Patchwork++: Fast and Robust Ground Segmentation Solving Partial Under-Segmentation Using 3D Point Cloud

Seungjae Lee[1,*], *Student Member, IEEE*, Hyungtae Lim[1,*], *Student Member, IEEE*,
and Hyun Myung[1,†], *Senior Member, IEEE*

Abstract— In the field of 3D perception using 3D LiDAR sensors, ground segmentation is an essential task for various purposes, such as traversable area detection and object recognition. Under these circumstances, several ground segmentation methods have been proposed. However, some limitations are still encountered. First, some ground segmentation methods require fine-tuning of parameters depending on the surroundings, which is excessively laborious and time-consuming. Moreover, even if the parameters are well adjusted, a partial under-segmentation problem can still emerge, which implies ground segmentation failures in some regions. Finally, ground segmentation methods typically fail to estimate an appropriate ground plane when the ground is above another structure, such as a retaining wall. To address these problems, we propose a robust ground segmentation method called *Patchwork++*, an extension of Patchwork. Patchwork++ exploits adaptive ground likelihood estimation (A-GLE) to calculate appropriate parameters adaptively based on the previous ground segmentation results. Moreover, temporal ground revert (TGR) alleviates a partial under-segmentation problem by using the temporary ground property. Also, region-wise vertical plane fitting (R-VPF) is introduced to segment the ground plane properly even if the ground is elevated with different layers. Finally, we present reflected noise removal (RNR) to eliminate virtual noise points efficiently based on the 3D LiDAR reflection model. We demonstrate the qualitative and quantitative evaluations using a SemanticKITTI dataset. Our code is available at `https://github.com/url-kaist/patchwork-plusplus`

I. INTRODUCTION

Recently, mobile robots are increasingly being employed to perform specific tasks in diverse and complex environments. Accordingly, to accomplish given tasks in various environments, several studies have improved the perception of the surrounding environments using a 3D light detection and ranging (LiDAR) sensor [1], [2]. Because most 3D LiDAR sensors are omnidirectional sensors that can accurately measure distances up to approximately 100 m, they are widely used for various purposes such as detection [3], segmentation [4], localization [5], and 3D registration [6]. Among the various algorithms in 3D perception via 3D LiDAR sensors, ground segmentation is one of the essential tasks for 3D perception. For example, ground segmentation

*Both authors have equally contributed.

[1]Seungjae Lee, [1]Hyungtae Lim, and [1]Hyun Myung are with the School of Electrical Engineering at KAIST (Korea Advanced Institute of Science and Technology), Daejeon, 34141, Republic of Korea. {sj98lee, shapelim, hmyung}@kaist.ac.kr

† Corresponding author: Hyun Myung

(a) 3D Point Cloud   (b) RANSAC [9]
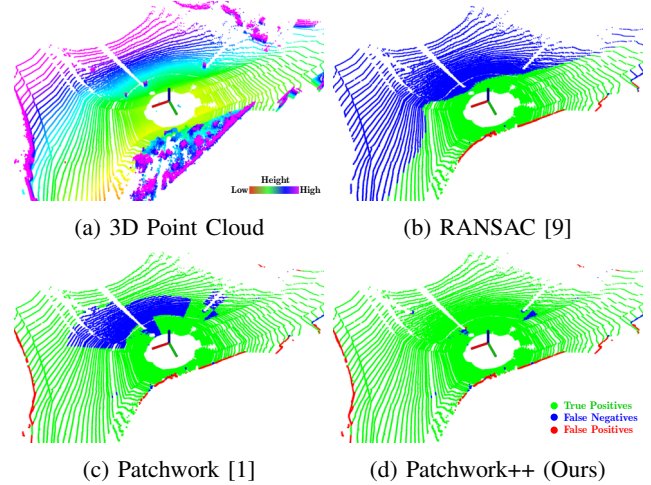
(c) Patchwork [1]   (d) Patchwork++ (Ours)

Fig. 1. Visual descriptions of our proposed method and generic problems on ground segmentation. (a) In the urban environment, the ground may not be flat, so (b) ground segmentation using a single planar model sometimes leads to lots of false negatives called under-segmentation. (c) Patchwork [1], our previous work, handles under-segmentation, yet it sometimes fails partially. This is because its parameters are deterministic; hence, these user-defined parameters may filter out the solid ground. (d) Unlike Patchwork, Patchwork++ updates its parameters based on previous ground segmentation; therefore, Patchwork++ successfully performs ground segmentation in uneven surroundings. The red, green, and blue axes represent x, y, and z coordinates, respectively.

can be employed in traversable area detection [7] and object clustering [8].

As machine learning has been applied to diverse research areas, various learning-based 3D perception methods have been proposed, exhibiting outstanding performance [10], [11]. But, unfortunately, there are two inherent potential limitations. First, learning-based methods require point-wise labeling by humans, which is both time-consuming and severely laborious. Second, the performance of learning-based ground segmentation can potentially be degraded when employed in scenes different from the training dataset or different sensor configurations. Consequently, in this study, we focus on ground segmentation in non-learning-based methods.

In general, ground segmentation functions as a preceding stage because the ground and non-ground points can be utilized for different purposes. For example, non-ground points go through another segmentation stage, which classifies these non-ground points into static and dynamic points. However, because ground points can be assumed as being static, the corresponding segmentation is not performed on the ground points.

Moreover, when ground segmentation is exploited as a preprocessing step, it should satisfy the following three requirements. First, ground segmentation should be sufficiently fast to be employed as a preliminary process for other algorithms such as object clustering. Second, it should ensure little performance perturbation to simultaneously guarantee both precision and recall performance. Third, it should be capable of dealing with uneven outdoor environments.

However, several ground segmentation methods fail to satisfy these requirements and even exhibit the under-segmentation problem, as represented in Fig. 1(b). Although our previous work, Patchwork [1], meets these requirements, it occasionally exhibits a *partial under-segmentation problem*, as represented in Fig. 1(c). These under-segmentation issues usually occur because of two potential reasons. First, some ground segmentation methods are deterministic. Second, there are some situations in which the assumption that the points with low z values are most likely to be considered as the ground points does not hold.

Therefore, *Patchwork++*, an advanced version of Patchwork, is proposed to tackle these problems. Accordingly, our proposed method becomes more generalized in complex urban environments. Our contributions are threefold as follows:

- First, adaptive ground likelihood estimation (A-GLE) and temporal ground revert (TGR) are proposed, which provide even fewer false negatives than Patchwork based on the coarse-to-fine strategy that addresses the partial under-segmentation issue.
- Second, two novel outlier rejection modules, i.e., reflected noise removal (RNR) and region-wise vertical plane fitting (R-VPF), are proposed to prevent the situations where our assumption does not hold by rejecting noises or non-ground points.
- Third, our proposed algorithm was evaluated with the SemanticKITTI dataset [12] and experimental evidence corroborates Patchwork++ exhibits promising performance compared with the state-of-the-art methods including Patchwork and shows the fastest speed as well.

II. Related Works

A. Learning-based Ground Segmentation Methods

Xu *et al.* [10] proposed a novel range-point-voxel fusion network, RPVNet, which ranks first on the SemanticKITTI leaderboard of semantic segmentation. RPVNet appropriately combines three different features from range image, point, and voxel to classify each point. Shen *et al.* [13] addressed the problem of the capability of segmentation on rough terrains and the efficiency of processing time by introducing the Jump-Convolution-Process, which facilitates the point cloud segmentation by solving the 2D smoothing problem. Paigwar *et al.* [11] proposed GndNet, which estimates the grid-wise ground elevation and segments ground and non-ground points using PointNet [4] to encode a pillar feature in each grid. As mentioned in Section I, there are several potential limitations in that these methods require human-made labeling by hands and the performance can be degraded when it comes to unforeseen situations that are not included in the training data.

B. Conventional Ground Segmentation Methods

Moosmann *et al.* [14] proposed a 3D point cloud segmentation method based on the idea of *local convexity* yielding a fast ground segmentation in a non-flat urban environment. Himmelsbach *et al.* [15] also proposed a line-based ground segmentation method providing a significantly high speed. Zermas *et al.* [16] proposed a fast ground plane fitting (GPF), which utilizes an initial seed extraction suitable for the ground plane estimation based on principal component analysis (PCA). Unlike other ground segmentation methods, such as RANSAC [9], this approach circumvents redundant random iterations, which are severely time-consuming. As an extension of GPF, Lim *et al.* [1] proposed a concentric zone-based region-wise ground segmentation with the GLE. Narksri *et al.* [17] proposed a RANSAC-based ground segmentation with multiple regions and addressed the problem posed by sloped terrain. Jiménez *et al.* [18] addressed the issue of local geometric variation of the ground by adopting the channel-based Markov random field.

Although various conventional ground segmentation methods have been proposed, severe problems remain inherent in these methods. In particular, under-segmentation is the representative problem that results in lots of false negatives, as represented in Fig. 1. For example, single planar model-based methods usually fail to estimate the ground plane with a slope because the single plane assumption is insufficient to model wide areas with partially bumpy regions as a single plane. Although Narksri *et al.* [17] addressed this problem by employing a RANSAC-based ground plane fitting in multiple regions, it still exhibits the under-segmentation problem because of its limitation in dealing with bumpy grounds.

C. Applications of Ground Segmentation

As mentioned in Section I, various object clustering methods [18]–[21] include ground segmentation as a prior step for object recognition. Because ground points are not the region of interest in object clustering methods, the elimination of ground points enhances both computational efficiency and accuracy. That is, approximately half of the points in a 3D LiDAR scan are given as ground points in outdoor environments. This implies that the overall algorithm can be faster, as the size of the input point cloud will be half of the original via ground removal.

In addition, ground segmentation is also employed in LiDAR odometry methods to obtain strong constraints using ground points as features [22]–[24]. For instance, Shan *et al.* [22] proposed LeGO-LOAM, a lightweight and ground-optimized LiDAR SLAM. According to the research, a few planar features are extracted from the ground to estimate the transformation in terms of z (height direction), roll angle, and pitch angle. Furthermore, Pan *et al.* [23] proposed MULLS, a versatile LiDAR SLAM method using the multi-metric linear least square. That is, MULLS separates points into several classes based on their geometric features to construct the
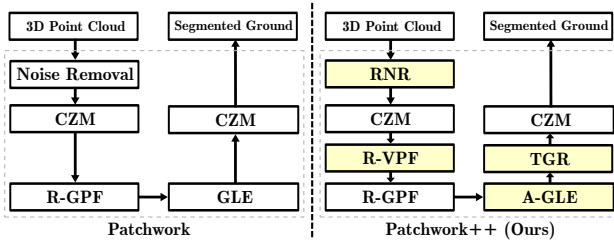
Fig. 2. (L-R) Flow charts of Patchwork [1] and Patchwork++. Unlike Patchwork, Patchwork++ consists of novel modules called Reflected Noise Removal (RNR), Region-wise Vertical Plane Fitting (R-VPF), Adaptive GLE (A-GLE), and Temporal Ground Revert (TGR). CZM and R-GPF represent the concentric zone model and region-wise ground plane fitting, respectively, which were proposed in the previous work [1].

multi-metric optimization problem. Certainly, the ground is also considered as one of the several types.

III. PATCHWORK++: FAST, ROBUST, AND ADAPTIVE GROUND SEGMENTATION

The following paragraphs highlight the problem definition and rationale behind each module of Patchwork++. Unlike Patchwork, the following four parts are proposed: RNR, R-VPF, A-GLE, and TGR, which are presented in Fig. 2.

A. Problem Definition

Given a 3D point cloud, P, which represents the set of measured points, p_i, and presents a snapshot of the surroundings, we attempt to classify every p_i into the ground, including road, terrain, and sidewalks, or non-ground class, which includes cars, buildings, humans, etc. Then, P can be represented as a union of two distinct sets: a set of ground points, G, and non-ground points, N. In contrast, a ground segmentation method outputs the estimated ground points, $\hat{G}$, while the remaining points are regarded as a set of non-ground points, $\hat{N}$. Therefore, P can be expressed as follows:

$$P = \bigcup_i p_i = G \cup N = \hat{G} \cup \hat{N}. \tag{1}$$

In this case, $\hat{G}$ and $\hat{N}$ can be divided into actual ground points and actual non-ground points, respectively. Specifically, $\hat{G} \cap G$, $\hat{G} \cap N$, $\hat{N} \cap G$, and $\hat{N} \cap N$ represent the set of true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN), respectively. In conclusion, the objective of this study is to estimate $\hat{G}$ that contains TP points as many as possible while rejecting FN and FP points as many as possible.

B. RNR: Reflected Noise Removal

For a given raw point cloud, some noise points are occasionally observed under the actual ground, as illustrated in Fig. 3. Unfortunately, R-GPF selects initial seeds based on the assumption that the points at the lowest part in the bin are ground [16]; hence, these noises can be regarded as initial seeds that trigger the under-segmentation problem in the corresponding bin.

To address this problem, these points are usually filtered out by rejecting points whose z value is smaller than a user-defined threshold, $z_{\min}$ [1], [2]. This filtering assumes that

the ground of surroundings is sufficiently flat. Unfortunately, this noise removal method also triggers a severe problem that erases a few true ground points undesirably in steep slope environments, as illustrated in Figs. 4(b) and (c). Therefore, it is necessary to discriminate between the actual ground points and noise points.

To this end, RNR is proposed by leveraging two key observations based on the research by Zhao *et al.* [25]. First, the virtual noise points can be obtained by the reflection on the reflective surfaces, i.e., bonnets and roofs of vehicles, or glass. This fact lets the noise points be located along the direction from the sensor origin onto the point of incidence, as shown in Fig. 3. It is noticeable that the smaller the incident angle, the lower the z value of the noise tends to occur. For this reason, it can be concluded that critical noises that hinder our assumption are highly likely to be originated from the rays from the bottom rings because the ray directions of the bottom rings tend to have relatively small incident angles.

Second, these noise points have relatively small intensities in general because the rays which correspond to the noise points undergo more reflections than the usual rays until they return to the sensor. For example, the ray of the virtual point takes additional reflection on the point of incidence, as illustrated in Fig. 3. Consequently, this additional reflection triggers the decrease of the intensity of the virtual point.

Therefore, firstly, RNR only checks a certain number of bottom rings, N_{noise}, to remove these noise points. Accordingly, RNR rejects the points that are located in these rings with a lower height than the noise removal height threshold, h_{noise}, and lower intensity than the noise removal intensity threshold, I_{noise}. Consequently, RNR efficiently removes noise points with unrealistically large negative z values with a negligible loss of actual ground points. In particular, h_{noise} is self-updated by A-GLE (see Section III.D).

C. R-VPF: Region-wise Vertical Plane Fitting

However, R-GPF still sometimes fails in estimating an appropriate ground plane in another situation where the desired ground is above vertical urban structures. This is because the points from the vertical structures are chosen as the initial seeds due to their smaller z values compared with actual ground points. For example, when retaining walls are located under the ground in the bin and close enough to the sensor, R-GPF selects the points from the wall as the initial seeds of the ground. Then, these points induce an unsuitable
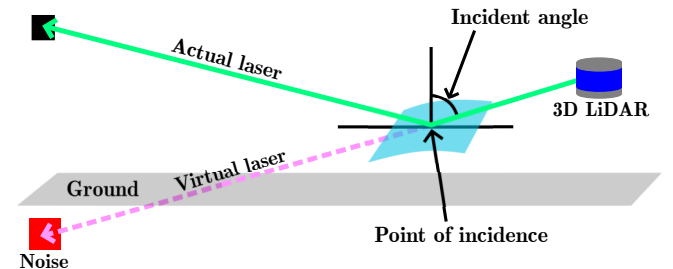

Fig. 3. Illustration of reflected noises in a 3D point cloud. Noise (red square) can be located below the ground due to the reflection of bottom lasers.
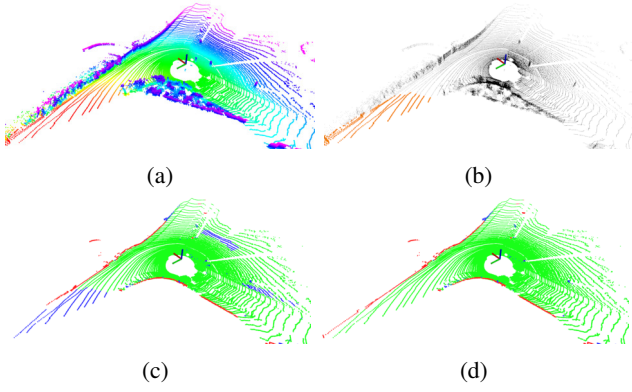
Fig. 4. (a) A 3D point cloud visualized by an elevation map, where red points represent the bottom part of the downhill. (b) Height threshold-based noise filtering triggers several false negatives because z values of some ground points are lower than $z_{\min}$ (orange parts). (c)–(d) Before and after the application of our RNR: (c) In Patchwork [1], some ground points are under-segmented (blue), (d) whereas our proposed method successfully identifies the points from downhill as ground points (best viewed in color).

ground fitting result with the ground plane points on them because PCA-based estimation is significantly sensitive to the outlier points, as illustrated in Fig. 5.

One may argue that this elevated ground should be considered as non-ground because this ground is distinguishable from another ground and the points from this ground have larger z values than points from the usual ground. However, some dynamic objects, such as humans, can still stand on that ground; thus, these points are considered actual ground points rather than non-ground points in this study.

To this end, R-VPF is proposed, allowing R-GPF to estimate more accurate ground planes by rejecting dominant and vertical points for preprocessing within a bin. Specifically, let P_n be the points in the n-th bin, which is divided based on the concentric zone model (CZM) [1]. That is, CZM divides a 3D point cloud into multiple subsets in the polar coordinate and bin sizes are different depending on which zone the bins belong to. Then, the estimated vertical plane points, $\hat{V}_n$, are determined via the following four steps.

First, R-VPF selects some points starting from the lowest point, as seed points in every k-th iteration. Second, R-VPF estimates the mean of seed points, $\mathbf{m}_n^k \in \mathbb{R}^3$, and the unit normal vector, $\mathbf{v}_{3,n}^k$, which corresponds to the eigenvector with the smallest eigenvalue from the PCA of seed points. Subsequently, the k-th potential vertical plane points, $\hat{W}_n^k$, are sampled among the candidate points, $\hat{P}_n^k$, as follows:

$$\hat{W}_n^k = \left\{ \mathbf{p} \in \hat{P}_n^k \;\middle|\; |(\mathbf{p} - \mathbf{m}_n^k) \cdot \mathbf{v}_{3,n}^k| < d_v \right\}, \quad (2)$$

where $\hat{P}_n^k \leftarrow P_n$ when $k = 1$, $\hat{P}_n^k \leftarrow P_n - \bigcup_{i=1}^{k-1} \hat{V}_n^i$, otherwise, and d_v denotes the distance margin for the estimated vertical plane. This implies that the points considered as vertical points, $\hat{V}_n^i$, in previous iterations are rejected in subsequent iterations.

Then, $\hat{V}_n^k$, which denotes the vertical points at the k-th iteration, is determined based on the following condition:

$$\hat{V}_n^k = \begin{cases} \hat{W}_n^k, & \text{if } \frac{\pi}{2} - \cos^{-1}\left(\mathbf{v}_{3,n}^k \cdot \mathbf{u}_z\right) < \theta_v \\ \varnothing, & \text{otherwise} \end{cases}, \quad (3)$$

where $\mathbf{u}_z$ denotes the unit normal vector in the z-axis direction, i.e., $[0\ 0\ 1]^\mathsf{T}$ and θ_v is the margin for the unit normal vector of the estimated plane.

Finally, the total vertical points, $\hat{V}_n$, are estimated by accumulating all the vertical points as follows:

$$\hat{V}_n = \bigcup_{k=1}^{K_v} \hat{V}_n^k, \quad (4)$$

where K_v denotes the number of iterations for R-VPF.

D. A-GLE: Adaptive Ground Likelihood Estimation

In our previous work [1], GLE, $f(\mathcal{X}_n|\theta_n)$, was employed to determine whether the region-wise estimated ground planes are ground or non-ground. For this process, GLE is determined by *uprightness*, *elevation*, and *flatness* indicator functions with $\mathbf{v}_{3,n}$, $\bar{z}_n$, and σ_n as variables, respectively. In detail, $\mathbf{v}_{3,n}$ is the eigenvector with the smallest eigenvalue from the PCA of seed points, $\bar{z}_n$ is the mean of seed points, and σ_n is the local surface variation [26], $\frac{\lambda_{3,n}}{\lambda_{1,n}+\lambda_{2,n}+\lambda_{3,n}}$, where $\lambda_{1,n}$, $\lambda_{2,n}$, and $\lambda_{3,n}$ are the eigenvalues from the PCA of given points in the descending order.

Accordingly, it is necessary to set the corresponding parameters $(\theta_\tau, \kappa(r_n), \sigma_{\tau,m})$ as the thresholds of uprightness, elevation, and flatness indicator functions, respectively. In this paper, $\bar{z}_n$, σ_n, $\kappa(r_n)$, and $\sigma_{\tau,m}$ are represented as e_n, f_n, $e_{\tau,m}$, and $f_{\tau,m}$ for simplicity. However, in our previous work, setting the optimal parameters, especially $e_{\tau,m}$ and $f_{\tau,m}$, depending on the surroundings is severely time-consuming. For example, in Fig. 6, the distributions of elevation and flatness values for true ground planes seem to be varied depending on the surroundings. Hence, this confirms that the adjustment of parameters is necessary for each environment.

To tackle this problem, A-GLE updates $e_{\tau,m}$ and $f_{\tau,m}$ in an adaptive manner based on the previously obtained estimation results. Unless A-GLE is initialized with abnormal values of $e_{\tau,m}$ and $f_{\tau,m}$, A-GLE can update these parameters properly. In addition, as mentioned in Section III.*B*, the noise removal height threshold, h_{noise}, is updated by A-GLE.
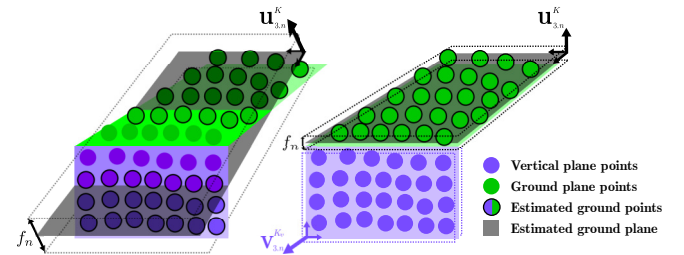

Fig. 5. (L-R): Visual descriptions of the estimated ground plane without R-VPF and with R-VPF. Once R-VPF is applied for preprocessing, R-GPF successfully estimates an accurate ground plane even though true ground points are located on the non-ground points. In addition, f_n, which is the flatness variable [1] of the estimated plane, becomes smaller. Accordingly, the estimated ground can be distinguishable as a ground owing to its small f_n.
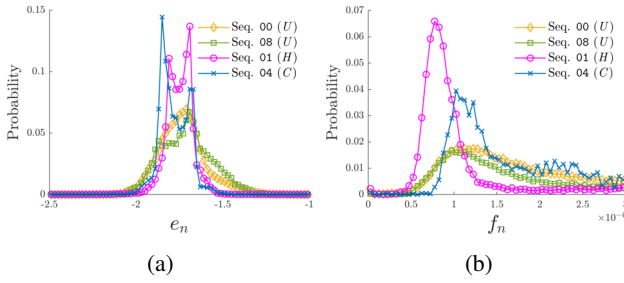
Fig. 6. Probability distributions of true ground points' (a) e_n and (b) f_n at the first ring of the concentric zone model in Sequence `00`, `01`, `04`, and `08`. Sequence `00` and `08` represent similar distributions. In contrast, Sequence `01` and `04` result in quite different distributions from each other. *U*, *H*, and *C* denote urban, highway, and country scenes, respectively.

Elevation　For the self-update of parameters, we propose a novel concept of *definite ground*, D_m, which represents the previously estimated ground planes satisfying uprightness and elevation conditions in the m-th ring of the concentric zone model. Accordingly, A-GLE adopts the properties of definite ground planes to update the elevation parameter for the next estimation.

Let E_m be the set of all e_n values of estimated ground planes in D_m. Then, based on E_m, $e_{\tau,m}$ is newly updated as follows:

$$e_{\tau,m} \leftarrow \text{mean}(E_m) + a_m \cdot \text{stdev}(E_m), \tag{5}$$

where $\text{mean}(\cdot)$ and $\text{stdev}(\cdot)$ denote the mean and standard deviation of a given set, respectively. $a_m > 0$ denotes the constant gain of the standard deviation term in the m-th ring of the concentric zone model for elevation.

Flatness　Before updating the flatness parameter $f_{\tau,m}$, it is found that a potential problem exists in its calculation in our previous work. Specifically, as mentioned above, f_n was calculated as $\frac{\lambda_{3,n}}{\lambda_{1,n}+\lambda_{2,n}+\lambda_{3,n}}$. To obtain a meaningful value from this approach, the value must be consistent for the same ground regardless of the ground's relative LiDAR position.

Unfortunately, because Patchwork leverages the concentric zone model, the ground is divided into several bins with the sensor origin as the center. Accordingly, the shape and size of the ground can be differentiated by the approach taken to divide the ground. Then, $\lambda_{1,n}$ and $\lambda_{2,n}$ can be changed even though the overall ground does not change. Therefore, one can argue that f_n calculated accordingly is illogical based on its potential inconsistency. To address this problem, Patchwork++ sets $f_n = \lambda_{3,n}$, which is a variation in the surface normal direction.

Similar to $e_{\tau,m}$, $f_{\tau,m}$ has been updated based on the distribution of D_m by A-GLE. Let F_m be the set of all f_n for the estimated ground planes in D_m. Then, $f_{\tau,m}$ can be updated as follows:

$$f_{\tau,m} \leftarrow \text{mean}(F_m) + b_m \cdot \text{stdev}(F_m), \tag{6}$$

where $b_m > 0$ denotes the constant gain of the standard deviation term in the m-th ring of the concentric zone model for flatness.

Noise Removal Height　A-GLE updates h_{noise} based on the mean of the elevation values of the estimated ground planes

at the closest ring of the concentric zone model as follows:

$$h_{\text{noise}} \leftarrow \text{mean}(E_1) + \delta, \tag{7}$$

where $\delta < 0$ means the distance margin of the noise removal.

For better understanding, Fig. 7 illustrates the flow in A-GLE. A-GLE stores the states from the previously estimated ground. Then, h_{noise}, $e_{\tau,m}$ and $f_{\tau,m}$ are updated for the next estimation based on the stored database.

E. TGR: Temporal Ground Revert

Through A-GLE, the estimation of typical grounds is successfully performed with a self-update of parameters. However, some unusual ground points within a bin do not satisfy the parameters' condition. For example, rough terrains with grass sometimes have higher e_n and f_n than the self-updated parameters. This is because A-GLE updates its parameters based on all values over time. Accordingly, this makes A-GLE play a role as a low-pass filter, which makes it difficult to estimate the bin as ground when f_n becomes temporarily large.

To address this problem, TGR is proposed to revert the under-segmented ground planes into the segmented ground bins using the D_m^t, which are definite ground planes only at time t. In other words, TGR compares each under-segmented bin's f_n with $f_{\tau,m}^t$, which can be calculated as follows:

$$f_{\tau,m}^t = \text{mean}(F_m^t) + c_m \cdot \text{stdev}(F_m^t), \tag{8}$$

where c_m denotes a constant gain of the standard deviation term for the m-th ring of the concentric zone model and F_m^t is the set of f_n in D_m^t. Therefore, the rejected ground points whose f_n is larger than $f_{\tau,m}^t$ can be reverted if $f_n < f_{\tau,m}^t$ is satisfied (see Section V.*D*).
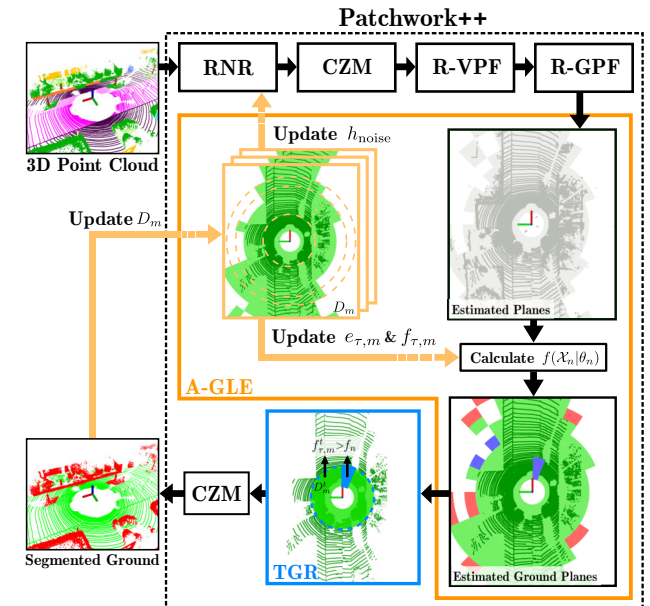

Fig. 7. Visual description of A-GLE and TGR. A-GLE stores the variables of the estimated ground planes, i.e., e_n and f_n, and updates its parameters based on the previous estimation results. TGR double-checks the under-segmented grounds based on the variables of the estimated ground planes at time t.
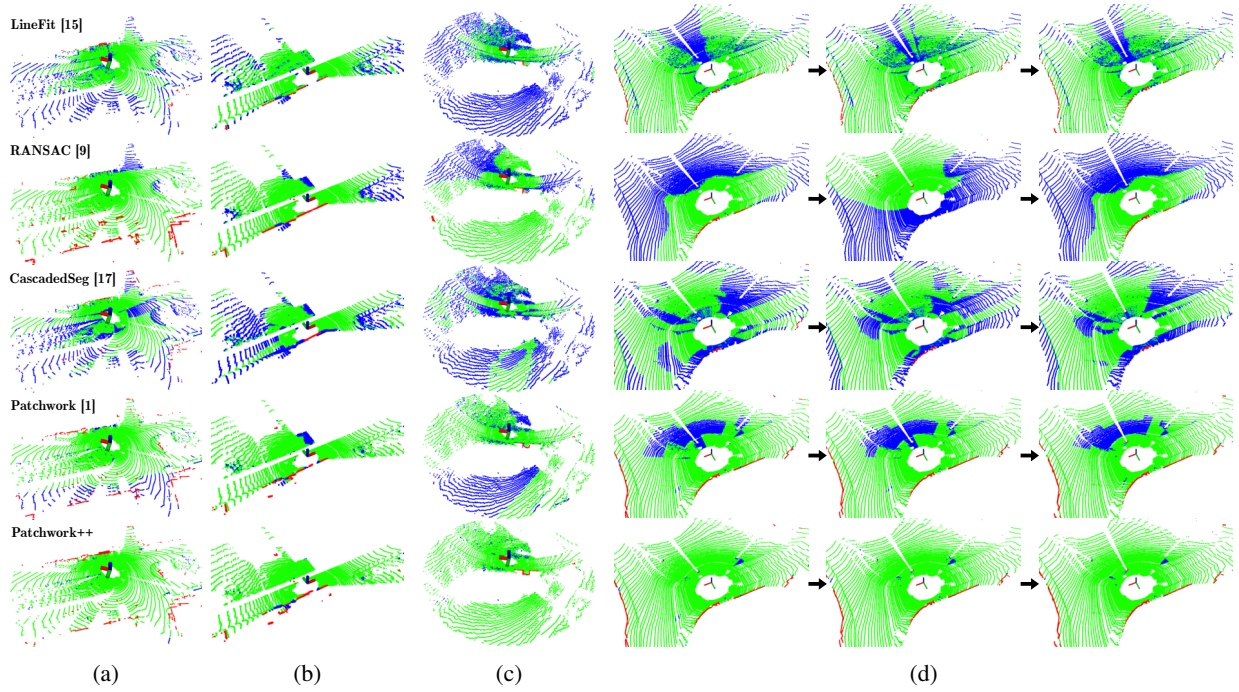
Fig. 8. Qualitative comparison between the proposed method, Patchwork++, and the state-of-the-art methods at (a) Sequence `06` around frame 718, (b) Sequence `03` around frame 504, (c) Sequence `09` around frame 250, and (d) Sequence `02` around frame 3,300. Patchwork++ successfully addresses the partial under-segmentation problem. Green, red, and blue points represent TP, FP, and FN, respectively. The fewer the red and blue points, and the more the green points, the better (best viewed in color).

IV. EXPERIMENTAL SETUP

A. Dataset and Error Metrics

In our experiments, the SemanticKITTI dataset [12] was adopted to compare the ground segmentation performance of our proposed method to that of other ground segmentation algorithms. Based on the point-wise labels, the points labeled as `road`, `parking`, `sidewalk`, `other ground`, `lane marking`, and `terrain` are considered ground points. The other points are regarded as non-ground points. However, unlike our previous work [1], the points labeled as `vegetation` are not evaluated as ground nor non-ground exceptionally because it is impractical to regard the `vegetation` as a single ground or non-ground class. Note that this implies that points labeled as `vegetation` are only excluded in the evaluation step; the points are still included in the input point cloud. *Precision*, *Recall*, and F_1-*score* are employed as error metrics as described in [1].

B. Parameters of Patchwork++

We set initial $e_{\tau,m}$ and $f_{\tau,m}$ both 0 for all m ($1 \leq m \leq 4$). The other parameters are identical to our previous work [1]. For RNR, we set $N_{\text{noise}} = 20$ and $I_{\text{noise}} = 0.2$. For R-VPF, we set $d_v = 0.1$, $\theta_v = 0.707$, and $K_v = 3$. For A-GLE, we set $a_m = 1$ for all m, $b_m = 3$ for $m = 1$, $b_m = 2$ otherwise, and $\delta = -0.5$. For TGR, we set $c_m = 1.5$ for all m.

V. RESULTS AND DISCUSSION

A. Comparison with the State-of-the-Art Methods

The overall performance of our proposed method and other ground segmentation methods are summarized in Table I

and Fig. 8. Our proposed algorithm exhibits the highest F_1-score compared with other ground segmentation methods, which include RANSAC [9], LineFit [15], GPF [16], R-GPF [2], and CascadedSeg [17]; our proposed method addresses the inherent problem of the previous noise filtering method (Figs. 8(a) and (c)), the unsuitable definition of f_n in the previous work (Fig. 8(b)), and also the partial under-segmentation issue (Figs. 8(c) and (d)). Consequently, Patchwork++ exhibits both higher precision and recall than Patchwork [1]. Furthermore, the novel Patchwork++ presents a lower standard deviation of recall. These experimental evidences corroborate the position that our proposed method facilitates accurate and robust ground segmentation in complex urban environments.

B. Effect of R-VPF

In Fig. 9, it was demonstrated that R-VPF allows R-GPF to estimate correct ground planes even though the ground points are located on a low fence or flower bed. As mentioned earlier, R-GPF is strongly based on the premise that the bottom points are likely to be ground points; hence, R-GPF

TABLE I. Performance comparison between our proposed method and other ground segmentation methods. Metrics are expressed as (mean)±(stdev).

Algorithm	Precision (%)	Recall (%)	F_1 (%)
LineFit [15]	**98.26±1.35**	87.88±7.94	92.75
RANSAC [9]	89.87±14.16	93.97±13.16	91.83
CascadedSeg [17]	95.25±7.88	74.53±10.78	83.59
GPF [16]	95.78±3.76	83.89±22.42	89.14
R-GPF [2]	74.68±15.7	98.15±**1.47**	84.52
Patchwork [1]	94.23±3.96	97.62±3.42	95.88
Patchwork++ w/o TGR	94.98±3.44	97.64±3.58	96.28
Patchwork++ (Ours)	94.92±3.50	**98.18±2.41**	**96.51**

13281

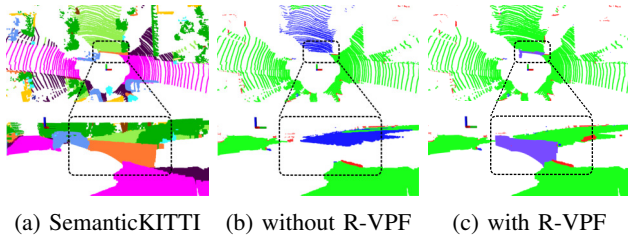(a) SemanticKITTI   (b) without R-VPF   (c) with R-VPF

Fig. 9. (a) A visualized point cloud for Sequence `10` around frame 369 based on the labels in the SemanticKITTI dataset [12]. Note that the orange color denotes the points labeled as a fence. (b)-(c) Before and after the application of R-VPF. (b) The points from the fence impede R-GPF from estimating the correct ground plane, so the points from the terrain are under-segmented (blue). (c) R-VPF tackles this problem by detecting vertical plane points in the preprocessing (purple), which is followed by successful ground segmentation by R-GPF even though ground points are located on the non-ground objects (best viewed in color).

without R-VPF occasionally leads to lots of FN (Fig. 9(b)). In contrast, R-GPF with R-VPF exhibits a successful ground segmentation by extracting points from the fence as non-ground points in advance (Fig. 9(c)).

C. Effect of A-GLE

In Fig. 10, the green points represent D_m and approximately 95.8% of them are true ground based on the SemanticKITTI dataset [12]. As the green points are determined by the self-updated $e_{\tau,m}$, this verifies that the resultant $e_{\tau,m}$ is sufficiently valid to estimate usual grounds and yields the appropriate D_m. Moreover, as 85.3% of the true ground points are included in the green points, it is reasonable to update $f_{\tau,m}$ based on D_m, which classifies the blue and red points in Fig. 10.

Similarly, the resultant $f_{\tau,m}$, which is drawn as a horizontal black-dotted line in Fig. 10, appears to be suited to distinguish blue and red points, which can be assumed as ground and non-ground, respectively. For the remaining estimated planes, which do not belong to D_m, blue points represent the estimated planes where more than half are composed of the actual ground points, while red points
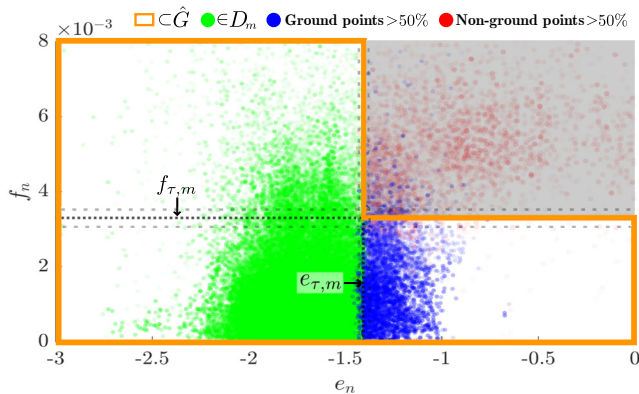

Fig. 10. Results of A-GLE. The black-dotted vertical and horizontal lines represent the final elevation and flatness parameters, respectively. The gray-dashed lines represent the standard deviation of the self-updated parameters. It is verified that the self-updated flatness threshold $f_{\tau,m}$ is sufficiently valid as it classifies blue and red points appropriately. On the other hand, the validity of $e_{\tau,m}$ is ensured because approximately 95.8% of the D_m points are ground points. The opacity of plotted points represents the number of estimated plane points (best viewed in color).
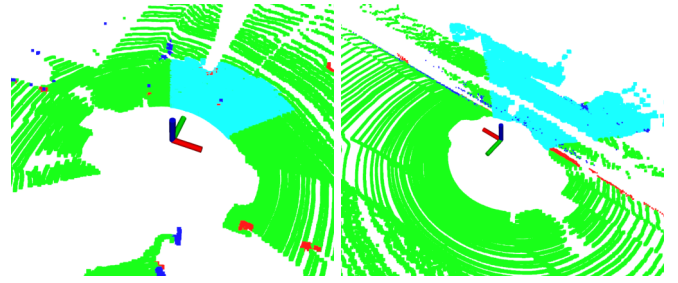

Fig. 11. (L-R) Examples of successful revert of ground points using TGR at Sequence `00` around frame 1,978 and Sequence `09` around frame 1,462. The cyan color denotes the reverted ground points, thus TGR significantly reduces false negatives (best viewed in color).

represent the others. Hence, this verifies that the self-updated $f_{\tau,m}$ is sufficiently valid to distinguish ground planes from non-ground planes.

D. Effect of TGR

Next, experimental evidences support that TGR resolves the partial under-segmentation issue, as shown in Fig. 11. Specifically, even though some bins have larger f_n than $f_{\tau,m}$ in (6), these are double-checked based on the distribution characteristics of the currently estimated ground planes at time t. Therefore, TGR significantly reverts numerous FNs into TPs. Accordingly, TGR increases the recall with a negligible decrease in precision, as presented in Table I.

E. Different Distributions of Self-Updated Parameters Depending on the Surroundings

Interestingly, it is determined that the self-updated parameters have different aspects depending on the surroundings, as presented in Fig. 12. For instance, in a highway scenario, the ground tends to be relatively flat compared with other scenarios, such that $e_{\tau,m}$ and $f_{\tau,m}$ are self-updated to small values. In contrast, country scenarios include relatively more steep roads, bumpy terrain, or sidewalks, such that $e_{\tau,m}$ and $f_{\tau,m}$ are set to higher values than the others. Therefore, it can be inferred that our A-GLE successfully updates the parameters depending on the surroundings, which reduces the effort required to update the parameter setting conducted by humans.
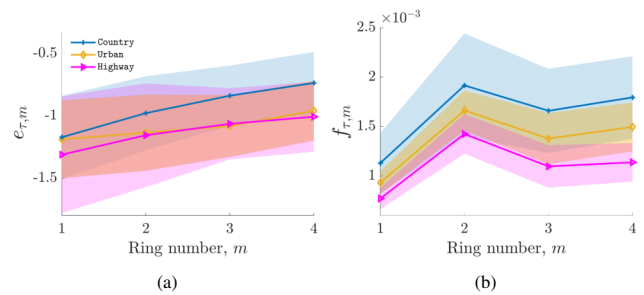

(a)   (b)

Fig. 12. Different distribution aspects of the self-updated (a) $e_{\tau,m}$ and (b) $f_{\tau,m}$ depending on the surroundings. Each colored area represents the standard deviation of self-updated parameters along the running time for the corresponding environment with the same color. `Country` includes Sequence `02`, `03`, `04`, `05`, `09`, and `10`; `Urban` includes Sequence `00`, `06`, `07`, and `08`; `Highway` denotes Sequence `01` in the SemanticKITTI dataset [12].

F. Algorithm Speed

Furthermore, our proposed method exhibits faster ground segmentation than Patchwork [1], which is noted in Table II. This is an implementation-level improvement; originally, Patchwork employs global sorting based on the z values before dividing a point cloud into CZM, which takes $O(N \log N)$ complexity, where N denotes the size of the point cloud. In contrast, the sorting step in the currently proposed method is performed after the division of the point cloud into CZM; thus, its complexity can be approximated by $O(LM \log M)$, where L and M denote the number of bins and the number of points within each bin, respectively. Note that $N \gg L$ and $N \gg M$, and let us assume that $N = LM$ for simplicity. Then $N \log N$ can be expressed as $LM(\log L + \log M)$. Consequently, the bin-wise sorting can reduce the complexity by $LM \log L$.

TABLE II. Speed comparison of ground segmentation methods on Sequence 05 of the SemanticKITTI dataset, using Intel(R) Core(TM) i7-7700K CPU (unit: Hz).

Method	Speed	Method	Speed
LineFit [15]	58.96	GPF [16]	29.72
RANSAC [9]	15.43	R-GPF [2]	35.30
CascadedSeg [17]	13.07	Patchwork [1]	43.97
Patchwork++ (Ours)	54.85	**Patchwork++ w/o TGR**	**67.84**

VI. Conclusion

In this study, a fast and robust ground segmentation method, *Patchwork++*, has been proposed. The proposed method was verified to be faster and more robust than other state-of-the-art methods. Moreover, Patchwork++ reduces the number of parameters that require fine-tuning depending on the surroundings. Therefore, it has become easier to use ground segmentation in practice.

In the future, we plan to adopt Patchwork++ for various applications, including LiDAR odometry, object recognition, and static map building. Then, we will study the effect of ground segmentation on several applications.

References

[1] H. Lim, M. Oh, and H. Myung, "Patchwork: Concentric zone-based region-wise ground segmentation with ground likelihood estimation using a 3D LiDAR sensor," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 6458–6465, 2021.

[2] H. Lim, S. Hwang, and H. Myung, "ERASOR: Egocentric ratio of pseudo occupancy-based dynamic object removal for static 3D point cloud map building," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2272–2279, 2021.

[3] Z. Li, F. Wang, and N. Wang, "LiDAR R-CNN: An efficient and universal 3D object detector," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 7546–7555.

[4] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[5] C. Sung, S. Jeon, H. Lim, and H. Myung, "What if there was no revisit? Large-scale graph-based SLAM with traffic sign detection in an HD map using LiDAR inertial odometry," *Intell. Service Robot.*, pp. 1–10, 2021.

[6] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Voxelized GICP for fast and accurate 3D point cloud registration," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 11 054–11 059.

[7] B. Suger, B. Steder, and W. Burgard, "Traversability analysis for mobile robots in outdoor environments: A semi-supervised learning approach based on 3D LiDAR data," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 3941–3946.

[8] Z. Yan, T. Duckett, and N. Bellotto, "Online learning for human classification in 3D LiDAR-based tracking," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 864–871.

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

[10] J. Xu, R. Zhang, J. Dou, Y. Zhu, J. Sun, and S. Pu, "RPVnet: A deep and efficient range-point-voxel fusion network for LiDAR point cloud segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 16 024–16 033.

[11] A. Paigwar, Ö. Erkent, D. Sierra-Gonzalez, and C. Laugier, "GndNet: Fast ground plane estimation and point cloud segmentation for autonomous vehicles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 2150–2156.

[12] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9297–9307.

[13] Z. Shen, H. Liang, L. Lin, Z. Wang, W. Huang, and J. Yu, "Fast ground segmentation for 3D LiDAR point cloud based on Jump-Convolution-Process," *Remote Sensing*, vol. 13, no. 16, p. 3239, 2021.

[14] F. Moosmann, O. Pink, and C. Stiller, "Segmentation of 3D LiDAR data in non-flat urban environments using a local convexity criterion," in *Proc. IEEE Intell. Veh. Symp.*, 2009, pp. 215–220.

[15] M. Himmelsbach, F. V. Hundelshausen, and H.-J. Wuensche, "Fast segmentation of 3D point clouds for ground vehicles," in *Proc. IEEE Intell. Veh. Symp.*, 2010, pp. 560–565.

[16] D. Zermas, I. Izzat, and N. Papanikolopoulos, "Fast segmentation of 3D point clouds: A paradigm on LiDAR data for autonomous vehicle applications," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 5067–5073.

[17] P. Narksri, E. Takeuchi, Y. Ninomiya, Y. Morales, N. Akai, and N. Kawaguchi, "A slope-robust cascaded ground segmentation in 3D point cloud for autonomous vehicles," in *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 497–504.

[18] V. Jiménez, J. Godoy, A. Artuñedo, and J. Villagra, "Ground segmentation algorithm for sloped terrain and sparse LiDAR point cloud," *IEEE Access*, vol. 9, pp. 132 914–132 927, 2021.

[19] Y. Li, C. Le Bihan, T. Pourtau, and T. Ristorcelli, "Insclustering: Instantly clustering LiDAR range measures for autonomous vehicle," in *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, 2020, pp. 1–6.

[20] M. Sualeh and G.-W. Kim, "Dynamic multi-LiDAR based multiple object detection and tracking," *Sensors*, vol. 19, no. 6, p. 1474, 2019.

[21] A. Asvadi, C. Premebida, P. Peixoto, and U. Nunes, "3D LiDAR-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes," *Robot. Auton. Syst.*, vol. 83, pp. 299–311, 2016.

[22] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized LiDAR odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4758–4765.

[23] Y. Pan, P. Xiao, Y. He, Z. Shao, and Z. Li, "MULLS: Versatile LiDAR SLAM via multi-metric linear least square," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 11 633–11 640.

[24] D.-U. Seo, H. Lim, S. Lee, and H. Myung, "PaGO-LOAM: Robust ground-optimized LiDAR odometry," in *Proc. Int. Conf. Ubiquit. Robot.*, 2021, Submitted.

[25] X. Zhao, Z. Yang, and S. Schwertfeger, "Mapping with reflection – detection and utilization of reflection in 3D LiDAR scans," in *Proc. IEEE Int. Symp. Saf. Secur. Rescue Robot.*, 2020, pp. 27–33.

[26] M. Weinmann, B. Jutzi, S. Hinz, and C. Mallet, "Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers," *ISPRS J. Photogramm. Remote Sens.*, vol. 105, pp. 286–304, 2015.