


Article

Real-Time Motion Tracking for Indoor Moving Sphere Objects with a LiDAR Sensor

Lvwen Huang ^{1,2,*} , Siyuan Chen ¹, Jianfeng Zhang ^{1,*}, Bang Cheng ³ and Mingqing Liu ¹

¹ College of Information Engineering, Northwest A&F University, Xianyang 712100, China; sy.chen@hotmail.com (S.C.); 18392105294@163.com (M.L.)

² Key Laboratory of Agricultural Internet of Things, Ministry of Agriculture, Xianyang 712100, China

³ College of Mechatronic Engineering and Automation, National University of Defense Technology, Changsha 410073, China; chengbang32@163.com

* Correspondence: huanglvwen@nwfau.edu.cn (L.H.); zjf@nwfau.edu.cn (J.Z.); Tel.: +86-137-0922-3117 (L.H.); +86-139-9287-4588 (J.Z.)

Received: 27 June 2017; Accepted: 21 August 2017; Published: 23 August 2017

Abstract: Object tracking is a crucial research subfield in computer vision and it has wide applications in navigation, robotics and military applications and so on. In this paper, the real-time visualization of 3D point clouds data based on the VLP-16 3D Light Detection and Ranging (LiDAR) sensor is achieved, and on the basis of preprocessing, fast ground segmentation, Euclidean clustering segmentation for outliers, View Feature Histogram (VFH) feature extraction, establishing object models and searching matching a moving spherical target, the Kalman filter and adaptive particle filter are used to estimate in real-time the position of a moving spherical target. The experimental results show that the Kalman filter has the advantages of high efficiency while adaptive particle filter has the advantages of high robustness and high precision when tested and validated on three kinds of scenes under the condition of target partial occlusion and interference, different moving speed and different trajectories. The research can be applied in the natural environment of fruit identification and tracking, robot navigation and control and other fields.

Keywords: 3D LiDAR; object tracking; Kalman filter; adaptive particle filter

1. Introduction

1.1. Application of LiDAR

Light Detection and Ranging (LiDAR) technology provides realistic 3-dimensional (3D) image information and has been widely utilized in various fields [1]. LiDAR sensors are commonly used in perception for autonomous vehicles because of their high accuracy, speed, and range. These characteristics make the sensors suitable for integration into the perception layer of controllers which have the capacity to avoid collisions with unpredicted obstacles [2]. LiDAR technology is also applied to field Autonomous Land Vehicles (ALVs) to detect potential obstacles. With a novel 3D LiDAR setup, the blind area around the vehicle is greatly reduced and the density of LiDAR data is greatly improved, which are critical for ALVs [3]. In addition to autonomous land vehicle applications, LiDAR is also used for navigation of unmanned aircraft systems [4]. The authors combined LiDAR to automatically identify ground objects that pose navigation restrictions such as airports and high-rises.

Meanwhile, in the field of agriculture and forestry, by combining field and LiDAR data in forests with coexisting evergreen and deciduous species, researchers modelled common forest stand variables (height, diameter, volume and biomass) with high accuracy [5]. At the same time, LiDAR point clouds data is used for comparative classification analysis of post-harvest growth detection in precision agriculture [6], while other researchers have proposed a new approach for discriminating maize

and weed plants from soil surface, evaluating the accuracy and performance of a LiDAR sensor for vegetation detection using distance and reflection values [7].

In other respects, LiDAR technology contributes to detect flood protection structures, natural or artificial in river floodplains and in coastal zones [8]. Also, 3D information derived from image dense matching or airborne LiDAR is very effective for building change detection [9]. Furthermore, for many robotics and intelligent vehicle applications, detection and tracking multiple objects based on LiDAR is one of the most important components [10].

1.2. Tracking Algorithms Based on LiDAR

LiDAR systems are commonly used for pedestrian recognition in ALVs, compared with cameras and can provide accurate range information and larger field of view [11]. For years, Kalman filters (KF) and Monte Carlo particle filters (PF) have been the two commonly used approaches to estimate motions of a target. In some early works, Song et al. [12] proposed a novel sparse learning-based object tracking algorithm utilizing 3D LiDAR data to realize moving object tracking of vehicles. The 3D point clouds acquired from LiDAR are first resampled on a virtual image plane, where the hypothesis of the targets is generated under the particle filtering framework. Guo et al. [13] proposed a pedestrian tracking algorithm initializing a KF to predict the possible position of the pedestrian centroid in the future frame. Meanwhile, Dewan et al. [14] have presented a novel model-free approach for detecting and tracking dynamic objects in 3D LiDAR scans obtained by a moving sensor. They sequentially detected multiple motions in the scene and segment objects using a Bayesian approach. Allodi et al. [15] have presented an obstacle detection, tracking and fusion algorithm which allows to reconstruct the environment surrounding the vehicle. An Unscented Kalman Filter (UKF) managing a variable number of observations, arbitrarily composable, allows to correctly address the combined tracking and fusion challenge. Moreover, Wasik et al. [16] have proposed a method based on the detection of circular features with least-squares fitting and filtering out outliers using a map-based selection. They have improved the estimate of the relative robot position and reduce its uncertainty by feeding measurements into a KF, resulting in an accurate tracking system.

For detecting and tracking moving objects in more complex cases, an occupancy grid tracking system based on particles [17] has been proposed. The proposed occupancy grid tracking solution can be classified as using the Descartes probability model of the reverse sensor and it generates a fully dynamic grid. To resolve ambiguities in complex dynamic scenes, Tuncer et al. [18] proposed a novel method for integrated tracking and segmentation of 3D LiDAR data with a non-parametric Bayesian method to combine segmentation and tracking components. In [19], Asvadi et al. proposed a 3D object tracking algorithm using a 3D-LiDAR, an RGB (Red, Green, Blue) camera and INS (Inertial Navigation System) (GPS (Global Position System)/IMU (Inertial Measurement Unit)) sensors data by analyzing sequential 2D-RGB, 3D point-cloud, and the ego-vehicle's localization data and outputs the trajectory of the tracked object, an estimation of its current velocity, and its predicted location in the 3D world coordinate system in the next time-step while in [20], feature matching, Iterative Closest Point (ICP), Kalman filtering, and dynamic mapping are combined together to estimate motions.

As mentioned above, the KF, the PF UKF and non-parametric Bayesian are used in detecting and tracking moving targets and there is no straightforward extension of their approach to a moving spherical object.

1.3. Application of Kalman Filter

The KF has achieved notable success in the areas of guidance, navigation, and control of vehicles, particularly aircraft and spacecraft. Srilekha et al. [21] introduced a new technique for detecting, tracking and counting the vehicles based on Kalman filtering. Huang et al. [22] proposed the Robust Strong Tracking Cubature Kalman Filter (RSTCKF) for spacecraft attitude estimation with a quaternion constraint. Furthermore, the KF is a widely applied concept in time series analysis used in fields such as signal processing and econometrics. Jain et al. [23] investigated the use of KF to estimate and track

both the laser PN (Phase Noises) and the NLPN (Nonlinear Phase Noises) in 100-Gb/s single channel coherent optical phase-modulated systems. The KF is one of the main topics in the field of robotic motion planning and control. Gulalkari et al. [24] proposed an object tracking and following six-legged robot (6LR) system that uses a Kinect camera based on KF and back-stepping control method. Lim et al. [25] proposed incorporating dead-reckoning using only encoder measurements, and a Kalman filter-based Gaussian Process to compensate the uncertainty. As for other aspects, Moon [26] developed a human skeleton tracking system using the Kalman filter framework, in which multiple Kinect sensors are used to correct inaccurate tracking data from a single Kinect sensor.

In order to solve the real-time tracking process for a moving sphere at indoor environments and in the future for spherical fruit identification and positioning with varying illumination, this paper first introduces the processes of sphere detection with 3D LiDAR, and then discusses the principles of the KF and PF algorithms. Next, with experimental modeling, data analysis of two tracking methods is compared, and finally we reach a conclusion. The tracking flowchart is as shown in Figure 1 below.

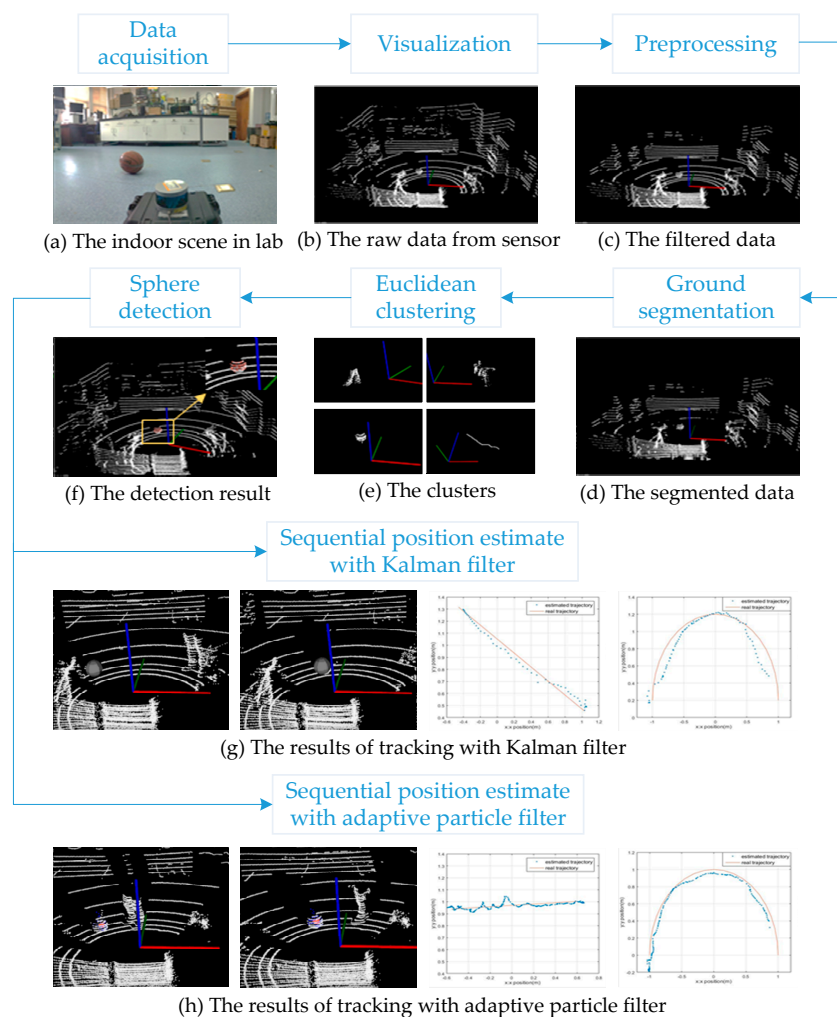


Figure 1. The tracking flowchart for a moving object.

2. Detection of Moving Spherical Object

2.1. The Velodyne System

The Velodyne VLP-16 3D LiDAR sensor obtains a 360-degree scene capture through the rotation of its internal motor. It is composed of 16 laser beams, which scan thousands of times per second. Each beam has a fixed pitch angle. The experimental scene and its visualization result are shown in Figure 2.

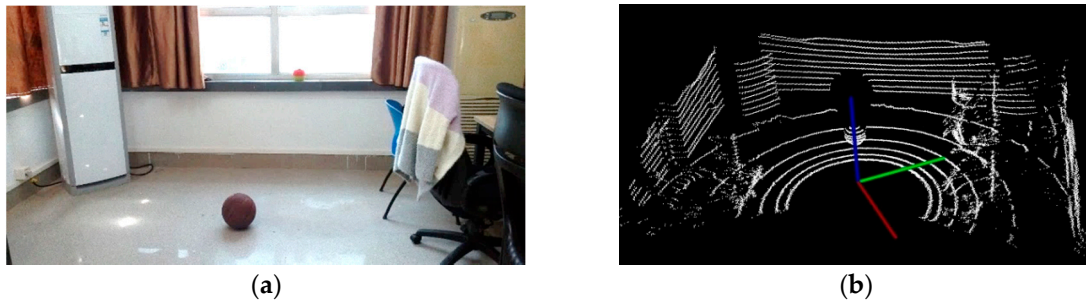


Figure 2. The results of visualization: (a) Experimental scene; (b) Point clouds visualization.

2.2. Outliers and Noise Filtering

To reduce the calculation of segmentation after the 3D data acquisition, it is necessary to eliminate some of the noise, outliers, holes, etc. by filtering according to some motion cues. Here we remove the coordinate origin $(0, 0, 0)$ and then use outlier filter proposed by Rusu et al. [27] which works well for indoor scenes. Firstly, we compute the average distance of each point between its nearest k neighbors. Next, we compute the mean μ , and standard deviation σ of all these distances to determine a distance threshold. The standard deviation coefficient α depends on the size of the analyzed neighborhood k . The distance threshold t will be equal to:

$$t = \mu + \alpha \times \sigma \quad (1)$$

In Equation (1), α is set to 1 and k is set to 30 here with the empiric and experimental threshold, especially for the indoor scene. Finally, the points can be classified as inliers or outliers if their average neighbor distance is below or above this threshold respectively. The results of filtering are shown below in Figure 3, where most of the noises and outliers marked by red ellipse are removed.

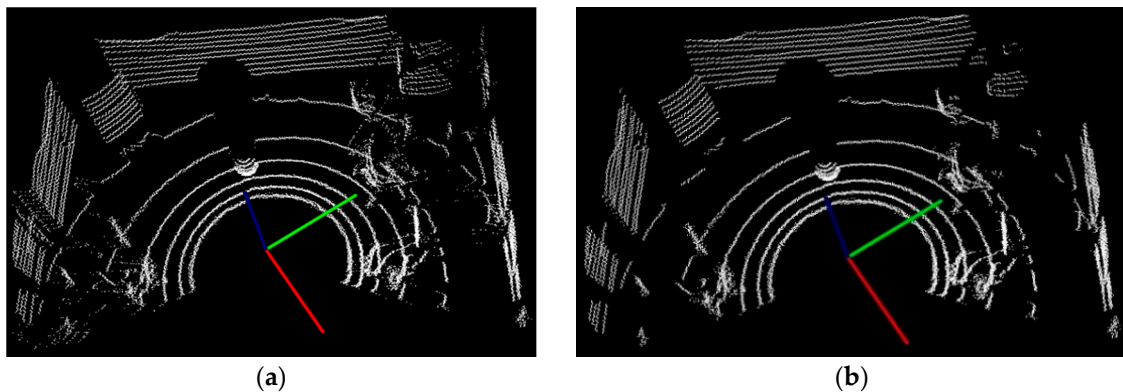


Figure 3. The results of outliers and noise filtering: (a) Before filtering; (b) After filtering.

2.3. Fast Ground Segmentation

For the indoor object motion tracking, the ground segmentation is essential to cancelling the ground background information. The fast ground segmentation algorithm proposed by Himmelsbach et al. [28] is used to remove the ground noise and to reduce the amount of subsequent calculations, which requires less runtime and obtains good segmentation results.

2.3.1. 3D Point-Cloud Data Set Mapping

1. Define the unordered 3D point clouds from a scan time t of the LiDAR sensor as $P_t = \{p_1, \dots, p_{N_t}\}$, where N_t denotes the number of 3D point clouds. The $p_i = (x_i, y_i, z_i)^T$

denotes a 3D point, given by the Euclidean coordinates to the ego-coordinate system with original point at the center of the LiDAR sensor.

- The x - o - y plane denotes a circle with a radius of R , and then cut the circle equally into multiple discrete sectors, as shown in Figure 4. The $\Delta\alpha$ denotes the angle of each sector plane, so the number of sectors $M = 2\pi / \Delta\alpha$.

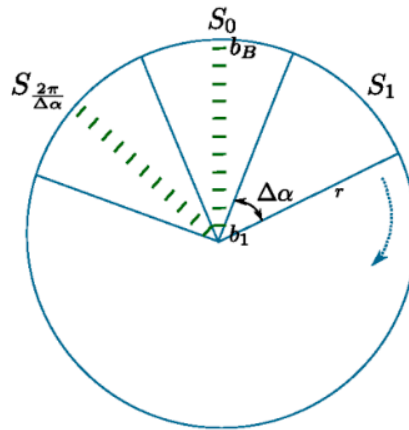


Figure 4. Partitioning the 3D space into segments of equal size.

- S_i represents each sector, where $1 \leq i \leq M$, so that each point can be classified into a sector plane according to its projection on the x - o - y plane, expressed as a segment (p_i):

$$\text{segment}(p_i) = \frac{\text{atan2}(y_i, x_i)}{\Delta\alpha}, \quad (2)$$

where $\text{atan2}(y_i, x_i)$ represents the angle within $[0, 2\pi)$ between the positive direction of x -axis and x - o - y plane, y_i representing y -value of p_i , x_i representing x -value of p_i , and $\Delta\alpha$ representing the angle of each sector plane.

We denote the set of all points mapped to the same segment s by P_s :

$$P_s = \{p_i \in P | \text{segment}(p_i) = s\}, \quad (3)$$

Define a mapping of all points P_s of the same segment to one of many bins b_j^s , $j = 1 \dots B$ discretizing the range component of the points, while the superscript s denotes the sector that the bin belongs to. The minimum or maximum range that a bin covers is expressed respectively by r_j^{\min} and r_j^{\max} . Obviously, a point $p_i \in P_s$ maps to bin b_j^s :

$$r_j^{\min} \leq \sqrt{x_i^2 + y_i^2} \leq r_j^{\max}, \quad (4)$$

The $P_{b_j^s}$ is denoted by the set of all points mapping to b_j^s . Given a set of $P_{b_j^s}$ of 3D points mapped to the same bin, a new set of 2D points $P'_{b_j^s}$ is simply defined as:

$$P'_{b_j^s} = \{p'_i | p_i \in P_{b_j^s}\}, \quad (5)$$

where p_i representing any point in 3D space, $p_i = (x_i, y_i, z_i)^T$, $P'_i = (\sqrt{x_i^2 + y_i^2}, z_i)^T$.

All points have been mapped to a segment and a corresponding segment bin. With the above mapping method of 3D point clouds data set, sorting from small to large by distance, the processed 3D

space point set is partially of order. A prototype point $p'_{b_j^s}$ is calculated for every non-empty bin points $P'_{b_j^s}$ whose point with lowest z -coordinate and most likely belonging to the ground plane.

2.3.2. Fast Ground Segmentation

On the basis of the above data mapping method of 3D point clouds data set, the ground model in a sectorial area S can be expressed as a set L_s of a line segment shown in Figure 5.

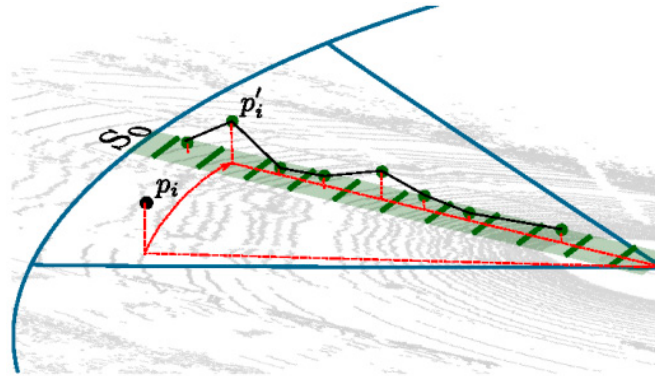


Figure 5. Mapping of 3D point p to a bin of the corresponding segment and resulting mapped point p'_i .

Calculate the distance between a point and the line L as shown in Figure 6. Calculate the two straight lines L_a and L_b , which pass through both ends of line segment AB , and perpendicular to line segment L . Then determine whether the point P is between the straight lines L_a and L_b (such as P_1 point) or outside the two lines (such as P_2 point, P_3 point). If the point is at the position P_1 , the distance L and P_1 can be directly calculated. If the point is at the position P_2 , then the distance between P_2 and point A is calculated. In the same way, if at point P_3 , the distance between P_3 and L is equal to the distance between P_3 and point B . The process of extracting lines for a segment is expressed in Algorithm 1 as follows.

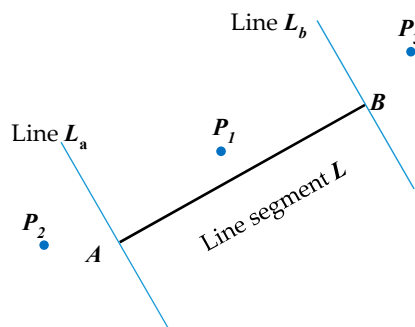


Figure 6. The distance between points and line segment L .

The thresholds mentioned in the algorithm are tested in the experiment with the settings shown in Table 1, while the threshold T_{ground} determines whether the point belongs to a ground point. It remains to formulate necessary conditions for a line $y = mx + b$ to be considered part of the ground plane:

1. The line's slope m must not exceed a certain threshold T_m .
2. The line's absolute y -intercept b must not exceed a certain threshold T_b .
3. The root mean square error of the fit must not exceed a certain threshold T_{RMSE} .
4. The distance of the first point of a line to the line previously fitted must not exceed T_{dprev} , enforcing smooth transitions between pairs of successive lines.

With the above bins and segments calculation, and the experimental thresholds trial selection, the fast ground segmentation results are shown in Figure 7, where the most of ground information can be effectively cancelled.

Algorithm 1. Extraction of lines for one segment S_s

```

1:  $L_s = \emptyset, c = 0, p_1 = \emptyset$ 
   %  $L_s$  denotes a set of line set in segment  $S$ ,  $c$  denotes count of loop,  $p_1$  denotes a point of line
2: for  $j = 0$  in MAX_B do           % for each bin
3:   if  $p'_{b_j} \neq \emptyset$  then     % if there is a point mapping in  $b_j^s$ 
4:     if  $|p_1| \geq 2$  then           % if the number of points in  $L_s$  bigger than two
5:        $(m_c, b_c) = \text{fitline}(p_1 \cup p'_{b_j})$  % line fit to get  $m_c, b_c$  of  $L$ 
6:       if  $|m_c| \leq T_m \wedge (m_c > T_{m_{small}} \vee |b_c| \leq T_b) \wedge \text{fitError}(m_c, b_c, p_1 \cup p'_{b_j}) < T_{RMSE}$  then
           % if match condition of thresholds
7:          $p_1 \leftarrow p_1 \cup p'_{b_j}$  % add  $p'_{b_j}$  to  $p_1$ 
8:       else
9:          $(m_c, b_c) \leftarrow \text{fitline}(p_1)$  % line fit to get  $m_c, b_c$  of  $L$ 
10:         $L_s \leftarrow L_s \cup \{(m_c, b_c)\}$ 
11:         $p_1 \leftarrow \emptyset$  % clear  $p_1$ 
12:         $c \leftarrow c + 1$  % next line segment
13:         $j \leftarrow j - 1$  % next distance
14:      else % if the number of points in  $L_s$  smaller than two
15:        if  $c = 0 \vee (p_1 \neq \emptyset) \vee \text{distpointline}(p'_{b_j}, m_{c-1}, b_{c-1}) \leq T_{dprev}$  then
           % if the first point or the distance of point and line match thresholds
16:           $p_1 = p_1 \cup p'_{b_j}$  % add  $p'_{b_j}$  to  $p_1$ 

```

Table 1. Threshold Setting.

T_m	T_b	T_{RMSE}	T_{dprev}	T_{ground}
0.08	0.2	1	0.04	0.08

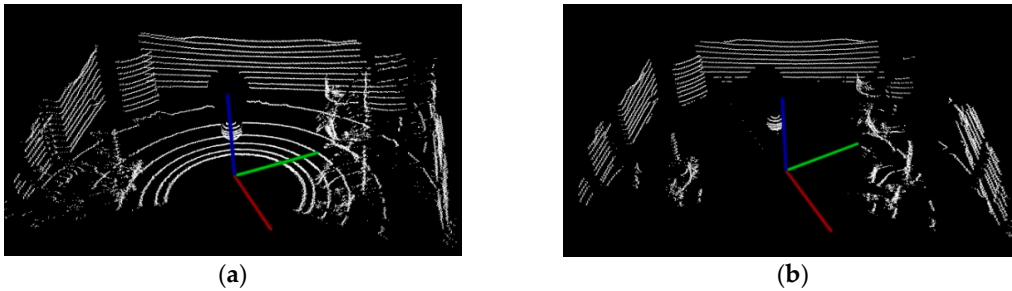


Figure 7. Results of fast ground segmentation: (a) Before segmentation; (b) After segmentation.

2.4. Object Segmentation

On the basis of fast ground segmentation to cancel the noises, the Euclidean clustering segmentation algorithm is used to segment the sphere target, and its point-cloud data needs to be trained to recognize the target.

2.4.1. Euclidean Clustering of Point Clouds

The Euclidean clustering algorithm is used to segment other disperse irrelevant background sorting point clouds data of same similarity within certain threshold. The basic idea is dividing n points into m classes randomly at first, and then making the points in each class have comparatively

high similarity while the points in different classes have comparatively low similarity. Then we calculate the Euclidean distance between clusters. The two clusters with a minimum distance could be merged into one cluster. We repeat the calculation of distances between clusters, and subsequent merging. With the repeated iteration until the distance between any two clusters is over than the given threshold, or the number of clusters is less than the given number, the segmentation is completed and the target sphere object is obtained. The distance threshold is set as 0.06 according to the number of sphere object's point clouds, the minimum number of points in each class as 100, and the maximum number as 2000. The results of Euclidean clustering are shown in Figure 8, where the surrounding of the target sphere can be clearly clustered. Figure 8e shows the basic contour of the target sphere. With the Euclidean clustering, we could extract and match the moving sphere during the indoor motion tracking process.

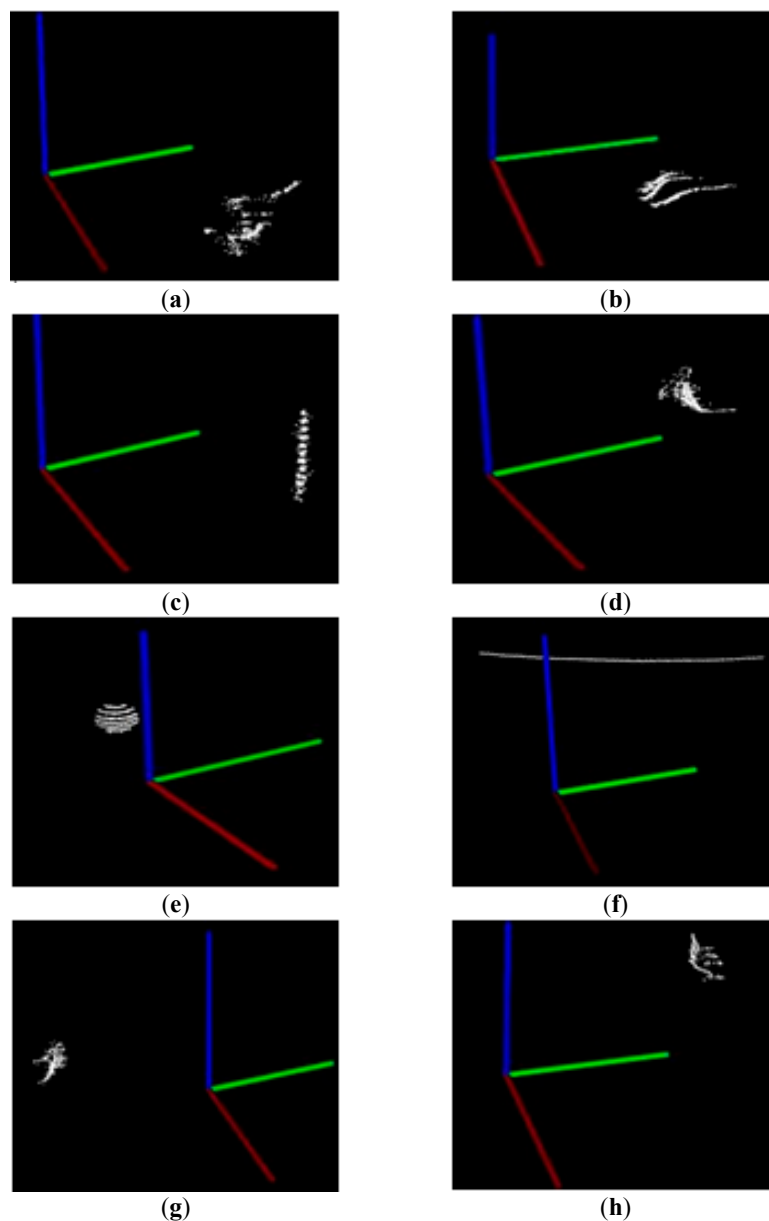


Figure 8. The segmentation results of Euclidean clustering: (a) The right chair; (b) The legs of the middle chair; (c) The pneumatic pole of the black chair; (d) The back of the middle chair; (e) The target sphere segmented; (f) The edge of window; (g) The part of chair near the air conditioning; (h) The back of blue chair.

2.4.2. VFH Descriptor Extraction

The numbers of clusters have been segmented in the process of Euclidean clustering. Subsequently the feature extraction descriptor for each cluster is employed to match the target sphere with Fast Library with Approximate Nearest Neighbors (FLANN). The feature description here describes the geometry and topology of the local or global of point clouds data sets, which can be easily understood as a point-cloud feature. The feature description of point-clouds generally consists of local and global feature descriptions. The local features describe the local geometry and shape characteristics of the point-cloud data, while the global features describe the global topological structure of the point clouds. Only for the motion tracking at the whole indoor environment to distinguish different poses, the global feature descriptor of Viewpoint Feature Histograms (VFH) is employed to estimate the feature of clusters and to extract the target. The visualization of sphere object and its VFH are shown in Figure 9.

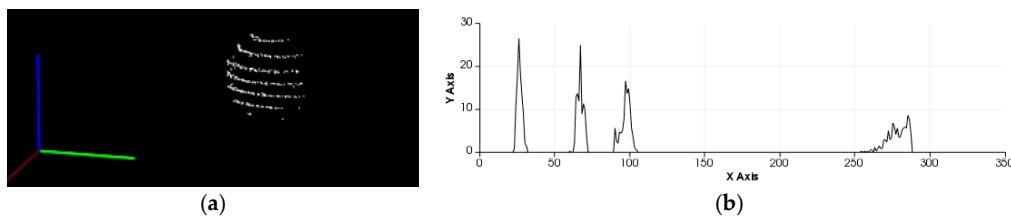


Figure 9. Results of VFH descriptor extraction: (a) Basketball's point clouds visualization; (b) VFH of the basketball.

2.4.3. Feature Match of FLANN

The point clouds feature model library is constructed with the FLANN and the extraction steps are listed as follows:

- (1) Acquire the point clouds data sets using the LiDAR sensor at different distance between the sphere object and sensor, and then extract VFH features for each point clouds model.
- (2) Load the above VFH features into memories and convert the data into matrix format.
- (3) Create the $k-d$ (k -dimensional) tree with the converted matrix data, and save the index of $k-d$ tree for the direct search match.
- (4) Input the VFH feature and the index of $k-d$, and search the nearest neighbor along the $k-d$ tree for the input data.
- (5) Achieve the target point clouds if the difference between the searching results and VFH is less than the given threshold.

3. Tracking

3.1. Kalman Filtering

For the motion tracking of a moving sphere, the KF [29] provides a highly computable method in the recursive way to estimate the state of the process and minimize the estimated mean square error. The state and measurement equations are used to describe a dynamic system. The state vector s_k of the system of time moment k is determined by both the state s_{k-1} at time moment $k-1$ and the observed noise. The measurement vector m_k is also determined by these two, which is by the observation function of state vector s_k at time moment k and the noise. The target motion tracking process with KF is shown in Figure 10. The state variables are the positions and velocities of the sphere object in the X , Y , and Z coordinate, expressed as the matrix $[x, y, z, v_x, v_y, v_z]^T$ and the observed variable z is the objects' position data real-time sensed by LiDAR sensor and real-time processed, denoted as $[x, y, z]^T$. Here, the state function is expressed as Equation (6):

$$s_k = As_{k-1} + w_{k-1}, \quad (6)$$

and the measurement function is given by Equation (7):

$$m_k = Hm_k + v_k \quad (7)$$

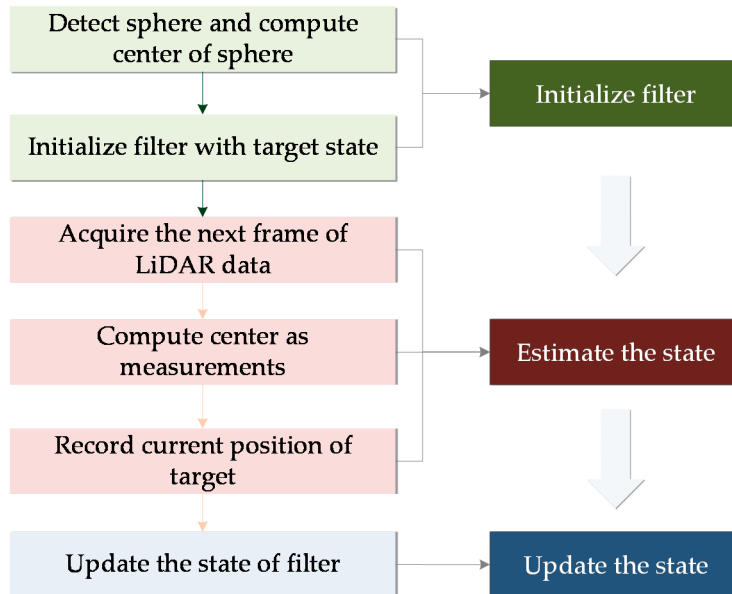


Figure 10. Sphere motion tracking process with KF based on LiDAR.

The random variables w_{k-1} and v_k represent the process and the measurement noise respectively, denoting the noises and disturbances of the moving sensing data. They are assumed to be independent of each other, and with normal probability distributions:

$$p(w) \sim N(0, Q) \quad (8)$$

$$p(v) \sim N(0, R) \quad (9)$$

The matrix Q denotes the process noise covariance and R the measurement noise covariance. Here assumed that Q and R are redefined as constant shown as follows:

$$Q = \begin{bmatrix} 0.05 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.05 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.05 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.05 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.05 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.05 \end{bmatrix} \quad (10)$$

$$R = \begin{bmatrix} 0.05 & 0 & 0 \\ 0 & 0.05 & 0 \\ 0 & 0 & 0.05 \end{bmatrix} \quad (11)$$

Define the state variable s_k as a six-dimensional vector shown in Equation (12):

$$s_k = [x_k, y_k, z_k, v_{x,k}, v_{y,k}, v_{z,k}]^T, \quad (12)$$

where x_k, y_k, z_k are respectively the coordinate value of the center sphere object in the x, y, z coordinate system, and $v_{x,k}, v_{y,k}, v_{z,k}$ is respectively the speed of the center coordinates in the x, y, z direction.

The time interval dt between the two frames of LiDAR is only one second, which is relatively short, and the motion could be considered as a uniform motion, so the state transition matrix A is expressed as:

$$A = \begin{bmatrix} 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

The measurement vector m_k is used to observe the center position of the moving spherical objects shown in Equation (14):

$$m_k = [x_k, y_k, z_k]^T, \quad (14)$$

and the corresponding observation matrix is as follows:

$$H = \begin{bmatrix} 1 & 0 & 0 & -dt & 0 & 0 \\ 0 & 1 & 0 & 0 & -dt & 0 \\ 0 & 0 & 1 & 0 & 0 & -dt \end{bmatrix}. \quad (15)$$

The variable \hat{s}_k^- ($-$ represents priori, and $\hat{\cdot}$ represents the estimate) denotes k -th priori state estimation when the k -th preceding state is known, and \hat{s}_k is the known posterior state estimation of measurement variable z_k at the k -th period.

The following Equation (16) predicts the current state value with the result of the previous finest state:

$$\hat{s}_k^- = A\hat{s}_{k-1} + w, \quad (16)$$

where \hat{s}_k^- is the a priori state estimate at step k given knowledge of the process prior to step k . A is the state transition matrix of the system shown in Equation (13), \hat{s}_{k-1} is the optimal estimate at time moment $t - 1$, and w is the system noise following Gaussian distribution.

The uncertainty of each moment is represented by the covariance matrix P , and the update formula is expressed in Equation (17):

$$P_k^- = AP_{t-1}A^T + Q, \quad (17)$$

where P_k^- is the covariance of \hat{x}_k^- , P_{t-1} is the covariance of \hat{x}_{k-1} , and Q is the covariance of the random signals w_{k-1} and v_k . The prediction Equations (16) and (17) update time t .

We define a priori and a posteriori estimate error as Equations (18) and (19), respectively;

$$e_k^- \equiv x_k - \hat{x}_k^-, \quad (18)$$

$$e_k \equiv x_k - \hat{x}_k, \quad (19)$$

Then, the a priori estimated error covariance is given by Equation (20):

$$P_k^- = E[e_k^- e_k^{-T}], \quad (20)$$

and the s posteriori estimate error covariance is:

$$P_k = E[e_k e_k^T], \quad (21)$$

The initial state P_0 of covariance matrix P_k is redefined as:

$$P_0 = \begin{bmatrix} 10 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 10 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 10 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 10 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 10 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 10 \end{bmatrix} \quad (22)$$

The posteriori state estimate \hat{s}_k as a linear combination of an a priori estimate \hat{s}_k^- and a weighted difference between an actual measurement m_k and a measurement prediction $H\hat{s}_k^-$, calculated as shown in Equation (23):

$$\hat{s}_k = \hat{s}_k^- + K_k(m_k - H\hat{s}_k^-), \quad (23)$$

where the matrix K_k is the gain that minimizes the posteriori error covariance. The difference $(m_k - H\hat{s}_k^-)$ is the measurement innovation (residual) that reflects the discrepancy between the predicted measurement $H\hat{s}_k^-$ and the actual measurement z_k .

To accomplish minimization, the Equations (18) and (23) are firstly substituted into Equation (19), with a unit matrix I , then we get Equation (24):

$$e_k = [I - K_k H]e_k^- - K_k v_k, \quad (24)$$

Substituting the above Equation (24) into Equation (21), then we get:

$$P_k = (I - K_k H)P_k^- (I - K_k H)^T + K_k R K_k^T, \quad (25)$$

Obtaining the indicated expectations, we take the derivative of the tracking result with respect to K_k . Set the result to zero, and then solve for K_k as below:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} = \frac{P_k^- H^T}{H P_k^- H^T + R}, \quad (26)$$

With the equations above, the smaller the observed noise covariance R , the larger the gain K_k . The smaller the covariance P_k^- , the smaller the gain K_k . The Kalman gain K_k works on two aspects, firstly, it weighs the size of the a priori estimated error of the covariance P_k^- and the observed noise covariance matrix R to determine the more convincing model between the prediction and observation model; secondly, it transforms the representation form of the residuals from the observation domain to the state domain.

To make the KF rung down till the end of the whole system running process, the covariance \hat{x}_k at state t needs to be updated as follows:

$$P_k = (I - K_k H)P_k^-, \quad (27)$$

After the time updating calculation of the Equations (14) and (19) and the measurement updating Equations (20), (23) and (24), the whole Kalman tracking process repeats again. A posteriori estimation of Equation (20) obtained from the previous calculation of Equation (18) is taken as the a priori estimation of Equation (24) of the next computation. The whole process of KF is shown in Algorithm 2 as follows.

Algorithm 2. Kalman Filter

Input: m_k , % object position for time step t from sensor
Output: \hat{s}_k , % a position estimation of object

1: **initialize** $t, \hat{s}_{k-1}, A, P, Q, R$ % t represents prediction time moment, \hat{s}_{k-1} is the known posterior
 % state estimation at time moment $k - 1$; A represents state transition matrix;
 % P represents the covariance matrix, Q denotes covariance of the random
 % signals, and R is the matrix of observation noise covariance.

2: **if** filterStop = false **then** % end with convergence of click the 'stop' button.

3: $\hat{s}_k^- \leftarrow A\hat{s}_{k-1}$ % calculate predicting position estimation, according to Equation (16)

4: $P \leftarrow APA^T + Q$ % calculate priori covariance matrix, according to Equation (17)

5: $K \leftarrow PC^T(CPC^T + R)^{-1}$ % calculate Kalman Gain matrix, according to Equation (26)

6: $\hat{s}_k \leftarrow \hat{s}_k^- + K(m_k - C\hat{s}_k^-)$ % calculate optimal estimation value, according to Equation (23)

7: $P \leftarrow (I - KC)P$ % calculate \hat{x}_k covariance, according to Equation (27), I denotes unit matrix

8: $t \leftarrow t + 1$

9: **end if**

3.2. Particle Filtering

Particle filtering is a non-parametric Monte Carlo method used to simulate the realization of the recursive Bayesian filter, which is applicable to any state space model for the non-linear non-Gaussian case, and its accuracy can reach the optimal estimate. Filtered particles are possibilities to describe the target state. The purpose of filtering is the most probable state of the filtered target. In the Bayesian estimation theory, the current state of the target is estimated using the previous state and the current measured value. The arbitrary probability distribution $p(x_k)$ can be Monte Carlo approximated using the discrete particle set as follows:

$$p(x_k|y_{1:k}) \approx \sum_{i=1}^{N_k} w_k^{(i)} \delta(x_k - x_k^{(i)}), \quad (28)$$

where $x_k^{(i)}$, $w_k^{(i)}$, N_k are respectively expressed as particle state, weight and total number under k time, where δ is Dirac's delta function.

The most basic and common PF implementation framework is Sequential Importance Sampling and Resampling (SISR) or Sampling Importance Resampling (SIR) filter, and the algorithm is shown below:

Step 1: For $i = 1, 2, \dots, N$, Initializing the particle set, $k = 0$:

Generating the sampled particles $\{x_0^{(i)}\}_{i=1}^N$ based on the priori distribution $p(x_0)$;

Step 2: For $k = 1, 2, \dots, N$, Executing the follow steps circularly:

Sequential importance sampling: for $i = 1, 2, \dots, N$, generating the sampled particles $\{\tilde{x}_k^{(i)}\}_{i=1}^N$ from the importance probability density, then calculating the particle weights, finally normalizing the weights so that the sum of the weights of the particles is 1;

Resampling: resampling the particles set $\{\tilde{x}_k^{(i)}, \tilde{w}_k^{(i)}\}$, and the resampled set is $\{x_k^{(i)}, 1/N\}$;

Printing: calculating the estimated state value: $\hat{x}_k = \sum_{i=1}^N \tilde{x}_k^{(i)}, \tilde{w}_k^{(i)}$.

Sequential importance sampling is the basis of particle filtering, which applies the sequential analysis method in statistics to the Monte Carlo method, so as to realize the recursive estimation of the probability density of posterior filtering. Assumed that the importance probability density function $q(x_{0:k}|y_{1:k})$ can be decomposed into:

$$q(x_{0:k}|y_{1:k}) = q(x_{0:k-1}|y_{1:k-1})q(x_k|x_{0:k-1}, y_{1:k}), \quad (29)$$

Let the system state be a Markov process, and the given system state is independent of each observation so that there is:

$$p(x_{0:k}) = p(x_0) \prod_{i=1}^k p(x_i|x_{i-1}), \quad (30)$$

$$p(y_{1:k}|x_{1:k}) = \prod_{i=1}^k p(y_i|x_i), \quad (31)$$

The recursive form of the posterior probability density function can be expressed as:

$$\begin{aligned} p(x_{0:k}|Y_k) &= \frac{p(y_k|x_{0:k}, Y_{k-1})p(x_{0:k}|Y_{k-1})}{p(y_k|Y_{k-1})} \\ &= \frac{p(y_k|x_{0:k}, Y_{k-1})p(x_k|x_{0:k-1}, Y_{k-1})p(x_{0:k}|Y_{k-1})}{p(y_k|Y_{k-1})} \\ &= \frac{p(y_k|x_k)p(x_k|x_{k-1})p(x_{0:k-1}|Y_{k-1})}{p(y_k|Y_{k-1})} \end{aligned} \quad (32)$$

In the update phase, the particles' weights are recalculated according to the likelihood function $p(x_{0:k}|Y_k)$:

$$\begin{aligned} w_k^{(i)} &\propto \frac{p(x_k^{(i)}|Y_k)}{q(x_k^{(i)}|Y_k)}, \\ &= \frac{p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})p(x_{0:k-1}^{(i)}|Y_{k-1})}{q(x_k^{(i)}|x_{0:k-1}^{(i)}, Y_k)q(x_{0:k-1}^{(i)}|Y_{k-1})} \\ &= w_{k-1}^{(i)} \frac{p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_k^{(i)}|x_{0:k-1}^{(i)}, Y_k)} \end{aligned} \quad (33)$$

In general, it is necessary to normalize the weight of the particle:

$$\tilde{w}_k^{(i)} = \frac{w_k^{(i)}}{\sum_{i=1}^N w_k^{(i)}}, \quad (34)$$

This results in an approximate representation of the posterior probability density function expressed by Equation (32). In practical application, the use of too many samples will result in a sharp increase in the computational complexity and the deterioration of the performance of the particle filter. However, it is very difficult to correctly approximate the posterior probability with a small amount of sampling, and the resampling process may also lead to particle deficiency. Therefore, it is necessary to determine the appropriate sampling quantity and improve the efficiency of sampling according to the state of the system, under the condition of ensuring the diversity of the particles. In this paper, adaptive particle filter based on Kullback-Leibler Distance (KLD) sampling proposed by Fox [30] is adopted to resample the particles.

The core idea of the KLD sampling is that in each iteration of the particle filter, using the probability $1 - \delta$ to make the error between the true posterior probability and the estimated probability density based on the sample less than ϵ :

$$n = \frac{1}{2\epsilon} \chi_{k-1, 1-\delta}^2 \quad (35)$$

where δ is set as 0.99 and ϵ as 0.2. So that the number of resampled samples is determined. The error is determined by calculating the KLD. The KLD is used to represent the approximation error between the two probability distributions p and q :

$$K(p, q) = \sum_x p(x) \lg \frac{p(x)}{q(x)}, \quad (36)$$

In the resampling, the smaller particles are neglected and the larger particles are copied. The number of particles in the resampling is determined by KLD sampling in the process of particle duplication, and the number of particles of the next importance is determined, adjusting the number of particles on-line and reducing the computational complexity. The process of KLD sampling is shown in Algorithm 3.

Algorithm 3. KLD sampling algorithm

Input: $s_{t-1} = \{\langle x_{t-1}^{(i)}, w_{t-1}^{(i)} \rangle | i = 1, \dots, n\}$, observations z_t , limits ε and δ ;
Output: S_t

- 1: $S_t := \emptyset, n = 0, k = 0, \alpha = 0$ % initializing
- 2: **do** % generating samples
- 3: sampling from discrete distributions under the weight of known s_{t-1} , the sequence is $j(n)$
- 4: sampling $x_t^{(n)}$ from $p(x_t | x_{t-1})$ using $x_{t-1}^{(j(n))}$
- 5: $w_t^{(n)} := p(z_t | x_t^{(n)})$ % calculate the importance weights
- 6: $\alpha := \alpha + w_t^{(n)}$ % update the normalization factor
- 7: $S_t := S_t \cup \{\langle x_t^{(n)}, w_t^{(n)} \rangle\}$ % insert the sample into the sample set
- 8: **if** $(x_t^{(n)})$ fall in *bin b* **then** % update the number of *bins*
- 9: $k := k + 1$
- 10: *b* := *non - empty*
- 11: $n := n + 1$ % update the number of generated samples
- 12: **while** $(n < \frac{1}{2\varepsilon} \chi_{k-1, 1-\delta}^2)$ % stop when come to the limits K-L with Equation (35)
- 13: **for** $i = 1, \dots, n$ **do** % normalize importance weights
- 14: $w_t^{(i)} = w_t^{(i)} / \alpha$

4. Experimentation and Discussions

In order to validate the robustness and real-time performance of the two tracking algorithms (KF and adaptive PF), some experiments were carried out on the moving spherical target with or without occlusion, obstacles, different speeds and different trajectories. We tied the ball with a rope and pulled the string to move the ball.

4.1. Target Tracking with Occlusion and Obstacle in the Environment

In target tracking processes, occlusion is a very common phenomenon. When the target is blocked, the valid information will be reduced and the tracking difficulty will be increased. Considering the partial occlusion of the spherical target with a long piece of wood, as shown in Figure 11a, the experimental results are shown in Figures 12 and 13 with two different tracking algorithms, respectively.



Figure 11. Experimental scenes: (a) Occlusion scene; (b) Obstacle scene.

In Figure 12, when the target is partially occluded, the KF clusters the wood point clouds together as the next frame's measurement, and some tracking loss occurs. However, in Figure 13, the PF algorithm can track the object effectively where the occluded wood board does not cluster with the target.

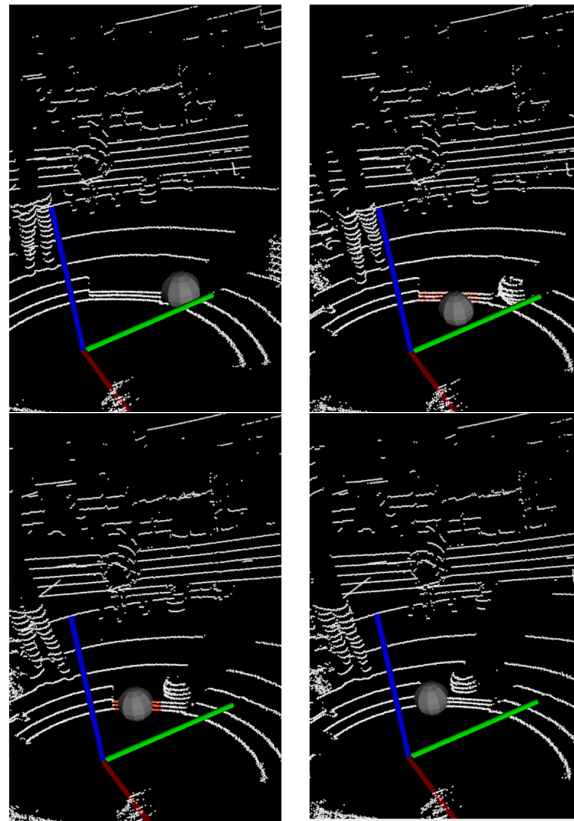


Figure 12. The tracking results with KF in occlusion scenes.

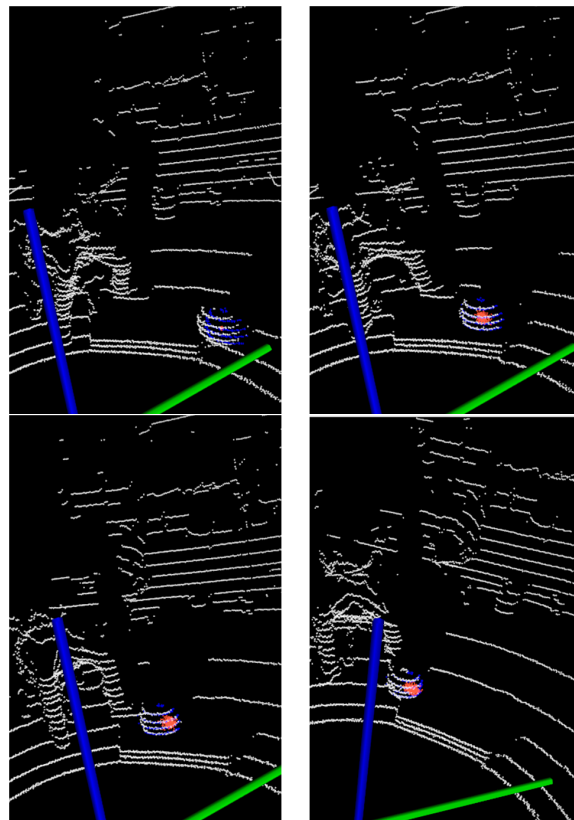


Figure 13. The tracking results with adaptive PF in occlusion scenes.

Obstacles in front of the moving target are common in practical applications, and when multiple targets appear in motion, they may interfere with each other. The carton box as an obstacle is shown in Figure 11b. The experimental results are shown in Figure 14.

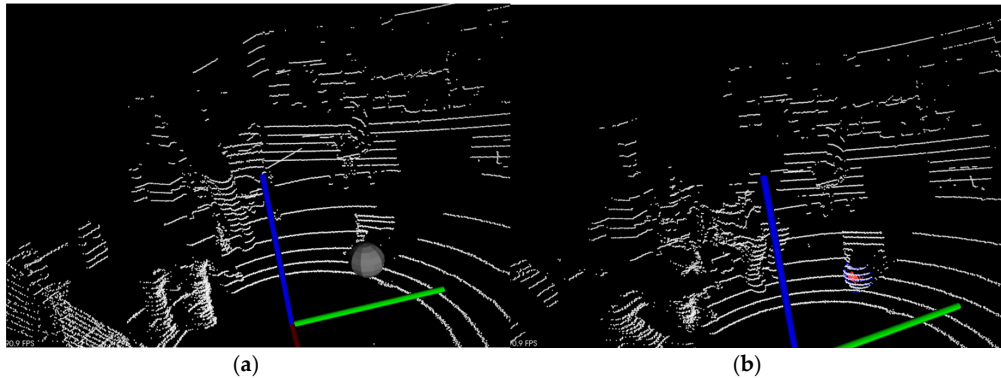


Figure 14. The tracking results in interference scenes: (a) Results with Kalman filter; (b) Results with adaptive particle filter.

The two methods can track the spherical target continuously in the case of an obstacle. In Figure 14, the KF is susceptible to near point-cloud in the tracking process, and the PF has strong robustness for the motion tracking.

4.2. Target Tracking at Different Moving Speeds

Whether the tracker can track moving targets at different speeds is also an important indicator of the performance. Due to the limitations of our laboratory environment and the point-cloud density of LiDAR equipment, the basketball position swings easily in the process of manually moving the basketball, especially at low speed, and a uniform speed can hardly be obtained. At the experimental process, the observed speed of movement of the basketball is 0.054 m/s at its low speed, and 0.125 m/s at its high speed. In Figure 15, the continuity of low-speed trajectory expressed by the light blue dots shows that the KF is more suitable for tracking at relative lower speeds. The continuity and density of high speed trajectory expressed by the red and brown fork shows the adaptive PF has better tracking performance. Basically, with more trials of different speed experiments, the adaptive PF has better tracking effects for the different moving speed than the KF.

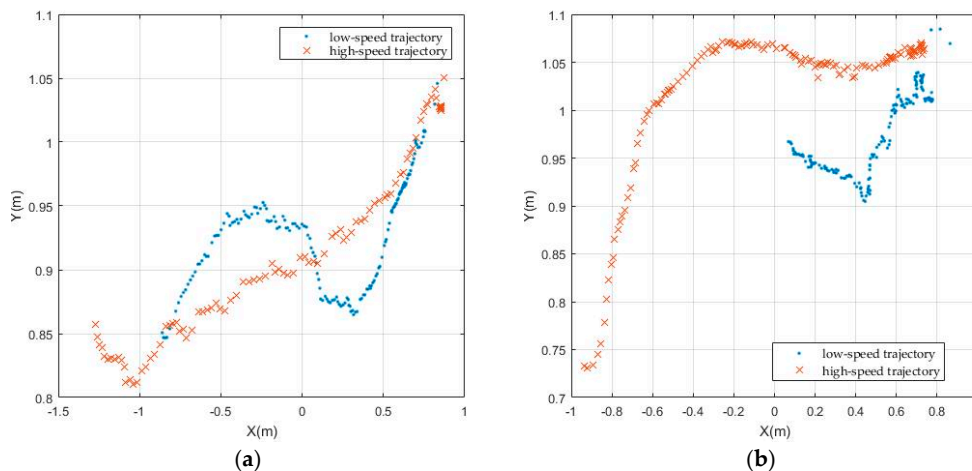


Figure 15. The tracking results at different moving speeds: (a) Results with KF; (b) Results with adaptive PF.

4.3. Target Tracking in Different Motion Trajectories

In an actual situation, the target may move in a variety of trajectories. The spherical target's real-time tracking is tested in three kinds of trajectories including straight line, curve and three-dimensional trajectory. The linear motion trajectory and error analysis are shown in Figure 16. Figure 16a,b show the actual trajectory and the estimated value of the spherical target in the rectilinear movement under the Kalman filter and the adaptive particle filter, respectively. The results indicate that the target moves as a curve in the same situation are shown in Figure 17. Obviously, the trajectory with the Kalman filter is smoother. However, the effect of PF tracking is better. The error comparison is shown in Figures 16c and 17c. Errors and fluctuations of the KF are greater than the adaptive PF.

The results of the trajectory tracking of the spherical target in the three-dimensional space are shown in Figure 18. It is easy to see that the KF and the PF can basically track the moving target in three-dimensional space, but fluctuate in the Z-axis direction.

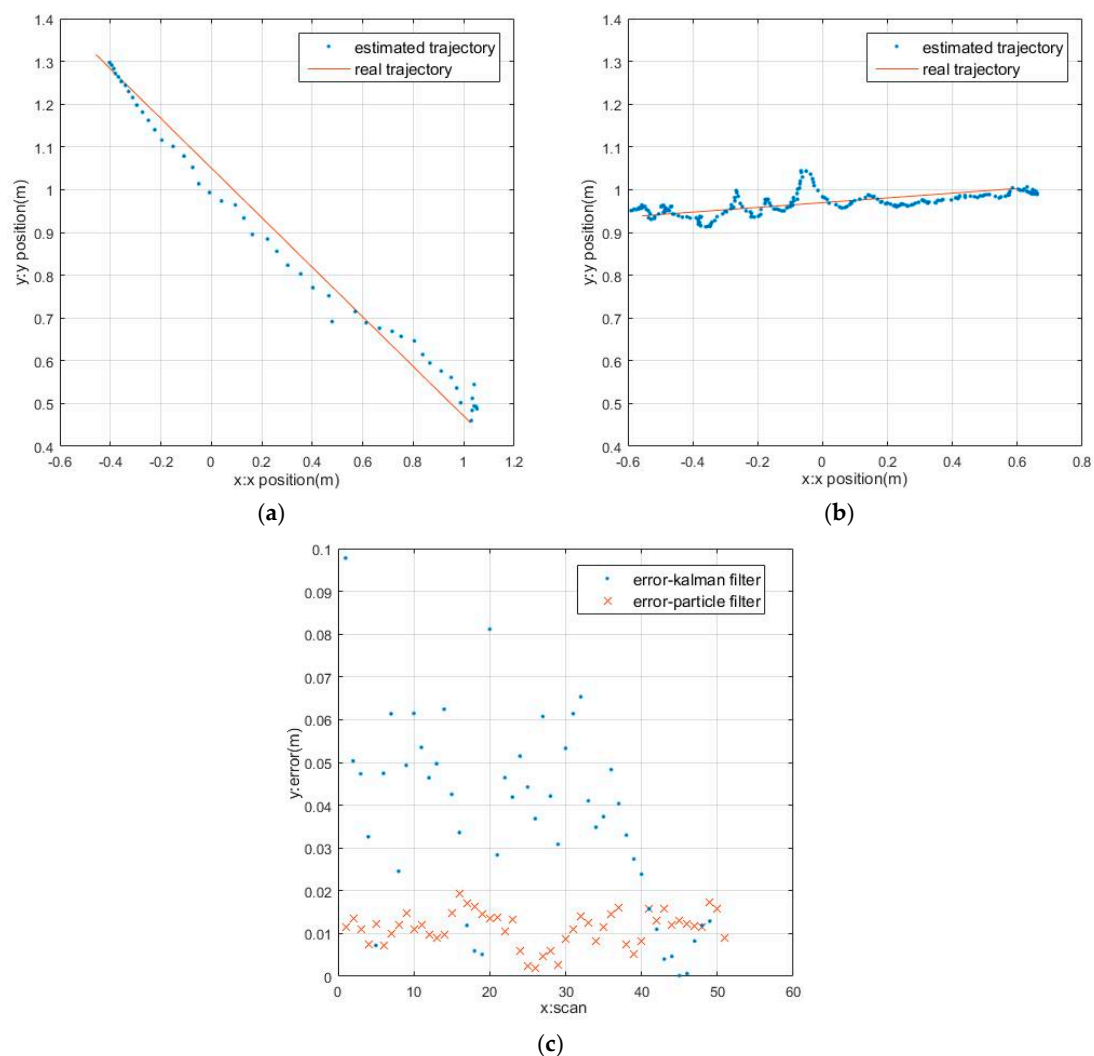


Figure 16. The tracking results in rectilinear motion: (a) Results with KF; (b) Results with adaptive PF; (c) The error of KF and PF.

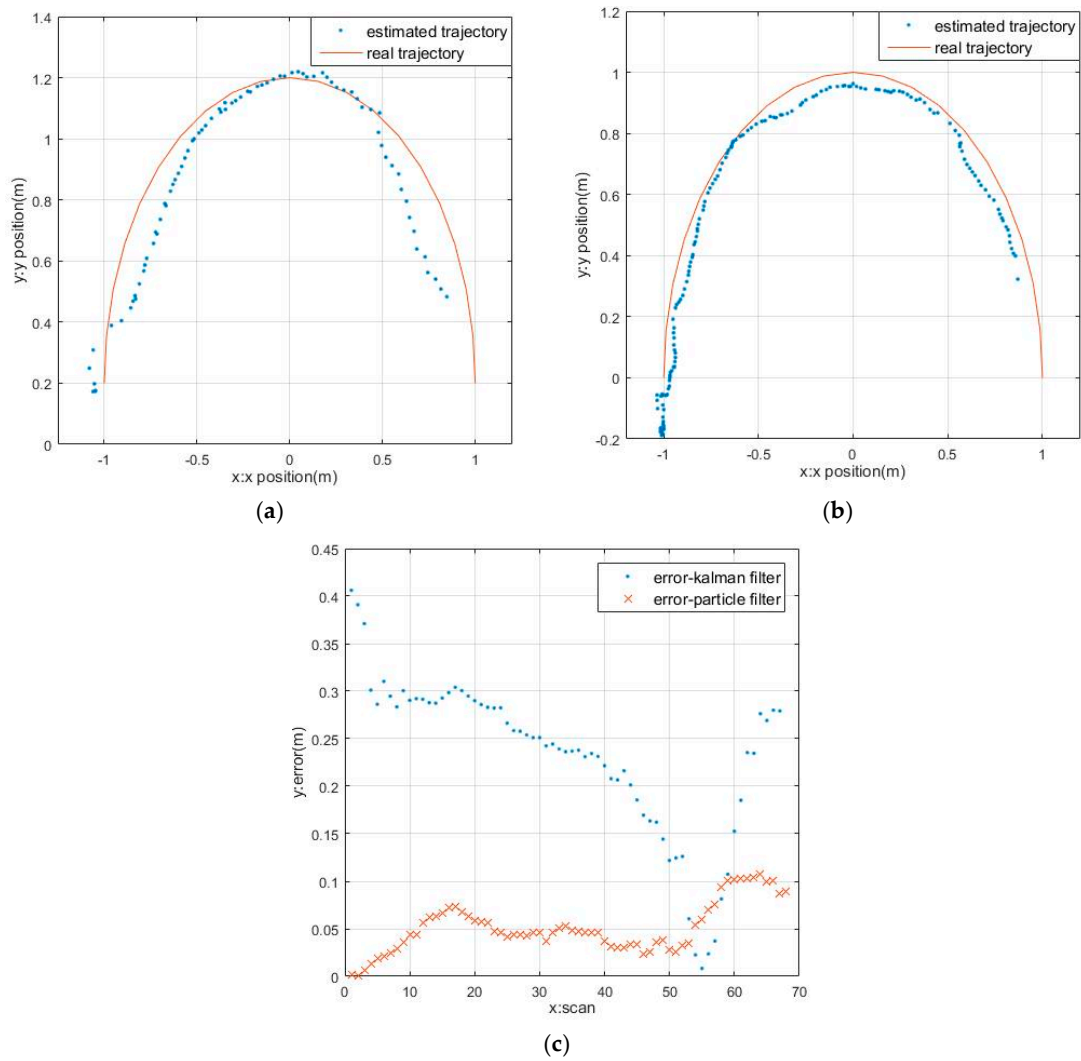


Figure 17. The tracking results in curvilinear motion: (a) Results with KF; (b) Results with adaptive PF; (c) The error of KF and PF.

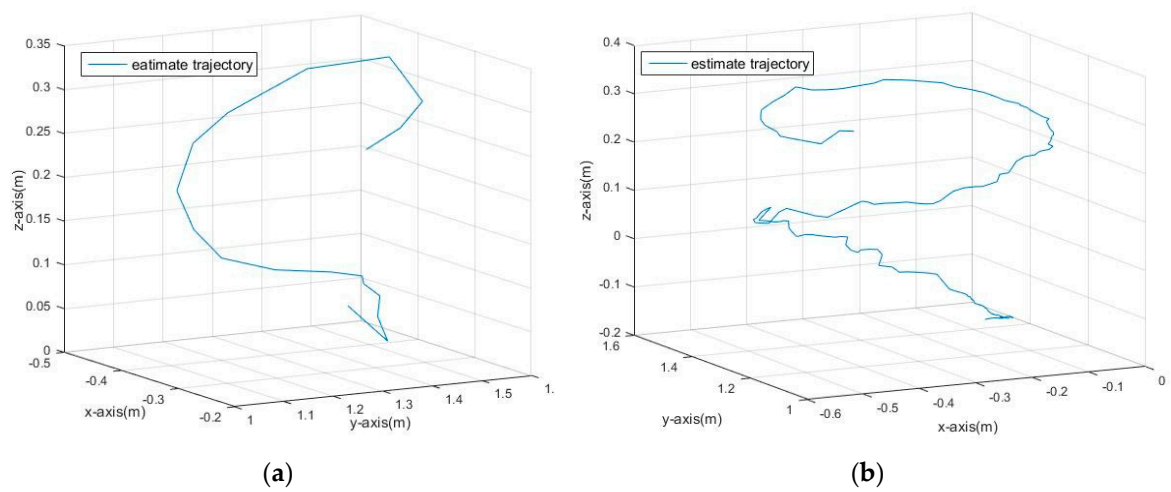


Figure 18. The tracking results in 3D space: (a) Results with KF; (b) Results with adaptive PF.

5. Conclusions

This paper has proposed a systematic real-time detection and tracking method for an indoor moving sphere using the VLP-16 3D LiDAR sensor. After a series of preprocessing steps of point-cloud data and global feature extraction, the Kalman filter and the adaptive particle filter method are used to estimate the real-time motion state of a spherical object. With three different kinds of indoor comparison experiments and analysis, the results show that the adaptive PF has better tracking performance. The specific work allows us to put forth two conclusions:

Firstly, the real-time detection of the spherical target is accomplished by acquiring the real-time point-cloud data of moving sphere at indoor, preprocessing with fast ground segmentation algorithm to remove outliers, and ground points clustering with Euclidean cluster algorithm, extracting target feature with VFH to establish model library and matching to detect spherical targets.

Secondly, the KF is used to real-time track the object, and the object position is estimated sequentially by real-time acquisition of the measured value, prediction and correction, while the adaptive PF is used to track the target, and the state of the target is estimated by sampling, calculating the weight and resampling. The efficiency of KF and adaptive PF in 3D lidar tracking is verified by indoor basketball tracking experiments, with a moving spherical object with or without occlusion and obstacles, respectively, at different speeds and over different trajectories. The experimental results show that adaptive PF has a small tracking error and strong robustness.

The motion tracking of a dynamic environment is one of the key components for intelligent agricultural harvesters to operate in real-world conditions. We will continue to exploit the 3D semantic perception with transfer learning and real-time location method of natural fruits for a better tracking performance.

Acknowledgments: This research was fully supported by Major Pilot Projects of the Agro-Tech Extension and Service in Shaanxi (No. 2016XXPT-00), Fundamental Research Funds for the Central Universities, Northwest A&F University (No. QN2013052).

Author Contributions: Lvwen Huang contributed the whole research, edited the English language and rewrote this paper. Siyuan Chen designed and implemented the tracking part of system and wrote this manuscript. Jianfeng Zhang edited the English expressing. Bang Cheng designed the detection part of the system and wrote the detection of this manuscript. Mingqing Liu assisted gathering the experimental data during the experiments and manuscript.

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

References

1. Eum, J.; Berhanu, E.; Oh, S. Unmanned aircraft platform based real-time lidar data processing architecture for real-time detection information. *KIISE Trans. Comput. Pract.* **2015**, *21*, 745–750. [[CrossRef](#)]
2. Dominguez, R.; Alonso, J.; Onieva, E.; Gonzalez, C. A transferable belief model applied to LIDAR perception for autonomous vehicles. *Integr. Comput.-Aided Eng.* **2013**, *20*, 289–302.
3. Shang, E.; An, X.; Wu, T.; Hu, T.; Yuan, Q.; He, H. Lidar based negative obstacle detection for field autonomous land vehicles. *J. Field Robot.* **2016**, *33*, 591–617. [[CrossRef](#)]
4. Feng, D.; Yuan, X. Automatic construction of aerial corridor for navigation of unmanned aircraft systems in class G airspace using LiDAR. In Proceedings of the Conference on Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications XIII, Baltimore, MD, USA, 18–19 April 2016; pp. 1–8.
5. Teobaldelli, M.; Cona, F.; Saulino, L.; Migliozzi, A.; D’Urso, G.; Langella, G.; Manna, P.; Saracino, A. Detection of diversity and stand parameters in Mediterranean forests using leaf-off discrete return LiDAR data. *Remote Sens. Environ.* **2017**, *192*, 126–138. [[CrossRef](#)]
6. Koenig, K.; Hoefle, B.; Haemmerle, M.; Jarmer, T.; Siegmann, B.; Lilienthal, H. Comparative classification analysis of post-harvest growth detection from terrestrial LiDAR point clouds in precision agriculture. *ISPRS J. Photogramm. Remote Sens.* **2015**, *104*, 112–125. [[CrossRef](#)]
7. Andujar, D.; Moreno, H.; Valero, C.; Gerhards, R.; Griepentrog, H.W. Weed-crop discrimination using LiDAR measurements. In Proceedings of the 9th European Conference on Precision Agriculture, Lleida, Spain, 7–11 July 2013; pp. 541–545.

8. Trmal, C.; Pons, F.; Ledoux, P. Flood protection structure detection with Lidar: Examples on French Mediterranean rivers and coastal areas. In Proceedings of the 3rd European Conference on Flood Risk Management (FLOODrisk), Lyon, France, 17–21 October 2016; pp. 1–5.
9. Du, S.; Zhang, Y.; Qin, R.; Yang, Z.; Zou, Z.; Tang, Y.; Fan, C. Building change detection using old aerial images and new LiDAR data. *Remote Sens.* **2016**, *8*, 1030. [[CrossRef](#)]
10. Hwang, S.; Kim, N.; Choi, Y.; Lee, S.; Kweon, I.S. Fast Multiple Objects Detection and Tracking Fusing Color Camera and 3D LIDAR for Intelligent Vehicles. In Proceedings of the 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), Xi'an, China, 19–22 August 2016; pp. 234–239.
11. Wang, H.; Wang, B.; Liu, B.; Meng, X.; Yang, G. Pedestrian recognition and tracking using 3D LiDAR for autonomous vehicle. *Robot. Auton. Syst.* **2017**, *88*, 71–78. [[CrossRef](#)]
12. Song, S.; Xiang, Z.; Liu, J. Object tracking with 3D LIDAR via multi-task sparse learning. In Proceedings of the 2015 IEEE International Conference on Mechatronics and Automation, Beijing, China, 2–5 August 2015; pp. 2603–2608.
13. Guo, L.; Li, L.; Zhao, Y.; Zhao, Z. Pedestrian tracking based on camshift with Kalman prediction for autonomous vehicles. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 120. [[CrossRef](#)]
14. Dewan, A.; Caselitz, T.; Tipaldi, G.D.; Burgard, W. Motion-based detection and tracking in 3D LiDAR Scans. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 4508–4513.
15. Allodi, M.; Broggi, A.; Giaquinto, D.; Patander, M.; Prioletti, A. Machine learning in tracking associations with stereo vision and lidar observations for an autonomous vehicle. In Proceedings of the IEEE Intelligent Vehicles Symposium, Gothenburg, Sweden, 19–22 June 2016; pp. 648–653.
16. Wasik, A.; Ventura, R.; Pereira, J.N.; Lima, P.U.; Martinoli, A. Lidar-based relative position estimation and tracking for multi-robot systems. In Proceedings of the Robot 2015: Second Iberian Robotics Conference, Advances in Robotics, Lisbon, Portugal, 19–21 November 2015; pp. 3–16.
17. Li, Q.; Dai, B.; Fu, H. LIDAR-based dynamic environment modeling and tracking using particles based occupancy grid. In Proceeding of the 2016 IEEE International Conference on Mechatronics and Automation, Harbin, China, 7–10 August 2016; pp. 238–243.
18. Tuncer, M.A.C.; Schulz, D. Integrated object segmentation and tracking for 3D LIDAR data. In Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics, Lisbon, Portugal, 29–31 July 2016; pp. 344–351.
19. Asvadi, A.; Girao, P.; Peixoto, P.; Nunes, U. 3D object tracking using RGB and LIDAR data. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016; pp. 1255–1260.
20. Shelhamer, E.; Long, J.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 640–651. [[CrossRef](#)] [[PubMed](#)]
21. Srilekha, S.; Swamy, G.N.; Krishna, A.A. A novel approach for detection and tracking of vehicles using Kalman filter. In Proceedings of the 7th International Conference on Computational Intelligence and Communication Networks (CICN), Jabalpur, India, 12–14 December 2015; pp. 234–236.
22. Huang, W.; Xie, H.; Shen, C.; Li, J. A robust strong tracking cubature Kalman filter for spacecraft attitude estimation with quaternion constraint. *Acta Astronaut.* **2016**, *121*, 153–163. [[CrossRef](#)]
23. Jain, A.; Krishnamurthy, P.K. Phase noise tracking and compensation in coherent optical systems using Kalman filter. *IEEE Commun. Lett.* **2016**, *20*, 1072–1075. [[CrossRef](#)]
24. Gulalkari, A.V.; Pratama, P.S.; Hoang, G.; Kim, D.H.; Jun, B.H.; Kim, S.B. Object tracking and following six-legged robot system using Kinect camera based on Kalman filter and backstepping controller. *J. Mech. Sci. Technol.* **2015**, *29*, 5425–5436. [[CrossRef](#)]
25. Lim, J.; Yoo, J.H.; Kim, H.J. A mobile robot tracking using Kalman filter-based gaussian process in wireless sensor networks. In Proceedings of the 15th International Conference on Control, Automation and Systems (ICCAS), Busan, Korea, 13–16 October 2015; pp. 609–613.
26. Moon, S.; Park, Y.; Ko, D.W.; Suh, I.H. Multiple kinect sensor fusion for human skeleton tracking using Kalman filtering. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 1–10. [[CrossRef](#)]
27. Rusu, R.B.; Marton, Z.C.; Blodow, N.; Dolha, M.; Beetz, M. Towards 3D point cloud based object maps for household environments. *Robot. Auton. Syst.* **2008**, *56*, 927–941. [[CrossRef](#)]

28. Himmelsbach, M.; Hundelshausen, F.V.; Wuensche, H.J. Fast segmentation of 3D point clouds for ground vehicles. In Proceedings of the 2010 IEEE Intelligent Vehicles Symposium (IV), San Diego, CA, USA, 21–24 June 2010; pp. 560–565.
29. Welch, G.; Bishop, G. *An Introduction to the Kalman Filter*; University of North Carolina at Chapel Hill: Chapel Hill, NC, USA, 2006; Volume 8, pp. 127–132.
30. Fox, D. KLD-sampling: Adaptive particle filters. In Proceedings of the Neural Information Processing Systems: Natural and Synthetic, Vancouver, BC, Canada, 3–8 December 2001; pp. 713–720.



© 2017 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).