sensors

MDPI

Article

Fast Dynamic Vehicle Detection in Road Scenarios Based on Pose Estimation with Convex-Hull Model

Kaiqi Liu and Jianqiang Wang *

School of Vehicle and Mobility, Tsinghua University, Beijing 100084, China
* Correspondence: wjqlws@tsinghua.edu.cn

Received: 5 July 2019; Accepted: 15 July 2019; Published: 17 July 2019

Abstract: Dynamic vehicle detection is of great significance for the safety of autonomous vehicles and the formulation of subsequent driving strategies. A pose-estimation algorithm, namely, the pose estimation with convex-hull model (PE-CHM), is proposed in this paper, and introduced in the dynamic vehicle detection system. In PE-CHM, the convex hull of the object's point-clouds is first extracted and the most fitted bounding box is determined by a multifactor objective function. Next, the center position of the target is inferred according to the location and direction of the target. With the obtained bounding box and the position inference, the pose of the target is determined, which reduces the interference of the missing contour on pose estimation. Finally, three experiments were performed to validate the performance of the proposed PE-CHM method. Compared with several typical model-based methods, PE-CHM can implement dynamic vehicle detection faster, which reduces the amount of calculation on the basis of ensuring detection efficiency.

Keywords: autonomous vehicle; environmental perception; Lidar; dynamic vehicle detection

1. Introduction

There are generally static and moving obstacles in traffic environments. For autonomous vehicles in road scenario, since moving objects have their own movement state and motion direction, their motion randomness affects the behavior decision of an autonomous vehicle, which poses a greater threat to traffic safety. Therefore, the detection of moving vehicles is an important task [1–3], and is considered in this work.

For vehicle detection, camera and Lidar are commonly used sensors. The camera has the closet structure to that of the human eye, and can reflect features such as the color, texture, and edges of the target [4–6]. A dynamic object detection method in monitoring scenario with fixed cameras has been proposed [7]. The method models each pixel as mixture of Gaussians and can deal with slow lighting changes and multimodal distributions caused by shadows, specularities, swaying branches and other troublesome features of the real world. But this kind of method is not suitable to detect the moving objects for the mobile platform. In [8], a front-vehicle detection with oriented fast and rotated brief (ORB) matching method and spatial constraints was proposed, which had a good performance in both vehicle detection efficiency and effectiveness. However, it is difficult to obtain the accurate distance and contour size of the target with a monocular camera. The distance information is of great significance in the formulation of driving strategies for autonomous vehicles, which can easily be obtained with a Lidar sensor.

There are many useful methods to detect vehicles with Lidar [9]. Among these detection methods, the training-based and model-based methods are two common vehicle detection methods [10,11].

1.1. Training-Based Method

The training-based method usually extracts features from the objects with point-clouds data and employs classifiers to detect both static and dynamic vehicles. In [12], an adaptive mean shift was employed to segment the point-clouds. From the segments, five object attributes, area, elongatedness, planarity, vertical position, and range, were utilized to classify objects into vehicles and nonvehicles with a support vector machine (SVM). In [13], a new approach for object classification was proposed. A convolutional neural network was employed to preliminarily classify the objects, which were refined with contextual features considering possible expected scene topologies. A dataset with 1485 objects was applied to test the method, and the F-score average reached 89%. In [14], three descriptors combined with an SVM were utilized for vehicle classification. The three features were the length and width of the approximated 2D bounding box derived from the convex hull, the four radius values of the 3D spheres, the radius difference between the frontal and the back sphere pairs, and the difference between the concave side profile hull of the candidate vehicle and the prototype shape. The average F-score reached 86%. In [15], multiple sensors were utilized to detect and track moving objects. Lidar are applied to detect the dynamic objects with cells change and determine the regions of interest (ROI). The Histogram of Oriented Gradient (HOG) descriptor from visual images as well as discrete Adaboost is employed to classify the objects. With the fused information, the detection in highway scenario reached 97.8%. In [16], a real-time 3D vehicle detection method was proposed. A pre-RoI-pooling convolution technique and pose-sensitive feature map were utilized to improve computation efficiency and detection accuracy. The method could complete detection within 0.09 s. In [17], three features, including the 2D distance of the cluster, total number of points and the direction of the clustered points' distribution, are utilized to classify the vehicle and pedestrian combined with the backpropagation artificial neural network (BP-ANN). The detection and tracking of the proposed roadside Lidar data processing procedure are above 95%.

In general, the training-based method has the ability to achieve good performance in vehicle detection. However, the method requires large amounts of data for model training, while complex and variable scenarios could hardly afford the adequate training data. At the same time, the training process usually costs a lot of computing resources. These constitute the main limitations of the training-based method.

1.2. Model-Based Method

In [18], an object-based hierarchical two-level model is proposed for the joint probabilistic extraction of vehicles with aerial Lidar point-clouds that are relatively dense compared with common vehicle Lidar, such as Velodyne HDL-64E. A dataset with 1009 vehicles was applied in the experiment, 97% (F-score) of which could be detected with the proposed method. In [19], a new detection method of line segments in scanning laser-range measurements was proposed that could be especially applied to vehicle detection. Feature extraction could be summed up in the detection of triangles in the cumulative function and property estimation. A model-based joint detection and tracking method was proposed in [20], which is a track-before-detect method. The method improved estimation-accuracy performance, robustness to false alarms, and track continuity based on a bounding-box model. In [21], the L-shape model was proposed for vehicle detection and tracking, which can overcome the appearance change problem caused by Lidar. However, precise corner point position and heading angle extraction are required for the best performance. The reference [11] also used the minimum rectangle area with L-shape cloud fitting for box-fitting, which detect and track the objects in a close area.

In [22], a likelihood-field-based vehicle measurement model as well as the pose estimation with modified scaling series (PE-MSS) algorithm were proposed to detect the dynamic vehicles. Motivated by the work in [22], we used the information obtained from the point set registration with the process of pose estimation and the proposed pose estimation based on coherent point drift (PE-CPD) algorithm [10],which did not bring an additional computation burden to the algorithm and improved detection efficiency. Although the detection method was proven to have the ability to

complete the detection of a scenario that contained a single vehicle within 100 ms under the MATLAB platform, the model-fitting process with likelihood-field-based model still required a large amount of computation and consumed many computing resources.

The main objective of this work was to propose a dynamic vehicle detection system with Lidar point-clouds that could quickly and accurately detect dynamic vehicles in the scenarios, thereby improving the ability for autonomous vehicles to sense environmental hazards. In this work, our contributions mainly focus on the proposed vehicle-pose estimation algorithm based on a point-cloud convex-hull model and position inference, and estimation results were introduced into the process of dynamic vehicle detection. Compared with the existing methods, the proposed pose-estimation method requires low computation and is robust to the environment, which reduces the amount of computation while maintaining good detection efficiency.

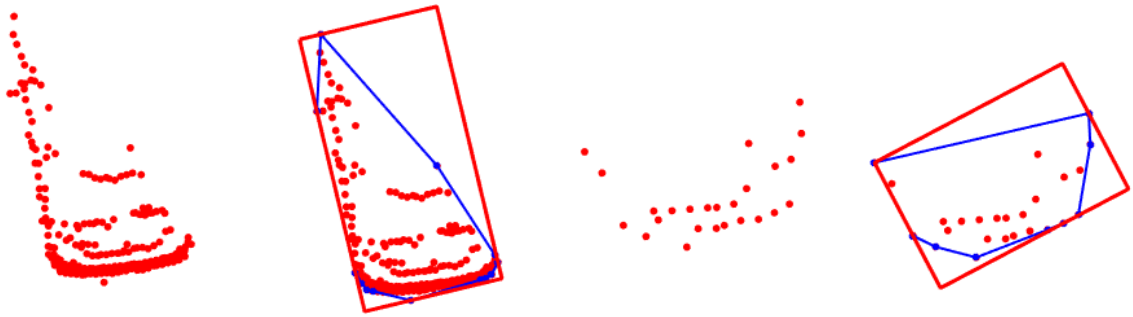
The remainder of the paper is organized as follows. In Section 2, the process of pose estimation with convex-hull model (PE-CHM) is described in detail. Based on PE-CHM, the dynamic vehicle detection system is introduced in Section 3. Experiments with several comparative methods were implemented and are analyzed in Section 4. Finally, concluding remarks are shown in Section 5.

2. Pose Estimation with CHM

Pose estimation of the target can be utilized for target detection [10,22]. The target pose in this work is denoted by (C_x, C_y, θ), in which (C_x, C_y) is the center position of the target in bird-view projection, and θ is the yaw angle. The process of the proposed PE-CHM method mainly includes two processes, convex-hull model fitting and position inference, which are detailed below.

2.1. Convex-Hull Model Fitting

Since most of the Lidar point-clouds come from the object contour, a point-cloud convex-hull model with a multifactor objective function is proposed to fit the point-cloud set. The model-fitting results of two vehicles are shown in Figure 1. The 3D point-clouds are projected on the x-y plane regardless of the height dimension. The detailed modeling process is shown in Algorithm 1.



(**a**) (**b**) (**c**) (**d**)

Figure 1. Convex-hull model-fitting results with multifactor objective function. (**a**,**c**) Original point-clouds of two vehicles. (**b**,**d**) Corresponding fitted bounding box with the convex-hull model. Blue lines show the convex hull of the point-cloud set.

The input of Algorithm 1 includes the point-clouds of object P and its approximate direction vector D_y that is obtained by the positions of the associated objects. First, the convex hull of the point-clouds set is determined [23]. The selected K points surround all the point-clouds in set P. As shown in Figure 2a, the two adjacent points $p_i = (x_i, y_i)$ and $p_{i+1} = (x_{i+1}, y_{i+1}), i = 1, ..., m-1$ in set K are sequentially selected to determine the equation of line 1 with slope k_1^i and intercept b_1^i, in which m is the number of points in K. The calculation process is shown in Equation (1).

$$k_1^i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \quad b_1^i = \frac{x_{i+1}y_i - y_{i+1}x_i}{x_{i+1} - x_i} \tag{1}$$

Algorithm 1 Convex-hull model (CHM) fitting.

Input: $P = (p_1, p_2, ..., p_n)^T$, D_y;

$K = \text{ConvexHull}(P_x, P_y)$;

for $i = 1, ..., m-1$ **do**

$(k_1^i, b_1^i) = \text{ParameterofLine}(p_i, p_{i+1})$;

$D^i = \text{DistancetoLine}(K, k_1^i, b_1^i)$;

$p_{max}^i = \{p_j | d_j = max(D^i)\}$;

$b_2^i = \text{ParameterofLine}(p_{max}^i, k_1^i)$;

$X_p^i = \text{ProjectiontoLine}(k_1^i, b_1^i)$;

$p_{pmax}^i = \{p_j | x_{pj} = max(X_p^i)\}$; $p_{pmin}^i = \{p_j | x_{pj} = min(X_p^i)\}$;

$b_3^i = \text{ParameterofLine}(p_{pmin}^i, -\frac{1}{k_1^i})$; $b_4^i = \text{ParameterofLine}(p_{pmax}^i, -\frac{1}{k_1^i})$;

$\mathcal{R}(i) = [k_1^i, b_1^i, b_2^i, b_3^i, b_4^i]$;

$\mathcal{A}(i) = \text{AreaofBoundingBox}(\mathcal{R}(i))$;

$\mathcal{D}(i) = \text{SumofMinDistancetoEdges}(K, \mathcal{R}(i))$;

$\mathcal{M}(i) = \text{MaxofMinDistancetoEdges}(K, \mathcal{R}(i))$;

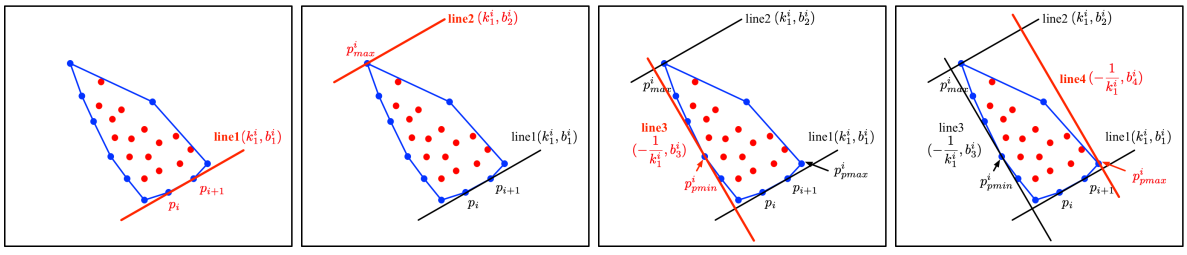
$\Theta(i) = \text{SelectYawAngleDif}(D_y, k_1^i)$;

end for

$(\mathcal{R}_{optimal}, \theta) = \text{SelectOptimalParameter}(\mathcal{R}, \mathcal{A}, \mathcal{D}, \mathcal{M}, \Theta)$;

$(x_{rect}, y_{rect}) = \text{IntersectionofLine}(\mathcal{R}_{optimal})$;

Output: $(x_{rect}, y_{rect}, \theta)$;



(**a**) (**b**) (**c**) (**d**)

Figure 2. The determination process of the bounding box. (**a**–**d**) indicate the determination process of line 1 to line 4, respectively. The dots denote the point-clouds. The polygon represented by the blue lines denote the convex hull of this set of point-clouds.

Next, point $p_{max}^i = (x_{max}^i, y_{max}^i)$ in K with the farthest distance from line 1 is selected. According to slope k_1 and the coordinates of p_{max}, intercept b_2^i of line 2 is calculated, which is shown in Figure 2b.

$$b_2^i = y_{\text{max}}^i - x_{\text{max}}^i k_1^i. \tag{2}$$

Subsequently, all points in set K are projected to line 1, and the x value of each projected point is obtained, which is represented as X_p^i.

$$X_p^i = \frac{K_x + k_1^i K_y - b_1^i k_1^i}{(k_1^i)^2 + 1}, \tag{3}$$

where K_x and K_y represent the x values and y values of the points in K. The points with the minimum and maximum projected x values $p_{pmax}^i = (x_{pmax}^i, y_{pmax}^i), x_{pmax}^i = \max(X_p^i)$ and $p_{pmin}^i = (x_{pmin}^i, y_{pmin}^i), x_{pmin}^i = \min(X_p^i)$ are selected to confirm the other two edges of the box with slope $-\frac{1}{k_1^i}$, which are shown in Figure 2c,d, respectively. The parameters are calculated with Equation (4).

$$b_3^i = y_{pmin}^i + \frac{1}{k_1^i} x_{pmin}^i, \quad b_4^i = y_{pmax}^i + \frac{1}{k_1^i} x_{pmax}^i. \tag{4}$$

All parameters of the four edges, including k_1^i and b_1^i-b_4^i, are recorded in $\mathcal{R}$. The area of the box determined by the four edges is recorded in $\mathcal{A}$. The sum value and maximum value of minimum distance of each point in K from the edges of the box were calculated and recorded in $\mathcal{D}$ and $\mathcal{M}$, respectively.

When the object in time step k was associated with the object at time step $k-1$, the approximate direction vector D_y can be obtained with the position of the object at these two time steps. The geometric center of the object at time step $k-1$ and k is denoted by (c_x^{k-1}, c_y^{k-1}) and (c_x^k, c_y^k), respectively. D_y is calculated with Equation (5).

$$D_y = (c_x^k - c_x^{k-1}, c_y^k - c_y^{k-1}). \tag{5}$$

If the object is not associated with other objects in adjacent scan frames, the vector is set to be empty. Vector D_y and slopes of bounding-box edges k_1^i and $-\frac{1}{k_1^i}$ are utilized to determine the object's approximate pose. The angles between vector D_y and the two straight lines whose slopes are k_1^i and $-\frac{1}{k_1^i}$ are calculated, respectively, and the smaller angle in $\Delta\theta_1$ and $\Delta\theta_2$ is stored in Θ. The angle difference is calculated with Equation (6).

$$\Delta\theta_1 = \operatorname{atan}\left(\left|\frac{k_1^i - k_{D_y}}{1 + k_1^i \cdot k_{D_y}}\right|\right), \Delta\theta_2 = \operatorname{atan}\left(\left|\frac{1 + k_1^i \cdot k_{D_y}}{k_1^i - k_{D_y}}\right|\right), \tag{6}$$

where k_{D_y} is the slope of vector D_y.

Then, points p_{i+1} and p_{i+2} are utilized in the next for-loop. After all the points in set K have been applied to construct the bounding box, an objective function is proposed to select the optimal parameters. Objective function F is show below.

$$F = \sum_{i=1}^{4} \omega_i \cdot \frac{f_i - \min(f_i)}{\max(f_i) - \min(f_i)}, f_i \in \{\mathcal{A}, \mathcal{D}, \mathcal{M}, \Theta\}, \tag{7}$$

where ω_i denotes the weights of each factor, $\sum_{i=1}^{4} \omega_i = 1$. f_i denotes one of the four impact factors. Set $\mathcal{D}$ and set $\mathcal{M}$ represent the sum and maximum value of the minimum distance of the points in set K from the bounding box, respectively. Set $\mathcal{A}$ stores the area value of each bounding box. Set Θ stores the angle between the direction vector and the yaw angle of each bounding box. In order to comprehensively consider the influence of the factors of different measurement spaces, the maximum and minimum values of the set are utilized to normalize the measurements. The parameters of the bounding box with the smallest F value are selected as the optimal parameters, and the corresponding bounding box is defined as the fittest box.

2.2. Position Inference

The convex-hull model in the last section can obtain the yaw angle of the target. To obtain complete pose information, the position of the target needs to be determined. From [22], the center of the bounding box with a minimum area is utilized as the initial center position. Then, the center position is iteratively calculated by model fitting. The reason that the center of the bounding box is not taken as the position of the target is that the obtained center position is inaccurate, as the contour of target is missing. As shown in Figure 1c, the vehicle is far from the Lidar and only one short side can be seen. It is obvious that the center of the bounding box in Figure 1d is not the actual center position of the vehicle, and there is a large deviation between them. In this section, the position-inference process is applied to obtain its center position without iteration.

Position inference uses a fixed model size to determine the center of the target based on its position relative to Lidar. Figure 3 shows the rectangular model of the vehicle. $O_e x_e y_e$ is the Lidar coordinate system. y_e axis points forward and x_e points to the right side of Lidar. $O_v x_v y_v$ is the coordinate system of the target vehicle with y_v points to the moving direction of target vehicle and x_v points to its right side. Angle θ between x_v and x_e is defined as the yaw angle of the target vehicle.

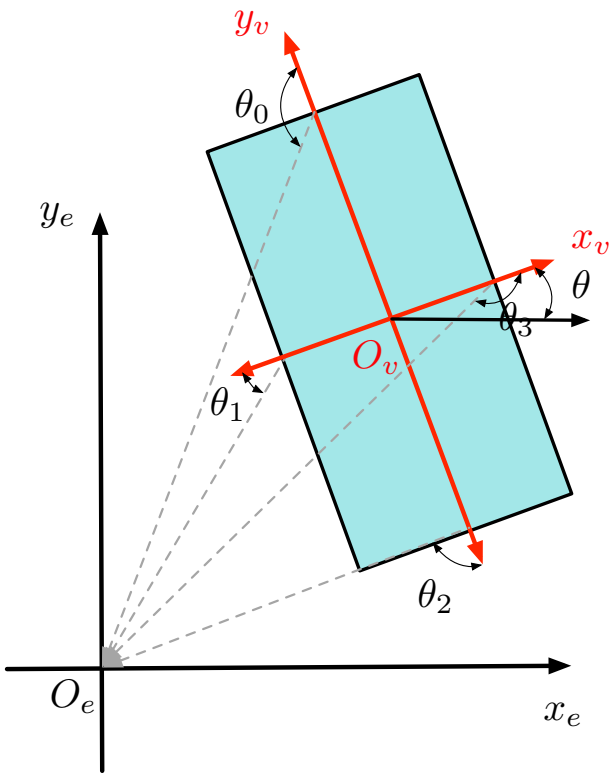


Figure 3. Rectangular model of the vehicle.

The obtained convex-hull model was employed to determine the visible edges of the targets. As shown in Figure 3, the vectors that point from the center of model to the center of each side constitute a normal vector set, and the vectors that point from the center of each side to the origin of the Lidar constitute the orientation vector set. For each edge, the angle between the orientation vector and normal vector show whether the edge can be seen under the perspective of the Lidar [22]. If the angle is greater than $\frac{\pi}{2}$, the edge is invisible; otherwise, the edge is visible.

Before inference, the geometrical center of the point-clouds is momentarily employed. The normal vector and orientation vector of each edge in the bounding box are obtained, by which the visible edges are determined. Generally, there are two edges visible in the scenario except when the object is directly in front of the Lidar, so that the angle between the normal vector and the orientation vector is

0, which is a small probability event and almost nonexistent in practice. It is assumed that there are theoretically always two visible edges, and the intersection point is denoted by p_{itsc}.

Figure 4 shows an example of position inference. Lines p_1p_2 and p_2p_3 are the two visible edges. The angle θ is its yaw angle. The angles between y_v and lines p_1p_2 and p_2p_3 are calculated, respectively. As can be seen from the Figure 4, the edge with the larger angle is the short edge, and the edge with the smaller angle is the long edge. The size of the model is assumed to be constant. The length and width of the model are set to L and W. Angle θ_b is determined depending on the relationship between the visible edge and y_v with Equation (8).

$$\theta_b = \begin{cases} \text{atan}(\frac{W}{L}), & \text{if } \langle \overrightarrow{p_2p_1}, y_v \rangle \leq \langle \overrightarrow{p_2p_3}, y_v \rangle \\ \text{atan}(\frac{L}{W}), & \text{otherwise} \end{cases} . \tag{8}$$

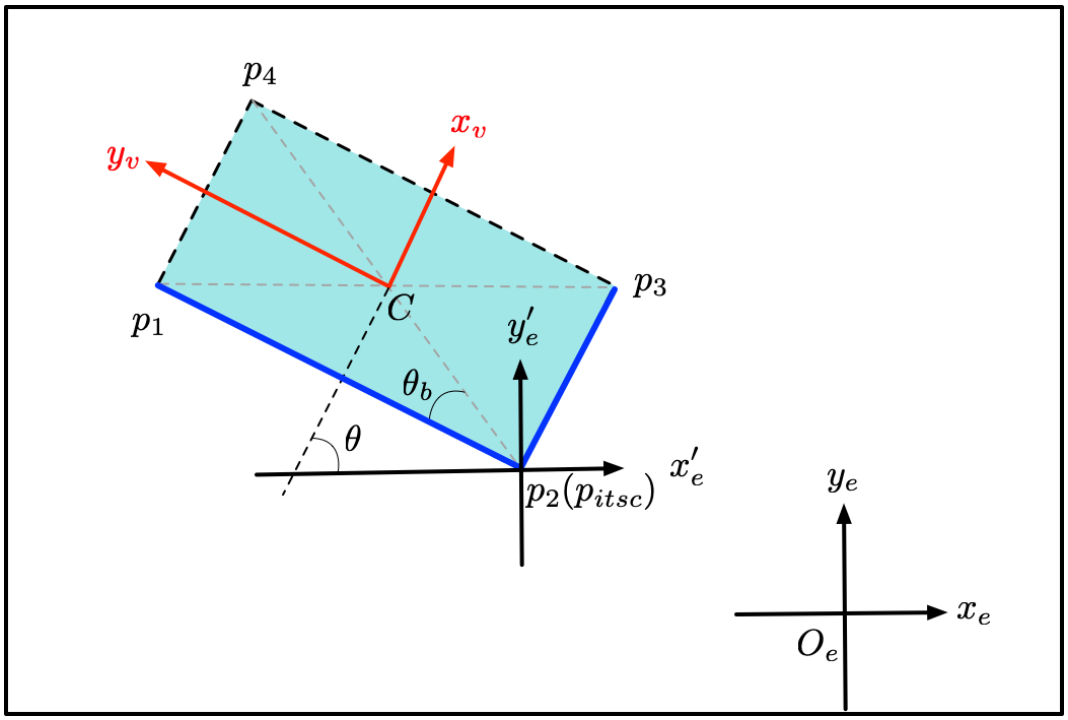


Figure 4. Center-coordinate determination with position inference. Two blue lines, visible edges. x_v and y_v, two axes of object's local coordinate system. x_e and y_e, two axes of global Lidar coordinate system. x'_e and y'_e, axes with p_{itsc} as the origin and parallel to x_e and y_e, respectively.

After calculating intersection p_{itsc} with the line equations of the two visible edges and angle θ_b, the coordinates of the model center can be obtained with Equations (9) and (10).

$$c_x = x_{p_{itsc}} + \frac{\sqrt{L^2 + W^2}}{2} \cdot \cos(\langle \overrightarrow{p_2p_1}, x'_e \rangle - \theta_b) \tag{9}$$

$$c_y = y_{p_{itsc}} + \frac{\sqrt{L^2 + W^2}}{2} \cdot \sin(\langle \overrightarrow{p_2p_1}, x'_e \rangle - \theta_b) \tag{10}$$

where $(x_{p_{itsc}}, y_{p_{itsc}})$ is the coordinate of intersection point p_{itsc}, $\langle \overrightarrow{p_2p_1}, x'_e \rangle$ denotes the angle between vector $\overrightarrow{p_2p_1}$ and x'_e.

2.3. Pose Estimation

The pose-estimation process uses the convex-hull model and position inference to determine the pose of target. The detailed description of pose estimation is described in Algorithm 2.

The inputs of Algorithm 2 include the point-cloud coordinates of the object and approximate direct vector D_y. The output is the center position and yaw angle of the target.

First, CHM fitting in Algorithm 1 was implemented to select the optimal bounding box. The center of the box can be obtained, and the theoretical visible edges were determined with the normal vector and orientation vector of the model. Then, the intersection point of the two visible edges is calculated. With the fixed size of the vehicle rectangular model and the obtained yaw angle, the center position of the target is finally infered.

Algorithm 2 Pose estimation with convex-hull model (PE-CHM).

Input: $P = (p_1, p_2, ..., p_n)^T$, D_y;

1: $(x_{rect}, y_{rect}, \theta) = \text{CHM\_Fitting}(P, D_y)$;

2: $c_x^m = \text{mean}(x_{rect})$, $c_y^m = \text{mean}(y_{rect})$;

3: $N^m = \text{NormalVector}(x_{rect}, y_{rect}, c_x^m, c_y^m)$;

4: $O^m = \text{OrientationVector}(x_{rect}, y_{rect})$;

5: $E_v = \text{FindVisibleEdge}(N^m, O^m)$;

6: $p_{itsc} = \text{FindIntersect}(E_v)$;

7: $(c_x, c_y) = \text{PositionInference}(\theta, D_y, p_{itsc})$;

Output: (c_x, c_y, θ);

3. Dynamic Vehicle Detection

In order to detect dynamic vehicles fast and accurately, a detection system with PE-CHM is proposed in this section. The framework of the detection process is shown in Figure 5.

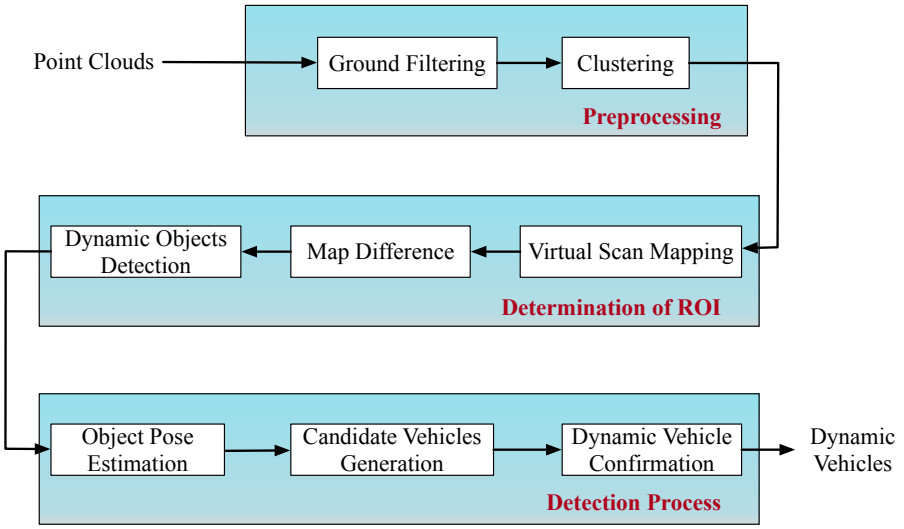


Figure 5. Framework of the dynamic vehicle detection based on pose estimation with convex-hull model (PE-CHM).

The detection system mainly included three parts. First, point-clouds are preprocessed to reduce the amount of data, and clustered based on the 3D point-cloud characteristics. Second, a virtual-scan map is constructed to determine the region of interest, and the dynamic objects are extracted from the clustering results. At this stage, the 3D coordinates of the point-clouds are projected onto the 2D x-y plane for subsequent processing. Third, the poses of the extracted objects are estimated,

combined with motion consistency, and the object is judged as a vehicle or nonvehicle. The detailed process is shown below.

3.1. Preprocessing

The number of point-clouds received by the Lidar is usually large, and a large percentage of the point-clouds come from the ground. In order to reduce the number of point-clouds to be processed and eliminate the interference of the measured outliers, the original point-clouds need to be preprocessed before detection [24]. The preprocessing process in this paper includes ground filtering and point-cloud clustering.

Many ground-filtering methods of point-clouds have been proposed [25–28]. Generally, road surfaces are relatively flat. Considering operation time, the extended Kalman filter (EKF)-based ground-filtering algorithm in [28] is utilized. Subsequently, the radially bounded nearest neighbor (RBNN) algorithm [29] is applied to cluster point-clouds above the ground. Among clustering results, clusters with few point-clouds are eliminated.

3.2. ROI Determination

Dynamic objects are detected from the preprocessing results, which constitute the ROI and avoid the complexity of detecting vehicles in the entire point clouds. The ROI extraction process includes virtual-scan mapping, virtual-scan difference, and dynamic-object detection.

3.2.1. Virtual-Scan Mapping

After preprocessing, the remained point-clouds above the ground are projected to the virtual-scan map for dynamic-object detection. The virtual-scan map is a polar grid representation of the point-clouds [10]. As shown in Figure 6, the polar grid map is divided into grids with the same center angles and radial lengths. The state of each grid is determined by the number and position of point-clouds. The grid without point-clouds projected is defined as free, as shown by the green grids in Figure 6. In each sector divided by the same center angle, the grid closest to the Lidar in the place where the object is located is defined as the occupied state, which are shown with the red color. The states of the grids between the occupied grid and the farthest grid where the object is located are defined as the occluded states, which are shown in yellow.

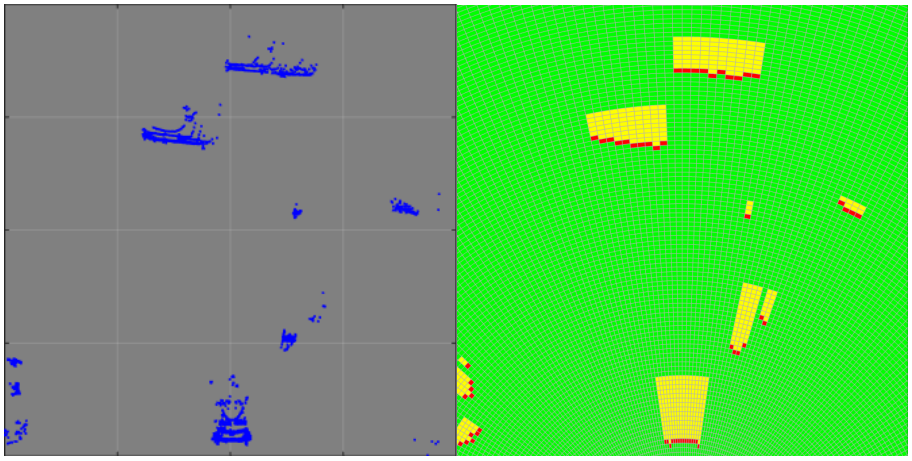


Figure 6. Virtual-scan mapping. (**left**) Point-clouds; (**right**), corresponding virtual scan. States of grids in green, red, and yellow are free, occupied, and occluded, respectively.

3.2.2. Virtual-Scan Difference

Since the origin of the point-cloud coordinates is the Lidar location, the coordinate systems of the two consecutive point-clouds should be unified before calculating the virtual-scan difference when the

Lidar is on a mobile platform. The transformation process with the information received by GPS-ISN is as follows.

$$T_c = \begin{bmatrix} R^k & t^k \\ 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} R^{k-1} & t^{k-1} \\ 0 & 1 \end{bmatrix} \tag{11}$$

$$R^k = R(\gamma^k) * R(\varphi^k) * R(\phi^k), \tag{12}$$

where $k-1$ and k represent two consecutive comments. R and t denote the rotation matrix and translation vector. γ, φ, and ϕ denote roll angle, pitch angle, and yaw angle of the ego vehicle, respectively. With the transformation matrix T_c, the coordinates of the point-clouds in time step $k-1$ are transformed into the coordinate system in time step k. The transformed coordinates of P_{k-1} are represented as P^*_{k-1}.

$$P^*_{k-1} = T_c \cdot P_{k-1}. \tag{13}$$

After converting the point-cloud coordinates in time step $k-1$ to the coordinate system in time step k, virtual-scan mapping is performed on the two frames of the point-clouds, respectively, and the map difference is implemented.

3.2.3. Dynamic-Object Detection

The difference of two consecutive virtual scans is applied to detect the dynamic objects in the scenario [15]. Dynamic objects are detected according to the change of the grid state. Adaptive threshold T_{do} of the changed grids for object o is employed in detection [10].

$$T_{do} = \left\lceil \frac{W}{A_r \cdot \|o\|} \right\rceil, \tag{14}$$

in which W denotes the width of the vehicle. A_r represents the angular resolution of the virtual scan. $\|\cdot\|$ is the Euclidean distance between object and Lidar. $\lceil \cdot \rceil$ is the rounding-up operation. According to the virtual-scan difference and adaptive detection threshold, dynamic objects in the scenario are detected, which become the ROI for subsequent dynamic vehicle detection.

3.3. Dynamic Vehicle Detection

In this section, the dynamic vehicle detection system is detailed. The flowchart of the first stage in the detection process is shown in Figure 7a.

As shown in Figure 7a, after obtaining the dynamic objects in Section 3.2.3 at time step k, the pose of the object is determined by PE-CHM with $D_y = \emptyset$, and the obtained yaw angle θ contains two angles that differ by 90 degrees. Subsequently, the point-clouds at time step $k-1$ are transformed into the coordinate system at time step k. Along the direction of yaw angle, velocity is evenly sampled in the range of $[v^s_1, v^s_2]$ to find the associated object at time step $k-1$. When the state of the objects at time step k and $k-1$ satisfy motion theory [22], they are associated as the same object. Motion theory determines whether objects are associated by inspecting the grid change between the objects' positions in the virtual scan at consecutive time steps. Approximate direction vector D_y is obtained by the associated objects. With the direct vector, the poses of the associated objects at time step $k-1$ and k are determined with PE-CHM. Motion consistency [22] is then applied to judge the dynamic objects. Motion consistency means that changes of velocity and yaw angle should be small. After the judgment of motion consistency, potential vehicles are detected.

Subsequently, as can be seen in Figure 7b, at time step $k+1$, the point-cloud coordinates of time step k are transformed into the coordinate system at time step $k+1$. The pose of the potential vehicles is applied to select their associated objects. The poses are predicted according to the obtained D_y, yaw angle, and velocity. Objects near the predicted poses are applied to get new approximate direct vector D_y, and are modeled with PE-CHM. If the associated objects at time step $k-1$ to $k+1$ satisfy motion consistency, the object at time step $k+1$ is first detected as a dynamic vehicle.

At time step $k+2$, for the detected dynamic vehicle, a similar process to Figure 7b is employed to update the state of the detected vehicle. If motion consistency is not satisfied, the vehicle is eliminated.

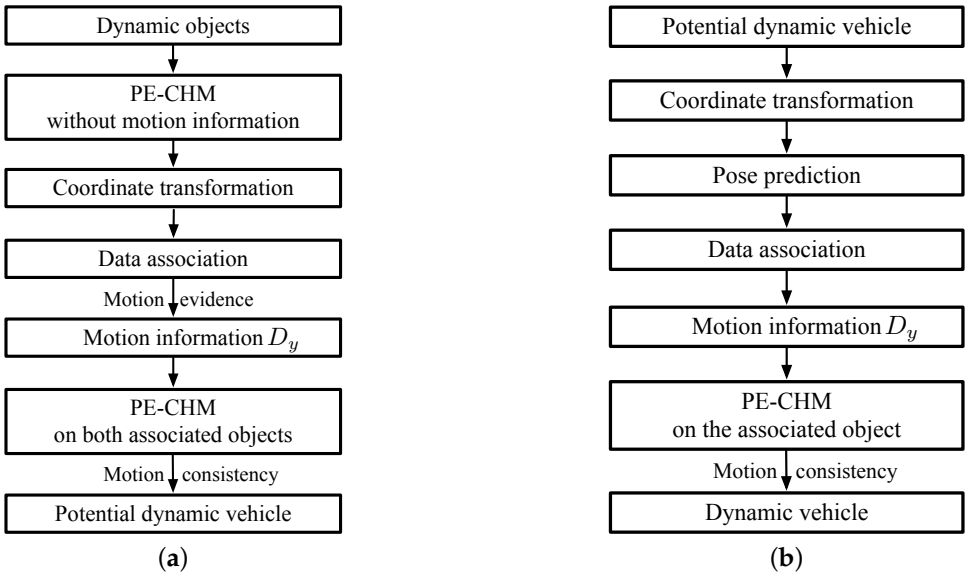


Figure 7. Flowchart of dynamic vehicle detection. (**a**) Potential dynamic vehicle detection. (**b**) Dynamic vehicle determination.

3.4. Computational Complexity

The purpose of this work was to develop a dynamic vehicle detection system that has the ability to detect targets quickly and accurately. In this section, the complexity of the detection process is analyzed compared with the method based on the likelihood-field-based model, which has been proven to perform well in dynamic vehicle detection [10]. Considering that the framework of the detection process in [10] and the process in this work are similar, mainly the complexity of the pose-estimation process is analyzed in this section.

By analyzing the process of Algorithm 2 and the process in [10], the largest computational complexity is concentrated on the solution of the point-cloud convex hull [23]. Therefore, in terms of complexity, their operation complexity is in the same order. Even so, the proposed method has a large advantage in reducing computing resources. Specifically, in Algorithms 1 and 2, the main operation focuses on the for loop to find the best bounding box. It is assumed that the convex-hull process obtains m points that can surround the objects, whose complexity is $O(m)$, in most cases $m < n$, and only when the target point-clouds returned from a single layer, $m = n$. In PE-CPD, meanwhile, calculation mainly focuses on the measurement model-fitting process. The integral value of each point needs to be calculated. Though the erf function was utilized instead of the integral operation for reducing computational complexity, the computational complexity of this process is still $O(n)$. At the same time, the introduction of the CPD process also brought computation complexity of $O(n)$. Therefore, in addition to $O(nlogn)$ that is needed by the convex hull, PE-CHM additionally brings computational complexity of $O(m) + O(1)$, while PE-CPD brings computational complexity of $2O(n) + O(1)$. Therefore, PE-CHM brings a reduction in the amount of calculation.

4. Experiments and Results

In this paper, three experiments were performed to validate the performance of the proposed pose-estimation algorithm and dynamic vehicle detection system. The KITTI dataset [30] was employed in the experiments. In the KITTI dataset, the autonomous driving platform was equipped with two high-resolution color cameras, two grayscale video cameras, a Velodyne HDL-64E, and a GPS localization system. The provided label set of the Velodyne point-clouds was located within the view

of the optical sensors, that is, in front of the ego vehicle. Therefore, only the point-clouds in front of the ego vehicle were processed in this paper. Considering that the provided labels are incomplete, the labels of each frame of the point-clouds were manually corrected. The angular resolution in virtual scan map is set 1 degree. The velocity range of $[v_1^s, v_2^s]$ when associating objects is set to $[-35, 35]$ (m/s). The sizes of vehicle model are set to $L = 4.8$ m and $W = 1.8$ m.

4.1. Pose-Estimation Accuracy

In KITTI, four datasets, 0014, 0018, 0056, and 0057, were utilized to verify pose-estimation accuracy. These four datasets were collected from the city scenarios and contained many vehicles. The optical images of these datasets are shown in Figure 8. In addition to separately counting the estimation accuracy for each dataset, the average results of the deviations of the four datasets and the pose estimation of targets with sparse points, which means that the objects returned less than 50 points, were calculated at the same time. PE-MSS [22], PE-CPD [27], and the model-fitting method in [14] were implemented as the comparative experiments. The estimated poses of vehicles in the range of 80 m are utilized to perform the statistics. The provided labels in KITTI were employed to obtain the statistical deviations in position and angle, respectively, which are shown in Figure 9. The position deviation represents the distance between the midpoint of the ground truth and the midpoint of the fitting model. The angle deviation is the difference between the angle of ground truth and the angle of the estimated pose.

Figure 8. The optical images of KITTI datasets. (**a**–**d**) represent the scenarios in 0014, 0018, 0056 and 0057 datasets, respectively.

Figure 9a,b show statistical errors in position estimation. Among the four pose-estimation methods, the convex-hull model-fitting method in [14] had a large fluctuation on estimation accuracy, that is, it had well-estimated results in some scenarios (such as the 0057 dataset), but it performed poorly in other scenarios, such as the 0014 dataset and estimation with sparse point-clouds. The reason for this result is that the criteria for model fitting is single and cannot be adapted to the model fitting requirements for various scenarios. The center of the model is determined by the center of bounding box. For targets with sparse point-clouds and a missing contour, deviation between the position of the bounding box and the real target's center is large. The proposed PE-CHM solves this problem by inferring the intersection with the two visible edges and using the fixed model size to determine the target's position. Meanwhile, the proposed model-fitting criteria that compose four factors make the model adapt to the environment. The pose estimation of the three methods with fixed model size, namely, PE-CPD, PE-MSS, and PE-CHM, are basically stable for each dataset, showing robustness to the environment. The PE-CHM and PE-CPD algorithms had similar performance, and both had better estimation accuracy than PE-MSS, which indicates that, by introducing motion information, the target's pose-estimation accuracy is improved, especially for a target with incomplete contour.

Figure 9c,d show the deviation results of the yaw angles. The disadvantages of the convex-hull model-fitting method using a single criterion for pose estimation are more obvious. Deviations between estimation and ground truth are large and estimations are unstable. The PE-MSS method performed badly in some situations, such as in the 0057 dataset and in the case with sparse point-clouds. The reason is that estimation with PE-MSS easily falls into local optimal estimation when vehicles are distant with sparse point-clouds and incomplete contours. The performances of the PE-CPD method and the proposed PE-CHM method were similar. The mean of the angular deviation of PE-CHM was 0.0079 rad larger than PE-CPD, and 0.0801 and 0.1033 rad smaller than PE-MSS and convex-hull model fitting, respectively.

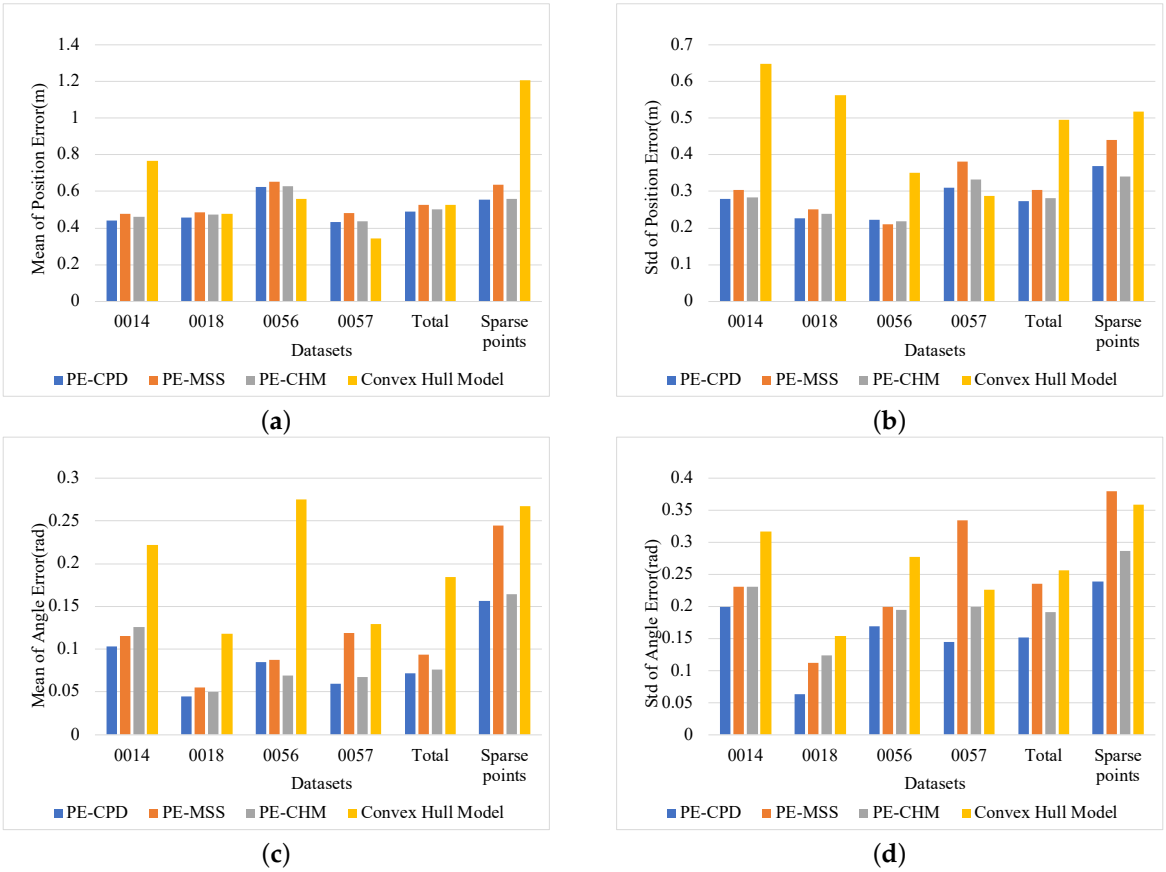


Figure 9. Statistical results of pose-estimation deviations. (**a**,**b**) Mean and standard deviation of position errors, respectively. (**c**,**d**) Mean and standard deviation of yaw angle errors, respectively.

Figure 10 shows an example of the pose-estimation results of two vehicles with only one visible short edge. Since PE-CPD used random sampling to determine the best fitted model, their results of 100 pose estimations varied over a small interval. PE-CHM and the convex-hull model use the objective function to determine the best bounding box without random sampling, therefore they only obtain one estimated pose. It is shown in Figure 10c that using the center of the bounding box as the pose of the objects brings a big deviation between estimated position and actual position. With the multifactor objective function in the convex-hull model and the position-inference process, the proposed PE-CHM had accurate estimation.

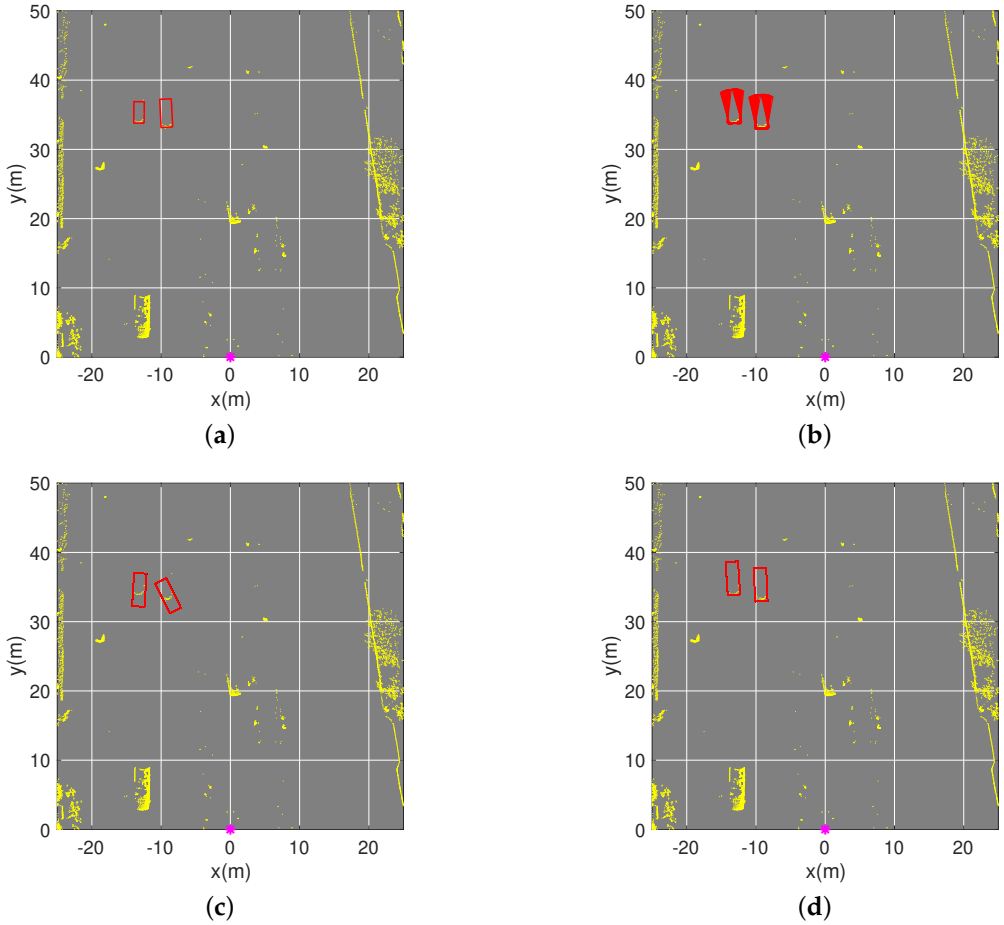


Figure 10. Pose estimation of two vehicles in the scenario. (**a**) Ground truth of their poses. (**b**–**d**) Estimated poses with the proposed pose estimation based on coherent point drift (PE-CPD), convex-hull model, and the proposed PE-CHM algorithm, respectively. Experiments were implemented one hundred times.

4.2. Dynamic Vehicle Detection

In this section, two datasets in KITTI, 0018 and 0056, were applied to verify vehicle detection performance. The distance of 40 m was utilized to divide the detection range into distant area and close area. There are totally 816 targets in 0018 dataset, among which 448 targets are in the close area. The 0056 dataset contains 407 targets in close area and 22 targets in distant area. PE-MSS-based detection [22], PE-CPD-based detection [27], and the detection method that introduces the convex-hull model [14] into our detection system were utilized as the comparative experiments. Detection range was set to $[0, 80]$ (m) from the Lidar. The detailed detection results are shown in Table 1. N_D and N_F denotes the numbers of detected vehicles and false alarms, respectively. N_T denotes the number of total vehicles in the scenarios. The precision P and recall R are calculated by Equation (15). As shown in Equation (16), the F_1 score is the harmonic average of P and R, which comprehensively represents the detection performance of the detection algorithm. The larger the value of F_1 score is, the better the detection performance will be.

$$P = \frac{N_D}{N_D + N_F}, R = \frac{N_D}{N_T} \tag{15}$$

$$F_1 = \frac{2PR}{P + R}. \tag{16}$$

Table 1 shows the detection results with the original convex-hull model and the PE-MSS-based method. The convex-hull method detected many false alarms from the scenarios, and the PE-MSS-

based method had poor detection performance in the distant area. The convex-hull model that was proposed in [14] only used the sum of minimum distances from the bounding box as the objective function to determine the optimal pose, in which the center of the optimal bounding box was utilized as the object position. Since objects in the scenario could be occluded by obstacles, and the contour of the objects in the distant area was partially lost, the obtained center position's change could cause a large deviation and the pose could be determined at a position with a large deviation on the yaw angle, which violates motion consistency and result in the loss of the vehicle. In the distant area, for objects with sparse point-clouds, the PE-MSS easily fell into the local extremum during pose estimation, resulting in wrong pose results. Inaccurate pose estimation also results in efficiency reduction in subsequent vehicle detection. For the close area, PE-MSS could have better pose estimation than the convex-hull model, so detection results were better.

Table 2 shows the results of the PE-CPD based method and the proposed PE-CHM-based method. Compared with the results in Table 1, both these methods had better detection performance in both the close and the distant area. Compared with the PE-CPD method, pose-estimation accuracy with the proposed PE-CHM was slightly reduced in the distant area, and the number of false alarms was slightly increased in the whole area. By comprehensively considering detection and false-alarm suppression performances, the F_1 scores of the PE-CPD method and the proposed method were 0.86 and 0.85, respectively, which indicates that the proposed method achieved similar detection performance to the PE-CPD method, offering the advantage of low computational complexity, which is explained in the next experiment.

Table 1. Dynamic vehicle detection results by the convex-hull model method and pose estimation with modified scaling series (PE-MSS) method.

Datasets	Range	N_T	Convex Hull Method					PE-MSS Method				
			N_D	N_F	P	R	F_1	N_D	N_F	P	R	F_1
0018	<40 m	448	348	17	0.95	0.78	0.86	357	22	0.94	0.80	0.86
	>40 m	368	196	70	0.74	0.53	0.62	39	9	0.81	0.11	0.18
	Total	816	544	87	0.86	0.67	0.75	396	31	0.93	0.49	0.64
0056	<40 m	407	314	97	0.76	0.78	0.77	359	32	0.92	0.88	0.90
	>40 m	22	9	52	0.15	0.41	0.22	0	9	0	0	-
	Total	429	323	149	0.68	0.76	0.72	359	41	0.90	0.84	0.87
Overall	<40 m	855	662	114	0.85	0.67	0.75	716	54	0.93	0.84	0.88
	>40 m	390	205	248	0.45	0.53	0.49	39	18	0.68	0.10	0.17
	Total	1245	867	362	0.71	0.70	0.70	755	72	0.91	0.61	0.73

Table 2. Dynamic vehicle detection results by the pose estimation based on coherent point drift (PE-CPD) method and the proposed pose estimation with convex-hull model (PE-CHM) method.

Datasets	Range	N_T	PE-CPD Method					PE-CHM Method				
			N_D	N_F	P	R	F_1	N_D	N_F	P	R	F_1
0018	<40 m	448	368	5	0.99	0.82	0.90	374	10	0.97	0.83	0.89
	>40 m	368	219	16	0.93	0.60	0.73	207	16	0.93	0.56	0.70
	Total	816	587	21	0.97	0.72	0.82	581	26	0.96	0.71	0.82
0056	<40 m	407	385	21	0.95	0.95	0.95	377	31	0.92	0.94	0.93
	>40 m	22	9	3	0.75	0.41	0.53	8	8	0.50	0.36	0.42
	Total	429	394	24	0.94	0.92	0.93	385	39	0.91	0.91	0.91
Overall	<40 m	855	753	26	0.97	0.88	0.92	751	41	0.95	0.88	0.91
	>40 m	390	228	19	0.92	0.58	0.72	215	24	0.90	0.55	0.68
	Total	1245	981	45	0.96	0.79	0.86	966	65	0.94	0.78	0.85

4.3. Operation Time

In this work, all experiments were implemented on a MacBook Pro with a 2.9 GHz Intel Core i5 CPU and 8 GB main memory. Detection methods were conducted in MATLAB 2017b. Detection operation time with PE-CPD [10], and proposed detection with PE-CHM are listed in Table 3.

Table 3. Operation-time comparison.

Methods	Total Time	Time/ Frame	Frame 221–230					
			Time	Read Data	Ground Tracking	RBNN	Virtual Scan	Detection Time/Frame
PE-CPD based	150.9 s	513 ms	1976 ms	42 ms	60 ms	681 ms	203 ms	98 ms
PE-CHM based	69.6 s	237 ms	1125 ms	38 ms	49 ms	628 ms	196 ms	21 ms

There are a total of 294 point-cloud frames in the 0056 dataset. The point-clouds were labeled with 0–293. The PE-CPD-based detection method utilized 150.9 s to finish detection, while PE-CHM-based detection reduced computation time by more than half. In frames 221–230, there was only a single vehicle in the scenario, and detailed statistics were recorded on the time consumption of each detection stage. It can be seen that the same preprocessing process had a similar operation time while ignoring the impact of the software. After removing the time consumption of the preprocessing process, PE-CPD needed an average of 98 ms to detect a point-cloud frame, and PE-CHM only needed 21 ms, which indicates that PE-CHM obviously improved pose-estimation efficiency.

5. Conclusions

To quickly and accurately detect dynamic vehicles in road scenarios, a novel pose-estimation method based on the convex-hull model and position inference was proposed in this paper. The main contributions are listed as follows.

First, multiple factors including the minimum boundary distance, minimum enclosed area, and minimum difference with the yaw angle, were utilized to construct the objective function for bounding box determination, which increases the adaptability of the convex-hull model to the environment. The construction of the convex-hull model is the foundation of the proposed pose estimation.

Second, a pose-estimation algorithm based on the convex-hull model and pose inference was proposed to estimate the pose of objects with a simple process. The proposed PE-CHM algorithm only uses contour point-clouds in the convex-hull set to determine the target pose, which reduces the amount of computation compared with the existing PE-CPD method.

Third, the proposed PE-CHM algorithm is applied to detect the dynamic vehicles in road scenarios. The experimental results showed that the proposed method could significantly reduce computation time while maintaining good detection efficiency.

Author Contributions: K.L. presented the algorithm and wrote the paper; J.W. reviewed and revised the paper.

Funding: This research was funded by the National Science Fund for Distinguished Young Scholars (51625503), the National Natural Science Foundation of China, the Major Project (61790561).

Acknowledgments: The authors are grateful to the reviewers and editor for their valuable comments and remarks to improve the quality of the contributions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zou, Y.; Zhang, W.; Weng, W.; Meng, Z. Multi-Vehicle Tracking via Real-Time Detection Probes and a Markov Decision Process Policy. *Sensors* **2019**, *19*, 1309. [CrossRef] [PubMed]
2. Velazquez-Pupo, R.; Sierra-Romero, A.; Torres-Roman, D.; Shkvarko, Y.; Santiago-Paz, J.; Gómez-Gutiérrez, D.; Robles-Valdez, D.; Hermosillo-Reynoso, F.; Romero-Delgado, M. Vehicle detection with occlusion handling, tracking, and OC-SVM classification: A high performance vision-based system. *Sensors* **2018**, *18*, 374. [CrossRef] [PubMed]
3. Guo, Z.; Cai, B.; Jiang, W.; Wang, J. Feature-based detection and classification of moving objects using LiDAR sensor. *IET Intell. Transp. Syst.* **2019**, *13*. . [CrossRef]
4. Nguyen, V.D.; Nguyen, T.T.; Nguyen, D.D.; Lee, S.J.; Jeon, J.W. A Fast Evolutionary Algorithm for Real-Time Vehicle Detection. *IEEE Trans. Veh. Technol.* **2013**, *62*, 2453–2468. [CrossRef]
5. Satzoda, R.K.; Trivedi, M.M. Multipart Vehicle Detection Using Symmetry-Derived Analysis and Active Learning. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 926–937. [CrossRef]
6. Tian, B.; Li, Y.; Li, B.; Wen, D. Rear-View Vehicle Detection and Tracking by Combining Multiple Parts for Complex Urban Surveillance. *IEEE Trans. Intell. Transp. Syst.* **2014**, *15*, 597–606. [CrossRef]
7. Stauffer, C.; Grimson, W.E.L. Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 747–757. [CrossRef]
8. Yang, B.; Zhang, S.; Tian, Y.; Li, B. Front-Vehicle Detection in Video Images Based on Temporal and Spatial Characteristics. *Sensors* **2019**, *19*, 1728. [CrossRef] [PubMed]
9. Ma, Y.; Anderson, J.; Crouch, S.; Shan, J. Moving Object Detection and Tracking with Doppler LiDAR. *Remote Sens.* **2019**, *11*, 1154. [CrossRef]
10. Liu, K.; Wang, W.; Tharmarasa, R.; Wang, J. Dynamic Vehicle Detection With Sparse Point Clouds Based on PE-CPD. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 1964–1977. [CrossRef]
11. Sualeh, M.; Kim, G.W. Dynamic Multi-LiDAR Based Multiple Object Detection and Tracking. *Sensors* **2019**, *19*, 1474. [CrossRef] [PubMed]
12. Yao, W.; Stilla, U. Comparison of Two Methods for Vehicle Extraction From Airborne LiDAR Data Toward Motion Analysis. *IEEE Geosci. Remote Sens. Lett.* **2011**, *8*, 607–611. [CrossRef]
13. Börcs, A.; Nagy, B.; Benedek, C. Instant Object Detection in Lidar Point Clouds. *IEEE Geosci. Remote Sens. Lett.* **2017**, *14*, 992–996. [CrossRef]
14. Börcs, A.; Nagy, B.; Baticz, M.; Benedek, C. A model-based approach for fast vehicle detection in continuously streamed urban LIDAR point clouds. In Proceedings of the Asian Conference on Computer Vision, Singapore, 1–2 November 2014; pp. 413–425.
15. Chavez-Garcia, R.O.; Aycard, O. Multiple sensor fusion and classification for moving object detection and tracking. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 525–534. [CrossRef]
16. Zeng, Y.; Hu, Y.; Liu, S.; Ye, J.; Han, Y.; Li, X.; Sun, N. RT3D: Real-Time 3-D Vehicle Detection in LiDAR Point Cloud for Autonomous Driving. *IEEE Robot. Autom. Lett.* **2018**, *3*, 3434–3440. [CrossRef]
17. Zhao, J.; Xu, H.; Liu, H.; Wu, J.; Zheng, Y.; Wu, D. Detection and tracking of pedestrians and vehicles using roadside LiDAR sensors. *Transp. Res. Part C Emerg. Technol.* **2019**, *100*, 68–87. [CrossRef]
18. Börcs, A.; Benedek, C. Extraction of Vehicle Groups in Airborne Lidar Point Clouds With Two-Level Point Processes. *IEEE Trans. Geosci. Remote Sens.* **2015**, *53*, 1475–1489. [CrossRef]
19. Fortin, B.; Lherbier, R.; Noyer, J.C. Feature Extraction in Scanning Laser Range Data Using Invariant Parameters: Application to Vehicle Detection. *IEEE Trans. Veh. Technol.* **2012**, *61*, 3838–3850. [CrossRef]
20. Fortin, B.; Lherbier, R.; Noyer, J.C. A Model-Based Joint Detection and Tracking Approach for Multi-Vehicle Tracking With Lidar Sensor. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 1883–1895. [CrossRef]
21. Kim, D.; Jo, K.; Lee, M.; Sunwoo, M. L-shape model switching-based precise motion tracking of moving vehicles using laser scanners. *IEEE Trans. Intell. Transp. Syst.* **2017**, *19*, 598–612. [CrossRef]
22. Chen, T.; Wang, R.; Dai, B.; Liu, D.; Song, J. Likelihood-Field-Model-Based Dynamic Vehicle Detection and Tracking for Self-Driving. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 3142–3158. [CrossRef]
23. Graham, R.L. An efficient algorithm for determining the convex hull of a finite planar set. *Inf. Process. Lett.* **1972**, *1*, 132–133. [CrossRef]
24. Liu, K.; Wang, W.; Wang, J. Pedestrian Detection with Lidar Point Clouds Based on Single Template Matching. *Electronics* **2019**, *8*, 780. [CrossRef]

25. Arastounia, M.; Lichti, D. Automatic object extraction from electrical substation point clouds. *Remote Sens.* **2015**, *7*, 15605–15629. [CrossRef]
26. Vaaja, M.; Kurkela, M.; Virtanen, J.P.; Maksimainen, M.; Hyyppä, H.; Hyyppä, J.; Tetri, E. Luminance-corrected 3D point clouds for road and street environments. *Remote Sens.* **2015**, *7*, 11389–11402. [CrossRef]
27. Liu, K.; Wang, W.; Tharmarasa, R.; Wang, J.; Zuo, Y. Ground Surface Filtering of 3D Point Clouds Based on Hybrid Regression Technique. *IEEE Access* **2019**, *7*, 23270–23284. [CrossRef]
28. Zeng, S. An object-tracking algorithm for 3-D range data using motion and surface estimation. *IEEE Trans. Intell. Transp. Syst.* **2013**, *14*, 1109–1118. [CrossRef]
29. Klasing, K.; Wollherr, D.; Buss, M. A clustering method for efficient segmentation of 3D laser data. In Proceedings of the IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; pp. 4043–4048.
30. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The KITTI dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [CrossRef]