the Pointwise Classification (PC): To assess the impact of the PC module on improving multi-resolution plane segmentation, we conducted experiments comparing the performance of our method with and without the PC module, using the original synthetic dataset. The dataset was devoid of noise to ensure a fair comparison. In cases where the PC module was absent, we directly applied PCA to each node, as the criteria for node division were inapplicable. As shown in Table I, using PC significantly reduced the mean plane directional error α to nearly zero and brought the number of segmented planes closer to the ground truth. These results indicate a notable enhancement in the quality of segmentation. We further provide a visualization of the segmentation quality in Figure 5, specifically under the condition where the noise level is set to 0.0. This demonstrates our method's enhanced capability in handling plane borders when incorporating the proposed PC module.

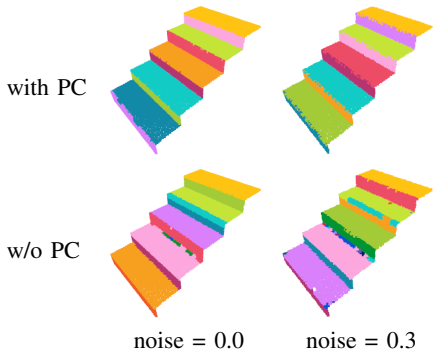

Fig. 5: Comparison of the results of plane segmentation with pointwise classification (PC) and without PC at two different noise levels on synthetic data that contains 10 planes. The results using PC both achieve the ground truth, but the results without PC contain 13 and 19 planes, respectively.

3) Noise Robustness of the Proposed Method: To demonstrate the noise robustness of the proposed method, we introduced different levels of pointwise noise $\mathcal{N} \sim (0, \sigma_i^2)$ into the input point cloud, with σ_i ranging from 0.0 to 0.3. For a fair comparison, we utilized furthest point sampling to sample 8192 points in this study. The corresponding results are presented in Table III. We observed that as the noise level increased from 0.0 to 0.3, α increased by only 1.36°, and r_p increased by only 2% when employing PC. In contrast, these metrics exhibited an increase of 2.8° and 15%, respectively, in the absence of PC. These results also show that PC improves the noise-robust qualities. The corresponding visualization is provided in Figure 5.

TABLE III: **Ablation study on noise robustness.** $\downarrow$ means the lower value is better. $\rightarrow 100$ means a value closer to 100 is better.

Noise Level (cm)		0.0	0.1	0.2	0.3
with PC	α (°) $\downarrow$	0.00	0.10	0.26	1.36
	$r_p \rightarrow 100$	100	100	101	102
	r_m (%) $\downarrow$	0.00	0.00	0.00	0.08
w/o PC	α (°) $\downarrow$	12.18	12.62	13.63	14.98
	$r_p \rightarrow 100$	138	144	152	153
	r_m (%) $\downarrow$	0.00	0.00	0.00	0.04

B. Evaluation on Real Data

As shown in Figure 6, we evaluated the proposed method in three scenarios with different terrains: (a) a stair, (b) grass with a vertical wall, and (c) an indoor composite scene. In particular, (c) is specifically designed to evaluate the maximum point numbers per frame, with the camera positioned approximately 1.5 meters from the wall. We recorded several metrics, including the point number of the

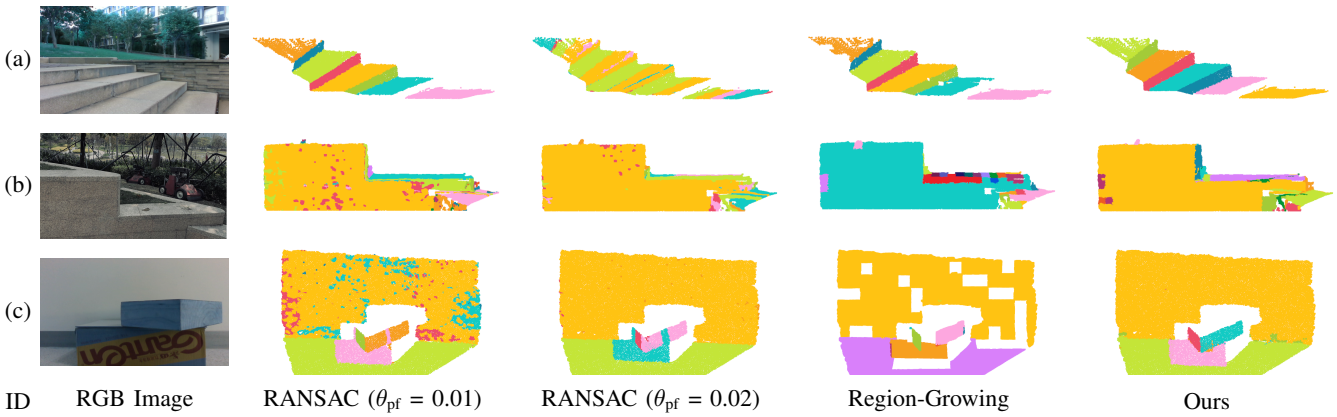

Fig. 6: **Qualitative evaluation results on real data.** We evaluated the proposed method across three diverse terrain scenarios: (a) a stair, (b) grass with a vertical wall, and (c) an indoor composite scene. Depth measurements exceeding 1.5m were disregarded. In each group, the left RGB image displays the ground truth, while the right columns present results for each experimental group, with distinct colors denoting individual planes.

original point cloud (N_m), the point number after down-sampling N_d, the time required for down-sampling t_d, octree construction time t_b, pointwise classification time t_c, and traversal time t_s. All time-related measurements are reported in milliseconds (ms). The results indicate that our method consistently achieves frame rates exceeding 35FPS across various scenarios.

TABLE IV: Details about the real data.

ID	N_m	N_d	t_d	t_c	t_b	t_s	Speed (FPS)
a	105033	5854	3.0	18.8	1.4	0.6	42.0
b	157418	6653	4.4	20.7	2.1	0.5	36.1
c	361516	4197	10.0	13.9	0.8	0.5	39.7

C. Comparison with Existing Methods

We also compared our method with the existing methods. We not only compared the scores of all the methods under different metrics using the same synthetic data as Section V-A.3 but also visually compared the segmentation results on real data. A detailed description of each comparison group is summarized as follows:

1) RANSAC Approach [16]: We conducted experiments using the RANSAC approach in two groups, where we set the plane fitting thresholds (θ_{pf}) to be 0.01m and 0.02m, respectively. Additionally, we set the maximum number of iterations to 1000 and excluded plane candidates with fewer than 50 points.

2) Region-Growing Approach [21], [23]: For this approach, we set the threshold for planar region identification to be 0.01m^2 and the threshold for plane merging to 0.86. When evaluating on real data, we use depth image patches of size 20×20 pixels. Notably, we employed 3D region-growing with a voxel size of 0.05m for synthetic data consisting solely of point clouds to conduct the experiments.

Results on synthetic data: The average processing times for the RANSAC, region-growing, and our method are 3.1FPS, 238.7FPS, and 35.3FPS, respectively. The results in Table V indicate the robustness of all three methods to noise. However, the region-growing approach leads to more missed points, resulting in losses of resolutions. Compared to existing methods, our approach balances efficiency and accuracy, ensuring more practical applicability.

TABLE V: **Comparison with existing methods on synthetic data.** $\downarrow$ means the lower value is better. $\rightarrow 100$ means a value closer to 100 is better.

Noise Level (cm)		0.05	0.10	0.15	0.20
RANSAC (θ_{pf} = 0.01)	α (°) $\downarrow$	0.08	0.11	0.19	0.21
	$r_p \rightarrow 100$	100	100	100	100
	r_m (%) $\downarrow$	0.00	0.04	0.05	0.07
Region-Growing	α (°) $\downarrow$	0.28	0.29	0.34	0.38
	$r_p \rightarrow 100$	100	102	102	103
	r_m (%) $\downarrow$	0.72	0.75	0.77	0.82
Ours	α (°) $\downarrow$	0.04	0.10	0.17	0.26
	$r_p \rightarrow 100$	100	100	101	101
	r_m (%) $\downarrow$	0.00	0.00	0.00	0.00

Results on real data: As shown in Figure 6, the RANSAC approach exhibits sensitivity to hyperparameter settings. Setting θ_{pf} to 0.02m yields improved results in scenes (b) and (c) but fails to segment planes in scene (a) accurately. The region-growing approach consistently results in inaccurate plane boundaries and resolution loss. Compared to existing methods, our approach demonstrates robustness across hyperparameter settings, reduces resolution loss, enhances the accuracy of plane boundaries, and ultimately yields superior outcomes.

VI. Conclusion

This paper introduced a novel multi-resolution method for efficiently extracting planar regions from unordered point clouds. Our approach pre-processes the point cloud with a deep learning approach, then uses an octree-based divide-and-conquer strategy for multi-resolution plane segmentation. We provide an in-depth analysis of point distribution within each octree's node, resulting in accurate and efficient plane segmentation. Extensive experiments demonstrated the effectiveness of our method in segmenting complex terrains in real-world scenarios, showcasing its robustness to noise and real-time performance (over 35FPS). In future research, we aim to use the segmented planes from multiple frames to construct robot-centric maps for perceptive locomotion.

REFERENCES

[1] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[2] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *arXiv preprint arXiv:2306.14874*, 2023.

[3] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter, "Perceptive locomotion in rough terrain–online foothold optimization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 4, pp. 5370–5376, 2020.

[4] O. Melon, R. Orsolino, D. Surovik, M. Geisert, I. Havoutis, and M. Fallon, "Receding-horizon perceptive trajectory optimization for dynamic legged locomotion with learned initialization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9805–9811, IEEE, 2021.

[5] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 279–286, IEEE, 2014.

[6] B. Aceituno-Cabezas, H. Dai, J. Cappelletto, J. C. Grieco, and G. Fernández-López, "A mixed-integer convex optimization framework for robust multilegged robot locomotion planning over challenging terrain," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4467–4472, IEEE, 2017.

[7] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernández-López, and C. Semini, "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 3, pp. 2531–2538, 2017.

[8] R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, and J. Pratt, "Footstep planning for autonomous walking over rough terrain," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pp. 9–16, IEEE, 2019.

[9] Y. H. Lee, Y. H. Lee, H. Lee, H. Kang, Y. B. Kim, J. H. Lee, L. T. Phan, S. Jin, H. Moon, J. C. Koo, *et al.*, "Whole-body motion and landing force control for quadrupedal stair climbing," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4746–4751, IEEE, 2019.

[10] S. Qi, W. Lin, Z. Hong, H. Chen, and W. Zhang, "Perceptive autonomous stair climbing for quadrupedal robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[11] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8352–8358, IEEE, 2021.

[12] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model-predictive control," *IEEE Transactions on Robotics (T-RO)*, 2023.

[13] D. J. Meagher, *Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer*. 1980.

[14] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[15] R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," in *Computer graphics forum*, vol. 26, pp. 214–226, Wiley Online Library, 2007.

[16] M. Lee, Y. Kwon, S. Lee, J. Choe, J. Park, H. Jeong, Y. Heo, M.-S. Kim, J. Sungho, S.-E. Yoon, *et al.*, "Dynamic humanoid locomotion over rough terrain with streamlined perception-control pipeline," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4111–4117, IEEE, 2021.

[17] S. Bertrand, I. Lee, B. Mishra, D. Calvert, J. Pratt, and R. Griffin, "Detecting usable planar regions for legged robot locomotion," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4736–4742, IEEE, 2020.

[18] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak, "Fast plane detection and polygonalization in noisy 3d range images," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3378–3383, IEEE, 2008.

[19] D. Holz and S. Behnke, "Fast range image segmentation and smoothing using approximate surface reconstruction and region growing.," in *IAS (2)*, pp. 61–73, 2012.

[20] C. Feng, Y. Taguchi, and V. R. Kamat, "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6218–6225, IEEE, 2014.

[21] P. F. Proença and Y. Gao, "Fast cylinder and plane extraction from depth cameras for visual odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6813–6820, IEEE, 2018.

[22] S. J. Prince, *Computer Vision: Models, Learning, and Inference*, pp. 264–266. Cambridge University Press, 2012.

[23] Z. Xu, H. Zhu, H. Chen, and W. Zhang, "Polytopic planar region characterization of rough terrains for legged locomotion," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8682–8689, IEEE, 2022.

[24] Z.-H. Lin, S.-Y. Huang, and Y.-C. F. Wang, "Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 1800–1809, 2020.