# Likelihood-Field-Model-Based Dynamic Vehicle Detection and Tracking for Self-Driving

Tongtong Chen, Ruili Wang, Bin Dai, Daxue Liu, and Jinze Song

***Abstract*—Dynamic vehicle detection and tracking is crucial for self-driving in urban environments. The main problem of the previous beam-model-based algorithms is that they cannot detect and track dynamic vehicles that are occluded by other objects. In this paper, we develop a novel dynamic vehicle detection and tracking algorithm to solve this problem for our autonomous land vehicle (ALV), which is equipped with a Velodyne LIDAR and a GPS-aid inertial navigation system. For detection, our improved two-dimensional virtual scan is presented to detect the potential dynamic vehicles with a scan differencing operation. Then, for each potential dynamic vehicle, a novel likelihood-field-based vehicle measurement model is proposed to weight its possible poses. Finally, our newly modified scaling series algorithm and the importance sampling technique are adopted to estimate the initial pose and the corresponding velocity for each vehicle, respectively. The scaling series algorithm coupled with a Bayesian filter (SSBF) was previously used to handle the tactile localization problem in static background scenes. For tracking dynamic vehicles, we improve the SSBF by adding the ego-motion compensation so that the improved algorithm is able to update the pose and velocity for each vehicle in dynamic background scenes. Both the quantitative and qualitative experimental results validate the performance of our dynamic vehicle detection and tracking algorithm on the KITTI datasets and the Velodyne data collected by our ALV in dynamic urban environments.**

***Index Terms*—Autonomous land vehicle (ALV), Bayesian filter, likelihood-field-based vehicle measurement model, scaling series algorithm, vehicle detection and tracking.**

## I. Introduction

DYNAMIC vehicle detection and tracking is essential for an ALV navigation in urban environments. For example, when changing lanes in heavy traffic, the Autonomous Land Vehicle (ALV) is required to thoroughly understand the behaviours of the nearby vehicles, not only their velocities and poses at current position, but also the possible poses in the few seconds, to make sure that there is enough room to change lanes safely. Additionally, predicting when other vehicles are about to enter an intersection is also important for an ALV to plan a suitable path to pass through the intersection while avoiding collisions. Furthermore, it is believed that dynamic vehicle tracking is beneficial to the Simultaneous Localization And Mapping (SLAM) problem [1]. For these reasons, dynamic vehicle detection and tracking has become a hot research topic in mobile robotics over the past few decades. A typical urban scene with six human-driven vehicles at an X-junction is illustrated in Fig. 1.

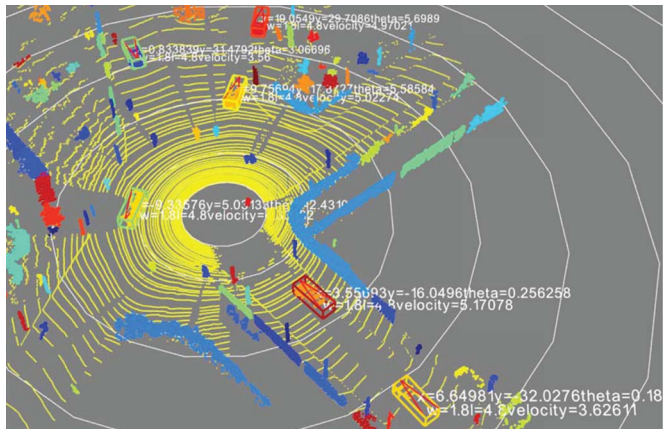


Fig. 1. Our algorithm tracks six dynamic vehicles with different poses while passing through an X-junction. The yellow points are ground, and the other points represent different objects. Six cuboids with different colors represent the tracked vehicles, and the red arrows on them are their orientations.

For our application, we are only concerned with laser-based dynamic vehicle detection and tracking from our ALV. Much research about this topic has been carried out over the last few years. Generally, these approaches can be divided into three subgroups: grid-based object detection and tracking [2]–[9], generic multi-purpose object detection and tracking [10]–[19] and model-based object detection and tracking [20]–[26].

*1) Grid-Based Object Detection and Tracking:* The grid-based methods mainly depend on the occupancy grid maps and avoid the data association problem in the other methods. The Bayesian Occupancy Filter (BOF) [2]–[4] is a generic occupancy grid framework that represents the surroundings as a two-dimensional grid of cells with occupancy state and velocity information. Then the Fast Cluster-Tracking Algorithm (FCTA) [27] is employed to detect and track the dynamic objects. Danescu *et al.* [5] proposed a particle-based approach to build the blocks of an environment. The particles with positions

Manuscript received November 30, 2015; revised March 6, 2016; accepted March 11, 2016. The work was supported in part by the National Natural Science Foundation of China under Grant 61075043 and Grant 61375050. The Associate Editor for this paper was J. E. Naranjo.

T. Chen is with the College of Mechatronic Engineering and Automation, National University of Defense Technology, Changsha 410073, China (e-mail: chentongtong5208@gmail.com).

R. Wang is with the School of Engineering and Advanced Technology, Massey University, Auckland 0632, New Zealand (e-mail: r.wang@massey.ac.nz).

B. Dai and D. Liu are with the Unmanned System Institute, National University of Defense Technology, Changsha 410073, China (e-mail: bindai.cs@gmail.com; liudaxue78@gmail.com).

J. Song is with the Graduate School, National University of Defense Technology, Hunan 410073, China (e-mail: nwsac97@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2016.2542258

and velocities in a particular cell represent the cell's velocity distribution and occupancy likelihood. The neighbour cells with similar speeds are clustered, and each cluster is represented in the form of an oriented cuboid. Moras *et al.* [6] employed the Dempster-Shafer theory on a grid representation to detect the moving objects in the surrounding area of the ego-vehicle. The movements of the objects can be modelled by analysing the conflicting information. Schutz *et al.* [7] proposed an object tracking algorithm without an approximated model. An object-local occupancy grid map is employed to model the free-formed object shapes from some raw laser measurements. Jungnickel *et al.* [8] utilized the Dempster-Shafer theory to estimate a dynamic belief of conflicting cells in an occupancy grid map. The dynamic cells are accumulated in an object-local occupancy grid map that uses the detached grid cells to model vehicles. Tanzmeister *et al.* [9] employed the Dempster-Shafer theory to fuse a particle map and an inverse sensor map. It can estimate a uniform, low-level, grid-based environmental model that includes dynamic objects with velocities. The grid-based methods are mainly utilized in SLAM rather than tracking, because of the low computational efficiency with fine grids [19].

*2) Generic Multi-Purpose Object Detection and Tracking:* The generic multi-purpose object detection and tracking methods can also be divided into two classes.

One class discards the three-dimensional shape information, and represents an object by its centroid. The Kalman filter is exploited to perform the tracking. Leonard *et al.* [10] employed a voting scheme to associate the new group of trunks to the old one, then a trivial Kalman filter is utilized to estimate the velocity of the bounding box of each trunk. Azim *et al.* [11] detected the dynamic objects in an octree-based occupancy grid map by analysing the inconsistencies between two consecutive scans. The centroid of the dynamic object is tracked with a Kalman filter. Kaestner *et al.* [12] proposed a generative object detection algorithm to extract dynamic objects with varying sizes and shapes in static background scenes. These dynamic objects are tracked with linear Kalman state estimation filters. Based on [12], Yang *et al.* [13] brought in the ego-motion compensation to detect and track objects in dynamic background scenes. Himmelsbach *et al.* [14] utilized the top-down knowledge about the geometries of the existing trackers to segment the moving objects from the static background. Then a Bayesian filter is used to update the state for each tracker. Choi *et al.* [15] proposed a novel algorithm for Multi-Target Tracking (MTT) with a Velodyne LIDAR. Object geometry is utilized to compensate the unintended dynamics caused by the shape change and occlusion in a Kalman filter. In [16], Wang *et al.* proposed a non-parametric representation to model various objects with different shapes. Each object model is tracked with a Bayesian filter to estimate the joint state that contains motion state and shape information. All these methods are computationally efficient, but not very accurate.

Other methods take advantage of the full three-dimensional shape or point cloud information, and utilize the matching technique to align the point clouds of the same object in the consecutive scans. Moosmann *et al.* [17] used a local convexity criterion [28] to generate various object hypotheses without a specific object model. The state is estimated by aligning the tracker's appearance points with the full input points using the Iterative Closest Point (ICP) algorithm. However, the performance degrades without a good initial alignment. Held *et al.* [18] combined a sparse laser and a high-resolution camera to obtain a dense coloured point cloud, then a colour-augmented filtered grid search algorithm is used to align two consecutive frames of the coloured point clouds for accurate velocity estimation. Recently, based on their measurement model and motion model, Held *et al.* [19] introduced Annealed Dynamic Histograms (ADH) to accurately track moving objects in real time. Given the point correspondences between two consecutive scans, the measurement and motion model can then be computed. The ADH tracker can achieve accurate velocity estimation when given promising data associations.

*3) Model-Based Object Detection and Tracking:* The model-based methods are exploited to track objects that have specific shapes, e.g., a two-dimensional rectangular model for a car, an elliptical model for a cyclist. Currently, most of the successful object tracking methods seem to be model-based, and researchers have to design different models for different objects [17]. Morris *et al.* [20] took view-dependent self-occlusion effects into account, and used 4 rectangles to represent the inside, outside and two visible surfaces of a vehicle, respectively. A gradient-based optimization technique is used to maximize the integrals of the associated points over the four rectangles to track a vehicle. Petrovskaya *et al.* [21] proposed a novel beam-based rectangular vehicle measurement model. The dynamic vehicles are detected with a scan differencing operation in a two-dimensional virtual scan, and the global map, motion evidence and motion consistency criteria are used to remove the false alarms. These finally detected vehicles are tracked with the Rao-Blackwellized particle filters. Based on reference [21], Wojke *et al.* [22] proposed temporal and geometric cues to detect the dynamic vehicles. The approach can obtain promising performance in unstructured forest regions without a global map. These detected vehicles are also tracked with the Rao-Blackwellized particle filters. Based on the measurement model in [21], Vu *et al.* [23] solved the dynamic object detection and tracking problem as finding the most likely trajectory that satisfies the constraints of the measurement model and the smoothness between two consecutive scans. A data-driven Markov chain Monte Carlo technique is exploited to search the solution efficiently in the spatio-temporal space. In [24], Liu *et al.* proposed a modified beam-based vehicle measurement model for a titled two-dimensional laser range finder. A particle filter is employed to obtain a preliminary tracking result which is then refined by a new precise tracking method. Both qualitative and quantitative experiments validate its performance. Granstrom *et al.* [25] used a multiple model Probability Hypothesis Density (PHD) filter to track dynamic vehicles, under an assumed rectangular shape, in heavy traffic that contains multiple cars, occlusions and manoeuver changes. However, the measurement model proposed in [21] and its variants [22]–[24] can only track the vehicles that are closest to the ego-vehicle; they cannot be utilized to track the vehicles that are fully occluded in the $xy$ plane by other objects in the crowded urban environments. This issue motivates the approach presented in this paper.

In this paper, we present a novel dynamic vehicle detection and tracking algorithm based on the likelihood field model for an ALV equipped with a Velodyne LIDAR and a GPS-aid Inertial Navigation System (GPS-INS). Our improved two-dimensional virtual scan is proposed to detect the potential dynamic vehicles with a scan differencing operation. Compared with the original one in [29], our improved virtual scan is able to detect the potential dynamic vehicles that are occluded in the $xy$ plane by other objects. For each potential dynamic vehicle, a novel likelihood-field-based vehicle measurement model is utilized to weight its possible poses. Then, the initial pose and velocity of each dynamic vehicle is estimated using our newly modified Scaling Series algorithm and the importance sampling technique, respectively. For tracking, we improve the SSBF by adding the ego-motion compensation for each tracker. Firstly, the ego-motion compensation is utilized to unify the ego-vehicle's local coordinate frames in two consecutive scans. Then our improved Scaling Series algorithm is run on the associated measurements to obtain a weighted particle set with a uniform prior. Finally, the weights are adjusted via the Bayesian recursion equation [30] to capture the vehicle dynamic model. These final weighted particles approximate the posterior belief of the tracker and the one with the maximum weight is chosen as the output of the tracking result. The proposed algorithm in this paper is suitable for any type of range sensors however we have demonstrated it with the data acquired from a Velodyne HDL-64E LIDAR because of its accurate and high frequency three-dimensional measurements.

In contrast to the previous references about vehicle detection and tracking, the contributions of this paper are three-fold. The significant one is that we propose a likelihood-field-based vehicle measurement model. This model is also coupled with our newly modified Scaling Series algorithm to estimate the poses of the vehicles efficiently. The second contribution is that an improved two-dimensional virtual scan is proposed to detect the potential dynamic vehicles, even they are occluded in the $xy$ plane by other objects. The final enhancement is that the Scaling Series algorithm, coupled with a Bayesian Filter (SSBF) that was used to handle the tactile localization problem in static background scenes in [31], is improved by adding the ego-motion compensation to track the dynamic vehicles in dynamic background environments.

This paper extends our previous work [32]–[34] that was presented at three conferences. Four novel contributions are listed below:

1) We construct a look-up table offline to approximate the most time-consuming function erf in the likelihood-field-based vehicle measurement model (Section II-B). The look-up table makes Algorithm 1 about 4.5 times faster than our previous studies (Section IV-A1).
(2) We bring in an annealing process for the measurement noise in Algorithm 1 (Section II-B and C), which makes Algorithm 1 much more stable (Section IV-A2).
(3) We improve the SSBF by adding the ego-motion compensation in Algorithm 2 (Section III-C2), which makes the vehicle tracking algorithm available in dynamic background scenes.
(4) We conduct a new quantitative experiment on the KITTI datasets [35], where our dynamic vehicle detection algorithm outperforms Petrovskaya's algorithm [29] (Section IV-B).

The rest of the paper is structured as follows: in the next section, both the novel likelihood-field-based vehicle measurement model and our newly modified Scaling Series algorithm are described. They are the bases of the successive dynamic vehicle detection and tracking algorithm. Section III describes the dynamic vehicle detection and tracking algorithm in detail. Both the quantitative and qualitative experimental results are shown in Section IV. Our algorithm is compared with the state of the art model-based dynamic vehicle detection and tracking algorithm in [21] on the KITTI datasets [35] and the Velodyne data collected by our ALV in various urban scenes. The results show the promising performance of our algorithm. Finally, conclusions and an outline of the future work are given in Section V.

## II. Vehicle Pose Estimation

In this paper, vehicle pose estimation is the basis of our dynamic vehicle detection and tracking algorithm. In this section, a more detailed description of the vehicle pose estimation algorithm than our previous publication [32] is presented. Furthermore, in order to improve the real-time performance, we construct a look-up table for the most time-consuming function erf. Also we add an annealing process for the measurement noise in the vehicle pose estimation algorithm to improve its stability. Compared with reference [29], our algorithm can also estimate the poses of the fully occluded vehicles in the $xy$ plane.

### A. Likelihood-Field-Based Vehicle Measurement Model

The likelihood field model has been widely used in mobile robotics [30]. Instead of the beam-based vehicle measurement model in [29], we propose to utilize four rectangles to represent the different parts of a vehicle and construct the following vehicle measurement model based on the likelihood field. Given a vehicle's pose $X = (x, y, \theta)$ ($(x, y)$[1] represents the position of the vehicle and $\theta$ is its orientation) with fixed width $W$, length $L$, and a cluster of associated measurements $Z = \{q_1, q_2, \ldots, q_n\}$. The measurement likelihood $p(Z|X)$ is defined as follows.

In this paper, each measurement $q_i = (x_i, y_i, z_i)^T$ is modelled as a two-dimensional normal distribution $p(x, y)$ in the $xy$ plane, with $x, y$ independent [20]. As we believe that the tracked vehicles are always running on the ground surface, the heights of their measurements are not so relevant to our applications

$$p(x,y) = \frac{1}{2\pi\sigma^2} \exp\left\{-\frac{(x-x_i)^2}{2\sigma^2}\right\} \exp\left\{-\frac{(y-y_i)^2}{2\sigma^2}\right\} \quad (1)$$

where $\sigma^2$ is the variance which represents the measurement noise.

[1]The meter is used as the length unit in this paper unless otherwise noted.

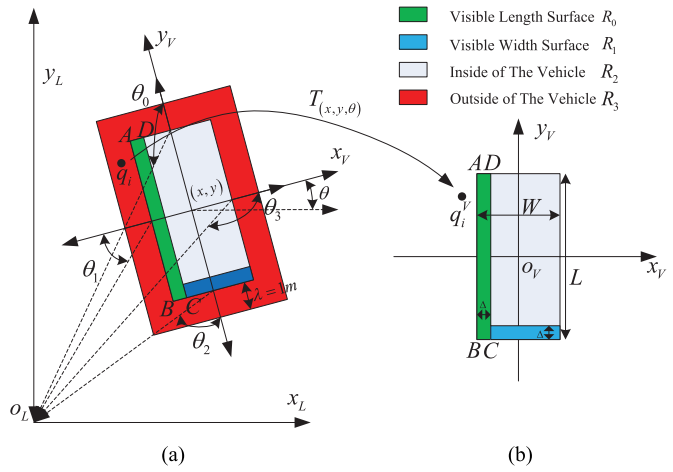



Fig. 2. Likelihood-field-based vehicle measurement model. (a) Four geometric regions of the model in the LIDAR coordinate frame $o_L x_L y_L$. The green and blue rectangles are two visible surfaces of the model; the gray and red rectangles represent the inside and outside regions of the model, respectively. (b) Target vehicle's local coordinate frame $o_V x_V y_V$ with the origin at the geometric center of the rectangle. In this paper, the width and length of the model are fixed to constants, and the width of the visible surface of the model is set to $\Delta$.

The likelihood-field-based vehicle measurement model is illustrated in Fig. 2. The green, blue, grey and red rectangles represent the visible length surface, the visible width surfaces, inside and outside regions of the vehicle, respectively. For every edge of the vehicle, we define the direction pointing to the origin $o_L$ of the LIDAR coordinate frame $o_L x_L y_L$[2] from its center as the corresponding *orientation vector*, and the orientation that is perpendicular to the edge and pointing outside the vehicle as its *norm vector* [32]. If the angle $\theta_i$ between the *orientation vector* and *norm vector* is less than $\pi/2$, then the edge is defined as a visible surface, e.g., angles $\theta_1, \theta_2$ of the visible length surface $R_0$ and the width surfaces $R_1$ in Fig. 2. For a vehicle with pose $X$ in the LIDAR coordinate frame $o_L x_L y_L$, we would expect that most of the measurements will fall in the visible surfaces of the vehicle (i.e. the green and blue rectangles), some measurements will be inside of the vehicle (i.e. the grey rectangle), while few will fall outside the vehicle in the red region, as it is believed that there are no objects in the vicinity of a moving vehicle [32].

In this paper, each measurement in $Z$ is assumed to be independent. Thus, the vehicle measurement model that contains $n$ measurements in $Z$ can be defined as follows:

$$p(Z|X) = \prod_{i=1}^{n} p(q_i|X). \tag{2}$$

For every target-vehicle, a target-vehicle's local coordinate frame $o_V x_V y_V$ is constructed at the geometric centre of the rectangle, with $x, y$ axes pointing right and forward, respectively, as shown in Fig. 2(b). The likelihood of each measurement $q_i$ is modelled as the exponential of the weighted sum of the integrals of the measurement $q_i^V$ over the four regions in $o_V x_V y_V$ [32]

$$p(q_i|X) = \eta_i \exp\left(\alpha \sum_{j=0}^{3} c_j I\left(q_i^V, R_j\right)\right)$$

$$q_i^V = \begin{bmatrix} x_i^V \\ y_i^V \end{bmatrix} = \overbrace{\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}}^{T_{x,y,\theta}} \begin{bmatrix} x_i - x \\ y_i - y \end{bmatrix} \tag{3}$$

where $q_i$ is a measurement in the LIDAR coordinate system $o_L x_L y_L$, and $q_i^V$ is the corresponding point in the target-vehicle's local coordinate frame $o_V x_V y_V$ obtained from the transformation $T_{x,y,\theta}$, as shown in (3). $\eta_i$ is a normalization constant. $R_0$, $R_1$, $R_2$, and $R_3$ are the visible length rectangle, the visible width rectangle, the inside and outside regions of the vehicle, respectively (i.e. the green, blue, grey and red regions in Fig. 2), and $c_0$, $c_1$, $c_2$, and $c_3$ are the corresponding integral weights over these regions. As we expect that most of the measurements would fall in the visible surface or inside the vehicle, $c_0$, $c_1$ and $c_2$ are chosen to be greater than zero, while $c_3$ is less than zero. $\alpha = 1/\sqrt{\sum_{k=0}^{2} \int\int_{R_k} c_k^2 dx dy}$. $I(q_i^V, R_j)$ represents the integral of the measurement $q_i^V$ over Rectangle $R_j$ in the target-vehicle's local coordinate system $o_V x_V y_V$, and it can be calculated as follow.

Assume that $I_A$, $I_B$, $I_C$, and $I_D$ are the integrals of the measurement $q_i^V$ over the quarter plane extending to positive infinity from the positions $A = (x_a^V, y_a^V)$, $B = (x_b^V, y_b^V)$, $C = (x_c^V, y_c^V)$ and $D = (x_d^V, y_d^V)$, respectively, in the target-vehicle's local coordinate frame $o_V x_V y_V$. The integral $I(q_i^V, R_0)$ of $q_i^V$ over the visible length rectangle $R_0$ can be calculated using (4) [32]

$$\begin{aligned} I\left(q_i^V, R_0\right) &= I_D + I_B - I_A - I_C; \\ \text{where } I_A &= \frac{1}{4}\left(1 + \text{erf}\left(\frac{x_i^V - x_a^V}{\sqrt{2}\sigma}\right)\right) \\ &\quad \times \left(1 + \text{erf}\left(\frac{y_i^V - y_a^V}{\sqrt{2}\sigma}\right)\right) \\ I_B &= \frac{1}{4}\left(1 + \text{erf}\left(\frac{x_i^V - x_b^V}{\sqrt{2}\sigma}\right)\right) \\ &\quad \times \left(1 + \text{erf}\left(\frac{y_i^V - y_b^V}{\sqrt{2}\sigma}\right)\right) \\ I_C &= \frac{1}{4}\left(1 + \text{erf}\left(\frac{x_i^V - x_c^V}{\sqrt{2}\sigma}\right)\right) \\ &\quad \times \left(1 + \text{erf}\left(\frac{y_i^V - y_c^V}{\sqrt{2}\sigma}\right)\right) \\ I_D &= \frac{1}{4}\left(1 + \text{erf}\left(\frac{x_i^V - x_d^V}{\sqrt{2}\sigma}\right)\right) \\ &\quad \times \left(1 + \text{erf}\left(\frac{y_i^V - y_d^V}{\sqrt{2}\sigma}\right)\right). \end{aligned} \tag{4}$$

[2]The origin $o_L$ of the coordinate frame $o_L x_L y_L$ is the centre of the LIDAR, with $x$-axis pointing right side of the ego-vehicle and $y$-axis pointing its driving direction.

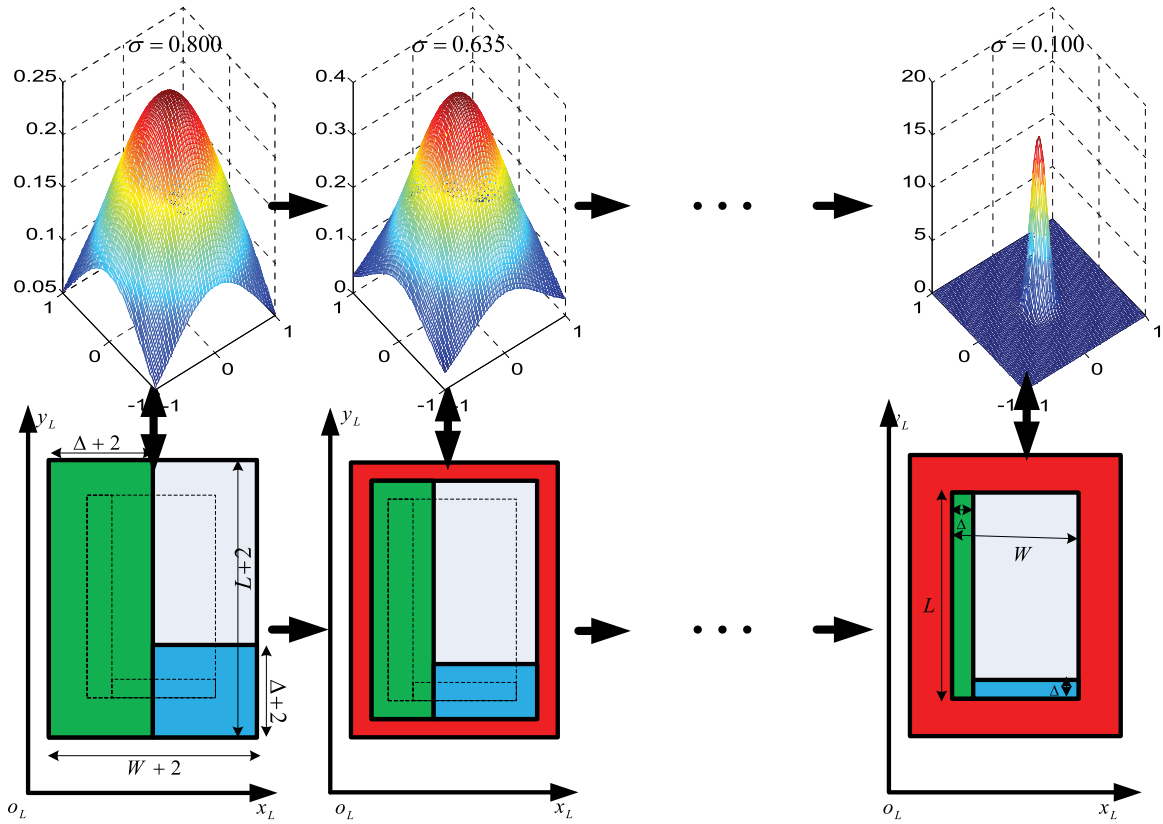



Fig. 3. Detailed annealing processes of the vehicle measurement model and measurement noise. The top row shows the annealing process of the measurement noise. $\sigma$ becomes smaller gradually. The bottom row presents the annealing process of the vehicle measurement model. The width of the visible surface becomes smaller synchronously. The green, blue, gray, and red regions represent the visible length and width surfaces, and regions inside and around the vehicle, respectively. The bottom left image represents the most relaxed vehicle measurement model, which is iteratively annealed to the normal one in the bottom right image. The biggest measurement noise and smallest measurement noise correspond to the most relaxed and normal vehicle measurement models, respectively.

Note that the likelihood-field-based vehicle measurement model is built directly on the original three-dimensional measurements $q_i$, while the measurement model in reference [29] and its variants [22]–[24] use the lengths of the two-dimensional rays in the $xy$ plane as the measurements. Thus, for the vehicles that are occluded by other objects in the $xy$ plane, but can still be detected by the Velodyne LIDAR, our measurement model can also estimate their poses accurately, while references [22]–[24], [29] failed to do so. Additionally, for the vehicles that are far away from the ego-vehicle, our measurement model is more robust than that in [22]–[24], [29].

### B. Model Computation

As the associated cluster $Z = \{q_1, q_2, \ldots, q_n\}$ may contain many measurements ($n$ is big) when the target-vehicle is very close to the ego-vehicle, the efficiency of computing $p(Z|X)$ may be very low, because function $\text{erf}(d/\sqrt{2}\sigma)$ in (4) will be calculated 32 times for each measurement $q_i$. The calculation of erf is the most time-consuming part of our likelihood-field-based vehicle measurement model. To solve this problem, in this paper, we define $d$ as a discrete variable and construct a look-up table offline to approximate to $\text{erf}(d/\sqrt{2}\sigma)$, because $\text{erf}(d/\sqrt{2}\sigma)$ is only a function of $d$ when given $\sigma$. The calculation of $\text{erf}(d/\sqrt{2}\sigma)$ can then be transformed to search an approximated value from the look-up table. In this way, the computational efficiency can be improved significantly.

### C. Annealing Processes of the Measurement Model and Noise

In this paper, the likelihood-field-based vehicle measurement model is combined with our modified Scaling Series algorithm to accelerate the vehicle pose estimation algorithm. The most important step of the combined algorithm is how to iteratively anneal from an artificially relaxed model to a normal one [32]. Compared with the method in [29] which only anneals the measurement model, in this paper, we propose to anneal both the measurement model and the measurement noise, synchronously, according to the character of our measurement model, to obtain a more stable result. The detailed process is illustrated in Fig. 3. The top and bottom rows present the annealing processes of the measurement noise and the vehicle measurement model, respectively. They are annealed synchronously. The top images show function $p(x, y)$ with different measurement noises $\sigma s$, while the bottom ones illustrate our measurement model with different widths of the visible surfaces. In the bottom images, the green and blue rectangular shapes are the visible surfaces of the vehicle, and the grey one is the region inside the vehicle, while the red represents the free region outside the vehicle. The bottom left image represents the most relaxed measurement model, while the bottom right shows the normal one.

In the beginning, the normal width $\Delta$ of the visible surface is inflated by 1 metre both inside and outside of the vehicle measurement model. Thus the width of the relaxed visible surface is $\Delta + 2$ [32]. In that case, the visible length and width surfaces $R_0, R_1$ are expanded to fully occupy the free region $R_3$ around the target-vehicle, as shown in the bottom left image of Fig. 3. Meanwhile, we enlarge the variance $\sigma^2$ of the measurement noise to be $0.8^2$, which makes the model $p(x, y)$ of each measurement much smoother. Both ways smooth the measurement likelihood $p(Z|X)$. Nevertheless, with the most relaxed vehicle measurement model and the enlarged measurement noise, some unreasonable particles that are located at 1 metre away from the actual target-vehicle will be deleted directly. In this way, when the inflating width of the visible surface and the variance $\sigma^2$ of the measurement noise are iteratively annealed from $\Delta + 2$, $0.8^2$ to $\Delta$, $0.1^2$, respectively, the high likelihood region will become smaller smoothly and converge to the actual vehicle pose. Adding the annealing process of the measurement noise makes our pose estimation algorithm much more stable, to some extent.

### *D. Our Modified Scaling Series Algorithm for Pose Estimation*

In this paper, the vehicle pose estimation problem in crowded urban environments inherits large uncertainties. In the aspect of position, the sampling radius may be several metres; additionally, the vehicles can move in arbitrary directions in urban environments, and the uncertainty of the direction is also great. Thus, the Scaling Series algorithm, which was proposed to deal with the problem in case of large uncertainty in [31], is utilized to estimate the poses of the vehicles in urban environments in this paper. Meanwhile, we modify the Scaling Series algorithm by bringing in an annealing process for the measurement noise. The detailed pose estimation algorithm is shown in Algorithm 1.

**Algorithm 1** Pose Estimation with Our Modified Scaling Series Algorithm (PE_MSS)

**Input**: $Z = \{q_1, q_2, \ldots, q_n\}$, $M$, $\Delta$, $N$, $L$
1: $(C_x^{\mathrm{mb}}, C_y^{\mathrm{mb}}, \theta^{\mathrm{mb}} = \min \mathrm{AreaRect}(Z)$;
2: $w = 1.0$, $\theta = \pi/2$, $\sigma = 0.8$, zoom $= 1/\sqrt[3]{2}$, $V_0 = (C_x^{\mathrm{mb}}$, $C_y^{\mathrm{mb}}, \theta^{\mathrm{mb}}, L/2, \theta)$;
3: **for** $n = 1 : N$ **do**
4: $\overline{\chi_n} = \mathrm{Rejection\_Sampling}(V_{n-1}, M)$;
5: $\omega_n = \mathrm{Compute\_Normalized\_Weights}(\overline{\chi_n}, 2w + \Delta, \sigma, Z)$;
6: $\chi_n = \mathrm{Prune}(\overline{\chi_n}, \omega_n)$;
7: $V_n = \mathrm{Union\_Delta\_Neighbourhoods}(\chi_n, w, \theta)$;
8: $w = w * \mathrm{zoom}$, $\theta = \theta * \mathrm{zoom}$, $\sigma = \sigma * \mathrm{zoom}$;
9: **end for**
10: $\chi = \mathrm{Rejection\_Sampling}(V_N, M)$;
11: $\omega = \mathrm{Compute\_Normalized\_Weights}(\chi, \Delta, \sigma, Z)$;
**Output**: $\{\chi, \omega\}$ // a weighted particle set approximating the prior belief

The algorithm takes as input a cluster of three-dimensional measurements $Z = \{q_1, q_2, \ldots, q_n\}$, the maximum number of iterations $N$, the normal width $\Delta$ of the visible surface in the normal vehicle measurement model, the particle number $M$ per ellipsoid neighbourhood, the length $L$ of the target-vehicle, and it outputs a set of weighted particles $\{\chi, \omega\}$ [32]. Each particle $(X, w)$ represents one possible pose $X$ of the target-vehicle with a belief $w$.

The first step of Algorithm 1 is to find the minimum area bounding rectangle of the projections in the $xy$ plane of the cluster $Z$. This corresponds to Line 1, and the position and orientation of the bounding rectangle are $(C_x^{\mathrm{mb}}, C_y^{\mathrm{mb}})$, and $\theta^{\mathrm{mb}}$, respectively. We believe that the actual position and orientation of the target-vehicle are near $(C_x^{\mathrm{mb}}, C_y^{\mathrm{mb}})$ and $\theta^{\mathrm{mb}}$. Thus, the initial sampling region is set to be an ellipsoid $V_0$ with the centroid $(C_x^{\mathrm{mb}}, C_y^{\mathrm{mb}}, \theta^{\mathrm{mb}})$ in Line 2. The sampling radii of the position and orientation are $L/2$, $\theta$, respectively. Additionally, we initialize $w = 1.0$, $\sigma = 0.8$ to inflate the vehicle measurement model and the measurement noise.

Lines 3–9 describe the annealing processes of the measurement model and the measurement noise. Drawing a set $\overline{\chi_n}$ with $M$ particles from $V_{n-1}$ uniformly corresponds to function *Rejection_Sampling* in Line 4. For every ellipsoid neighbourhood, $M$ particles are drawn to form a particle set $\overline{\chi_n}$ [32]. The detailed implementation can be found in [31]. The relaxed likelihood-field-based vehicle measurement model coupled with an enlarged variance of the measurement noise is utilized to calculate the corresponding measurement likelihood $p(Z|X)$ for each particle in $\overline{\chi_n}$. As the particles in $\overline{\chi_n}$ are with a uniform prior, these measurement likelihoods, after being normalized, can be regarded as their weights. This process corresponds to function *Compute_Normalized_Weights* in Line 5. The next step is to prune the particles with low weights in $\overline{\chi_n}$ in Line 6. Note that the particles with high weights in $\overline{\chi_n}$ will not be duplicated in this step. For each residual particle, function *Union_Delta_Neighbourhoods* in Line 7 is exploited to construct an ellipsoid neighbourhood with the particle itself as the centroid. The sampling radii of the position and orientation are $w, \theta$, respectively. The last step corresponds to Line 8. It serves to anneal the sampling radii $w, \theta$ of the position and orientation. The variance $\sigma^2$ of the measurement noise is annealed synchronously.

After the annealing processes, the final particle set $\chi$ is drawn with function *Rejection_Sampling* in Line 10, and the corresponding normalized weights $\omega$ is calculated with the normal vehicle measurement model and measurement noise in Line 11. The weighted particle set $\{\chi, \omega\}$ approximates the prior belief of the pose of the target-vehicle. From the weighted particle set, the particle with the maximum weight is selected as the initial pose of the target-vehicle when implementing the dynamic vehicle detection algorithm in Section III-B2.

Fig. 4 illustrates the intermediate results in the annealing process when estimating the pose of the yellow vehicle (the red cuboid) in the top left image. Image 1 shows the initial pose of the vehicle by finding the minimum area bounding rectangle of the measurements (i.e., the yellow pixels). Image 10 presents the final weighted particle set which is obtained by using the normal vehicle measurement model and the normal


Fig. 4. Intermediate results in the annealing process when estimating the pose of the yellow vehicle (the red cuboid) in the left top image. The blue and red rectangles represent the weighted particles.

measurement noise. The particle with the maximum weight among the final weighted particles is shown in Image 11. It is also the initial pose of the yellow vehicle when implementing the dynamic vehicle detection algorithm in Section III-B2.

## III. Dynamic Vehicle Detection and Tracking

In this section, a more detailed introduction of our dynamic vehicle detection and tracking algorithm is presented. All the computations in this section are made with respect to the ego-vehicle's local coordinate frame, rather than the world coordinate frame. Thus the detection and tracking results can be used for the local path planning directly. We assume that the ego-vehicle's local coordinate system is tied to the ego-vehicle's center with $x$-axis pointing right, $y$-axis pointing forward and $z$-axis pointing upward.

### A. Overview

The framework of our dynamic vehicle detection and tracking algorithm is illustrated in Fig. 5. In this paper, the points that are collected within one scan (required 0.1 s) are treated as if they are collected at the same time. A scan collected at time step $t$ is represented as $S_t$. The inputs of the algorithm are Scan $S_t$, the global coordinate $(x_t^e, y_t^e, z_t^e)$ and the yaw angle $\psi_t^e$ of the ego-vehicle, where the global coordinate $(x_t^e, y_t^e, z_t^e)$ and the yaw angle $\psi_t^e$ are provided by a GPS-INS. The outputs of the algorithm are a list of Trackers ${}^tT_*$. The roll and pitch angles of the ego-vehicle are ignored in this paper since the ground surface is assumed to be flat in urban environments. At the beginning, Scan $S_t$ is preprocessed. Ground segmentation is performed to remove the points on the ground, and the residual points are clustered into different objects $O_t$. Both the objects $O_{t-1}$ at time step $t-1$ after the ego-motion compensation and the objects $O_t$ are projected into our improved two-dimensional virtual scan. Dynamic objects can be detected through a scan differencing operation in the two-dimensional virtual scan and will become dynamic vehicle hypotheses $T_t$ when meeting the motion evidence criterion proposed in [29]. Meanwhile, at time step $t$, the dynamic vehicle hypotheses $T_{t-1}$ and the existing

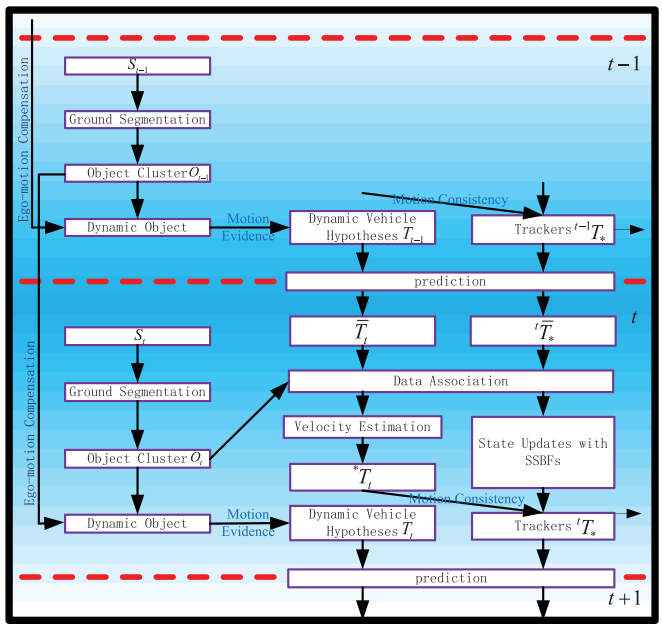



Fig. 5. Framework of our dynamic vehicle detection and tracking algorithm in this paper.

Trackers ${}^{t-1}T_*$ at the last time step are predicted to become $\overline{T}_t$ and ${}^t\overline{T}_*$, which are all oriented boxes, using the vehicle dynamic model, respectively. The objects $O_t$ in the scope of $\overline{T}_t$ and ${}^t\overline{T}_*$ are the associated measurements of $T_{t-1}$ and ${}^{t-1}T_*$ at time step $t$, respectively. For these existing Trackers ${}^{t-1}T_*$, their poses and velocities are updated to become ${}^tT_*$ using the SSBFs, which are improved by adding the ego-motion compensation in this paper. Our improved SSBFs can track dynamic vehicles in dynamic background environments. For the dynamic vehicle hypotheses $T_{t-1}$, the velocities are estimated again with the associated objects using the importance sampling technique, respectively. The particles ${}^*T_t$ with the best velocities will be finally merged into the existing trackers ${}^tT_*$ if their motion consistency criteria are satisfied too. Otherwise they will be discarded. The final existing Trackers ${}^tT_*$ and the newly generated dynamic vehicle hypotheses $T_t$ will be used in the next time step circularly. In this paper, dynamic vehicles that are 50 meters away from the ego-vehicle will be deleted from Trackers ${}^tT_*$.

### B. New Trackers Generation

In this subsection, we assume that the ground segmentation and object clustering steps have been performed for Scan $S_t$ collected at time step $t$. New trackers generation starts from a series of objects $O_t = \{o_t^1, o_t^2, \ldots, o_t^n\}$, and outputs some new trackers ${}^*T_t$ that will be merged into the existing trackers ${}^tT_*$ and some dynamic vehicle hypotheses $T_t$. New trackers generation contains two steps: dynamic object detection and then dynamic vehicle hypotheses generation and verification. They are explained below.

*1) Dynamic Object Detection:* In order to improve the computational efficiency, a two-dimensional polar grid map (termed as a *two-dimensional virtual scan*) that will be partitioned into

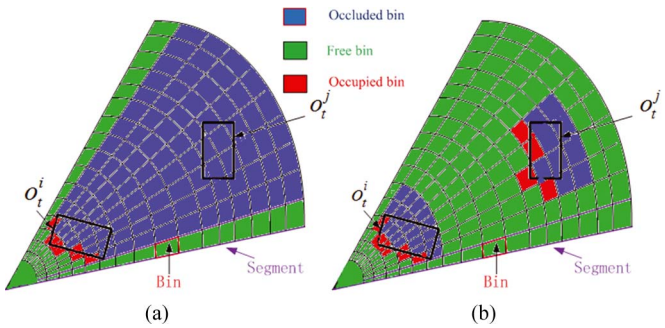



Fig. 6. Comparisons of (a) the original 2-D virtual scan in [29] and (b) our improved one. Two rectangles represent two objects. The red, blue, and green regions represent the occupied, occluded, and free bins, respectively.

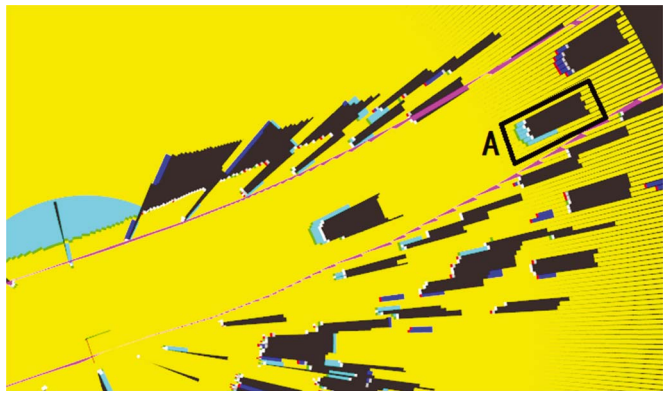



Fig. 7. Results of a scan differencing operation between two consecutive scans in our improved 2-D virtual scan. The yellow lines represent segments. The red pixels are new obstacles, and the green ones are obstacles that disappear. The cyan pixels represent the regions that were occluded in the last time step, but are free now, whereas the blue ones are the regions that were free in the last time step but are occluded now. The two pink curves represent the road kerbs that are obtained from a digital road map.

$N_s$ segments is constructed in [29]. Every segment will be divided into $N_b$ bins to discretize the range component with a resolution of 0.2 meter, as shown in Fig. 6(a). All three-dimensional points of the objects $O_t$ are projected onto the two-dimensional virtual scan. In this paper, we propose an improved two-dimensional virtual scan. Instead of recording the range to the nearest object in each segment and classifying the bins beyond the recorded range as occluded space [the blue bins in Fig. 6(a)], we take each object $o_t^i \in O_t$ as a processing unit and do not need to record the nearest range in each segment at all. At the beginning, we initialize all bins of our improved two-dimensional virtual scan as free space. For each object $o_t^i$, we calculate the nearest and furthest bins $b_i^{\min}, b_i^{\max}$ that $o_t^i$ covers. Then for each segment $j$ that is covered by $o_t^i$, we search the nearest bin $b_i^j$ that is occupied by the points of $o_t^i$. Thus each bin in segment $j$ contains the following information: Bin $b_i^j$ is occupied (the red bins), the bins between $b_i^{\min}$ and $b_i^j$ remain free (the green bins), while the bins between $b_i^j$ and $b_i^{\max}$ are occluded ones(the blue bins), as shown in Fig. 6(b). In our improved two-dimensional virtual scan [Fig. 6(b)], the bins that are occluded by other objects in the $xy$ plane can also be classified into free, occupied and occluded bins, like the ones covered by Object $o_t^j$ and occluded by Object $o_t^i$ in Fig. 6(b). However, in Fig. 6(a), these corresponding bins are all classified as the occluded bins (the blue ones) in the original virtual scan, indiscriminately.

In order to detect the dynamic objects, we need to know what changes take place over time. With two consecutive scans $S_{t-1}$ and $S_t$, these changes can be easily detected by a scan differencing operation after the ego-motion compensation provided by the GPS-INS is performed. That is because Scans $S_{t-1}, S_t$ are collected at different ego-vehicle's local coordinate frames $O_{t-1}x_{t-1}y_{t-1}z_{t-1}$ and $O_tx_ty_tz_t$. The scan differencing operation can be performed only when Scans $S_{t-1}$ and $S_t$ are in the same coordinate frame. This can be realized through the ego-motion compensation.

Assume that the global coordinates and the yaw angles of the ego-vehicle are $(x_t^e, y_t^e, z_t^e; \psi_t^e)$ and $(x_{t-1}^e, y_{t-1}^e, z_{t-1}^e; \psi_{t-1}^e)$, respectively, when collecting Scans $S_t$ and $S_{t-1}$. For each point $(x_{t-1}^i, y_{t-1}^i, z_{t-1}^i)^T$ in Scan $S_{t-1}$ that is collected in the ego-vehicle's local coordinate frame $O_{t-1}x_{t-1}y_{t-1}z_{t-1}$, we can obtain a corresponding point $(x_{t-1}^{i*}, y_{t-1}^{i*}, z_{t-1}^{i*})^T$ in the local coordinate frame $O_tx_ty_tz_t$ with (5). All these corresponding points form a new scan $S_{t-1}^*$ in the local coordinate frame $O_tx_ty_tz_t$. An orientation $\theta_{t-1}$ in the $xy$ plane of the local coordinate frame $O_{t-1}x_{t-1}y_{t-1}z_{t-1}$ corresponds to $\theta_{t-1}^*$ in $O_tx_ty_tz_t$ with (6):

$$\begin{bmatrix} tx \\ ty \\ tz \end{bmatrix} = \begin{bmatrix} \cos\psi_{t-1}^e & -\sin\psi_{t-1}^e & 0 \\ \sin\psi_{t-1}^e & \cos\psi_{t-1}^e & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1}^i \\ y_{t-1}^i \\ z_{t-1}^i \end{bmatrix} + \begin{bmatrix} x_{t-1}^e - x_t^e \\ y_{t-1}^e - y_t^e \\ z_{t-1}^e - z_t^e \end{bmatrix}$$

$$\begin{bmatrix} x_{t-1}^{i*} \\ y_{t-1}^{i*} \\ z_{t-1}^{i*} \end{bmatrix} = \begin{bmatrix} \cos\psi_t^e & \sin\psi_t^e & 0 \\ -\sin\psi_t^e & \cos\psi_t^e & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} tx \\ ty \\ tz \end{bmatrix} \tag{5}$$

$$\theta_{t-1}^* = \theta_{t-1} + \left(\psi_{t-1}^e - \psi_t^e\right). \tag{6}$$

We can construct an improved two-dimensional virtual scan with Scans $S_{t-1}^*$ and $S_t$ in the local coordinate frame $O_tx_ty_tz_t$. Each bin of our improved virtual scan contains two states. The changes are computed by comparing two states of the same bin in our improved virtual scan. For each object, if the number of its bins, whose states change in our improved virtual scan, is more than 4, then the object is a dynamic one. Fig. 7 shows the results of a scan differencing operation in our improved two-dimensional virtual scan. The red pixels are new obstacles, and the green ones represent obstacles that disappear. The cyan pixels represent the regions that are occluded at time step $t-1$, but free at $t$, while the blue ones are the regions that are free at $t-1$, but occluded at $t$. The two pink curves represent the road kerbs that are obtained from a prior digital road map. Note that the bins in Rectangle A are always occluded if constructing an original virtual scan in [29], which will make the successive detection of dynamic objects impossible in this region.

*2) Dynamic Vehicle Hypotheses Generation and Verification:* In this paper, with a backward search, only three consecutive scans $S_{t-1}$, $S_t$, and $S_{t+1}$ are required to generate a new tracker. For each dynamic object in $S_t$, its pose is estimated with the pose estimation algorithm in Section II, and the corresponding velocity can be estimated using the importance sampling technique with Scan $S_{t-1}$. For the dynamic object that meets the motion evidence criterion in [29], a dynamic vehicle hypothesis would be generated, which will be verified with the motion consistency criterion in the next scan $S_{t+1}$. The detailed process is presented as follows:

Firstly, a vehicle is fitted for each dynamic object in Scan $S_t$. We can obtain a weighted particle set $\{\chi, \omega\}$ using Algorithm 1 (PE_MSS) for each dynamic object, and each particle $(X, w) \in \{\chi, \omega\}$ represents one possible pose $X = (x, y, \theta)$ of the vehicle with a weight $w$. The particle with the maximum weight in $\{\chi, \omega\}$ is chosen to represent the initial pose of the vehicle, and the other particles in the particle set will be discarded directly.

Secondly, the velocity $v_t$ of the vehicle is estimated using the importance sampling technique in Scan $S_{t-1}$ via a backward search. In this stage, we assume that the vehicle is running from time step $t$ to $t-1$ in reverse with a velocity that is sampled from a uniform distribution $U(-35 \text{ m/s}, 35 \text{ m/s})$. The sampled negative velocity (its direction is opposite to the estimated orientation $\theta$) means that the actual moving direction of the vehicle from time step $t-1$ to time step $t$ is the same as the estimated orientation $\theta$, while the sampled positive velocity (its direction is the same as the estimated orientation $\theta$) represents that the actual moving direction of the vehicle is opposite to $\theta$. The velocity is estimated using the importance sampling technique, and all the particles are weighted using the likelihood-field-based vehicle measurement model in Section II-A. We believe that the particle with the maximum weight is the corresponding vehicle in Scan $S_{t-1}$. The actual velocity $v_t$ of the vehicle is opposite to that of the particle with the maximum weight. Once we have found the corresponding vehicle in Scan $S_{t-1}$, a dynamic vehicle hypothesis will be generated if the object in Scan $S_t$ meets the motion evidence criterion in [29].

Thirdly, for a hypothesis that meets the motion evidence criterion in [29], the importance sampling technique is also performed against the next scan $S_{t+1}$ to obtain the corresponding pose $(x_{t+1}, y_{t+1}, \theta_{t+1})$ and velocity $v_{t+1}$ at time step $t+1$. We believe that the velocities and the orientations of the same vehicle in two consecutive scans $S_t$ and $S_{t+1}$ should be similar (termed *motion consistency*). Only the hypothesis that meets the motion consistency criterion will be finally verified as a new tracker, and merged into the existing trackers.

### C. Data Association and Trackers Update

This subsection will present how to associate the objects $O_t$ in Scan $S_t$ to each tracker ${}^{t-1}T_*^i$ according to its pose $X_{t-1}$ and velocity $v_{t-1}$ in the local coordinate frame $O_{t-1}x_{t-1}y_{t-1}z_{t-1}$. Using these associated objects, the pose and the velocity of Tracker ${}^{t-1}T_*^i$ can be updated in the local coordinate frame $O_t x_t y_t z_t$ via our improved SSBF.

*1) Data Association:* Data association will use the ego-motion compensation provided by the GPS-INS and the vehicle dynamic model of Tracker ${}^{t-1}T_*^i$. Assuming that the pose and velocity of Tracker ${}^{t-1}T_*^i$ in $O_{t-1}x_{t-1}y_{t-1}z_{t-1}$ are $X_{t-1} = (x_{t-1}, y_{t-1}, \theta_{t-1})$ and $v_{t-1}$, respectively, (5) and (6) can be used to compensate for the ego-motion of the ego-vehicle, and separate the ego-motion from the independent vehicle dynamic model of Tracker ${}^{t-1}T_*^i$. The corresponding pose $X_{t-1}^* = (x_{t-1}^*, y_{t-1}^*, \theta_{t-1}^*)$ in coordinate frame $O_t x_t y_t z_t$ can be calculated with (5) and (6). After that, the prediction $\overline{X}_t = (\overline{x}_t, \overline{y}_t, \overline{\theta}_t)$ can be achieved with (7) in $O_t x_t y_t z_t$ via the vehicle dynamic model.

$$\begin{aligned} \overline{v}_t &= v_{t-1} + \varepsilon_v \\ \overline{x}_t &= x_{t-1}^* + \overline{v}_t \Delta t * \cos\left(\theta_{t-1}^* + \varepsilon_1\right) \\ \overline{y}_t &= y_{t-1}^* + \overline{v}_t \Delta t * \sin\left(\theta_{t-1}^* + \varepsilon_1\right) \\ \overline{\theta}_t &= \theta_{t-1}^* + \varepsilon_1 + \varepsilon_2 \end{aligned} \tag{7}$$

where $\varepsilon_v$, $\varepsilon_1$, and $\varepsilon_2$ are drawn from a Gaussian distribution with zero mean and a diagonal covariance matrix, randomly. $\Delta t$ is the time interval from $t-1$ to $t$. (7) combines both the ego-motion compensation and the tracker's own motion.

The prediction of Tracker ${}^{t-1}T_*^i$ in the local coordinate frame $O_t x_t y_t z_t$ is an oriented box ${}^t b_*^i$ with pose $\overline{X}_t$, width $W$ and length $L$. All objects $O_t$ in Scan $S_t$, whose oriented bounding boxes are contained by ${}^t b_*^i$, are the associated objects of Tracker ${}^{t-1}T_*^i$ in $O_t x_t y_t z_t$. All points of these associated objects form Set ${}^t Z_*^i$, which will be utilized to update the pose and the velocity of Tracker ${}^{t-1}T_*^i$ in $O_t x_t y_t z_t$ at time step $t$.

*2) Trackers Update:* In this paper, our improved SSBF is utilized to update the pose and velocity of a tracker in dynamic background scenes. The SSBF does not forget any information from the prior step, thus the results can converge to the true posterior belief quickly. The detailed algorithm is depicted in Algorithm 2.

**Algorithm 2** Vehicle Tracking Algorithm: Our Improved SSBF for Tracker ${}^{t-1}T_*^i$

**Input**: $\{\chi_{t-1}, \omega_{t-1}\}$ // the belief at time step $t-1$ ${}^t Z_*^i = \{q_1, q_2, \ldots, q_n\}$ // The associated measurements ${}^{t-1}T_*^i = (X_{t-1}^o, v_{t-1}^o)$, $M$, $\Delta$, $N$, $L$, $\alpha$, $\beta$

1: $\{\chi_{t-1}^*, \omega_{t-1}\} = \text{Ego\_Motion}(\{\chi_{t-1}, \omega_{t-1}\})$;
2: $X_{t-1}^{o*} = \text{Ego\_Motion}(X_{t-1}^o)$;
3: $\{\chi_t, \omega_t\} = \text{PE\_MSS}({}^t Z_*^i, M, \Delta, N, L)$;
4: **for** each $\{X_t, w_t\} \in \{\chi_t, \omega_t\}$ **do**
5: $\quad s = \sum_{\{X_{t-1}^*, w_{t-1}\} \in \{\chi_{t-1}^*, \omega_{t-1}\}} p(X_t | X_{t-1}^*) w_{t-1}$;
6: $\quad w_t = w_t s$;
7: **end for**
8: normalize weights $\omega_t$;
9: $X_t^o = \text{Best\_Particle}(\chi_t, \omega_t)$;
10: $v_t^o = \text{Best\_Velocity}(X_t^o, X_{t-1}^{o*})$;

**Output**: $\{\chi_t, \omega_t\}$, //approximated posterior belief $\text{bel}_t$ at $t$ ${}^t T_*^i = (X_t^o, v_t^o)$; //tracking result

Algorithm 2 takes as input the belief $\{\chi_{t-1}, \omega_{t-1}\}$ that is represented by a weighted particle set, the output pose $X^o_{t-1}$ and velocity $v^o_{t-1}$ in $O_{t-1}x_{t-1}y_{t-1}z_{t-1}$ at time step $t-1$, and the associated measurements ${}^tZ^i_*$ in $O_t x_t y_t z_t$. The other parameters are some constants used in Algorithms 1 and 3. It outputs the approximated posterior belief $\{\chi_t, \omega_t\}$ and the final tracking result ${}^tT^i_* = (X^o_t, v^o_t)$ in $O_t x_t y_t z_t$ at time step $t$, where ${}^tT^i_*$ is the output of this cycle and $\{\chi_t, \omega_t\}$ will be used as the input in the next cycle.

Algorithm 2 contains five steps. The first step is to compensate for the ego-motion of the ego-vehicle from time step $t-1$ to $t$. This corresponds to function *Ego_Motion* in Lines 1 and 2. For every particle $\{X_{t-1}, w_{t-1}\} \in \{\chi_{t-1}, \omega_{t-1}\}$ in $O_{t-1}x_{t-1}y_{t-1}z_{t-1}$, we can calculate a corresponding particle $\{X^*_{t-1}, w_{t-1}\} \in \{\chi^*_{t-1}, \omega_{t-1}\}$ in $O_t x_t y_t z_t$ using (5) and (6) with the information provided by the GPS-aid INS. In the same way, the output pose $X^o_{t-1}$ of the last circle in the local coordinate frame $O_{t-1}x_{t-1}y_{t-1}z_{t-1}$ can also obtain the corresponding pose $X^{o*}_{t-1}$ in $O_t x_t y_t z_t$.

The second step serves to obtain a weighted particle set $\{\chi_t, \omega_t\}$ with a uniform prior from the associated measurements ${}^tZ^i_*$ using the pose estimation algorithm in Section II. This corresponds to function *PE_MSS* in Line 3, which is implemented in Algorithm 1.

The lines from 4 to 7 describe the process of adjusting the weight $w_t$ of the particle $X_t$ in $\{\chi_t, \omega_t\}$ one by one via the Bayesian recursion equation (8) [30] to capture the vehicle dynamic model $p(X_t|X^*_{t-1})$ of Tracker ${}^{t-1}T^i_*$.

$$\text{bel}_t = \eta p(Z_t|X_t) \int p\left(X_t|X^*_{t-1}\right) \text{bel}_{t-1} dX^*_{t-1}. \tag{8}$$

As the exact dynamics of Tracker ${}^{t-1}T^i_*$ is unknown in advance, linear motion [30] is utilized to describe the dynamic model of a tracker in this paper. The motion is approximated by a perturbing orientation $\Delta\theta_1$, followed by a translation $\Delta s$ and a final perturbing orientation $\Delta\theta_2$. We assume that they are all independent. From time step $t-1$ to $t$, a tracker will advance from $X^*_{t-1} = \{x^*_{t-1}, y^*_{t-1}, \theta^*_{t-1}\}$ to $X_t = \{x_t, y_t, \theta_t\}$ with velocity $v^o_{t-1}$ according to $p(X_t|X^*_{t-1}, v^o_{t-1})$ in $O_t x_t y_t z_t$. Algorithm 3 depicts the process to compute $p(X_t|X^*_{t-1})$ in detail.

**Algorithm 3** Compute $p(X_t|X^*_{t-1})$ from time step $t-1$ to $t$

**Input**: $X^*_{t-1} = \{x^*_{t-1}, y^*_{t-1}, \theta^*_{t-1}\}$, $X_t = \{x_t, y_t, \theta_t\}$, $v^o_{t-1}$, $\alpha, \beta$

1: $\theta_{E1} = 0$, $s_E = v^o_{t-1}\Delta t$, $\theta_{E2} = 0$;

2: $\theta_{A1} = a\tan 2(y_t - y^*_{t-1}, x_t - x^*_{t-1}) - \theta^*_{t-1}$;

3: $s_A = \sqrt{(x_t - x^*_{t-1})^2 + (y_t - y^*_{t-1})^2}$;

4: $\theta_{A2} = \theta_t - a\tan 2(y_t - y^*_{t-1}, x_t - x^*_{t-1})$;

5: $\Delta\theta_1 = \theta_{A1} - \theta_{E1}$, $\Delta s = s_A - s_E$, $\Delta\theta_2 = \theta_{A2} - \theta_{E2}$;

6: $\Delta\theta_1 \sim N(0, \alpha s_A)$, $\Delta s \sim N(0, \beta s_A)$, $\Delta\theta_2 \sim N(0, \alpha s_A)$;

7: $p(X_t|X^*_{t-1}, v^*_{t-1}) = p(\Delta\theta_1)p(\Delta s)p(\Delta\theta_2)$;

**Output**: $p(X_t|X^*_{t-1})$

Fig. 8. Our ALV used for the experiments in various environments. It is equipped with a Velodyne HDL-64E S2 LIDAR, a NovAtel SPAN_CPT GPS-aid INS, and some cameras for perception.

Algorithm 3 takes as input: the pose $X^*_{t-1}$, the velocity $v^o_{t-1}$, the pose $X_t$ (obtained by Algorithm 1) in $O_t x_t y_t z_t$, and some constants $\alpha, \beta$. It outputs the probability $p(X_t|X^*_{t-1})$, which is obtained by multiplying the individual probabilities $p(\Delta\theta_1), p(\Delta s)$, and $p(\Delta\theta_2)$. In this algorithm, $N(0, \alpha s_A)$ represents a Gaussian distribution with mean zero, variance $\alpha s_A$. $\Delta t$ is the time interval from time step $t-1$ to $t$.

The fourth step of Algorithm 2 is to normalize the weights $\omega_t$ in Line 8. The weighted particle set $\{\chi_t, \omega_t\}$ approximates to the posterior belief $\text{bel}_t$ of Tracker ${}^tT^i_*$ at time step $t$. Lines 9 and 10 correspond to functions *Best_Particle* and *Best_Velocity*, respectively, which are to find the particle $X^o_t$ with the maximum weight from $\{\chi_t, \omega_t\}$ and the corresponding velocity $v^o_t$ as the outputs of this circle. They form the final step of Algorithm 2.

## IV. Experimental Evaluation and Analysis

The performance of our dynamic vehicle detection and tracking algorithm proposed in this paper has been evaluated on the KITTI datasets [35] and the Velodyne data collected by our ALV in various urban scenes. Two state-of-the-art methods proposed by Petrovskaya *et al.* [21], [29] are adopted to be compared with our method. We implement all the algorithms in C++ as well as the point cloud library (PCL) [36], and conduct several quantitative and qualitative comparison experiments. Our ALV is a modified Toyota Land Cruiser, as shown in Fig. 8, equipped with one Velodyne HDL-64E S2 LIDAR, one NovAtel SPAN_CPT GPS-aid INS and some other sensors, such as cameras, laser range finders, etc. The Velodyne LIDAR rotates with a speed of 10 HZ. Thus, every scan contains up to 0.13 million three-dimensional points. In all the experiments, we use a notebook computer with an Intel Core i5 CPU with 2.5 GHZ dominant frequency and 12 GB main memory.

In this paper, the Gaussian-Process-based ground segmentation algorithm in [37] is utilized to remove the ground points and the residual obstacle points are clustered into different objects with the Radially Bounded Nearest Neighbour (RBNN) graph in [38]. In the implementation of our Algorithm 2, the parameters are empirically set to $M = 16$, $\Delta = 0.4$, $N = 9$,

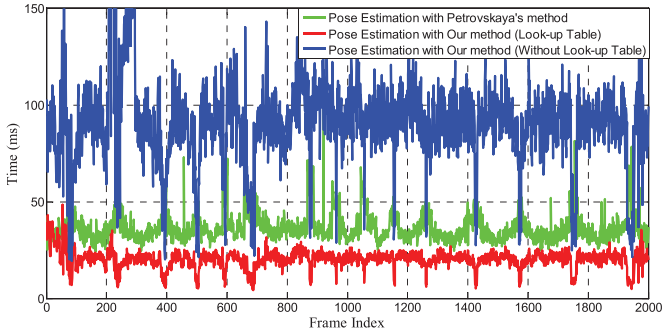



Fig. 9. Runtime of our pose estimation algorithm (red and blue curves) compared with that of Petrovskaya's algorithm [29] (green curve) for the 2000 scans acquired in an open square.

$L = 4.8$, $W = 1.8$, $\alpha = 0.1$, $\beta = 0.5$. The original virtual scan in Petrovskaya's algorithm [21], [29] and the improved one in our algorithm are both divided into $N_s = 720$ segments. Each segment is then divided into $N_b = 250$ bins to discretize the range component with a resolution of 0.2 meter.

### A. *Quantitative Experiments on Pose Estimation and Velocity Estimation*

In order to evaluate the precision of pose estimation and velocity estimation quantitatively, we conducted an experiment in an open square where there was only one parked vehicle as the target-vehicle. Our ALV ran around the parked vehicle clockwise and counter-clockwise with different radii. Thus the viewpoint and occlusion changed in each scan. The experiment lasts for about 3.5 minutes and contains 2000 scans. We labelled the pose $(x, y, \theta)$ of the parked vehicle with fixed $L = 4.8$ and $W = 1.8$ in every scan manually.

*1) Runtime:* As the algorithm is applied to our ALV, the real-time performance of the pose estimation algorithm, which is the core of our dynamic vehicle detection and tracking algorithm, is our focus. Both our pose estimation algorithm (Algorithm 1) and Petrovskaya's algorithm [29] are implemented offline, using the 2000 scans. Fig. 9 illustrates the results of our algorithm (the red and blue curves) compared with that of Petrovskaya's algorithm [29] (the green curve).

The blue curve depicts the runtime of our pose estimation algorithm without a look-up table for function erf $\left(d/\sqrt{2}\sigma\right)$. Most of the time, with an average of 90.64 milliseconds (ms) per scan, our algorithm cannot be performed in real time at all. However, after we construct a look-up table offline to approximate to function erf $\left(d/\sqrt{2}\sigma\right)$, the computational efficiency can be improved greatly, like the red curve. The runtime per scan (20.24 ms) is only about two ninths of that of the same algorithm without the look-up table. The runtime of Petrovskaya's algorithm [29] is represented by the green curve. With an average of 36.75 ms per scan, the computational efficiency is about half of that of our algorithm with a look-up table. Thus, our pose estimation algorithm with a look-up table can estimate 3 or 4 vehicles' poses per scan in real time. Meanwhile, we believe that our algorithm can simultaneously handle more vehicles with parallel computing technology.

From Fig. 9, one can find an interesting issue, that the green and red curves change periodically with opposite trends. The opposite trends occur periodically when our ALV is in front of the parked vehicle, and their orientations are perpendicular. Under these circumstances, the number of the laser points hitting the parked vehicle is the least, which increases the uncertainty of the pose of the parked vehicle. This corresponds to function *Rejection_Sampling* to generate more valid particles in Algorithm 1. In order to evaluate these particles, the weight $p(Z|X)$ of each particle needs to be calculated. In Petrovskaya's algorithm [29], the runtime is proportional to the number of the valid particles, because the numbers of rays hitting the same vehicle with different poses change little. Thus, the time spent in calculating the weight of each particle is almost the same. However, in our algorithm, the fewer the points hitting the parked vehicle, the shorter the time will be spent in computing the weight for each particle ($n$ in (2) becomes smaller). Although the number of valid particles increases too, in our algorithm, the total runtime of our pose estimation algorithm becomes smaller. Thus, when our ALV runs around the parked vehicle, the time curves of our pose estimation algorithm and Petrovskaya's algorithm [29] change inversely with regard to the number of points hitting the parked vehicle.

*2) Precision of Pose Estimation:* The dataset with labelled vehicles can also be utilized to evaluate the precision of the pose estimation algorithm. Petrovskaya's algorithms [21], [29] are also adopted to be compared with our algorithms (Algorithms 1 and 2). We conduct two comparison experiments: one is to estimate the parked vehicle's pose only with the pose estimation algorithm (Algorithm 1 and Petrovskaya's algorithm in [29]); the other one is with the vehicle tracking algorithm (Algorithm 2 and Petrovskaya's algorithm in [21]). The main difference between the two algorithms is whether they utilize the vehicle dynamic model to obtain the prior pose information about the parked vehicle, between two consecutive scans. Fig. 10 and Table I illustrate the experimental results of our algorithms and Petrovskaya's algorithms [21], [29]. In Fig. 10, the top left and bottom left images (i.e. the red bars) show the error distributions of our pose estimation algorithm (Algorithm 1) and our vehicle tracking algorithm (Algorithm 2) in the aspects of position and orientation, respectively. Meanwhile, the top right and bottom right ones are the corresponding error distributions of Petrovskaya's algorithms [21], [29]. All these distributions are nearly Gaussian. From the left two images of Fig. 10, one can find that in the aspects of position and orientation, both error distributions of our vehicle tracking algorithm become thinner than those of our pose estimation algorithm. That is because our vehicle tracking algorithm utilizes the vehicle dynamic model of the parked vehicle between two consecutive scans to obtain the prior pose information. This is approximately equivalent to bringing in a mean filter to the results of our pose estimation algorithm. The same phenomenon happens to Petrovskaya's vehicle tracking algorithm [21] (the bottom right image). The top two images show the error distributions of our pose estimation algorithm (the red bars) compared with Petrovskaya's pose estimation algorithm [29] (the green bars). One can easily find that the mean error of our pose estimation algorithm in the aspect of

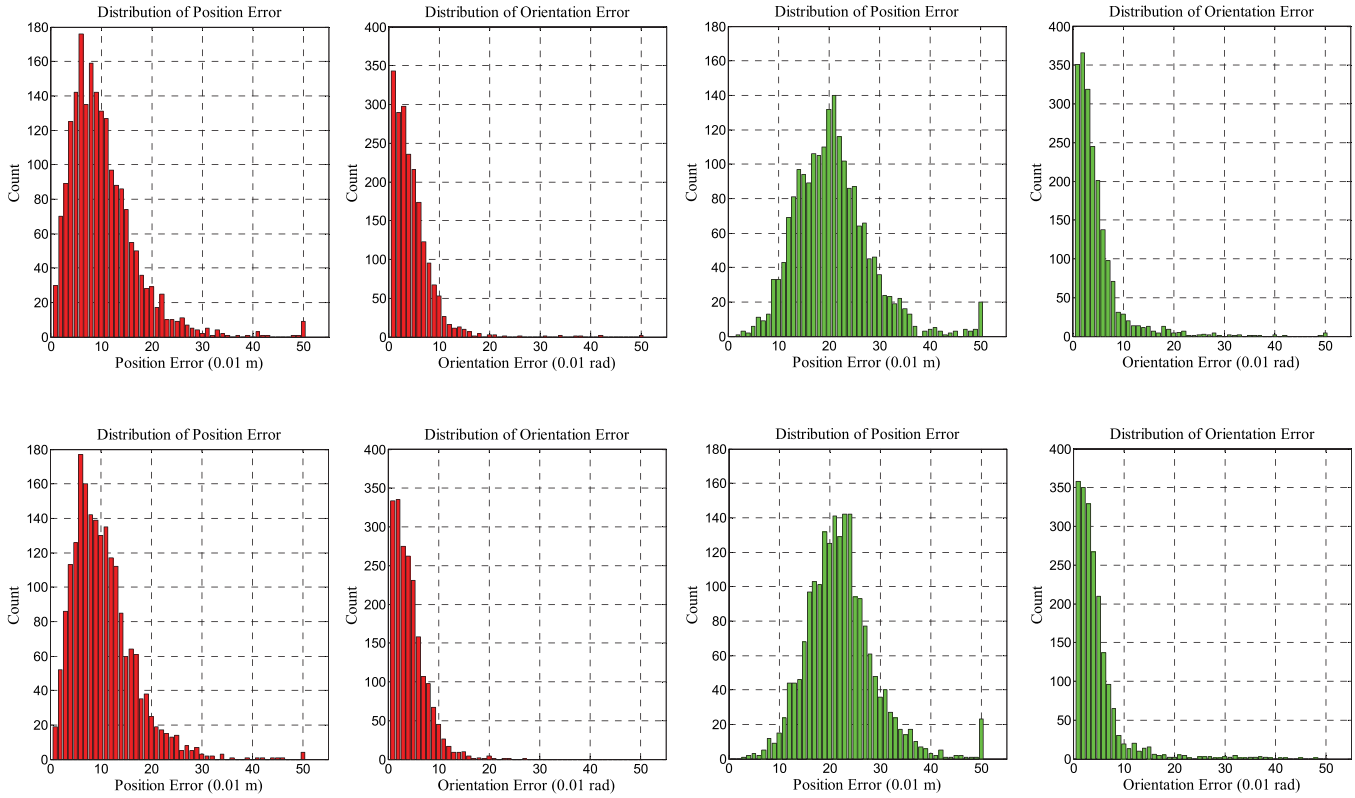



Fig. 10. Error distributions of different algorithms in the aspects of position and orientation. Left: The results of (top) our vehicle pose estimation algorithm and (bottom) our vehicle tracking algorithm. Right: The results of (top) Petrovskaya's vehicle pose estimation method [29] and (bottom) Petrovskaya's vehicle tracking method [21].

TABLE I
THE STATISTICAL ERRORS OF OUR ALGORITHMS AND PETROVSKAYA'S ALGORITHMS [21], [29]

| | | Our Pose Estimation Algorithm Without Noise Annealing | Our Pose Estimation Algorithm | Petrovskaya's Pose Estimation Algorithm [29] | Our Tracking Algorithm | Petrovskaya's Tracking Algorithm [21] |
|---|---|---|---|---|---|---|
| position error | mean (m) | 0.1034 | 0.1000 | 0.2054 | 0.1000 | 0.2195 |
| | var | 0.0099 | 0.0071 | 0.0104 | 0.0041 | 0.0066 |
| orientation error | mean (rad) | 0.0446 | 0.0420 | 0.0435 | 0.0386 | 0.0412 |
| | var | 0.0034 | 0.0027 | 0.0045 | 0.0010 | 0.0026 |
| velocity error | mean (m/s) | - | - | - | 0.6953 | 0.6872 |
| | var | - | - | - | 0.1017 | 0.0819 |

position is about half of that of Petrovskaya's algorithm [29]. That is because the measurements of our measurement model and Petrovskaya's model [29] are three-dimensional points and lengths of the two-dimensional rays hitting the target-vehicle, respectively, when we calculate the measurement likelihood $p(Z|X)$. For every target-vehicle, especially the one far away from the ego-vehicle, the number of the three-dimensional points is much more than that of the two-dimensional rays, thus our likelihood-field-based vehicle measurement model is much more stable. Additionally, calculating the lengths of the two-dimensional rays can also bring in some errors.

Table I presents the statistical errors in the aspects of position and orientation. From Table I, one can find that the pose obtained using our tracking algorithm is the most accurate with the minimal mean errors (0.1000 metre in position and 0.0386 rad in orientation). The worst is Petrovskaya's pose estimation algorithm [29] with mean errors in the aspects of position and orientation of 0.2054 metre and 0.0435 rad, also with the biggest variances 0.0104 and 0.0045, respectively. In addition, both in our algorithms (Algorithms 1 and 2) and Petrovskaya's algorithms [21], [29], the performance of the vehicle tracking algorithm is more accurate than that of the pose estimation algorithm.

In order to demonstrate the advantage of bringing in an annealing process for the measurement noise in Algorithm 1, we also conduct a comparison experiment to estimate the poses of the parked vehicles without the annealing process for the measurement noise. In Algorithm 1, we fix the variance $\sigma^2$ of the measurement noise as $0.1^2$, and the other parameters are the same. Table I manifests the experimental results (Column 3). In the aspects of position and orientation, the mean errors of our pose estimation algorithm without the measurement noise annealing are 0.1034 metre and 0.0446 rad, respectively. Although their means change little, both variances become bigger. Results indicate that the normal pose estimation algorithm with both the measurement model annealing and the measurement noise annealing (Column 4) is more stable than the one without the measurement noise annealing (Column 3).

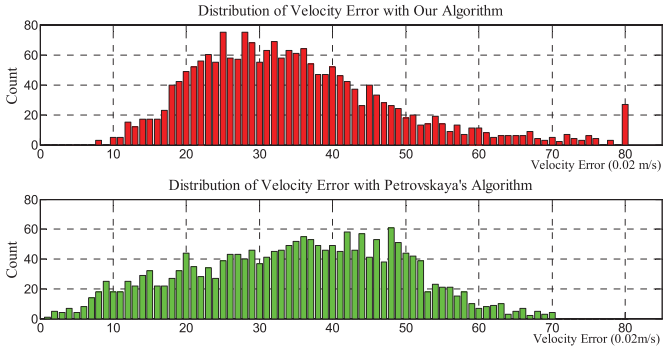



Fig. 11. Velocity error distributions of our vehicle tracking algorithm and Petrovskaya's algorithms [21].

*3) Precision of Velocity Estimation:* Although the target-vehicle is static in the global coordinate frame, it appears to be moving in the ego-vehicle's local coordinate frame with a reverse direction to our ALV. As we have recorded the velocities of our ALV from a GPS-INS in advance, we could calculate statistics for the relative velocity errors of the parked vehicle in the ego-vehicle's local coordinate frame, and evaluate the precision of velocity estimation with our vehicle tracking algorithm (Algorithm 2), quantitatively.

Fig. 11 illustrates the relative velocity error distributions of our vehicle tracking algorithm (top) compared with Petrovskaya's algorithm [21] (bottom). From this figure, one can find that the performance of velocity estimation with our vehicle tracking algorithm is a little poorer than that of Petrovskaya's algorithm [21]. Table I presents the mean and variance of the relative velocity error. The mean and variance of our tracking algorithm are 0.6953 m/s and 0.1017, respectively, while Petrovskaya's mean and variance are 0.6872 m/s and 0.0819. The main reason is that in our vehicle tracking algorithm (Algorithm 2), we do not sample the velocity of the target-vehicle directly. After the output pose $X_t^o$ is obtained, the velocity $v_t^o$ is estimated by comparing the relative positions between $X_t^o$ and $X_{t-1}^{o*}$. This results in some errors with our vehicle tracking algorithm. However, Petrovskaya's algorithm [21] utilizes a Rao-Blackwellized particle filter to sample the velocity directly. Thus, the velocity estimation of Petrovskaya's vehicle tracking algorithm [21] is a little more accurate than that of our vehicle tracking algorithm.

### *B. Quantitative Experiments on Dynamic Vehicle Detection*

In this section, we will conduct a quantitative experiment on the public KITTI datasets [35] (*2011_09_26_drive_0056* and *2011_09_26_drive_0057*) to evaluate the performance of our dynamic vehicle detection algorithm. Petrovskaya's dynamic vehicle detection algorithm [29] is adopted to be compared with. The cars, vans, trucks, pedestrians and cyclists have already been labelled in the form of cuboids in the KITTI datasets. Since we only focus on dynamic vehicles in this paper, these labelled cars and vans are gathered into the vehicle class. We also manually remove the static vehicles and the ones that are 50 meters away from the ego-vehicle. Both Petrovskaya's algorithm and our algorithm ignore the heights of the vehicles. Thus, in order to evaluate the performance, both the labelled cuboids and the detected ones are projected into the $xy$ plane to become the corresponding rectangles $R_l, R_d$. Detections are judged to be true or false positives by measuring the rectangle overlap. To be considered a correct detection, the area of the overlap $(\text{area}(R_d \bigcap R_l))$ between the detected rectangle $R_d$ and the labelled one $R_l$ must exceed 50% of the area of their union $(\text{area}(R_d \bigcup R_l))$.

The dataset *2011_09_26_drive_0056* is collected while the ego-vehicle is running on a straight urban road with some houses, trees and fences on the roadside. The other dataset *2011_09_26_drive_0057* is collected while the ego-vehicle is stopping behind a static vehicle at an intersection. Fig. 12 shows two scenes chosen from the two datasets. For each scene, the bottom left image presents the detection result (the green cuboids) of our algorithm while the bottom right one shows the result (the green cuboids) of Petrovskaya's algorithm [29]. These white cuboids represent the labelled vehicles. One can find that without occlusions, both Petrovskaya's algorithm and our algorithm can obtain good results, as shown in the left scene of Fig. 12. However, the performance of Petrovskaya's algorithm [29] degrades in the scenes where dynamic vehicles are occluded by other objects in the $xy$ plane, like the right scene in Fig. 12.

Table II presents the statistical results of our dynamic vehicle detection algorithm compared with Petrovskaya's algorithm [29] on the two KITTI datasets. The dataset *2011_09_26_drive_0056* contains 294 scans, and there are 404 dynamic vehicles. One can find that both Petrovskaya's algorithm and our algorithm can obtain promising results although only three consecutive scans are used to detect the dynamic vehicles. That is because there are few occlusions in this dataset. Our accuracy is 85.4% while Petrovskaya's is 80.0%. However, there are also 220 false positives being detected with Petrovskaya's algorithm [29]. That is because there are many houses and fences on the roadside, and the projections of their corners and sides in the $xy$ plane are almost the same as the visible sides of vehicles. Additionally, the number of the two-dimensional rays hitting a vehicle is always less than that of the three-dimensional points. This makes our algorithm (using three-dimensional points as measurements) more robust than Petrovskaya's algorithm (using two-dimensional rays as measurements). The dataset *2011_09_26_drive_0057* contains 361 scans, and there are 668 dynamic vehicles. However, the ego-vehicle stops behind a static vehicle. Most of these dynamic vehicles that are passing through the intersection are occluded by the static vehicle in the $xy$ plane. Thus, the performance of Petrovskaya's algorithm [29] and our algorithm degrades at the same time, even though our algorithm can still detect more dynamic vehicles (431) than Petrovskaya's algorithm [29] (319). That is because these occluded dynamic vehicles have no two-dimensional rays hitting them in the $xy$ plane, although they can still be detected by the Velodyne LIDAR. The detected three-dimensional points can be used as the measurements of the likelihood-field-based vehicle measurement model in our dynamic vehicle detection algorithm. However, Petrovskaya's dynamic vehicle detection algorithm [29] is based on the beam-based vehicle measurement model that utilizes the lengths of the two-dimensional rays as its measurements. The beam-based measurement model has no measurements at all. Thus it fails

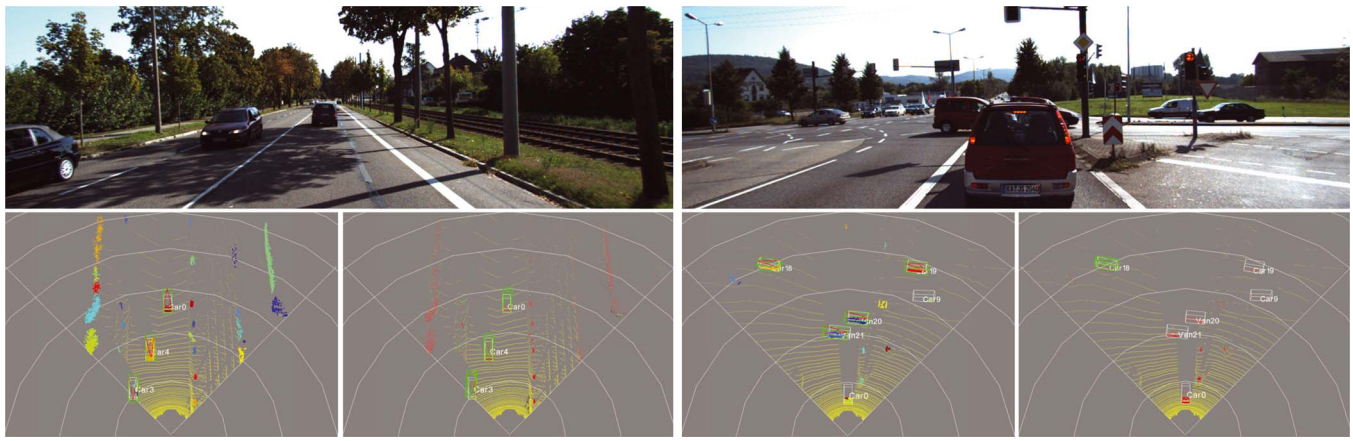


Fig. 12. The dynamic vehicle detection results on two scenes. The two scenes are chosen from the two KITTI data sets, respectively. These green cuboids are detected vehicles, and the white ones are ground truth. For every scene, the bottom left image is the result of our algorithm, whereas the bottom right one is the result of Petrovskaya's algorithm [29].

TABLE II
THE STATISTICAL RESULTS OF PETROVSKAYA'S DYNAMIC VEHICLE DETECTION ALGORITHM [29] AND OUR ALGORITHM

| Datasets | Total Vehicles | Our Algorithm | | Petrovskaya's Algorithm [29] | |
|---|---|---|---|---|---|
| | | True Positive (%) | False Positive (%) | True Positive (%) | False Positive (%) |
| 2011_09_26_*drive*_0056 | 404 | 345 (85.4%) | 48 (11.9%) | 323 (80.0%) | 220 (54.5%) |
| 2011_09_26_*drive*_0057 | 668 | 431 (65.0%) | 48 (7.2%) | 319 (47.8%) | 79 (11.8%) |
| Overall | 1072 | 776 (72.4%) | 96 (9.0%) | 642 (59.9%) | 299 (27.9%) |

to detect the dynamic vehicles that are occluded by the static vehicle in the $xy$ plane. As the ego-vehicle stops, the background changes little in the dataset *2011_09_26_drive_0057*. Thus, the false positive rates of Petrovskaya's algorithm and our algorithm decrease too. They are 11.8% and 7.2%, respectively. From Table II, one can easily find that our dynamic vehicle detection algorithm outperforms Petrovskaya's algorithm [29].

### C. Qualitative Experiments

In order to evaluate the usefulness of the proposed dynamic vehicle detection and tracking algorithm to our ALV intuitively, we conduct a qualitative experiment to detect and track dynamic vehicles in urban environments. The urban scenes include two T-junctions, one intersection and a curved road. There are also five vehicles in the scenes, and one of them is static on the roadside. Fig. 13 illustrates the dynamic vehicle detection and tracking results obtained with our algorithm in urban environments. This experiment lasts for about a half minute and collects 240 successive scans. The result is shown once every 10 scans. In Fig. 13, the yellow points represent ground, and the other colours are different objects obtained with the RBNN graph in [38]. The cuboids with different colours (buff, red, cyan, and deep yellow) represent the tracked vehicles with different poses. Each white circle represents 10 meters away from our ALV.

In the first 5 images of Fig. 13, only two dynamic vehicles (the buff and red cuboids) are detected and tracked. One (the buff cuboid) is in front of our ALV, and the other (the red cuboid) is behind it. The parked vehicle on the roadside (labelled with a black circle in Image 1) is not detected because it is not moving. Because of the ego-motion compensation, our dynamic vehicle detection algorithm will not identify the parked vehicles as dynamic vehicles. After our ALV has passed through the first T-junction, the third dynamic vehicle (the cyan cuboid) that is entering the first T-junction is detected in Image 6. This vehicle is going to turn right and follow our ALV in the following 10 images. The process of merging into the traffic flow is presented exactly in these images, because of the accurate pose estimation for the cyan vehicle with our vehicle tracking algorithm. Our ALV is moving on as shown in Image 9. When it is approaching the second T-junction in Image 9, the fourth dynamic vehicle (the deep yellow cuboid) is about to enter the junction too. Our ALV is capable of discovering the deep yellow vehicle in time to keep safe. Also, the fourth dynamic vehicle will turn right to merge into the traffic flow and follow our ALV, and then will turn left in the next intersection. The whole process is shown from Image 10 to Image 24. One can see that our dynamic vehicle tracking algorithm is able to estimate the poses of the vehicles accurately in all scans, although with different viewpoints and occlusions. After the buff vehicle passes through the intersection, it will enter a curved road from Image 17. Our vehicle tracking algorithm is also able to estimate its poses accurately during its turn on the curved road.

Compared with Petrovskaya's vehicle tracking algorithm [21], one advantage of our algorithm is that it is able to detect and track the dynamic vehicles that are occluded by other objects in the $xy$ plane, like the cyan vehicle that is labelled with three red circles in Images 14, 15, and 16 in Fig. 13. Although they are fully occluded by the red and deep yellow vehicles in the $xy$ plane, our vehicle tracking algorithm can still track them accurately. However, Petrovskaya's algorithm [21] will fail. Fig. 14 shows the details of why Petrovskaya's algorithm [21] fails. It is a crowded urban environment at an intersection with twelve dynamic vehicles. Four of them (in the red, green, blue and purple rectangles of Fig. 14) are fully occluded by other

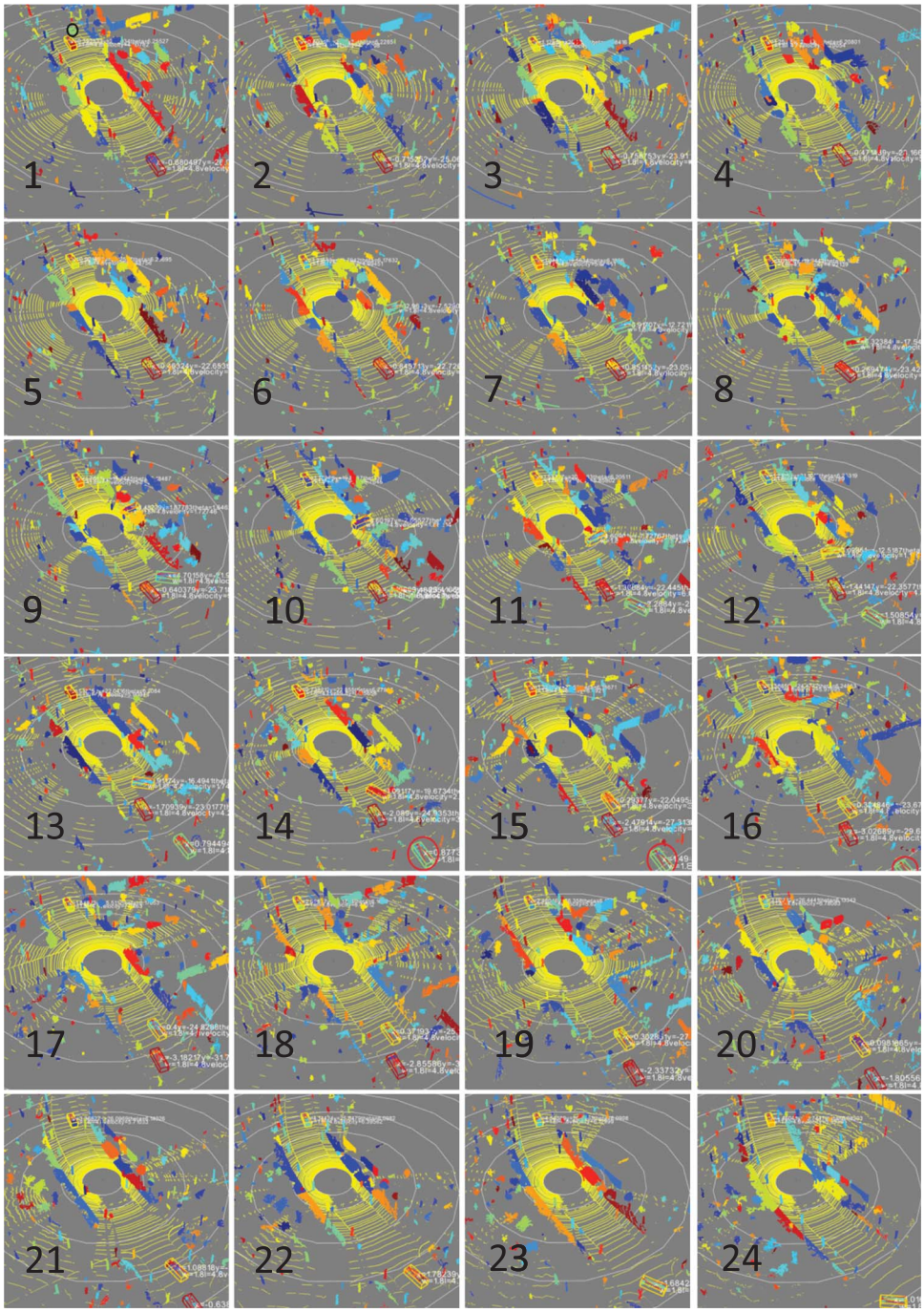



Fig. 13. Some dynamic vehicle detection and tracking results in urban environments with our algorithm. The scan sequence contains 240 scans, and the result is shown once every 10 scans. The yellow points represent the ground. Each white circle represents 10 m away from our ALV. These cuboids with different colors are different tracked vehicles.

vehicles in the $xy$ plane. Although there are no two-dimensional rays (yellow lines) hitting the four vehicles in the original virtual scan, they also have some three-dimensional points (red points) as their measurements, as shown in the righthand image of Fig. 14. With Petrovskaya's vehicle measurement model that depends on the lengths of these two-dimensional rays, the weights of the particles, that represent the poses of the four vehicles in Rao-Blackwellized particle filters, cannot be calculated at all, because no two-dimensional rays hit them. In contrast, our vehicle measurement model takes these three-dimensional points as their measurements directly. Thus, the weights of the particles in our improved SSBF can still be calculated to evaluate these particles. That is the reason why our tracking algorithm can track the occluded vehicles, but Petrovskaya's algorithm [21] fails. This ability will be crucial for an ALV to detect and track the dynamic vehicles that are about to enter an intersection earlier, where there are always some low bushes on the roadside. This also confirms the superiority of our dynamic vehicle detection and tracking algorithm presented in this paper.

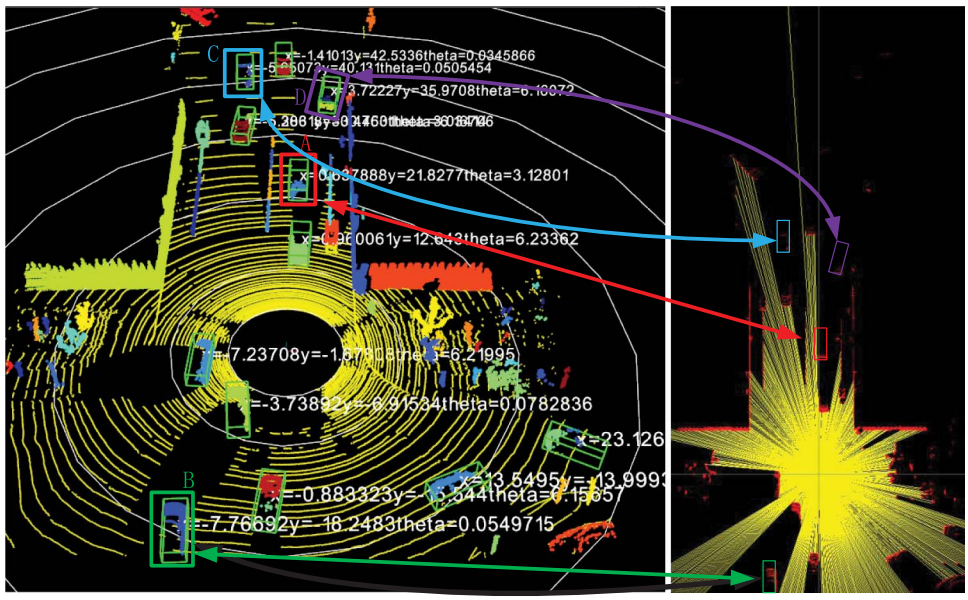



Fig. 14. Another crowded urban environment at an intersection. Our tracking algorithm can estimate the poses of the vehicles that are fully occluded in the $xy$ plane by other objects (four vehicles in the red, green, blue, and purple rectangles), compared with Petrovskaya's algorithm [21].

Our algorithm had been successfully used in our ALV in the 2015 Chinese Future Challenge, where the ALV needed to finish the tasks such as pedestrian detection, sign recognition, dynamic vehicle detection and tracking, parallel parking and so on. With the algorithm in this paper, our ALV accomplished the dynamic vehicle detection and tracking task perfectly and won the third place finally.

## V. Conclusion and Future Work

This paper presents a novel real-time dynamic vehicle detection and tracking algorithm for our ALV. A novel likelihood-field-based vehicle measurement model, combined with our newly modified Scaling Series algorithm, is proposed to estimate the poses of the vehicles. It can naturally handle the situation where the dynamic vehicles are fully occluded by other objects in the $xy$ plane but can still be detected by our Velodyne LIDAR. Moreover, in order to detect these dynamic vehicles occluded by other objects, an improved two-dimensional virtual scan is proposed. Finally, these detected dynamic vehicles will be tracked using our improved Scaling Series algorithm coupled with a Bayesian Filter (SSBF) in dynamic background urban environments. Both quantitative and qualitative experiments on the KITTI datasets and the Velodyne data collected by our own ALV validate the performance of our dynamic vehicle detection and tracking algorithm.

In this paper, the width and length of our likelihood-field-based vehicle measurement model are fixed. Thus, different values need to be chosen manually, in advance, for vehicles with different sizes. In the future, we will investigate how to choose the width and length of our vehicle measurement model automatically according to the measurements of the vehicles detected. This will make our dynamic vehicle detection and tracking algorithm suitable for all kinds of vehicles and will improve its robustness and accuracy too.

## Acknowledgment

The authors would like to thank all participants in the laboratory, who had joined the 2015 Chinese Future Challenge, for their useful discussions and suggestions.

## References

[1] C. C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *Int. J. Robot. Res.*, vol. 26, no. 9, pp. 889–916, Sep. 2007.

[2] C. Coue, C. Pradalier, C. Laugier, T. Fraichard, and P. Bessiere, "Bayesian occupancy filter for multitarget tracking: An automotive application," *Int. J. Robot. Res.*, vol. 25, no. 1, pp. 19–30, Jan. 2006.

[3] T. Gindele, S. Brechtel, J. Schroder, and R. Dillmann, "Bayesian occupancy grid filter for dynamic environments using prior map knowledge," in *Proc. IEEE IV Symp.*, 2009, pp. 669–676.

[4] A. Negre, L. Rummelhard, and C. Laugier, "Hybrid sampling Bayesian occupancy filter," in *Proc. IEEE IV Symp.*, 2014, pp. 1307–1312.

[5] R. Danescu, F. Oniga, and S. Nedevschi, "Modeling and tracking the driving environment with a particle based occupancy grid," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1331–1342, Dec. 2011.

[6] J. Moras, V. Cherfaoui, and P. Bonnifait, "Moving objects detection by conflict analysis in evidential grid," in *Proc. IEEE IV Symp.*, 2011, pp. 1122–1127.

[7] M. Schutz, N. Appenrodt, J. Dickmann, and K. Dietmayer, "Occupancy grid map-based extended object tracking," in *Proc. IEEE IV Symp.*, 2014, pp. 1205–1210.

[8] R. Jungnickel and F. Korf, "Object tracking and dynamic estimation on evidential grids," in *Proc. 17th ITSC*, 2014, pp. 2310–2316.

[9] G. Tanzmeister, J. Thomas, D. Wollherr, and M. Buss, "Grid-based mapping and tracking in dynamic environments using a uniform evidential environment representation," in *Proc. IEEE ICRA*, 2014, pp. 6090–6095.

[10] J. Leonard *et al.*, "A perception-driven autonomous urban vehicle," *J. Field Robot.*, vol. 25, no. 10, pp. 727–774, Oct. 2008.

[11] A. Azim and O. Aycard, "Detection,classification and tracking of moving objects in a 3D environments," in *Proc. IEEE IV Symp.*, 2012, pp. 802–807.

[12] R. Kaestner, J. Maye, Y. Pilat, and R. Siegwart, "Generative object detection and tracking in 3D range data," in *Proc. IEEE ICRA*, 2012, pp. 3075–3081.

[13] Y. Yang, G. Yan, H. Zhu, M. Y. Fu, and M. L. Wang, "Moving object detection under dynamic background in 3D range data," in *Proc. IEEE IV Symp.*, 2014, pp. 394–399.

[14] M. Himmelsbach and H. J. Wuensche, "Tracking and classification of arbitrary objects with bottom-up/top-down detection," in *Proc. IEEE Intell. Veh. Symp.*, 2012, pp. 577–582.

[15] J. Choi, S. Ulbrich, B. Lichte, and M. Maurer, "Multi-tracking using a 3D-lidar sensor for autonomous vehicles," in *Proc. IEEE 16th ITSC*, 2013, pp. 881–886.

[16] D. Z. Wang, I. Posner, and P. Newman, "Model-free detection and tracking of dynamic objects with 2D lidar," *Int. J. Robot. Res.*, vol. 34, no. 7, pp. 1039–1063, Jun. 2015.

[17] F. Moosmann and C. Stiller, "Joint self-localization and tracking of generic objects in 3D range data," in *Proc. IEEE ICRA*, 2013, pp. 1138–1144.

[18] D. Held, J. Levinson, and S. Thrun, "Precision tracking with sparse 3D and dense color 2D data," in *Proc. IEEE ICRA*, 2013, pp. 1138–1145.

[19] D. Held, J. Levinson, S. Thrun, and S. Savarese, "Robust real-time tracking combining 3D shape, color, and motion," *Int. J. Robot. Res.*, vol. 35, no. 1–3, pp. 30–49, Jan. 2016.

[20] D. D. Morris, R. Hoffman, and P. Haley, "A view-dependent adaptive matched filter for ladar-based vehicle tracking," in *Proc. 14th Int. Conf. Robot. Appl.*, 2009, pp. 1–9.

[21] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Auton. Robots*, vol. 26, no. 2, pp. 123–139, Apr. 2009.

[22] N. Wojke and M. Haselich, "Moving vehicle detection and tracking in unstructured environments," in *Proc. IEEE ICRA*, 2012, pp. 3082–3087.

[23] T. D. Vu and O. Aycard, "Laser-based detection and tracking moving objects using data-driving Markov chain Monte Carlo," in *Proc. IEEE ICRA*, 2009, pp. 3800–3806.

[24] Z. Liu, D. Liu, and T. Chen, "Vehicle detection and tracking with 2D laser range finders," in *Proc. 6th Int. CISP*, 2013, pp. 1006–1013.

[25] K. Granstrom, S. Reuter, D. Meissner, and A. Scheel, "A multiple model PHD approach to tracking of cars under an assumed rectangular shape," in *Proc. 17th Int. Conf. Inf. Fusion*, 2014, pp. 1–8.

[26] 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, Aug. 2015.

[27] Q. Baig, M. Perrollaz, J. B. D. Nascimento, and C. Laugier, "Using fast classification of static and dynamic environment for improving Bayesian occupancy filter (BOF) and tracking," in *Proc. ICARCV*, 2012, pp. 656–661.

[28] F. Moosmann, O. Pink, and C. Stiller, "Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion," in *Proc. IEEE Intell. Veh. Symp.*, 2009, pp. 215–220.

[29] A. Petrovskaya and S. Thrun, "Efficient techniques for dynamic vehicle detection," in *Proc. 11th ISER*, 2009, pp. 79–91.

[30] S. Thrun, W. Burgard, and D. Fox, *Probabilistics Robotics*. Cambridge, MA, USA: MIT Press, 2005.

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

[32] T. Chen *et al.*, "Likelihood-field-model-based vehicle pose estimation with velodyne," in *Proc. IEEE 18th ITSC*, 2015, pp. 296–302.

[33] T. Chen, B. Dai, D. Liu, H. Fu, and J. Song, "Likelihood-field-model-based dynamic vehicle detection with velodyne," in *Proc. 7th Int. Conf. IHMSC*, 2015, pp. 497–502.

[34] T. Chen, B. Dai, D. Liu, and J. Song, "Likelihood-field-model-based vehicle tracking with velodyne," in *Proc. 8th Int. CISP*, 2015, pp. 1374–1379.

[35] 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.

[36] R. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE ICRA*, 2011, pp. 1–4.

[37] T. Chen, B. Dai, R. Wang, and D. Liu, "Gaussian-process-based real-time ground segmentation for autonomous land vehicle," *J. Intell. Robot. Syst.*, vol. 76, no. 3/4, pp. 563–582, Dec. 2014.

[38] K. Klasing, D. Wollherr, and M. Buss, "A clustering method for efficient segmentation of 3D laser data," in *Proc. IEEE ICRA*, 2008, pp. 4043–4048.


**Tongtong Chen** received the M.S. degrees in control science and engineering from National University of Defense Technology, Changsha, China, in 2011. He is currently working toward the Ph.D. degree in computer science with the College of Mechatronic Engineering and Automation, National University of Defense Technology.

His research interests include 3-D point cloud processing, 3-D object recognition, and laser-based vehicle detection and tracking.

**Ruili Wang** received the Ph.D. degree in computer science from Dublin City University, Dublin, Ireland.

He is currently an Associate Professor with the School of Engineering and Advanced Technology, Massey University, Auckland, New Zealand, and is also the Director of the Center of Language and Speech Processing. His research interests include speed processing, language processing, image processing, data mining, intelligent systems, and complex systems.

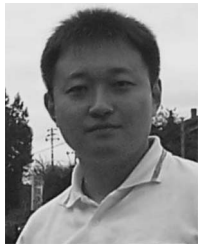
Dr. Wang serves as an Associate Editor and a member of the editorial boards of five international journals. He has received one of the most prestigious research grants in New Zealand, i.e., the Marsden Fund.

**Bin Dai** received the Ph.D. degree in control science and engineering from National University of Defense Technology, Changsha, China, in 1998.

He is currently a Professor with the Unmanned System Institute, National University of Defense Technology. His research interests include pattern recognition, computer vision, and intelligent vehicles.

**Daxue Liu** received the M.S. and Ph.D. degrees in control science and engineering from National University of Defense Technology, Changsha, China, in 2003 and 2009, respectively.

He is currently a Lecturer teaching pattern recognition with the Unmanned System Institute, National University of Defense Technology. His research interests include pattern recognition and intelligent vehicle control.



**Jinze Song** received the M.S. and Ph.D. degrees in control science and engineering from National University of Defense Technology, Changsha, China, in 2003 and 2009, respectively.

He is currently a Lecturer with the Graduate School, National University of Defense Technology. His research interests includes intelligent vehicle control.