LiDAR-Based Ground Segmentation with Structured Point Clouds for Multi-Sensor AMRs*

Hamid Didari[1] and Gerald Steinbauer-Wagner[1]

***Abstract*— LiDAR-based perception is a popular component of autonomous mobile robots (AMRs) for obstacle avoidance and traversable area detection. Traditional ground segmentation approaches, such as ring-based methods, often assume a fixed sensor placement and may struggle in multi-LiDAR or tilted sensor configurations. To overcome these limitations, we propose a novel segmentation approach based on the organized point cloud representation, which preserves the spatial arrangement of LiDAR data in a structured 2D format. Our method first organizes the raw point cloud into a structured array, ensuring direct neighborhood accessibility without additional spatial searches. We then use a rolling window over the array to estimate surface normal vectors. Ground segmentation is performed iteratively by classifying normal vectors based on orientation and height consistency. A likelihood approach is further utilized to segment points by assigning them to their corresponding normal vectors. Furthermore, we evaluate our method through experimental tests on a real-world multi-LiDAR AMR in five different scenarios within unstructured environments, achieving an average accuracy of 0.939.**

***Index Terms*— Mobile Robots, Scene Understanding, Off-Road Navigation**

I. INTRODUCTION

The deployment of autonomous mobile robots (AMRs) in logistics has become increasingly prevalent due to their potential to enhance productivity and reduce costs. Research by Keith and La [10] highlights that AMRs improve efficiency by minimizing manual labor in repetitive tasks, leading to lower operational costs and increased throughput. Similarly, a multiple case study by Grover et al. [7] identifies AMRs as key enablers of digital transformation in Industry 4.0 warehouses, where they contribute to cost reduction through efficient material handling and workflow optimization. Economic analyses further indicate that AMRs can lead to substantial long-term savings. A study by Zhang et al. [11] evaluates the return on investment (ROI) of AMR deployment, showing that companies recover their initial investment due to reduced labor expenses and increased productivity.

Despite their benefits, widespread AMR adoption in dynamic and unstructured environments faces several challenges, with local perception being one of the most critical. AMRs rely on LiDAR, cameras, and radar to perceive their surroundings, but sensor noise, occlusions, and environmental variations pose challenges. Among various perception technologies, LiDAR-based perception is particularly effective in enabling AMRs to navigate complex environments by accurately detecting obstacles and identifying traversable areas. By generating high-resolution 3D point clouds, LiDAR sensors provide a precise spatial representation of the surroundings, allowing robots to differentiate between safe paths and potential hazards. This capability is critical for obstacle avoidance and traversable area detection, especially in outdoor and dynamic environments where other sensors may struggle due to lighting variations.

Ground segmentation is a fundamental task in LiDAR-based perception, facilitating accurate obstacle detection and navigation. Traditional methods often employ geometric approaches, such as plane fitting and elevation thresholding, to distinguish ground from non-ground points. However, these methods may struggle with complex terrains and require manual parameter tuning. To overcome these challenges, modern approaches integrate probabilistic models and machine learning techniques. Markov Random Fields (MRF) has been used to model spatial relationships between points, improving segmentation accuracy in uneven terrains [15]. Additionally, deep learning-based methods, such as Convolutional Neural Networks (CNNs), can learn complex patterns in point cloud data, enabling robust ground segmentation in diverse environments [14].

A more recent and efficient approach is ring-based ground segmentation, such as Patchwork, which leverages the structure of LiDAR point clouds to classify ground and obstacles [12]. Patchwork progressively segments the ground from near to far distances using LiDAR's ring structure, improving computational efficiency and robustness in unstructured outdoor terrains. However, a major limitation of Patchwork and many deep learning-based segmentation methods is the assumption that the LiDAR sensor is mounted horizontally at the robot's center. In reality, AMRs may use multiple LiDARs positioned at different angles or orientations—such as tilted or vertically mounted sensors—to enhance 3D coverage.

To overcome this limitation, we developed a segmentation approach based on the organized point cloud representation instead of partitioning space into rings. An organized point cloud is a structured representation where LiDAR points are stored in a 2D array format, preserving their spatial arrangement as captured by the sensor. This contrasts with an unstructured point cloud, where points are stored in a random order without inherent neighborhood relationships. The advantage of an organized point cloud is that each point's neighboring points are directly accessible using fixed

[1]Hamid Didari and Gerald Steinbauer-Wagner are with the Institute of Software Technology, Graz University of Technology, Graz, Austria. {hamid.didari, steinbauer}@ist.tugraz.at

*This work was funded by the Austrian Research Funding Association (FFG) under the scope of the THINK.WOOD.INNOVATION program.

https://doi.org/10.34749/3061-0710.2025.16

indices.

By leveraging this structure, our method ensures that partitioning is independent of the LiDAR setup, making it more adaptable to different sensor configurations. We compute the normal vector for each partition and fit a likelihood model to the points within it. Since the robot is assumed to be on a traversable surface, partitions with a zero mean height are classified as ground. From these initial ground partitions, we iteratively expand to neighboring partitions that are unlabeled. A partition is labeled as ground if its surface inclination angle is below a predefined threshold and its height difference from a known ground partition is within acceptable limits.

Furthermore, we use the likelihood model of each normal vector to classify points in the point cloud. This structured approach allows us to determine which normal vector a given point belongs to, even when neighboring points in the organized point cloud are not necessarily part of the same surface. This is because neighboring in the organized point cloud is based on the sensor's capturing position rather than the actual 3D spatial arrangement.

The structure of the remaining sections of the paper is as follows: the next section gives an overview of the related work, followed by Section III, which provides details on the developed method. In the consecutive section, the evaluation and results are presented, and lastly, in Section V we conclude the paper with drawn conclusions and future work.

II. Related Work

There are different approaches for point cloud segmentation. One approach works directly on the point cloud, as demonstrated by Diaz et al. [4], who proposed two methods for ground segmentation: Normal Vector-Based Filtering, which utilizes KNN, PCA, and Naïve Bayes classification, followed by RANSAC plane fitting to refine ground points; and Voxel-Based Filtering, which structures the point cloud into 3D voxels, applies height-based filtering, 3D adjacency segmentation, and statistical refinement. While the first method achieved slightly higher accuracy, the voxel-based approach was faster, making it the preferred choice for real-time applications. Another notable approach is the fast segmentation method proposed by Himmelsbach et al. [8], designed for autonomous ground vehicles. Their method splits the segmentation process into two steps: local ground plane estimation and fast 2D connected components labeling. This strategy efficiently processes large, unordered 3D point clouds by first separating ground and non-ground points using local plane fitting and then clustering the remaining points based on spatial connectivity. Golovinskiy and Funkhouser [6] introduced a min-cut-based segmentation method that formulates point cloud segmentation as a graph optimization problem. Their approach constructs a k-nearest neighbors graph, applies a background penalty function, and enforces foreground constraints to achieve robust segmentation. The segmentation is determined by solving a global min-cut optimization, which minimizes the cost of separating object points from the background. The method supports both automatic and interactive segmentation and is particularly effective in complex urban environments where object-background separation is challenging. More recently, Huang et al. [9] introduced a coarse-to-fine MRF-based approach to improve ground segmentation accuracy while maintaining computational efficiency. Their method first performs coarse segmentation using local feature extraction to classify points into high-confidence obstacle, ground, and unknown points. The MRF model is then constructed using the coarsely segmented data, eliminating the need for prior knowledge. The graph cut algorithm minimizes the MRF model to refine segmentation results. Additionally, deep learning-based approaches have gained traction for efficient and accurate segmentation of LiDAR point clouds. One such method is SalsaNet, introduced by Aksoy et al. [1], which is an encoder-decoder-based deep learning model designed for fast road and vehicle segmentation. SalsaNet processes LiDAR point clouds in a Bird-Eye-View (BEV) projection and utilizes ResNet blocks in the encoder for efficient feature extraction. It also incorporates a class-balanced loss function to address the imbalance between road and vehicle classes in autonomous driving scenarios. Building upon SalsaNet, Cortinhal et al. [3] introduced SalsaNext, an improved network for semantic segmentation of LiDAR point clouds with uncertainty estimation. SalsaNext extends SalsaNet by incorporating a novel context module, a residual dilated convolution stack, and a pixel-shuffle layer in the decoder to improve segmentation accuracy while maintaining efficiency. Additionally, SalsaNext applies Bayesian treatment to estimate epistemic and aleatoric uncertainties, making it a robust choice for safety-critical applications such as autonomous driving.

III. Method

To overcome the limitations of ring-based segmentation approaches, such as the assumption that the LiDAR is mounted horizontally at the center of the robot, and to support multi-LiDAR configurations, we propose a method that utilizes the organized point cloud representation instead of partitioning the space into concentric rings. Our approach arranges each LiDAR data into a structured array, $\mathbb{R}^{m\times n\times 3}$, where m and n represent the sensor's vertical and horizontal resolution, respectively. This format preserves the spatial arrangement of points as captured by the sensor, with each cell storing its corresponding (x,y,z) coordinates. By maintaining this structured representation, we eliminate the need for a kd-tree [2] to find neighboring points when computing normal vectors. Traditional kd-tree search has a complexity of $\mathcal{O}(\log N)$ for nearest neighbor queries, where N is the number of points in the cloud. In contrast, our structured representation enables direct access to neighboring points in constant time $\mathcal{O}(1)$, reducing computational overhead. The segmentation pipeline consists of the following steps: (1) finding neighboring points based on their indices in the 2D structured array and removing outliers for each LiDAR, (2) estimating normal vectors, (3) classifying normal vectors, (4)

https://doi.org/10.34749/3061-0710.2025.16

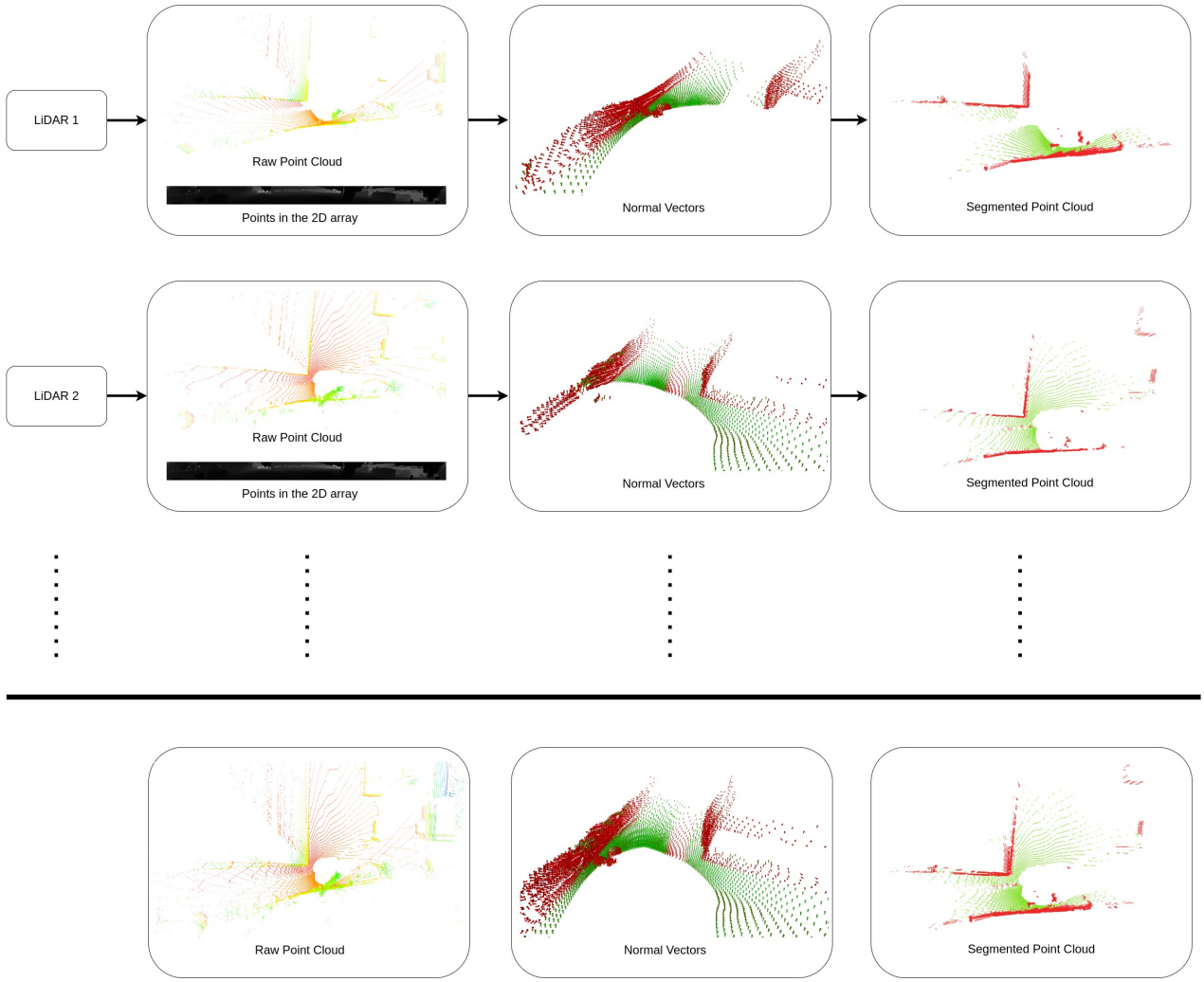

Fig. 1. Segmentation Pipeline: First, for each LiDAR, points are sorted into a structured 2D array. Then, normal vectors are estimated and labeled, followed by assigning points to their corresponding normal vectors.

assigning labels to the points using Likelihood Estimation, and (5) merge the labeled points from different LiDARs into one. A high-level overview of the pipeline is shown in Figure 1.

A. Rolling Window

Given an array where each cell corresponds to a point $P \in \mathbb{R}^{m \times n \times 3}$, we use a rolling window instead of fixed partitioning. This technique ensures that each point's local neighborhood is dynamically considered, leading to better spatial consistency and more accurate normal estimations. A rolling window of size $w \times w$ is defined as:

$$W_{i,j} = \left\{ \mathbf{p}_{u,v} \;\middle|\; i - \frac{w}{2} \leq u \leq i + \frac{w}{2},\ j - \frac{w}{2} \leq v \leq j + \frac{w}{2} \right\}. \quad (1)$$

Since being in the same window does not necessarily imply that all points belong to the same physical surface, we first apply an outlier removal step. Given a point $\mathbf{p} = (x, y, z) \in W_{i,j}$, we define its radial distance as:

$$d_{\mathbf{p}} = \sqrt{x^2 + y^2 + z^2}. \quad (2)$$

A point is considered an outlier and removed if:

$$\frac{|d_{\mathbf{p}} - \bar{d}w_{i,j}|}{\bar{d}w_{i,j}} > \tau_d, \quad (3)$$

where $\bar{d}_{w_{i,j}}$ is the mean distance of all points in Wi, j, and τ_d is a predefined threshold controlling the allowed deviation.

B. Normal Estimation

For each window $W_{i,j}$, we estimate the surface normal vector $\mathbf{n}_{w_{i,j}}$ by fitting a plane using PCA. Given a local set of k neighboring points $\mathscr{X}_{w_{i,j}} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_k\}$ from the window, the normal vector $\mathbf{n}_{w_{i,j}}$ is computed as:

$$\mathbf{n}_{w_{i,j}} = \arg\min_{\mathbf{n}} \sum_{\mathbf{p} \in \mathscr{X}_{w_{i,j}}} (\mathbf{n} \cdot (\mathbf{p} - \bar{\mathbf{p}}_{w_{i,j}}))^2, \quad (4)$$

where $\bar{\mathbf{p}}_{w_{i,j}}$ is the centroid of the points in $\mathscr{X}_{w_{i,j}}$.

https://doi.org/10.34749/3061-0710.2025.16

C. Ground Classification

Since the robot operates on a traversable surface, each $w_{i,j}$ with a mean height $\bar{h}_{w_{i,j}}$ close to zero (in the robot's base frame) is initially classified as ground:

$$G_{w_{i,j}} = \begin{cases} 1, & \text{if } |\bar{h}_{w_{i,j}} - h_r| < \varepsilon_h, \\ -1, & \text{otherwise.} \end{cases} \tag{5}$$

Here, $G_{w_{i,j}} = 1$ indicates that the region is labeled as ground, while $G_{w_{i,j}} = -1$ denotes an unknown classification. h_r represents the reference ground height, and ε_h is a predefined height tolerance threshold.

Next, we apply an iterative expansion strategy. For each labeled ground window $w_{i,j}$, we iteratively check its neighboring window $w_{m,n}$ and classify it as ground if:

$$|\bar{h}_{w_{m,n}} - \bar{h}_{w_{i,j}}| < \delta_h \quad \text{and} \quad \theta_{m,n} < \theta_{\text{thresh}}, \tag{6}$$

where $w_{m,n}$ is a neighbor of $w_{i,j}$, δ_h is the allowed height difference between neighboring window, and $\theta_{m,n}$ is the surface inclination angle computed as:

$$\theta_{m,n} = \cos^{-1}(\mathbf{n}_{m,n} \cdot \mathbf{z}), \tag{7}$$

with $\mathbf{z}$ being the global vertical unit vector. This process is repeated iteratively until no new windows are labeled as ground. Finally, any remaining unlabeled windows are classified as non-ground.

D. Point Labeling Using Likelihood

After estimating the normal vectors and labeling the windows, we use a likelihood approach to estimate whether a given point belongs to a particular surface. This is particularly useful for correctly classifying points that are not direct neighbors in the 2D array but belong to the same physical surface due to edge continuity.

To achieve this, we model the likelihood of a point $\mathbf{p}$ belonging to the surface associated with the normal vector $\mathbf{n}_{w_{i,j}}$ as:

$$L(\mathbf{p} \mid \mathbf{n}_{w_{i,j}}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(d_{\mathbf{p},w_{i,j}})^2}{2\sigma^2}\right), \tag{8}$$

where $d_{\mathbf{p},w_{i,j}}$ is the perpendicular distance of the point $\mathbf{p}$ from the plane defined by the normal vector $\mathbf{n}_{w_{i,j}}$, and $\sigma_{w_{i,j}}$ represents the standard deviation of distances for points within $w_{i,j}$.

The perpendicular distance is computed as:

$$d_{\mathbf{p},w_{i,j}} = (\mathbf{p} - \bar{\mathbf{p}}_{w_{i,j}}) \cdot \mathbf{n}_{w_{i,j}}, \tag{9}$$

where $\bar{\mathbf{p}}_{w_{i,j}}$ is the centroid of $w_{i,j}$.

A point $\mathbf{p}$ is classified as belonging to the surface associated with normal vector $\mathbf{n}_{w_{i,j}}$ if its likelihood $L(\mathbf{p} \mid \mathbf{n}_{w_{i,j}})$ exceeds a predefined threshold τ_L.

To assign labels to individual points, we compute the likelihood of each point belonging to different windows and assign it the label of the window that maximizes the likelihood. Given a point $\mathbf{p}$, its assigned label is:

Fig. 2. The experimental robot setup equipped with two 32-layer Hesai LiDARs.

$$\ell(\mathbf{p}) = \arg\max_{w_{i,j}} L(\mathbf{p} \mid \mathbf{n}_{w_{i,j}}). \tag{10}$$

This approach ensures that each point is assigned to the most probable surface, leading to more consistent and accurate segmentation.

IV. Results

AMRs are deployed in various environments, necessitating different LiDAR configurations based on the specific application. For instance, autonomous vehicles often utilize high-resolution 128-layer LiDARs, like in the KITTI dataset [5], to achieve a comprehensive 360-degree view. In contrast, our application involves operation in unstructured environments, where dense point cloud coverage in front of the robot is crucial for distinguishing traversable and non-traversable slopes. To achieve this without relying on an expensive 128-layer LiDAR, we employ two 32-layer Hesai LiDARs, mounted on the front left and right of the robot. This configuration enhances the density of LiDAR points in the robot's immediate path. The experimental robot setup and LiDAR configuration are illustrated in Figure 2.

To assess the performance of the developed method, we compute error metrics across five different scenarios, including slopes, unstructured environments, and campus areas, as shown in Figure IV, and compare them to Patchwork [12]. Additionally, we use Label Cloud [13] to manually annotate points in the point cloud. In the following section, we introduce the error metrics and evaluate the performance of the developed method across these five scenarios.

A. Error Metrics

To quantitatively evaluate the performance of our method, we employ five metrics: *Precision*, *Recall*, *F1 score*, *Accuracy*, and *Coverage*. These metrics assess the classification performance based on the number of correctly and incorrectly classified points.

Let the number of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) be denoted as N_{TP}, N_{TN}, N_{FP}, and N_{FN}, respectively. The evaluation metrics are defined as follows:

- **Precision** (Positive Predictive Value): measures the proportion of correctly identified positive instances among

https://doi.org/10.34749/3061-0710.2025.16

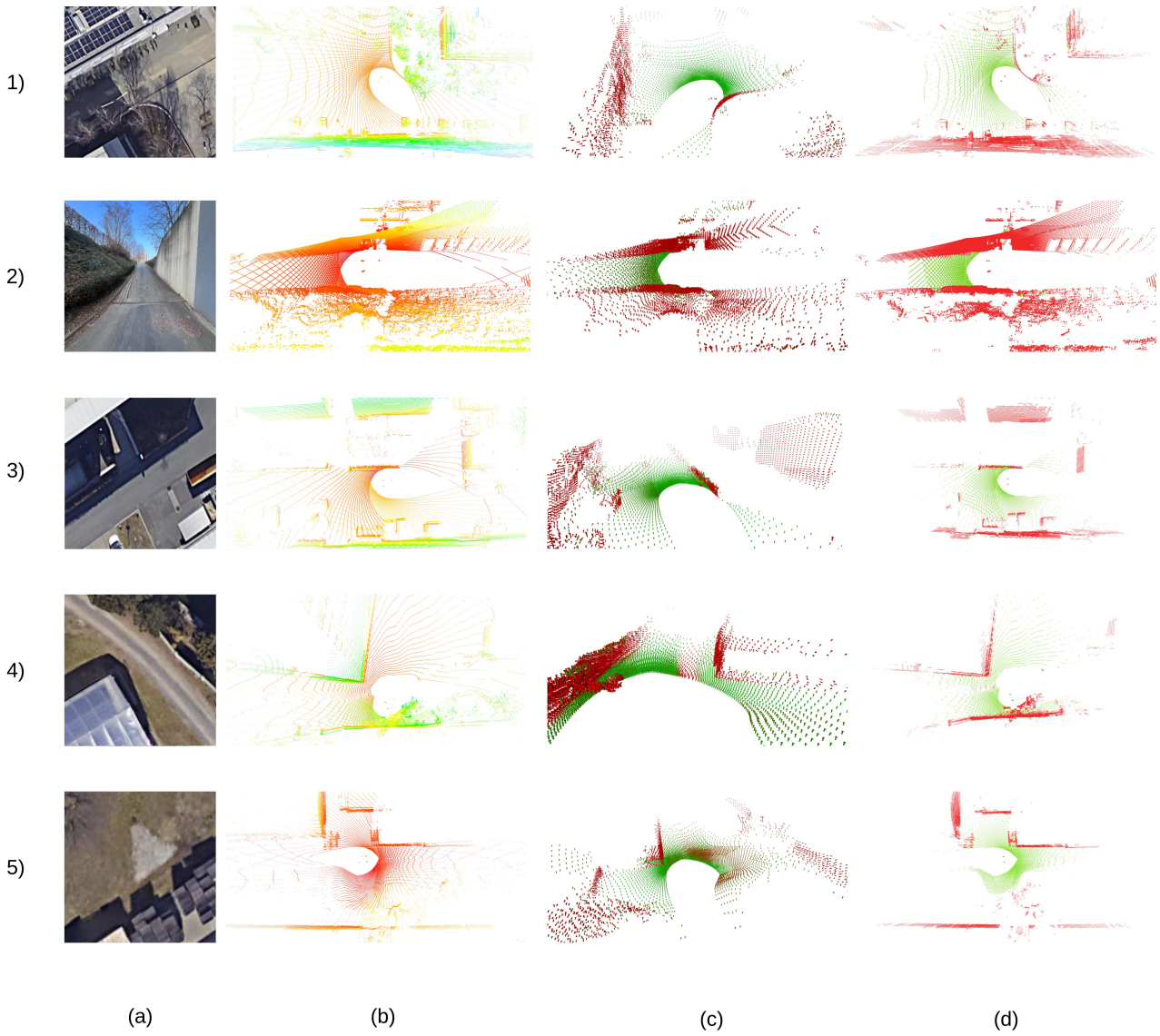

Fig. 3. Illustration of the ground segmentation process. (a) environment of the scenario. (b) raw LiDAR point cloud with colors representing point height. (c) extracted normal vectors and classification results, distinguishing segmented ground (green) and non-ground (red) regions. (d) final segmented point cloud, where green points are labeled as ground and red points as non-ground. Scenario 1 and 3 feature flat ground, scenario 2 includes a slope in front of the robot, scenario 4 depicts a road with a ditch on the right side, and scenario 5 represents an off-road area with varying slopes.

all predicted positive instances.

$$\text{Precision} = \frac{N_{\text{TP}}}{N_{\text{TP}} + N_{\text{FP}}} \tag{11}$$

- **Recall** (Sensitivity or True Positive Rate): measures the proportion of correctly identified positive instances out of all actual positive instances.

$$\text{Recall} = \frac{N_{\text{TP}}}{N_{\text{TP}} + N_{\text{FN}}} \tag{12}$$

- **F1 Score**: the harmonic mean of Precision and Recall, providing a balanced measure of model performance, especially in cases of class imbalance.

$$F_1 = \frac{2 \cdot N_{\text{TP}}}{2 \cdot N_{\text{TP}} + N_{\text{FP}} + N_{\text{FN}}} \tag{13}$$

- **Accuracy**: measures the overall proportion of correctly classified instances out of all instances.

$$\text{Accuracy} = \frac{N_{\text{TP}} + N_{\text{TN}}}{N_{\text{TP}} + N_{\text{TN}} + N_{\text{FP}} + N_{\text{FN}}} \tag{14}$$

- **Coverage**: measures the proportion of labeled points among all data points.

$$\text{Coverage} = \frac{N_{\text{TP}} + N_{\text{TN}} + N_{\text{FP}} + N_{\text{FN}}}{N_{\text{total}}} \tag{15}$$

A high Precision indicates fewer false positives, while a high Recall suggests fewer false negatives. The F1 Score provides a trade-off between these two metrics, and Accuracy gives an overall measure of classification correctness. Coverage ensures an assessment of how much of the dataset

https://doi.org/10.34749/3061-0710.2025.16

is labeled. This measure also depends on the sensor setup; for instance, in our setup, since the sensor is installed tilted, as we move farther from the sensor, the density of points in an area becomes lower and lower, making it difficult to calculate normal vectors. These metrics collectively provide a comprehensive evaluation of the segmentation performance.

B. Performance Evaluation

One of the key aspects of evaluating our method is the coverage of labeled points, which directly impacts segmentation accuracy. Labeling a point requires the assignment of a normal vector, but in some cases, this is not feasible. For instance, points that are farther from the LiDAR sensor tend to have larger spatial gaps, making it challenging to compute a reliable normal vector. Additionally, points in high-variance regions, such as those affected by vegetation or irregular surfaces, may lack sufficient neighboring points to form a well-defined surface. In such cases, points remain unlabeled due to insufficient data.

The results indicate that scenarios with a higher presence of trees, such as Scenario 1, tend to have a lower coverage value, as a greater proportion of points do not belong to distinct, continuous surfaces. Conversely, environments with fewer trees lead to higher coverage. On average, across the five evaluated scenarios, our method achieves a coverage value of 0.897, as detailed in Table I. PatchWork shows better coverage since it uses the entire point cloud for labeling rather than processing each LiDAR separately. It also performs better in scenarios with fewer slopes. Overall, while PatchWork achieves higher coverage, our method demonstrates better average performance across the five scenarios.

Scenario	Coverage		Accuracy		Precision		Recall		F1 Score	
	Ours	Patchwork	Ours	Patchwork	Ours	Patchwork	Ours	Patchwork	Ours	Patchwork
1	0.749	**0.972**	0.84	**0.912**	0.828	**0.898**	0.843	**0.914**	0.834	**0.904**
2	0.907	**0.961**	**0.971**	0.925	**0.892**	0.777	**0.971**	0.92	**0.930**	0.820
3	0.920	**0.981**	0.952	**0.971**	0.953	**0.972**	0.951	**0.970**	0.952	**0.971**
4	**0.962**	0.926	**0.970**	0.921	**0.955**	0.907	**0.970**	0.921	**0.962**	0.913
5	0.949	**0.956**	**0.961**	0.951	**0.948**	0.938	**0.960**	0.950	**0.955**	0.945
Avg	0.897	**0.959**	**0.939**	0.936	**0.915**	0.898	**0.939**	0.935	**0.927**	0.911

TABLE I

PERFORMANCE METRICS COMPARISON: OURS VS. PATCHWORK [12]

The segmentation method demonstrates high accuracy across different environments, as indicated by the average accuracy of 0.939 and an F1-score of 0.927. These values suggest that the approach consistently distinguishes ground points from non-ground points with minimal misclassifications. The average precision of 0.915 indicates that false positives were minimized, meaning non-ground points were rarely misclassified as ground, while the Recall (0.939 avg.) confirms that most ground points were correctly identified.

V. CONCLUSION AND FUTURE WORK

In this paper, we presented a LiDAR-based ground segmentation method that efficiently preserves spatial structure by using an organized point cloud, making it adaptable to different sensor configurations. By directly accessing neighboring points, we extract normal vectors and classify them as ground or obstacles. Furthermore, we assign points to normal vectors using a likelihood-based approach. Experimental evaluations across five diverse scenarios showed an average accuracy of 0.939 and an F1-score of 0.927, demonstrating its reliability in distinguishing ground from non-ground points. The method also adapts well to unstructured terrains and multi-LiDAR configurations, proving useful for real-world robotic navigation.

One limitation of our work is that we process each point cloud separately and do not consider the overlap between point clouds. This overlap can be addressed in future work by incorporating LiDAR transformations relative to each other. By doing so, we can directly access local neighboring points from multiple LiDARs, improving segmentation accuracy and consistency.

REFERENCES

[1] E. E. Aksoy, S. Baci, and S. Cavdar, "Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 926–932.

[2] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, p. 509–517, Sept. 1975. [Online]. Available: https://doi.org/10.1145/361002.361007

[3] T. Cortinhal, G. Tzelepis, and E. E. Aksoy, "Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving," 2020.

[4] N. Diaz, O. Gallo, J. Caceres, and H. Porras, "Real-time ground filtering algorithm of cloud points acquired using terrestrial laser scanner (tls)," *International Journal of Applied Earth Observation and Geoinformation*, vol. 105, p. 102629, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0303243421003366

[5] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[6] A. Golovinskiy and T. Funkhouser, "Min-cut based segmentation of point clouds," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, 2009, pp. 39–46.

[7] A. Grover and M. H. Ashraf, "Leveraging autonomous mobile robots for industry 4.0 warehouses: A multiple case study analysis," *The International Journal of Logistics Management*, vol. 35, 07 2023.

[8] M. Himmelsbach, F. v. Hundelshausen, and H.-J. Wuensche, "Fast segmentation of 3d point clouds for ground vehicles," in *2010 IEEE Intelligent Vehicles Symposium*, 2010, pp. 560–565.

[9] W. Huang, H. Liang, L. Lin, Z. Wang, S. Wang, B. Yu, and R. Niu, "A fast point cloud ground segmentation approach based on coarse-to-fine markov random field," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7841–7854, 2022.

[10] R. Keith and H. M. La, "Review of autonomous mobile robots for the warehouse environment," *arXiv*, vol. 2406.08333, 2024. [Online]. Available: https://arxiv.org/abs/2406.08333

[11] I. Kubasáková, J. Kubáňová, D. Benčo, and N. Fábryová, "Application of autonomous mobile robot as a substitute for human factor in order to increase efficiency and safety in a company," *Applied Sciences*, vol. 14, no. 13, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/13/5859

[12] H. Lim and J. Kim, "Patchwork: Robust ground segmentation in point clouds for autonomous vehicles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 932–939. [Online]. Available: https://doi.org/10.1109/ICRA2021.9561584

[13] C. Sager, P. Zschech, and N. Kühl, "labelcloud: A lightweight domain-independent labeling tool for 3d object detection in point clouds," 2021.

[14] D. Zermas, I. Izzat, and N. Papanikolopoulos, "Cnn for very fast ground segmentation in velodyne lidar data," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2931–2938, 2018.

[15] X. Zhao and C. Wu, "Markov random field for ground segmentation in 3d lidar data," *Remote Sensing*, vol. 2, no. 3, pp. 833–852, 2010. [Online]. Available: https://www.mdpi.com/2072-4292/2/3/833