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 with 128-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...
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.

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 ($\beta$), azimuth ($\alpha$), and R (distance between LiDAR and the data point).

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.

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
where,
\[ x = \frac{(\alpha + \text{offset})}{\alpha_{\text{resolution}}} \]  \hspace{1cm} (3)
\[ y = n \text{ when } \omega \text{ is fired by } \text{nth laser} \]  \hspace{1cm} (4)

It is notable that since laser emits in pairs, the precise horizontal angle is not the exact value of \( \alpha \); instead, an offset is needed to amend the value of \( \alpha \). Due to the effect from the offset, the value of \( \alpha + \text{offset} \) 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 \( \alpha + \text{offset} > 360 \):
\[ x = \frac{(\alpha + \text{offset} - 360)}{\alpha_{\text{resolution}}} \]  \hspace{1cm} (5)
If \( \alpha + \text{offset} < 360 \):
\[ x = \frac{(\alpha + \text{offset} + 360)}{\alpha_{\text{resolution}}} \]  \hspace{1cm} (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 \( \alpha_{\text{resolution}} \) is obtained to be 0.2 since the time for each fire sequence is 55.296\( \mu \text{s} \).

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 (\( \alpha_{\text{resolution}} = 0.2 \)-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>
<thead>
<tr>
<th>( x )</th>
<th>( y )</th>
<th>[...]</th>
<th>(13)</th>
<th>(14)</th>
<th>(15)</th>
<th>[...]</th>
</tr>
</thead>
<tbody>
<tr>
<td>597</td>
<td>0</td>
<td>0</td>
<td>10.80</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>598</td>
<td>0</td>
<td>10.83</td>
<td>10.88</td>
<td>10.77</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>599</td>
<td>0</td>
<td>10.84</td>
<td>10.64</td>
<td>10.71</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>600</td>
<td>0</td>
<td>10.85</td>
<td>10.65</td>
<td>10.73</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>601</td>
<td>0</td>
<td>10.76</td>
<td>10.66</td>
<td>10.74</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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]:

\[ \min \{d(a,b) : a \in A, b \in B\} \]  \hspace{1cm} (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 \( GRID(X) \) is set to -1 after confirming that \( grid(X) \) is 0 is to mark \( X \) 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

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y )</th>
<th>( R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>597</td>
<td>0 0 10.80 10.88 0</td>
</tr>
<tr>
<td>598</td>
<td>0 0 10.84 10.77 0</td>
<td></td>
</tr>
<tr>
<td>600</td>
<td>0 10.85 10.65 0 0</td>
<td></td>
</tr>
<tr>
<td>601</td>
<td>0 10.76 10.66 0 0</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE III
A Sample of Adjacent Data of Two-Vehicles

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y )</th>
<th>( R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>597</td>
<td>0 0 22.80 22.65 22.59</td>
<td></td>
</tr>
<tr>
<td>598</td>
<td>0 10.83 22.88 22.70 22.81</td>
<td></td>
</tr>
<tr>
<td>599</td>
<td>0 10.84 22.64 22.73 10.77</td>
<td></td>
</tr>
<tr>
<td>600</td>
<td>0 10.85 10.65 10.73 10.66</td>
<td></td>
</tr>
<tr>
<td>601</td>
<td>10.79 10.76 10.66 10.74 10.81</td>
<td></td>
</tr>
</tbody>
</table>

Initialize \( \text{Grid} \) to 0 with the same size as \( \text{grid} \)  
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 \( \text{grid}(\text{block_counter}) \) is equal to 0  
Set \( \text{Grid}(\text{block_counter}) \) to -1  
Go to next iteration  
End  
If \( \text{Grid}(\text{block_counter}) \) is not equal to 0  
Go to next iteration  
End  
Set neighbor_index of block_counter by calling the function of Getneighbor  
(Getneighbor finds out the neighbor's positions)  
Set \( \text{Grid}(\text{block_counter}) \) to indicator  
Set Unsearched_block to the positions where \( \text{Grid}(\text{neighbor_index}) \) equal to 0  
If \( |\text{grid}(\text{Unsearched_block}) - \text{grid}(\text{block_counter})| \) (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 \( \text{Grid}(\text{Seed}) \) to indicator  
Set Seed_counter to 1  
While neighborhood_list has non zero values  
ReSet neighbor_index of neighborhood_list(Seed_counter)  
ReSet Unsearched_block to the positions where \( \text{Grid}(\text{neighbor_index}) \) equal to 0  
If \( |\text{grid}(\text{Unsearched_block}) - \text{grid}(\text{neighborhood_list(Seed_counter))}| \) (difference) is less than distance_threshold  
ReSet Seed to the positions where the difference is less than distance_threshold  
ReSet List_length_More to the number of Seed  
Set neighborhood_list(List_length:List_length_More) to Seed  
Set \( \text{Grid}(\text{Seed}) \) 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 occupies

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 A1 and A2, are from a single vehicle and to find the nearest blocks of the separated clusters, three conditions need to be examined for A1(xr1, yr1) and A2 (xr2, yr2):

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.

\[ y_r = y_{n1} \cap y_{n2} \neq \emptyset \] (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:

\[ |x_1 - x_2| = \min |x_{n1} - x_{n2}| \] (9)

And

\[ |x_1 - x_2| < T_r \] (10)

where \( x_{n1} \) and \( x_{n2} \) are the largest row numbers or smallest row numbers corresponding to the common columns \( y_r \) for clusters A1 and A2, 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.

\[ |A_1(x_1, y) - A_2(x_2, y)| < T_d \] (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 equipped at the height 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.

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 VI and Figure 3 are two ways to demonstrate the result of the merging process. The two clusters (vehicle #1 and vehicle #2) in Figure 1 are merged into a single vehicle (vehicle #1).

Table VII shows that the unconnected data of one vehicle in Table V is identified correctly and compared to the ground-truth numbers of 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 with different parameters, namely the distance of counted region growing (d), the distance of merging process (Te), the threshold for row intervals (Tr), are given. The results are compared to the number of vehicles in the ground truth. The ground 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%.

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, Tr=20, Ta = 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

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

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).

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


