Dynamic Vehicle Detection With Sparse Point Clouds Based on PE-CPD

Kaiqi Liu, Wenguang Wang, Ratnasingham Tharmarasa, and Jun Wang

***Abstract*— Detecting dynamic vehicles is of great significance in the field of autonomous vehicles. In the literature, a few vehicle detection methods are proposed to detect vehicles within 50 m from the Lidar, where the point clouds are relatively dense. It is a great challenge to detect vehicles that are far from the Lidar because of sparse point clouds. Fewer returned point clouds will result in a larger fitting randomness and lower detection rate. To tackle this issue, a dynamic vehicle detection method based on likelihood-field-based model combined with coherent point drift (CPD), which includes the steps of dynamic object detection and dynamic vehicle confirmation, is proposed in this paper. An adaptive threshold based on the distance and grid angular resolution is applied to detect the dynamic objects. The pose estimation based on CPD (PE-CPD) is proposed to estimate the vehicle pose. The scaling series algorithm coupled with a Bayesian filter that is improved by PE-CPD is utilized for updating the vehicle states. Finally, comparative experiments of vehicle detection based on KITTI data sets are conducted. The results show that the proposed method improves the detection rate, especially the detection in the radius of 40–80 m, which is termed as distant area, compared with the method based on pose estimation with modified scaling series.**

***Index Terms*— Autonomous vehicles, dynamic vehicle detection, sparse point clouds, pose estimation.**

I. Introduction

THE study of self-driving has become a research hotspot of intelligence technology in recent decades. To replace the drivers, a series of sensors, such as camera [1]–[3], radar [4]–[6] and Lidar [7]–[11], are assembled on the host vehicle for environment perception. Based on the information obtained from these sensors, the autonomous vehicles automatically plan the route to the intended destination. Because of the ability of high-resolution 3D mapping, the Lidar plays an important role in autonomous driving.

When the vehicle is on an urban road, there are mainly two kinds of obstacles around it. One is stationary objects, such as street lamps, trees, posts and parked vehicles. The other is dynamic objects, such as vehicles, pedestrians and bicycles. Stationary objects determination plays an important role in the field of Simultaneous Localization and Mapping (SLAM) that can generate a map of the environment and localizes the ego-vehicle [12]. Strongly coupled with SLAM, Detection and Tracking of Moving Objects (DATMO) is also at the foundation of the autonomous driving [13]. These moving objects in the environment have more randomness in positioning, which make the detection more difficult. In such a case, this paper intends to put emphasis on the dynamic vehicle detection, in which the motion information could be employed. Put aside the characteristics of dynamic vehicles, the mainstream vehicle detection methods can be classified as training-based method and model-based method.

A. Training-Based Method

Training-based method is a typical approach to detect objects. In [14], a robust vehicle detection method is proposed. In this method, three novel features: position-related shape, object height along length and reflective intensity histogram are used for training, and the kernel based Support Vector Machine (SVM) is used for classification. In [15], a supervised adaboost trained classifier is proposed for multiclass objects classification with some geometric features. The objects are tracked by integrating consecutive classification decision values. Reference [16] sums up 31 features for good classification performance. Radial basis function additive kernel SVM is employed to reduce the computation. A two-layer classifier is depicted in [17]. In this classifier, the first layer and the second layer are unsupervised and supervised, respectively.

Besides the traditional training-based method, deep learning has become a hot topic because of its high accuracy. In [18], energy minimization approach with Fast R-CNN pipeline is proposed and get good performance on the test set of the KITTI benchmark. In the moderate setting, the method gets an average precision of 88.64%. The Multi-View 3D network (MV3D), which fuses Lidar point clouds and images and gets excellent results in both 2D and 3D detection, is proposed in [19]. In [9], deep learning-based object appearance models and contextual scene analysis are used for object classification. The overall harmonic mean of precision and recall is 89%. In [20], statistical and geometrical features combined with the neural network classifier are utilized to detect vehicles. Numbers of objects including vehicles and non-vehicles in several different environments are labeled manually from KITTI datasets. 70% are used for training, 15% for validation and 15% for testing. The obtained classifier performance is above 98.3%.

Manuscript received December 12, 2017; revised May 21, 2018; accepted July 2, 2018. Date of publication September 18, 2018; date of current version May 1, 2019. This work was supported by the National Natural Science Foundation of China under Grant 61771028 and Grant 61671035. The Associate Editor for this paper was J. Li. *(Corresponding author: Wenguang Wang.)*

K. Liu, W. Wang, and J. Wang are with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China (e-mail: buaaliukaiqi@buaa.edu.cn; wwenguang@buaa.edu.cn; wangj203@buaa.edu.cn).

R. Tharmarasa is with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON L8S 4K1, Canada (e-mail: tharman@mail.ece.mcmaster.ca).

Digital Object Identifier 10.1109/TITS.2018.2857510

In general, the more data the experiments use, the higher the classification precision achieves [20]. However, it is hard to provide a wealth of training data sets for all kinds of scenario with a variety of complex obstacles in actual scene, which is a main limitation of the training-based method. In addition, computational complexity is also another issue. High computational resources are needed to get real-time performance.

B. Model-Based Method

In recent years, model-based method becomes an effective object detection method [21]. Compared to the former method, model-based method does not require a lot of training data. In [22], an object is modelled as a rectangular with few feature parameters including directional vectors, corner points, a center point and dimensions of two vertical sides. The object model is defined using the special characteristics of laser scanning and is used to support robust feature parameter extraction on partial observation data. In [23], the separable likelihood model and a geometric vehicle model constructed by three rectangles are utilized. By using these models, the tracking approach is able to fully use all available data and to cope with ambiguous situations or partial occlusion of objects. In [24], a fast model-based object tracking framework with Kalman Filter is introduced. A simple box model is used to fit the cluster and to deal with the splitting, merging and degradation. In [25], a search-based L-shape model is used for vehicle detection. The approach is computationally efficient and easy to implement for involving very few parameters. In [26], the box model is used to extract the centroid of the detected object and a Constant Velocity (CV) model is applied to capture the target trajectories. In [27], the vehicle is described by a boxcar model, whose motion of the center of gravity has to be estimated. An approach based on sequential Monte Carlo methods is proposed for the vehicle detection and tracking.

In [28]–[30], the polar angular grid map for dynamic object detection is introduced, and the vehicle measurement model that consists of three rectangles is established. The dynamic objects that satisfy the motion evidence theory and motion consistency are regarded as vehicles. Reference [21] modifies the polar grid map and proposes the likelihood-field-based vehicle measurement model to detect dynamic vehicles. It is shown that the fewer points the target vehicle contains, the less time the pose estimation takes, which is an advantage for real-time detection. However, as the number of points decreases, the pose estimation accuracy will drop. Because the fewer points cannot fit the model well, the obtained pose has a lot of randomness. In this case, the ego-vehicle cannot perceive the distant objects. This motivates us to research the detection of dynamic vehicles with sparse point clouds. In this paper, sparse point clouds refer to the points that are farther than 40 meters from Lidar.

To detect dynamic objects, existing methods use constant parameters [21], which cannot adapt well to the radial divergence of the Lidar. As for pose estimation of objects with sparse point clouds, the accuracy of Pose Estimation with Modified Scaling Series (PE_MSS) will drop. Because, the randomness of the particles will become large as the number of points decreases. In order to compensate for the initial error caused by the minimum area bounding rectangle process, large sampling range will be used in early iteration, which is likely to make the estimation fall into the local extremum so as to get inaccurate pose result. To tackle these issues, an adaptive threshold can be utilized. A pose estimation based on Coherent Point Drift (PE-CPD) is proposed for better fitting the vehicle measurement model. Three main contributions of the paper are listed below:

1) To improve the pose estimation, the PE-CPD algorithm, which is more suitable for pose estimation of vehicles with sparse point clouds, is proposed. Model selection and CPD are the foundation of PE-CPD.
2) To make the dynamic object detection more robust, an adaptive threshold technique, in which the parameters are adjusted with the distance and angular resolution, is proposed.
3) To improve the detection results in distant area, a novel detection method, which is based on the likelihood-field-based model combined with PE-CPD, is proposed.

The remainder of the paper is organized as follows. In section II, the detailed derivation and improvements of PE-CPD are described. In Section III, the implementation process of the vehicle detection method is depicted. The experiments are shown in section IV, in which the KITTI datasets are used to verify the validity of the proposed method. Finally, concluding remarks are given in section V.

II. Pose Estimation Based on CPD

Pose estimation can be utilized for vehicle detection [21]. Specifically, model-based detection methods usually use the fitting degree between the model and the point clouds to determine whether the point clouds represent vehicles. After fitting, the obtained poses in consecutive frames can be applied for confirming the set of point clouds, which is also called as motion consistency. Therefore, not only in the model fitting but also in the motion consistency, pose estimation is of great significance. One way for pose estimation is using several particles that are drawn randomly around the position of the interested object. Each particle represents a possible state of the object. By calculating the likelihood between the object and each particle, the one with the maximum likelihood is winnowed down to be the pose of the object. As for distant objects, fewer returned point clouds make the accurate pose estimation difficult due to the large randomness of the model fitting. In order to estimate the pose of the vehicle accurately, PE-CPD is proposed in this section.

The flow chart of PE-CPD is shown in Fig. 1. In this section, it is assumed that the two objects in CPD have been associated. The detailed association process is explained in section III.C. To estimate the vehicle pose, first, CPD with motion calibration is implemented on the associated objects. Second, the model mode is selected based on the motion information to obtain the initial pose of the object. Third, the particles are scattered near the initial pose to

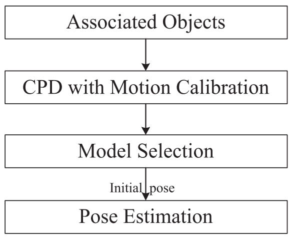


Fig. 1. The flow chart of PE-CPD.

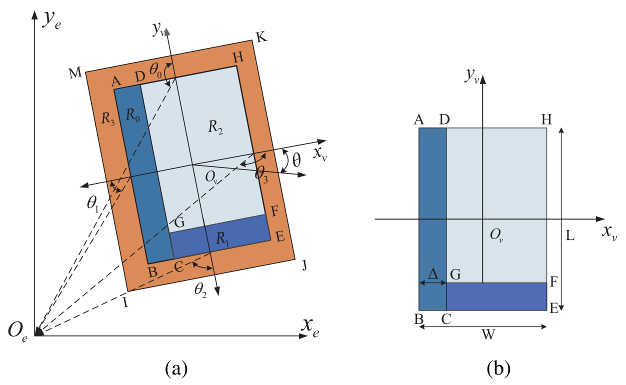


Fig. 2. The vehicle measurement model. (a) Ego-vehicle coordinate system $O_e x_e y_e$. R_0 and R_1 are the long side and wide side of the visible surfaces, respectively. R_2 and R_3 are inside area and outside area of the vehicle, respectively. (b) Target vehicle local coordinate system $O_v x_v y_v$. The length L and width W of the model are fixed to constants in this paper. The width of the visible side is Δ.

perform the pose estimation. The detailed processes are described below.

A. Measurement Model

In this paper, the vehicle measurement model proposed in [21] is utilized, which is shown in Fig. 2. The model is made up by four rectangles and is used for fitting the objects. $O_e x_e y_e$ is the ego-vehicle coordinate system. $O_v x_v y_v$ is the target vehicle coordinate system. In order to facilitate a unified description, it is specified that the y-axis of ego-vehicle local coordinate system points forward, x-axis points to the right side and z-axis points upward. The target vehicle coordinate system is specified with y-axis pointing to the invisible short side and z-axis pointing upward. The x-axis can be determined from y-axis and z-axis.

The center of the fitting model can describe the position of the vehicle and the orientation refers to the yaw angle. Considering that the vehicles are running on the ground, the state $X = (c_x, c_y, \theta)$ can be regarded as the pose of a vehicle regardless of the height. (c_x, c_y) and θ are the center and the yaw angle, respectively. The length and the width of the vehicle model are represented by L and W, which are assumed to be known in this paper. In Fig. 2(a), the *orientation vector* is oriented from the midpoint of each side to the origin O_e. The *normal vector* is perpendicular to each side and pointing outside.

It is assumed that the measurements are independent of each other [21]. The total number of measurement is n. Hence, the measurement set can be written as $B = \{q_1, ..., q_n\}$. The likelihood $p(B|X)$, which represents the fitting degree between the model and the point clouds, can be written as below.

$$p(B|X) = \prod_{i=1}^{n} p(q_i|X) \tag{1}$$

$$p(q_i|X) = \eta_i \exp(\alpha \sum_{j=0}^{3} c_j I(q_i^v, R_j)) \tag{2}$$

$$q_i^v = \begin{bmatrix} x_i^v \\ y_i^v \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_i - c_x \\ y_i - c_y \end{bmatrix} \tag{3}$$

where q_i^v is the transformation coordinate of q_i in vehicle coordinate system O_v. $I(q_i^v, R_j)$ is the integral of measurement q_i^v in the area R_j, $j = 0, 1, 2, 3$. R_j represents the regions with visible long side, visible wide side, inside of the vehicle and outside of the vehicle, respectively. c_j is the weight of the region. The weights $c_0 \sim c_2$ are positive while c_3 is negative. Considering that the visible sides, R_0 and R_1, play an important role in determining the vehicle pose, c_0 and c_1 can be set to larger values than c_2. The coefficient α is calculated by (4).

$$\alpha=1\Big/\sqrt{\sum_{j=0}^{2}\int\int_{R_j} c_j^2 dxdy} \tag{4}$$

For example, the integral $I(q_i^v, R_0)$ is calculated by (5)(6). The i-th measurement $q_i = (x_i, y_i, z_i)$ from the vehicle can be modelled as a two-dimensional normal distribution $p(x, y)$.

$$I(q_i^v, R_0) = \int_{x_B}^{x_C} \int_{y_B}^{y_A} p(x, y) dxdy \tag{5}$$

$$p(x, y) = \frac{1}{2\pi\sigma_1^2} \exp\{-\frac{(x - x_i)^2}{2\sigma_1^2}\} \exp\{-\frac{(y - y_i)^2}{2\sigma_1^2}\} \tag{6}$$

where (x, y) is the point in the xy plane. (x_i, y_i) is the projection coordinate of measurement q_i. σ_1^2 is the variance of measurement noise. It is easy to obtain the coordinates of vertexes $A(x_A, y_A)$, $B(x_B, y_B)$ and $C(x_C, y_C)$ based on (c_x, c_y), L, W and Δ.

More detailed explanation of the vehicle model and a fast look-up table process for integral can be found in [21].

B. Coherent Point Drift With Motion Calibration

Fewer measurements cannot fit the likelihood-field-based model well since the obtained pose has a lot of randomness. To improve the pose estimation, the point registration algorithm can be employed to obtain the direction information. CPD is a valid point clouds registration method, which can assign correspondences between two sets of points and recover the transformation that maps one point set to the other [31]–[33].

In the registration of standard CPD, the target first rotates to an appropriate angle and then translated to the position. In this case, the obtained rotation matrix R_v and the translation vector t_v from time step $k-1$ to time step k are relative to the origin O_e of ego-vehicle, which is shown in Fig. 3(a). To get the

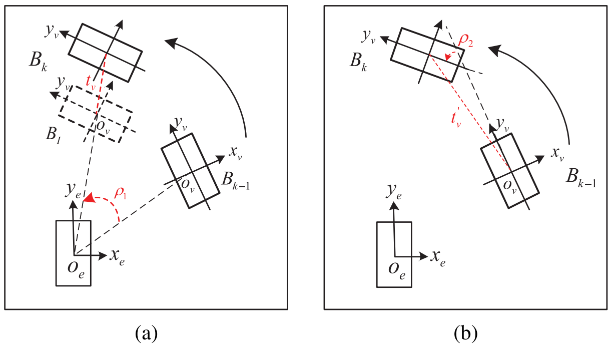


Fig. 3. Motion calibration. B_{k-1} and B_k represent the same target vehicle at consecutive time steps. (a) denotes the transformation process in standard CPD. The target at time step $k-1$ is rotated relative to the ego-vehicle and translated to the position at time step k. (b) is the transformation process in our motion calibration. The translation is relative to the ego-vehicle, and the rotation is relative to the target vehicle.

variation in yaw angle of the target vehicle and its possible motion direction, the motion calibration is used.

As shown in Fig. 3(b), the yaw angle of target vehicle changes around its center. $B_{k-1} = \begin{bmatrix} x_1^{k-1}, \ldots, x_m^{k-1} \\ y_1^{k-1}, \ldots, y_m^{k-1} \end{bmatrix}$ represents the vehicle point clouds at time step $k-1$. B_k is the point clouds of the same vehicle at time step k. $\overline{B}_{k-1} = \left[x_c^{k-1}, y_c^{k-1}\right]^T$ and $\overline{B}_k = \left[x_c^k, y_c^k\right]^T$ represent the geometrical centers. $R_v = \begin{bmatrix} \cos\rho_1 & -\sin\rho_1 \\ \sin\rho_1 & \cos\rho_1 \end{bmatrix}$ and $t_v = [x_t, y_t]^T$ are rotation matrix and translation vector relative to ego-vehicle, respectively. $R'_v = \begin{bmatrix} \cos\rho_2 & -\sin\rho_2 \\ \sin\rho_2 & \cos\rho_2 \end{bmatrix}$ is the rotation matrix relative to the target vehicle. ρ_1 and ρ_2 shown in Fig. 3 are rotation angles that are based on coordinate system O_e and O_v, respectively. Then, at current time step k, we can get

$$R_v B_{k-1} + t_v 1_m = R'_v[B_{k-1} - \overline{B}_{k-1}1_m] + \overline{B}_{k-1}1_m + t'_v 1_m \tag{7}$$

where

$$1_m = [1, 1, \ldots, 1_m] \tag{8}$$
$$t'_v = \overline{B}_k - \overline{B}_{k-1} \tag{9}$$

From (7) to (9), the rotation matrix R'_v can be obtained.

$$R'_v = [R_v B_{k-1} + t_v 1_m - t'_v 1_m - \overline{B}_{k-1}1_m][B_{k-1} - \overline{B}_{k-1}1_m]^{-1} \tag{10}$$

In a small time interval, the y-axis is approximately collinear with the translation vector t'_v. After motion calibration, the unit vector of t'_v, which is also the y-axis direction vector of the vehicle, can be regarded as a feature as in (11) and be used in PE-CPD. The rotation matrix R'_v can be used for motion consistency in dynamic vehicle confirmation.

$$D_v^y = \frac{t'_v}{\|t'_v\|} \tag{11}$$

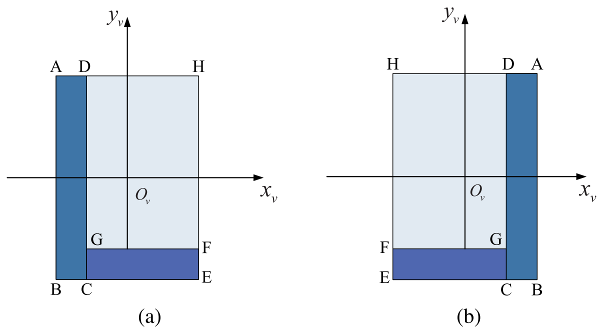


Fig. 4. The vehicle measurement model mode. (a) is Model I, (b) is Model II.

C. Vehicle Model Selection

In general, the vehicle has two visible sides. By calculating the minimum area bounding rectangle of the possible point clouds, the position and orientation can be obtained. Minimum area bounding rectangle is the rectangle that can surround point clouds in terms of the minimum area principle. However, if the vehicle has only one visible side for its special posture, or the vehicle only has a few returned points, the estimated orientation of the target will easily become inaccurate. For example, when the bounding rectangle of the visible part of object is exactly a square, the deviation between the estimated orientation and truth may be nearly 90 degrees. The large deviation may be made up by large sampling range in the early iterations and increasing particle amount. But the large sampling range may cause the pose estimation to deviate from the optimal position and fall into the local extremum. Increasing particle amount will increase the computational cost.

For the purpose of obtaining accurate pose estimation, a model selection approach is proposed in this section. Due to the asymmetry of the vehicle measurement model, two model modes are shown in Fig. 4. If a wrong mode is used for the pose estimation, the movement consistency will be disturbed and the target may be lost.

The proposed model selection approach is described in Algorithm 1. Inputs are the point set B and y-axis direction vector D_v^y, and outputs are the initial object center coordinate (c_x^{mb},c_y^{mb}), the direction angle θ^{mb} and the model mode.

First of all, the minimum area bounding rectangle is exploited for object B to obtain the coordinates of the four corners $X_{rec} = (x_{bl}, x_{br}, x_{ur}, x_{ul})$, $Y_{rec} = (y_{bl}, y_{br}, y_{ur}, y_{ul})$, which corresponds to line 1 in Algorithm 1. (c_x^{mb}, c_y^{mb}) is the center of the rectangle. Subsequently, function NormalVector is applied to get *normal vector* N_{vec} for each side of the rectangle. To pick out the visible sides, the angle between the *orientation vector* and *normal vector* of the same side is calculated. If the angle is greater than $\frac{\pi}{2}$, the side is invisible, otherwise the side is visible. For example, in Fig. 2, θ_1 and θ_2 are less than $\frac{\pi}{2}$, so the sides AB and BE are visible. The sides AH and EH are invisible because the angles θ_3 and θ_4 are greater than $\frac{\pi}{2}$. S_v is the set of visible sides. After getting the normal vectors N_{vec} and y-axis direction of the target vehicle by (11), the sides that are vertical to y-axis direction

Algorithm 1 Model Selection

```
Input: $B = (q_1, q_2, ..., q_n)^T$, $D_v^y = (x_v, y_v)$
 1: $(X_{rec}, Y_{rec}) \leftarrow \mathrm{MinAreaBoundRect}(Bx, By)$;
 2: $c_x^{mb} \leftarrow \mathrm{mean}(X_{rec})$, $c_y^{mb} \leftarrow \mathrm{mean}(Y_{rec})$;
 3: $N_{vec} \leftarrow \mathrm{NormalVector}(X_{rec}, Y_{rec})$;
 4: $S_v \leftarrow \mathrm{FindSideVisible}(X_{rec}, Y_{rec}, N_{vec})$;
 5: $VS_y \leftarrow \mathrm{FindVerticalSide}(D_v^y, N_{vec})$;
 6: if $(S_v \cap VS_y)$ then
 7:     $\theta^{mb} \leftarrow \mathrm{atan2}(y_v, x_v) + \pi/2$;
 8: else
 9:     $\theta^{mb} \leftarrow \mathrm{atan2}(y_v, x_v) - \pi/2$;
10: end if
11: $VS_x \leftarrow \mathrm{FindVerticalSide}(D_v^x, N_{vec})$;
12: if $(S_v \cap VS_x)$ then
13:     $mode \leftarrow Model$ II;
14: else
15:     $mode \leftarrow Model$ I;
16: end if
Output: $(c_x^{mb}, c_y^{mb}, \theta^{mb}, mode)$
```

can be obtained by function FindVerticalSide based on the angle between D_v^y and each vector in N_{vec}. The side with the smallest angle, which is nearly equal to zero, is the vertical side of D_v^y. In general, the pose estimation may be accurate with the two sides visible. But in some special orientations, the target vehicle only has one visible side, which is hard to get accurate pose estimation with a wrong model mode. This issue can be overcome by feature D_v^y.

The yaw angle of the minimum area bounding rectangle, θ^{mb}, can be obtained in line 6-10. If there are elements in the intersection of S_v and VS_y, θ^{mb} can be obtained by rotating 90 degrees counterclockwise from D_v^y. Conversely, the rotating direction is clockwise. Similarly, the model mode can be estimated by judging whether the vertical side of D_v^x is visible. If the vertical side is invisible, Model I is chosen to estimate the object, conversely, Model II will be chosen. D_v^x denotes the x-axis direction of the target vehicle.

$$D_v^x = (\cos(\theta^{mb}), \sin(\theta^{mb})) \tag{12}$$

Fig. 5 shows three different cases of model selection. The vehicles in (a)(b)(c) and (d)(e)(f) have two visible sides, while the vehicles in (g)(h)(i) have only one visible side. First, function MinAreaBoundRect is applied to get a rectangular box with minimum area that can surround the point clouds. The vehicle position can be obtained by (x_A, x_B, x_E, x_H) and (y_A, y_B, y_E, y_H). Next, four normal vectors are gotten with function NormalVector. With *orientation vector* and *normal vector*, the visible sides can be determined. In (a)(b)(c), the visible sides are AB and BE. Afterwards, the side AH that is perpendicular to D_v^y is extracted. Since AH is invisible, the y-axis of the vehicle has the same direction as D_v^y. Then, the x-axis can be obtained by rotating D_v^y 90 degrees clockwise. The side EH that is perpendicular to x-axis is invisible, hence the selected vehicle model is Model I. In (d)(e)(f), the visible sides are BE and EH. The side BE is perpendicular to D_v^y. Since BE is visible, the y-axis is opposite to D_v^y and the

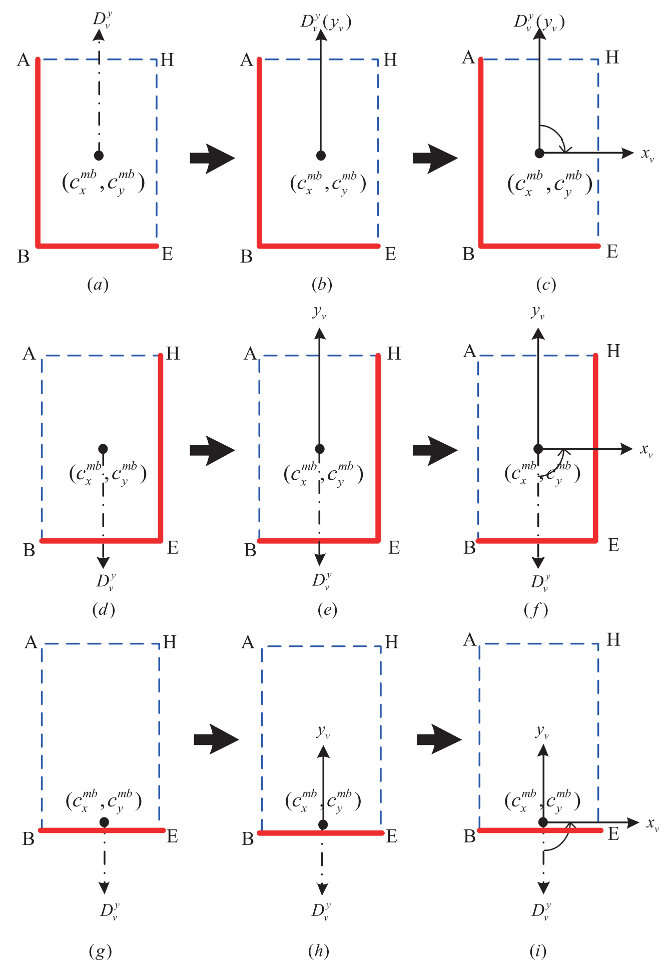


Fig. 5. Process of model selection. (a)(b)(c), (d)(e)(f) and (g)(h)(i) are three examples of model selection processes. The first two vehicles have two visible sides. The third vehicle only has one visible side. The red solid lines represent the visible sides and the blue dotted lines represent the invisible sides. The selected models are Model I, Model II and Model I, respectively.

x-axis is obtained by rotating D_v^y 90 degrees counterclockwise. The side EH that is perpendicular to x-axis is visible, hence the selected vehicle model is Model II. In (g)(h)(i), D_v^y is perpendicular to the visible side BE, so the y-axis points to the opposite side. The x-axis is obtained by rotating D_v^y 90 degrees counterclockwise. The selected vehicle model is Model I. In addition, it is obvious that the initial vehicle positions in (g)(h)(i) are not accurate with only one side visible. As the velocity judgment in motion consistency has a range of changes, the center deviation has little effect on vehicle detection.

D. Pose Estimation With CPD

The vehicle pose estimation in complex urban environments may inherit large uncertainties, which will be more serious if the returned points from objects are sparser. Fig. 6 shows the uncertainty of pose estimation with PE_MSS [21].

In Fig. 6, the vehicle is more than 35 meters away from the Lidar. Among 100 runs of experiments, we obtained three poses of the vehicle by PE_MSS. The occurrences of three different poses are 89 times, 8 times and 3 times as

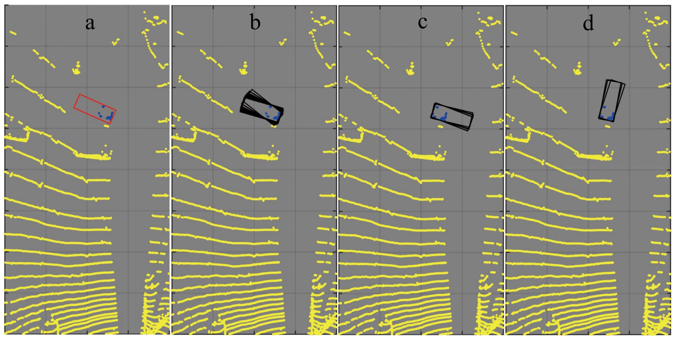


Fig. 6. Pose Estimation on the object that is more than 35 meters away from the Lidar with PE_MSS. The red box in (a) is the ground truth supported by KITTI. (b)-(d) are results of 100 repeated experiments. Among 100 experiments, 89 times we got the correct poses in (b) and 11 times we got the wrong poses in (c) and (d).

in Fig. 6(b), (c) and (d), respectively. From the ground truth, the pose in Fig. 6(b) is the correct one, hence the obtained accuracy is 89%. However, when conducting the motion consistency judgment, the vehicle will be confirmed if and only if its direction in three consecutive frames is consistent. Hence, the accuracy of the vehicle detection is reduced to $0.89^3 = 70\%$.

The pose estimation error caused by the sparse points can be compensated to some extent by adding initial pose feature, which can be obtained by CPD combined with motion calibration. A novel pose estimation algorithm called PE-CPD is proposed based on CPD and model selection. The algorithm is described in Algorithm 2.

Algorithm 2 The Pose Estimation with CPD (PE-CPD)

```
Input: $B = (q_1, q_2, ..., q_n)^T$, $N$, $\Delta$, $T_{PE}$, $L$, $D_v^y$, $zoom$
 1: $(c_x^{mb}, c_y^{mb}, \theta^{mb}, mode) \leftarrow$ ModelSelection($B, D_v^y$);
 2: Initialize $w$, $\psi$, $\sigma_2$, $Reg_0$;
 3: for $i \leftarrow 1 : T_{PE}$ do
 4:     $\overline{\chi_i} \leftarrow$ Even_Density_Cover($Reg_{i-1}, N$);
 5:     $\omega_i \leftarrow$ Compute_Normalized_Weights($\overline{\chi_i}, \sigma_2, B, 2w + \Delta, mode$);
 6:     $\chi_i \leftarrow$ Prune($\overline{\chi_i}, \omega_i$);
 7:     $Reg_i \leftarrow$ Union_Delta_Neighbourhoods($\chi_i, \omega, \psi$);
 8:     $w \leftarrow w * zoom$, $\psi \leftarrow \psi * zoom$, $\sigma_2 \leftarrow \sigma_2 * zoom$;
 9: end for
10: $\chi \leftarrow$ Even_Density_Cover($Reg_{T_{PE}}, N$);
11: $\omega \leftarrow$ Compute_Normalized_Weights($\chi, \sigma_2, B, \Delta, mode$);
Output: $\{\chi, \omega\}$
```

The inputs of Algorithm 2 contain a point set B and six parameters. $B_{n\times 3}$ is the data set of current time step k. N denotes the number of particles per ellipsoid field to be estimated. Δ specifies the width of the visible side. T_{PE} represents the number of iterations. L is the length of the vehicle. D_v^y is the y-axis direction vector of object obtained by motion calibration. The algorithm gives a set of weighted particles $\{\chi, \omega\}$ in the end.

First, the function ModelSelection is applied to obtain a set of initial pose parameters of the bounding rectangle and the model mode. The sampling region is centered in (c_x^{mb}, c_y^{mb}) with orientation θ^{mb}. Subsequently, in each iteration, N particles are distributed uniformly at position Reg_i and a set of particles $\overline{\chi_i}$ is obtained. Next, the likelihood of the measurement B is calculated at each pose in $\overline{\chi_i}$, here the model $mode$ is used to determine the integration domain in line 5. As a result of normalization, the likelihood ω_n can be regarded as the weight of the corresponding particle. In line 6, the particles with low weights are pruned. The remaining particles will decide the new sampling regions in the next iteration, which corresponds to line 7. The operation in line 8 represents the annealing. In an iteration, w and ψ are sampling ranges, which are centered on $(c_x^{mb}, c_y^{mb}, \theta^{mb})$ and will gradually reduce with the iteration processing. After T_{PE} iterations are finished, the particles are evenly distributed in area $Reg_{T_{PE}}$ for the last time, and a set of particles with corresponding weights $\{\chi, \omega\}$ are obtained. The particle with the maximum weight is considered to be the pose of the object.

The main distinction between PE_MSS and proposed PE-CPD is that applying CPD in pose estimation will reduce the sampling range of particles and improve the accuracy of estimation, especially the estimation of vehicle orientation. Both PE_MSS and PE-CPD obtain the pose estimation by drawing particles uniformly in the sampling region. The probability density function of the continuous uniform distribution is,

$$f(x) = \begin{cases} \dfrac{1}{b-a} & a \le x \le b \\ 0 & otherwise \end{cases} \tag{13}$$

where a and b are two boundaries. The variance of the distribution is

$$\begin{aligned} V(X) &= E(X^2) - [E(X)]^2 \\ &= \int_a^b x^2 \frac{1}{b-a} dx - \left(\int_a^b x \frac{1}{b-a} dx \right)^2 = \frac{(b-a)^2}{12} \end{aligned} \tag{14}$$

Therefore, the smaller the sampling range $[a, b]$, the higher the stability of pose estimation. It is assumed that the initial orientation sampling range is $[-\psi, \psi]$. After n iterations, the variance of the orientation estimation σ_θ is as shown in (15) considering the annealing processes.

$$\sigma_\theta = \frac{1}{2\sqrt{3}} \cdot 2\psi \cdot zoom^n \tag{15}$$

In PE_MSS, the implement of minimum area bounding rectangle cannot accurately determine the object pose, hence the initial orientation sampling range is set to be $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. After 9 iterations, the variance of the orientation estimation σ_θ is

$$\sigma_\theta = \frac{1}{2\sqrt{3}} \cdot 2 \cdot \frac{\pi}{2} \cdot \left(\frac{1}{\sqrt[3]{2}}\right)^9 = \frac{\pi}{16\sqrt{3}} \tag{16}$$

In PE-CPD, the approximate translation vector and rotation angle will be obtained from CPD with motion calibration. As shown in Fig. 7, B_k and B_{k+1} represent the vehicle at time step k and $k+1$, respectively. The change of yaw angle of vehicle is $\Delta\theta$. Considering that the y-axis of target vehicle

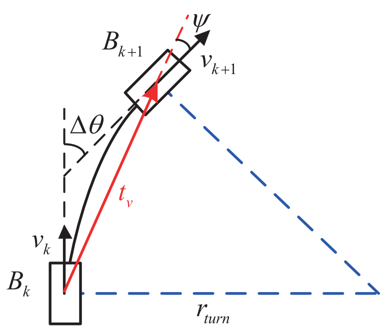


Fig. 7. Sampling range calculation after CPD. The black solid line is the driving route. The blue dotted lines r_{turn} indicates the equivalent turning radius. The red vector represents the translation vector. $\Delta\theta$ is the change of yaw angle. ψ is the deviation between translation vector and the yaw angle at time step $k + 1$, which can be used as sampling range.

is obtained by the translation vector t_v, the initial sampling range can be set to $\psi = \frac{1}{2}\Delta\theta$.

$\Delta\theta$ can be estimated from vehicle parameters and some driving common sense. The minimum turning radius of Hyundai Motors Veracruz in [34] is $5.7m$. It is assumed that the vehicle passing a 90-degree corner (for example, turn at a crossroad) at a speed of $30km/h$. The travel time approximately equal to $1.07s$. The change rate of vehicle yaw angle is $1.46rad/s$. In case that the scanning frequency of the Lidar is 10 Hz, we can get $\psi \approx 0.073rad$. Hence, we use $\psi = \frac{\pi}{36}rad$ as the initial sampling boundary in the experiment. After the same 9 iteration process, the variance of the orientation estimation will become

$$\sigma_\theta = \frac{1}{2\sqrt{3}} \cdot 2 \cdot \frac{\pi}{36} \cdot (\frac{1}{\sqrt[3]{2}})^9 = \frac{\pi}{288\sqrt{3}} \quad (17)$$

Therefore, the accuracy of pose estimation by PE-CPD is higher than PE_MSS. The proof of convergence of Scaling Series algorithm, which is the basis of the pose estimation, is given in [35].

III. Vehicle Detection Method

To deal with the vehicle detection issues caused by sparse point clouds, a novel method is proposed in this section. All operations are carried out in the ego-vehicle's local coordinate system. The origin is located in the center of ego-vehicle. Whenever a new dynamic object appears in the scene, PE-CPD is employed to estimate the pose and the motion consistency is utilized in three consecutive frames to determine whether the object is a vehicle. After vehicle confirmation, the scaling series algorithm coupled with a Bayesian filter (SSBF) with PE-CPD is applied to update the pose of the vehicle in a new scan.

A. Algorithm Overview

The framework of the proposed detection method is shown in Fig. 8. The method mainly contains ground tracking, clustering, virtual scan mapping, dynamic object detection and dynamic vehicle detection. When the point clouds data is received, ground tracking algorithm in [32] is applied for

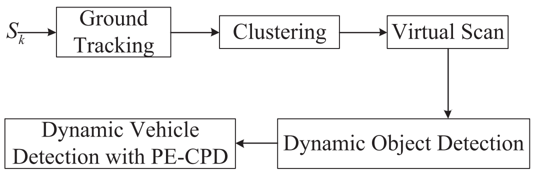


Fig. 8. Flow chart of dynamic vehicle detection with PE-CPD.

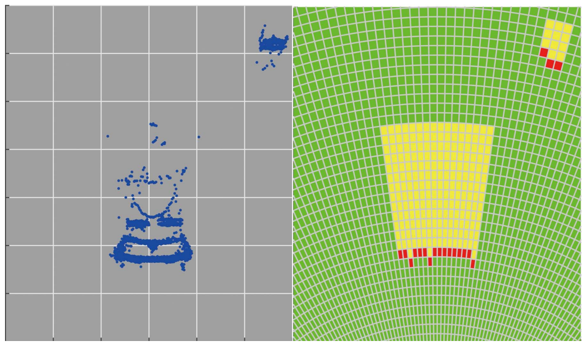


Fig. 9. Example of virtual scan map. The left part is the point clouds, and the right part is the virtual scan map. The green bins in virtual scan represent free states. The red bins and the yellow bins represent occupied states and occluded states, respectively.

ground elimination. RBNN clustering is utilized for clustering the remaining point clouds [36]. After clustering, all clusters are projected on the virtual scan map. Virtual scan is a kind of grid map in polar coordinate, which is consistent with the scanning characteristics of the Lidar [21].

Difference between two consecutive frames is implemented to detect the dynamic objects. Since the points of the two frames can be translated to the same coordinate system with the GPS-INS, the difference map can be obtained by comparing the consecutive frames of virtual scans directly. After pose estimation, motion consistency is employed to confirm the potential vehicles. If the object in three consecutive frames has consistency, the object will be confirmed as a vehicle. For the detected vehicles, the SSBF algorithm improved by PE-CPD is utilized to estimate the pose of the vehicle at the next time step.

B. Dynamic Object Detection

After ground elimination and clustering, a number of clusters, $O_k = \{o_k^1, o_k^2, ..., o_k^n\}$, are obtained at time step k. Virtual scan is applied for dynamic objects detection in this section. The surrounding space of ego-vehicle is divided into several angle grids with the same central angle. Each fan is called a segment. Then a fixed length, which is determined empirically, is utilized to divide each segment into multiple bins in the radial direction. According to the points in the bin, the grid state is labeled to be free, occupied or occluded, as shown in Fig. 9.

At the beginning, all bins are initialized as free space. For each object o_k, the furthest bin b^{max} that o_k covers is calculated. For each segment j that is covered by the object

o_k, the nearest bin b_j^{min} will be labelled as occupied. The bins between b_j^{min} and b^{max} will be labelled as occluded [21].

After constructing the virtual scan, the difference between two consecutive frames is used to detect dynamic objects. In order to detect the difference, two consecutive point clouds sets should be transformed into the same coordinate system. From the GPS-INS, we can get the translation vector t^{k-1}, roll angle γ^{k-1}, pitch angle φ^{k-1} and yaw angle ϕ^{k-1} at time step $k-1$ and t^k, γ^k, φ^k, ϕ^k at time step k. The transformation matrix T_c is

$$T_c = \begin{bmatrix} R^k & t^k \\ 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} R^{k-1} & t^{k-1} \\ 0 & 1 \end{bmatrix} \tag{18}$$

$$R^k = R(\gamma^k) * R(\varphi^k) * R(\phi^k) \tag{19}$$

In (19), $R(\gamma^k)$, $R(\varphi^k)$ and $R(\phi^k)$ denote the rotation matrices caused by γ^k, φ^k and ϕ^k, respectively. R^k represents the rotation matrix of ego-vehicle. The rotation matrix and the translation vector are combined to obtain the transformation matrix T_c. By using T_c, the point clouds in U_{k-1} can be translated into the coordinate system $O_e x_e y_e z_e$ at time step k.

$$U_{k-1}^* = T_c \cdot U_{k-1} \tag{20}$$

After coordinate transformation, for each object o_k in scan U_k, the changed bins from time step $k-1$ to time step k are employed for dynamic object detection. The dynamic object detection in reference [21] is conducted with constant threshold. But for Lidar data, the same object will return different illuminated points from different distances. Constant threshold is not effective to detect dynamic objects. If a large threshold is utilized, the objects far from the Lidar will be easy to lost. Conversely, it is inevitable to push more clusters into the dynamic objects set, which will cause false alarms and increase computational cost. In order to detect dynamic objects in different distances, a novel threshold based on distance and grid angular resolution is utilized. Generally, the states of the bins in the front and the rear of the vehicle will change as the vehicle moves. Assuming that the vehicle is running at speed v that is the absolute speed relative to the ground, the amount of changed bins Γ can be approximated by (21). Considering the self-occlusion and possible velocity of vehicle, the single-layer grid occupied by the short side of vehicle is utilized to define the threshold as in (22).

$$\Gamma \approx < \frac{2 \cdot W \cdot v \cdot \delta k}{l \cdot A_r \cdot \|o_k\|} > \tag{21}$$

$$Th_d = < \frac{W}{A_r \cdot \|o_k\|} > \tag{22}$$

where W is the width of the vehicle. l is the bin length along the radial direction of virtual scan. A_r is the angular resolution of the virtual scan. $\|o_k\|$ represents the Euclidean distance between the object and the Lidar. $< \cdot >$ denotes rounding up operation, since the threshold of changed states needs to be an integer. If the change in the object state is more than Th_d, the object o_k is detected to be a dynamic object.

C. Dynamic Vehicle Detection

The detection results of the last section include dynamic vehicles and false alarms caused by noise or disturbance. For example, the leaves and shrubs may cause false alarms if they are affected by the wind. In order to separate real dynamic vehicles from disturbances, the motion evidence theory and motion consistency can be used. In [28], motion evidence theory considers the regions cleared by the vehicle as it moves, which means that the cleared area behind the vehicle should be occupied in the prior frame and free in the current frame due to the high scanning rate. Similarly, the area in front of the moving vehicle should be free in the prior frame and occupied in the current frame. Motion consistency means that the velocities and orientations of the same vehicle in two consecutive frames should be similar [21]. For the distant objects, the pose estimation may not be accurate because of the sparse point clouds. Therefore, PE-CPD algorithm is proposed to estimate the pose of distant vehicles.

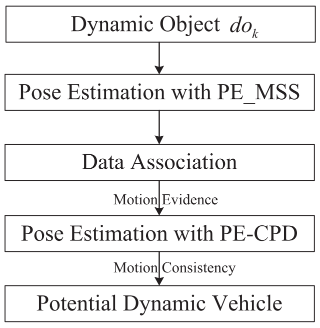

Dynamic Object do_k
Pose Estimation with PE_MSS
Data Association
Motion Evidence
Pose Estimation with PE-CPD
Motion Consistency
Potential Dynamic Vehicle

Fig. 10. Framework of dynamic vehicle detection with PE-CPD.

Fig. 10 shows the process of vehicle detection. For a dynamic object do_k, we temporarily use PE_MSS to fit the object and obtain the estimated pose $X_k = (c_{xk}, c_{yk}, \theta_k)$. Subsequently, the pose X_k is applied to match the associated object in scan U_{k-1}. Along the direction θ_k, a velocity is sampled from a uniform distribution $v = [v_1, v_2]$. Once there is an associated object do_{k-1} in scan U_{k-1}, the motion evidence theory is used to determine whether the associated objects in scan U_k and U_{k-1} meet the motion evidence. If the motion evidence theory is satisfied, CPD is conducted to determine the translation vector t_v and the change of yaw angle $\Delta\theta$ of the object from time step $k-1$ to time step k. Combined with the feature D_v^y by CPD, PE-CPD is implemented on the object at time step $k-1$ to obtain the yaw angle θ_{k-1}.

After obtaining θ_{k-1}, if the difference between θ_k and θ_{k-1} is larger than a predefined threshold and does not satisfy the motion consistency, the object is detected as a false alarm. Otherwise, the pose estimation results are used to calculate the velocity of the object.

A similar process is applied on the dynamic object do_{k+1}. If the velocity, the yaw angle of the object and the rotation angle obtained by rotation matrix in CPD satisfy the motion consistency, the object do_{k+1} can be marked as a vehicle.

D. State Update

For the detected vehicles, the poses are updated by SSBF algorithm with PE-CPD in the new received scans. The state updating process is shown in Fig. 11.

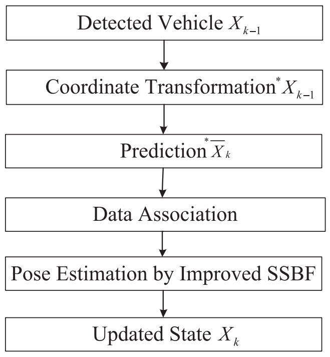

Detected Vehicle X_{k-1}
Coordinate Transformation $^*X_{k-1}$
Prediction $^*\overline{X}_k$
Data Association
Pose Estimation by Improved SSBF
Updated State X_k

Fig. 11. Framework for updating the states of detected vehicles.

In Fig. 11, the state of the detected vehicle at time step $k-1$ is $X_{k-1} = (c_{x_{k-1}}, c_{y_{k-1}}, \theta_{k-1})$. When a new scan is received at time step k, the coordinates of X_{k-1} are transformed into the coordinate system $O_e^k x_e^k y_e^k$ and state $^*X_{k-1} = (^*c_{x_{k-1}}, ^*c_{y_{k-1}}, ^*\theta_{k-1})$ is obtained. The predicted state of $^*X_{k-1}$, $\overline{X}_k = (\overline{c_{x_k}}, \overline{c_{y_k}}, \overline{\theta}_k)$, can be obtained from the following equations.

$$\overline{c_{x_k}} = {}^*c_{x_{k-1}} - \overline{v}_{k-1} * \delta k * \sin(^*\theta_{k-1}) \tag{23}$$

$$\overline{c_{y_k}} = {}^*c_{y_{k-1}} + \overline{v}_{k-1} * \delta k * \cos(^*\theta_{k-1}) \tag{24}$$

$$\overline{\theta}_k = {}^*\theta_{k-1} \tag{25}$$

where δk is scanning interval. $\overline{v}_{k-1}$ is the estimated velocity at time step $k-1$.

The objects in the rectangle area with center $\overline{X}_k = (\overline{c_{x_k}}, \overline{c_{y_k}}, \overline{\theta}_k)$, length L and width W are associated with the state $^*X_{k-1}$. After combining, the objects are represented by B_k.

Subsequently, the SSBF algorithm improved by PE-CPD is utilized to update the state. (26) and (27) are the Bayesian recursion process. In (26), $p(^*X_{k-1}|B_{k-1})$ can be obtained by the pose estimation of B_{k-1}. $p(X_k|^*X_{k-1})$ is the state transition probabilities [21]. In (27), $p(B_k|X_k)$ is the likelihood of object B_k. After normalizing, the updated state X_k can be obtained. The velocity is computed from X_k and $^*X_{k-1}$.

$$p(X_k|B_{k-1}) = \int p(X_k|^*X_{k-1})p(^*X_{k-1}|B_{k-1})d^*X_{k-1} \tag{26}$$

$$p(X_k|B_k) = \eta p(B_k|X_k)p(X_k|B_{k-1}) \tag{27}$$

where η is a normalization coefficient.

E. Comparison With Likelihood-Model-Based Method

There are three important distinctions between the proposed detection system and the likelihood-model-based method. First, while the likelihood-model-based method uses a constant threshold to detect dynamic objects, the proposed method selects them by the adaptive threshold that varies with distance and grid angular resolution. Considering the characteristics of point clouds, the adaptive threshold can better suppress false alarms and detect the dynamic objects as much as possible.

Second, while PE_MSS takes the omnidirectional angle as the initial uncertainty, PE-CPD exploits the motion information obtained by CPD registration as the prior. In this way, the pose estimation results can be more accurate, especially for the vehicles with sparse point clouds. In addition, motion calibration process is proposed to make the results of CPD better fit the model.

Third, in the state update phase, the acquired motion information is applied in PE-CPD to reduce the initial uncertainty. Compared with PE_MSS, PE-CPD can maintain the motion consistency more stably. PE-CPD also uses model selection process to better fit the model. Empirical comparison to the likelihood-model-based method is provided in Section IV-A, IV-B and IV-C.

F. Computational Complexity

In this section, the complexity of the proposed pose estimation method is analyzed. According to [32] and [33], the EM operation is the main time-consuming part of CPD and its complexity is $O(n)$, n is the number of point clouds returned from the object. In Algorithm 1, the core of function MinAreaBoundRect is to find the convex hull of input point clouds. Graham [37] has published an excellent Graham Scan Algorithm with time complexity $O(nlogn)$. The remaining operations of Algorithm 1 is irrelevant to the input, so the complexity is $O(1)$. Therefore, the total time complexity of ModelSelection is $O(nlogn)$.

In Algorithm 2, the time consumption is mainly concentrated in the **for** loop, and the most time-consuming step is on line 5, which calculates the normalized weight of each point in object B. As shown in [21], look-up table can be used to calculate the integral in (2). The complexity of this step is $O(n)$. Since the iteration number and particle number are given at the beginning of the process, the complexities of other lines are $O(1)$, and the total complexity of Algorithm 2 is $O(n)$.

Up to now, the complexity of several major parts of the proposed pose estimation has been analyzed. Consequently, the overall complexity is $O(nlogn) + O(n) + O(n) = O(nlogn)$. By analyzing the PE_MSS algorithm in [21], it can be concluded that its total computational complexity is $O(nlogn) + O(n) + O(1) = O(nlogn)$ as $O(nlogn)$ is the complexity of function MinAreaRect, $O(n)$ is the complexity of normalized weights calculation and $O(1)$ is the total complexity of the remaining steps. Hence, PE-CPD and PE_MSS have the same order of magnitude. Since [21] proves the real-time detection performance with PE_MSS, the proposed detection method will satisfy the real-time requirement in theory.

IV. Experiments

The KITTI datasets [38] are used to validate the performance of the proposed detection method. The autonomous driving platform in KITTI was equipped with two high-resolution color and grayscale video cameras, a Velodyne laser scanner and a GPS localization system. The ego-vehicle posture can be obtained by the GPS. The label set for the point clouds is provided by the dataset, which can be extracted from the optical images. Therefore, the point clouds in our experiments includes only optically visible regions, that

TABLE I
THE DYNAMIC OBJECT DETECTION RESULTS OF ADAPTIVE THRESHOLD AND CONSTANT THRESHOLD

Method	Dataset	Dynamic Objects	Detected Objects	Detected Dynamic Objects	Precision	Recall	F_1 score
Proposed Method	2011_09_26_drive_0018	815	2749	654	0.24	0.80	0.37
	2011_09_26_drive_0056	421	2199	414	0.19	0.98	0.32
	Total	1236	4948	1068	**0.22**	**0.86**	**0.35**
Chen's Method	2011_09_26_drive_0018	815	3684	565	0.15	0.69	0.25
	2011_09_26_drive_0056	421	4492	411	0.09	0.98	0.17
	Total	1236	8176	976	**0.12**	**0.79**	**0.21**

is, the area with $x \geq 0$ in each frame. Before detection, the coordinates of the datasets are converted to the specified ego-vehicle coordinate system.

The likelihood-field-based vehicle detection and tracking method proposed in [21] is used to compare with the proposed method. The comparison experiment is implemented with following three aspects. Firstly, the performance of dynamic object detection based on constant threshold and adaptive threshold is compared. Then, the accuracy of pose estimation by PE_MSS and PE-CPD is compared on the vehicles with sparse point clouds. Finally, the detection results are compared with Chen's method [21]. After the comparative experiments, the major time costs of the proposed method are listed with MATLAB.

The inputs in Algorithm 2 are $N = 16$, $\Delta = 0.4$, $T_{PE} = 9$, $W = 1.8$, $L = 4.8$. The initial parameters are w=1, ψ=$\frac{\pi}{36}$, σ_2=0.8, $zoom$=$1\Big/\sqrt[3]{2}$ and Reg_0=$(c_x^{mb}, c_y^{mb}, \theta^{mb}, L/2, \psi)$. The sampling range of velocity for associated object is v= $[-35m/s, 35m/s]$. The detailed experiments are described in the following sections.

A. Dynamic Object Detection

In this section, the datasets 2011_09_26_drive_0018 and 2011_09_26_drive_0056 are used to count the detected dynamic objects. The proposed method uses adaptive threshold, while Chen's method uses constant threshold (4 bins). The size of the grid angular resolution affects the construction of the virtual scan. We choose the grid angular resolution to be 1 degree. The coverage of the virtual scan is a circle with a radius of 80m. The detection results are shown in TABLE I. F_1 score can be used for evaluating the detection results because it includes both precision P and recall R [17].

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN} \tag{28}$$

$$F_1 = \frac{2PR}{P + R} \tag{29}$$

where TP denotes the correctly detected objects, FP is the false alarms, FN represents the missing objects.

In TABLE I, F_1^{adp} of proposed method is 0.14 greater than F_1^{con} of Chen's method, which indicates that the adaptive threshold improves the detection performance of dynamic objects. These two datasets totally have 564 frames. Since the dynamic objects are detected from the difference of consecutive virtual scans, the 1236 dynamic vehicles are distributed in 562 frames. Using adaptive threshold, we can reduce the number of false alarms by 46%.

Herein, several frames of point clouds are used to intuitively illustrate the advantages of the adaptive threshold. These point clouds are from the datasets 2011_09_26_drive_0056 and 2011_09_26_drive_0057. There are 4, 3 and 4 dynamic objects in the scenes, respectively. The comparative results are shown in Fig. 12. Fig. 12(a)(c)(e) are results with the constant 4 bins. (b)(d)(f) are results with the proposed novel adaptive threshold.

In Fig. 12(a)(b), the dynamic objects detected with constant threshold and adaptive threshold are 1 and 3, respectively. The reason for the one missed target in Fig. 12(b) is that the missed vehicle has few changed bins with the sparse scanning points so that neither the constant threshold nor the adaptive threshold can detect it. In Fig. 12(c), only two dynamic vehicles are detected, while in (d), all the three vehicles are detected. The lost vehicle in Fig. 12(c) is turning at that moment and is far away from the Lidar. By employing the adaptive threshold, the vehicle in this special posture is detected. In Fig. 12(e)(f), three dynamic vehicles are in the same direction as ego-vehicle, so only the short side can be seen. In (e), the nearest vehicle and the bicycle are detected, while all the four dynamic objects are detected in (f).

The purpose of dynamic object detection is to suppress the false alarms on the basis of detecting dynamic objects as much as possible. From the framework in Fig. 10, the pose estimation will be made on each detected dynamic object. Hence, more detected objects mean more complicated operations. Using adaptive threshold can tackle the issue to some extent. The results of TABLE. I and Fig. 12 show that the proposed threshold will reduce the false alarms without reducing the objects detection compared with constant threshold. In addition, it also shows the necessity of subsequent pose estimation for winnowing out the substantial numbers of false positives and detect the dynamic vehicles.

B. Pose Estimation by PE-CPD

In this section, four groups of data in KITTI datasets, 0011, 0014, 0018 and 0057, are utilized for verifying the stability of PE-CPD. The ground truth supported by KITTI website is applied for selecting the dynamic vehicles in each frame. Pose estimation by PE-CPD and PE_MSS are implemented on the vehicles. The poses of all labelled vehicles in radius of 80m are estimated. Position error and angle error are utilized to show the estimation accuracy. Position error is the distance between the midpoint of the ground truth and the midpoint of

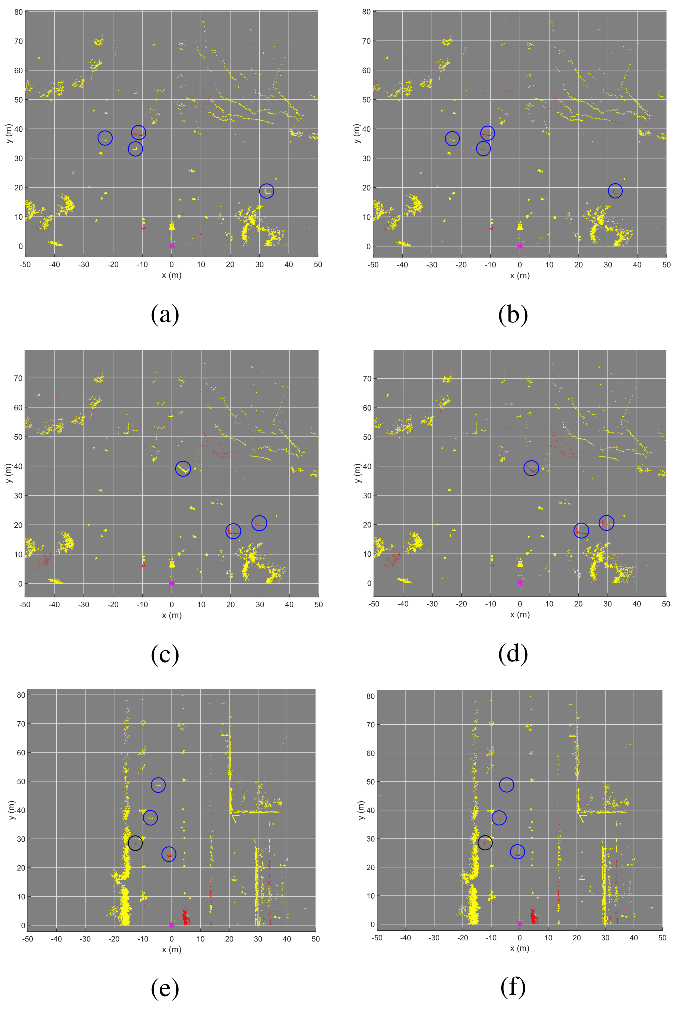


Fig. 12. Comparison of the dynamic object detection. (a)(c)(e) are the results with the constant threshold(4 bins). (b)(d)(f) are the results with the proposed adaptive threshold. The yellow points are scan points. The red points are the detected dynamic objects. The mark * in magenta indicates the Lidar. The grid size is $10m \times 10m$. The objects in blue circles are dynamic vehicles. The black circles in (e) and (f) are running bicycles. The ground returns are eliminated by EKF.

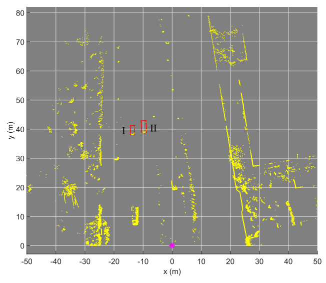


Fig. 13. The ground truth of two distant vehicles.

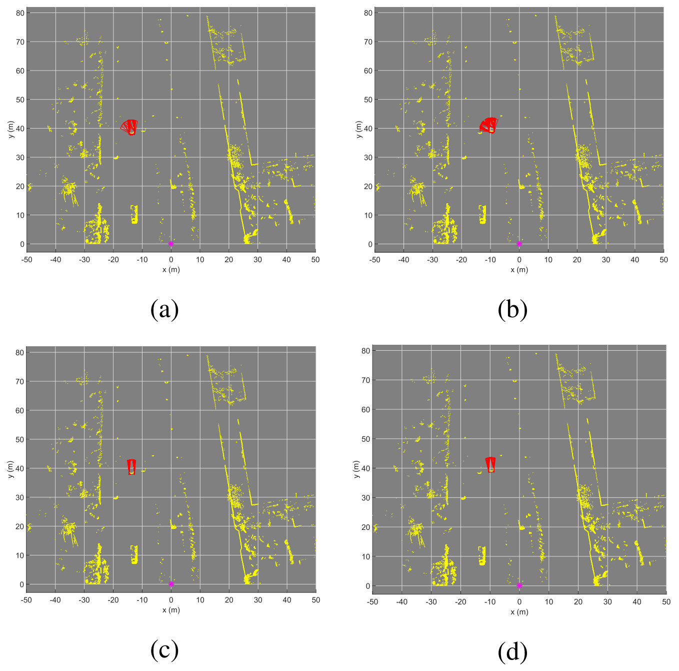


Fig. 14. Statistic experiments of the pose estimation by PE_MSS and proposed PE-CPD. (a)(c) are the estimated results of target I. (b)(d) are the estimated results of target II. (a)(b) are estimated by PE_MSS. (c)(d) are estimated by PE-CPD. The mark * in magenta indicates the position of the Lidar, which is the origin of the ego-vehicle coordinate system. The grid size is $10m \times 10m$.The ground returns are eliminated by EKF.

the fitting rectangle. Angle error is the difference between the ground truth angle and the estimated posture angle.

TABLE II presents the statistical errors of two pose estimation methods. PE-CPD gets more stable and accurate results than PE_MSS in both position and angle estimation. For the vehicles with sparse point clouds (less than 50 points), PE-CPD has more obvious advantages. The position error and the angle error are reduced by about 0.1 meters and 4.8 degrees, respectively. At the same time, it can be seen that the pose estimation with PE-CPD is more stable by analyzing the standard deviation of errors. Since more than one-fifth of the vehicles in the TABLE have sparse point clouds, the statistical errors are slightly large.

The first frame of 2011_09_26_drive_0056 dataset is utilized for visually verifying the stability of PE-CPD. There are 2 vehicles in front of ego-vehicle. The ground truth is labelled in Fig. 13. Only the short side can be seen and both of them are about 40 meters from the Lidar. 100 runs of experiments are implemented.

Fig. 14 shows the comparative results of pose estimation by PE_MSS and PE-CPD. In Fig. 14(a) and (b), the results by PE_MSS show great instability in repeated experiments due to the sparse point clouds. Statistical results of the deviation of each parameter in state $X = (c_x, c_y, \theta)$ are shown in Fig. 15. The results of PE-CPD is more robust than PE_MSS in all three curves. The results of PE-CPD are closer to the ground truth. It should be noted that the results of both PE-CPD and PE_MSS have a deviation of about $0.1m$ from the ground truth in Fig. 15(b). The ground truth supported by the KITTI dataset shows the minimum circumscribed rectangle of the vehicle, while there is a width Δ=$0.4m$ of the visible side in the fitting model. Therefore, as long as the object that has one side visible can be fitted in the short side part of the model, the object can be detected as a vehicle, which results in a undulating change of the short side position. The results show that PE-CPD can get more accurate pose estimation than

TABLE II

THE POSE ESTIMATION ERRORS OF PE-CPD AND PE_MSS

		PE-CPD						PE_MSS					
		Dataset				Total	sparse points	Dataset				Total	sparse points
		0011	0014	0018	0057			0011	0014	0018	0057		
Position Error/(m)	mean	0.3280	0.4313	0.4708	0.4402	**0.4301**	**0.5575**	0.3959	0.4554	0.5085	0.4913	**0.4724**	**0.6523**
	std	0.2521	0.3088	0.2459	0.3134	**0.2901**	**0.3861**	0.3576	0.3170	0.2989	0.3886	**0.3437**	**0.4932**
Angle Error/(rad)	mean	0.0691	0.1171	0.0538	0.0653	**0.0772**	**0.1879**	0.0976	0.1189	0.0743	0.1225	**0.1050**	**0.2722**
	std	0.1955	0.2257	0.1221	0.1686	**0.1818**	**0.2906**	0.2928	0.2310	0.1901	0.3430	**0.2711**	**0.4152**

TABLE III

DYNAMIC VEHICLE DETECTION RESULTS BY PROPOSED METHOD AND CHEN'S METHOD

Datasets	Range	Total Vehicles	Proposed Method					Chen's Method				
			Detected Vehicles	False Alarms	P	R	F_1	Detected Vehicles	False Alarms	P	R	F_1
0018	<40m	448	368	5	0.99	0.82	0.90	357	22	0.94	0.80	0.86
	>40m	368	219	16	0.93	0.60	0.73	39	9	0.81	0.11	0.18
	Total	816	587	21	0.97	0.72	0.82	396	31	0.93	0.49	0.64
0056	<40m	407	385	21	0.95	0.95	0.95	359	32	0.92	0.88	0.90
	>40m	22	9	3	0.75	0.41	0.53	0	9	0	0	-
	Total	429	394	24	0.94	0.92	0.93	359	41	0.90	0.84	0.87
0057	<40m	732	540	6	0.99	0.74	0.85	466	37	0.93	0.64	0.75
	>40m	381	233	18	0.93	0.61	0.74	142	50	0.74	0.37	0.50
	Total	1113	773	24	0.97	0.69	0.81	608	87	0.87	0.55	0.67
Overall	<40m	1587	1293	32	0.98	0.81	0.89	1182	91	0.93	0.74	0.83
	>40m	771	461	37	0.93	0.60	**0.73**	181	68	0.73	0.23	**0.35**
	Total	2358	1754	69	0.96	0.74	**0.84**	1363	159	0.90	0.58	**0.70**

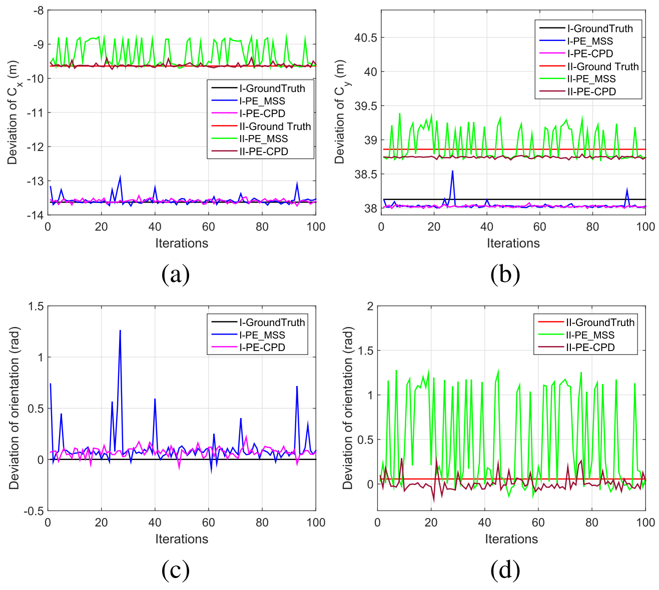


Fig. 15. Statistical results of the deviation of the pose estimation by PE_MSS and proposed PE-CPD. (a) shows the statistical deviation of c_x for target I and target II. (b) shows the statistical deviation of c_y for target I and target II. (c) and (d) are the statistical deviation of orientation for target I and target II, respectively.

PE_MSS for the distant targets, which is consistent with the theoretical analysis in Section II-D.

C. Vehicle Detection

In the KITTI datasets, 2011_09_26_drive_0056 and 2011_09_26_drive_0057 have already been used in [21] to compare with the vehicle detection results in [28]. Here, we use these two datasets as well as 2011_09_26_drive_0018 to verify the performance of the proposed method. Before experiments, the datasets are processed manually as the label sets given by KITTI are not complete. The dynamic vehicles within 80m are detected in the scene. It is specified that the distance 40m is the boundary between the distant area and the close area.

The dataset 0018 contains 270 frames with 816 dynamic vehicles. There are 448 vehicles in the close area and 368 vehicles in the distant area. The dataset 0056 contains 294 frames in total. Among 429 dynamic vehicles, there are 407 vehicles that are in the range of 40m. The dataset 0057 contains 361 frames with 1113 dynamic vehicles, among which there are 381 vehicles in the distant. F_1 score is applied to evaluate the detection results.

As shown in TABLE III, for the dataset 0018, 587 vehicles are detected in total, which is 191 vehicles more than Chen's method. Although the proposed method gets better results in both distant and close areas, the superiority is more pronounced in distant area. Among 368 vehicles, the proposed method detects 219 vehicles, while number of detected vehicles by Chen's method is only 39. The F_1 score is improved by 3 times. Similar results are obtained in dataset 0056 and 0057. The proposed method can detect more vehicles with less number of false alarms.

According to the total results of three datasets, the proposed method gets excellent performance, especially in distant vehicles detection. Among the close 1587 vehicles, the proposed method can detect about 81% of them, while Chen's method

TABLE IV
OPERATION TIME OF THE PROPOSED METHOD

Total Time	Time/Frame	Frame 221-240					
		Time	Read Data	Ground Tracking	RBNN	Virtual Scan	Detection Time/Frame
185.2s	636ms	4996ms	84ms	123ms	2377ms	562ms	93ms

detects only 74%. Meanwhile, the proposed method reduces the false alarms by approximately 65%. In distant area, the performance of both methods is declined. The target is too far to return enough point clouds. The accuracy of pose estimation is decreased because of the incomplete target contour, which will affect the motion consistency judgment and cause the target to be lost. In this case, CPD is applied to provide target motion information as much as possible in the absence of contour information. With the motion information, the pose estimation becomes more accurate and the motion consistency is maintained. By using the proposed method, the total F_1 score can reach 0.73 and is doubled compared with Chen's method.

In total, the proposed method increases the number of detected dynamic vehicles by 28.7% and decreases the false alarms by more than a half. By combining the results of detection and false alarms, F_1 score of proposed method reaches 0.84, which is 0.14 greater than Chen's method.

D. Operation Time

In all the experiments, we use a computer with an Intel Core i7 CPU with 4GHZ dominant frequency and 16GB main memory. All the algorithms are conducted in MATLAB R2014b. The operation time of 0056 dataset is shown in TABLE. IV. The total 294 frames of point clouds are marked from 0 to 293. Frames 0 to 2 are used to construct environment and the operation time starts from frame 3. It takes 185.2s to detect vehicles in total 291 frames. To extract the consuming time of detection process, the operation time of each process in frame 221 to 240 is calculated. There is only one dynamic vehicle in each frame. The total running time is 4996ms. After removing the consuming time of data reading, ground segmentation, clustering and virtual scan mapping, it takes an average of 93ms to detect the dynamic vehicle in each frame. With the advanced hardware platform and parallel computing technology, the proposed detection method can get real-time performance.

V. CONCLUSION AND FUTURE WORK

To improve the efficiency of long-range target detection, a robust dynamic vehicle detection method based on PE-CPD is proposed in this paper. The method mainly includes the steps of dynamic object detection and dynamic vehicle confirmation. To detect dynamic objects, an adaptive threshold based on the distance and angular resolution is proposed. PE-CPD is applied to estimate the pose of object with motion calibration and model selection. For the sake of updating states of detected vehicles, the SSBF improved by PE-CPD is applied for vehicle detection at the next time step. Three comparative experiments with existing methods are implemented with KITTI datasets. The total F_1 score increases by 0.14 compared to Chen's method. In distant area, in particular, the F_1 score increases by 0.38. The total number of false alarms decreases by more than 56%. The results validate the performance of the proposed method, especially for the long distance detection of vehicles with sparse point clouds. Meanwhile, with the experiments of operation time, it is proved that the proposed method can meet real-time requirements with the advanced hardware platform and parallel processing.

Nevertheless, the fixed model size is a limitation factor for both the existing methods and the proposed method. It is necessary to change the model size in order to detect the bus, the truck and other larger targets. Therefore, the future work may focus on solving the problem with adaptive model size.

REFERENCES

[1] D. Neumann, T. Langner, F. Ulbrich, D. Spitta, and D. Goehring, "Online vehicle detection using Haar-like, LBP and HOG feature based image classifiers with stereo vision preselection," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 773–778.

[2] B. A. Alpatov and M. D. Ershov, "Real-time stopped vehicle detection based on smart camera," in *Proc. 6th Medit. Conf. Embedded Comput. (MECO)*, Jun. 2017, pp. 1–4.

[3] E. Başer and Y. Altun, "Classification of vehicles in traffic and detection faulty vehicles by using ANN techniques," in *Proc. Electr. Electron., Comput. Sci., Biomed. Eng. Meeting (EBBT)*, Apr. 2017, pp. 1–4.

[4] Y. Zhao and Y. Su, "Vehicles detection in complex urban scenes using Gaussian mixture model with FMCW radar," *IEEE Sensors J.*, vol. 17, no. 18, pp. 5948–5953, Sep. 2017.

[5] T. Giese, J. Klappstein, J. Dickmann, and C. Wöhler, "Road course estimation using deep learning on radar data," in *Proc. 18th Int. Radar Symp. (IRS)*, Jun. 2017, pp. 1–7.

[6] L. Daniel, D. Phippen, E. Hoare, A. Stove, M. Cherniakov, and M. Gashinova, "Multi-height radar images of road scenes at 150 GHz," in *Proc. 18th Int. Radar Symp. (IRS)*, Jun. 2017, pp. 1–7.

[7] V. Magnier, D. Gruyer, and J. Godelle, "Automotive lidar objects detection and classification algorithm using the belief theory," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 746–751.

[8] M. Pavelka and V. Jirovský, "Lidar based object detection near vehicle," in *Proc. Smart City Symp. Prague (SCSP)*, May 2017, pp. 1–6.

[9] A. Börcs, B. Nagy, and C. Benedek, "Instant object detection in Lidar point clouds," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 7, pp. 992–996, Jul. 2017.

[10] O. Kechagias-Stamatis, N. Aouf, and M. A. Richardson, "3D automatic target recognition for future LIDAR missiles," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 52, no. 6, pp. 2662–2675, Dec. 2016.

[11] L. Kurnianggoro and K.-H. Jo, "Object classification for LIDAR data using encoded features," in *Proc. 10th Int. Conf. Hum. Syst. Interact. (HSI)*, Jul. 2017, pp. 49–53.

[12] P. Corcoran, A. Winstanley, P. Mooney, and R. Middleton, "Background foreground segmentation for SLAM," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1177–1183, Dec. 2011.

[13] A. Azim and O. Aycard, "Detection, classification and tracking of moving objects in a 3D environment," in *Proc. 4th IEEE Intell. Veh. Symp.*, Jun. 2012, pp. 802–807.

[14] J. Cheng, Z. Xiang, T. Cao, and J. Liu, "Robust vehicle detection using 3D Lidar under complex urban environment," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 691–696.

[15] P. Merdrignac, E. Pollard, and F. Nashashibi, “2D laser based road obstacle classification for road safety improvement,” in *Proc. IEEE Int. Workshop Adv. Robot. Social Impacts (ARSO)*, Jun. 2015, pp. 1–6.

[16] B. Kim, B. Choi, S. Park, H. Kim, and E. Kim, “Pedestrian/vehicle detection using a 2.5-D multi-layer laser scanner,” *IEEE Sensors J.*, vol. 16, no. 2, pp. 400–408, Jan. 2016.

[17] A. Maligo and S. Lacroix, “Classification of outdoor 3D lidar data based on unsupervised Gaussian mixture models,” *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 1, pp. 5–16, Jan. 2017.

[18] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “Multi-view 3D object detection network for autonomous driving,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6526–6534.

[19] X. Chen *et al.*, “3D object proposals for accurate object class detection,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 424–432.

[20] Q. Tang, L. Kurnianggoro, and K.-H. Jo, “Statistical and geometrical features for LiDAR-based vehicle detection,” in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Dec. 2016, pp. 192–197.

[21] T. Chen, R. Wang, B. Dai, D. Liu, and J. Song, “Likelihood-field-model-based dynamic vehicle detection and tracking for self-driving,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 11, pp. 3142–3158, Nov. 2016.

[22] 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.

[23] A. Scheel, S. Reuter, and K. Dietmayer, “Using separable likelihoods for laser-based vehicle tracking with a labeled multi-Bernoulli filter,” in *Proc. 19th Int. Conf. Inf. Fusion (FUSION)*, Jul. 2016, pp. 1200–1207.

[24] Y. Ye, L. Fu, and B. Li, “Object detection and tracking using multi-layer laser for autonomous urban driving,” in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 259–264.

[25] X. Zhang, W. Xu, C. Dong, and J. M. Dolan, “Efficient L-shape fitting for vehicle detection using laser scanners,” in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 54–59.

[26] A. H. A. Rahman, H. Zamzuri, S. A. Mazlan, and M. A. A. Rahman, “Model-based detection and tracking of single moving object using laser range finder,” in *Proc. 5th Int. Conf. Intell. Syst., Modelling Simulation*, Jan. 2014, pp. 556–561.

[27] B. Fortin, R. Lherbier, and J. C. Noyer, “A model-based joint detection and tracking approach for multi-vehicle tracking with lidar sensor,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 1883–1895, Apr. 2015.

[28] A. Petrovskaya and S. Thrun, “Efficient techniques for dynamic vehicle detection,” in *Experimental Robotics* (Springer Tracts in Advanced Robotics), vol. 54. Berlin, Germany: Springer, Jan. 2008, pp. 79–91.

[29] A. Petrovskaya and S. Thrun, “Model based vehicle tracking for autonomous driving in urban environments,” in *Robotics: Science and Systems IV*. Cambridge, MA, USA: MIT Press, Jan. 2009, pp. 175–182.

[30] A. Petrovskaya and S. Thrun, “Model based vehicle detection and tracking for autonomous urban driving,” *Auton. Robots*, vol. 26, nos. 2–3, pp. 123–139, Apr. 2009.

[31] A. Myronenko and X. Song, “Point set registration: Coherent point drift,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2262–2275, Dec. 2010.

[32] S. Zeng, “An object-tracking algorithm for 3-D range data using motion and surface estimation,” *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1109–1118, Sep. 2013.

[33] S. Zeng, “A tracking system of multiple LiDAR sensors using scan point matching,” *IEEE Trans. Veh. Technol.*, vol. 62, no. 6, pp. 2413–2420, Jul. 2013.

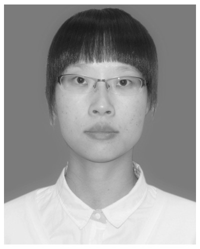
[34] C.-K. Kim, Y.-M. Han, B.-H. Bae, and J.-H. Kim, “The research of path planning algorithm considering vehicle’s turning radius for unmanned ground vehicle,” in *Proc. 11th Int. Conf. Control, Automat. Syst.*, Oct. 2011, pp. 754–756.

[35] A. Petrovskaya and O. Khatib, “Global localization of objects via touch,” *IEEE Trans. Robot.*, vol. 27, no. 3, pp. 569–585, Jun. 2011.

[36] K. Klasing, D. Wollherr, and M. Buss, “A clustering method for efficient segmentation of 3D laser data,” in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2008, pp. 4043–4048.

[37] R. L. Graham, “An efficient algorith for determining the convex hull of a finite planar set,” *Inf. Process. Lett.*, vol. 1, no. 4, pp. 132–133, 1972.

[38] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.



Kaiqi Liu received the B.S. degree from the School of Electronic and Information Engineering, Beihang University, Beijing, China, in 2013, where she is currently pursuing the Ph.D. degree. Her research interests include 3-D point clouds processing, point clouds segmentation, and Lidar-based objects detection.

Wenguang Wang received the B.S. degree from Jianghan Petroleum University, Jingzhou, China, in 1997 and the Ph.D. degree from Beihang University, Beijing, China, in 2007. He is currently an Associate Professor with the School of Electronic and Information Engineering, Beihang University. His research interests include signal processing, target detection and tacking, image understanding, and so on.

Ratnasingham Tharmarasa received the B.Sc.Eng. degree in electronic and telecommunication engineering from University of Moratuwa, Moratuwa, Sri Lanka, in 2001, and the M.A.Sc. and Ph.D. degrees in electrical engineering from McMaster University, Hamilton, ON, Canada, in 2003 and 2007, respectively.

From 2001 to 2002, he was an Instructor in electronic and telecommunication engineering at the University of Moratuwa. From 2002 to 2007, he was a Graduate Student/Research Assistant with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON, Canada, where he is currently a Research Associate. His research interests include target tracking, information fusion, and sensor resource management.

Jun Wang received the B.S. degree from Northwestern Polytechnical University, Xi’an, China, in 1995, and the M.S. and Ph.D. degrees from Beihang University, Beijing, China, in 1998 and 2001, respectively. He is currently a Full Professor with the School of Electronic and Information Engineering, Beihang University. His research interests include signal processing, DSP/FPGA real-time architecture, target recognition and tacking, and so on.