LoPF: An Online LiDAR-only Person-Following Framework

Xiangyu Chen[1], Jinhao Liu[1], Jin Wu[2], Chaoqun Wang[1*], and Rui Song[1,3*]

***Abstract*—This paper presents an online LiDAR-only person-following (LoPF) framework, which can be applied in both indoor and outdoor environments with only sparse LiDAR point cloud data as input. The framework consists of two cascade backbone procedures. The former is pedestrian detection, which is developed for classifying a set of human clusters from noisy and sparse LiDAR point clouds. An originally designed voxel feature is integrated for depicting the 3D information of each potential cluster. A U-shaped convolutional neural network (CNN) is proposed to achieve human cluster detection. Then the latter procedure, pedestrian tracking, is developed to follow the target. Specifically, to achieve long-term person-following, an unscented Kalman filter (UKF), a position filter, and a target reidentification module are combined. The reidentification module contains a specifically designed support vector machine (SVM) target classifier. The developed framework is verified by extensive real-world experiments in outdoor environments using mobile robots. Compared with the state of the art, our method is better in target identification and tracking, realizing better performance in the person-following.**

***Index Terms*—Person-following, voxel feature, online SVM, LiDAR.**

I. Introduction

In recent years, mobile robots have become increasingly popular in industrial sites and our daily lives. They can undertake various tasks ranging from material transportation, surveillance, and disaster relief to offer companionship and entertainment. In these applications, the ability to automatically follow the target person is incrementally necessary, which liberates the hands and provides a flexible interactive mode for human-machine collaboration.

Various types of sensors are employed for the person-following task, among which the vision-based methods have been widely investigated. However, these approaches are still susceptible to illumination change. Specifically, intense sunlight or deep darkness often causes the failure of the following task. Comparatively, LiDAR is more robust in outside environments that can provide reliable data flow. Additionally, LiDAR is deemed to be more precise in distance measurement than the other sensors in this situation. However, using only LiDAR scans is still challenging for the person-following task. The generated point cloud from LiDAR is sparse and even sparser along with the increase in the detection range, leading to incomplete object outlines. Moreover, observations with noises make it more difficult to distinguish people from the background.

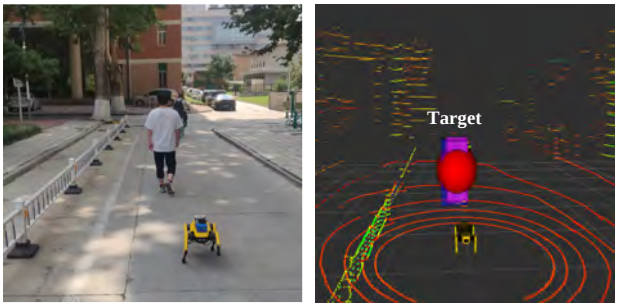


Fig. 1. The performance of the LiDAR-only person-following framework in an outdoor environment. The right figure indicates the sparse point cloud during the person-following.

There is an increasing number of attempts on achieving the LiDAR-based person-following task. This task is normally segmented into detection and tracking. In terms of detection, conventional methods such as SVM are applied for this task [1]–[3]. These methods select some potential point clusters from LiDAR data. Some features are extracted from each cluster and are used to identify pedestrians. These methods normally have satisfactory real-time performance but their detection accuracy leaves room for improvement. Recently, the booming popularity of integrating deep learning methods into this mission has appeared. These deep-learning based methods [4]–[6] are used for detecting objects from sparse point cloud data and are more precise than other traditional methods in datasets [7]. The high accuracy is attributed to their neural networks, which are trained by plenty of data extracted from similar scenes. However, they fail to be adopted on the robot platforms to perform the person-following task because the time cost on handling the entire points is too large to realize the real-time application. As for tracking, researchers prefer to use some filter-based methods [8], [9] to track the target from the detection results. Therefore, the tracking performance is partially determined by the detection results. The main drawback of these methods is that they will track the wrong target once the target overlaps or disappears in the LiDAR view.

In this context, we develop a deep learning-based person-

This work was supported in part by the National Natural Science Foundation of China under Grant No. 62103237 and in part by Natural Science Foundation of Shandong Province under Grant NO.ZR2021QF122.

[1] School of Control Science and Engineering, Shandong University, Jinan, China. Email: {cxy980212,ljh6182}@163.com, {chaoqunwang,rsong}@sdu.edu.cn.

[2] Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology. Email: jin_wu_uestc@hotmail.com.

[3] Shandong Research Institute of Industrial Technology, Shandong University, Jinan, China. Email: rsong@sdu.edu.cn.

*Corresponding Author.

following framework using only sparse LiDAR point clouds. This framework combines the advantages of traditional and deep learning-based methods to achieve real-time tracking and provides a solution for target loss. It consists of two main components as pedestrian detection and pedestrian tracking. Firstly, we preprocess the point cloud to remove the noisy ground points and group the filtered point cloud into several clusters. Subsequently, we extract the geometric features of the clusters, including a specifically designed feature, and put the features into a developed neural network for pedestrian detection. Moreover, a reidentification module is proposed to achieve continuous person-following by using an online updated SVM. Our main contributions are summarized as follows:

- A voxel feature is proposed to represent the 3D information of a point cloud cluster, which is not sensitive to the sparseness of LiDAR points.
- We employ a special U-shaped deep neural network to detect pedestrians from sparse point clusters, which improves pedestrian detection accuracy and is deployed in practical applications.
- We design an online learning mechanism to continuously enhance the robot's ability to detect and reidentify the target when the target is lost.

The effectiveness of the developed framework and its submodules are justified in real-world environments. The code of the developed framework can be found at *https://github.com/SDURobotNav/LOPF*.

The remainder of this paper is organized as follows. Sec. II reviews related work in this field. Then, Sec. III displays our framework and showcases details of our system. Moreover, in Sec. IV, we design some experiments and show the results to justify the robustness of our system. Finally, in Sec. V, we conclude this study and list future research directions.

II. Related Work

Person-following has been an active research trend over the past few years. In many applications, this task is achieved by using localization devices. For example, Nguyen *et al.* [10] use the ultra-wideband (UWB) device to achieve precise person-following tasks in indoor environments. Tian *et al.* [11] combine UWB with a low-cost inertial navigation system (INS) to improve the long-term person-following performance. Wu *et al.* [12] choose an ultrahigh-frequency radio frequency identification (RFID) device fixed on the target as the label, and the base station is mounted on the robot platform. They successfully integrate the tracking module into the athlete training task. There exist methods that use special materials (e.g., reflection materials) for the person-following task. Michal *et al.* [13] use an infrared camera to detect the retroreflective stripes on the soldier's back for person-following. Zhang *et al.* [14] use LiDAR to detect the high reflection marker for target identification. These methods are effective in certain confined applications. However, they need extra accessories, which limits the applications in complex scenarios.

To decrease the reliance on extra markers, many approaches attempt to accomplish the person-following task through efforts on the sensor side. Some researchers utilize vision-based methods with learning algorithms in person detection. Pang *et al.* [15] propose a leader-following framework that applies the single-shot multibox detector (SSD) [16] network using an image stream as input data with a quadruped robot. Liu *et al.* [17] integrate an SVM-based method with the SSD to detect the candidates robustly. The effectiveness of their method is verified by the target tracking performance with a quadcopter robot. Naseer *et al.* [18] extract both the image and depth information from the RGB-D camera to obtain position of the target person for tracking. However, vision-based methods are easily affected by illumination variations. Thus, in outdoor nonstatic environments, researchers are in favor of LiDAR-based methods.

LiDAR is more robust when facing changing environments and is more precise in distance measurement than camera. Luis *et al.* [3] use 3D LiDAR data to detect pedestrians. The developed pipeline first segments the point cloud into clusters. Then, an SVM classifier is used to extract the clusters that represent pedestrians. Häselich *et al.* [1] enrich the number of extracted features for classification by SVM, which enhances the detection performance. Additionally, Spinello *et al.* [19] propose an AdaBoost-based detection method called the bottom-up top-down detector to further improve the detection accuracy. Along with the development of deep-learning theory, Alvarez *et al.* [20] utilize a CNN network called PeTra to detect the person's legs from the LiDAR data. Yan *et al.* [2] merge the online-learning algorithm into a detection module in their system. This combination enables the tracking system to adjust its classifier dynamically. Nevertheless, these detection methods lack satisfactory robustness to achieve the person-following task in complex outdoor environments.

Basically, a tracking routine proceeds to follow the target after it is captured. For the target tracking module, filter-based methods are soaring in popularity. The Kalman filter (KF) with its variations and particle filter (PF) have been representative algorithms for accomplishing the tracking task. Patel *et al.* [21] utilize the KF to track a moving object. In these works [1], [2], [22], UKF is used for tracking multiple people in large-scale environments. Additionally, Okuma *et al.* [23] present a boost particle filter to achieve multiple object tracking. Yuan *et al.* [24] integrate the particle filter into a mobile robot to make the robot follow a target person. However, traditional tracking methods cannot achieve sustainable tracking once the target object disappears. They lack a reidentification module for the following target.

To achieve the long-term tracking task, we use the online learning-based method for target reidentification. Online learning-based methods have been developed for the last few decades. Hao *et al.* [25] first propose online SVM learning for classification problems. Their algorithm can efficiently handle the data that are ordered chronologically. Yan *et al.* [2] design an online learning framework for pedestrian detection. This framework can autonomously label the real-time data flow for updating the classifier with the latest training data. Dilmen *et al.* [26] propose an online updated SVM classifier. The

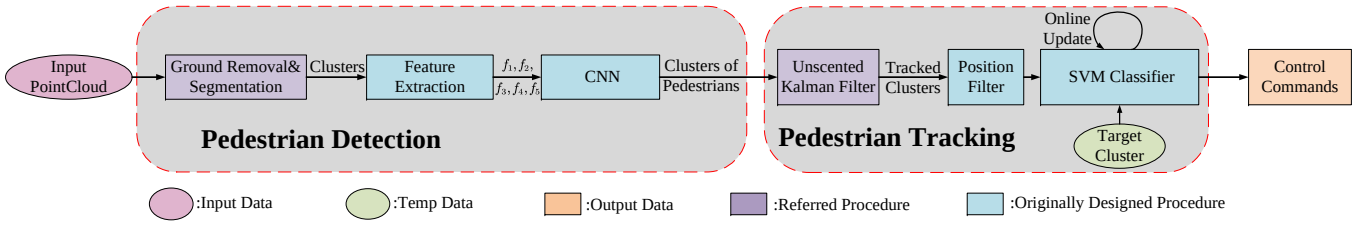


Fig. 2. The framework of the online LiDAR-only person-following framework. The framework has two parts: 1) **Pedestrian Detection** and 2) **Pedestrian Tracking**. The pedestrian detection module consists of ground removal and segmentation, feature extraction, and CNN for classification. The pedestrian tracking module consists of UKF, a position filter, and an online SVM classifier.

parameters of their prediction model are updated by a UKF when receiving new samples.

III. Methodology

The developed person-following framework is indicated in Fig. 2, which consists of two modular components, the **Pedestrian Detection** and **Pedestrian Tracking**. The former part takes the raw point LiDAR data as input and outputs the potential pedestrian clusters. Then, the latter part proceeds to track pedestrian clusters and follow the target person who is marked from the extracted pedestrian clusters at the beginning of the task. The key functions of the two components are introduced in the following sections.

A. Pedestrian Detection

The pedestrian detection module receives the raw point cloud $\mathcal{P}$ from LiDAR and preprocesses the data to obtain a filtered point cloud $\mathcal{P}^-$. $\mathcal{P}^-$ is then segmented into several clusters $\{C_1, C_2, ..., C_n\}$. For each cluster C_i, five designed features $\{f_1, f_2, f_3, f_4, f_5\}$ are extracted. These features are fed into our proposed CNN model to determine whether C_i is a pedestrian cluster. Detailed procedures are listed as follows.

1) Preprocessing: The raw point cloud $\mathcal{P}$ can be easily contaminated by the point cloud on the ground, as shown in Fig. 3 (a), which greatly affects the detection performance. To alleviate this problem, we employ the RANSAC algorithm [27] to match a ground plane from the raw point cloud and remove the points that are in the extracted ground plane. In addition, a statistical outlier removal algorithm [28] is used to remove the outliers. V_i is denoted as the average distance among $p_i \in \mathcal{P}$ and its m neighbor point $p_j \in \mathcal{P} \setminus \{p_i\}$ from near to far. We assume that the set $\mathcal{V} = \{V_1, V_2, ..., V_n\}$ fits a Gaussian distribution $N(\mu, \sigma^2)$. The point p_i, of which the V_i is out of the standard deviation σ of the generated distribution, is regarded as an outlier. After removing the ground points and outlier points, a low-noise point cloud $\mathcal{P}^-$ is obtained, as shown in Fig. 3(b).

2) Cluster Detection: The point cloud $\mathcal{P}^-$ is segmented into subclusters with Euclidean cluster extraction (ECE) and dp-means [29] by using the point cloud library (PCL) [30]. The ECE algorithm segments the point cloud into scattered clusters by Euclidean distance. To further refine the point clusters, the dp-means algorithm proceeds to divide each cluster into subclusters until the distance among the separated clusters is out of a given threshold. This procedure effectively segments the point set $\mathcal{P}^-$ into small clusters $\{C_1, C_2, ..., C_n\}$.

(a) Prior to Removal (b) After Removal

Fig. 3. Ground LiDAR point cloud removal for better person-following performance.

TABLE I. Features of a Cluster

Feature	Description	Dimension
f_1	Number of points in the cluster [31]	1
f_2	Minimum cluster's distance from the sensor [31]	1
f_3	The normalized 2D histogram for the main plane [32]	14×7
f_4	The normalized 2D histogram for the secondary plane [32]	9× 5
f_5	The voxel feature for the cluster	8×8×4

3) Feature Extraction: For each cluster C_i, its features are extracted to determine whether it is a pedestrian cluster. The selected features are shown in Table I. Specifically, given the cluster C_i, f_1 represents the number of points in this cluster. f_2 is the minimum distance from the points in cluster C_i to the sensor, i.e., $f_2 = \min ||\mathbf{x}_* - p_j||, p_j \in C_i$, where $\mathbf{x}_*$ is the origin of the LiDAR frame. By performing the principal component analysis (PCA), we obtain two main planes by using the points in C_i. f_3 and f_4 are separately 2D histograms with respect to these two planes (see [33]), which can be regarded as the front view and side view of the cluster C_i, respectively. Typically, off-the-shelf algorithms use cutting slices of cluster C_i in a certain direction as features to extract 3D information [2]. Having insufficient points in the cluster incurs the poor descriptive ability of these mentioned features.

Considering these limitations, we design a feature f_5 named the *voxel feature* to describe the 3D information of a point cluster. This *voxel feature* is inspired by *Minecraft*[1] and *LEGO*[2]. Generating f_5 is named as cluster voxelization. We create a voxelized cube $\mathcal{C}_i$ with $\Delta_x, \Delta_y, \Delta_z$ indicating

[1]A video game, https://www.minecraft.net
[2]A brand of toy, https://www.lego.com

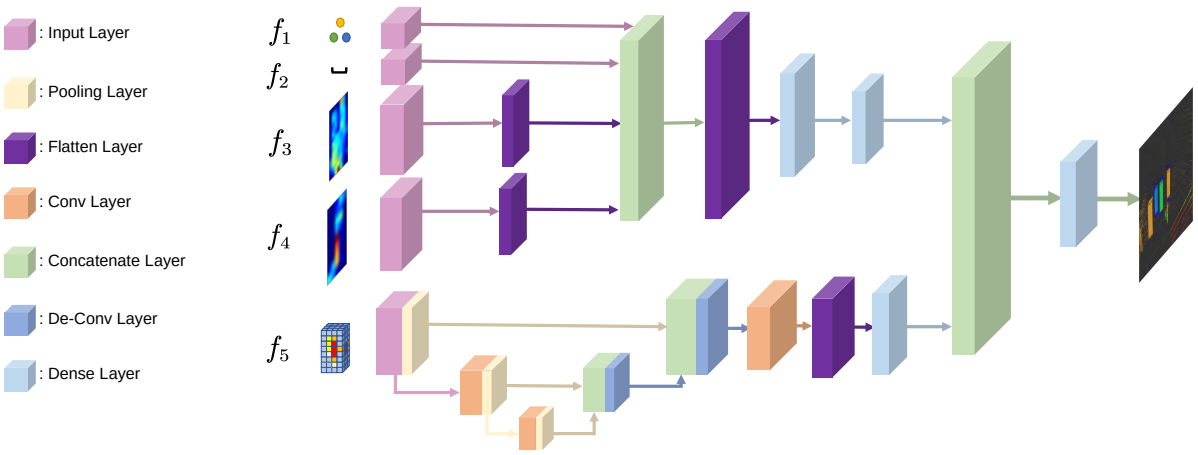


Fig. 4. The developed network for pedestrian detection. The network consists of two kinds of inputs, the 2D features (f_1, f_2, f_3, f_4) and the 3D feature (f_5).

the number of voxels in the x, y, z directions, as shown in Fig. 5. To map all the points in cluster C_i into this cube, we search all the points in C_i to obtain the maximum and minimum coordinates in the x, y, z directions in the current LiDAR frame. These coordinates are marked as $\{x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}\}$. Then, we denote a local coordinate frame on the cube. The coordinate of a point $p_i \in C_i$ in the LiDAR frame is $[x, y, z]$. Its coordinate in the local frame is denoted as $[\hat{x}, \hat{y}, \hat{z}]$, which is calculated by the transform equation:

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} \gamma_1 & & \\ & \gamma_2 & \\ & & \gamma_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} \gamma_1 \cdot x_{min} \\ \gamma_2 \cdot y_{min} \\ \gamma_3 \cdot z_{min} \end{bmatrix}, \tag{1}$$

where

$$\begin{cases} \gamma_1 = \frac{\Delta_x}{x_{max} - x_{min} + 1} \\ \gamma_2 = \frac{\Delta_y}{y_{max} - y_{min} + 1} \\ \gamma_3 = \frac{\Delta_z}{z_{max} - z_{min} + 1} \end{cases}. \tag{2}$$

Using $[\hat{x}, \hat{y}, \hat{z}]$, one can locate the voxel to which point p_i belongs. We search all the points in C_i to obtain their locations in the local feature frame. Each voxel c_j in the cube $\mathcal{C}_i$ counts the number of points from C_i located there and obtains the amount N_{c_j}. To avoid the effect of the number of points in a cluster when handling different clusters, we normalize N_{c_j}. Thus, we calculate the number of points in C_i and obtain N_{tot}. The quantity associated with cell c_j in $\mathcal{C}_i$ is calculated by $\bar{N}_{c_j} = N_{c_j}/N_{tot}$. Fig. 5 indicates the generated feature, where the colorful voxels represent the voxelized cluster. The color indicates the number of points in the corresponding voxel. The feature f_5 briefly highlights the 3D distributions of points in a cluster and is added to our framework for better classifying the potential persons.

4) CNN for Pedestrian Classification: With the feature mentioned above, we design a CNN model to select pedestrian clusters $\hat{P}_i, \hat{P}_i \in \mathcal{P}^-$. The network structure is shown in Fig. 4. It takes as inputs both the 2D information (f_1, f_2, f_3, f_4) and 3D information (f_5). The backbone of the network consists of two main pipelines.

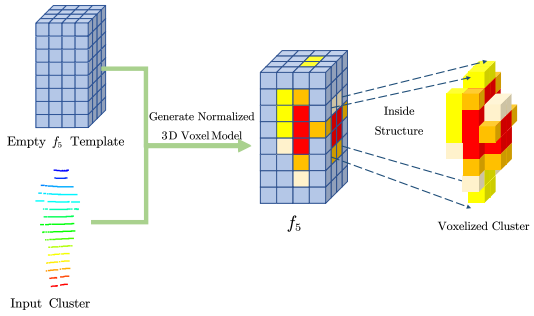


Fig. 5. Cluster voxelization. The point cluster is scattered into a spatial partitioning cube.

The first pipeline in the CNN model for processing the 2D information is a lightweight neural network. f_3 and f_4 are flattened by the flatten layers, and then a concatenate layer is used to merge f_1, f_2 and the flattened f_3, f_4. Finally, we use two dense layers to obtain the output. This part of the network mainly uses a few layers to densely combine the mentioned four features to obtain the information of all four features.

The second pipeline processes the voxelized feature to extract the 3D information. Thus, a subnetwork is built upon the U-Net [34] diagram that is effective for image processing with less data support. The U-shaped network is a universal structure, which has been applied in many studies [35]–[37] in the computer vision area. The U-shaped network is favorable to reduce information loss in convolution operations and obtain the profile information, which shows the obvious difference between pedestrians and other objects. Thus, U-shaped network is more suitable to handle the designed feature f_5 for detecting pedestrians. We follow the structure and redesign the configuration, such as the layer combination and the size of each layer, to fit our proposed f_5. To combine the 2D and 3D information, the outputs of these two pipelines are concatenated with

$$[t_1, t_2] = S(D(f_{2d} \oplus f_{3d})), \tag{3}$$

where $S(\cdot)$ represents the "softmax" activation function, $D(\cdot)$

represents the operation in the dense layer, f_{2d} represents the output of the first pipeline, f_{3d} represents the output of the second pipeline, and $\oplus$ represents the concatenation operation in deep learning. The final result is a 2×1 dimensional vector as $[t_1, t_2]$ with $t_1 + t_2 = 1$. t_1 represents the probability to be a person. Once $t_1 > t_2$, the cluster is regarded as a person. To keep the scalability of the network to multi-object detection and tracking in the future, we adopt $[t_1, t_2]$ instead of a single value as the output.

We obtain detection results from our proposed network by handling the 2D and 3D features of point clusters rather than the whole point cloud data. Thus, the designed system combines the benefits of both traditional and deep learning-based methods to improve the accuracy of detection while ensuring real-time performance.

B. Pedestrian Tracking

From the detected pedestrian clusters, the cluster representing the target person is specified at the beginning of the following task. Considering that there are multiple people in the environment, we assign a unique ID to each detected person. The UKF with the global nearest neighbor data association algorithm [38], [39] is utilized to track the position of multiple people. This filter uses the observed pedestrians' position as input and assumes that them moves with a constant velocity. For the person-following task, we assume that the target only has a short movement between adjacent observations, which acts as a strong constraint and forms a target position filter toward further improving the target tracking. The position filter searches the potential candidates in a preset range around the last location of the target and then chooses the nearest one as the target.

Sustainable person-following by using LiDAR is a tough task for robots in the wild . The tracked target may be lost due to disturbances such as occlusions. Thus, we develop a classification method that incrementally learns the features of the clusters online to clearly distinguish the target and the rest. In particular, the SVM is trained online with collected features from the segmented clusters during the following procedure. The developed framework is shown in Fig. 6, which consists of two main modules, the data collection and the training of the SVM classifier. The dataset for classifier training stems from the tracking results by the mentioned filters and the self-output of the classifier. The UKF and the target position filter, when tracking robustly, can help to determine the followed person. Thus, the results are used to label the selected clusters to enrich the training data. In addition, the output of the online SVM is also used to augment and update the training dataset in a similar manner. Specifically, considering the possible outputs of the incorrect pedestrian classification that recognizes object clusters as humans, we further enrich the dataset by adding feature data extracted from some surrounding objects, such as cars, trucks, and trees. The data representing environmental objects are collected offline as partial training data. Notably, the size of the training dataset is limited, meaning that only the latest data online are stored in this container except the object data.

With the training dataset, the online SVM classifier is updated by feeding the latest training data. We use the Gaussian Radial Basis Function kernel [40] as our online SVM classifier kernel. To save computational resources, we developed an event-triggered working mechanism. We set the triggered interval as 30 s considering our configurations, but actually, it can be set in a broader range. It is desirable to trigger classifier training when the robot enters a new scene. This is evaluated by the overlap between the detection range of the employed sensor. The detection diameter of the sensor is 44.36 m. Normally, the walking speed of the pedestrian is approximately 1.1 m/s to 1.5 m/s. Therefore, there is a new scene approximately every 29 s - 40.3 s. To simplify the problem, we set the triggered interval as 30 s. In addition, the SVM works in an event-triggered fashion to reduce the computational burden. If the target is lost, the SVM classifier is activated to find the target person from all detected people in the current scenario. If the target is detected, the SVM classifier will be dormant but updated at the given time interval. With the developed framework, the person-following robustness is expected to be guaranteed in more complex scenarios.

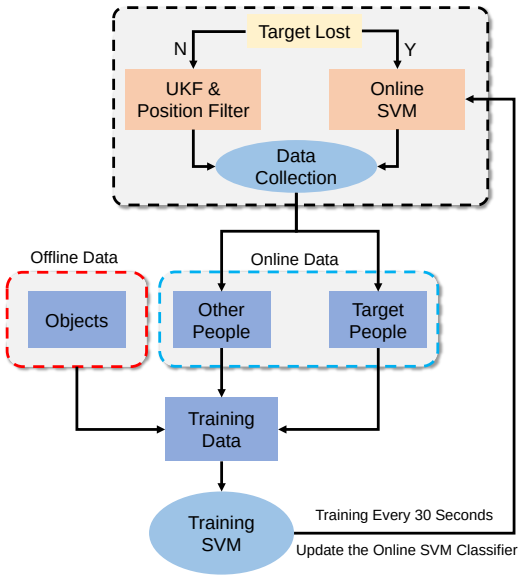


Fig. 6. Online classifier for improving the person-following robustness and recapturing the target when the target is lost.

As indicated in Fig. 6, in our work, we use an online learning method to handle the real-time input extracted from sensor data for reidentifying the target. The differences between our reidentification module and other online learning methods lie in three aspects. Firstly, the training data of the online SVM classifier are autonomously labeled with the target and the other people according to the results from the position filter and the CNN model. Second, we update the classifier with a fixed time interval because it is desirable to update the classifier along with the changes in environments. The fixed time interval is approximated by the walking speed and detection range. Thirdly, the use of the reidentification module is event triggered. To economize the computational resources,

the module works only if the target has been lost. Thus, our method will autonomously change their classifier model when following different target persons.

IV. Experiments and Results

We conduct a series of experiments to verify the submodules and the whole framework. We use a quadruped robot and a differential-driven wheeled mobile robot as our experimental platforms, which are shown in Fig. 7. The VLP-16 produced by Velodyne is utilized as the LiDAR sensor, which provides 16 scan channels including 1,800 points per channel with 360-degree horizontal and 30-degree vertical field-of-view [41]. The data flow is preprocessed by a laptop. The laptop uses an Intel i7-10875 as the CPU and a GTX 2060 (6 GB memory) as the GPU. The software used in this work is Ubuntu 18.04 64-bit with the ROS Melodic. The computer processes the sensor data and generates the control commands. We use the visual servoing method as the control driver, which transfers the relative position relationship between the target person and the robot to velocity commands. The robot receives these commands and makes corresponding actions.

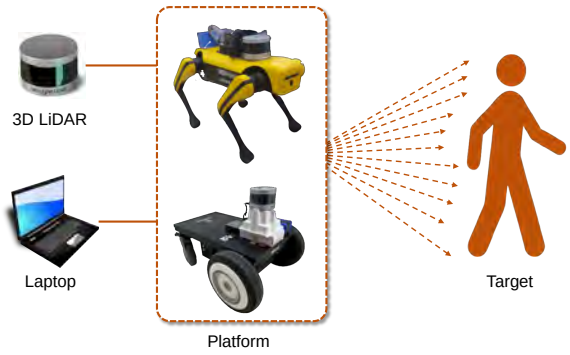


Fig. 7. The robot platforms for the experiments. They are utilized for evaluating the developed person-following framework.

A. Preparatory Work

To evaluate the developed framework, we collect a dataset that is necessary for training the proposed CNN. In addition, we need to select a suitable size of the developed feature f_5 via experimental evaluation, which is critical for the network to achieve better performance.

1) Collect Training Data: The training data include many point clusters with two labels, person and objects. The point clusters are segmented and labeled from the PCD files by using the annotation tool [42]. These PCD files are extracted from existing datasets [19], [42] and the datasets we collected. In detail, 3,610 clusters are segmented and labeled from the published dataset, and 2,491 clusters are from ours. All clusters are formatted into PCD files, including 4,447 pedestrian clusters and 1,654 object clusters. Finally, these labeled clusters are the training samples for the proposed CNN. In the following procedures, we use the training setting shown in Table II to train the designed network. We use Adam [43] as the training optimizer. β_1 and β_2 are the parameters of Adam. The dataset is separated into a training set and a test set according to the training set proportion value shown in Table II. The former is used for training the network, while the latter is used for evaluating the training performance after finishing the training process.

TABLE II. Training Settings

Config	Value
Optimizer	Adam [43]
Learning Rate	10^{-3}
β_1	0.9
β_2	0.99
ϵ	10^{-8}
Epochs	1000
Training Set Proportion	0.9

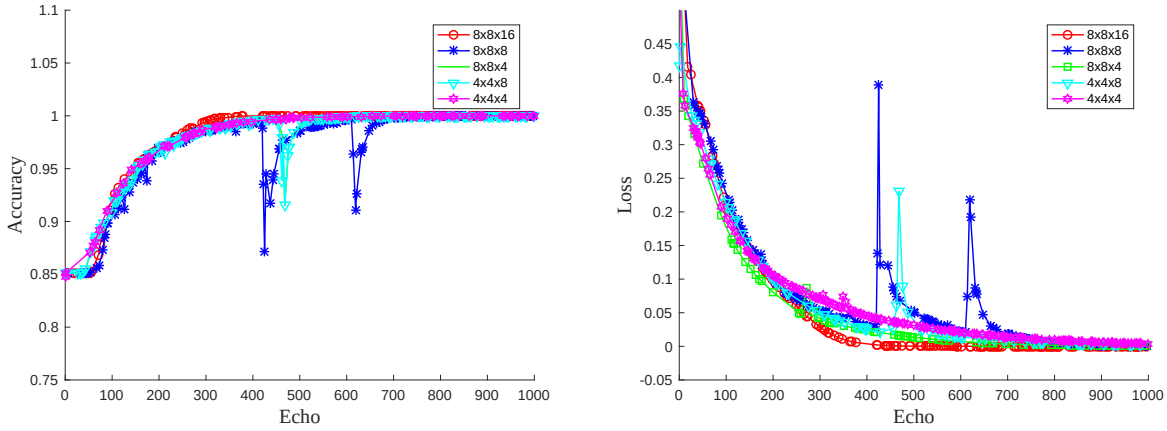
2) Choose the Size of f_5: The ideal size of f_5 should lead to a reasonable tradeoff between the time cost and detection accuracy considering practical applications. We select a group of f_5 features with different sizes for the evaluation. The candidate sizes of feature f_5 are $\{8\times8\times16, 8\times8\times8, 8\times8\times4, 4\times4\times8, 4\times4\times4\}$. The numbers represent the arrangement of the cube in the corresponding f_5. For example, the f_5 with $8\times8\times4$ has 4 layers, and each layer has 64 cubes, which are arranged as 8 rows and 8 columns. In our designed network, there are two 3D pooling layers processing the f_5. The size of the pooling layer kernel is normally set as $2\times2\times2$ [44]. The pooling layer with this size can halve the size of the input in all three dimensions. The size of the input is halved twice by the two pooling layers. Therefore, the size of f_5 in each dimension must be a multiple of 4 to ensure that each voxel is processed equally. Thus, the minimum size of the feature is $4\times4\times4$. Based on the pedestrian profile, we set the number in the first dimension to be equal to that in the second dimension. We are not allowed to design input structures more sophisticated than $8\times8\times8$, because they spend too much computational power and the time cost is high, which is detailed in Table III.

TABLE III. Comparison Results with Different Shapes of the Feature f_5

	$8\times8\times16$	$8\times8\times8$	$8\times8\times4$	$4\times4\times8$	$4\times4\times4$
Accuracy	88.7%	93.0%	92.75%	92.5%	91.25%
Time Cost	151μs	93μs	71μs	51μs	41μs
Convergence Epoch	414	804	628	623	707
Time Delay	30.2 ms	18.6 ms	14.2 ms	10.2 ms	8.2 ms

The dataset collected in 1) has abundant data, which include most normal objects and people with different postures and sizes. Thus, the network with the selected optimal size of f_5 can have better performance in widespread situations. The *Accuracy*, *Time Cost per sample*, *Convergence Epoch*, and *Time Delay per Second* are the evaluation indices employed for choosing the best f_5 size. The *Accuracy* represents the detection performance of the network using the test set extracted from the dataset. It is calculated by

$$Accuracy = \frac{N_t}{N_{test}}, \tag{4}$$



(a) (b)

Fig. 8. Training results with different sizes of the developed voxel feature. (a) The training accuracy with different voxel feature sizes. (b) The training loss with different voxel feature sizes.

where N_t is the number of true predictions and N_{test} is the number of total clusters in the testing data. The *Time Cost per Sample* represents the detection speed, and the *Convergence Epoch* indicates the convergence performance of the network. The network reaches convergence when the accuracy is over a threshold in the training process. More iterations of training do not result in a significant improvement in the performance of the model. Similar to [45], we record the point of the epoch when the designed network reaches 99.9% accuracy as the "Convergence Epoch". To evaluate the real-time performance, we calculate the *Time Delay per Second*. The number of clusters segmented in one frame of LiDAR data is denoted as $N_{cluster}$ and the rate of the LiDAR data is denoted as r_L. Then, we calculate the time delay per second T_d by

$$T_d = N_{cluster} \times r_L \times T_c, \tag{5}$$

where T_c is the *Time Cost per Sample*. In our experiment, N_c is 20, and $r_L = 10$ Hz. The network training result is shown in Table III. Overall, all these sizes f_5 can make the proposed network converge according to the results of the *Convergence Epoch* metric. The $accuracy$ for all the methods is acceptable except that the $accuracy$ with $8 \times 8 \times 16$ is lower than 90% compared with other methods. It is observed that the time delay over 20 ms per second is critical for real-time performance. Thus, the size $8 \times 8 \times 16$ is not available because of its large time delay.

In terms of the training results shown in Fig. 8, the sizes $8 \times 8 \times 8$ and $4 \times 4 \times 8$ have some significant fluctuations. The fluctuation is the sudden plunges in the accuracy curve or the surges in the loss curve. The fluctuations in the network training process easily make the network unstable. The fluctuations are normally caused by the choice of parameters and the network structure. All these designed comparison experiments have the same parameters except the size of f_5. Therefore, the two sizes of f_5 are not suitable for the developed framework. Based on these analyses, the size $4\times4\times4$ and the size $8\times8\times4$ have better performance than the others. Finally, we use the $8 \times 8 \times 4$ feature with the higher $Accuracy$ as our final size of f_5 since the time cost or delay is tolerated in our testing.

3) Ablation Study: We perform an ablation study of all features. We supply an experiment that evaluates the detection accuracy with four and five features separately. To make the experiment more inclusive, we further add the detection results by using only feature f_5. We record the detection accuracy in the experiment, and the results are reported in Table IV.

TABLE IV. ABLATION STUDY OF THE SELECTED FEATURES

	f_1, f_2, f_3, f_4	f_5	f_1, f_2, f_3, f_4, f_5
Accuracy	85.5%	88.25%	92.75%

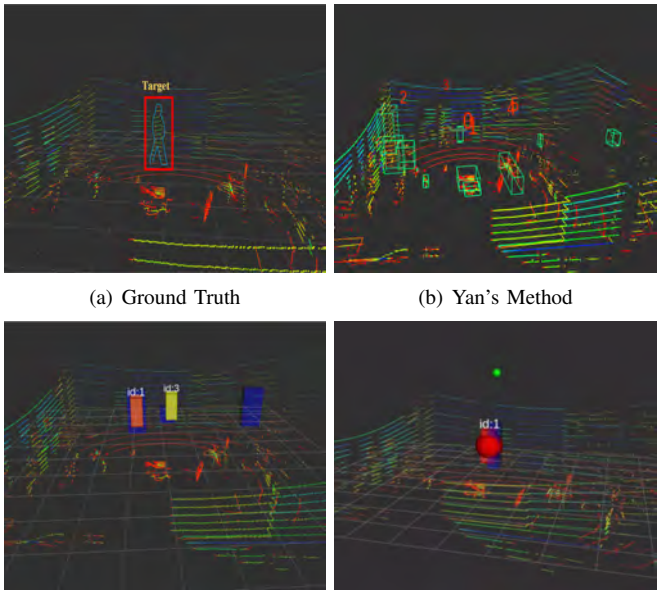
As the table shows, with only four features, the detection accuracy is the lowest compared with other groups. With only feature f_5, the detection module achieves higher accuracy. The detection module using all five features can achieve the highest detection accuracy, which also demonstrates the effectiveness of the feature f_5. Therefore, we use all these five features for better detection accuracy.

B. Experiments Using Offline Data Flow

To show the strengths of the developed method, we compare our method with two state-of-the-art pedestrian-tracking algorithms, Koide's method [22] proposed in 2019 and Yan's method [2] proposed in 2020. The main differences between our algorithm and the other two algorithms can be described in two parts. The first difference is the pedestrian detector, which includes discrepancies in the detector structure and selected features. They use SVM as their detector with their selected features. We propose a novel combination of features as input and propose a U-shaped neural network as the pedestrian detector. The second difference is the online update reidentification module. This module autonomously generates a dataset by the detection results to train an online classifier to identify the target from the detected people. The experiments are performed using a collected data flow, which includes the

data flow we collected and a published data flow by L-CAS [42].

1) Performance with Our Data Flow: The indoor environment is approximately 30 m^2, and we record a scene in which a person walks randomly in a cluttered environment, as indicated in Fig. 11. We evaluate all three algorithms by using this single-person scenario. Two metrics are employed as the criteria for further quantitative evaluation. The $Candidate\ Clusters$ is the average number of plausible clusters extracted using different methods in all frames. The $Number\ of\ People$ represents the average number of people detected by the employed methods in all frames. In the single-person scenario, the detection performance of these three methods is shown in Table V and Fig. 9.



(a) Ground Truth (b) Yan's Method

(c) Koide's Method (d) Ours

Fig. 9. Experimental results using the collected indoor data flow. (a) indicates the actual position of people shown in the LiDAR point cloud. (b)-(d) are the results of these three methods. The detected person clusters are represented by the cube box frame, cube box, and sphere box, respectively.

TABLE V. STATISTICS OF EXPERIMENTAL RESULTS OF SINGLE-PERSON TRACKING

	Ground Truth	Koide's Method	Yan's Method	Ours
Candidate Clusters	1	2.72	5.40	1.34
Number of People	1	2.67	3.78	1.23

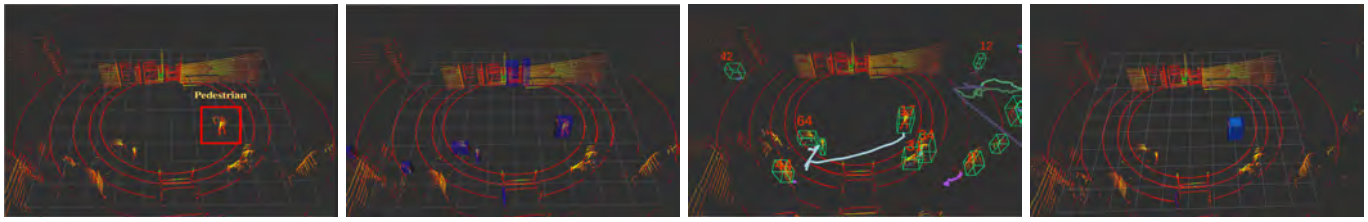
In this table, the number is the average value for all frames. The results show that our algorithm has the best detection result in the single-person scenario because the number of people detected by using our method is the closest to the ground truth in the real scenario. Fig. 9 shows a snapshot of the experiment using these three methods. Our method can detect a single person reliably, while the other two methods showcase many incorrect identifications. The proposed method also has less incorrect detection, such as walls, boxes, and shelves. This intuitively demonstrates the effectiveness and accuracy of our method when dealing with only one single person. We use four metrics to evaluate the performance by using this data flow. These metrics are listed as follows:

- Time of Tracking (TT): This metric is the normalized CT [46] metric for better comparison. The value of this metric is calculated by
$$TT = \frac{CT}{T_{total}}, \tag{6}$$
where CT is the time duration of persistently tracking the target, which is used for evaluating the person-following performance based on vision, and T_{total} represents the total time of the data flow. The closer to 1 the metric is, the better tracking performance the method has.
- ID Switches (IDS) [47]: The target is assigned an ID, and the metric records the number of ID switches, indicating the performance of target tracking on another side. With these methods, each detected person is assigned a specific ID. We make the robot track the target person by specifying the assigned ID. If the ID changes, the tracking for the target object fails.
- Positive Prediction Value (PPV) [48]: This metric is calculated by
$$PPV = \frac{N_r}{N_p}, \tag{7}$$
where N_r is the number of correct detections and N_p is the total number of detections for all pedestrians. The PPV value can reflect the performance of detection for all pedestrians.
- Accuracy (ACC) [48]: This metric is calculated by
$$ACC = \frac{N_c}{N_{total}}, \tag{8}$$
where N_c is the number of correct cluster detection results and N_{total} is the number of total clusters. This value can reflect performance cluster detection.

TABLE VI. EXPERIMENTAL STATISTICAL RESULTS OF THE DATA FLOW

	Koide's Method	Yan's Method	Ours
TT [46]	**0.893**	0.337	0.882
IDS [47]	2	5	**0**
PPV [48]	0.291	0.223	**0.988**
ACC [48]	0.297	0.319	**0.891**

The results are shown in Table VI. Our method achieves the best performance regarding the last three metrics. In terms of the TT, our method is slightly weaker with 0.882 than 0.893 using Koide's method. The better performance of our method benefits from the better pedestrian classifier and the developed person reidentification mechanism. For single-person tracking, the results of the ID switch (IDS) in Table VI indicate that both Yan's method and Koide's method change the target ID several times in the data flow. The person-detection modules of these two methods are of lower accuracy and are not specifically designed for one target. They fail to recognize the same target and often misidentify the target as another person. Thus, the two methods cannot sustainably track the target person, while our method can reliably and continuously detect and make

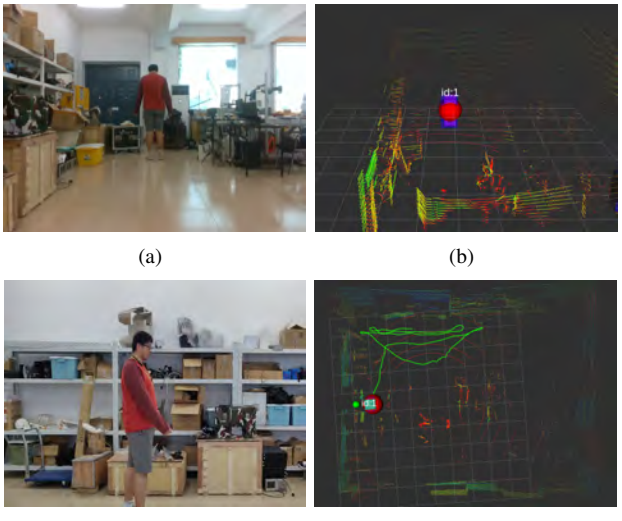


(a) Ground Truth (b) Koide's Method (c) Yan's Method (d) Ours

Fig. 10. Person detection using different methods with L-CAS data flow. In (b)-(d), the detection results are marked by the blue cube box frame, cyan cube box frame, and sphere box, respectively.

the robot follow the target. The weakness of our algorithm according to the first metric is caused by a longer time delay in the target detection using the developed network, which is acceptable in real-world applications. The results demonstrate the better performance of the developed method.

Specifically, we mark the detected target and record the target trajectory when it is captured by the developed method, as indicated in Fig. 11(b) and Fig. 11(d). The green line in Fig. 11(d) demonstrates the continuous tracking performance using the developed method. The target is prone to be lost in the tracking using the other methods, which further demonstrates the benefits of our approach.



(a) (b)

(c) (d)

Fig. 11. The environment for collecting the offline data and the corresponding point cloud. (a) and (c) show the real-world environment configurations. (b) and (d) indicate the generated point cloud.

2) Using L-CAS Data Flow: The L-CAS dataset is collected by using a mobile robot in both moving and stationary scenarios in a large and crowded academic building. We, employ Yan's method, Koide's method, and our method using the LiDAR data in this data flow. The representative results using one instance in the data flow are depicted in Fig. 10. There is a person in this instance who is marked by the red rectangle in Fig. 10(a). Fig. 10(b)-(c) showcase the performance of Koide's method and Yan's method. Both methods can predict the target person. However, there are many incorrect predictions that regard the objects, such as sofas, walls, and chairs, as pedestrians. These inaccurate predictions may be acceptable in obstacle avoidance applications since safety is always the priority. However, in person-following applications, these incorrect predictions may increase the probability of target loss, leading to the failure of the tracking module. Comparatively, the developed method achieves better performance in not only detecting the target reliably.

Quantitative experiments are performed to show the performance of these three methods on the L-CAS dataset. The dataset provides boxes marking the position of people in each frame, and these methods also provide their predicted boxes. If there is an overlap between the predicted box and the annotated box in the dataset, we deem this prediction of people to be correct and calculate the distance d between the centroids of the predicted box and the annotated box. We count the number of right predictions as N_{rp} and the number of predictions by these methods as N_p. Therefore, the precision of prediction P_{LCAS} is defined as

$$P_{LCAS} = \frac{N_{rp}}{N_p}, \tag{9}$$

and the position error e, which is the average d of all correct predictions, is calculated as

$$e = \frac{\sum d_i}{N_{rp}}. \tag{10}$$

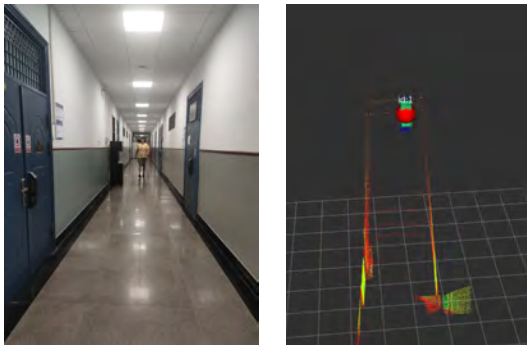
We report their performance by using the two metrics in Table VII. As the table shows, our method has the highest P_{LCAS} and the smallest position error e. The results show that our method is able to precisely predict pedestrians and locate them. Therefore, our method outperforms the other two methods in this quantitative experiment.

TABLE VII. QUANTITATIVE EXPERIMENTAL RESULTS WITH L-CAS DATASET

	P_{LCAS}	e/m
Koide's Method	0.098	0.133
Yan's Method	0.154	0.127
Ours	**0.459**	**0.102**

C. Experiment in Real-world Scenarios

1) Pedestrian Identification: We conduct extensive experiments to evaluate the performance of the developed pedestrian detection and tracking approach in indoor environments. Firstly, we compare the detection range of the developed



(a) (b)

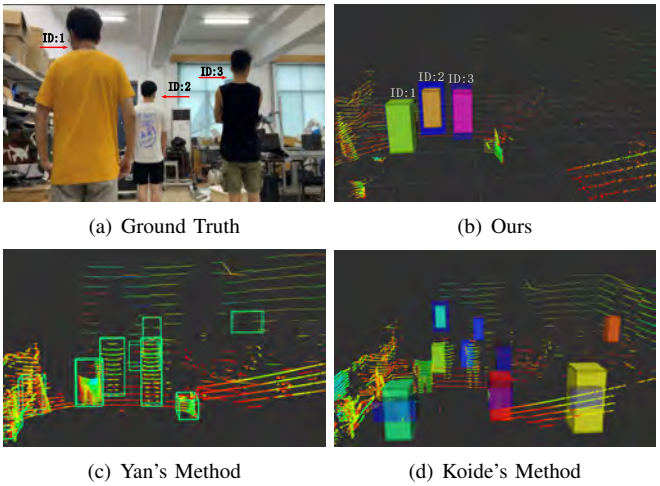
Fig. 12. The detection range evaluation in a long corridor with a fixed-base LiDAR. (a) The experimental configuration. (b) The corresponding LiDAR point cloud.

approach with other methods. The VLP-16 is mounted on a fixed base with 1 m above the ground in a long corridor. We separately run the three methods when a person moves from the sensor nearby to far, as indicated in Fig. 12. Notably, these methods run round by round in this process. We count this round when the method can predict the target, and the total successful rounds are recorded in Table VIII. In addition, we record the max range for detecting the target with different methods. As shown in Table VIII, the maximum detection range of the developed method is 22.18 m, which is slightly shorter than that of Koide's method but much longer than Yan's method. The successful prediction rounds of the developed method are higher than those of the others. These experiments primarily demonstrate the effectiveness of the developed method in indoor real-world environments.

TABLE VIII. RESULTS OF DETECTION RANGE EVALUATION

	Koide's Method	Yan's Method	Ours
Maximum Distance / m	**22.26**	17.98	22.18
Successful Rounds	46	7	**49**

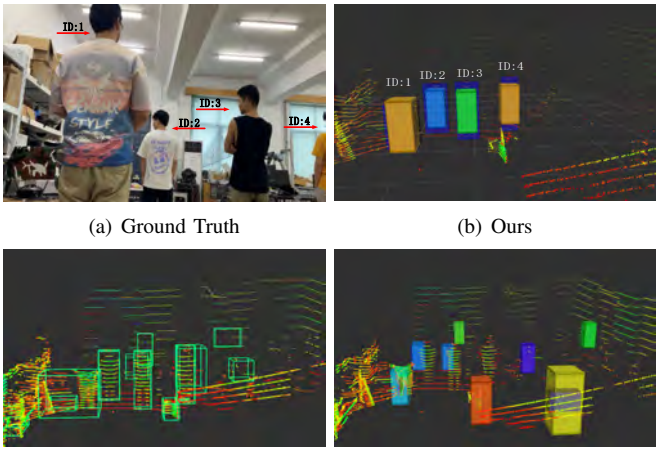
To achieve the person-following task in more realistic scenarios, the developed algorithm is expected to identify the target in densely populated environments. To evaluate the performance of our method in these environments, we conduct experiments in cluttered environments with massive items and multiple people walking randomly. The LiDAR is fixed on a base above the ground, and there are multiple people ahead. The environment configurations and the detection results are shown in Fig. 13 and Fig. 14 intuitively. With multiple people in this environment, the developed method can detect all people from the LiDAR point clouds. We also use the other two methods to detect people in this scenario. From the detection results shown in Fig. 13 and Fig. 14, it is obvious that our method is better than both of them. Yan's method and Koide's method both have some incorrect detection results, which directly weakens the multipeople tracking performance because the tracking modules in all three algorithms are based on their detection results. These experiments further demonstrate the strength of the developed method in cluttered and



(a) Ground Truth (b) Ours

(c) Yan's Method (d) Koide's Method

Fig. 13. The detection of three people in cluttered environments. (a) is the real scenario of three people. In (b)-(d), the colorful cubes or boxes are the detection results of ours, Yan's method, and Koide's method.



(a) Ground Truth (b) Ours

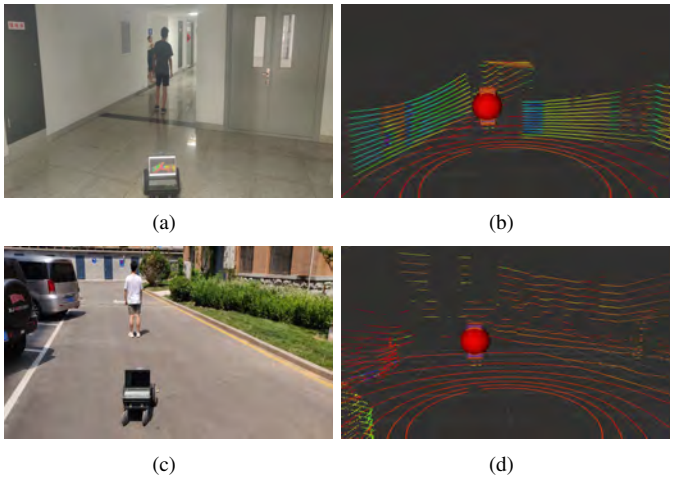
(c) Yan's Method (d) Koide's Method

Fig. 14. The detection of four people in clustered environments. (a) is the real scenario of four people. In (b)-(d), the colorful cubes or boxes are the detection results of ours, Yan's method, and Koide's method.

populated environments, paving the way for person-following using robots in these scenarios.

2) Case Study 1: To verify the effectiveness of the developed framework, we implement it on a wheeled mobile robot platform. The experiments are conducted in both indoor and outdoor environments. The target person walks from a start location to a final destination. The goal of the robot is to reliably follow the designated target via the LiDAR sensor. The performance of our method is intuitively shown in Fig. 15. As the figure shows, the developed framework can follow the target person reliably and persistently with the wheeled mobile platform in different environments. Details can be found in the attached video[3], which demonstrates the efficacy of the developed framework. To show the strength of our framework in the application case, we compare our method with Yan's and Koide's methods. The duration time for tracking the target with different methods is recorded in Table IX. Our method

[3]https://youtu.be/eocqMLzRR1s



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

Fig. 15. Person-following experiments using a mobile robot. (a) and (b) are the real scene of person-following and the point cloud in an indoor environment. (c) and (d) are the real scene of person-following and the point cloud in an outdoor environment.

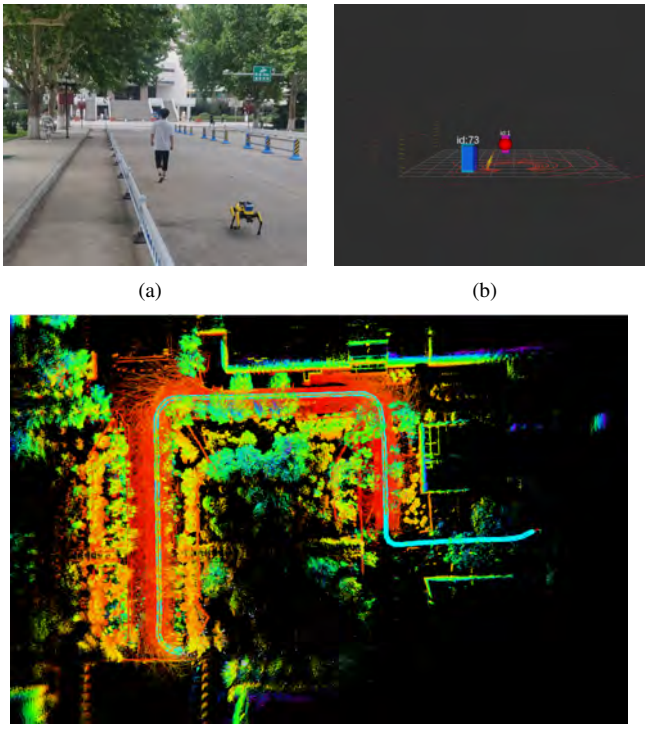
TABLE IX. TRACKING DURATION TIME OF THE WHEELED MOBILE ROBOT

	Koide's Method	Yan's Method	Ours
Outdoor Case / s	50.77	18.67	64.74
Indoor Case / s	23.60	19.45	68.25

can track the target persistently when he moves from the target to the goal. Comparatively, Yan's method and Koide's method cannot completely achieve the person-following task in these two scenarios.

3) Case Study 2: We evaluate the performance of the developed method in the long-term person-following task in outdoor environments. The relative experiments have been recorded in the attached video[3]. A LiDAR sensor is mounted on the quadruped robot platform, and the robot is expected to follow the target person to travel through the campus. A snapshot of our campus and the tracking performance are shown in Fig. 16. The target is set to walk along a predefined path. We record the performance of the following function over this course. As indicated in Fig. 16(b), the robot can follow the target reliably, and even the quadruped robot platform is not stable with vibrations, which may introduce much noise to the point cloud data. More intuitively, we depict the trajectory of the robot when tracking the target on the campus, as shown by the cyan line in Fig. 16(c). It can be seen that the tracking trajectory is dense without any break, indicating the persistent tracking performance of our method. We also implement Koide's and Yan's methods in this outdoor scenario by using their tracking modules, but neither of these two methods can provide consistent tracking of the target person, which demonstrates the strength of our developed method.

We also test the reidentification module in outdoor experiments. Fig. 17 shows the performance with and without the reidentification module when the target is lost. It is obvious that the color of the target cube changes without the proposed reidentification module, while no color change occurs with



(a) (b) (c)

Fig. 16. Experiments in indoor environments using a quadruped robot. (a) Outdoor tracking scene. (b) Point cloud in the tracking procedure, where the red sphere box indicates the target and the blue cube indicates the photographer. (c) The corresponding point cloud map and the robot trajectory of the long-term person-following.

the module. The color of the cubes is relevant to the given ID in the detection module. When using the reidentification module, the color of the target cube is still orange when the target is captured again, which is shown in Fig. 17 (a) and Fig. 17 (c). However, when not using the reidentification module, the color changes from orange to light yellow, which means that the robot regards the target as a new man. Thus, our reidentification module is effective when the target is lost.

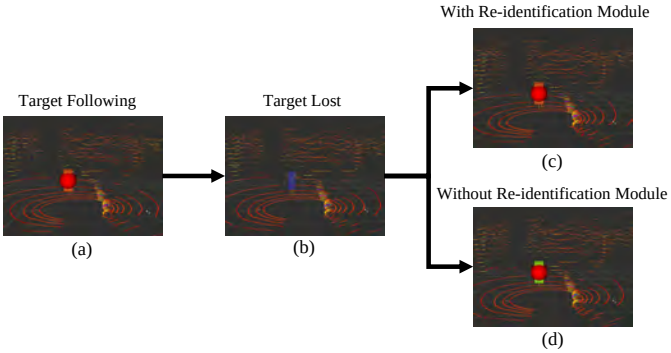


Fig. 17. The performance of our algorithm with and without the reidentification module. (a) is the point cloud before the target is lost. (b) is the point cloud when our algorithm fails to detect the target. (c) and (d) show the performance of our algorithm with and without the proposed reidentification module.

V. Conclusions and Future Work

This paper presents a person-following system using only the 3D LiDAR as the input sensor. The whole system consists of two components, pedestrian detection and pedestrian tracking. The pedestrian detection module incorporates a novel U-shaped CNN framework that specifically inputs a dedicated designed voxel feature. Taking only the sparse LiDAR point cloud as input, the proposed detection module is effective in complex environments and more accurate than the state of the art. The tracking module is employed to track all the pedestrians and successfully improves the robustness of person-following. The SVM-based target recapturing mechanism is triggered aperiodically to find the target when it is lost in complex scenarios. The developed framework is evaluated in real-world outdoor environments. The results demonstrate that, compared with other methods, our method is more effective in leading the robots robustly traveling through a large-scale environment.

Based on the realistic experimental results, we find that some limitations may exist in the developed framework. When facing a crowd of people in the environment, our system cannot precisely determine the number of people. Besides, the frequent and dense occlusions will greatly affect the detection results. In the future, we will integrate the camera sensor into our person-following framework. The visual information is expected to improve the detection accuracy and the reidentification effectiveness of our framework.

References

[1] M. Häselich, B. Jöbgen, N. Wojke, J. Hedrich, and D. Paulus, "Confidence-based pedestrian tracking in unstructured environments using 3d laser distance measurements," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 4118–4123.

[2] Z. Yan, T. Duckett, and N. Bellotto, "Online learning for 3d lidar-based human detection: experimental analysis of point cloud clustering and classification methods," *Autonomous Robots*, vol. 44, no. 2, pp. 147–164, 2020.

[3] L. E. Navarro-Serment, C. Mertz, and M. Hebert, "Pedestrian detection and tracking using three-dimensional ladar data," *The International Journal of Robotics Research*, vol. 29, no. 12, pp. 1516–1528, 2010.

[4] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[5] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," 2017.

[6] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.

[7] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[8] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers, "Tracking multiple moving targets with a mobile robot using particle filters and statistical data association," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 2. IEEE, 2001, pp. 1665–1670.

[9] A. Leigh, J. Pineau, N. Olmedo, and H. Zhang, "Person tracking and following with 2d laser scanners," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 726–733.

[10] A. Bastida-Castillo, C. D. Gómez-Carmona, E. De la Cruz-Sánchez, X. Reche-Royo, S. J. Ibáñez, and J. Pino Ortega, "Accuracy and inter-unit reliability of ultra-wide-band tracking system in indoor exercise," *Applied Sciences*, vol. 9, no. 5, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/5/939

[11] Q. Tian, K. I.-K. Wang, and Z. Salcic, "A low-cost ins and uwb fusion pedestrian tracking system," *IEEE Sensors Journal*, vol. 19, no. 10, pp. 3733–3740, 2019.

[12] C. Wu, B. Tao, H. Wu, Z. Gong, and Z. Yin, "A uhf rfid-based dynamic object following method for a mobile robot using phase difference information," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–11, 2021.

[13] M. Perdoch, D. M. Bradley, J. K. Chang, H. Herman, P. Rander, and A. Stentz, "Leader tracking for a walking logistics robot," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 2994–3001.

[14] H. Zhang, H. Liu, L. Deng, P. Wang, X. Rong, Y. Li, B. Li, and H. Wang, "Leader recognition and tracking for quadruped robots," in *2018 IEEE International Conference on Information and Automation (ICIA)*, 2018, pp. 1438–1443.

[15] L. Pang, Z. Cao, J. Yu, P. Guan, X. Rong, and H. Chai, "A visual leader-following approach with a t-d-r framework for quadruped robots," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 4, pp. 2342–2354, 2021.

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 21–37.

[17] Y. Liu, Q. Wang, H. Hu, and Y. He, "A novel real-time moving target tracking and path planning system for a quadrotor uav in unknown unstructured outdoor scenes," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2362–2372, 2019.

[18] T. Naseer, J. Sturm, and D. Cremers, "Followme: Person following and gesture recognition with a quadrocopter," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 624–630.

[19] L. Spinello, M. Luber, and K. O. Arras, "Tracking people in 3d using a bottom-up top-down detector," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1304–1310.

[20] C. Álvarez-Aparicio, Á. M. Guerrero-Higueras, F. J. Rodríguez-Lera, J. Ginés Clavero, F. Martín Rico, and V. Matellán, "People detection and tracking using lidar sensors," *Robotics*, vol. 8, no. 3, p. 75, 2019.

[21] H. A. Patel and D. G. Thakore, "Moving object tracking using kalman filter," *International Journal of Computer Science and Mobile Computing*, vol. 2, no. 4, pp. 326–332, 2013.

[22] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional lidar-based system for long-term and wide-area people behavior measurement," *International Journal of Advanced Robotic Systems*, vol. 16, 02 2019.

[23] K. Okuma, A. Taleghani, N. d. Freitas, J. J. Little, and D. G. Lowe, "A boosted particle filter: Multitarget detection and tracking," in *European conference on computer vision*. Springer, 2004, pp. 28–39.

[24] J. Yuan, H. Chen, F. Sun, and Y. Huang, "Multisensor information fusion for people tracking with a mobile robot: A particle filtering approach," *IEEE Transactions on Instrumentation and Measurement*, vol. 64, no. 9, pp. 2427–2442, 2015.

[25] Z. Hao, S. Yu, X. Yang, F. Zhao, R. Hu, and Y. Liang, "Online ls-svm learning for classification problems based on incremental chunk," in *International Symposium on Neural Networks*. Springer, 2004, pp. 558–564.

[26] E. Dilmen and S. Beyhan, "An enhanced online ls-svm approach for classification problems," *Soft Computing*, vol. 22, no. 13, pp. 4457–4475, 2018.

[27] J. Gai, L. Tang, and B. Steward, "Plant recognition through the fusion of 2d and 3d images for robotic weeding," in *2015 ASABE Annual International Meeting*. American Society of Agricultural and Biological Engineers, 2015, p. 1.

[28] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3d point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, 2008.

[29] B. Kulis and M. I. Jordan, "Revisiting k-means: New algorithms via bayesian nonparametrics," 2012.

[30] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China: IEEE, May 9-13 2011.

[31] C. Premebida, O. Ludwig, and U. Nunes, "Exploiting lidar-based features on pedestrian detection in urban scenarios," in *2009 12th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2009, pp. 1–6.

[32] K. Kidono, T. Miyasaka, A. Watanabe, T. Naito, and J. Miura, "Pedestrian recognition using high-definition lidar," in *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 405–410.

[33] L. E. Navarro-Serment, C. Mertz, and M. Hebert, "Pedestrian detection and tracking using three-dimensional ladar data," *The International Journal of Robotics Research*, vol. 29, no. 12, pp. 1516–1528, 2010.

[34] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[35] Y. Zhou, W. Huang, P. Dong, Y. Xia, and S. Wang, "D-unet: a dimension-fusion u shape network for chronic stroke lesion segmentation," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 18, no. 3, pp. 940–950, 2019.

[36] J. Li, Z. Pan, Q. Liu, and Z. Wang, "Stacked u-shape network with channel-wise attention for salient object detection," *IEEE Transactions on Multimedia*, vol. 23, pp. 1397–1409, 2020.

[37] Z. Huang, J. Wu, and F. Xie, "Automatic surface defect segmentation for hot-rolled steel strip using depth-wise separable u-shape network," *Materials Letters*, vol. 301, p. 130271, 2021.

[38] P. Konstantinova, A. Udvarev, and T. Semerdjiev, "A study of a target tracking algorithm using global nearest neighbor approach," in *Proceedings of the International Conference on Computer Systems and Technologies (CompSysTech'03)*, 2003, pp. 290–295.

[39] S. S. Blackman, "Multiple-target tracking with radar applications," *Dedham*, 1986.

[40] S. S. Keerthi and C.-J. Lin, "Asymptotic behaviors of support vector machines with gaussian kernel," *Neural computation*, vol. 15, no. 7, pp. 1667–1689, 2003.

[41] C. Glennie, A. Kusari, and A. Facchin, "Calibration and stability analysis of the vlp-16 laser scanner." *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 40, 2016.

[42] Z. Yan, T. Duckett, and N. Bellotto, "Online learning for human classification in 3D LiDAR-based tracking," in *In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, September 2017, pp. 864–871.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[44] R. Hou, C. Chen, and M. Shah, "Tube convolutional neural network (t-cnn) for action detection in videos," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5822–5831.

[45] N. Messina, G. Amato, F. Carrara, C. Gennaro, and F. Falchi, "Solving the same-different task with convolutional neural networks," *Pattern Recognition Letters*, vol. 143, pp. 75–80, 2021.

[46] K. Koide, J. Miura, and E. Menegatti, "Monocular person tracking and identification with on-line deep feature selection for person following robots," *Robotics and Autonomous Systems*, vol. 124, p. 103348, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889019302891

[47] A. Leigh, J. Pineau, N. Olmedo, and H. Zhang, "Person tracking and following with 2d laser scanners," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 726–733.

[48] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.