Chu *et al. Hum. Cent. Comput. Inf. Sci.* (2019) 9:17
https://doi.org/10.1186/s13673-019-0178-5

Human-centric Computing and Information Sciences

RESEARCH Open Access

Enhanced ground segmentation method for Lidar point clouds in human-centric autonomous robot systems

Phuong Minh Chu[1], Seoungjae Cho[1], Jisun Park[1], Simon Fong[2] and Kyungeun Cho[1*]

*Correspondence:
cke@dongguk.edu
[1] Department of Multimedia Engineering, Dongguk University-Seoul, 30 Pildong-ro 1-gil, Jung-gu, Seoul 04620, Republic of Korea
Full list of author information is available at the end of the article

Abstract

Ground segmentation is an important step for any autonomous and remote-controlled systems. After separating ground and nonground parts, many works such as object tracking and 3D reconstruction can be performed. In this paper, we propose an efficient method for segmenting the ground data of point clouds acquired from multi-channel Lidar sensors. The goal of this study is to completely separate ground points and nonground points in real time. The proposed method segments ground data efficiently and accurately in various environments such as flat terrain, undulating/rugged terrain, and mountainous terrain. First, the point cloud in each obtained frame is divided into small groups. We then focus on the vertical and horizontal directions separately, before processing both directions concurrently. Experiments were conducted, and the results showed the effectiveness of the proposed ground segment method. For flat and sloping terrains, the accuracy is over than 90%. Besides, the quality of the proposed method is also over than 80% for bumpy terrains. On the other hand, the speed is 145 frames per second. Therefore, in both simple and complex terrains, we gained good results and real-time performance.

Keywords: Human-centric, Internet of things, Autonomous robot, Point cloud, Ground segmentation

Introduction

Internet of things (IoT) is growing fast in over the world [1–7]. In an IoT-based system for the autonomous vehicles, light detection and ranging (Lidar) sensors are often used to collect data of surrounding environments. Furthermore, in human-centric autonomous systems, robots also have several attached cameras and an inertial measurement unit-global positioning system (IMU-GPS) sensor. In each frame, the Lidar sensor returns a point cloud that describes the terrain around the robot. The data from the Lidar sensor are transferred to a computer and split into two groups: ground and nonground. The first group includes ground points of terrain which a robot can traverse. On the other hand, the second group consists of nonground points which the robot cannot traverse such as cars, trees, walls, etc. If the terrain is sloping such that the autonomous robot cannot traverse it, the corresponding points are clustered into the nonground group. The segmentation of three-dimensional (3D) point cloud ground data is a fundamental

step that is extremely important for robot operation. Especially, ground cloud segmentation is a pre-processing step for many terrain reconstruction applications [8–10]. The ground segmentation result is used for recognizing objects, classifications, and feature extraction. Dealing with large datasets in real time and in urgent situations such as rescuing people in distress is a challenging task. This is why a fast and accurate ground segmentation method is necessary for real-time autonomous systems.

In recent years, ground segmentation has become an important and challenging task, and is now the focus of considerable research. However, ground segmentation remains an open problem because of the complexity of the input data and the real-time requirements. Therefore, this paper proposes a fast and highly accurate ground segmentation method for 3D Lidar point clouds. The main contribution of this paper is to provide an infinitely faster ground segmentation approach than previous ones based on geometry features and distribution of points in each scanline. In addition, the proposed method also performs high accuracy in various terrains.

The remainder of this paper is organized as follows. The next section presents several related works. "Ground segmentation algorithm" section proposes the novel ground segmentation approach. "Experiments and analysis" section summarizes the results from experiments. The discussion and conclusion are presented in "Discussion" and "Conclusion" sections, respectively.

Related works

Numerous approaches have been used to segment 3D point cloud data. However, the discovery of a fast and accurate ground segmentation method is still a challenging task for real-time autonomous systems. We roughly categorize these techniques in relation to our present research as outlined below.

Typically, current object tracking approaches [11–13] segment objects on the ground from the background in a frame-by-frame manner using two-dimensional (2D) images. These approaches can be extended to enable segmentation with 3D images using large datasets such as 3D point cloud data. For example, a fully automatic approach for 3D point cloud segmentation [14] uses the ground segmentation results for detection and geometric modeling.

Wallenberg et al. [15] proposed a purely color-based leaf segmentation using data gathered from a Kinect sensor. Their algorithm segments an RGB image (typically, the leaves on a plant from the background) from a color camera based on color and depth information. In [16], the authors proposed an active segmentation technique based on the depth and color information of RGBD images, but with the goal of tracking objects and keeping them in the center of the image.

Hernández [17] focused on the automatic detection and classification of artifacts located at the ground region. Although this method achieves good results on flat terrain, it cannot be applied to sloping or rugged terrain. In [18], the authors described the segmentation of ground into flat and non-flat urban environments using local convexity measures. Their results indicate good performance over a variety of terrain, but the computational cost of the algorithm means that this method cannot process data in real time. Cho [19] and Lin [20] proposed different ground segmentation approaches by dividing a point cloud dataset into smaller parts such as voxels or blocks. The results are

good in certain cases and their algorithms work well with all kinds of terrain but can be time-consuming. Douillard [21] proposed a set of segmentation methods designed for various 3D point cloud densities. Although this method achieves good results within a reasonable computation time, the dependency on sets of adjacent points (four neighboring points) for each calculation makes it time-consuming to build a terrain mesh and implement further computations. In [22], the authors introduced a new segmentation method based on scanline segmentation. This approach enhances the processing efficiency of the massive amounts of data using GPU acceleration. However, this method only works well in urban areas.

Wellington et al. presented a method for a generative terrain model by exploiting the natural structure observed by the sensors [23]. Their model exploits a 3D spatial structure in outdoor domains and uses a set of noisy data to classify obstacles and estimate the ground height and vegetation height. To detect obstacles while supporting the ground estimation process, their model includes two Markov random fields, a hidden semi-Markov model (HSMM), and voxel models. Their approach models 3D structures efficiently in vegetation and smooth terrain using a 150×150 grid of 15-cm square voxel columns. Therefore, this method is computationally demanding. In [24], the authors proposed a region-growing algorithm based on an octree-based voxelized representation to extract segments and a separate process to refine the segmentation. According to their results, this approach performs particularly well in urban environments and is computationally efficient. Zhang [25] proposed a ground segmentation method by combining a Markov random field with loopy belief propagation to update the ground-height probabilities and segmentation. Their algorithm can segment rough and steeply sloped regions with good results. However, this method cannot operate in real time because the average processing time of their algorithm is greater than 1 s. Therefore, this method cannot meet the twin requirements of real-time processing and good quality.

The fast ground segmentation method [26] has the ability to segment normal terrain accurately and efficiently. However, for complex terrain, this method becomes inefficient. To overcome the issues discussed above and upgrade previous approach, we propose a novel segmentation algorithm that deals with a wide variety of terrain and is sufficiently fast for real-time operation.

Ground segmentation algorithm

In this section, a heuristic method is proposed using geometry features and distribution of points in each scanline. Point cloud data in each frame are segmented by considering both directions: vertical and horizontal. Figure 1 shows an example of a point cloud obtained in one frame. This figure also gives some examples of vertical and horizontal directions.

System overview

In this study, the data in each frame are segmented through a three-stage process. An overview of the main algorithm is illustrated in Fig. 2. In each frame, the 3D range sensor returns a point cloud. All points in the cloud are in local coordinates and the origin is the sensor location. First, the data from one frame are processed in the vertical direction. Each point is assigned a temporary label: ground or nonground. In the

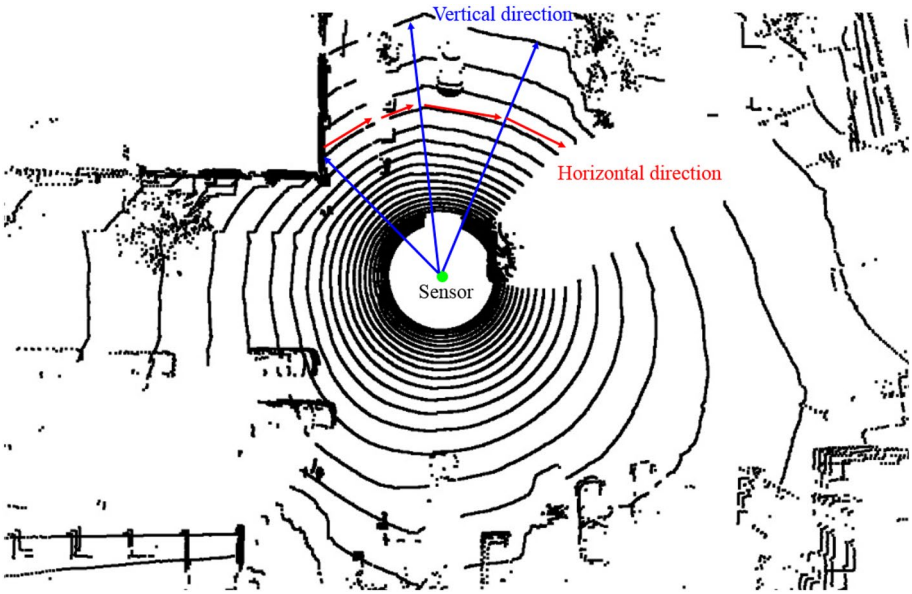


Fig. 1 Example of a point cloud obtained in one frame

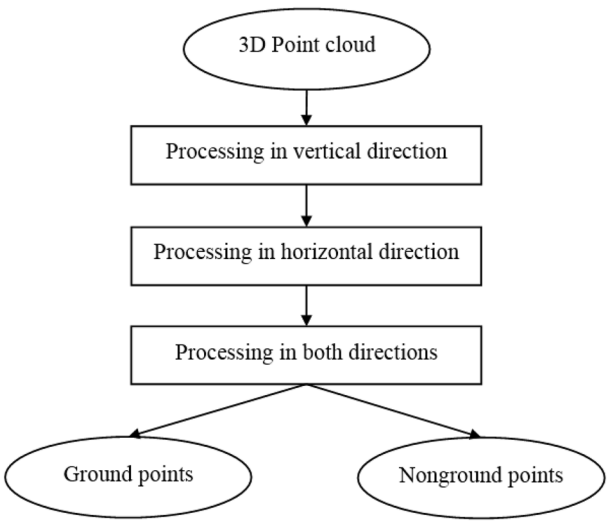


Fig. 2 Overview of the ground segmentation algorithm

second stage, each scanline in the horizontal direction is processed. If an n-channel laser sensor is employed, we will obtain n scanlines in the horizontal direction. In horizontal direction, the resolution of the Lidar sensors is higher than that of vertical direction. Therefore, the processing algorithm for the horizontal direction is totally different from the algorithm for the vertical direction. After the second stage, the data in both directions are processed. The label initially assigned to each point can be changed in the second and third stages. Finally, we obtain two groups: ground points and nonground points. After segmentation, all ground and nonground points

are converted to global coordinate using GPS-IMU data for further processing steps. The details of the main algorithm are described below.

Processing in the vertical direction

In the first stage, the fast ground segmentation technique described in study [26] is used. The algorithm is illustrated in Fig. 3. After receiving a local point cloud, all points are divided into vertical lines. All lines begin with a ground point at the scanner location and end at the points in the furthest scanline. In theory, if an n-channel scanner is used, we obtain n points in each vertical line. Because we have added the ground point at the sensor position, the number of points is $n+1$. In practice, as the robot moves, we will lose many points. These lost points occur when the laser line does not collide with any object, i.e., there is no signal feedback. In each vertical line, all start-ground points and threshold points are searched. The first start-ground point is always the first point of the vertical line, which is the ground point at the sensor position. From this point, the first threshold point is determined by considering each pair of consecutive points. Next, we assign a ground label from the start-ground

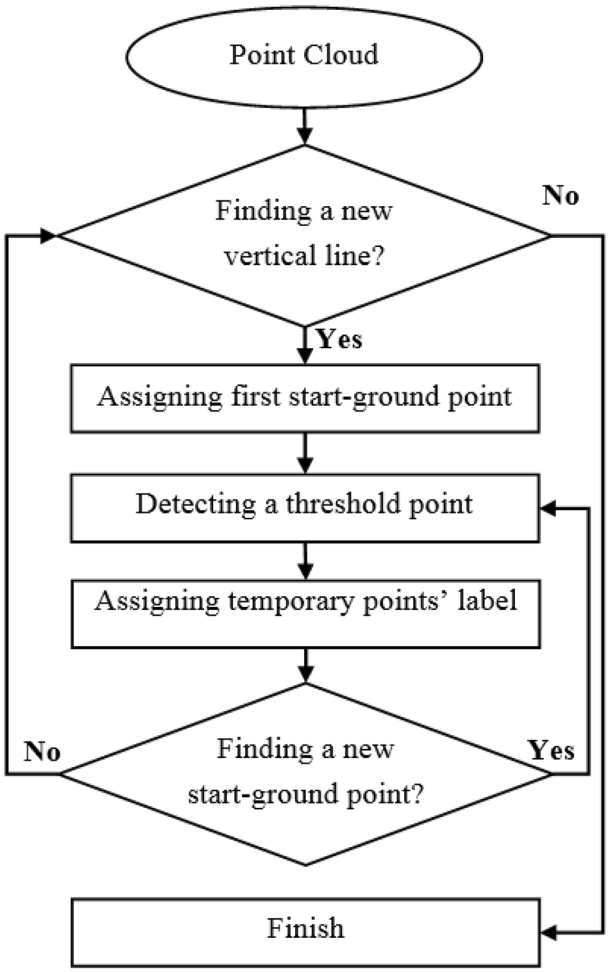


Fig. 3 Algorithm of the processing in vertical direction

point to the next (threshold) point. After the threshold point, nonground labels are assigned until the next start-ground point is identified. These point labels are temporary, and will be reconsidered in subsequent stages.

Processing in the horizontal direction

In the second stage, each scanline is processed separately in three steps, as shown in Fig. 4. As mentioned above, if an n-channel Lidar sensor is used, we have n scanlines in each frame. Generally, the scanlines are circular. First, each scanline is divided into smaller "level-2" lines (see "Dividing a scanline into level-2 lines" section). Each "level-2" line contains a list of consecutive points in a scanline. This division is dependent on the distance between each pair of consecutive points. In the second step, all level-2 lines are classified and labeled, and reduce the number of types of line from four to two (see "Classification and labeling of level-2 lines" section). Each line is then either a ground line or a nonground line. Lines in which all points have the ground label are ground lines, and those in which all points have the nonground label are nonground lines. In the third step, each level-2 line is considered in relation to the other level-2 lines in the horizontal direction, and update the labels of any abnormal level-2 lines. If the label of a level-2 line changes, the labels of all points on the line change accordingly.

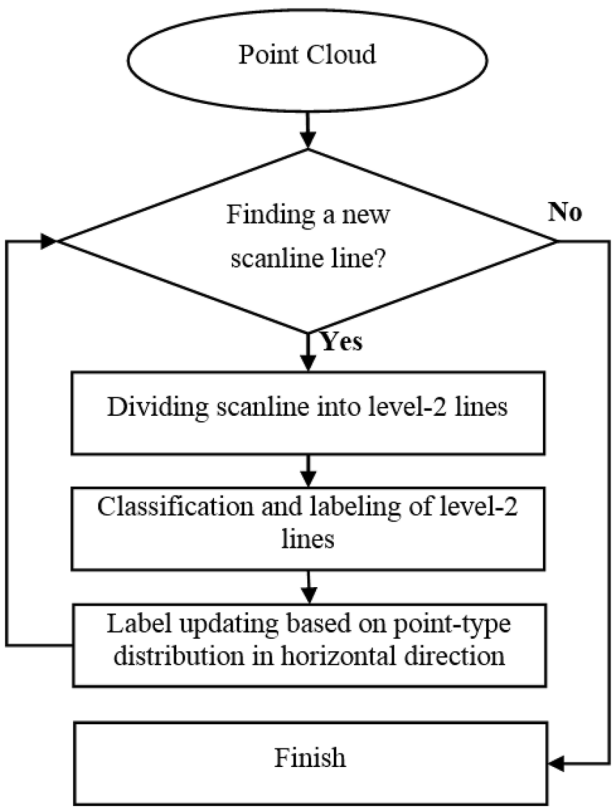


Fig. 4 Algorithm of the processing in horizontal direction

Dividing a scanline into level-2 lines

The method of dividing scanlines into level-2 lines is inspired by real-life observations. In the scanline, the distance between two consecutive points on a single object is less than the distance between two consecutive points on two different objects. Based on this observation, each scanline is divided into smaller lines.

The distance between two consecutive points in each scanline is calculated. If the distance is less than some constant minimum value $d_{\min}$, we place both points on one level-2 line. Otherwise, the previous point is placed on the current level-2 line and the next point is assigned to a new level-2 line. The value of $d_{\min}$ depends on the type of Lidar sensor and number of channels of the scanline.

Classification and labeling of level-2 lines

Each level-2 line is a set of points. There are four types of level-2 line. The first type contains only ground points, whereas the second type contains only nonground points. The third and fourth types contain both ground and nonground points. In the third type, the lines include separate and distinct ground and nonground parts. In the fourth type, the ground and nonground points alternate and are mixed together.

In the next step, the number of types is reduced from four to two. All lines of the first and second types are maintained. For the third type, we calculate the average height of all points in the ground and nonground parts in each level-2 line. Depending on the average heights, the third line type is splited into two cases: (i) If the difference between the average height of the ground points and the nonground points is less than $h_{\min}$, all points in this line will be assigned the same label. The density of ground and nonground points are compared. If the ground points constitute the majority, all nonground labels are changed to ground labels. Otherwise, the nonground labels are maintained and the ground labels are changed. This produces a line that is of the first or second type. (ii) If the difference between the average height of the ground points and nonground points is greater than $h_{\min}$, each line is splited into two smaller lines. The first line contains only ground points and the second contains only nonground points. For lines of the fourth type, the same adjustment method is used as for case (i) above. In general, the Lidar sensor data contain errors of a few centimeters. Therefore, we define $h_{\min}$ so as to ignore the errors of the 3D range sensor.

Label updating based on point-type distribution in horizontal direction

We now have two types of level-2 lines: ground and nonground. In the third step, the labels of the level-2 lines are adjusted according to the point-type distribution in the horizontal direction. This step incorporates Algorithm 1 and Algorithm 2. In the first algorithm, each pair of consecutive level-2 lines in each scanline is considered. Here, n_{i} and $n_{\mathrm{i+1}}$ denote the number of points in lines $\mathrm{L_i}$ and $\mathrm{L_{i+1}}$, respectively. An abnormal case occurs if $\mathrm{L_i}$ and $\mathrm{L_{i+1}}$ have a similar average height but different types. In this case, the ratio of the number of points in $\mathrm{L_i}$ and $\mathrm{L_{i+1}}$ is calculated. If the ratio r_{i} is greater than some threshold $r_{\max}$, there is a high probability that the next line $\mathrm{L_{i+1}}$ is incorrectly labeled. Therefore, the type of $\mathrm{L_{i+1}}$ should be changed. If $1 - r_{\mathrm{i}}$ is

greater than r_{max}, the type of L_i is also changed. After this step, L_i and L_{i+1} have the same type.

Algorithm 1: Label updating based on point-type distribution in each pair of consecutive level-2 lines

For each scanline S **in** List of Scanlines **do**
For each level-2 line L_i **in** S **do**
  If $L_i \neq$ Last_line_in_S **then**
    If Type(L_i) ≠ Type(L_{i+1}) **then**
      $h_i \leftarrow$ Average Height Of L_i()
      $h_{i+1} \leftarrow$ Average Height Of L_{i+1}()
      If $|h_i - h_{i+1}| < h_{min}$ **then**
        $r_i \leftarrow n_i/(n_i + n_{i+1})$
        If $r_i > r_{max}$ **then**
          Change type of L_{i+1}()
        Else If 1- $r_i > r_{max}$ **then**
          Change type of L_i()
        End
      End
    End
  End
End

In the second algorithm, each group of three consecutive level-2 lines is considered. The details of the adjustment are described in Algorithm 2. We must check two conditions: (i) Is the label of the center line different from the labels of the previous and next lines? (ii) Is the difference in the average height of the lines less than some threshold? If the answer to both conditions is *yes* for any group of three consecutive lines, there is a high probability that the center line is incorrectly labeled. In this case, the labels of all points on the center line are changed.

Algorithm 2: Label updating based on point-type distribution in each group of three consecutive level-2 lines

For each scanline S **in** List of Scanlines **do**
For each level-2 line L_i **in** S **do**
  If $L_i \neq$ First_line_in_S **AND** $L_i \neq$ Last_line_in_S **then**
    If Type(L_{i-1}) ≠ Type(L_i) **AND** Type(L_i) ≠ Type(L_{i+1}) **then**
      $h_{i-1} \leftarrow$ Average Height Of L_{i-1}()
      $h_i \leftarrow$ Average Height Of L_i()
      $h_{i+1} \leftarrow$ Average Height Of L_{i+1}()
      If $|h_i - h_{i-1}| < h_{min}$ **AND** $|h_i - h_{i+1}| < h_{min}$ **then**
        Change type of L_i()
      End
    End
  End
End

Processing in both directions

In the third stage, all level-2 lines in both the vertical and horizontal directions are processed. The details are described in Algorithm 3. For each level-2 line L in scanline S, two level-2 lines are created in the next and previous scanlines (the first and last scanlines are ignored in this step). From each point P in L, a corresponding point P_N is determined in the next scanline that has the same horizontal index as P. Then, P_N is placed in L_N. In the same way, we find P_P in the previous scanline and place it in L_P. It is not always possible to find P_N or P_P if they are lost points. Algorithm 3 calculates the gradient between L and L_N. This function finds the center point of each line and identifies an angle g_n by calculating the gradient between the two center points. A maximum angle g_{max} is defined. The g_{max} value represents the maximum slope which robot can traverse. If g_n is less than g_{max} and L and L_N have different types, there is a high probability that L or L_N is incorrectly labeled. To make a more accurate decision, the previous line L_P is considered. If the current line L is determined to be incorrectly labeled, the labels of all points in L are changed. Alternatively, if L_N is found to be incorrectly labeled, we change its label to match that of L. After this stage, all labels of the points in the current frame are fixed to either ground points or nonground points.

Algorithm 3: Processing in both directions

```
For each scanline S in List of Scanlines do
  If S ≠ First_scanline AND S ≠ Last_scanline do
    For each level-2 line L in S do
      Create level-2 line L_N in next scanline()
      Create level-2 line L_P in previous scanline()
      n_ns ← Number of Points in L_N having same label as L()
      n_n ← Number of Points in L_N
      r_n ← n_ns/n_n
      If r_n < 1 - r_max then
        g_n ← Get Gradient between L and L_N()
        If g_n < g_max then
          n_ps ← Number of Points in L_P having same label as L()
          n_p ←Number of Points in L_P
          r_p ← n_ps/n_p
          If r_p < 1 - r_max then
            Change type of L()
          Else If r_p > r_max then
            Assign label of L to all points in L_N()
          End
        End
      End
    End
  End
End
```

Experiments and analysis

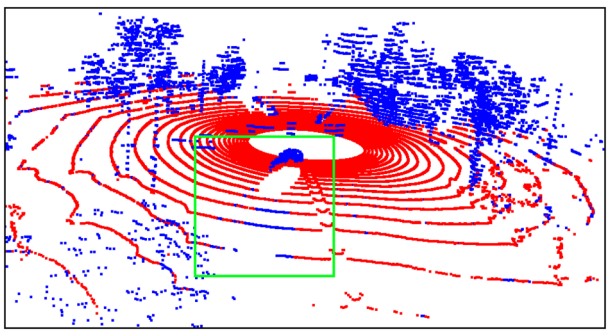
We implemented experiments to verify the proposed ground segmentation method both in qualitative and quantitative terms. Moreover, several state-of-the-art methods were used for comparison to demonstrate the effectiveness and high quality of the proposed approach.

Experimental method

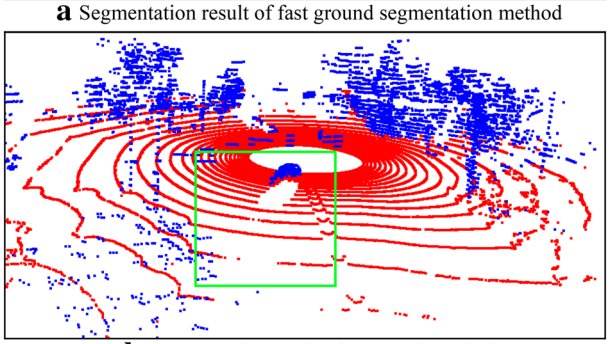
For the experiments and analysis, datasets captured from a Lidar sensor (Velodyne HDL-32E, Velodyne Inc., Morgan Hill, CA, USA) were employed. The first dataset was obtained from a simple, flat terrain. The second dataset was obtained from sloping terrain, and the third dataset was taken from more complex regions. To evaluate the quantitative results, a set of ground truth data was constructed. For each dataset, we ran the proposed method and fast ground segmentation method [26] and compared the results. We also compare the performance with other done methods using Velodyne HDL-32E sensor. The experiments were conducted using a PC equipped with an Intel Core i7-6700 3.4 GHz CPU and 16 GB RAM. We set $d_{\min} = 20$ cm, $g_{\max} = 30°$, $h_{\min} = 10$ cm, and $r_{\max} = 0.7$ according to the capabilities of the robot and sensor. For the first stage, the same parameters as the experiments reported in [26] were used. These values are not dependent on the features of the terrain. For other robots or Lidar sensors, different values can be chosen.

Experimental results

All experiments produced favorable results on both the simple and complex terrain. Figure 5 shows the results from one frame of the flat dataset segmented at a crossroads (red and blue points represent ground and nonground points, respectively.). The flat terrain contains cars, pedestrians, lampposts, and trees. Figure 6 demonstrates the results for the sloped terrain. The sloped terrain includes many trees on both sides of the robot. The results on sloped and bumpy terrain are shown in Fig. 7. The sloped and bumpy terrain also contains many trees and bushes. Figures 5a, 6a,

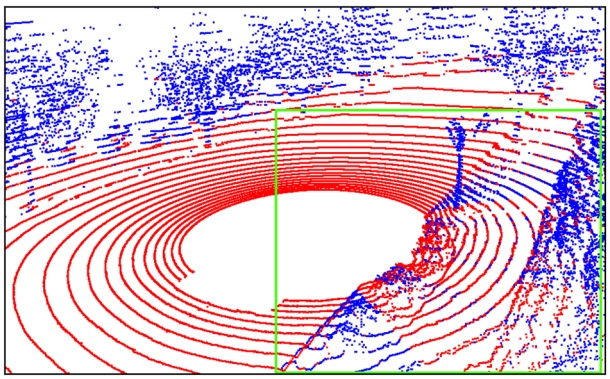


a Segmentation result of fast ground segmentation method

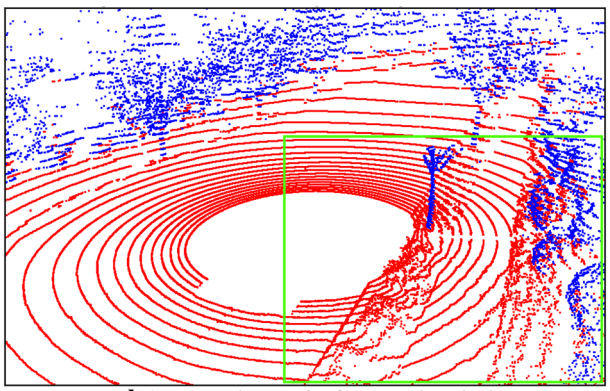


b Segmentation result of proposed method

Fig. 5 Qualitative comparison of segmentation results on flat terrain



a Segmentation result of fast ground segmentation method



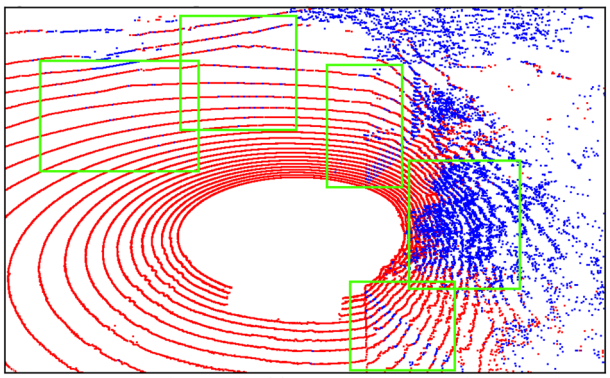
b Segmentation result of proposed method

Fig. 6 Qualitative comparison of segmentation results on sloping terrain

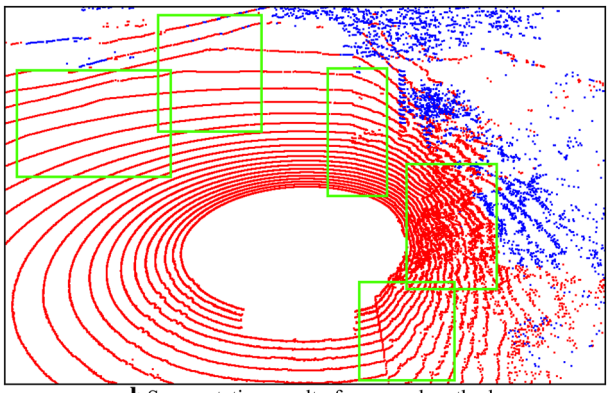
and 7a show the results given by the fast ground segmentation method. Figures 5b, 6b, and 7b show the results given by the method proposed in this paper. For each type of terrain, the proposed method outperforms the previous ground segmentation method. The results of fast segmentation method contain several errors. Sometimes, the ground parts were determined as nonground ones. The areas in which the segmentation performance is superior are marked by green rectangles. By using the proposed method, the flat and sloped roads are well segmented and defined as ground, and humans, trees, walls, lampposts, and cars are identified as nonground points. In addition, Fig. 7 shows the perspective view of variable terrain. The terrain contains both sloping and undulating mountain roads, but the segmentation results meet our expectations.

Experimental analysis

Table 1 compares the processing time of the proposed method with that of other methods using the same Velodyne HDL-32E sensor. Each frame of data contains approximately 60,000 points, and the proposed algorithm required an average of 6.9 ms per frame. The method proposed here is slightly slower than that in fast ground segmentation method, which averages just 4.7 ms, but the difference is not large. This suggests that the proposed technique could process 145 frames per second (fps). The Velodyne



a Segmentation result of fast ground segmentation method



b Segmentation result of proposed method

Fig. 7 Qualitative comparison of segmentation results on sloping and bumpy terrain

Table 1 Comparison of average processing time per frame

Method	Average processing time per frame (ms)
Fast ground segmentation [26]	4.7
Voxel based ground segmentation [19]	19.31
Loopy belief propagation based ground segmentation [25]	1000
Proposed method	6.9

HDL-32E sensor captures data at 10 fps. Therefore, the segmentation system works well in real time, as it operates 14.5 times faster than the sensor.

To analyze the accuracy of the proposed method, we compared our results with the ground truth data. We also compared the results by calculating the true positive rate (TPR) of each method. The results are presented in Table 2. In flat and sloping terrain, the proposed method performs slightly better than the fast ground segmentation method [26]. However, over complex terrain, the new method is significantly better, with an improvement in accuracy of approximately 18%.

Table 2 Quantitative results using ground truth data

Terrain	Total number of frames	TPR of fast ground segmentation [26] (%)	TPR of proposed method (%)
Flat	47	94.61	94.71
Sloping	84	91.25	91.60
Sloping and bumpy	286	63.14	81.10

Discussion

The experimental results demonstrated that the proposed method could well separate ground and nonground from a 3D point cloud at high speed. The proposed method can be applied to the autonomous robots and remote-controlled systems. For examples, the object detection and tracking applications can be performed from nonground part. Furthermore, we can reconstruct the 3D scene in real time by applying different methods for the ground and nonground. In future work, we will modify the proposed ground segmentation method to improve quantitative quality on various datasets.

Conclusion

In this paper, we proposed a novel ground segmentation method for Lidar point clouds that uses local coordinates to deal with each received data frame. The core ideas of this method are that the point cloud is not only processed along each vertical scanline, as in previous research, but along each horizontal scanline and in both directions simultaneously. The experimental results using this extended method indicate that our approach is fast and effective over both simple and complex terrain. In future work, we will extend the proposed method to other, more complex terrain types, and will further enhance the quality of the algorithm.

Authors' contributions
PMC, SC, JP have written the source codes. SF contributed to the discussion and analysis of the results. KC provided full guidance. PMC has written the paper. All authors read and approved the final manuscript.

Author details
[1] Department of Multimedia Engineering, Dongguk University-Seoul, 30 Pildong-ro 1-gil, Jung-gu, Seoul 04620, Republic of Korea. [2] Department of Computer and Information Science, University of Macau, Macau 3000, China.

Competing interests
The authors declare that they have no competing interests.

Availability of data and materials
Not applicable.

Consent for publication
Not applicable.

Ethics approval and consent to participate
Not applicable.

Funding
This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korea government (MSIP) (2018R1A2B2007934).

Publisher's Note

Received: 21 November 2018 Accepted: 25 April 2019
Published online: 09 May 2019

References

1. Kang WM, Moon SY, Park JH (2017) An enhanced security framework for home appliances in smart home. Hum Centric Comput Inform Sci 7(6):1–12
2. Jo H, Yoon YI (2018) Intelligent smart home energy efficiency model using artificial TensorFlow engine. Hum Centric Comput Inform Sci 8(9):1–18
3. Gubbi J, Buyya R, Marusic S, Palaniswami M (2013) Internet of Things (IoT): a vision, architectural elements, and future directions. Fut Gen Comput Syst 29(7):1645–1660
4. Lee W, Cho S, Chu P, Vu H, Helal S, Song W, Jeong YS, Cho K (2016) Automatic agent generation for IoT-based smart house simulator. Neurocomputing 209:14–24
5. Kaur J, Kaur K (2017) A fuzzy approach for an IoT-based automated employee performance appraisal. Comput Mater Continua 53(1):23–36
6. Xiao B, Wang Z, Liu Q, Liu X (2018) SMK-means: an improved mini batch k-means algorithm based on mapreduce with big data. Comput Mater Continua 56(3):365–379
7. Zhao X, Wu J, Zhang Y, Shi Y, Wang L (2018) Fault diagnosis of motor in frequency domain signal by stacked de-noising auto-encoder. Comput Mater Continua 57(2):223–242
8. Khatamian A, Arabnia HR (2016) Survey on 3D surface reconstruction. J Inform Process Syst 12(3):338–357
9. Chu PM, Cho S, Fong S, Park YW, Cho K (2017) 3D reconstruction framework for multiple remote robots on cloud system. Symmetry 9(4):1–16
10. Chu PM, Cho S, Sim S, Kwak K, Cho K (2018) Multimedia system for real-time photorealistic nonground modeling of 3D dynamic environment for remote control system. Symmetry 10(4):1–15
11. Ren X, Malik J (2007) Tracking as repeated figure/ground segmentation. In: 2007 IEEE conference on computer vision and pattern recognition, Minneapolis, MN, USA, 17–22 June, pp 1–8
12. Li F, Kim T, Humayun A, Tsai D and Rehg JM (2013) Video segmentation by tracking many figure-ground segments. In: IEEE international conference on computer vision, Sydney, Australia, 1–8 December, pp 2192–2199
13. Kuettel D, Ferrari V (2012) Figure-ground segmentation by transferring window masks. In: 2012 IEEE conference on computer vision and pattern recognition (CVPR), Providence, RI, USA, 16–21 June, pp 558–565
14. Verma V, Kumar R, Hsu S (2006) 3D Building detection and modeling from aerial Lidar data. In: 2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06), New York, NY, USA, 17–23 June, pp 2213–2220
15. Wallenberg M, Felsberg M, Forssén P (2011) Leaf segmentation using the Kinect. In: Proceedings SSBA'11 symposium on image analysis, Linköping, Sweden, 17–18 March
16. Tomori Z, Gargalik R, Hrmo I (2012) Active segmentation in 3D using Kinect sensor. In: Proc. Int'l conference computer graphics visualization and computer vision, Plzen, Czech, 25–28 June, pp 163–167
17. Hernández J, Marcotegui B (2009) Point cloud segmentation towards urban ground modeling. In: 2009 urban remote sensing event, Shanghai, China, 20–22 May, pp 1–5
18. Moosmann F, Pink O, Stiller C (2009) Segmentation of 3D Lidar data in non-flat urban environments using a local convexity criterion. In: IEEE intelligent vehicles symposium, Xi'an, China, 3–5 June, pp 215–220
19. Cho S, Kim J, Ikram W, Cho K, Jeong Y, Um K, Sim S (2014) Sloped terrain segmentation for autonomous drive using sparse 3D point cloud. Sci World J 2014:1–10
20. Lin X, Zhanga J (2015) Segmentation-based ground points detection from mobile laser scanning point cloud. In: The international archives of the photogrammetry, remote sensing and spatial information sciences, 2015 international workshop on image and data fusion, Hawaii, USA, 26 June, pp 99–102
21. Douillard B, Underwood J, Kuntz N, Vlaskine V, Quadros A, Morton P, Frenkel A (2011) On the segmentation of 3D LIDAR point clouds. In: IEEE international conference on robotics and automation, Shanghai, China, 9–13 May, pp 2798–2805
22. Hu X, Li X, Zhang Y (2013) Fast filtering of LiDAR point cloud in urban areas based on scan line segmentation and GPU acceleration. IEEE Geosci Rem Sens Lett 10(2):308–312
23. Wellington C, Courville A, Stentz A (2006) A generative model of terrain for autonomous navigation in vegetation. Int J Robot Res 25(12):1287–1304
24. Vo AV, Truong LH, Laefer DF, Bertolotto M (2015) Octree-based region growing for point cloud segmentation. ISPRS J Photogram Rem Sens 104:88–100
25. Zhang M, Morris DD, Fu R (2015) Ground segmentation based on loopy belief propagation for sparse 3D point clouds. In: 2015 international conference on 3D vision, Lyon, France, 19–22 October, pp 615–622
26. Chu PM, Cho S, Sim S, Kwak K, Cho K (2017) A fast ground segmentation method for 3D point cloud. J Inform Process Syst 13(3):491–499