remote sensing

Article

Moving Object Detection and Tracking with Doppler LiDAR

Yuchi Ma [1], John Anderson [2], Stephen Crouch [3] and Jie Shan [1,*]

[1] Lyles School of Civil Engineering, Purdue University, West Lafayette, IN 47907, USA; ma522@purdue.edu
[2] Geospatial Research Lab, Corbin Field Station, 15319 Magnetic Lane, Woodford, VA 22580, USA; John.Anderson@erdc.dren.mil
[3] Blackmore Sensors and Analytics, Inc., Bozeman, MT 59718, USA; crouch@blackmoreinc.com
* Correspondence: jshan@purdue.edu

Received: 9 April 2019; Accepted: 10 May 2019; Published: 14 May 2019

Abstract: In this paper, we present a model-free detection-based tracking approach for detecting and tracking moving objects in street scenes from point clouds obtained via a Doppler LiDAR that can not only collect spatial information (e.g., point clouds) but also Doppler images by using Doppler-shifted frequencies. Using our approach, Doppler images are used to detect moving points and determine the number of moving objects followed by complete segmentations via a region growing technique. The tracking approach is based on Multiple Hypothesis Tracking (MHT) with two extensions. One is that a point cloud descriptor, Oriented Ensemble of Shape Function (OESF), is proposed to evaluate the structure similarity when doing object-to-track association. Another is to use Doppler images to improve the estimation of dynamic state of moving objects. The quantitative evaluation of detection and tracking results on different datasets shows the advantages of Doppler LiDAR and the effectiveness of our approach.

Keywords: laser scanning; Doppler LiDAR; moving object tracking; moving object detection

1. Introduction

Object tracking is defined as the method of following one or more objects over a sequence of time steps. It is studied extensively due to its broad applications in visual surveillance, sports, traffic monitoring, robotics, and autonomous driving. It is a challenging topic because the number of objects in the scene can vary, and their dynamic states may change as they move and interact with the environment. To track multiple objects, basically there are two challenges: (1) how any number of objects in a scene is determined and (2) how an identified object is associated in continuous and discontinuous frames.

Current research on moving object detection and tracking is mainly vision-based (images and/or videos) [1–6]. Information collected by optical sensors offers substantial and distinct features thus enabling easy segmentation of objects in crowded scenes. However, with optical sensors alone it is difficult to derive objects' precise 3D positions and the results are heavily subject to illumination conditions. As such, some efforts try to combine 2D/3D laser rangers and cameras by using data fusion approaches [7]. Such hybrid methods always face the precise calibration problem between the ranging device and the camera [8]. Other research has been carried out exclusively on ranging/laser scanner data in the form of 3D point clouds. The advantages of a laser scanner include: its high accuracy in position measurement (precision level can be 1 cm), direct measurement of 3D position, penetration of various barriers (e.g., trees, bushes) and an active sensing modality allowing it to work both during day and night. Due to these features, research on detection and tracking with LiDAR is gaining increased attention.

To detect an object from point clouds, we can use either model-based approaches or model-free methods [9]. Model-based approaches detect objects based on prior model information. Such approaches are preferred when the interest targets are specified and therefore can be modeled in advance [9–11]. Model-free approaches don't require prior model information so that objects of arbitrary shapes and sizes can be detected. For example, [12] project all 3D points onto 2D occupancy grids parallel to the ground and clusters all connected cells into one large cell; the clustered objects are then classified as human or non-human by a trained support vector machine. [13] propose a simultaneous detection and tracking method without specifically detecting the objects in each frame. [14] uses a cluster detector to detect moving objects in LiDAR scans and tracks them with an unscented Kalman filter and nearest neighbor data association methods. [15] construct a system using a variant of Random Sample Consensus (RANSAC) that estimates the vehicle pose and detects moving objects via distinguishing non-stationary objects from stationary ones by the spatial consistency and differentiating moving objects from outlying objects by temporal consistency. [16] detect static points by traversing a voxel grid and finding differences in volumetric occupancy with "point shadows". In addition, some methods use point cloud descriptors to assist detection and object-to-track association. In the work of [9], uniformly sampled key points in consecutive scans are matched by their Signature of Histograms of OrienTations or SHOT [17]. [18] organize local convexity criterion with the normal vector and flatness value for each point, which are used to detect moving objects and manage tracks. [19] obtain high-order feature representations for 3D local patches by deep learning techniques; then automatically detect 3D vehicles using Deep Boltzmann Machine Hough Forests.

Most of current approaches, model-based or model-free, exclusively concentrate on certain objects but face limitations when applied on others. For example, methods proposed by [20] only detect pole-like objects in urban scenes; the motion-based method proposed by [9] can segment and track vehicles of various types but it does not work well for pedestrians because slowly-moving pedestrians do not give enough motion cues; [13] focuses on detecting and tracking pedestrians in LiDAR scans. In practice, however, all types of moving objects can be of interest and present in the same scene. Moreover, in some specific applications, detecting moving objects and measuring speed by calculating multiple pulse bounces with commonly used pulsed LiDAR scans is slow and error-prone. For example, [12] needs to segment and recognize point clouds of pedestrians in multiple Velodyne 64 scans before tracking them; [13] distinguish moving points in HDL-64E Velodyne laser scans with two methods: one is to assess the number of nearest points within a certain temporal window with the idea that moving object should have moved out of its original occupied volume; another is to label moving points after drawing the distance map of the static environment with multiple frames. If obscurants only allowed some points to be observed in only one frame, those methods would fail to tell whether those points are moving or not.

Doppler LiDAR (also named Frequency-Modulated Continuous Wave (FMCW) Laser Detection And Ranging (LADAR)) [21] offers remedies to those limitations. Unlike pulsed LiDAR, Doppler LiDAR continuously illuminates the target with a continuous wave. The frequency shift between the reference signal and the reflected signal is used to derive the range. However, when the target is moving, the range and Doppler effect can both cause frequency shifts, resulting in an ambiguity in the measurement between range and speed. The ambiguity is solved by the use of sequential chirps moving in upward and downward directions, where the surface acoustic wave device only compresses one of them [22,23]. Therefore, Doppler LiDAR can collect precise 3D spatial information and relative speed between the moving objects and the sensor on the beam direction, from which we can directly get the motion cue with high precision. Moreover, it is worth noting that Doppler LiDAR allows for a high precision velocity estimate to be made very quickly compared to RADAR. A processing interval can be about 1000 times less for a Doppler LiDAR than a FMCW RADAR because LiDAR's carrier frequency is 1000 times higher.

In this research, we take advantages of Doppler LiDAR and develop a model-free, detection-based tracking approach for moving objects. The framework of the proposed approach is shown in Figure 1.

Its first step is to detect moving objects in the scene. Moving points, indicated by Doppler information, are clustered to determine the number of moving objects and their approximate positions, thus we can address the first challenge of tracking multiple objects. Then, a region growing technique is used to completely segment each moving object. Results of detection are input to the tracking algorithm. In every frame, each detected object has basically three possible affiliations: (1) it belongs to an existing track, (2) it is the starting of a new track, or (3) it is a false alarm. This leads to the second challenge of tracking multiple objects mentioned before. To better handle potential wrong object-to-track associations, false alarm and missing detections, Multiple Hypotheses Tracking (MHT) was proposed [24]. When receiving a new observation, MHT considers its three possible affiliations by retaining all potential object-to-track assignments in the form of target trees. With new observations in each frame, target trees keep spawning and growing as a set of hypotheses is propagated with the expectation that future measurements will resolve assignment ambiguities [25]. Therefore, MHT can recover wrong object-to-track assignments and is a principal reason it was selected to manage tracks in this research. To increase the accuracy and robustness of object-to-track association, Oriented Ensemble of Shape Functions (OESF), a global Point Cloud Descriptor (PCD), is proposed to measure the structure similarity between tracked point clouds of objects and new detections. We also integrate speed information into MHT to improve the statue prediction, thus increasing robustness in tracking. We have described the proposed model-free detection-based tracking method as "MHT-PCD-Speed".

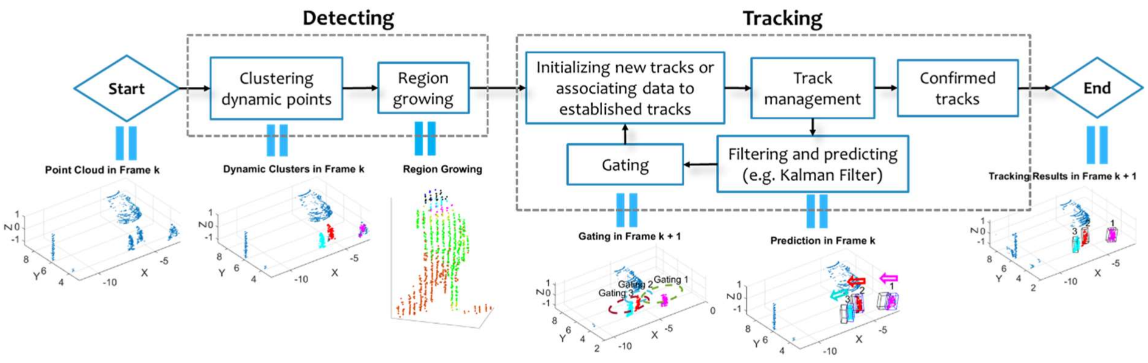

Detecting
Tracking
Start
Clustering dynamic points
Region growing
Initializing new tracks or associating data to established tracks
Track management
Confirmed tracks
End
Gating
Filtering and predicting (e.g. Kalman Filter)
Point Cloud in Frame k
Dynamic Clusters in Frame k
Region Growing
Gating in Frame k + 1
Prediction in Frame k
Tracking Results in Frame k + 1

Figure 1. Workflow of the proposed detection-based tracking approach.

This paper is organized as follows. In Section 2, datasets collected via Doppler LiDAR are introduced. Sections 3 and 4 introduce and describe the detection and tracking session. Section 5 presents experiments and analysis. Finally, we offer conclusions of the study in Section 6.

2. Data

This study uses two datasets collected by two types of Doppler LiDAR, a static terrestrial Doppler LiDAR and a mobile Doppler LiDAR which both provide point by point velocity measurements. The static terrestrial Doppler LiDAR [21] is designed for fine scanning of certain region or targets, which is realized by a partially reflective mirror that targets the laser beam at a certain direction with a scanning angle of 4°. The scanning frequency is 5 Hz. The maximum scanning range can be over 1 km. We collected data with the static terrestrial Doppler LiDAR in October 2017. The testing field is located near Blackmore Sensors and Analytics Inc. Bozeman, Montana, USA (Figure 2a,b).

The mobile Doppler LiDAR is designed for autonomous driving or mapping. In the system, four co-registered Doppler LiDAR scanners are mounted on the top of a vehicle (bottom left of Figure 2). The horizontal scanning angle range for each scanner is 40°. As the scanning scope of neighboring scanners yields an overlapping area, the total horizontal scanning angle range for the system is 120°. The maximum scanning range can be about 400 m. The test on the mobile Doppler LiDAR scans was conducted in the downtown area of San Francisco, California during September 2018 (Figure 2d,e).

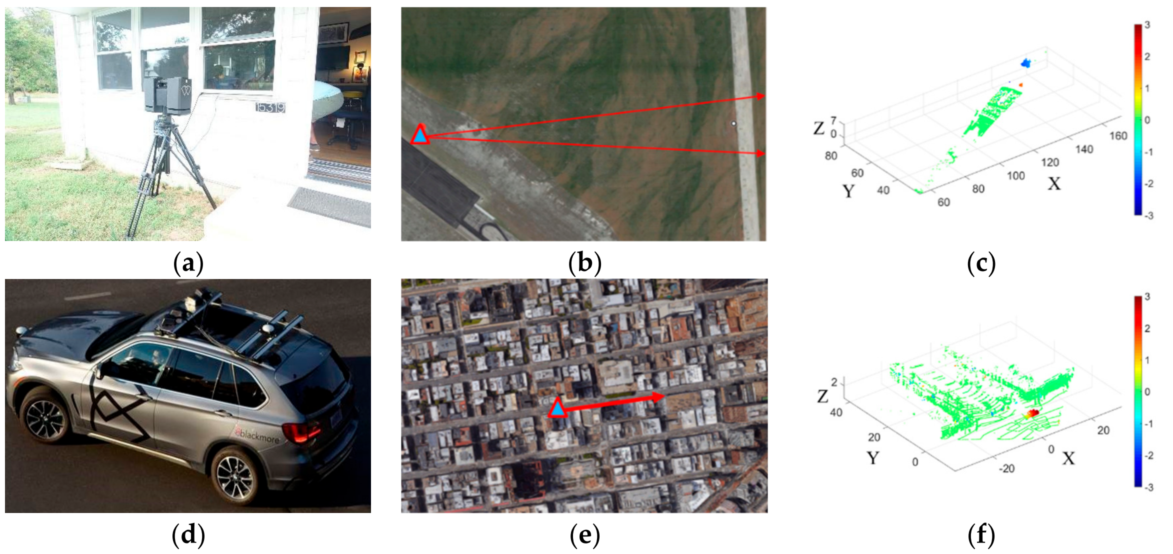


Figure 2. The scanners and study areas. The static terrestrial (**a**) and the mobile (**d**) Doppler LiDAR systems, the test site ((**b**,**e**): blue triangle: scanner; red arrow: beam direction), and one frame point cloud color coded by speed ((**c**,**f**): unit: m/s).

Both scanners record Doppler imaging, indicating the relative speed between scanned points and the scanning system (Figure 2c,f). Positive speed suggests the target moving towards the sensor along the sensor's line of sight, while a negative speed means the target is moving away from the sensor.

3. Moving Object Detection

This section first uses the Doppler image to detect and cluster moving points in the scene to determine the number of moving objects and their approximate positions in the scene. Then, a region growing technique is applied to completely segment moving objects.

3.1. Moving Points Detection and Clustering

Doppler images can be used to easily differentiate moving objects from the stationary background. However, it is difficult to directly and completely segment moving objects because not all points on the objects have the same motion state. While every part of a rigid body maintains nearly the same motion state, different parts of a non-rigid body can move in various ways (see Figure 3). For example, a pedestrian's two feet rhythmically interchange their duties by landing and moving. Moreover, due to noise in speed measurement, the measured speed of static background points is not exactly zero but a very small number (e.g., 10^{-4} m/s).

To detect the moving part of a moving object, we first use a speed threshold ϵ_v to distinguish static points and moving points. Because the precision of the radial speed measurement is ±0.1 m/s, we set ϵ_v as 0.1, namely, points whose radial speed magnitude larger than 0.1 m/s are detected as moving points, while the rest points are static.

The second step clusters detected moving points into dynamic clusters. It can be carried out according to the spatial-temporal similarity of the detected data. To process the Doppler LiDAR scans, the selected clustering method should: not assume the number of clusters be known as a priori, be capable of discovering clusters of arbitrary shapes, and have good computational efficiency on a large dataset.

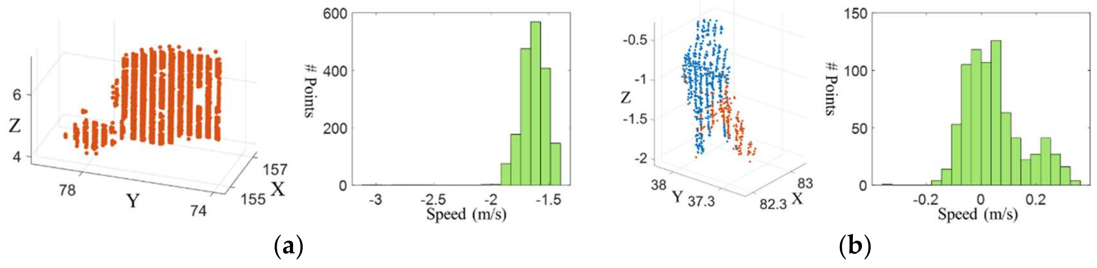


Figure 3. Moving objects and their speed histograms. It shows the point cloud of a segmented truck (**a**) and a segmented pedestrian (**b**) and the corresponding speed histograms (x-axis: speed at 0.1 m/s bin); y-axis: number of points). In this illustration, moving points are colored in red while static points blue.

A bottom-up density-based clustering algorithm, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [26], satisfies those requirements. It requires two parameters ϵ_r and $MinPts$, where ϵ_r is the spatial distance threshold between density reachable points, and $MinPts$ is the minimum number of points in the neighborhood. With the idea that data in the same cluster should be close not only in spatial domain but also in temporal domain, we adopt the Spatial-Temporal-DBSCAN (ST-DBSCAN) that additionally considers temporal similarity with a temporal threshold ϵ_T [27]. Moreover, in Doppler LiDAR scans, the point density varies with respect to the range, i.e., closer objects are more densely scanned than distant ones. To make the spatial metrics work on clusters in different distances, we make the distance threshold adaptive to range r. In summary, our ST-DBSCAN works in the following steps on a 3D point cloud dataset $D = \{p_1, p_2, \ldots, p_n\}$ frame by frame:

(1) Randomly choose an unvisited point $p_{\boldsymbol{i}} = (x_i, y_i, z_{\boldsymbol{i}}, t_i, r_i)$;
(2) Calculate the adaptive distance threshold $\epsilon(r_i)$ with the range r_i and the azimuth sampling resolution r_θ;
(3) Find all the neighboring points within spatial distance $\epsilon(r_i)$ and within temporal distance ϵ_T to p_i. If the number of neighboring points is larger than $MinPts$, go to step 4; else, label p_i as the visited noisy point and go back to step 1;
(4) Cluster all points that are density reachable or density connecting [26] to point p_i and label them as visited. Go back to step 2;
(5) Terminate the process after all the points are visited. The output is a set of clusters of dynamic points $C = \{c_1, \ldots, c_m\}$.

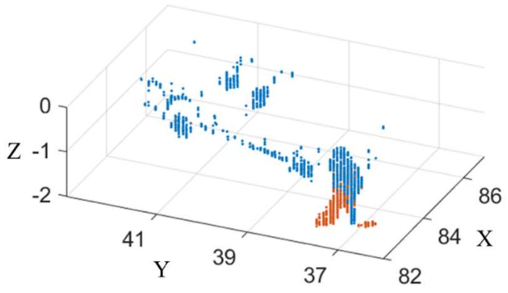
$MinPts$ is set to smooth the density estimate, which is empirically set as 40 in this study. ϵ_T is set as 0.02s. The distance threshold $\epsilon(r_i)$ decides the range of density reachability [26]. The distance between two neighboring points (point density) is associated with the azimuth sampling resolution r_θ and the distance from the point to the sensor r_i. To distinguish different objects reliably, the distance between two objects should be at least twice lager than the arc length between neighboring beams. Considering the potential non-reflection beams, we set a slightly loose threshold $\epsilon(r_i) = 3r_i r_\theta$.

3.2. Clustering Static Points—Completing the Objects

The result of dynamic points clustering is a group of dynamic point clusters $C = \{c_1, \ldots, c_m\}$. Further steps need to be taken to complete the entire moving objects by including static points. The idea is to use afore detected dynamic clusters C as seeds for region growing. A spatial threshold ϵ_{D_i} and a temporal threshold ϵ_T are set as the growing criteria. Considering the point densities on different objects are varying, ϵ_{D_i} is uniquely set as the mean distance within the dynamic cluster c_i. The mean distance within c_i is estimated using the mean value of the mean distance of each point to its K-nearest neighboring points in the cluster. To avoid overgrowing the cluster into ground points, we filter ground points beforehand using Simple Morphological Filter developed by [28]. In summary, the region growing algorithm works on every dynamic cluster c_i in dynamic clusters C with all other static points P_s as follows:

(1) Choose an unvisited dynamic cluster c_i from C;
(2) For each point p_j in c_i, calculate the mean distance $\overline{D}_{p_j}$ from p_j to its K-nearest neighboring points. The mean distance within cluster c_i with M points is calculated as $\epsilon_{D_i} = \frac{1}{M}\sum_{j=1}^{M}\overline{D}_{p_j}$;
(3) Find all static points P_s' in P_s within ϵ_{D_i} and ϵ_T from at least one point in c_i;
(4) If no point is left in P_s', go to step 1; otherwise, merge P_s' to cluster c_i and go to step 2;
(5) Terminate the region growing process when all members in C are visited.

The value of ϵ_T is set the same as in ST-DBSCAN. The value of K is set the same as the value of *MinPts*. After several iterations, the entire moving object can be completely segmented (see Figure 4). The next step is to track trajectories of detected objects between frames.



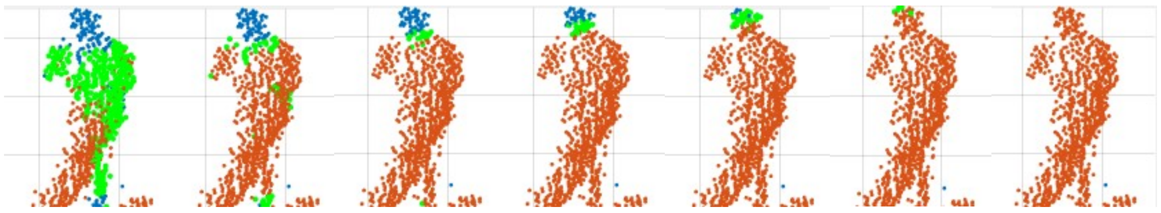


Figure 4. Clustered moving points and the static background in one frame (**top**). The growing process over seven iterations (**bottom**). Points on moving objects are in red while static points are in blue. Points in green are growing points in each iteration.

4. Moving Object Tracking

As introduced before, we adopt MHT to track moving objects in Doppler LiDAR scans. The objective of this stage is to establish the relation between objects detected in consecutive frames and determine their moving trajectories. The process of MHT includes finding candidates for object-to-track association (Gating), proposing track hypotheses and ranking each hypothesis according to their likelihood (Scoring), and determining the best set of hypotheses (Global Hypothesis Formation) [6,25,29,30]. The Kalman filter is used in MHT to estimate the dynamic states of moving objects. In the following sections, we present the implementation of the Kalman filter with Doppler speed information and then describe how the tracking algorithm works.

4.1. Kalman Filter with Doppler Images

Kalman filtering is a process of using a motion model (e.g., physical laws of motion) to estimate the state of a dynamic system [31]. In the Kalman filter model for a moving object,

$$\widetilde{S}^k = FS^{k-1} + W^k \tag{1}$$

$$\widetilde{\Sigma}_{SS}^k = F\Sigma_{SS}^{k-1}F^T + Q^k \tag{2}$$

where S^{k-1} and Σ_{SS}^{k-1} are the state vector and its covariance at frame $k-1$, consisting of position component X^{k-1} and speed components V^{k-1}; F^k is the transition matrix; W^k and Q^k are the white noise of Gaussian and its covariance at frame k. The predicted state vector and its variance at frame

k are respectively $\widetilde{X}^k$ and $\widetilde{\Sigma}^k_{XX}$. A measurement matrix H is used to project $\widetilde{S}^k$ from state space to measurement space and convert $\widetilde{\Sigma}^k_{SS}$ to the system uncertainty U^k with the measurement noise $n^k \sim N(0, R^k)$:

$$\widetilde{z}^k = H\widetilde{S}^k + n^k \tag{3}$$

$$U^k = H\widetilde{\Sigma}^k_{SS}H^T + R^k \tag{4}$$

Then, we can calculate Kalman gain K^k and update the predicted state and its variance:

$$K^k = \widetilde{\Sigma}^k_{SS}H^T(U^k)^{-1} \tag{5}$$

$$S^k = \widetilde{S}^k + K^k(z^k - \widetilde{z}^k) \tag{6}$$

$$\Sigma^k_{SS} = (I - K^kH)\widetilde{\Sigma}^k_{SS} \tag{7}$$

In our study, the position components x^k in z^k are estimated using the mean position of all points p_j on the detected object. The speed component v^k in z^k is estimated by the measured radial (in beam direction) speed $|v_b|$. Given a position observation x^k in current frame k together with the position of the scanner O, we can get the beam vector v^0_b. If the moving direction v^0 of the object is known, the moving speed can be estimated by:

$$v^k = \frac{|v_b|}{v^0_b \cdot v^0} v^0 \tag{8}$$

When a track is initialized with the speed state vector as a zero vector, we assume the vector from X^{k-1}, the updated position statue of the track in the last frame $k-1$, to x^k as the moving direction v^0. In following frames, the moving direction is estimated using the updated speed state from Kalman filter. Afterwards, based on the predicted and updated motion state, track hypotheses are proposed by performing gating as discussed below.

4.2. Gating and Proposing Track Hypotheses

Gating is the process of finding potential object-to-track associations [6,29]. Intuitively, detections in consecutive frames that are close to each other are more likely to be the same object. To quantify the likelihood, a gating area for each track hypothesis is calculated based on the position estimation in the form of (Mahalanobis) distance, which measures how many standard deviations away the jth new position observation x^k_j is from the mean of the predicted position distribution $\mathrm{N}(\widetilde{X}^k_{l_i}, \widetilde{\Sigma}^k_{XX})$ of track l_i in frame k:

$$d^2_m = (\widetilde{X}^k_{l_i} - x^k_j)^T (\widetilde{\Sigma}^k_{XX})^{-1} (\widetilde{X}^k_{l_i} - x^k_j) \tag{9}$$

A gating threshold λ_{d_m} is set to decide candidates for object-to-track associations. To illustrate the gating process, we offer an example in Figure 5a, in which observations are given in three consecutive frames, namely, two observations in frame $k-1$ and three observations in frame k and $k+1$. Gating shows that the candidates of z^{k-1}_1, the first observation in frame $k-1$, are z^k_1 and z^k_3, while the candidate of z^{k-1}_2 is z^k_2. Similar associations are indicated by the arrows from frame k to $k+1$ in Figure 5a. After gating, multiple tracking hypotheses can be proposed in the form of target trees. Existing target trees are extended by including the candidate observations, with each candidate observation "spawning" a separate new branch (i.e., branches in the red rectangle in Figure 5b). Under the assumption that each new observation in any current frame may start a new track, each new observation is used to start a new track as the root node (i.e., the root node is shown as the green rectangle in Figure 5b). In addition, each existing tree spawns a new branch with a dummy observation to account for missing detections [6] (i.e., shown as the branch in the yellow rectangle in Figure 5b).

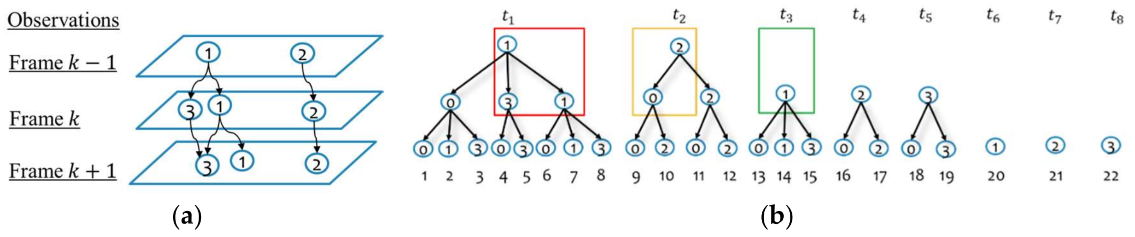


Figure 5. Gating and association principles. (**a**) Observations in three consecutive frames. There are two observations in frame $k-1$ and three object observations in frame k and $k+1$. The arrows indicate the potential object-to-track associations. (**b**) Initialized and updated target trees.

4.3. Scoring

Up to now, a set of track hypotheses are proposed while most of them are incompatible. To evaluate alternative track hypotheses, we need a probabilistic score function to rank each of them. A track is composed of a set of observations. The Likelihood Ratio (LR) of a hypothesis H_i saying a set of observations D belongs to a track l is defined as [29]:

$$LR_i = \frac{p(D|H_i)p_0(H_i)}{p(D|H_0)p_0(H_0)} = L_0 \frac{p(z_j^{1:K}\big|H_i)}{p(z_j^{1:K}\big|H_0)} = L_0 \frac{\prod_{k=1}^{K} p(z_j^k\big|z_j^{1:k-1},H_i)}{\prod_{k=1}^{K} p(z_j^k\big|H_0)} \tag{10}$$

where $p_0(H_i)$ and $p_0(H_0)$ are the prior probability of hypothesis H_i and false alarm hypothesis H_0, both of which are set as constant thus their ratio is denoted as a constant L_0. The larger $p(D|H_i)$ is and the smaller $p(D|H_0)$ is, the LR is larger and hypothesis H_i is more likely to be true. We separate the LR into a product of motion (MOT) term and a Point Cloud Descriptor (PCD) term. Under the assumption that measurement error from frame-to-frame is independent, the LR is partitioned into:

$$LR_i = L_0 \prod_{k=1}^{K} LR_{MOT_i}(k) LR_{PCD_i}(k) \tag{11}$$

The MOT term LR_{MOT} accounts for the spatial closeness of an object-to-track association. The likelihood that z_j^k belongs to track l given the previous observations $z_j^{1:k-1}$ is assumed to be a Gaussian distribution. The conditional probability that z_j^k is a false alarm is set as a constant V_0^{-1}. Using d_{m}^2 from Formula (9) and denoting M as the data dimension, LR_{MOT} for track l in the kth frame:

$$LR_{MOT_i}(k) = \frac{p(z_j^k\big|z_j^{1:k-1},H_i)}{p(z_j^k\big|H_0)} = \exp\left(-\frac{d_m^2}{2}\right)V_0\left((2\pi)^{\frac{M}{2}}\sqrt{\left|\widetilde{\Sigma}_j^k\right|}\right)^{-1} \tag{12}$$

The PCD term LR_{PCD} evaluates the structural similarity of an object-to-track association. We propose a global point cloud descriptor, Oriented Ensemble of Shape Function (OESF), to derive the feature vector of each observation. Details about OESF is given in the next section. The OESF feature vector of the $j-th$ observation in frame k is notated as β_j^k. Assuming that associating observations based on PCD is a first-order Markov process, the PCD term is:

$$LR_{PCD_i}(k) = \frac{p(\beta_j^k\big|\beta_j^{1:k-1},H_i)}{p(\beta_j^k\big|H_0)} = \frac{p(\beta_j^k\big|\beta_j^{k-1},H_i)}{p(\beta_j^k\big|H_0)} = \frac{r(\beta_j^k,\beta_j^{k-1})}{\gamma_0} \tag{13}$$

The likelihood of β_j^k belonging to track l given the previous PCD feature vector β_j^{k-1} is defined as $r(\beta_j^k,\ \beta_j^{k-1})$, the correlation coefficient between two *OESF* feature vectors β_j^{k-1} and β_j^k. $p(\beta_j^k \big| H_0)$ is set as a constant γ_0. We apply *log* on the LR to get the Log Likelihood Ratio (LLR) as the score function:

$$S_i(K) = LLR_i = \ln(LR_i) = \sum\nolimits_{k=1}^{K} \left(LLR_{MOT_i}(k) + LLR_{PCD_i}(k)\right) + \ln(L_0) \tag{14}$$

Then, we can update the track score recursively with different weights w_{MOT} and w_{PCD} on the motion term and PCD term correspondingly [6,29]:

$$S_i(k) = S_i(k-1) + \Delta S_i(k) \tag{15}$$

$$\Delta S_i(k) = \begin{cases} 0 & k = 0 \\ w_{MOT} LLR_{MOT_i}(k) + w_{PCD} LLR_{PCD_i}(k) & k \neq 0 \end{cases} \tag{16}$$

where w_{MOT} is set as 1. As we need to rely more on the PCD term if we have more candidates in the gating area, w_{PCD} is set as μn, in which n is the number of candidates and μ is the PCD weight ratio. Then the weights for the PCD term and the MOT term are normalized:

$$w_j = \frac{w_j}{w_{MOT} + w_{PCD}}, \quad j = MOT \ or \ PCD \tag{17}$$

According to scores of each track hypothesis, we can find a set of tracks with the highest overall score. Before that, we introduce the proposed OESF in the next section.

4.4. Feature Description of Point Clouds of Objects

The proposed OESF is inspired by Ensemble of Shape Function (ESF) introduced by [32]. ESF describes 3D shapes with multiple shape functions, including the distances between point pairs (D2), the square root of the area of a triangle formed by three randomly chosen points (D3), and the angle formed by three randomly chosen points (A3) [33]. ESF uses voxel grids as an approximation of the surface of the point cloud, based on which D2, D3, and A3 are extended to be either inside the surface, outside the surface, or a mixture of both. The D2 ratio is also used, which is the ratio between parts of the D2 inside and outside the surface. Values of the ten shape functions are normalized separately into a 64-bin histogram and concatenated to form the feature vector with a length of 640.

ESF gives a significant boost in classifying objects scanned by near-range depth sensors. However, it is not quite applicable to objects in our point clouds. One reason is that the dimensionality of the ESF feature vector is large. As such, it tends to overfit sparsely-scanned objects and amplify the effect of minor structural differences between objects of the same category. Also, ESF distinguishes shape functions as inside, outside, or a mixture of both, which is not applicable to comparatively sparse points on objects segmented in LiDAR point clouds. Moreover, ESF is scale-invariant but the dimension information is very important for distinguishing different moving objects in street scene (e.g., vehicles in different sizes, pedestrians of different heights etc.).

Since shape and orientation of the same moving object, either rigid or non-rigid, in neighboring frames is unlikely to change dramatically, we propose a novel point cloud descriptor to reflect the similarity between objects in neighboring frames, named Oriented Ensemble Shape Functions (OESF). To derive OESF feature vector, the oriented bounding box with center $B(x_B,\ y_B,\ z_B)$ and orientation $E(\vec{e}_x,\ \vec{e}_y,\ \vec{e}_z)$ is estimated based on the convex hull of the point cloud of the object. Five shape functions D2, D3, Ax, Ay, and Az are adopted (see Figure 6). D2 and D3 are the same as the original ESF. For each point p_a on the object, we have a center vector $\overrightarrow{Bp_a}$. The angles between $\overrightarrow{Bp_a}$ and $\vec{e}_x$, $\vec{e}_y$, $\vec{e}_z$ are respectively Ax, Ay, and Az. Results of each shape function are summarized separately into a 32-bin histogram and each histogram is normalized to be summed up as 100. Furthermore, to make OESF scale-variant, we restrict the range of the D2 histogram as 0–5 m. In addition, we introduce a layer distribution function. After rotating the points of the object to make its bounding box along with axes, the layer distribution function divides the observation point cloud into 32 uniform layers and counts the percentage of points in each layer to derive a 32-bin histogram. Finally, the six 32-bin histograms are concatenated to derive the OESF feature vector with the length of 192.

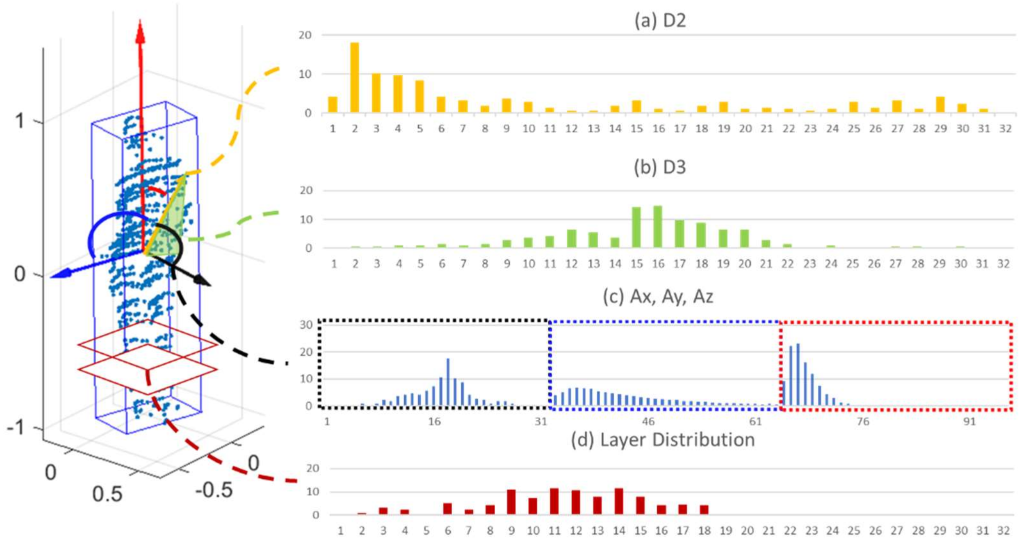


Figure 6. Oriented Ensemble of Shape Functions (OESF): (**a**) D2; (**b**) D3; (**c**) Ax, Ay, and Az; (**d**) layer distribution function.

To test its effectiveness in measuring similarity between objects in LiDAR point clouds, we calculate OESF feature vectors of segmented moving objects from both KITTI Vision Benchmark (a project of Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago) [34] and our Doppler LiDAR scans (see Table 1 and Figure 7). We use the correlation coefficient to represent the similarity between OESF feature vectors of two objects. The corresponding results based on ESF feature vectors are also given for comparison.

Our test proves the OESF feature vector can clearly show the level of similarity between objects. For the same pedestrian or vehicle in neighboring frames, the correlation coefficients are likely to be more than 0.80. We manually segment objects in KITTI datasets while the objects in Doppler LiDAR are our detection results, therefore the correlation coefficients between the same objects in KITTI datasets are comparatively lager than those in Doppler LiDAR datasets. In Doppler LiDAR, for different objects in the same categories, their correlation coefficients decrease to below 0.70, which can decrease the risk of identity switch. Moreover, the correlation coefficients between objects in different categories (e.g., pedestrian and car) are as low as about 0.20. Therefore, with the point cloud descriptor term in scoring function, false alarm detections can hardly influence the tracking results.

On the other hand, due to overfitting sparsely-scanned objects, the correlation coefficients of ESF feature vectors of two objects become higher. It can hardly differentiate different pedestrians since the correlation coefficients, either between the same or different pedestrians, are very high (larger than 0.90). In fact, the correlation coefficients of ESF feature vectors of the same pedestrian in neighboring frames can be as high as its counterpart of ESF feature vectors of two different pedestrians. Also, if different pedestrians of different heights keep a similar posture, the correlation coefficients of their ESF feature vectors would also be very high for the reason that ESF is scale-invariant. For the same reason, ESF's ability to differentiate different cars is not as good as OESF's. Moreover, ESF tends to result in much higher correlation coefficients between objects in different categories, which increases the vulnerability to false alarm detections.

Table 1. Test OESF by calculating correlation coefficients r between objects in point clouds.

Dataset	Objects	#Pair of Samples	Mean of r_{OESF}/r_{ESF}	Std. Dev. of r_{OESF}/r_{ESF}
KITTI	The Same pedestrian in neighboring frames	100	0.8848/0.9168	0.0700/0.0550
	Different pedestrians	100	0.8248/0.9543	0.1097/0.0262
	The same car in neighboring frames	50	0.9081/0.9241	0.0649/0.0741
	Different cars	150	0.7051/0.8412	0.1252/0.0772
	Objects in different categories	200	0.1348/0.5219	0.0871/0.1706
Doppler LiDAR	The Same pedestrian in neighboring frames	100	0.8248/0.9543	0.1097/0.0262
	Different pedestrians	100	0.6466/0.9534	0.1442/0.0207
	The same car in neighboring frames	50	0.8016/0.8797	0.1297/0.1045
	Different cars	150	0.5617/0.7349	0.0967/0.1514
	Objects in different categories	200	0.2191/0.5323	0.0593/0.1568

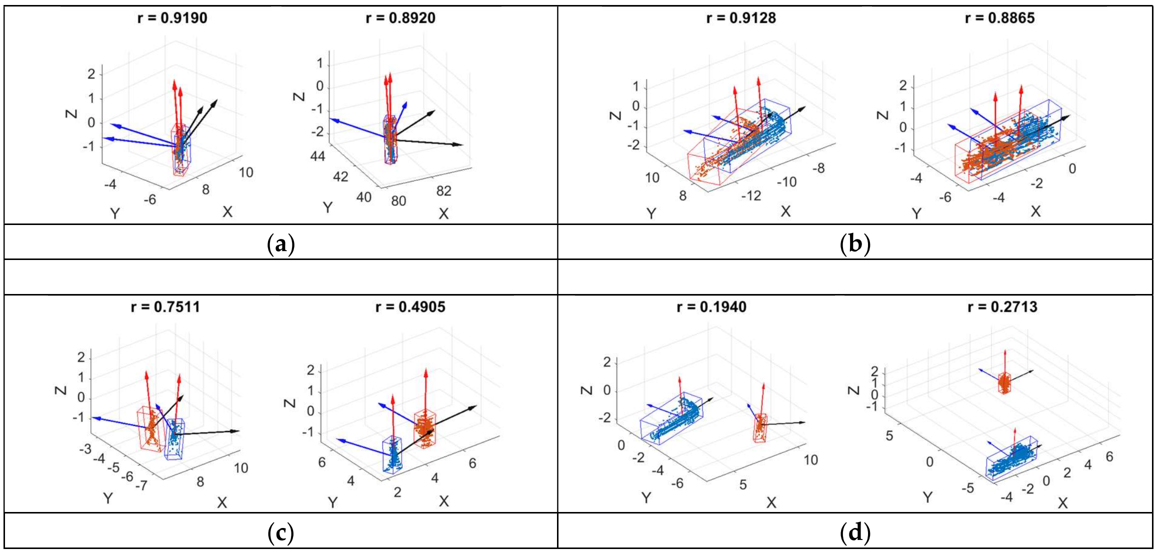


Figure 7. Calculated correlation coefficients (r) between OESF feature vectors of objects in the KITTI dataset (left) and Doppler LiDAR dataset (right): (**a**) the same pedestrian in neighboring frames; (**b**) the same vehicle in neighboring frames; (**c**) different pedestrians; (**d**) a pedestrian and a vehicle.

4.5. Managing and Confirming Track Hypotheses

Before confirming any track hypothesis, to manage tracks efficiently, track hypotheses are organized in several forms of data structure. Different tracks originating from the same root observation form a target tree, also called a family [6,29]. Tracks from the same family are incompatible. If incompatible tracks are grouped into the same cluster, we can improve computational efficiency by processing each cluster independently. For that reason, clustering is adopted to decompose all families into smaller disjoint clusters. Clustering essentially divides a large tracking problem into several smaller sub-problems [25].

After clustering, we need to format a global (joint) hypothesis, which is defined as a group of tracks that are compatible with each other [6]. The final confirmed tracks compose the most likely global hypothesis, which is the combination of all compatible tracks that gives the highest overall score. The objective function of finding the most likely global hypothesis is [6,29]:

$$\begin{array}{c} max \sum_{l=1}^{n_{t_1}} s_l^{t_1} k_l \ldots \sum_{l=1}^{n_{t_i}} s_l^{t_i} k_l \ldots \sum_{l=1}^{n_{t_T}} s_l^{t_T} k_l \\ \sum_{l=1}^{n_{t_i}} k_l = 1, k_l = 0 \; or \; 1 \; (\text{constraint}) \end{array} \tag{18}$$

in which T is the number of trees; n_{t_i} is the number of tracks in tree t_i; $s_l^{t_i}$ is the score of track l in tree t_i; κ_l is the binary constraint deciding whether track l should be included in the global hypothesis or not. κ_l ensures one observation at most appear in one track thus that all tracks in the global hypothesis are compatible with each other. The multidimensional track assignment problem can be solved as a Maximum Weight Independent Set Problem (MWISP) [35]. Following [6], we use the algorithm proposed by [36] to solve the MWISP and find the most likely global hypothesis. As a result, a set of tracks giving the highest score is confirmed.

Notably, with new observations in each frame, branches of target trees are growing exponentially. Besides clustering, N-Scan Pruning is another way to keep MHT efficient, which prunes branches of target trees regularly to ease the computational burden [6,29,35]. The value of N determines how many frames a hypothesis be kept before it is confirmed or discarded (pruned). For a confirmed branch l in tree t_i at current frame k, any other subtrees in t_i which do not spawn from the same node N frames ago as l are pruned. Branches in other target trees within the same cluster as l are pruned as well. The node l spawned from N frames ago is set as the new root node for tree t_i. For those target trees initialized less than N frames ago, no prune needs to be done (see Figure 8).

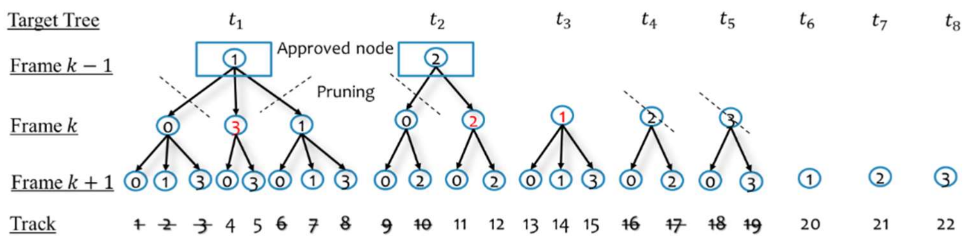


Figure 8. Pruning trees given N is equal to 2. The red nodes are the new nodes after pruning.

5. Experiments and Evaluations

Experiments have been conducted on the two datasets collected respectively by the static terrestrial Doppler LiDAR and mobile Doppler LiDAR. Both systems scan surroundings repetitively at a high frequency. We evaluate the performance of detection and tracking algorithms separately. Values for parameters used in the proposed method are summarized in Table 2. The Mahalanobis distance threshold λ_{d_m} is set as 3, according to the 3σ rule. L_0, the ratio of the prior probability of a hypothesis H_i and false alarm hypothesis H_0, is set as 10 because detected moving objects are less likely to be false alarm in Doppler LiDAR scans. V_0^{-1} is set to determine the conditional probability of null hypothesis, which should be a low value and set as 0.01. The PCD weight ratio μ is empirically set as 0.3. N for N-Scan pruning is set to determine how many frames a hypothesis should be kept if there are no consecutive new observations to update the track. A large N would increase the computational burden as much more hypotheses are kept, and identity switch errors are more likely to happen. If N is set too small, MHT would not be able to resolve assignment ambiguities with future measurements. As a trade-off, N is set as 5 with the idea that any moving objects should not be missed more than one second.

Table 2. Parameter values in the experiments.

	Detection			**Tracking**				
Parameter	ϵ_v	*MinPts*	ϵ_T	λ_{d_m}	L_0	V_0^{-1}	μ	N
Value	0.1	40	0.02s	3	10	0.01	0.3	5

5.1. Moving Object Detection

Due to the limited horizontal scanning angle, there are only a few moving objects in each frame (about one moving object per frame on average) in the static terrestrial Doppler LiDAR scans. Pedestrians are the main moving objects while at the far end vehicles pass through the scanning region now and then. Two sequences of data (sequences A and B) are tested. The length of sequence A and B are 72 frames and 81 frames respectively. The street scenarios scanned by the mobile Doppler LiDAR system are far more crowded with pedestrians, bicyclists, and vehicles (about 19 moving objects per frame on average). Two sequences of data (sequences C and D) are used for testing. In sequence C there is a total of 90 frames. The mobile system moves fast (>6 m/s) at first and then slows down and stops at a road cross waiting for people walking across the street. Sequence D has a total of 107 frames, while the mobile system moves on a busy street at 8 m/s.

For quantitative evaluation, we compare the detection results with manually labeled ground truth of the moving objects. Precision (P), the ratio between the number of correct detections and all detections, and Recall (R), the ratio between the number of correct detections and ground truth, are calculated over all frames. The F1 score ($2R \times P/(R+P)$) is calculated with the average precision and recall of all frames. In addition, Object Recall (ObjRcl), the ratio between the number of frames in which a certain object is detected and frames in which the object is actually there, is also used [9]. Results are shown in Table 3, where the computation time is based on Matlab (developed by The MathWorks, Inc). The computation time of detection is related to the number of points and number of moving objects in each frame. There are respectively about 80,000 and 16,000 points in each frame of sequence A and B while there are more moving objects in sequence A, thus it takes about two times longer to process sequence A than to process sequence B. Sequences C and D both have about 160,000 points in each frame. However, there are fewer moving points but more static points in each frame of sequence D. Therefore, region growing on sequence D requires more computation time, resulting in longer overall processing time.

Table 3. Precision, Recall, F1 Score, and Object Recall.

	Seq. ID	**#Grd TruthObj**	**#Correct Obj**	**#Wrong Obj**	**Precision**	**Recall**	**F1 Score**	**ObjRcl**	**Sec./Frame**
Terrestrial	A	107	102	9	0.9189	0.9533	0.9358	0.9667	0.65
	B	52	48	0	1	0.9231	0.9600	0.9757	0.26
Mobile	C	1683	1366	264	0.8380	0.8116	0.8246	0.8089	9.39
	D	1230	893	210	0.8096	0.7260	0.7655	0.7347	12.97

As sequences A and B are collected in non-crowded area with few moving objects per frame, the proposed detection algorithm performs well. In sequence C and D, the environment is more complex with ten times more objects appearing, resulting in more missing detection (false negative) and false alarm (false positive), thus the precision decreases to about 80%. Fast-moving objects (truck: site A in Figure 9b,d); sedan: site B in Figure 9b,c) and comparatively slow-moving objects (pedestrian: site C in Figure 9b,d) in different sizes and various categories have been successfully detected and segmented. With the usage of adaptive distance threshold $\epsilon(\mathrm{r_i})$ in clustering, objects close to each other (less than 0.2 m) can be successfully segmented in most cases (For example, four close pedestrians at site C in Figure 9d have been well segmented).

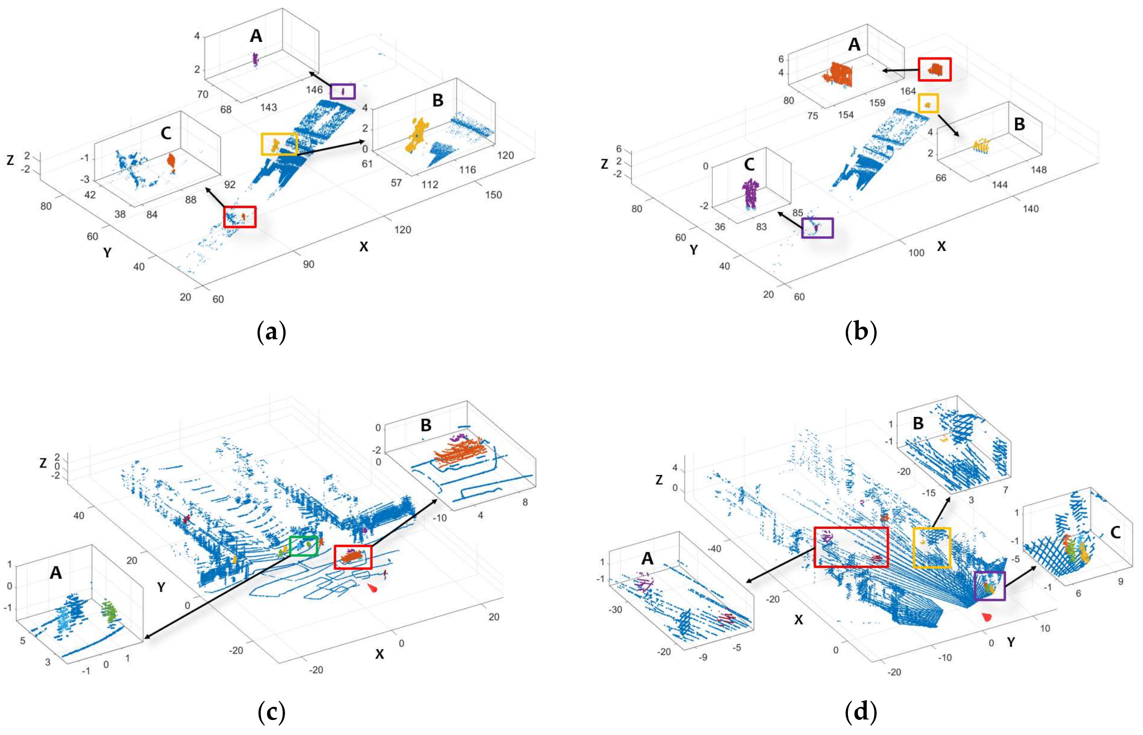


Figure 9. Detection results on static (sequences (**a**,**b**)) and mobile datasets (sequences (**c**,**d**)).

When comes to errors, in sequence A and B, missing detection rarely happens but false alarms are found in several frames, whereas both missing detection and false alarms happen more frequently in sequences C and D. There are mainly two causes to false alarms. One cause is that part of a stationary object is moving due to external force. For example, the object at site A in Figure 9a is a stationary tree but it is detected as a moving object. That is because part of its branches is detected as moving. After growing, the entire tree is segmented as a moving object. Strictly speaking, it is not a false alarm because its branches are actually moving forced by the wind. However, in the evaluation we still regard such detections as wrong detections. Another cause of false alarm is that one object is segmented as two or more objects. It happens when parts of the object have no reflected points thus dividing the object to several isolated parts. For example, the car at site B in Figure 9c is segmented into two objects in red and purple respectively. The purple part is the roof rack on the top of the car. The connection part between the car and the roof rack is not fully scanned. As the car is very close to the scanner, a small distance threshold $\epsilon(r_i)$ (less than 5 cm) is used in clustering. Therefore, the car has been segmented into two objects.

In some cases, missing detection happens when multiple objects close to each other are grouped as one object. For example, in Figure 10a, there are two people moving side by side on the left side of the street (see red rectangle). These two people appear in the scene from frames three to 81 and has been constantly segmented from frames seven to 79. After frame 79, there are very few points reflected from these two people as they are moving out of the field of view.

Missing detection also happens if the object is right in front of the scanner and moving perpendicular to the beam. In this situation, the radial speed of points on that object is almost zero. For example, there are three pedestrians at site A in Figure 9c. The pedestrians in sky blue and green are successfully detected and segmented while the one between them are missed due to the same reason we addressed above. Moreover, when severe occlusion happen, objects would be partially scanned or totally missed. For instance, at site B in Figure 9d, the segmented object in yellow is a pedestrian while only the upper part of its body has been scanned.

In summary, for moving object detection, our method can detect an arbitrary number of objects with high precision. Instead of deriving motion cues by processing a sequence of point clouds, moving objects can be efficiently detected and segmented frame by frame. False alarms can be effectively avoided except in the condition that a large object is very close to the scanner. Missing detection would happen when multiple close-by objects are detected as one object or objects moving perpendicular to the beam have very few moving points reflected.

5.2. Moving Object Tracking

Due to the narrow horizontal scanning angle of static terrestrial Doppler LiDAR, most of the fast-moving objects only appear once over all frames in sequences A and B. We need to observe the same object in at least two frames to track it. Therefore, we do not evaluate the tracking algorithm on sequences A and B. Some tracking results on sequences C and D are shown in Figures 10–12, in which the red arrow indicates the position and direction of the mobile platform; the dark blue points indicate static points; clustered point clouds in different colors represent individual detected moving objects; blue cubes with unique track ID are the tracking results.

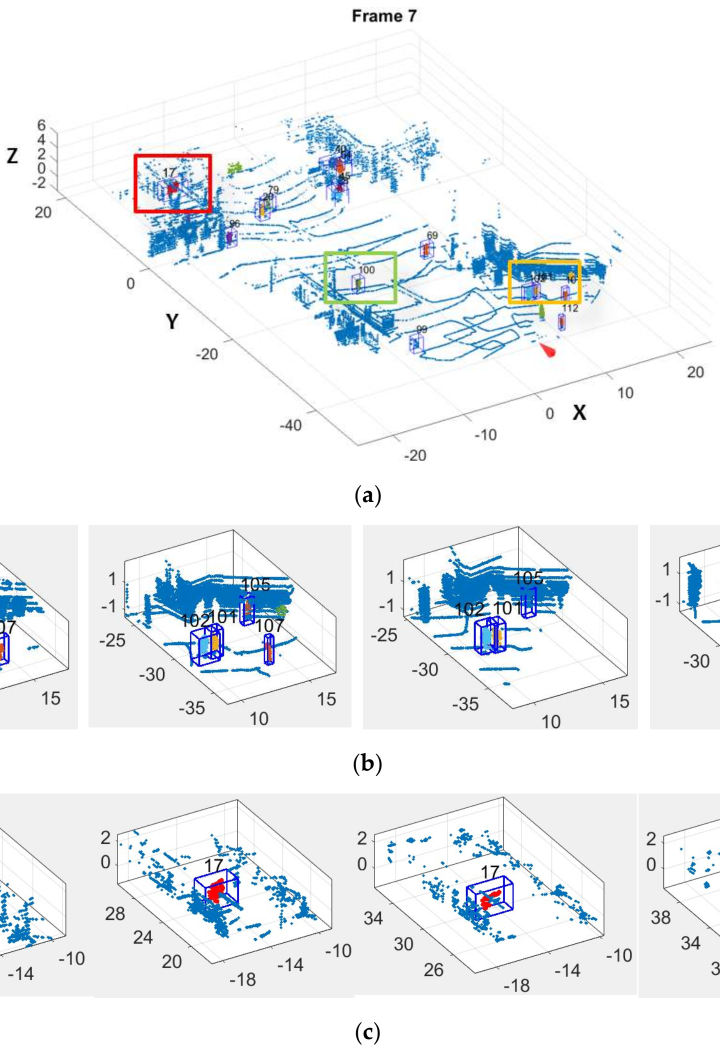
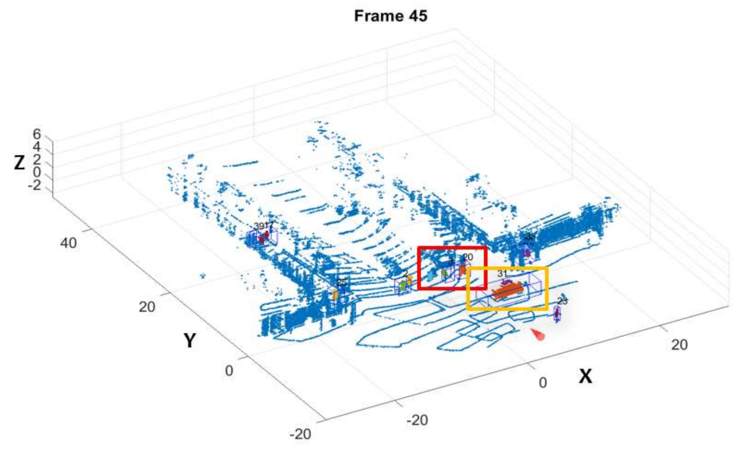
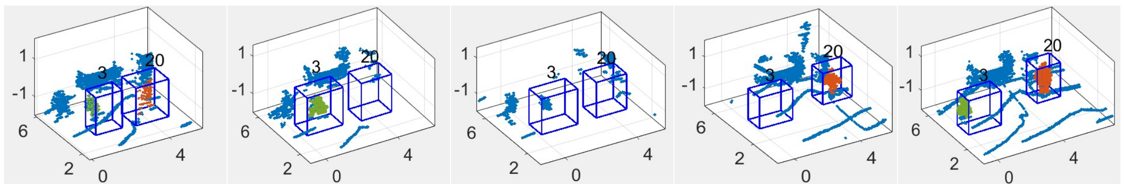


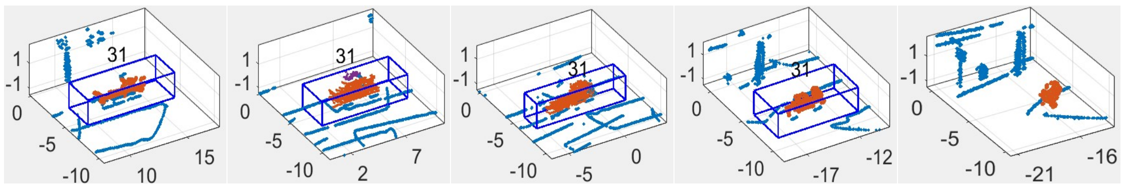
Figure 10. (**a**) Tracking results of frame seven in sequence C; (**b**) Tracks #101, #105, #107 in frames two, 10, 11 and 14 (Track #107 leaves the scene, so do Track #101 and Track #105 in frame 14); (**c**) Track #17 in frames 7, 49, 67 and 79.



(**a**)

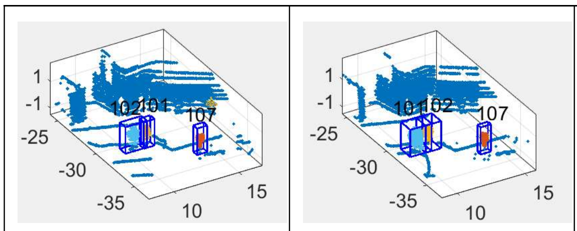


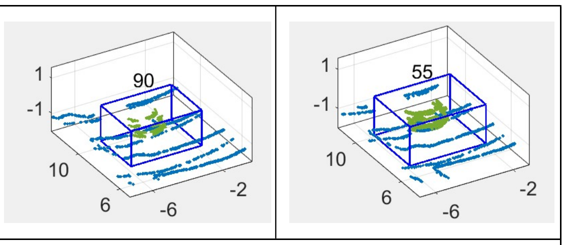
(**b**)



(**c**)

Figure 11. (**a**) Tracking results of frame 45 in sequence C; (**b**) Tracks #3 and #20 of pedestrians in frames 45 to 49 (Due to occlusion, the pedestrian in Track #3 isn't scanned in frame 47 and 48, and the pedestrian in Track #3 isn't scanned in frames 46 and 47. However, both tracks are kept and recovered when missing objects appear again); (**c**) Track #31 of a car in frames 41, 45, 49, 56, 57. Half of the car has left the scene in frame 57.





(**a**)	(**b**)

Figure 12. (**a**) One Identity SWitch (IDSW) example in frames seven and eight of sequence C using the original Multiple Hypothesis Tracking (MHT); (**b**) One IDSW example in frames 20 and 21 of sequence C using the original MHT and MHT-Point Cloud Descriptor (PCD).

To evaluate the tracking results, we use metrics proposed by MOT16, a benchmark for Multi-Object Tracking in videos [37]. We count the number of True Positive (TP) and False Alarm (or False Positive, FP). A moving object that missed being tracked is a False Negative (FN). The mismatch error, IDentity SWitch (IDSW) is counted if a target has been labeled with different track numbers. To combine multiple sources of errors and evaluate multiple object tracking accuracy, Multiple Object Tracking Accuracy (MOTA) [37,38] is widely used to evaluate a tracker's performance:

$$MOTA = 1 - \frac{\sum_k (FN_k + FP_k + IDSW_k)}{\sum_k GT_k} \tag{19}$$

Furthermore, to measure the track quality, we classify each ground truth trajectory as Mostly Tracked (MT, with at least 80% tracked), Mostly Lost (ML, with at most 20% tracked), and Partially Tracked (PT) [37]. Notably, a tracked moving object can be wrongly labeled with a different track ID in three scenarios. One scenario is that it leaves the field-of-view for several frames (within N frames) and then reappears; another is that the structure of the moving point cloud object changes a lot due to its deformation or its relative position to the scanner; the third scenario is that its label exchanges with the label of other track. All are counted as IDSW.

It shows that the proposed tracking method can track objects of various sizes in highly crowded streets. For example, as shown in Figure 10b, three pedestrians have been tracked over 14 frames until they leave the field of view. In most cases, false alarms in detection would not influence tracking because they don't appear constantly in the scene and their structures are very different from the real moving objects. For example, the roof rack on top of the car shown in Figure 9c is a false alarm detection but it is not tracked at all (see Figure 11c).

MHT keeps multiple possible tracks over N frames with the expectation that future inputs can resolve assignment ambiguities [25]. Therefore, multiple tracks can be maintained even when missing detection happens. For example, as shown in Figure 11b, two pedestrians walking in opposite directions are tracked as Track #3 and #20 respectively. Occluded by a moving vehicle (Track #31), the pedestrian in Track #3 is missed in frames 47 and 48. Similarly, the pedestrian in Track #20 is missed in frames 46 and 47. Both tracks retain and successfully retrack corresponding pedestrians when they appear again in later frames.

To evaluate the effects of including the point cloud descriptor term and speed measurement in our MHT-PCD-Speed, we compare its results with the ones of the original MHT, and the ones of MHT with PCD term (named MHT-PCD) (See Table 4). Adding PCD and speed to MHT does not have any practically meaningful influence on the computational efficiency. However, it is highly dependent on the number of moving objects detected. With more observations, MHT needs to keep more hypotheses. There are more moving objects in sequence C and most of them stay in the field of view for more than 30 frames because the scanning platform stops at an intersection in most of the frames. Therefore, when working on sequence C, the MHT needs to initialize more track trees and keep growing them for those moving objects that stay in the field of view for many frames. That is why it takes twice as much time to process sequence C than to process sequence D.

Table 4. Track Quality Evaluation on Sequences C and D.

Method	Seq.	Avg FN	Avg FP	IDSW	MOTA (%)	MT (%)	ML (%)	Sec./Frame
MHT-PCD-Speed	C	3.66	0.40	31	76.45	66.13	9.68	1.53
	D	2.25	0.75	19	72.42	51.65	27.47	0.68
MHT-PCD	C	3.89	0.59	35	73.96	56.45	12.90	1.59
	D	2.67	0.85	27	67.26	43.96	28.57	0.68
MHT	C	4.83	0.64	37	68.55	53.23	16.13	1.54
	D	3.56	0.98	41	57.28	41.76	32.97	0.67

Compared to MHT-PCD-Speed, the original MHT is more likely to make IDSW error in a crowded scene. For example, Figure 12a shows the same three people as shown in Figure 10b. While well tracked with MHT-PCD-Speed method, in the tracking result by the original MHT Track #101 and #102 switch track IDs with each other. That is because the original MHT forms tracks only based on spatial relation between track's predicted positions and detected objects' positions. MHT-PCD can kind of solve such problem as the structure of the point cloud is considered, but both the original MHT and MHT-PCD are subject to IDSW errors when objects rapidly change their dynamic states. For example, as shown in Figure 12b, there is a car coming to stop at the crossroad. It is originally tracked as #90 but switched to #55 in the next frame by both MHT and MHT-PCD. The Kalman filter predicts the position based on previous position and moving speed. Without speed measurement, the Kalman filter doesn't know the car is slowing down and predicts the position of the car using a previously estimated speed, thus the predicted position is far away from the measurement position, out of the gating area. Therefore, both the original MHT and MHT-PCD start a new Track #55. This error can be avoided by setting a larger gating threshold λ_{d_m} or increasing the scanning frequency. However, both solutions are not optimal as IDSW are more likely to happen with larger λ_{d_m}, and increasing scanning frequency will definitely impose computational burdens. Moreover, if the object moves fast (e.g., the car shown in Figure 11c), the original MHT and MHT-PCD may totally lose its track because under poor speed estimation the right candidate may not always be in the gating area. With the speed measurement estimated from Doppler images, those errors can be mostly avoided, resulting in less IDSW and more MT. As shown in Table 3, our extension to MHT by including the PCD term and Doppler speed measurement is proven to be effective as MOTA of MHT-PCD-Speed is the highest in both tests. For the overall performance in terms of MOTA the proposed MHT-PCD-Speed outperforms the original MHT by about 8% and 15% on sequence C and sequence D, respectively.

We also compare the speed estimation to the ground truth data. Because the speed state is initialized as 0 in Kalman filter, there are big jumps on speed estimations during the first few frames. Figure 13a shows the speed of a walking person appearing in the scene for 73 frames with a speed of about 1.3 m/s. The MHT-PCD-Speed not only completely tracks the person but also keeps a low RMSE (Root Means Square Error), which is 0.12 m/s. In contrary, the original MHT only tracks the person for 22 frames and then loses it. The MHT-PCD can completely track the person but with much worse speed estimation, whose RMSE is 0.81 m/s. Similarly, Figure 13b shows the speed of a slow-down vehicle appearing in the scene for 26 frames. The MHT-PCD-Speed can well estimate the speed of the moving objects while the original MHT and MHT-PCD give bad estimation. Figure 13c shows the speed of the passing vehicle in Track #31 given in Figure 11c. As explained before, the original MHT and the MHT-PCD totally lost that fast passing vehicle. The MHT-PCD-Speed well estimates its speed except in frames seven and eight, where the vehicle is right in front the scanner and moving perpendicular to the beam. At that moment, the front part of the vehicle is moving away (reflected points with negative radial speed) while the rear part is moving close to the scanner (reflected points with positive radial speed), resulting a poor overall speed estimation, but such poor estimation can be recovered very soon. As a summary, the original MHT and MHT-PCD requires a large amount of consecutive position observations to get a precise and accurate speed estimation, which is difficult to realize in complex street scenes, while our MHT-PCD-Speed can derive a precise speed estimation within three to five frames. It is also more robust to observation errors and can recover the state estimation very soon.

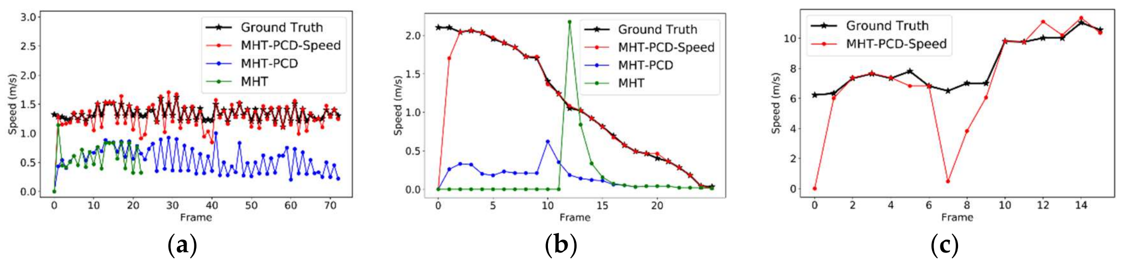


Figure 13. (**a**) Speed comparison of a pedestrian; (**b**) speed comparison of a slow-down vehicle; (**c**) speed comparison of the passing vehicle in Track #31.

We additionally attempt to reveal the applicability of MHT-PCD-Speed by summarizing the tracking results according to the category, dimension, and speed of moving objects. In terms of dimension, the object is regarded as of small size if both the ranges of x and y of its point clouds (dx and dy) are less than 0.5 m; otherwise, it is regarded as of large size. In terms of speed, we set two thresholds (0.5 m/s and 1.0 m/s) to determine whether the object is moving slow, middle, or fast.

Considering some objects are rapidly changing their motion state, we label those objects who rapidly change their speed from slow to fast or from fast to slow within one second as "change speed rapidly" (see Table 5). There is no obvious difference on its performances on objects in different categories and dimensions. Objects moving slow (less than 0.5 m/s) are more likely to be mostly tracked.

Table 5. Track Quality Evaluation on Objects of Different Types.

Type	Classification	#Obj	MT (%)	ML (%)
Category	Pedestrian	142	56.34	20.42
	Vehicle	10	51.65	20.00
	Bicyclist	1	1	0
Dimension	Small (dx < 0.5 and dy < 0.5)	140	56.42	20.00
	Large (dx > 0.5 || dy > 0.5)	13	61.54	23.08
Speed	Slow (v < 0.5 m/s)	60	53.23	16.13
	Middle (0.5 < v < 1 m/s)	70	41.76	32.97
	Fast (v > 1 m/s)	18	33.33	5.56
	Changing rapidly	5	40.00	20.00

In summary, in the proposed MHT-PCD-Speed, the use of MHT makes the method robust to missing detection. It also makes it possible to fix wrong tracking with both previous and later observations. By introducing PCD term in scoring and considering structural similarity with OESF, the tracking method can process highly crowded scenes with few IDSW mistakes, and false alarm detections are unlikely to influence tracking results. Moreover, it has been demonstrated that the use of speed based on Doppler images enhance the tracking reliability and increase the precision of dynamic state estimation on moving objects, especially those whose speed is changing fast.

6. Conclusions

In this study, we propose a novel model-free detection-based tracking approach, MHT-PCD-Speed, to detect and track moving objects in Doppler LiDAR scans. Our tests reveal the unique advantages of Doppler LiDAR and demonstrate the effectiveness and robustness of the proposed detection and tracking approach. Our basic contributions include:

- The study reveals that the use of Doppler images can enhance the tracking reliability and increase the precision of dynamic state estimation;
- In detection, moving objects are clustered and segmented based on speed information with an adaptive ST-DBSCAN and a region growing technique. The detection method doesn't require multiple sequential frames of point clouds as input;
- In tracking, the dynamic state of moving objects is estimated with position observation and speed observation, increasing the precision of dynamic state estimation on moving objects, especially those whose speed is changing fast;
- A point cloud descriptor, OESF, is proposed and added in the scoring process of MHT, which allows managing tracks not only according to spatial closeness but also similarity in structure of detections in neighboring frames.

The performance of the detection approach and tracking approach has been discussed in detail. False positives occur when objects are moving perpendicular to the beam direction. Also, if the moving object possesses a large dimension, parts of its body may move close to the scanner while the other parts move away, resulting a wrong speed estimation. One solution to these two limitations is to mount several scanners and detect moving objects in point clouds collected by each scanner independently before projecting those point clouds to the same coordinate system. With several scanners, it can be guaranteed an object doesn't move perpendicular to all beams simultaneously. This solution requires the information of precise position and pose of each scanner, which is stored in the scanning system but not included in the output files. After we get access to that necessary information, we will further process point clouds from each scanner independently to improve the results.

Also, a more representative and robust point cloud descriptor is needed. Current OESF relies heavily on the point distribution of the objects, which is subject to objects' gestures as well as the scanning angle. One solution is to introduce vision-based sensors so that the current point cloud descriptor can be extended by including RGB features. With that additional information, close objects (such as the multiple hand-in-hand people whom we track as one object) can be separated, making the framework more resistant to IDSW errors. Another solution is to mine highly arbitrary features with deep learning methods, which requires massive data and computational power.

In terms of computational efficiency, out method clearly cannot meet the requirement for real-time application at this time. The processing time is related to the number of moving objects and the number of tracks in the scene. In detection, the most time-consuming part is the region growing which requires searching nearest neighbors and computing distance between points in dynamic clusters to all nearby static points. In tracking, adding PCD and speed to MHT does not have significant influence on the computation time. We need a more efficient way to manage multiple hypotheses besides clustering and N-Scan pruning. Moreover, another way to improve the efficiency is to realize the method with some more basic programming language (e.g., C/C++) instead of Matlab or have it accelerated under a Graphics Processing Unit (GPU) environment.

Our method is not directly applicable to point clouds collected by commonly-used pulsed LiDAR sensors (e.g., Velodyne HDL-64E) because speed information is required in every single frame. When it comes to the applicability of MHT-PCD-Speed, Table 5 shows it works indifferently on objects in different categories and dimensions while better on tracking slow-moving (less than 0.5 m/s) objects. However, we do not have a conclusion about the applicability of our detection-based tracking method on Doppler LiDAR for several reasons. One reason is that we don't have a balanced dataset. Although in total we have about 3000 objects appearing over all frames, the number of unique objects is 153, with less than 10% of them being vehicles and only one bicyclist. Therefore, we cannot conclude whether our method works better on tracking pedestrians or vehicles. Another reason is that the way to decide the size of objects is not very reliable as the size of point clouds of objects would change due to the movement of non-rigid body objects and the variation of scanning angle. Namely, a pedestrian who has a large stride may be labelled as a small object and large object alternately. Moreover, moving objects constantly interact with the environment, which can affect the tracking results. For example, the tracking results on the same pedestrian would be totally different if he or she is moving alone in an open field, moving together with a crowd of people, or moving close to static objects (e.g., walls). This study does reveal the advantages of the state-of-the-art Doppler LiDAR and proves the effectiveness of our detection-based tracking approach. However, a benchmark Doppler LiDAR dataset which considers and controls the environmental variables will need to be acquired before we can conclude the applicability of the proposed method and compare our method with other tracking methods.

Author Contributions: Y.M. and J.S. designed the conceptualization and methodology of the paper. Y.M. carried out the tests and evaluation under the advisory of J.S. J.A. and S.C. involved in data collection. All authors contributed to result interpretation and the writing.

Funding: The work was supported in part by the Army Research Office.

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

References

1. Pourezzat, A.A.; Sadabadi, A.A.; Khalouei, A.; Attar, G.T.; Kalhorian, R. Object Tracking: A Survey. *Adv. Environ. Biol.* **2013**, *7*, 253–259.
2. Prajapati, D.; Galiyawala, H.J. A Review on Moving Object Detection and Tracking. *Int. J. Comput. Appl.* **2015**, *5*, 168–175.

3. Schindler, K.; Ess, A.; Leibe, B.; Van Gool, L. Automatic detection and tracking of pedestrians from a moving stereo rig. *ISPRS J. Photogramm. Remote Sens.* **2010**, *65*, 523–537. [CrossRef]
4. Cao, Y.; Guan, D.; Wu, Y.; Yang, J.; Cao, Y.; Yang, M.Y. Box-level segmentation supervised deep neural networks for accurate and real-time multispectral pedestrian detection. *ISPRS J. Photogramm. Remote Sens.* **2019**, *150*, 70–79. [CrossRef]
5. Luo, H.; Li, J.; Chen, Z.; Wang, C.; Chen, Y.; Cao, L.; Teng, X.; Guan, H.; Wen, C. Vehicle Detection in High-Resolution Aerial Images via Sparse Representation and Superpixels. *IEEE Trans. Geosci. Remote Sens.* **2015**, *54*, 103–116.
6. Kim, C.; Li, F.; Ciptadi, A.; Rehg, J.M. Multiple Hypothesis Tracking Revisisted. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015.
7. Spinello, L.; Triebel, R.; Siegwart, R. Multimodal People Detection and Tracking in Crowded Scenes. In Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, Chicago, IL, USA, 13–17 July 2008; pp. 1409–1414.
8. Kaestner, R.; Maye, J.; Pilat, Y.; Siegwart, R. Generative object detection and tracking in 3d range data. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; IEEE: Piscataway, NJ, USA; pp. 3075–3081.
9. Dewan, A.; Caselitz, T.; Tipaldi, G.D.; Burgard, W. Motion-based detection and tracking in 3D LiDAR scans. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 4508–4513.
10. Cui, J.; Zha, H.; Zhao, H.; Shibasaki, R. Laser-based detection and tracking of multiple people in crowds. *Comput. Vis. Image Underst.* **2007**, *106*, 300–312. [CrossRef]
11. Brock, O.; Trinkle, J.; Ramos, F. Model Based Vehicle Tracking for Autonomous Driving in Urban Environments. *MIT Press* **2009**, *1*, 175–182.
12. Wang, H.; Wang, B.; Liu, B.; Meng, X.; Yang, G. Pedestrian recognition and tracking using 3D LiDAR for autonomous vehicle. *Rob. Auton. Syst.* **2017**, *88*, 71–78. [CrossRef]
13. Xiao, W.; Vallet, B.; Schindler, K.; Paparoditis, N. Simultaneous detection and tracking of pedestrian from panoramic laser scanning data. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2016**, *3*, 295–302. [CrossRef]
14. Yan, Z.; Duckett, T.; Bellotto, N. Online learning for human classification in 3D LiDAR-based tracking. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017.
15. Yang, S.; Wang, C. Simultaneous egomotion estimation, segmentation, and moving object detection. *J. F. Robot.* **2011**, *28*, 565–588. [CrossRef]
16. Schauer, J.; Nüchter, A. Removing non-static objects from 3D laser scan data. *ISPRS J. Photogramm. Remote Sens.* **2018**, *143*, 15–38. [CrossRef]
17. Salti, S.; Tombari, F.; Di Stefano, L. SHOT: Unique signatures of histograms for surface and texture description. *Comput. Vis. Image Underst.* **2014**, *125*, 251–264. [CrossRef]
18. Moosmann, F.; Stiller, C. Joint self-localization and tracking of generic objects in 3D range data. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 1146–1152.
19. Yu, Y.; Li, J.; Guan, H.; Wang, C. Automated Detection of Three-Dimensional Cars in Mobile Laser Scanning Point Clouds Using DBM-Hough-Forests. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 4130–4142. [CrossRef]
20. Yokoyama, H.; Date, H.; Kanai, S.; Takeda, H. Detection and classification of pole-like objects from mobile laser scanning data of urban environments. *Int. J. Cad/Cam* **2013**, *13*, 31–40.
21. Anderson, J.; Massaro, R.; Curry, J.; Reibel, R.; Nelson, J.; Edwards, J. LADAR: Frequency-Modulated, Continuous Wave LAser Detection And Ranging. *Photogramm. Eng. Remote Sens.* **2017**, *83*, 721–727. [CrossRef]
22. Halmos, M.J.; Henderson, D.M. Pulse compression of an FM chirped CO 2 laser. *Appl. Opt.* **1989**, *28*, 3595–3602. [CrossRef]
23. Slotwinski, A.R.; Goodwin, F.E.; Simonson, D.L. Utilizing GaalAs Laser Diodes As A Source For Frequency Modulated Continuous Wave (FMCW) Coherent Laser Radars. In *Laser Diode Technology and Applications*; International Society for Optics and Photonics: Bellingham, WA, USA, 1989; pp. 245–252.
24. Reid, D.B. An Algorithm for Tracking Multiple Targets. *IEEE Trans. Automat. Contr.* **1979**, *24*, 843–854. [CrossRef]

25. Amditis, A.; Thomaidis, G.; Maroudis, P.; Lytrivis, P.; Karaseitanidis, G. Multiple Hypothesis Tracking Implementation. *Laser Scanner Technol.* **2012**.
26. Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Kdd, Portland, OR, USA, 2–4 August 1996; Volume 96, pp. 226–231.
27. Birant, D.; Kut, A. ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data Knowl. Eng.* **2007**, *60*, 208–221. [CrossRef]
28. Pingel, T.J.; Clarke, K.C.; McBride, W.A. An improved simple morphological filter for the terrain classification of airborne LIDAR data. *ISPRS J. Photogramm. Remote Sens.* **2013**, *77*, 21–30. [CrossRef]
29. Blackman, S.; Popoli, R. *Design and Analysis of Modern Tracking Systems (Artech House Radar Library)*; Artech House: Norwood, MA, USA, 1999.
30. Fortmann, T.; Bar-Shalom, Y.; Scheffe, M. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE J. Ocean. Eng.* **1983**, *8*, 173–184. [CrossRef]
31. Yilmaz, A.; Javed, O.; Shah, M. Object tracking: A survey. *Acm Comput. Surv.* **2006**, *38*, 13. [CrossRef]
32. Wohlkinger, W.; Vincze, M. Ensemble of shape functions for 3D object classification. In Proceedings of the 2011 IEEE International Conference on Robotics and Biomimetics, Phuket, Thailand, 7–11 December 2011; pp. 2987–2992.
33. Osada, R.; Funkhouser, T.; Chazelle, B.; Dobkin, D. Matching 3D models with shape distributions. In Proceedings of the International Conference on Shape Modeling and Applications, Genova, Italy, 7–11 May 2001; pp. 154–166.
34. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 16–21 June 2012; pp. 3354–3361.
35. Papageorgiou, D.J.; Salpukas, M.R. The maximum weight independent set problem for data association in multiple hypothesis tracking. In *Optimization and Cooperative Control Strategies*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 235–255.
36. Östergård, P.R.J. A new algorithm for the maximum-weight clique problem. *Nord. J. Comput.* **2001**, *8*, 424–436. [CrossRef]
37. Milan, A.; Leal-Taixé, L.; Reid, I.; Roth, S.; Schindler, K. MOT16: A benchmark for multi-object tracking. *arXiv* **2016**, arXiv:1603.00831.
38. Stiefelhagen, R.; Bernardin, K.; Bowers, R.; Garofolo, J.; Mostefa, D.; Soundararajan, P. The CLEAR 2006 evaluation. In *Proceedings of the International Evaluation Workshop on Classification of Events, Activities and Relationships*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 1–44.