Fast Segmentation of 3-D Point Clouds Based on Ground Plane State Tracking

Jin Chen
School of Mechanical Engineering
Shanghai Jiao Tong University
Shanghai, CN
chenjin920414@sjtu.edu.cn

Yafei Wang
School of Mechanical Engineering
Shanghai Jiao Tong University
Shanghai, CN
wyfjlu@sjtu.edu.cn

Kunpeng Dai
School of Mechanical Engineering
Shanghai Jiao Tong University
Shanghai, CN
daikunpeng@sjtu.edu.cn

Taiyuan Ma
School of Mechanical Engineering
Shanghai Jiao Tong University
Shanghai, CN
nanshanyi@sjtu.edu.cn

***Abstract*—3D point cloud segmentation is the first and essential step for LIDAR-based perception, and its result has a great impact on subsequent tasks such as classification and tracking. This paper proposes a fast and precise two-step 3D point cloud segmentation algorithm based on ground plane state tracking. The algorithm first extracts the points belonging to the ground. To avoid processing the whole point cloud in every frame and improve computational efficiency of the algorithm, we introduce tracking idea into the ground point extraction step and estimate the ground plane state by the fusion of prior information and measurement information. the second step is to cluster the remaining non-ground points. We introduce an adaptive threshold-based RBNN (Radially Bounded Nearest Neighbor strategy) clustering algorithm which reduces the number of mis-segmentation by determining the adaptive threshold function according to the characteristics of Lidar point cloud. Experimental results demonstrate that the algorithm consumes less time and achieves high segmentation accuracy than previous works.**

Keywords—autonomous vehicle, point cloud, Lidar-based perception, real-time segmentation

I. INTRODUCTION

Lidar can depict the 3D features of the surrounding objects and is known for its capability to acquire accurate distance information in a dense three dimensional (3D) cloud of points. Because of these characteristics, Lidar is widely used in autonomous vehicle perception system. The Lidar-based autonomous vehicle perception system generally consists of three parts: segmentation, classification, and tracking. Segmentation is the first step of the perception system that separates raw point cloud data into disjunct subsets representing different objects so that the autonomous vehicle can detect and locate obstacles in surrounding area. Since segmentation pre-processing is the step before detection and classification, its quality directly affects the performances of these subsequent perceptual steps.

Autonomous vehicle system generally operates in cities at high speed, while vehicles, bicycles, pedestrians, and other obstacles in complex traffic environment will pose a threat to it. This situation requires autonomous perception system to quickly and accurately locate obstacles around and determine the area can be passed. Therefore, the point cloud segmentation algorithm needs to achieve high real-time and accuracy performance. The main challenge to ensure real-time performance is that the vast amount of potentially 3D point data needs to be processed. It is necessary to design segmentation algorithm with high efficiency and low computational complexity. The Lidar is scanned around the scene gathering distance measurements in the form of point cloud at specified angle intervals, which makes the gap between adjacent two points larger with distance growth. This feature of Lidar is a huge challenge to ensure the accuracy of the segmentation algorithm. The faulty segmentation of point cloud is generally divided into two types : over-segmentation and under-segmentation. over-segmentation is separating one object into multiple portions, and under-segmentation is incorrectly combining multiple targets into one target. Over-segmentation and under-segmentation can cause misjudgment of the perception system and affect subsequent autonomous vehicle subsystem, e.g. pass-planning or decision. The frequency of over-segmentation and under-segmentation is another important indicator for evaluating the superiority of the segmentation algorithm.

Several approaches have been proposed to segment point cloud using grid [1-3]. The point cloud is projected onto a 2D plane which is divided into several grid cells. The height of cells is calculated by points inside the cell which dividing the grid cells as ground or non-ground. Kiyosimo Kidono et al. [1] divide ground and non-ground points according to the difference between the maximum height and the minimum height in grid map cell, and cluster the occupied cell by a distance-based clustering algorithm. M. Himmelsbach et al. [2] propose fan-shaped grid cell to divide ground and non-ground points, then points labeled as non-ground are clustered making use of euclidean point distances.

Other approaches have focused on performing ground extraction and clustering non-ground points. Dimitris Zermas et al. [4] use deterministically assigned seed points to extract the ground surface in an iterative fashion and then cluster the remaining non-ground points taking advantage of the structure of the LIDAR point cloud. Igor Bogoslavskyi et al. [5] convert the raw measurements of Lidar into a range image. They extract ground plane according to range measurements and propose to an angle-based methodology to cluster non-ground points. Myung-Ok Shin et al. [6] combine a 2D grid and an undirected graph structure to directly cluster non-ground points without ground extraction and employ Gaussian process to reduce over-segmentation. Similar to aforementioned works, our approach firstly extract points belonging to the ground, then cluster the remaining points into meaningful sets. The main contribution of this paper is in two aspects. Firstly, we propose a real-time and stable method to extract ground plane based on the tracking of ground plane state using particle filter. Secondly, we propose an adaptive

threshold based RBNN algorithm to cluster non-ground points, reducing over-segmentation and under-segmentation in the driving environments.

The reminder of this paper is organized as follows. In Section II, we give a detailed description of our approach. We then provide experimental results in Section III that verify the superiority of our algorithm. Finally, we conclude the paper in Section IV.

II. SEGEMENTATION METHODOLOGY

The following paragraph describes the processing flow of our segmentation algorithm. The first step of our algorithm is tracking ground plane state by particle filter for the fast extraction of the ground points, which we call Ground Plane State Tracking (GPST). Then, we present a point cloud clustering methodology named Adaptive Threshold Radially Bounded Nearest Neighbor strategy (AT-RBNN).

Each paragraph consists of three parts: (i) a brief reasoning behind the algorithm selection (ii) the overview of the algorithm according to the pseudocode diagrams (iii) the discussion of the algorithm implementation details.

A. Ground Plane State Tracking

The purpose of target detection based on LiDAR is to cluster point cloud belonging to obstacles around the autonomous vehicle. Ground point extraction is to distinguish non-ground points from all points which reduces the number of points involved in non-ground points clustering and has a great impact on the accuracy of the proceeding steps. This paper proposes a fast, efficient and accurate ground extraction algorithm based on ground plane state tracking. For ground points removal, we make two realistic assumptions. First, we assume that the ground surface can be represented by flat model. Second, we assume that in the extremely short time interval (such as 0.1s), the two ground surface have a large similarity, and the previous ground plane state has great value for estimating the current ground plane state. In order to effectively evaluate the accuracy of the ground plane state, we take the average value of the distance from the preselected ground points to the ground plane as the observation measurement. In view of the nonlinearity of the observation equation, we use particle filter theory to track the ground plane state.

The plane can be represented by the following simple linear model:

$$z = ax + by + c \tag{1}$$

Let us note $X = (a, b, c)^T$ the state vector that characterizes the tracked ground plane. Under the assumption that the ground plane changes slowly, the state equation of system is described as follow:

$$X_k = X_{k-1} + v_s \tag{2}$$

Where v_s is a three-dimensional white Gaussian noise with zero mean and Q covariance. Lidar acquires the position coordinate about the vast amount of points (up to 10,0000 points per scanning),which brings two difficulties to the definition of the measurement equation. First, how to avoid high computational complexity caused by iterative calculation of all points; Second, which variable can effectively reflect the accuracy of ground plane state estimation. In order to avoid these problems, we determine the potential ground points by the ground state at the last moment, and then reasonably

Algorithm 1: Pseudocode of the ground plane state tracking methodology based on particle filter

Input: P:a point cloud received by a 360° coverage LiDAR sensor
Output: P_g:points belonging to ground surface
P_{ng}:points not belonging to ground surface

Initialization:
N_s: the number of particles
v_s: the noise of state model
v_m: the noise of measurement model
Main Loop:
for $k = 0; k \leq t$ **do**
  if $k = 0$ **then**
    X_0 = **ExtractInitialState**(P);
    $\left[\{x_i^0, w_i^0\}_{i-1}^{N_s}\right]$ = **Initialize**(X_0);
  else
    for $i = 1; i \leq N_s$ **do**
      $X_k^i = f(X_{k-1}^i, v_s)$;
      $d_k^i = h(X_k^i, v_t)$;
      $w_k^i = w_{k-1}^i p(d_k^i | X_k^i)$;
    end
    $w_k^i = w_k^i / \sum_{i=1}^{N_s} w_k^i$;
    $\hat{X}_k = \sum_{i=1}^{N_s} w_k^i X_k^i$;
    $N_{eff} = 1 / \sum_{i=1}^{N_s} (w_k^i)^2$;
    if $N_{eff} < N_{threshold}$ **then**
      $\left[\{x_i^k, w_i^k\}_{i=1}^{N_s}\right]$ = **Resampling**$(\left[\{x_i^k\}_{i=1}^{N_s}\right])$;
    end
  end
end

sample the potential ground points, and take the average value of the distance from the sample points to the ground plane at this moment as the measurement. Therefore, the measurement equation is as follows:

$$d_k = h(X_k, v_t) \tag{3}$$

v_t is a 1-dimentional white Gaussian noise with zero mean and R covariance. The measurement function h is strongly nonlinear.

The plane state tracking solution uses particle filter based on the sequential Monte Carlo techniques. In this paper, the particles are described by the state vector $X_k^i = (a_k^i, b_k^i, c_k^i)$ and their related importance weights w_k^i. The basic structure of our method can be summarized as

1) ExtractInitialState: Applying particle filter to track the ground state first needs to obtain the initial state of the ground state we called Extract Initial State. In our method, Extracting initial state can use any ground segmentation algorithm based on single frame detection. In this paper ,we use a iterative multiple plane fitting technique called Ground Plane Fitting (GPF), which is presented at [4]. The approach iteratively extracts seed points with low height values and determine plane state by singular value decomposition (SVD) methodology.
2) Initialization: each particle j = 1,2, … , N_p is drawn as $X_0^j \sim N(X_0, Q_0)$ where X_0 is the initial state vector and Q_0 the relative covariance matrix:
$$Q_0 = diag(\sigma_{0,a}^2, \sigma_{0,b}^2, \sigma_{0,c}^2) \tag{4}$$
The importance weights are set to $w_0^j = \frac{1}{N_s}$.

3) Evolution: the prior density is used as proposal distribution. The particles evolve in the state-space according to the dynamic equation 2, i.e. N_p samples X_k^j , j $= 1,2, \dots, N_p$, are drawn according to $p(X_k^j|X_{k-1}^j)$ via the law $p(W_k)$.
4) Measurement Calculation: the difficulty lies in the calculus the particles weights since they are strongly linked to the modeling of the measurement equation. For each X_k^j , a particle-based measure d_k^j is calculated by the highly nonlinear equation h. In our approach h is estimated as (for each particle j):
   a) Each point in the point cloud segment P is evaluated against the candidate plane, the state vector of which is $\hat{X}_k$, and produces the distance from the point to its orthogonal projection on the candidate plane. This distance is compared to a user defined threshold Th_{pg}, which decide whether the point is used to calculate the measurement d_k or not. The set P_{ps} contains these pre-selected points.
   b) In order to improve the computational efficiency, N_{dp} points are sampled from the set P_{ps} for the final calculation of d_k, which made up the point set P_m produces the distance from to its orthogonal projection on the plane represented by X_k^j, and d_k^j is the mean of the distance. The equation is as follow:

$$d_k^j = \frac{\sum_{i=1}^{P_m} D(X_k^j, p_i)}{N_{P_m}} = \frac{\sum_{i=1}^{N_{P_m}} \frac{|ax_i+by_i-z_i+c|}{\sqrt{1+a^2+b^2}}}{N_{P_m}} \quad (5)$$

   In the equation, p_i is the i-th point of P_m, and its coordinates are (x_i, y_i, z_i). N_{P_m} is the number of points in the set P_m.
5) Updating weights: Particle weights w_k^j are updated using forecasted observations $h(\cdot)$ and measured observations d_k at time k according to (6)

$$w_k^j = \frac{w_{k-1}^j p(d_k^j|X_k^j) p(X_k^j|X_{k-1}^j)}{q(X_k^j|X_{k-1}^j, d_k)} \quad (6)$$

Where $q(X_k^j|X_{k-1}^j, d_k)$ denotes the posterior PDF, which is an important term because it significantly influences the filter performance. Generally, the prior transition is used as the proposal distribution, where

$$q(X_k^j|X_{k-1}^j, d_k) = p(X_k^j|X_{k-1}^j) \quad (7)$$

The weight updating then simplifies to

$$w_k^j = w_{k-1}^j \, p(d_k^j|X_k^j) \quad (8)$$

In this paper, we use Gaussian error distributions for all observation perturbations, partly for simplicity, partly because we have no knowledge of the appropriate distribution to use. However, the two main reasons are that under ideal condition, the ground points are all on the ground plane, the measurement value is 0 and that a random perturbation with Gaussian distributions does not yield unrealistic results. Thus, the likelihood function can be expressed as

$$p(d_k^i|X_k^i) = \frac{1}{(2\pi)^{\frac{1}{2}}\sigma_r} \times \exp(-\frac{1}{2\sigma_r}(d_k^i)^2) \quad (9)$$

The updated particle weights can be obtained according to

$$w_k^i = w_{k-1}^i \, p(d_k^i|X_k^i) \quad (10)$$

Finally, according to the following equation, the updated particle weights are normalized.

$$w_k^i = \frac{1}{\sum_{j=1}^{N_s} w_k^j} \quad (11)$$

6) Estimation: using the importance weights, the estimate is then:

$$\hat{X}_k = \sum_{i=1}^{N_s} w_k^i X_k^i \quad (12)$$

7) Resampling: according to an estimation of the effective sample size $N_{eff} = 1 / \sum_{i=1}^{N_s} (w_k^i)^2$, the resampling process is performed as [7] and [8].
8) Finally, calculate the distance from the point belonging in the point cloud segment P to its orthogonal projection on the plane represented by $\hat{X}_k$. We set the threshold Th_{fg} to compare with the distances to distinguish the non-ground points from the ground points.

B. Non-ground points clustering

The second step of segmentation described in this subsection is to cluster non-ground points behind ground extraction. Given the set of remaining points $P_{ng} = \{p_i | p_i = (x, y, z)_i, i = 0,1, \dots \dots, N_{ng} - 1\}$, our goal is to find m clusters $C_j, j = 1, \dots \dots, m$ so that one cluster contains all points belonging to one object, and using simple mechanisms will ensure the fast calculation time and low complexity of the process.

For achieving the goal of high real-time and accuracy, we choose the RBNN (radially bounded nearest neighbor) point cloud clustering algorithm proposed in [9], and improve it in the following aspect according to the lidar data characteristic and the application scenario. The lidar date in the perception sector of autonomous driving has its own special features. In the case of the same range, the distance between two adjacent vertical points is greater than the distance between two adjacent horizontal points, and as the range increases, the distance between two points increases. In this situation, the RBNN algorithm is easy to cause over-segmentation and under-segmentation errors. Therefore, we determine the equation of the nearest neighbor search adaptive threshold based on the Lidar characteristics.

The velodyne HDL-32E lidar that we use scans the environment by rotating in a clock wise at a frequency by $10Hz$. While the sensor is rotating, all the 32 lasers that are positioned vertically interval at a certain angle $\delta\phi_v$ are emitted simultaneously at same rotation angle $\delta\phi_h$. Thus, the point belonging in the horizontal m-th layer and the vertical n-th layer can be expressed as p_{mn}; each p_{mn} has a range r_{mn}. The HDL-32E has a 360° horizontal field of view with about 0.17° resolution. And it has a 40° vertical field of view, with +/- 20° up and down. The angular resolution in the vertical direction is about 1.3° . The difference between horizontal and vertical resolution causes the resultant point clouds to be denser in the transverse direction and relatively sparse in the radial direction.

Algorithm 2: Pseudocode of the non-ground points clustering based on AT-RBNN algorithm

Input: P_{ng}:points not belonging to ground surface
Output: P_{ngl}:points with labels that represent their cluster indentity

Initialization:
$label$: the cluster identity
N_{ng}: the number of points in P_{ng}
Main Loop:
$label = 0$;
Buildkdtree(P_{ng});
for $i = 0; i \leq N_{ng} - 1$ **do**
  if $p_i.label \neq 0$ **then**
    continue;
  else
    $th_d =$ **CalculateDistanceThreshold**(p_i);
    $P_{neighbors} =$ **findNeighbors**(p_i);
    for $j = 0; j \leq N_{P_{neighbors}}$ **do**
      if $p_i.label \neq 0$ ***and*** $p_j.label \neq 0$ **then**
        $p_j.label = p_i.label$;
      else
        if $p_j.label \neq 0$ **then**
          $p_i.label = p_j.label$;
        end
        if $p_i.label \neq 0$ **then**
          $p_j.label = p_i.label$;
        end
      end
    end
    if $p_i.label = 0$ **then**
      $label = label + 1$;
      $p_i.label = label$;
      for $j = 0; j \leq N_{P_{neighbors}}$ **do**
        $p_j.label = label$;
      end
    end
  end
end

The RBNN algorithm traverses each point in the point cloud P_{ng}, obtains neighbor points $P_{ne} = \{p_{ne} | p_{ne} = (x, y, z)_i, i = 0,1, \ldots\ldots, N_{ne} - 1\}$ within a certain distance threshold th_d of the target point p_i, and detects these neighbor points P_{ne}, so that each point acquires a label l that represents its cluster identity. The distance threshold directly affects the performance of the clustering algorithm. In order to determine the reasonable variable threshold, we further study and analyze the point cloud distribution of typical obstacle appearing in the traffic scene. Fig. 1 illustrates an example of the point cloud distribution on the tail of vehicle, which embodies the Lidar data characteristic that the distance between two adjacent vertical points is larger than the distance between two adjacent horizontal points. If we search for the target point neighbor using the vertical distance interval as the threshold, the points between different layers will be assigned to different clusters, as shown in Fig. 1. for avoiding this undesirable result, we determine the distance threshold th_d based on the vertical distance interval.

Fig. 2 shows a depiction where two adjacent laser beams from scanner are separated by a pitch angle hit the objective. The inner range R_v with the pitch angle ϕ_v belongs to the v-th layer and the outer range R_{v+1} with the pitch angle ϕ_{v+1}

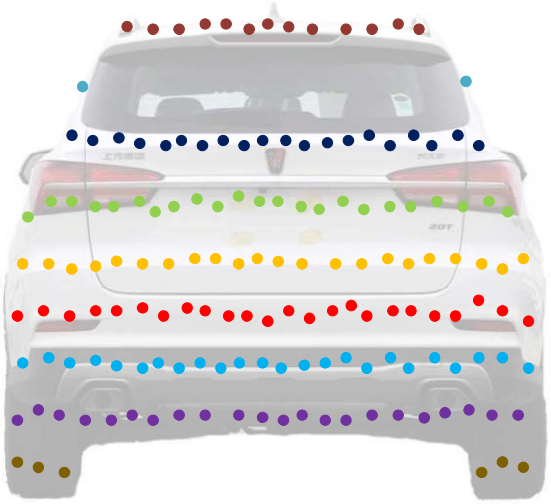

Fig. 1. Point Cloud distribution disgram on certical surface

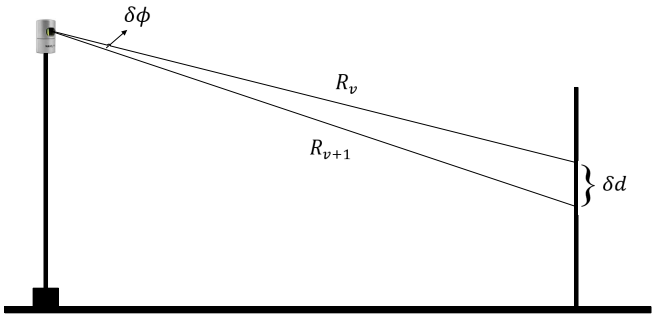

Fig. 2. LiDAR laser setup:Adjacent lasers in the radial direction

belongs to the $(v + 1)$-th layer. Thus, the difference in pitch angles $\delta\phi$ that shown in the figure is $(\phi_{v+1} - \phi_v)$. It is important to note no matter how the Lidar orientation or the terrain elevation changes, the difference in pitch angles between the adjacent lasers does not change and the $\delta\phi_v \approx 1.7°$ in our experiment. The inner range R_v and outer range R_{v+1} can be measured by Lidar data. Therefore, we can determine equation 13 based on the geometric relationship shown in Fig. 2 to calculate the distance between two adjacent points on the same obstacle in the vertical direction.

$$\delta d = \sqrt{{R_v}^2 + {R_{v+1}}^2 - 2R_v R_{v+1}\cos(\delta\phi)} \tag{13}$$

The clustering algorithm in this paper has to obtain the adjacent points of the target point within a certain distance. The distance threshold th_d needs to be determined based on the given target point. Thus, we get the equation 14 for calculating the distance threshold th_d according to the equation 13.

$$th_d = k\sqrt{2(1 - \cos(\delta\phi))}R_v \tag{14}$$

There are two changes compared to equation 14 of equation 13. Firstly, since the value $\delta R = |R_{v+1} - R_v|$ is small and the neighbors should be determined only by the information of the target point, we use R_v instead of R_{v+1}. Secondly, We multiply the original equation by a factor k, and definitely $k > 1$. This is mainly due to the following considerations: i) the noise of the lidar data, ii) The surface of the obstacle is not necessarily a vertical plane, a cylindrical surface such as a tree. These factors lead to an increase in the

distance of adjacent points, and the introduction of k can effectively reduce the impact of these conditions on the clustering results.

The principle and structure of AT-RBNN algorithm is depicted in Alg. 2. In the main loop of Alg. 2, we initialize the label of cluster identity and build the kd-tree for searching for neighbor points. Each point in the point cloud P_{ng} is evaluated and if it has been assigned to a cluster, skip this point and evaluate the next one. This is a very efficient step in the algorithm, because it avoids performing a nearest neighbor query for every point in the point cloud. For the current point p_i which has not been assigned to a cluster, we calculate the distance threshold th_d by equation 14 and use the kd-tree to obtain the set $P_{neighbors}$ of all adjacent points whose distance from current point is less than th_d. For evaluating each point p_j in the set $P_{neighbors}$, if the current point p_i and its adjacent point p_j have been already assigned to a specific cluster, merge the points in the same cluster according to the label of p_i. If only one point of the two points in p_i p_j is assigned to cluster, assign the unassigned point to the cluster containing the other point.

III. EXPERIMENTAL RESULTS

We evaluate our segmentation approach using a dataset gathered by our owning autonomous vehicle. However, the method is not restricted to a particular platform or sensor. Our experimental driving platform is a ROEWE ERX5 equipped with a Velodyne HDL-32 laser range finder. For visualization, we utilized Point Cloud Library(PCL)[10].

A. Ground Plane State Tracking

To compared the performance of the proposed ground extraction algorithm Ground Plane State Tracking(GPST), two other ground extraction algorithm were tested together: a RANSAC plane fitting implementation which is used in similar scenarios, a Ground Plane Fitting(GPT) algorithm from the study[4].

Fig.3 shows the comparison of the three algorithms in real-time performance. The GPST algorithm in this paper performs best, followed by the GPT algorithm.

B. Non-ground points clustering

To compared the real-time performance of the proposed non-ground points clustering algorithm adaptive threshold-based radially bounded nearest neighbor strategy (AT-RBNN), two other clustering algorithm were tested together: a Euclidean Cluster Extraction (ECE), a Scan-Line Run (SLR) algorithm from the study[4].

In order to evaluate the clustering performance of AT-RBNN algorithm, we compare it with Scan-Line Run (SLR) algorithm from the study [4] and evaluate the algorithm accuracy based on the frequency of over-segmentation in the same scene. Fig. 5 and Fig. 6 show the final segmentation results of the AT-RBNN and SLR algorithms in the same scene. It can be seen that several objects in the lower left corner of Fig. 6 are incorrectly segmented into multiple parts. In Fig. 5, there is no over-segmentation in the segmentation results of AT-RBNN algorithm. Table 1 shows the superiority of the AT-RBNN algorithm in this respect by the data.

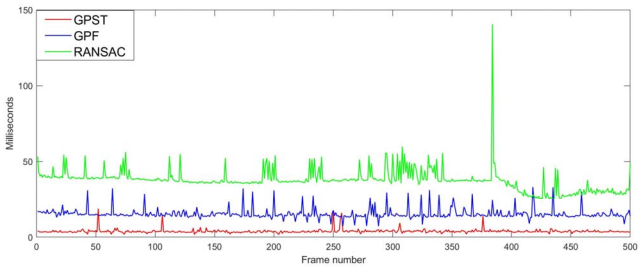

Fig. 3. Comparison of the ground extraction processing times for 500 frames from our own data set on an Intel Core i7-8750H at 2.21GHz.

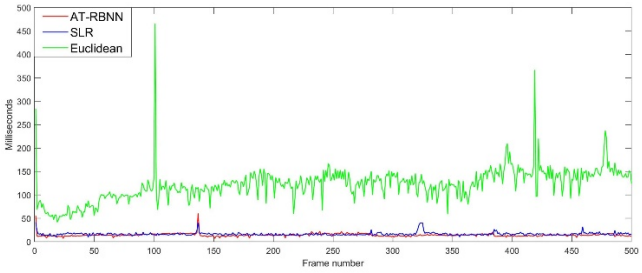

Fig. 4. Comparison of the non-ground clusterig processing times for 500 frames from our own data set on an Intel Core i7-8750H at 2.21GHz.

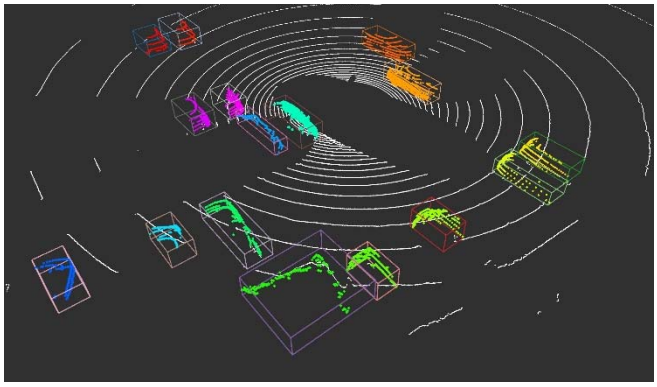

Fig. 5. Segmentation results of AT-RBNN algorithm proposed in this paper

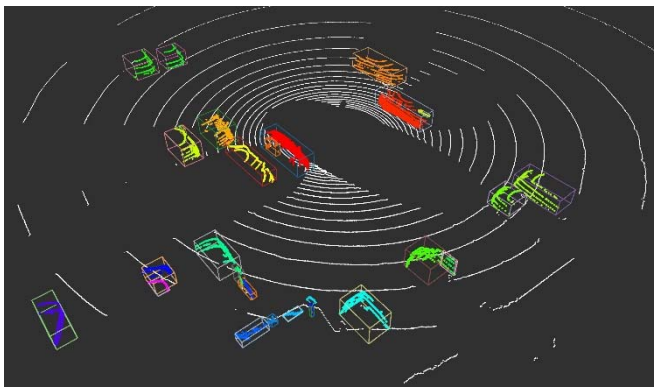

Fig. 6. Segmentation results of SLR algorithm

TABLE I. ACCURACY COMPARISION IN 216 FRAMES CONTAINING 3456 TARGETS

Algorithm	***Euclidean***	***SLR***	***AT-RBNN***
Over-segmentation	625	432	224
Precision	81.92%	87.5%	93.53%

IV. CONCLUSION AND FUTURE WORK

We have studied the problem of point cloud segmentation in autonomous vehicle field and have proposed an approach

that initially extracts the ground points and consequently clusters the non-ground points based on adaptive threshold. The proposed solution performs better in real-time and segmentation accuracy, which makes it ideal for real applications in autonomous vehicle on data gathered by LiDAR data. The algorithm has been successfully tested in a dataset gathered by our owning autonomous vehicle. Experiments show that our algorithm's performance on bumpy ground surface needs to be improved. The next step is to improve the performance of our algorithm when encountering rough, uneven terrain and rapid slope changes. For this we plan to introduce more complex ground model and point cloud organizational structure to update the data set.

REFERENCES

[1] K. Kidono, T. Miyasaka, A. Watanabe, T. Naito, and J. Miura, "Pedestrian recognition using high-definition LIDAR," in Proc. IEEE Intell. Vehicles Symp. (IV), Jun. 2011, pp. 405–410.

[2] M. Himmelsbach, F. V. Hundelshausen, and H.-J. Wuensche, "Fast segmentation of 3D point clouds for ground vehicles," in Proc. IEEE Intell. Vehicles Symp. (IV), Jun. 2010, pp. 560–565.

[3] D. Held, J. Levinson, S. Thrun, and S. Savarese, "Combining 3D shape, color, and motion for robust anytime tracking," in Proc. Robot., Sci. Syst., Berkeley, CA, USA, Jul. 2014.

[4] Zermas, Dimitris, I. Izzat, and N. Papanikolopoulos. "Fast segmentation of 3D point clouds: A paradigm on LiDAR data for autonomous vehicle applications." IEEE International Conference on Robotics & Automation 2017.

[5] Bogoslavskyi, Igor, and C. Stachniss. "Efficient Online Segmentation for Sparse 3D Laser Scans." Pfg – Journal of Photogrammetry Remote Sensing & Geoinformation Science 85.1(2017):41-52.

[6] Shin, Myung Ok , et al. "Real-Time and Accurate Segmentation of 3-D Point Clouds Based on Gaussian Process Regression." IEEE Transactions on Intelligent Transportation Systems (2017):1-15.

[7] S. Arulampalam, S. Maskell, N. J. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," IEEE Trans. Signal Process., vol. 50, no. 2, pp. 174–188, Feb. 2002.

[8] H. Moradkhani, K. Hsu, H. Gupta, and S. Sorooshian, "Uncertainty assessment of hydrologic model states and parameters: Sequential data assimilation using the particle filter," Water Resour. Res., vol. 41, no. 5, p. W05012, May 2005.

[9] Reddy, Satish K. , and P. K. Pal . " [ACM Press the 2015 Conference - Goa, India (2015.07.02-2015.07.04)] Proceedings of the 2015 Conference on Advances In Robotics - AIR \"15 - Segmentation of point cloud from a 3D LIDAR using range difference between neighbouring beams." (2015):1-6.

[10] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2011, pp. 1–4.