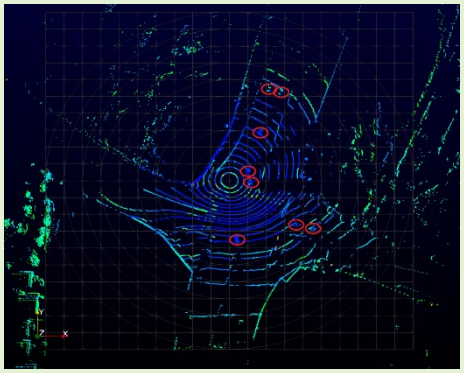
An Unsupervised Clustering Method for Processing Roadside LiDAR Data with Improved Computational Efficiency

Yibin Zhang, Nischal Bhattarai,Junxuan Zhao, Hongchao Liu, Hao Xu

***Abstract*— In transportation, LiDAR has been primarily used in autonomous vehicles to assist self-driving until recently when people realized it could also be installed at the roadside to support connected vehicles and infrastructure systems. Unlike onboard LiDAR sensors used in autonomous vehicles, roadside applications must perform complete background filtering and clustering as well as tracking real-time traffic movements within the detection zone. This paper presents an unsupervised clustering method for roadside or infrastructure-based LiDAR applications. It first converts 3D LiDAR data points into 2D so that only target points (after background filtering) will be saved in the channel-azimuth 2D structure; then, a method combining the region growing algorithm and counted component labeling is used to perform clustering. Lastly, a merging process is conducted to enhance the connected component labeling method for better outcomes. Experimental studies demonstrate that the proposed method could reach 0.011s per frame (10 Hz sensor rotation frequency) in clustering while maintaining high accuracy.**



***Index Terms*— connected component labeling, computational efficiency, object clustering, region growing, roadside LiDAR.**

I. Introduction

Automated driving systems are revolutionary technologies that could shape our future by minimizing and eventually eliminating human errors in driving. However, autonomous vehicles have some critical barriers to overcome towards full self-driving automation. The obstacles may include, but are not limited to, the challenge to identify objects, the limits of computation power of onboard computers, and the difficulty to react appropriately to any scenario a car may encounter while on the road. A solution to the problem is to develop infrastructure-based systems that can help self-driving cars better understand roads and help nonmotorized road users get active protection when an automatic system fails. Studies on the application of LiDAR technology at the infrastructure side and the development of connected vehicle and infrastructure systems were recently conducted by several groups of researchers [1]-[6]. By timely collecting and analyzing data from the roadside LiDAR and sending dedicated information to cars (e.g., broadcasting traffic light's color directly to a vehicle's computer), researchers expect the infrastructure-based system can take the substantial burden out of an autonomous vehicle's onboard computers [7] and significantly improve the efficiency of self-driving.

Infrastructure-based LiDAR usually works independently and thus has a heavier computation load. A busy intersection can easily have thousands of vehicles during peak hours along with other modes of road users such as pedestrians and cyclists. Roadside or infrastructure-based LiDAR sensors need to monitor every road user and track their movements in real-time to capture risky behaviors (such as red-light running) and trigger protective actions. Another challenge to processing roadside LiDAR data applications is its cost. Commercially available LiDAR sensors include products with 1, 4, 8, 16, 32, 64, and 128 laser channels, and the number of laser channels determines the accuracy and detection range as well as the cost. Automated vehicles are primarily equipped with128-channel products, while roadside applications must use low-cost products with fewer channels, limited range, and lower resolution [6] [8]. The detection accuracy may also be affected by other factors, such as retrieved aerosol [9]. Despite the limits of using low-cost LiDAR sensors, an infrastructure-based system must be able to detect and analyze the situation, activate the warning scheme, and deliver the information to pedestrians, bicyclists, and drivers in a very short time horizon. To this end, current methods for point cloud data processing, esp. the machine learning methods for background filtering, object identification[10], and movement tracking, need to be thoroughly reviewed and further developed for infrastructure-based applications; copy of existing methods for autonomous vehicles will not work. For instance, for onboard LiDAR, the

Yibin Zhang, Nischal Bhattarai, Junxuan Zhao, and Hongchao Liu are with Department of Civil, Environmental, and Construction Engineering, Texas Tech University, TX79415, USA (e-mail: yibin.zhang@ttu.edu; Nischal.Bhattarai@ttu.edu, Junxuan.Zhao@ttu.edu; Hongchao.Liu@ttu.edu;).

Hao Xu is with Department of Civil and Environmental Engineering, University of Nevada. NV89557, USA (e-mail: haox@unr.edu)

random point cloud should be removed [11], whereas, for roadside LiDAR, the point cloud of static objects like buildings and trees should be removed.

II. Related Works

Background filtering, object clustering, object classification, and object tracking are four major steps for processing roadside LiDAR data; this study is focused on clustering. Both supervised classification [12][13] and unsupervised clustering [14] are used in object grouping. In supervised classification, the class or label of an object must be given; machine learning methods such as neural networks [15] and decision trees [16] are commonly used in supervised clustering. To automatically cluster objects in real-time, an unsupervised recognition method should be applied because it is impossible to specify the labels beforehand. Density-based spatial clustering of applications with noise (DBSCAN), K-means clustering methods, and their variations are popular unsupervised methods [17][18]. However, K-means requires the number of clusters as an input [19], which does not fit with infrastructure-based LiDAR applications in which the number of road users is unknown. DBSCAN algorithm is better suited for vehicle clustering as it separates clusters based on the density of points in the point cloud without requiring the number of clusters as a hyperparameter. Usually, DBSCAN can produce fairly good results if computation time is not an issue, making it good for offline applications.

Over-segmentation (see Figure 1) is also a problem that decreases the precision of clustering [20]. The Figure on the top of Figure 1 shows the raw data of vehicles after the background filtering. The circled object on the top of Figure 1 should be one vehicle. The Figure on the bottom is the presentation of over-segmentation. X-axis and Y-axis represent the horizontal and vertical distances from the top-view, and the location of LiDAR is represented by the star at coordinates (0,0). As can be seen, one vehicle was identified as two objects (vehicle #1 and vehicle #2) if the over-segmentation is not stressed. Researchers used the Gaussian Process (GP) regression [21] to improve clustering results. However, the fact that the GP regression is applied to every point only works for the situation containing extremely large or small objects due to time consideration. This may not be suitable for infrastructure-based LiDAR sensors because the two similar size objects may occur due to occlusion [22].

The introduction by far has illustrated the differences between the applications of vehicle-mounted and infrastructure-based LiDAR sensors as well as the major challenges to processing infrastructure-based LiDAR data in real-time. This paper presents a so-called counted region growing method derived from the common concept of region growing [23][24] and connected component labeling [25][26] approach to improve the computation efficiency of clustering. A merging process is added after applying the counted region growing to overcome the problem of over-segmentation.

The rest of this paper is structured as follows: section III introduces the 2D data structure; section IV presents the counted region growing method and the merging process; section V presents the experimental study along with discussions, and section VI concludes the study with the cons and pros of the proposed approach.

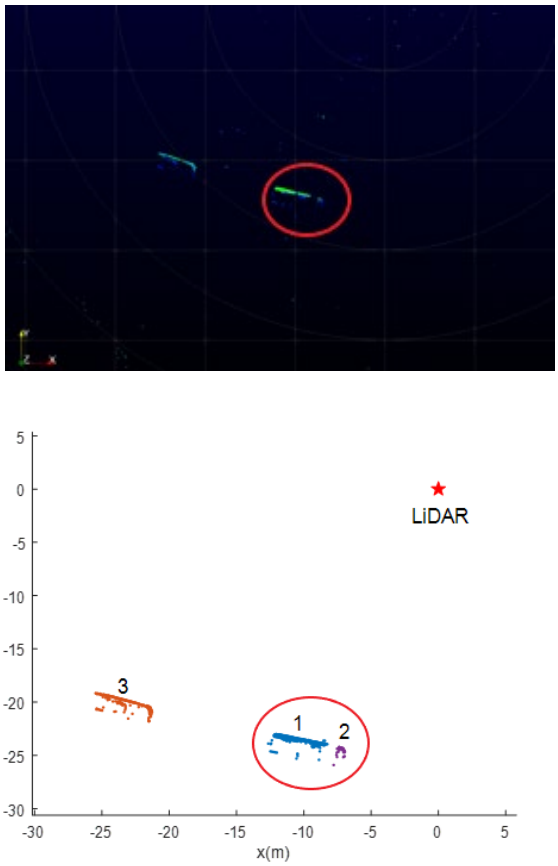


Fig. 1. Over-segmentation of vehicle

III. 2D Data Structure

Converting 3D data structure to 2D [27][28] is a common procedure for point cloud data processing, which can be done by established algorithms such as SLAM (Simultaneous Localization and Mapping) [29][30]. As aforementioned, LiDAR products use a different number of paired lasers to measure the distance to objects. Lasers are fired from top to bottom while the LiDAR sensor rotates around the center to form a circle. Thus every data point can be located by a vertical angle (ω), azimuth (α), and R (distance between LiDAR and the data point.

More specifically, the sensor uses the time-of-flight (ToF) [31] methodology to store each laser pulse's fire time and direction. After the laser hits an obstacle and is reflected, LiDAR registers the time-of-acquisition. The rotation rate influences the resolution of horizontal angle (α), which can be obtained from the following equation:

$$\alpha_{resolution} = \frac{s}{60} \times 360 \times t \quad (1)$$

where s is the rotation speed (cycle/min), t is the time of firing sequence (s/cycle).

The 2D data structure can be made as follows:

$$grid(x, y) = R \quad (2)$$

where,

$$x = (\alpha + offset)/\alpha_{resolution} \quad (3)$$

$$y = n \; when \; \omega \; is \; fired \; by \; nth \; laser \quad (4)$$

It is notable that since laser emits in pairs, the precise horizontal angle is not the exact value of α; instead, an offset is needed to amend the value of α. Due to the effect from the offset, the value of $\alpha + offset$ may exceed the range from [0, 360], therefore, to obtain the accurate reference, it can be converted to a different scale by covering all values using the angular resolution as:

If $\alpha + offset > 360$:

$$x = (\alpha + offset - 360)/\alpha_{resolution} \quad (5)$$

If $\alpha + offset < 360$:

$$x = (\alpha + offset + 360)/\alpha_{resolution} \quad (6)$$

A LiDAR that contains 32 beam lasers is used in this study. All 32 laser beams are fired in pairs and recharged periodically at an interval called Firing Sequence, and Firing Sequence determines the vertical angle of data points. Meanwhile, the sensor's motor is set to 10 Hz, and the value of $\alpha_{resolution}$ is obtained to be 0.2 since the time for each fire sequence is 55.296μs.

Based on the analysis above, a 2D data structure was obtained to store the data points. The column ranges from [1, 32] because there are 32 lasers, and the row ranges from [1, 1800] because the sensor generates one scan (360-degree rotation) with 1800 azimuth intervals ($\alpha_{resolution} = 0.2$-degree), R is the 3D distance between the sensor and each data point (if exists). The offsets are provided as unique values for a specific type of LiDAR sensor from manufacturers. As shown in Table I, the numbers in the first column represent the azimuth intervals (x), the numbers in the first row represent the laser order (y), and the numbers in the blocks are R values (3D distance). 0 means no data in that block. No data means that the laser fired by LiDAR does not hit any objects or data is filtered after the background filter.

TABLE I
A SAMPLE OF THE 2D DATA STRUCTURE

x \ y / R	…	13	14	15	…
…	…	…	…	…	…
597	0	0	10.80	0	0
598	0	10.83	10.88	10.77	0
599	0	10.84	10.64	10.71	0
600	0	10.85	10.65	10.73	0
601	0	10.76	10.66	10.74	0
…	…	…	…	…	…

Note: x: azimuth intervals. y: laser order. R:D distance. The meanings of the parameters are the same for the following Table.

IV. METHODOLOGICAL APPROACH

A. Counted Region Growing

Based on the 2D data structure, a revised region growing method dubbed "counted region growing" was developed to automatically cluster vehicles without prior knowledge of the number of vehicles. The counted region growing is featured by a combined region growing and connected component labeling, introduced in this section.

As the 2D structure is like Image data, each block can be treated as a pixel. Similar to region growing, two clusters are determined by the distance between the seed point and the neighbors. The criteria between two sets of clusters A and B are as follows [32] :

$$min\{d(a,b): a \in A, b \in B\} \quad (7)$$

where $d(a,b)$ is the distance between instances a and b that belong to clusters A and B, respectively.

Then, clustering can be performed according to the following steps:

1) *Find the seed point: set the seed point by searching the unlabeled blocks from the first to the last grid block until it finds one with a nonzero value.*
2) *Grow the region of seed point: search the neighbors of the seed point. If the difference between the values of the neighbors and seed point satisfies a given condition, set the neighbor as the new seed point and search again until none of the neighbors satisfies the condition.*
3) *Label the region: label each block in the same region and go back to step one, continue searching from the previous seed point.*
4) *Delete the unnecessary regions: after the search is completed for the entire grid, delete the regions whose number of data points is less than a threshold.*

It is notable that the location of any value in a matrix can be presented in either linear indexing or an index with two subscripts. For example, for a matrix A with 3 rows and 3 columns, *A(4)* is the same as *A(1,2)* since the columns are counted firstly. Detailed steps of the counted region growing method can be summarized as followed on pseudocode.

Since the detective zone of LiDAR is a circle or a semi-sphere that is continuous, once the searching area of seed point exceeds the row limitation - for instance, the searching area of *A(1799,13)* will exceed 1800, which is the boundary of the 2D grid - it should contain the rows that go back from the first row, vice versa. Figure 2 illustrates the flow chart of the proposed method. The reason why $GRID(X_i)$ is set to -1 after confirming that $grid(X_i)$ is 0 is to mark X_i as a block that has been searched to decrease computation time.

Ideally, the data points of a vehicle collected by a LiDAR sensor should be similar in value and close to each other in the 2D grid, as shown in Table I. In reality, however, some objects of certain materials may not reflect laser beams but rather absorb them, causing the collected distance value to be disconnected. Table II shows that the upper triangle and lower triangle should be one vehicle, however, the data are disconnected. Another example shown in TABLE III is that the data points collected from two vehicles are connected. In reality, they are two different vehicles (each color represents one single vehicle.)

Selection of the number of neighbors (n) for searching and the distance threshold (d) is critical to encounter the above two

problems. However, since the traffic flow varies, such as different headspaces or lane widths, and installations, different types of LiDAR or occlusion situations, the determination of n and d should be based on real conditions. Especially for n, the selection of n needs to consider the type of LiDAR sensor and the tradeoff between the searching area and the noise.

TABLE II
A SAMPLE OF DISCONNECTED DATA

x \ y R	…	13	14	15	…
…	…	…	…	…	…
597	0	0	10.80	10.88	0
598	0	0	0	10.77	0
599	0	10.84	0	0	0
600	0	10.85	10.65	0	0
601	0	10.76	10.66	0	0
…	…	…	…	…	..

The type of LiDAR sensor, and the actual field environment also affect the selection. As for d, a number less than 1.5 m (the general width of a vehicle) should be considered because larger values will increase the error of two adjacent vehicles being identified as one

TABLE III
A SAMPLE OF ADJACENT DATA OF TWO-VEHICLES

x \ y R	…	13	14	15	…
…	…	…	…	…	…
597	0	0	22.80	22.65	22.59
598	0	10.83	22.88	22.70	22.81
599	0	10.84	22.64	22.73	10.77
600	0	10.85	10.65	10.73	10.66
601	10.79	10.76	10.66	10.74	10.81
…	…	…	…	…	…

Initialize **Grid** to 0 with the same size as **grid**
Input the **distance_threshold** and **number_threshold**
Set **indicator** to 1
Initialize the **neighborhood_list**
While **block_counter** is less than the total_block of grid
  If **grid(block_counter)** is equal to 0
    Set **Grid(block_counter)** to -1
    Go to next iteration
  End
  If **Grid(block_counter)** is not euqal to 0
    Go to next iteration
  End
  Set **nerghbor_index** of **block_counter** by calling the function of **Getneighbor**
  (**Getneighbor** finds out the nerghbor's positions)
  Set **Grid(block_counter)** to **indicator**
  Set **Unsearched_block** to the positions where **Grid(nerghbor_index)** equal to 0
  If |**grid(Unsearched_block)** - **gird(block_counter)**| (difference) is less than **distance_threshold**
    Set **Seed** to the positions where the difference is less than **distance_threshold**
    Set **List_length** to the number of **Seed**
    Set **neighborhood_list(1:List_length)** to **Seed**
    Set **Grid(Seed)** to **indicator**
    Set **Seed_counter** to 1
    While **neighborhood_list** has non zero values
      ReSet **nerghbor_index** of **neighborhood_list(Seed_counter)**
      ReSet **Unsearched_block** to the positions where **Grid(nerghbor_index)** equal to 0
      If |**grid(Unsearched_block)** - **grid(neighborhood_list(Seed_counter))**| (difference) is less than **distance_threshold**
        ReSet **Seed** to the positions where the difference is less than **distance_threshold**
        Set **List_length_More** to the number of **Seed**
        Set **neighborhood_list(List_length:List_length_More)** to **Seed**
        Set **Grid(Seed)** to **indicator**
        Set **List_length** to **List_length** + **List_length_More**
        Set **neighborhood_list(Seed_counter)** to 0
        Add 1 to **Seed_counter**
      Else
        Set **neighborhood_list(Seed_counter)** to 0
        Add 1 to **Seed_counter**
      End
    End
    Add 1 to **indicator**
  Else
    Add 1 to **indicator**

Go to next iteration
End
End
Set **Grid(Grid==-1)** to 0
Set **Count_indicator** to the number of blocks that each indicator occupants
Set **Todelete_indicator** to the **indicators** whose **Count_indicator** is less than **number_threshold**
Set **Grid(Grid==Todelete_indicator)** to 0
Return Grid

B. Merging Process

Careful selection of the parameters is important but not enough to offset the errors caused by occlusion and other real-world situations such as the gap between a truck tractor and a semi-trailer; as shown in Table IV, in some cases, the detached blocks may be too large.

This section introduces a merging process to merge the disconnected data points when they belong to one object. In order not to compromise computational efficiency, the strategy of the proposed merging process is to only compare the nearest blocks of each cluster after the counted region grows. If the difference of the values of the nearest blocks is within the threshold, these clusters are regarded as one cluster

To ensure the separated clusters, namely A_1 and A_2, are from a single vehicle and to find the nearest blocks of the separated clusters, three conditions need to be examined for $A_1(x_{n1}, y_{n1})$ and $A2\ (x_{n2}, y_{n2})$:

TABLE IV
A SAMPLE OF UNCONNECTED DATA OF ONE-VEHICLES

x \ y	12	13	14	15	16
586	10.91	10.75	10.73	10.67	0
587	0	10.84	10.80	10.88	0
588	0	0	0	0	0
…	…	…	…	…	…
599	0	0	0	0	0
600	0	10.76	10.66	10.81	0
601	10.78	10.90	10.82	10.76	10.83

.

1) *The separated clusters have the common columns in the 2D grid. The common columns indicate the objects are fired by the same laser ID. The equation could be represented as follows.*

$$y_n = y_{n1} \cap y_{n2} \neq \emptyset \tag{8}$$

2) *The row intervals (the difference between rows) of the common columns are the nearest and within the threshold. Finding the nearest row interval improves the speed of the merging process, while the threshold guarantees that the clusters are close to each other. In detail:*

$$|x_1 - x_2| = min\ |x'_{n1} - x'_{n2}| \tag{9}$$

And

$$|x_1 - x_2| < T_r \tag{10}$$

where x'_{n1} and x'_{n2} are the largest row numbers or smallest row numbers corresponding to the common columns y_n for clusters A_1 and A_2, respectively. T_r is the threshold for row intervals.

Find y, i.e., the column where x_1 and x_2 exist.

3) *The difference of the values in the nearest blocks is within the threshold to verify if they belong to the same vehicle.*

$$|A_1(x_1, y) - A_2(x_2, y)| < T_d \tag{11}$$

where T_d is the threshold for the difference of the values in the nearest blocks. For clusters that do not have common columns, once the difference of the row numbers of their nearest blocks is within the T_r, the criterion of judging whether two clusters belong to one object is to compare the minimum and maximum value of each cluster. Again, once the difference of value either for minimum or maximum is within T_d, it is believed that these two clusters are one object.

V. EXPERIMENTAL STUDY

The data used in this study was collected by VLP-32 of Velodyne LiDAR sensors at the Veterans & Mira Loma intersection (Figure.2) at the City of Reno, Nevada. The portable LiDAR was equiped at the height [33] of 2 meters . The speed limit for Veterans road segment is 55 MPH while for Mira Loma road segment is 30 MPH. According to a study performed at a similar site, Veterans Pkwy & E Greg St intersection, average speed was about 15 MPH at the stop bar for northbound vehicles that slowed but did not stop.

Fig. 2. Veterans & Mira Loma site

Before converting 3-D data to the 2-D structure, background filtering was applied to filter the background data points from all frames, and the method was introduced in a previous study of the authors [6]. The algorithm was programmed in Matlab

and run on a laptop with Intel i7 CPU @2.6 GHz and 16 GB RAM. The total clustering time for 1000 frames was 10.644302s, which is almost 6 times faster than DBSCAN's 60.178293s.

In this study, 24 neighbors of a seed point are searched. As shown in Table V, the seed point highlighted in red is centered, while the gray blocks represent the search area of the seed point. 10 data points are the minimum requirement for a cluster, which means an object should contain at least 10 data points; otherwise, the object will be deleted.

TABLE V
SEARCH AREA OF THE SEED POINT

x \ y R	…	13	14	15	…
…	…	…	…	…	…
597	0	0	10.80	0	0
598	0	10.83	10.88	10.77	0
599	0	10.84	10.64	10.71	0
600	0	10.85	10.65	10.73	0
601	0	10.76	10.66	10.74	0

Table VI and Figure 3 are two ways to demonstrate the result of the merging process. From the data view, Table VI shows that the unconnected data of one vehicle in Table V is identified as one object. Figure 3 visualizes the results of the merging process. The two clusters (vehicle #1 and vehicle #2) in Figure 1 are merged into a single vehicle (vehicle #1).

TABLE VI
MERGING RESULT OF UNCONNECTED DATA OF ONE-VEHICLES

x \ y R	12	13	14	15	16
586	10.91	10.75	10.73	10.67	0
587	0	10.84	10.80	10.88	0
588	0	0	0	0	0
…	…	…	…	…	…
599	0	0	0	0	0
600	0	10.76	10.66	10.81	0
601	10.78	10.90	10.82	10.76	10.83

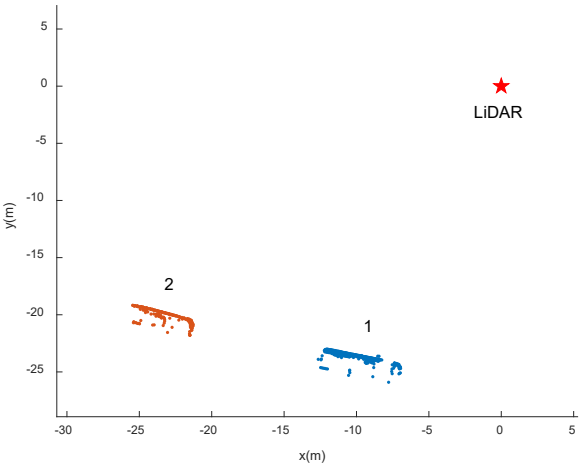


Fig. 3. The Result of The Merging Process

Constrained by the misalignment of LiDAR [34] and other conditions relevant to field implementation, such as roadway geometry and traffic conditions, there should be no universal threshold values that apply to all environments. It is found that based on the conditions our dataset was built, the method reaches the best performance when $d = 1, T_r = 20, T_d = 0.5$ (see Table VII). In Table VII, the numbers of vehicles identified by the proposed method with different parameters, namely the distance of counted region growing (d), the distance of merging process (T_d), the threshold for row intervals (T_r), are given. The results are compared to the number of vehicles in the ground truth. The ground truthground truth was obtained by manually counting the vehicles through Velodyne LiDAR visualization software (Veloview) using original .pcap file, which is 2170 vehicles in 550 frames. The rate here is just the total number of detected vehicles divided by the ground truth. The number of vehicles identified by the proposed method is close to the true number of vehicles since all rates are above 95%.

TABLE VII
NUMBER OF VEHICLES IDENTIFIED BY DIFFERENT PARAMETERS

	0-149 Frames	150-299 Frames	300-449 Frames	449-549 Frames	Total	Rate
d=1, T_r=20, T_d = 0.5	412	258	629	834	2133	0.9829
d=3, T_r=20, T_d = 0.5	416	256	647	813	2132	0.9824
d=1.5, T_r=20, T_d = 2.5	414	257	622	800	2093	0.9645
d=1.5, T_r=50, T_d = 0.5	415	258	626	829	2128	0.9806
d=2, T_r=30, T_d = 1.5	415	252	633	825	2125	0.9792
Ground Truth	414	256	647	853	2170	/

Note: d: distance of counted region growing. T_d: distance of merging process. T_r: threshold for row intervals

However, since the situation in which vehicles are divided into two objects due to large occlusion exists, the number of vehicles being identified correctly should be paid more attention. To further demonstrate the accuracy of the proposed method (d=1, T_r=20, T_d = 0.5), the numbers of vehicles being identified correctly are compared to the ground-truth numbers of vehicles in Table VIII.

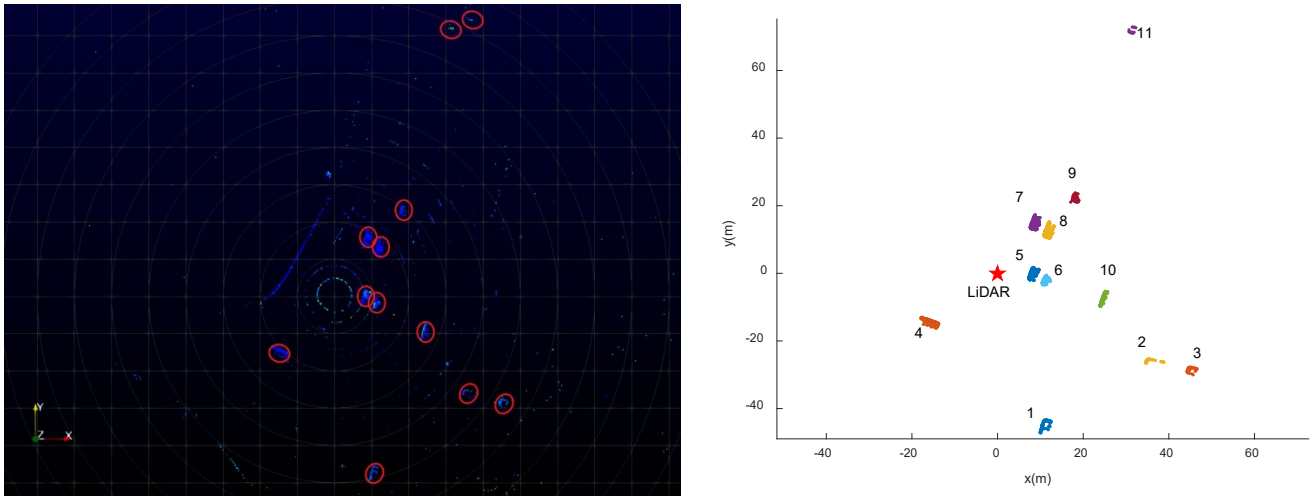
The accuracy in Table VIII is the ratio of the number of vehicles being identified correctly and the ground truth. The mean accuracy is 0.9668, as shown in Table VIII.

As stated in [6] although the detection range of LiDAR could reach 100m for a 16-laser LiDAR, the effective range is only 30m for high-quality detection.

TABLE VIII
ACCURACY RESULT

Frames	Ground Truth	Correctly Identified	Accuracy
0-49	158	158	1.0000
50-99	144	143	0.9930
100-149	112	110	0.9821
150-199	69	69	1.0000
200-249	85	84	0.9882
250-299	102	97	0.9510
300-349	125	114	0.912
350-399	249	239	0.9598
400-449	273	266	0.9743
450-499	338	321	0.9497
500-549	515	497	0.9650
Total	2170	2098	0.9668

In the case of 32-laser LiDAR sensors that are used in this study, 50-75m seems to be the upper limit for best detection quality, albeit the 200m range claimed by the manufacturer. Occlusion and the point density would be two major factors that influence the accuracy. For example, the error for 500-549 frames is mostly from the far vehicles that have too few points to identify, even for manual. As shown in Figure 4, the furthest vehicle is hard to recognize even for manual, but it is a vehicle because the vehicle will be shown clearly with the frame goes. The proposed method detects 11 vehicles since the furthest vehicle only has a few points. However, Figure 4 (A) and (B) shows a case with a vehicle about 60 meters away from the sensor, which was clearly identified (vehicle #11).



A. 12 vehicles in ground truth B. 11 vehicles are detected

Fig. 4. Clustering Result of Other Frames

VI. CONCLUSION

An unsupervised clustering method for roadside LiDAR application was presented. The proposed method is developed based on the region growing algorithm coupled with counted component labeling and a revised merging process. A major thrust of the study lies in its capability to improve computation speed and over-segmentation, which is critical to roadside LiDAR application. At a speed of 0.011s per frame, the algorithm lays a solid ground for real-world applications such as red-light running and jaywalking protections. The threshold values were determined by trials, which should be used as references rather than recommendations considering the variety of real-world situations. Studies that include nonmotorized road users such as pedestrians and cyclists are currently ongoing.

REFERENCES

[1] Gargoum, S. and El-Basyouny, K., 2017, August. Automated extraction of road features using LiDAR data: A review of LiDAR applications in transportation. In 2017 4th International Conference on Transportation Information and Safety (ICTIS) (pp. 563-574). IEEE.

[2] Lesani, A., Nateghinia, E. and Miranda-Moreno, L.F., 2020. Development and evaluation of a real-time pedestrian counting system for high-volume conditions based on 2D LiDAR. Transportation research part C: emerging technologies, 114, pp.20-35.

[3] Lee, J.S., Jo, J.H. and Park, T.H., 2019. Segmentation of vehicles and roads by a low-channel lidar. IEEE Transactions on Intelligent Transportation Systems, 20(11), pp.4251-4256.

[4] Lato, M., Hutchinson, J., Diederichs, M., Ball, D. and Harrap, R., 2009. Engineering monitoring of rockfall hazards along transportation corridors: using mobile terrestrial LiDAR. Natural Hazards and Earth System Sciences, 9(3), pp.935-946.

[5] Klautau, A., González-Prelcic, N. and Heath, R.W., 2019. LIDAR data for deep learning-based mmWave beam-selection. IEEE Wireless Communications Letters, 8(3), pp.909-912.

[6] Zhao, J., Xu, H., Liu, H., Wu, J., Zheng, Y. and Wu, D., 2019. Detection and tracking of pedestrians and vehicles using roadside LiDAR sensors. Transportation research part C: emerging technologies, 100, pp.68-87.

[7] Gao, H., Cheng, B., Wang, J., Li, K., Zhao, J. and Li, D., 2018. Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment. IEEE Transactions on Industrial Informatics, 14(9), pp.4224-4231.

[8] Zhao, J., Xu, H., Tian, Y. and Liu, H., 2020. Towards application of light detection and ranging sensor to traffic detection: an investigation of its built-in features and installation techniques. Journal of Intelligent Transportation Systems, pp.1-22.

[9] Lay-Ekuakille, A., De Tomasi, F., Perrone, M.R. and Trotta, A., 2003, May. Output characterization of ground-based and integrated optical sensor for retrieved aerosol error minimization. In Proceedings of the 20th IEEE Instrumentation Technology Conference (Cat. No. 03CH37412) (Vol. 2, pp. 1018-1021). IEEE.

[10] Nash, G. and Devrelis, V., 2020, October. Flash LiDAR Imaging and Classification of Vehicles. In 2020 IEEE Sensors (pp. 1-4). IEEE.

[11] Yahya, M.A., Abdul-Rahman, S. and Mutalib, S., 2020, November. Object Detection for Autonomous Vehicle with LiDAR Using Deep Learning. In 2020 IEEE 10th International Conference on System Engineering and Technology (ICSET) (pp. 207-212). IEEE.

[12] Zhang, Y., Yin, Y., Guo, D., Yu, X. and Xiao, L., 2014. Cross-validation based weights and structure determination of Chebyshev-polynomial neural networks for pattern classification. Pattern recognition, 47(10), pp.3414-3428.

[13] Ou, G. and Murphey, Y.L., 2007. Multi-class pattern classification using neural networks. Pattern Recognition, 40(1), pp.4-18.

[14] Zhong, C., Miao, D. and Wang, R., 2010. A graph-theoretical clustering method based on two rounds of minimum spanning trees. Pattern Recognition, 43(3), pp.752-766.

[15] Lidkea, V.M., Muresan, R. and Al-Dweik, A., 2020. Convolutional neural network framework for encrypted image classification in cloud-based ITS. IEEE Open Journal of Intelligent Transportation Systems, 1, pp.35-50.

[16] Champahom, T., Jomnonkwao, S., Chatpattananan, V., Karoonsoontawong, A. and Ratanavaraha, V., 2019. Analysis of rear-end crash on Thai highway: Decision tree approach. Journal of Advanced Transportation, 2019.

[17] Morsdorf, F., Meier, E., Kötz, B., Itten, K.I., Dobbertin, M. and Allgöwer, B., 2004. LIDAR-based geometric reconstruction of boreal type forest stands at single tree level for forest and wildland fire management. Remote sensing of environment, 92(3), pp.353-362.

[18] Tonini, M. and Abellan, A., 2014. Rockfall detection from terrestrial LiDAR point clouds: A clustering approach using R. Journal of Spatial Information Science, 2014(8), pp.95-110.

[19] Pham, D.T., Dimov, S.S. and Nguyen, C.D., 2005. Selection of K in K-means clustering. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 219(1), pp.103-119.

[20] Shin, M.O., Oh, G.M., Kim, S.W. and Seo, S.W., 2017. Real-time and accurate segmentation of 3-D point clouds based on Gaussian process regression. IEEE Transactions on Intelligent Transportation Systems, 18(12), pp.3363-3377.

[21] Williams, C.K. and Rasmussen, C.E., 2006. Gaussian processes for machine learning (Vol. 2, No. 3, p. 4). Cambridge, MA: MIT press.

[22] Song, X., Pi, R., Lv, C., Wu, J., Zhang, H., Zheng, H., Jiang, J. and He, H., 2021. Augmented Multiple Vehicles' Trajectories Extraction under Occlusions with Roadside LiDAR Data. IEEE Sensors Journal.

[23] D. H. Ballard and C. Brown, Computer Vision. Berlin, Germany: Springer Verlag, 1982.

[24] Hojjatoleslami, S.A. and Kittler, J., 1998. Region growing: a new approach. IEEE Transactions on Image processing, 7(7), pp.1079-1084.

[25] He, L., Chao, Y., Suzuki, K. and Wu, K., 2009. Fast connected-component labeling. Pattern recognition, 42(9), pp.1977-1987.

[26] Dillencourt, M.B., Samet, H. and Tamminen, M., 1992. A general approach to connected-component labeling for arbitrary image representations. Journal of the ACM (JACM), 39(2), pp.253-280.

[27] Wang, H., Wang, B., Liu, B., Meng, X. and Yang, G., 2017. Pedestrian recognition and tracking using 3D LiDAR for autonomous vehicle. Robotics and Autonomous Systems, 88, pp.71-78.

[28] Wu, B., Wan, A., Yue, X. and Keutzer, K., 2018, May. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In 2018 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1887-1893). IEEE.

[29] Bailey, T. and Nieto, J., 2008. Scan-SLAM: recursive mapping and localisation with arbitrary-shaped landmarks. In Robotics: Science and Systems Conference (RSS).

[30] Huang, S. and Dissanayake, G., 2007. Convergence and consistency analysis for extended Kalman filter based SLAM. IEEE Transactions on robotics, 23(5), pp.1036-1049.

[31] Chen, G., Wiede, C. and Kokozinski, R., 2021. Data Processing Approaches on SPAD-Based d-TOF LiDAR Systems: A Review. IEEE Sensors Journal, 21(5), pp.5656-5667.

[32] Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O.P., Tiwari, A., Er, M.J., Ding, W. and Lin, C.T., 2017. A review of clustering techniques and developments. Neurocomputing, 267, pp.664-681.

[33] Zhao, Junxuan, Hao Xu, Yuan Tian, and Hongchao Liu., 2020. Towards application of light detection and ranging sensor to traffic detection: an investigation of its built-in features and installation techniques. Journal of Intelligent Transportation Systems, 1-22.

[34] Oh, S., You, J.H., Eskandarian, A. and Kim, Y.K., 2021. Accurate Alignment Inspection System for Low-Resolution Automotive LiDAR. IEEE Sensors Journal, 21(10), pp.11961-11968.