# GC-Net: Gridding and Clustering for Traffic Object Detection With Roadside LiDAR

Liwen Zhang, Jianying Zheng, Rongchuan Sun, and Yanyun Tao, *Soochow University, Suzhou, 215131 , China.*

*The emerging intelligent transportation systems puts higher demands on the collection and analysis of the traffic data. LiDAR can provide high-precision point clouds of traffic objects, making it a promising choice for the surveillance device. This article focuses on the traffic object detection with roadside LiDAR: estimating both positions and categories of them. To overcome the challenges posed by point clouds, we propose GC-net, which is based on a three-stage pipeline, including gridding, clustering, and classification. First, we design a one-to-one mapping on raw point cloud as data preprocessing, which transforms the data structure from the graph to the grid. Then, we propose an efficient clustering algorithm: Grid-Density-Based Spatial Clustering of Applications with Noise to search the traffic objects. It exploits index information in the grid data to simplify the computational complexity. Last, we train a CNN-based classifier to categorize the found objects by extracting the local features, which performs well even the global shapes are defective. It only employs object-wise supervision, which reduces the difficulty of creating datasets. Based on the point clouds collected in real urban traffic scenarios, comparative experiences show that the proposed GC-net achieves a superior performance both in detection accuracy and computational speed, which are significant indicators for the real-time traffic surveillance systems.*

The rapid development of the city has brought more pressure and challenges to the urban transportation systems, such as congestion and detour, which have become a thorny issue and drawn worldwide attention. The concept of intelligent transportation systems (ITS) is proposed to solve these problems and provide better services for traffic participants, including drivers, cyclists, and pedestrians.[1] Surveillance devices are significant road infrastructures in ITS. They can provide the vehicle with its surrounding road condition data through connected vehicle technologies, thus assisting driving and improving traffic safety and mobility. Moreover, traffic data collected by them is valuable for the study on the behaviors of traffic participants.[2,9] Therefore, high-precision and reliable surveillance devices are essential prerequisites for ITS. LiDAR is an advanced sensor for data collection, which has proven its worth in autonomous driving. Equipped with many laser emitters, LiDAR collects the data by emitting laser actively and receiving the signal reflected from the target. It can obtain the accurate position of the target in three-dimensional (3-D) space directly by calculating the delay from the launch of laser to the return. The advantages cater to the acquirements of ITS, e.g., long detection range, high precision, independence of ambient light, and all-time work, which have made LiDAR a promising choice for surveillance in the future.

1541-1672 © 2020 IEEE
Digital Object Identifier 10.1109/MIS.2020.2993557
Date of publication 13 May 2020; date of current version 14 September 2021.

Analysis of the point clouds collected by roadside LiDAR is critical for ITS, where the traffic object detection is a fundamental task. In this article, we focus on the traffic objects detection with roadside LiDAR: both locate and categorize them via the morphological features of the raw point clouds (as shown in Figure 1).

When it comes to the traffic object detection in LiDAR point clouds, some past works only employ the prior knowledge for the detection.[5] locates the moving objects by the relative displacement between the objects and the background, and then directly

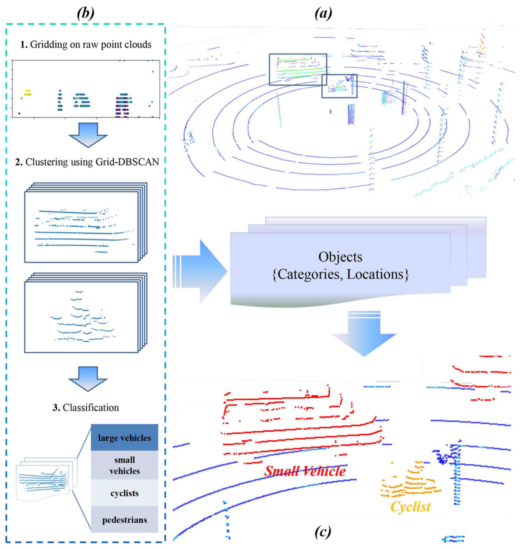

**FIGURE 1.** Pipeline of our works: (a) shows the raw point clouds collected by roadside LiDAR. The detection method, (b) estimates the information of traffic objects in the scene, such as category and location. Finally mark out them as shown in (c).

categorizes them only by prior information, such as a set of thresholds of size. Also, many kinds of hand-crafted features are proposed, which are based on projecting the raw point clouds onto a 2-D feature map.[6] matches the "L"-shaped features from the bird's view.[7] counts the changes in the height of the point clouds' lateral projection, and then uses SVM, decision tree or other algorithms for prediction. This pipeline is also applicable to other hand-crafted features commonly used in images, such as HOG,[12] etc.

However, simply using the prior information or hand-crafted features for detection does not accommodate the complex urban traffic scenarios. There are three challenges to be overcome: first, LiDAR has a low spatial resolution, resulting in that features in point clouds are scarce. The number of points in one flame collected by LiDAR is about 20000 to 100000, depending on the number of laser emitters equipped. It is far smaller than the pixels in one picture that are millions or more. Second, in the traffic scenarios, objects are occluded between each other and the features of point clouds are defective, which increases the difficulty for recognition. Third, it is difficult and laborious for researchers to create a 3-D point clouds dataset with the point-wise or voxel-wise ground truth.

For the first challenge, many works[6,7] project 3-D point clouds into a low-dimensional space to increase the density of data in disguise. It makes some of the semantic information lost during the simple projection or many-to-one mapping. However, some works[8,11] directly deploy the deep neural networks to extract complex features by learning from the raw point clouds data. Their success has proven that the semantic information of point clouds objects can be extracted directly from the Cartesian coordinate, which is the fundamental descriptor of point clouds.

For the second challenge, extracting the effective local features,[8] rather than the global features,[11] can reduce the errors caused by feature defects. Nowadays, convolution neural network (CNN) models have been applied in many state-of-art algorithms. Compared with the conventional hand-crafted explicit features, CNN could extract a large number of high-dimensional local features by weights-sharing convolution kernels and deep network structures. When objects are partially occluded, a considerable amount of local features can still be preserved in the rest of the point clouds, catering to the complex urban traffic scenarios.

For the third challenge, we divide the process of detection into clustering and classification. First search the potential objects via the density-based clustering, which is unsupervised. Then categorize these clusters by CNN-based classifier, which extracts semantic features. Only the latter needs object-wise annotation as a low-level supervision, which greatly reduces the threshold of making datasets for researchers.

However, unlike the images, whose pixels distribute in a grid structure, the arrangement of point clouds data is disordered, and the total number is indefinite. This structure of the point clouds, called graph, does not represent the spatial neighborhood relationship between points. Therefore, the convolution operation cannot be directly implemented, unless a proper processing to point clouds is performed.

In this article, we propose the GC-net. It refers to a three-stage pipeline, containing gridding, clustering, and classification. First, to make point clouds as a suitable input for CNN, we establish a one-to-one mapping from the Cartesian coordinate system to a spherical coordinate system centered on LiDAR. It converts the raw point clouds into a grid feature map, which realizes the transformation of data structure but preserves the original Cartesian coordinates as descriptors.

Then, the Grid-DBSCAN (abbreviated as GDN) is proposed as the clustering algorithm to accurately and quickly search the objects. Many works[3,7] tended to use the conventional dense-based clustering algorithms. They always have a high computational complexity, which is not conducive to the real-time algorithm. Benefit to the gridding preprocessing, GDN can directly determine the neighborhoods of points without a large amount of calculation of Euclidean distance by exploiting the index information of the grid. It significantly reduces the computational overhead of clustering.

Last, an efficient CNN model is implemented to predict the categories of clusters. We construct a variety of feed-forward paths in the network to make the model adapt to the point cloud data. Since the descriptors of point clouds are Cartesian coordinates, we can directly mark out both the position and categories of traffic objects in the original scene according to the result of classification.

Our experiments are based on the point clouds collected from the real urban traffic scenarios. LiDARs (using VLP16 in this work) are deployed as the surveillance on the roadside, collecting point clouds from various scenes including straight roads and intersections. Traffic objects are classified into four categories: large vehicles (e.g., vans and buses), small vehicles (e.g., cars), cyclists, and pedestrians. Statistics and analysis of experiments have proved that the proposed GC-net achieves superior results on both computation speed and detection accuracy in different scenarios and complexities, especially when the features are defective.

The key contributions of this article are threefold as follows.

1. We propose the GC-net, a CNN-based real-time detection method with road-side LiDAR, to predict both positions and categories of every traffic object in the point-cloud scene. It only needs the object-wise supervision, which greatly reduces the threshold of creating datasets for researchers.
2. We propose a preprocessing method on the raw point clouds collected from urban traffic scenarios, realizing the transformation for data structure from the graph to the grid. It makes the realistic point clouds a suitable input for the CNN model.
3. We design GDN, an efficient clustering algorithm that exploits both the index information of grid data and the prior knowledge of traffic scenarios. It greatly simplifies the computational complexity, which is indispensable for the real time.

## DATA COLLECTION AND PREPROCESS

Several open-source point cloud datasets have been created, but they are mostly oriented to autonomous driving or part detection of 3-D models. Therefore, it is necessary to establish a new dataset with road-side surveillance LiDARs for our work. LiDARs (we use VLP-16 in this article, whose major parameters are summarized in Table 1) are deployed at the roadside to collect point clouds from multiple urban traffic scenarios, including straight roads and intersections (as shown in Figure 2). To reduce the occlusion between traffic objects, LiDARs are erected at 1.8 m, higher than cars, cyclists, and pedestrians, lower than large vehicles, such as vans and buses. When collecting point clouds in straight roads, the scanning range is set as 180° facing to the road. When collecting point clouds in intersections, LiDARs are deployed at the corner, and the scanning range is set as 270°, covering the intersection.

**TABLE 1.** Major parameters of VPL-16.

| Horizontal field of view | 360° |
|---|---|
| Rotation rate | 10 Hz |
| Vertical field of view | 30°(±15°) |
| Laser emitters | 16 |
| Vertical angular resolution | 2° |
| Horizontal angular resolution | 0.2° |
| Wave length | 903 nm |
| Accuracy of point clouds | ±3 cm |

Since the location of the roadside LiDAR is fixed, the point clouds of the background are relatively constant, thereby can be easily filtered out, such as street lights, street trees, and guardrails. Background modeling[3,7] is applied as background filtering in this article, which is a classic and sufficient method for static background. We create the background model by superimposing the point clouds from 1000 random frames. In the model, the point cloud density of background areas is significantly higher than that of the foreground areas, such as lanes and sidewalks. Therefore, we can use it as a threshold to filter out the background points. Unlike the pixels in RGB images, the point clouds in the foreground and background do not overlap each other in the 3-D space. So this method can effectively separate the point clouds of the foreground from the background.

## METHOD DESCRIPTION

GC-net refers to a three-stage pipeline, containing gridding, clustering, and classification. Based on the gridding of data structure, we propose the GDN to cluster the point clouds into objects, and then build a CNN model to classify them. To this end, GC-net covers the following three main steps.

### Data Structure Transformation

CNN is typically applied to extract the semantic information of images, which can be represented by

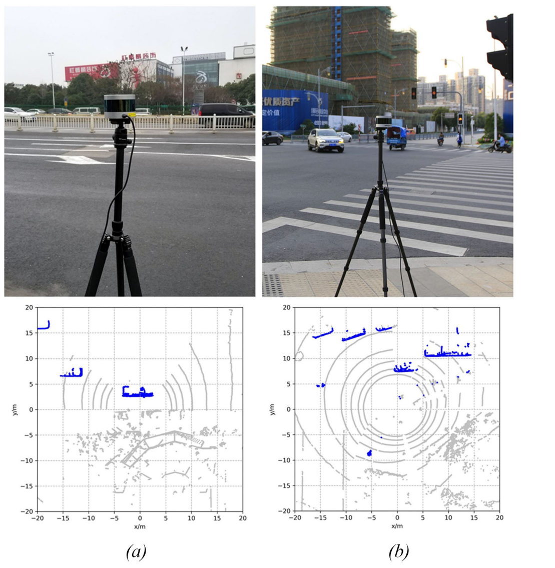

**FIGURE 2.** Two real urban traffic scenarios and the corresponding raw point clouds. Background are marked with sliver, traffic objects.

an $H \times W \times N$ tensor. Where *H* and *W* are the height and weight of the input feature, respectively, and *N* is the number of channels, such as $N = 3$ for the RGB pictures. This structure of data is called the "grid." In addition to the descriptors, each element in the grid has index information of the position itself, helping to easily find the adjacent elements, and then extract the local features from them.

However, the structure of point cloud's data is called "graph." It is just a set of Cartesian coordinates (*x,y,z*), which is disordered in the arrangement, and indefinite in total number. Compared with the gird, the elements in the graph cannot directly find their neighbors from the data structure. The only way to search the adjacent elements is calculating the distance from one to all of the others, which greatly increases the complexity of the algorithm.

Fortunately, point clouds collected by LiDAR have another representation in addition to Cartesian coordinates. Set the LiDAR as a particle, where all laser is emitted from, the Cartesian coordinate of each point can be uniquely transformed into a spherical coordinate system centered on LiDAR. The spherical coordinates corresponding to the original point clouds are $(d, \theta, \varphi)$, whose conversion formulas are

$$\begin{aligned} d &= \sqrt{x^2 + y^2 + z^2} \\ \theta &= \arcsin \tfrac{z}{d} \\ \varphi &= \arctan \tfrac{y}{x} \end{aligned} \tag{1}$$

where $\theta$ and $\varphi$ are the angle in the vertical and horizontal (azimuth), *d* is the distance to LiDAR. Since LiDAR is equipped with a certain number of laser emitters arranged at the same angular spacing in the vertical direction, which sample the scene at a fixed frequency in the horizontal direction, the point clouds have a certain resolution in both directions as described in Table 1. So the distribution of $\theta$ and $\varphi$ is discrete and even.

Therefore, we can build a feature map $F_{(i,a,N)}$, where *i* is the vertical angle which corresponds to the ID of laser emitter, *a* is the azimuth. *N* is the number of channels, for example, $N = 1$ when the descriptors are distance, or $N = 3$ when the descriptors are 3-D coordinates. If an element in the feature map does not get valid data, such as background and sky, the value of it can be set to 0 (in our work) or null as needed.

Compared to the voxelization or projection, which is a many-to-one mapping, since the transformation from Cartesian coordinates to the spherical is unique, the mapping between the "graph" and "grid" is one-to-one. Information contained in the data will not lose. It is worth noting that the data structure is transformed from graph to grid, though the features are not. Not to find new features, the essence of this transformation is to reorder the points and fill them in a grid, which determines the neighbor relationship containing the semantic information.

## Clustering the Points Into Individuals

In traffic scenarios, if special circumstances, such as accidents are not considered, traffic participants always tend to maintain a safe distance with each other, and the distribution of point clouds is obviously clusters in road space. Therefore, clustering algorithms can be implemented to divide the point clouds into objects.[3,7]

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)[4] is a classical clustering algorithm. It assumes that the clustering results can be determined by the density of samples, which can be characterized by a set of fixed neighborhood parameters ($\varepsilon$, *Mpt*). Where $\varepsilon$ is the neighbor radius, *Mpt* is the minimum number of points in the cluster. If a cluster contains less than *Mpt* points, it will be considered as noises.

Using DBSCAN to search the potential objects has the following advantages: first, it does not need to specify the number of objects in the scene in advance, second, based on the density, it could find clusters of any shape and remove noises in scenes. However, the conventional DBSCAN, which is designed for deploying on the graph-structure data directly, has to calculate the Euclidean distance from one point to all of the others. Therefore, it is unwise to use the conventional DBSCAN directly in this article because the computation expense is too high.

After transforming the point clouds to the grid structure, we propose the Grid-DBSCAN (GDN) to search for objects. It assumes that the adjacent points in the real-road space are definitely the neighbor elements in the corresponding grid, the other way around maybe not. The process of GDN is similar to the conventional, but the most obvious difference is that our algorithm searches a small neighborhood of an element in the feature map directly by the index information, and then uses a set of adjacency parameters to determine whether the elements in the neighborhood are really adjacent in real-road space. This design greatly reduces the complexity of calculation and enables the algorithm to cope with changing densities of point clouds.

**Algorithm 1:** Grid-DBSCAN (GDN)

```
1:  input: feature map $F_{(i,a,N)}$ consisting of p; neighborhood
           parameter $\{P, Neb, \varepsilon, Mpt\}$.
2:  Initialize: the set of core points: $\Omega\{q_1, q_2, \ldots\} = \emptyset$;
                the sets of clusters: $\Psi_k = \emptyset$; k = 1;
                counter = 0.
3:  for p in F:
4:    Determine p-centered $Neb_p$ through index information;
5:    for p' in $Neb_p$:
6:      if $\|P' - P\|_2 \leq \varepsilon$:
7:        counter = counter+1
8:      if counter $\geq$ Mpt:
9:        Add p into the core points set $\Omega$
10:   end for
11: end for
12: for q in $\Omega$:
13:   remove q from $\Omega$
14:   add q into the cluster $\Psi_k$
15:   for q in $\Psi_k$:
16:     for q' in $Neb_q$:
17:       if q' in $\Omega$ is true:
18:         remove q' from $\Omega$
19:         add q' into the cluster $\Psi_k$
20:     end for
21:   end for
22:   save $\Psi_k$ as cluster k
23:   k = k+1
24: end for
```

The full progress of GDN is shown in Algorithm 1. When dealing with the point *p*, the neighborhood parameters {*P*, *Neb*, $\varepsilon$, *Mpt*} are applied as the clustering parameters. *P* is the coordinates of point *p*. If

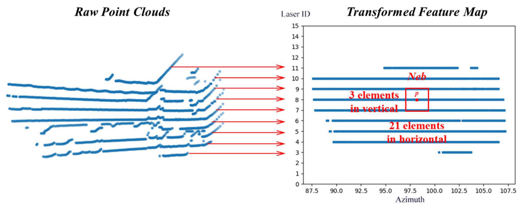

**FIGURE 3.** Visualization of the gridding on data structure. The range of a *Neb* in GDN is shown in the right. What we construct from the raw point clouds to the feature map is a one-to-one mapping; each point has a unique corresponding element in the feature map.

somewhere has no data returned, the *P* of these elements will be null, which does not participate in the calculation of clustering. *Neb* is the *p*-centered neighborhood in feature map, and the *p′* in *Neb* satisfies $\|P' - P\|_\infty \leq 2$, as visualized in Figure 3. Since the angular resolution of the LiDAR is $2^\circ$ in vertical and $0.2^\circ$ in horizontal (as listed in Table 1), the density of the elements in the horizontal direction is 10 times that of the vertical direction. If *p* is located at the boundary of the input feature map, the corresponding *Neb* will shrink adaptively. After determining the neighborhood relationship in the grid, $\varepsilon$ and *Mpt* are used for judging the real-spatial adjacency by only calculating the Euclidean distance between *p* and *p,′* which greatly simplifies the computational complexity.

## Classifier Based on CNN

Based on the results of GDN, it just needs to classify the objects we have found for the traffic object detection. CNN models have been widely used for the classification, detection, and segmentation tasks in image, which extracts the local features by weight-sharing convolution kernels. It can accurately recognize the pattern of objects even the global feature is broken, catering to the complex urban traffic scenarios.

The proposed classifier is built on a deep convolutional neural network structure, as shown in Figure 4. In theory, more layers the neural network has, more complex the features could be extracted, and better the performance our model should achieve. However in fact, simply and directly stacking layers cannot improve the performance, and even cause degradation. To solve it, the convolution layers are divided into many "Residual Blocks" which have familiar structures. They directly add the input *x* to the output *f(x)* via a feed-forward path. When the depth of structure exceeds the need of solution space, this design enables the conv-blocks to degrade to an identity

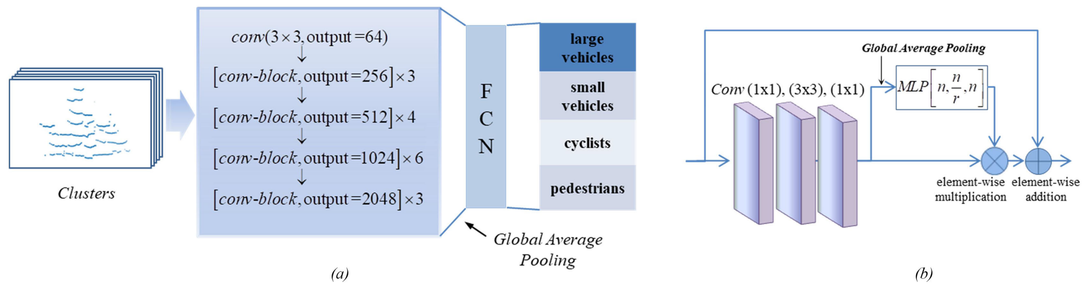

**FIGURE 4.** Macrostructure of GC-net (a) and the microstructure of a conv-block (b). Conv-Blocks share a similar structure to extract local features from the clusters achieved in GDN. "r" in MLP is a hyperparameter that is generally set to 16. The input and output of conv-block share the same size, so these similar blocks can be easily stacked together.

mapping (*f(x)* $= 0$) during training, rather than the degradation in performance. In other words, it makes our model depth-adaptive to the changeable real tasks.[13] At the same time, we build another feed-forward path with a concise multilayer perception (MLP) to explicitly model the interdependencies between the output channels of convolution, called "Squeeze-and-Excitation module."[10] It learns the weight $w_c$ of each channel, which can be considered as a type of feature learned by network, to recalibrates the channel-wise output by multiplying the corresponding $w_c$.

Fully connected networks is implemented to transforms the features extracted in the former into the probability distribution of the categories where the object should belong to. We use the softmax function to describe the predicted probability distribution of the categories

$$S_j = \frac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}} \tag{2}$$

where $S_j$ is the probability distribution for the categories predicted by model, *N* is the total number of categories, and $a_j$ is the activation value of the neuronal for the $j_{\text{th}}$ category. Then, the cross-entropy loss and stochastic gradient descent (SGD) is applied to evaluate the performance of model and optimize the parameters.

Last but not least, it is noteworthy that creating a training set for our network is very convenient. Since GC-net is built on clustering and classifier, we only need to feed it the point-cloud clusters with the corresponding object-wise labels during training, instead of laboriously marking the entire scene point by point.

## EXPERIMENTS AND ANALYSIS

Detection involves two equally important steps: searching and classification, which means the proposed GC-net should estimate both the positions and categories of every traffic object from the raw point-cloud scene. The searching depends on the efficient clustering algorithm GDN, whereas the classification relies on the CNN-based classifier. In order to evaluate the performance of the proposed method, we design several experiments based on the point clouds collected in the real-traffic scenarios.

### Clustering and Dataset Creation

GDN is proposed as a clustering algorithm. It has a set of parameters {*P*, *Neb*, $\varepsilon$, *Mpt*}, whose values will directly affect the accuracy of clustering. It should be pointed out that the {$\varepsilon$, *Mpt*} are hyperparameters, which need to be predefined manually before the machine-learning process. It is wise to use the prior knowledge in traffic scenarios to figure out the physical meanings of these hyperparameters and optimize their values. *Mpt* is the minimum number of points in the cluster, which can distinguish objects from noises. For catering to the varying density of point clouds, we set *Mpt* $= 10$. It means that we consider clusters with more than at least 10 points are traffic objects, otherwise are noises. $\varepsilon$ is the minimum distance between the traffic objects, which is a global parameter that does not change with distance or point clouds density. Clustering gets the best performance in our testing by setting $\varepsilon = 0.8$ m.

In experiments (see the results in Table 2 and the visualization in Figure 5), we use the *recall* under different thresholds of Intersection-over-Union (IoU) to evaluate the performance of several clustering algorithms. Compared with the conventional DBSCAN, GDN performs well in computation speed. The principles of both algorithms are similar, however, since GDN uses index information to quickly determine the neighborhood (Algorithm 1) rather than calculating the Euclidean distance from one point to all of the others. Therefore, the amount of computation is greatly reduced.

**TABLE 2.** Results of clustering under different thresholds of IoU.

| Method | Time (s) | Recall (%) under IoU = 0.95 | | Recall (%) under IoU = 0.7 | |
|---|---|---|---|---|---|
| | | Straight roads | Intersection | Straight roads | Intersection |
| GDN | 0.006 | 95.24 | 90.59 | 98.65 | 94.54 |
| DBSCAN[4] | 0.105 | 95.26 | 91.08 | 98.52 | 95.06 |
| K-means[14] | 0.009 | 91.10 | 77.62 | 95.52 | 81.61 |
| GMM[15] | 0.026 | 90.57 | 79.40 | 92.67 | 80.95 |
| Mean swift[16] | 0.037 | 89.02 | 68.54 | 93.13 | 84.54 |

K-means,[14] Gaussian mixture models (GMM),[15] and mean swift[16] are other popular clustering algorithms with lower computation complexity than the conventional DBSCAN. In experiments, GDN performs significantly better than them, especially in complex scenes. Clusters found by GDN can be any shapes, as opposed to K-means and GMM that assume that clusters should be convex. GDN can also filter the residual noises in the scene, but K-means and GMM will be seriously affected. Worst of all, these two algorithms require a prespecified and constant number of objects as the hyperparameter, which cannot be satisfied in reality. Another algorithm mean shift accommodates variable numbers and sizes of objects; it could also filter out noises. However, it is not adapted to the real-traffic scenarios, for example, the density centers calculated by mean swift often lie between two cars (as shown in Figure 5).

Based on the results of GDN, we create a dataset for the experiment in the next section, containing traffic objects, which have been manually identified and labeled. Traffic objects in our dataset are divided into four categories: large vehicles (vans and buses), small vehicles (cars), cyclists, and pedestrians. It has a total of 4550 objects, including 860 large vehicles, 1530 small vehicles, 1450 cyclists, and 710 pedestrians. This dataset will be applied to train the classifier in the next part.

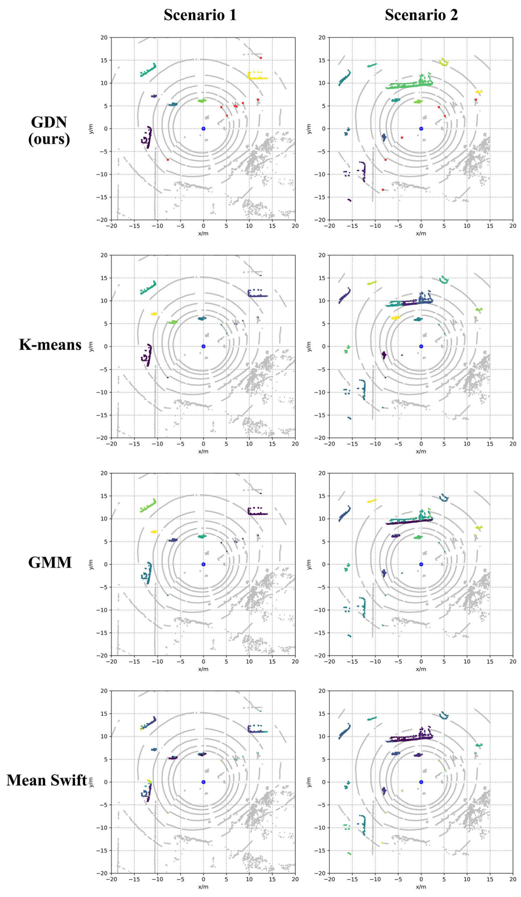

**FIGURE 5.** Comparison between different clustering algorithms. Different colors represent different clusters, which need to be further classified. The background points are annotated in silver, and noises are annotated in red.

## Evaluation on the Detection

We use the dataset created above to train the GC-net model. During training, SGD is implemented to optimize our model. The values of two hyperparameters of SGD: learning rate and weight decay are chosen by the random search, which has the best performance in training the network in 20 epochs. Finally, we set the learning rate 0.0035 with the weight decay 0.00005, and the learning rate decreases by 5% in each epoch. Our CNN model is trained for 50 epochs with the batch size of 64 on GPU: NVIDIA GTX1080Ti.

Several comparative experiments are conducted to evaluate the performance of GC-net under different difficulties. Each category of objects is independently divided into three difficulty levels according to the number of containing points: easy (top 20%), moderate (middle 60%), and difficult (last 20%). In easy cases, most methods can achieve great results, because the shapes of point clouds are relatively complete at this time. As the difficulty increases, the accuracy of detection may decrease. Therefore, how to overcome the sparseness caused by the long distance and occlusion is the key to our task.

Experiments (as described in Table 3) show that our GC-net achieves the best accuracy at all levels of difficulty (visualized in Figure 6). Our CNN model has a deeper network structure that could extract richer

**TABLE 3.** Results of detection under different difficulties.

| | Method | Average precision (%) | | | |
|---|---|---|---|---|---|
| | | Feature | Easy | Moderate | Hard |
| 1 | GC-net (GDN + improved CNN model) | | 95.82 | 89.14 | 68.55 |
| 2 | GDN + VGG16 | | 91.60 | 80.02 | 49.75 |
| 3 | 2-D Gaussian kernals[8] + improved CNN model | Local | 91.75 | 86.83 | 60.50 |
| 4 | Bird's-eye projection[7] + improved CNN model | | 90.45 | 72.33 | 45.16 |
| 5 | GDN + Pointnet[11] | | 94.61 | 80.40 | 52.85 |
| 6 | HOG[12]+SVM | Global | 90.19 | 74.08 | 33.97 |

features when facing difficult objects. For the ablation study, we design experiences **2, 3,** and **4**. Method **2** employs the VGG16 as the CNN model. It has a deep but plain structure without the paths we implemented between the convolution blocks in GC-net. The performance of VGG16 has an obvious degradation caused by the depth. The comparison between **1** and **2** proves that our design makes the CNN model more effective and depth-adaptive.

Methods **3** and **4** employ different methods to transform the raw point clouds; then use the same CNN model as GC-net. Method **3** projects the point clouds of each scan line of the LiDAR into a likelihood map, which is converted by the 2-D Gaussian kernels.[8] Method **4** uses a simple bird's-eye projection.[7] As a comparison, GC-net directly transforms the data structure of point clouds to the grid by a spherical coordinate system. The performance shows that our gridding transformation method could preserve more preciously original features.

Pointnet[11] used in experience **5** extracts the global features of 3-D point clouds by mapping them into a high-dimensional space. It directly takes the graph-structure data as input. In experience 6, the HOG features[12] of the lateral projection of point clouds are extracted for the classification based on SVM. Both methods extract the global features, and the experiments show that they perform poorly in the cases of sparseness, because the global shapes are seriously broken. As a comparison, the local features learned by CNN in GC-net could always show a strong representation of the semantic meanings of objects in high levels of difficulty.

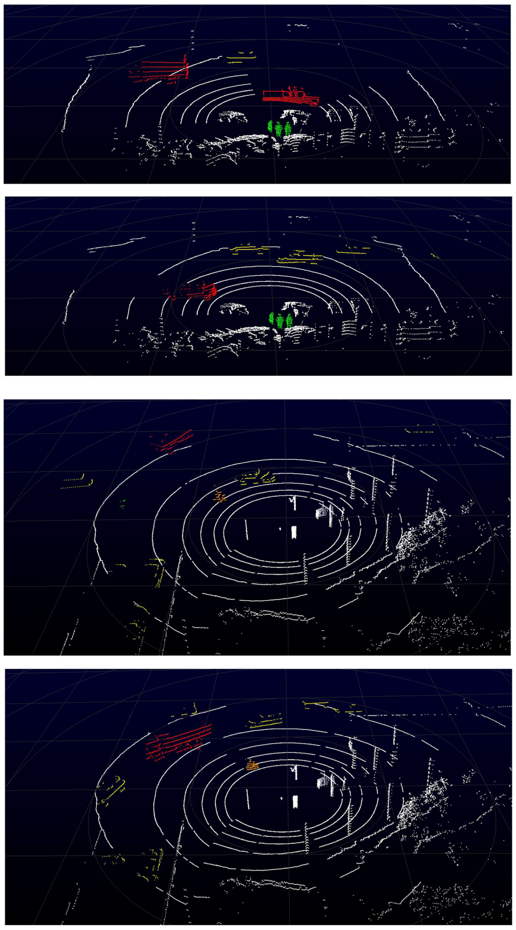

**FIGURE 6.** Visualization of traffic object detection under point clouds. Traffic objects are marked with different colors, where yellow represents large vehicles, red represents small vehicles, orange represents cyclists, and green represents pedestrians. Background points are annotated in sliver, noises are marked with blue.

## CONCLUSION

In this article, we present GC-net for the traffic object detection with roadside LiDAR. Our work consists of three main parts: First, we design a one-to-one mapping method to realize the gridding of data structure through a LiDAR-centric spherical coordinate system. Second, the GDN (Grid-DBSCAN) is proposed to quickly and preciously cluster the raw point clouds into individuals. And third, a CNN model is implemented for classification, which performs well in complex scenes. Experiments show that GC-net greatly reduces the computational cost compared with the conventional methods, which makes our algorithm suitable for real-time traffic surveillance systems. Also, it needs only object-wise supervision but performs well in complex scenes, and be robust to the low-quality point clouds. Further investigations include fusing the point clouds data collected by multiple LiDARs, and the reconstruction of the traffic scene from point clouds.

## ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 61973225, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20160324, and in part by the Natural Science Foundation of Jiangsu Colleges and Universities under Grant 16KJB580009.

## REFERENCES

1. J. Zhang, F. Wang, K. Wang, W. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1624–1639, Dec. 2011.
2. C. Vilarinho, J. P. Tavares, and R. J. F. Rossetti, "Design of a multiagent system for real-time traffic control," *IEEE Intell . Syst.*, vol. 31, no. 4, pp. 68–80, Jul./Aug. 2016.
3. Y. Cui, H. Xu, J. Wu, Y. Sun, and J. Zhao, "Automatic vehicle tracking with roadside LiDAR data for the connected-vehicles system," *IEEE Intell. Syst.*, vol. 34, no. 3, pp. 44–51, May/Jun. 2019.
4. M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithmfor discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, 1996, vol. 34, pp. 226–231.
5. H. Zhao *et al.*, "Detection and tracking of moving objects at intersections using a network of laser scanners," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 655–670, Jun. 2012.
6. H. Zhao, C. Wang, Y. Lin, F. Guillemard, S. Geronimi, and F. Aioun, "On-road vehicle trajectory collection and scene-based lane change analysis: Part I," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 1, pp. 192–205, Jan. 2017.
7. Z. Zhang, J. Zheng, H. Xu, X. Wang, X. Fan, and R. Chen, "Automatic background construction and object detection based on roadside LiDAR," *IEEE Trans. Intell. Transp. Syst.*, doi: 10.1109/TITS.2019.2936498.
8. Y. Tatebe, D. Deguchi, Y. Kawanishi, I. Ide, H. Murase, and U. Sakai, "Pedestrian detection from sparse point-cloud using 3DCNN," in *Proc. Int. Workshop Adv. Image Technol.*, 2018, pp. 1–4.
9. Q. Wang, J. Zheng, H. Xu, B. Xu, and R. Chen, "Roadside magnetic sensor system for vehicle detection in urban environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 5, pp. 1365–1374, May 2018.
10. J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf . Comput . Vis . Pattern Recognit .*, 2018, pp. 7132–7141.
11. R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf . Comput . Vis . Pattern Recognit .*, 2017, pp. 77–85.
12. Q. Yuan, A. Thangali, V. Ablavsky, and S. Sclaroff, "Learning a family of detectors via multiplicative kernels," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 3, pp. 514–530, Mar. 2011.
13. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput . Vis . Pattern Recognit .*, 2016, pp. 770–778.
14. P. K. Agarwal and N. H. Mustafa, "k-means: Projective clustering," in *Proc. ACM SIGMOD Conf. Princ. DB Syst.*, 2004, pp. 155–165.
15. S. J. Roberts, D. Husmeier, I. Rezek, and W. Penny, "Bayesian approaches to Gaussian mixture modeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 11, pp. 1133–1142, Nov.1998.
16. D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.
17. Z. Luo, M. V. Mohrenschildt, and S. Habibi, "A probability occupancy grid based approach for real-time LiDAR ground segmentation," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 998–1010, Mar. 2020.

**LIWEN ZHANG** is currently working toward the MS degree with the School of Rail Transportation, Soochow University, Suzhou, China. His research interests include machine learning and its applications in intelligent transportation systems. He received the BS degree from the School of Rail Transportation, Soochow University. He is the corresponding author of this article. Contact him at lwzhang96@stu.suda.edu.cn.

**JIANYING ZHENG** is an associate professor with the School of Rail Transportation, Soochow University. He received the BS degree from the University of Science and Technology of China and the PhD degree from Shenyang Institute of Automation, Chinese Academy of Sciences. His research interests include wireless sensor networks, intelligent transportation systems, and multiagent systems. He is the corresponding author of this article. Contact him at zhengjianying@suda.edu.cn.

**RONGCHUAN SUN** is an associate professor with the School of Mechanical and Electric Engineering, Soochow University, Suzhou, China. His research interests include robotic mapping, navigation, and path planning. He received the BS degree from the University of Science and Technology of China and the PhD degree from the Shenyang Institute of Automation, Chinese Academy of Sciences. Contact him at sunrongchuan@suda.edu.cn.

**YANYUN TAO** is an associate professor with the School of Rail Transportation, Soochow University, Suzhou, China. His research interests include intelligent computation and machine learning. He received the BS degree from Hohai University and the PhD degree from the East China University of Science and Technology. Contact him at taoyanyun@suda.edu.