electronics

MDPI

Article

Pedestrian Detection with Lidar Point Clouds Based on Single Template Matching

Kaiqi Liu, Wenguang Wang * and Jun Wang *

School of Electronic and Information Engineering, Beihang University, Beijing 100191, China
* Correspondence: wwenguang@buaa.edu.cn (W.W.); wangj203@buaa.edu.cn (J.W.);
Tel.: +86-10-8231-7240 (W.W.); +86-10-8233-9767 (J.W.)

Received: 7 June 2019; Accepted: 8 July 2019; Published: 11 July 2019

Abstract: In the field of intelligent transportation systems, pedestrian detection has become a problem that is urgently in need of a solution. Effective pedestrian detection reduces accidents and protects pedestrians from injuries. A pedestrian-detection algorithm, namely, single template matching with kernel density estimation clustering (STM-KDE), is proposed in this paper. First, the KDE-based clustering method is utilized to extract candidate pedestrians in point clouds. Next, the coordinates of the point clouds are transformed into the pedestrians' local coordinate system and projection images are generated. Locally adaptive regression kernel features are extracted from the projection image and matched with the template features by using cosine similarity, based on which pedestrians are distinguished from other columnar objects. Finally, comparative experiments using KITTI datasets are conducted to verify pedestrian-detection performance. Compared with the STM with radially bounded nearest neighbor (STM-RBNN) algorithm and the KDE-based pedestrian-detection algorithm, the proposed algorithm can segment gathering pedestrians and distinguish them from other columnar objects in real scenarios.

Keywords: Lidar; pedestrian detection; point clouds; template matching

1. Introduction

With the development of intelligent transportation systems, environmental perception is becoming increasingly significant in the fields of advanced driver assistance systems (ADAS) and autonomous vehicles [1]. Considering that there is a large number of traffic accidents every year, pedestrian detection has become an urgent challenge in need of a solution. Pedestrians are vulnerable in traffic. Millions of people in the world are killed or injured by traffic accidents every year (https://www.who.int/violence_injury_prevention/road_safety_status/2015/zh/). To some extent, effective pedestrian detection would reduce traffic accidents to protect pedestrians from vehicle injuries. Meanwhile, detecting the position and movement of a pedestrian has a great effect on the motion planning of autonomous vehicles.

There are dynamic and stationary pedestrians on road scenarios, in which their motion and position are random. Compared with vehicles, pedestrian profile sizes are smaller. In addition, pedestrians in these scenarios often gather together, which makes it difficult to accurately segment them. All these factors make pedestrian detection difficult.

In the field of pedestrian detection, the camera is one of the most commonly used sensors. It has the closest structure to that of human eyes and can directly reflect the world how humans perceive it. In optical images, it is easy to obtain color, texture and contour features of targets. These features, combined with learning methods, can classify objects in road scenarios. Dalal proposed the histogram of oriented gradient (HOG) feature for pedestrian detection [2]. HOG is insensitive to light and shadows, which widely interests researchers [3]. Scale-invariant feature transform (SIFT) and Haar-like features are also commonly used in pedestrian detection [4,5]. For autonomous vehicles, determining the position of pedestrians is significant to designate driving strategies. However, it is difficult to obtain accurate pedestrian-position information with monocular images.

A Lidar sensor can obtain the contour, position and distance information of objects, which makes it an excellent sensor in 3D environment perception [6,7]. Generally, pedestrian-detection methods with Lidar can be divided into three categories: training-based methods, foreground and background segmentation methods and kernel density estimation (KDE)-based methods.

1.1. Training-Based Methods

Features play an important role in training-based methods. A pedestrian classification system with three features, including the total number of points in a cluster, distance to the Lidar and spatial distribution direction of clustered points, was utilized in Reference [8]. The backpropagate autoencoder-artificial neural network was trained for the classification. An AdaBoost classifier, combined with a radius feature, a convexity feature, a local minimum feature and a robust compactness feature were utilized to detect pedestrians in Reference [9]. Fifteen unique features of Lidar point clouds, such as the number of points, the normalized Cartesian dimension and the central moment, were used to detect pedestrians in urban scenarios [10]. The segmentation method in Reference [11] was used for point-cloud segmentation in Reference [12]. Thirty-one features and a radial basis fuction additive kernel support vector machine (SVM) were also used for pedestrian detection. Three-dimensional features from point clouds and two-dimensional features from projection images were extracted to classify pedestrians with a radial basis function kernel SVM [13]. In order to detect pedestrians in a long-range area with sparse point clouds, radial basis function-based interpolation was implemented to robustly extract features and SVMs are applied to learn the classifiers [14].

In general, training-based methods perform well in pedestrian detection with a large number of training data but complex and changeable real scenarios make it difficult to obtain sufficient negative training samples, which limit the performance of these methods.

1.2. Foreground and Background Segmentation Methods

In addition to directly extracting pedestrians from scenarios, there is also a kind of method based on the foreground and background segmentation of point clouds. "Background" refers to fixed ground, buildings, trees and stationary pedestrians and vehicles. "Foreground" refers to moving objects such as cyclists and moving pedestrians and vehicles. Fixed sensors are required in these methods, which are common in monitoring scenarios or when the vehicle is stationary at the roadside or an intersection. A visual monitoring system that passively observes moving objects was proposed in Reference [15]. Over time, the pixel values of the particular pixel in the background obey adaptive Gaussian distribution. With illumination changing, the background pixel values obey the adaptive mixture of Gaussians. After the background model is constructed, pixel values that do not satisfy the distribution of the background pixel values are regarded as moving foreground pixels and foreground targets are obtained by morpholological processing. A similar segmentation method was applied in Lidar monitoring scenarios [16] and a point-cloud foreground detection method based on the dynamic Markov model was proposed to detect moving pedestrians.

Although this kind of method can adaptively detect background and foreground pixels, it has some drawbacks. First, the utilized sensors in this method should be fixed in the scenarios in order to segment foreground and background. If the sensors have movements, the background information is changed, which makes it difficult to collect a large number of data to establish the background model. Second, this method needs to accumulate the pixel values in each pixel, which demands much computation and many computing resources.

1.3. KDE-Based Method

In the field of pedestrian detection with Lidar, single-line Lidar has been applied to perceive the environment earlier. In Reference [17], four single-line Lidars were constructed in the scenarios whose height was set to sixteen centimeters in order to detect the legs of pedestrians with KDE. With the development of Lidar sensors, 2.5D and 3D Lidar have attracted increasing attention. In Reference [18], a fusion method based on KDE was proposed for pedestrian detection. The geometric features of pedestrians and the returned layer numbers are applied to determine their positions.

The KDE-based method can effectively detect upright pedestrians in a scenario but it is difficult for it to accurately distinguish columnar objects with similar sizes to pedestrians, such as mailboxes and guideboards.

The main objective of the paper is to propose a pedestrian detection system using Lidar point clouds with the ability to detect gathering pedestrians and avoid the training process of a large number of data, thereby improving the ability of the intelligent transportation system in environmental perception. In this work, our contributions are demonstrated by the detection system named single template matching with kernel density estimation clustering (STM-KDE). Compared with the existing methods, STM-KDE can well segment the gathering pedestrians with the KDE-based clustering, which ignores the problem that the point clouds are distributed uneven but only focuses on the accumulation of the point clouds at the object's upright position. Meanwhile, the system does not need a large amount of data for model training and only a single pedestrian template is needed for detecting pedestrians in the scenarios. In the end, the qualitative and quantitative experiments validate the performance of the proposed detection system. In the range of 15 m, by considering the detection results and the false alarms, the F_1 score of STM-KDE exceeds the comparative methods by 4% and 19%, respectively. Meanwhile, the proposed detection method does not improve the computational complexity on the basis of improving the detection efficiency compared with existing method.

The remainder of the paper is organized as follows. In Section 2, the implementation process of the proposed STM-KDE detection system is depicted. The experiments are shown in Section 3, in which two KITTI datasets are employed. Finally, concluding remarks and the contributions are given in Section 4.

2. Materials and Methods

With the influence of environmental interference and measurement noise, it is difficult to distinguish pedestrians from objects with similar shapes. In addition, pedestrians in road scenarios often appear in groups. The segmentation performance of point clouds is poor within a short distance. In order to solve these problems, a pedestrian-detection method based on STM-KDE is proposed. The method uses one template to match objects in scenarios and detect pedestrians. It should be noted that pedestrian detection in this paper includes pedestrians and cyclists, because the state of people while cycling is similar to that of walking and, compared with the human body, the number of point clouds scanned by Lidar on the bicycle is smaller, which can be ignored. The detection flowchart is shown in Figure 1.

As shown in Figure 1, there are three main processes in the proposed detection algorithm.

1. Point-cloud preprocessing. This process includes ground segmentation and grid filtering, which is mainly utilized to reduce the number of point clouds and improve the efficiency of the algorithm.
2. Pedestrian clustering based on KDE, which is employed to extract candidate pedestrians from point clouds with size limitation.
3. Template matching. Three-dimensional point clouds are projected onto the 2D plane, from which contour features are extracted. Cosine Similarity between the features of the template and the projection image is calculated for pedestrian detection.

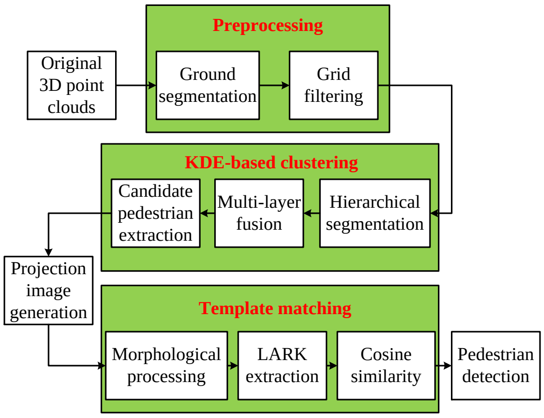


Figure 1. Flowchart of pedestrian detection.

2.1. Preprocessing

In the field of autonomous vehicles, Velodyne HDL-64E [19,20] is a commonly used Lidar with 64 laser emitters and 64 receivers, which is equivalent to using 64 laser lines to scan the environment. Every second, Velodyne HDL-64E receives millions of point clouds, half of which are returned from the ground. Ground point clouds not only increase computation but also interfere with target detection. The purpose of point-cloud preprocessing is to reduce the point clouds to be processed and improve the efficiency of the subsequent detection. In this paper, preprocessing mainly consists of two processes: ground segmentation and grid filtering. In general, ground modeling is utilized to segment point clouds into ground points and nonground points [19,21–23]. Grid filtering [20] filters outliers and tall structural objects that are obviously nonhuman beings. Grid filtering is implemented according to the height information of point clouds that fall into the same grid. The detailed process is as follows.

First, nonground point clouds are projected onto a grid map that is evenly divided along the x and y directions. The points in grid cell (i, j) are expressed as $\mathbf{P}_{ij} = \left\{ \mathbf{p}_1^{ij}, ..., \mathbf{p}_\lambda^{ij} \right\}$, where λ is the number of point clouds in grid cell (i, j). The maximum height, minimum height and mean height of the points in grid (i, j) are denoted as z_{ij}^{max}, z_{ij}^{min} and z_{ij}^{mean}, respectively. Next, the point clouds are filtered by the following criteria.

1. Generally, points that are returned by a human are dense. If the point number contained in a grid cell is smaller than a preset value, these points may come from the measurement noise, mis-segmented ground points and other interference objects. Therefore, these points are filtered out.
2. If the height difference of the point clouds in a grid cell that can be calculated with z_{ij}^{max} and z_{ij}^{min} is small and the average height z_{ij}^{mean} of the point clouds is small, these points may come from roads or low obstacles and are filtered out.
3. If the height difference of the point clouds in a grid cell is large or the maximum height z_{ij}^{max} is big, these points are likely to come from tall buildings and are therefore filtered out.

2.2. Clustering Based on KDE

Before detecting, it is necessary to determine the detection region of interest. Sliding windows in point clouds is a useful way to extract objects in training-based pedestrian-detection methods [24]. However, searching with sliding windows in whole point clouds results in a large amount of computation. The distance relationship between point clouds can also be utilized to perform 3D clustering [25,26]. The purpose of clustering is to segment point clouds based on their characteristics. For example, in Reference [25], a radially bounded nearest neighbor (RBNN) algorithm is proposed to cluster the point clouds. RBNN is easy to implement and segments point clouds based on their relative spatial position. However, RBNN uses a fixed spherical radius to search for neighbor points. For adjacent objects, such as multiple pedestrians gathering together, its segmentation performance is poor. Undersegmentation occurs and makes multiple pedestrians group into one segmentation. To solve the above problems, a clustering method based on KDE is proposed to extract candidate pedestrians.

2.2.1. Hierarchical Segmentation

Hierarchical segmentation uses scanning breakpoints in each layer to segment the point clouds. The distance between adjacent points $\mathbf{p}_{i-1}$ and $\mathbf{p}_i$ is compared with the threshold in Equation (1).

$$\eta_{seg}{=}\varepsilon \times r \times \sin(\alpha) \tag{1}$$

where ε is a constant, r denotes the distance between the point and Lidar and α expresses the horizontal scanning angular resolution of the Lidar. If the distance between $\mathbf{p}_{i-1}$ and $\mathbf{p}_i$ is smaller than η_{seg}, points $\mathbf{p}_{i-1}$ and $\mathbf{p}_i$ belong to the same segment. On the contrary, they are used as the end point and start point of two segments. Subsequently, the above process is implemented on adjacent points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. After comparing the distances in the scanning layer in order, the hierarchical-segmentation results of all scanning layers $\mathbf{C}$ are obtained.

Next, geometric conditions are utilized to extract the candidate pedestrian segment. As shown in Equation (2), $l_{\mathbf{C}_i}$ and $w_{\mathbf{C}_i}$ represent the length and width of the segment, respectively. η_{ped} expresses the pedestrian-size that is applied to extract the candidate pedestrian segmentation in single layer.

$$\mathbf{C}_p = \{\mathbf{C}_i | l_{\mathbf{C}_i} \leq \eta_{ped}, w_{\mathbf{C}_i} \leq \eta_{ped}, \mathbf{C}_i \in \mathbf{C}\} \tag{2}$$

The clustering process is shown in Algorithm 1.

Algorithm 1 Hierarchical segmentation.

Input: $\mathbf{P} = (\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_\kappa)^T, \varepsilon, \alpha, \eta_{ped}$;
$\mathbf{C} = \emptyset, \mathbf{C}_p = \emptyset, idx_{start} = 1$;
for $i = 2, ..., \kappa$ **do**
  $dis_i = \|\mathbf{p}_i - \mathbf{p}_{i-1}\|$;
  $\eta^i_{seg} = \varepsilon \cdot \|\mathbf{p}_i\| \cdot \sin(\alpha)$;
  if $dis_i > \eta^i_{seg}$ **then**
    $\mathbf{C}_s = \{\mathbf{p}_x | x = idx_{start}, ..., i-1\}$;
    $\mathbf{C} = \mathbf{C} \cup \mathbf{C}_s$;
    $idx_{start} = i$;
  end if
end for
$\mathbf{C}_s = \{\mathbf{p}_x | x = idx_{start}, ..., \kappa\}$;
$\mathbf{C} = \mathbf{C} \cup \mathbf{C}_s$;
for $\mathbf{C}_i \in \mathbf{C}$ **do**
  if $l_{\mathbf{C}_i} < \eta_{ped}$ and $w_{\mathbf{C}_i} < \eta_{ped}$ **then**
    $\mathbf{C}_p = \mathbf{C}_p \cup \mathbf{C}_i$;
  end if
end for
Output: A set of candidate pedestrian clusters $\mathbf{C}_p$ in each scanning layer

2.2.2. Multilayer Fusion

Generally, pedestrians in road scenarios remain upright. The body of the pedestrian returns multiple scanning layers. The accumulation of scanning points at the position of pedestrians is larger than at the position of noise and other environmental objects. Therefore, KDE is utilized to calculate accumulation probability. In each scanning layer, Gaussian distribution is established in geometric center $\mathbf{c}_p = (x_p, y_p)$ of candidate pedestrian cluster $\mathbf{C}_p$. Clusters in multiple scanning layers are fused to determine whether $\mathbf{C}_p$ is a disturbance in a single scanning layer or a columnar object that exists at a similar position in each scanning layer. The probability function is

$$p(\mathbf{c}) = \frac{1}{N_s} \sum_{p=1}^{n} \varphi(\mathbf{c} - \mathbf{c}_p, w) \tag{3}$$

$$N_s = \sum_{p=1}^{l_N} N_p \tag{4}$$

$$N_p = \begin{cases} 1, & if\ 0 < h_{lidar} + r \times \tan(\phi_p) < h_{ped} \\ 0, & else \end{cases} \tag{5}$$

where $\mathbf{c} = (x, y)$ denotes the coordinates of the points in the $x - y$ plane, n denotes the number of clusters in $\mathbf{C}_p$, w is the window length that depends on pedestrian size and N_s can be regarded as the theoretical number of scanning layers returned by a pedestrian. l_N is the total number of scanning layers. $\Phi = \{\phi_1, ..., \phi_N\}$ represents the pitch angle corresponding to each scanning layer. h_{ped} denotes the average pedestrian height. In Equation (3), $\varphi(\cdot)$ denotes the probability density function of Gaussian distribution.

$$\varphi(\mathbf{c}, w) = \frac{1}{(2\pi)^{d/2} w^d \left|\Sigma\right|^{1/2}} \exp(-\frac{\mathbf{c}^T \Sigma^{-1} \mathbf{c}}{2w^2}) \tag{6}$$

where Σ is a covariance matrix and the identity matrix is utilized in this paper [17]. It is considered that the coefficient in Equation (6) is a constant; the equation can be simplified to the following form.

$$p(\mathbf{c}) = \frac{1}{N_s} \sum_{i=1}^{n} \exp(-\frac{(\mathbf{c} - \mathbf{c}_i)^T (\mathbf{c} - \mathbf{c}_i)}{2w^2}) \tag{7}$$

After fusing multiple layers, a local extreme value appears at the position of upright pedestrians. The mean shift algorithm in Reference [27] was utilized to extract the position of the local maxima.

It is assumed that the obtained positions of the m local maximum values are expressed by $\mathbf{P}_{peak} = \{\mathbf{p}_1, ..., \mathbf{p}_m\}$, $\mathbf{p}_i = (f_{pi}, x_{pi}, y_{pi})$, where f_{pi} is the probability of the ith extremum, whose position is represented by (x_{pi}, y_{pi}). The positions of candidate pedestrians are selected from the positions of probability values that are greater than a preset threshold. The positions of the selected γ maximum extremums are denoted by $\mathbf{P}_p = \left\{\mathbf{p}_1, ..., \mathbf{peak}_\gamma\right\}$. With the local maximum position as the center, the point clouds of candidate pedestrian $\mathbf{c}_{ped}$ are extracted according to the probable pedestrian size. All candidate pedestrian clusters are expressed as $\mathbf{C}_{ped} = \left\{\mathbf{c}_{ped}^1, ..., \mathbf{c}_{ped}^\gamma\right\}$.

2.3. Template Matching

2.3.1. Projection-Image Generation

For each candidate pedestrian cluster $\mathbf{c}_{ped}^i$ in $\mathbf{C}_{ped}$, principal component analysis (PCA) [28] was implemented to determine the main plane and local coordinate system. The plane determined by eigenvectors corresponding to the two large eigenvalues is defined as the main plane of the pedestrian point clouds. In these two eigenvectors, the eigenvector that has the smallest angle with vector $(0, 0, 1)$ is defined as the z' axis and another is defined as the y' axis. The eigenvector corresponding to the smallest eigenvector is the x' axis. As shown in Figure 2, $Oxyz$ is the Lidar coordinate system. $O'x'y'z'$ is the local coordinate system in which the pedestrian is located. $O'y'z'$ denotes the main plane of the pedestrian. $Ox'y'z'$ is the coordinate system that has the same origin as $Oxyz$ and each coordinate axis of $Ox'y'z'$ is parallel to coordinate system $O'x'y'z'$. (x, y, z), (x', y', z') and (x'', y'', z'') represent the coordinates in $Oxyz$, $O'x'y'z'$ and $Ox'y'z'$, respectively. The coordinate transformation from Lidar coordinate system $Oxyz$ to local coordinate system $O'x'y'z'$ is as follows [14].

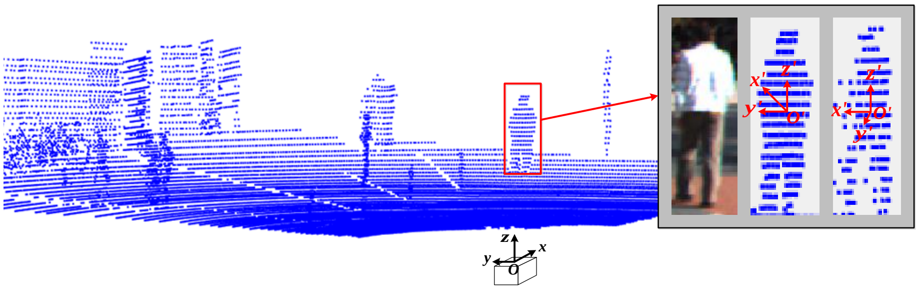


Figure 2. Pedestrian-detection coordinates. $Oxyz$, Lidar coordinate system; $O'x'y'z'$, local coordinate system.

$$\begin{pmatrix} x'' \\ y'' \\ z'' \end{pmatrix} = \begin{pmatrix} \cos(\psi_{ox',ox}) & \cos(\psi_{ox',oy}) & \cos(\psi_{ox',oz}) \\ \cos(\psi_{oy',ox}) & \cos(\psi_{oy',oy}) & \cos(\psi_{oy',oz}) \\ \cos(\psi_{oz',ox}) & \cos(\psi_{oz',oy}) & \cos(\psi_{oz',oz}) \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{8}$$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x'' - O'_x \\ y'' - O'_y \\ z'' - O'_z \end{pmatrix} \tag{9}$$

where ψ denotes the angle between the two axes; and O'_x, O'_y and O'_z represent the x, y and z coordinates of point O', which is determined by the geometric center of the candidate pedestrian cluster.

After transforming the coordinates of clusters in the Lidar coordinate system to the local coordinate system, the point clouds are projected onto the main plane. As shown in Figure 2, $O'y'z'$ is the main plane, in which the grid is divided with a fixed length in the direction of the y' and z' axes. In each grid, the minimum distance of the points to the main plane is regarded as the pixel value. If there are no points in the grid, the pixel value is set to 0. Figure 3c is the projection result of point clouds in Figure 3b.

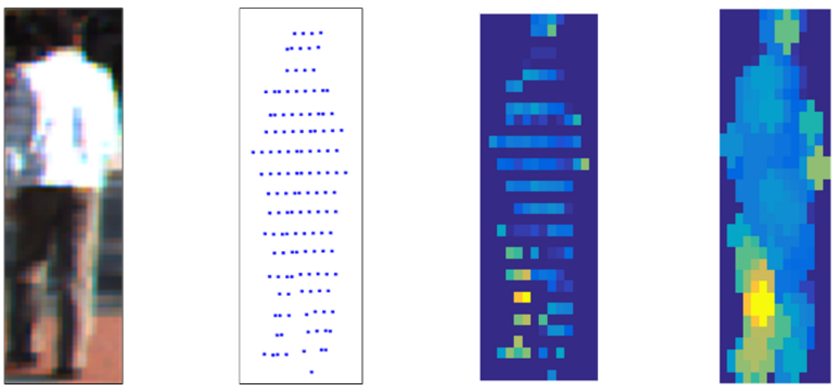


Figure 3. Projection-image generation. (**a**) Pedestrian optical image; (**b**) corresponding point clouds; (**c**) projection image generated by projecting point clouds onto main plane; (**d**) result of morphological processing on (**c**).

Since the Lidar discretely collects 3D points , the projection image obtained in Figure 3c is a discrete range image. The discrete image is filled into a complete contour image by using morphological processing, such as morphological expansion and hole filling. Subsequent feature extraction and template matching are implemented in the projection image like Figure 3d.

2.3.2. Feature Extraction

As shown in Figure 3d, the projection image of pedestrian point clouds shows a clear pedestrian outline that can clearly distinguish the shape of the pedestrian's head, waist, arms and legs by visual observation. The features from the contour of objects can be extracted to build the template for matching. The locally adaptive regression kernel (LARK) feature can estimate the contour of an object by calculating the similarity of the current pixel with its surrounding pixels. It has achieved success in the field of face detection [29,30]. The LARK feature of pixel $\mathbf{x}$ is calculated as follows.

$$\mathbf{K}(\mathbf{x}_l - \mathbf{x}, \Sigma_l) = \frac{\sqrt{\det(\Sigma_l)}}{2\pi h^2} \exp(-\frac{(\mathbf{x}_l - \mathbf{x})^T \Sigma_l (\mathbf{x}_l - \mathbf{x})}{2h^2}), \ \ l = 1, ..., P^2 \tag{10}$$

The local window is established with $\mathbf{x}$ as its center and P as the length of the window. There are P^2 pixels in total in the local window. The pixel coordinates in the local window are denoted as $\mathbf{x}_l = (x_1, x_2)^T$, h denotes the smoothing parameter, Σ_l represents the covariance matrix calculated by the spatial gradient vectors in a local window of $\mathbf{x}$. The calculation of Σ_l is as follows [31].

$$\Sigma_l = \begin{pmatrix} \sum\limits_{x_j \in \mathbf{w}_l} g_{\mathbf{x}_1}(\mathbf{x}_j) g_{\mathbf{x}_1}(\mathbf{x}_j) & \sum\limits_{x_j \in \mathbf{w}_l} g_{\mathbf{x}_1}(\mathbf{x}_j) g_{\mathbf{x}_2}(\mathbf{x}_j) \\ \sum\limits_{\mathbf{x}_j \in \mathbf{w}_l} g_{\mathbf{x}_2}(\mathbf{x}_j) g_{\mathbf{x}_1}(\mathbf{x}_j) & \sum\limits_{\mathbf{x}_j \in \mathbf{w}_l} g_{\mathbf{x}_2}(\mathbf{x}_j) g_{\mathbf{x}_2}(\mathbf{x}_j) \end{pmatrix} \tag{11}$$

where $\mathbf{w}_l$ represents the local window of $\mathbf{x}_l$. $g_{\mathbf{x}_1}(\cdot)$ and $g_{\mathbf{x}_2}(\cdot)$ are the first derivatives in the x and y directions.

For each pixel in the projection image, Equations (10) and (11) are utilized to extract LARK features $\mathbf{K}_i \in \mathbb{R}^{P \times P}$. Subsequently, the LARK feature is normalized as a column vector $\mathbf{l}_i \in \mathbb{R}^{P^2 \times 1}$. The LARK vectors of all pixels constitute the LARK matrix of image

$$\mathbf{L} = [\mathbf{l}_1, ..., \mathbf{l}_i, ..., \mathbf{l}_M] \in \mathbb{R}^{P^2 \times M}, \tag{12}$$

where M is the pixel number of the projection image.

2.3.3. Cosine Similarity

The angle between the two vectors can be used to describe the difference between them. After LARK feature extraction, cosine similarity is utilized to characterize the matching degree between the template feature matrix and the feature matrix of the candidate pedestrian. Cosine similarity is widely used in many fields [32,33]. The cosine similarity of LARK matrices is defined as follows

$$\rho(\mathbf{L}_{test}, \mathbf{L}_{temp}) = \sum_{i=1}^{M} \frac{(\mathbf{l}_{test}^i)^T \mathbf{l}_{temp}^i}{\|\mathbf{L}_{test}\|_F \, \|\mathbf{L}_{temp}\|_F}, \tag{13}$$

where $\mathbf{L}_{test}$ is the LARK feature matrix of the candidate pedestrian and $\mathbf{L}_{temp}$ is the LARK feature matrix of the template. M is the column number of the feature matrix, which is the number of pixels in Equation (12). $\mathbf{l}_{test}^i$ and $\mathbf{l}_{temp}^i$ are the ith column vector of the corresponding kernel matrix, respectively. $\|\cdot\|_F$ denotes the Frobenius norm.

2.4. Pedestrian Detection

A pedestrian-detection algorithm based on STM-KDE is proposed in this paper. The algorithm used only one template to detect pedestrians in the whole frame of point clouds without a large number of training data. The algorithm process is shown in Algorithm 2.

Algorithm 2 Pedestrian-detection method based on single template.

Input: $\mathbf{P}_w = (\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n)^T$, η_c, η_{det}, $\mathbf{L}_{temp}$;
1: $\mathbf{P}_a = \text{GroundFilter}(\mathbf{P}_w)$;
2: $\mathbf{P}_e = \text{GridFilter}(\mathbf{P}_a)$;
3: $\mathbf{C}_{ped} = \text{CandidatePedestrianExtract}(\mathbf{P}_e, \eta_c)$;
4: **for** $\mathbf{c}^i_{ped} \in \mathbf{C}_{ped}$ **do**
5: $\quad \mathbf{I}_i = \text{RangeImage}(\mathbf{c}^i_{ped})$;
6: $\quad \mathbf{L}^i_{test} = \text{LarkExtract}(\mathbf{I}_i)$;
7: $\quad \rho_i = \text{CosineSimilarity}(\mathbf{L}_{temp}, \mathbf{L}^i_{test})$;
8: $\quad$ **if** $\rho_i > \eta_{det}$ **then**
9: $\qquad \mathbf{c}^i_{ped} \in ped$;
10: $\quad$ **else**
11: $\qquad \mathbf{c}^i_{ped} \notin ped$;
12: $\quad$ **end if**
13: **end for**
Output: Pedestrians point clouds.

The inputs of the algorithm include: original point clouds $\mathbf{P}_w$, the LARK feature matrix of template $\mathbf{L}_{temp}$, threshold of clustering η_c and the threshold of template matching η_{det}. The output is the detected pedestrian point clouds. Before detection, a pedestrian cluster with a complete contour is manually extracted from the point clouds to establish the template, which is projected onto the main plane to construct the template projection image. The template matrix of LARK feature $\mathbf{L}_{temp}$ is extracted from the projection image. Then, ground segmentation and grid filtering in Lines 1 and 2 are implemented to filter out ground points and tall buildings. Next, the clustering method based on KDE is utilized to obtain candidate pedestrian $\mathbf{c}^i_{ped}$, which is shown in Line 3. After coordinate transforming and main-plane projection, the LARK feature matrix is extracted from the projection image with morphological processing. Cosine similarity is calculated to detect pedestrians with preset threshold η_{det}.

2.5. Comparing with Existing Methods

There are three important distinctions between the proposed STM-KDE algorithm and existing pedestrian-detection methods. First, while the KDE-based method in Reference [18] uses the KDE process twice to fuse the information for pedestrian detection, the proposed algorithm only uses KDE once to fuse the central information of the segmentation results in multiple scanning layers for pedestrian clustering, which reduces the amount of computation in this process.

Second, the single template-matching method was first utilized for face detection in optical images [29]. This paper applies this detection idea to pedestrian detection with point clouds and proposes the novel STM-KDE detection method that avoids the need for a large number of training data.

Third, compared with existing methods, the proposed algorithm solves the problem of neighboring pedestrian detection with the combination of KDE clustering and LARK contour feature matching, which avoids the problems caused by sliding-window processing or undersegmentation.

2.6. Computation Complexity

The complexity of the proposed STM-KDE detection method is analyzed in this section. The detection method mainly includes two stages: the candidate pedestrian extraction and pedestrian detection with single template.

For candidate pedestrian detection, it is assumed that there are totally n point clouds and m hierarchical segmentations with k point clouds in the scenario. The complexity of Hierarchical segmentation is $O(n)$. Since only center position is utilized to extract the candidate pedestrian, the complexity of KDE-based segmentation is $O(m), m < k < n$.

For pedestrian detection with single template, since the LARK feature is extracted for each pixel, therefore, the complexity of single template matching process is $r \times O(m)$ as r denotes the pixel number of the projection image. By integrating the computation of the two processes, the total complexity of the STM-KDE is $O(n) + O(m) + r \times O(m) = O(n)$.

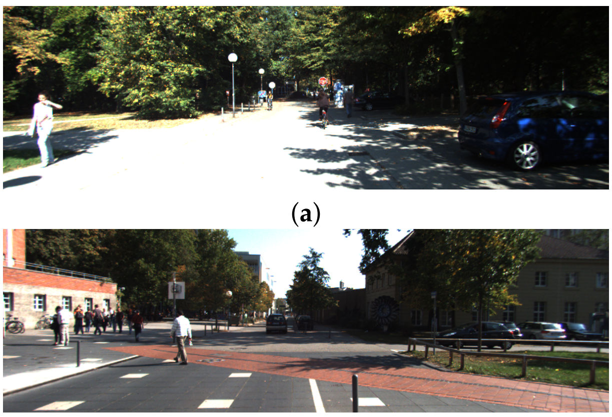
While in Reference [18], there are two-stage of KDE process and the complexity of the first-stage KDE process is the sum of $O(n)$ and $O(k)$. The complexity of the second-stage KDE is $O(c)$ with c is the number of selected cluster center in each scanning layer, $c < n$. Therefore, the total complexity of the KDE-based detection is $O(n) + O(k) + O(c) = O(n)$.

In conclusion, the total complexities of the two methods have the same order of magnitude. The proposed detection method does not improve the computational complexity comparing with existing method.

3. Results

KITTI datasets [34] were used to verify the detection efficiency of the proposed STM-KDE algorithm. The 0047 and 0016 datasets were used in the experiments. In the 0047 dataset, the vehicle was driving in the campus and the Lidar was on a mobile platform. The optical image of the scenario is shown in Figure 4a. The sensor platform in the 0016 dataset was static. The sensor was fixed at the road junction in the campus, where there was a large number of pedestrians. The scenario is shown in Figure 4b. Since these two datasets did not provide pedestrian ground truth, we manually labeled the position of each pedestrian. The 0047 dataset contained 31 frames of point clouds and a total of 118 pedestrians. The 0016 dataset contained 186 frames of point clouds and a total of 2174 pedestrians in the scenario.

Three experiments were implemented in the paper. The size of grid filtering l_{grid} was set to 0.2 m. Since the segmentation in Hierarchical segmentation is a rough selection process for candidate pedestrians, ε was set to 20. Thresholds in grid filtering η_n and pedestrian clustering η_{ped}, η_c were set to 4 and 0.8 m and 0.4 m, respectively, considering the distribution of the point clouds and the size of the pedestrians.



(**a**)

(**b**)

Figure 4. Scenario optical images: (**a**) 0047 and (**b**) 0016 dataset.

3.1. Clustering

In this section, some areas in the 0016 dataset were utilized to verify the efficiency of the proposed clustering based on KDE. The areas were close to the Lidar. The point clouds were relatively dense and many pedestrians gathered in these areas. In the original point-cloud coordinate system, the x and

y range of the selected areas were at $x \in [0, 20]$ (m) and $y \in [-7, 13]$ (m), respectively. Since there was no ground truth provided, the clustering results are qualitatively described.

Figure 5 shows the clustering results selected every 10 frames. The first column represents the original point clouds. The second column shows the RBNN clustering results, in which the radius of the sphere was set to 0.5 m. The third column is the results of the proposed KDE clustering. Only the extracted candidate-pedestrian clusters are shown in the results. In Figure 5, different types of point-cloud clusters are represented by different colors.

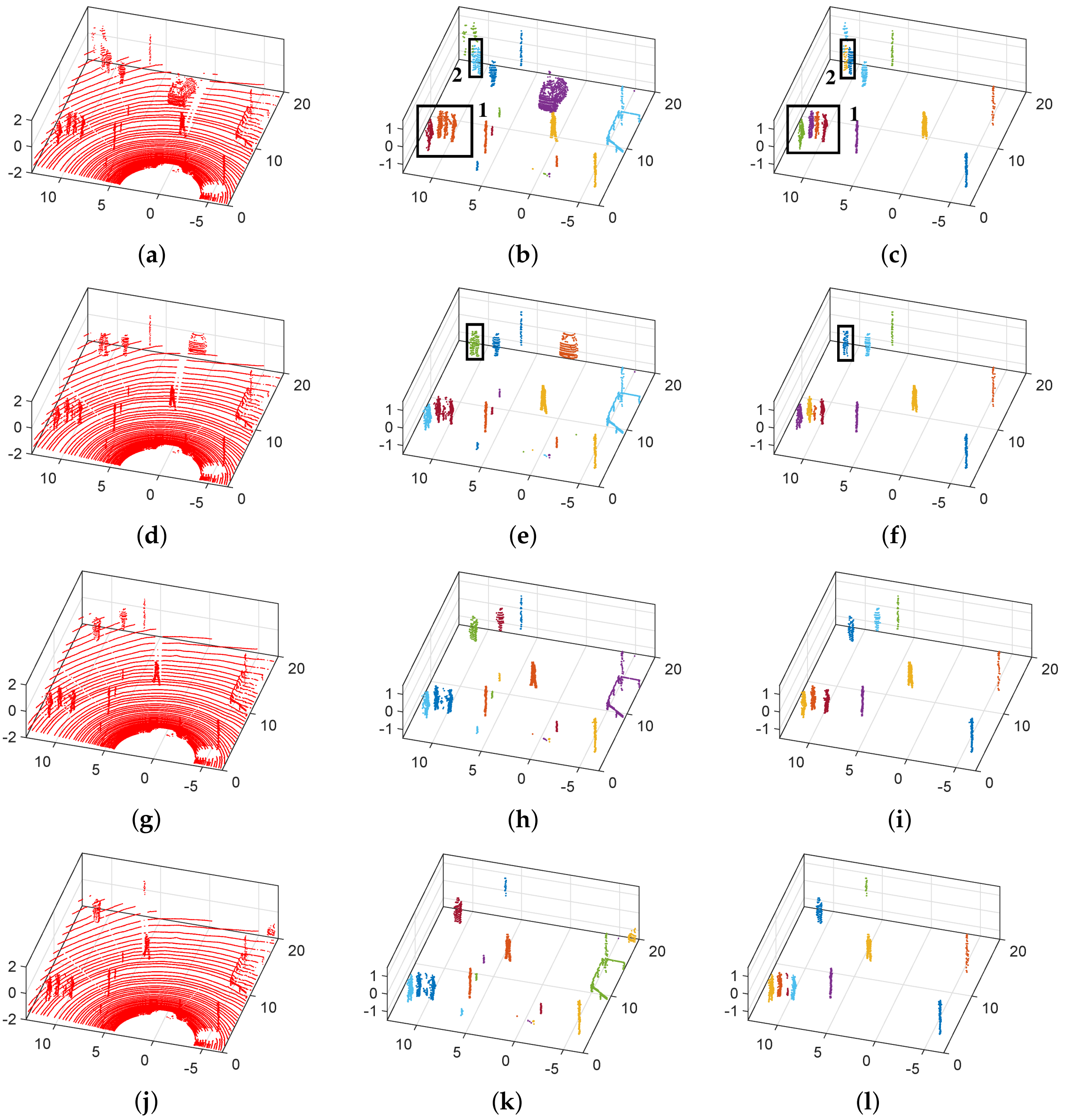


Figure 5. Clustering results. Images in the first column are the original point clouds. Results in the second column are clustering with radially bounded nearest neighbor (RBNN). The results in the third column are the clustering with kernel density estimation (KDE). (**a**) Original point clouds (Frame 1); (**b**) RBNN (Frame 1); (**c**) KDE (Frame 1); (**d**) original point clouds (Frame 11); (**e**) RBNN (Frame 11); (**f**) KDE (Frame 11); (**g**) original point clouds (Frame 21); (**h**) RBNN (Frame 21); (**i**) KDE (Frame 21); (**j**) original point clouds (Frame 31); (**k**) RBNN (Frame 31); (**l**) KDE (Frame 31).

As shown in Figure 5, the clustering results of the proposed KDE method are more accurate for the clustering of gathering pedestrians. In the black box 1 in Figure 5b,c, there are four pedestrians in total. With the RBNN algorithm, there are three pedestrians among them are undersegmented (in orange points). While the KDE-based clustering method could segment the four gathering pedestrians well. The similar results are shown in box 2. With KDE-based clustering, the two pedestrians are segmented well. However, it is worth noting that if the pedestrians stood within a close distance and there was occlusion in the scenario, neither algorithm could accurately segment the pedestrians, as shown in the black box of Figure 5e,f. Therefore, it can be seen that pedestrian size is small and at the same time occlusion often happens in the environment, which causes many difficulties for subsequent pedestrian detection.

3.2. Template Matching

The density of Lidar point clouds decreases with the increase of distance due to Lidar's scanning characteristics. In real scenarios, pedestrians may encounter occlusion, which leads to the reduction of returned point clouds. Figure 6 shows the projection image of pedestrians with different numbers of point clouds and their cosine similarities with the template. The pedestrians are extracted in different positions based on the ground truth. Some pedestrians that are occluded by obstacles are split into multiple small areas, like Figure 6b,d,e. As shown in Figure 6, with the number of point clouds increasing, the contour of pedestrians is gradually completed and similarity is also higher. When a pedestrian is occluded, as shown in Figure 6b,d,e, occlusion causes the contour to be missing and reduces similarity between pedestrian and template.

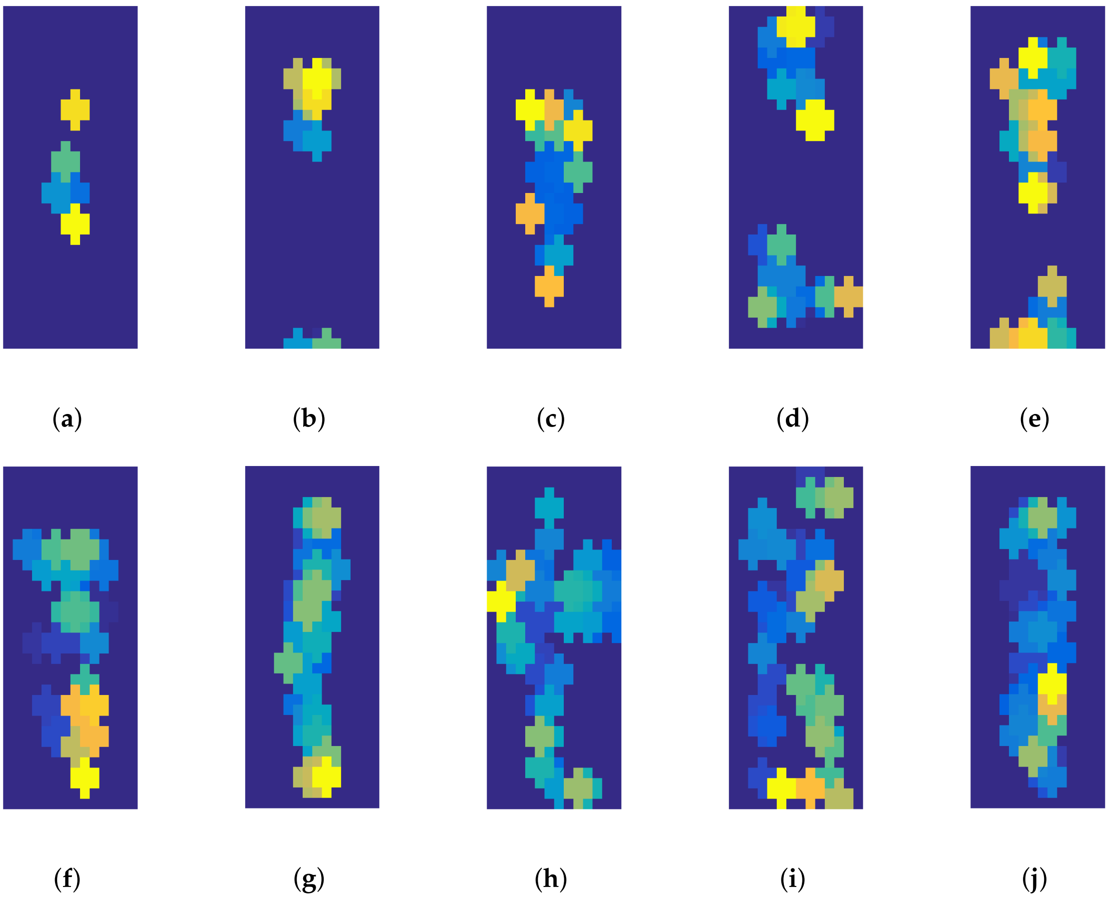


Figure 6. Projection image with different number of point clouds. Numbers in parentheses indicate cosine similarities. (**a**) Five points (0.36); (**b**) 10 points (0.38); (**c**) 15 points (0.46); (**d**) 22 points (0.49); (**e**) 26 points (0.53); (**f**) 30 points (0.54); (**g**) 35 points (0.58); (**h**) 40 points (0.59); (**i**) 43 points (0.61); (**j**) 49 points (0.62).

Figure 7 shows the statistical relationship between the number of point clouds and cosine similarity. Each red point in Figure 7a represents a test sample. Figure 7b shows a curve of the similarity variation trend with the number of point clouds. The x-coordinate of the curve is quantization value $\mathcal{X}$ of number of point clouds x using Equation (14), in which s is the quantization unit. For pedestrians that are quantized to the same value, similarity is calculated by their mean value. It is assumed that the samples are denoted by (X, Y), in which $X = x_1, ..., x_n$ represents the set of point clouds numbers, $Y = y_1, ..., y_n$ represents the set of cosine similarities between the samples and the template. Then X is quantified to $\mathcal{X}$ with Equation (14) and the cosine similarities of the obtained same quantified $\mathcal{X}$ are averaged to $\overline{Y}$. Figure 7b shows the relationship of $(\mathcal{X}, \overline{Y})$. As can be seen from Figure 7, similarity gradually increases with the increase of number of point clouds.

$$\mathcal{X} = \left\lceil \frac{x}{s} \right\rceil \times s \tag{14}$$

Figure 8 shows similarity variation with pedestrian distance. Figure 8a shows the similarity and pedestrian distance of each test sample. Figure 8b shows the curve of similarity and quantized distance, which is obtained by using the same quantified process as Figure 7b. Similarity gradually decreases with the increase of distance. In summary, it can be seen that a template-based detection method can obtain good matching results at close distance with dense point clouds.

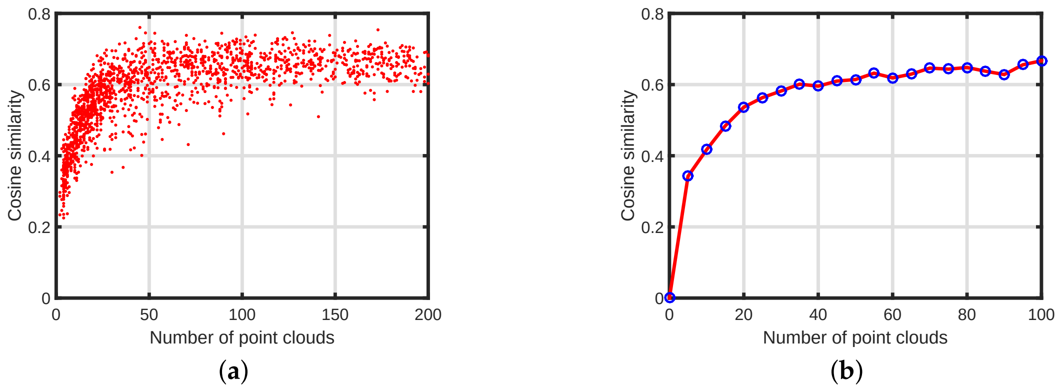


Figure 7. Relationship between number of point clouds and cosine similarity. (**a**) Relationship between number of point clouds and similarity for each sample. (**b**) Statistical curve of relationship between quantized number of point clouds and average similarity.

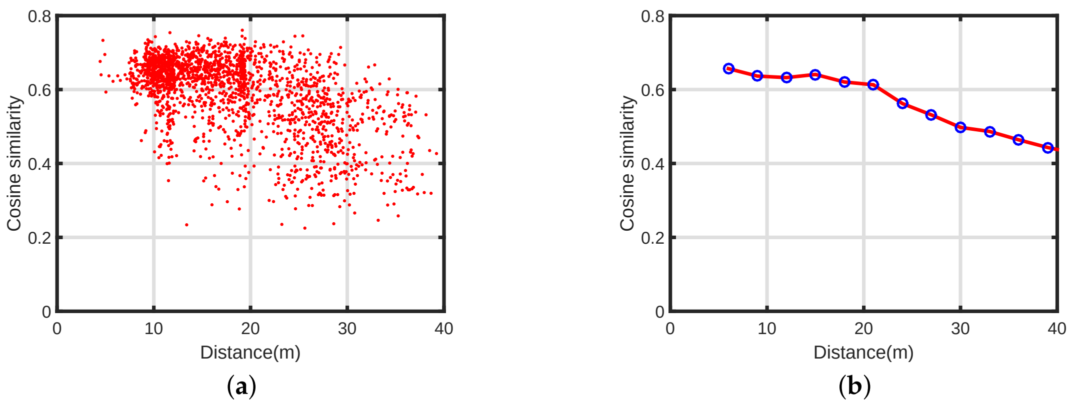


Figure 8. Relationship between pedestrian distance and cosine similarity. (**a**) Relationship between distance and similarity for each sample. (**b**) Statistical curve of relationship between quantized distance and average similarity.

3.3. Pedestrian Detection

The pedestrian-detection method in Reference [18] and single template matching with RBNN (STM-RBNN) were utilized to compare detection efficiency. A receiver operating characteristic (ROC) curve was adopted to evaluate pedestrian-detection performance with detection rate and false-alarm number. The calculation of the detection rate is in Equation (15).

$$R_D = \frac{N_P}{N_{PA}}, \tag{15}$$

where N_P denotes the number of detected pedestrians and N_{PA} represents the total number of pedestrians in the scenarios.

Figure 9 shows the ROC curve of pedestrian detection in the range of 50 m. The black solid line shows the performance of the STM-KDE algorithm. The blue dash-and-dotted line shows the performance of the STM-RBNN algorithm. The red dotted line is the results of Gidel's method based on KDE. It can be seen that, under the same false-alarm number, the detection rate of the two STM-based methods achieves better performance than the method based on KDE. The KDE-based pedestrian-detection method models upright pedestrians as cylinders and regards objects with high probability as candidate pedestrians. However, there are many objects that have a similar size to pedestrians in the scenario, such as street signs and mailboxes. Figure 10 shows a detection example of objects that may cause false alarms. Figure 10a is the optical image of the scenario. The object in the blue circle is a street sign that has similar height to a pedestrian. Detection results are shown in Figure 10b,c. The STM-based method can distinguish the object, while the KDE method detects it as a pedestrian. In addition, a tree trunk in the bush was wrongly detected as a pedestrian with the KDE-based method. Therefore, the KDE-based pedestrian-detection method is more suitable in simple scenarios with fewer columnar targets.

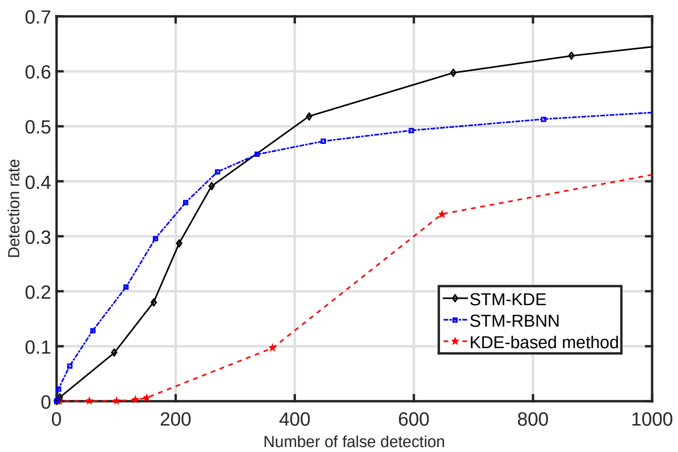
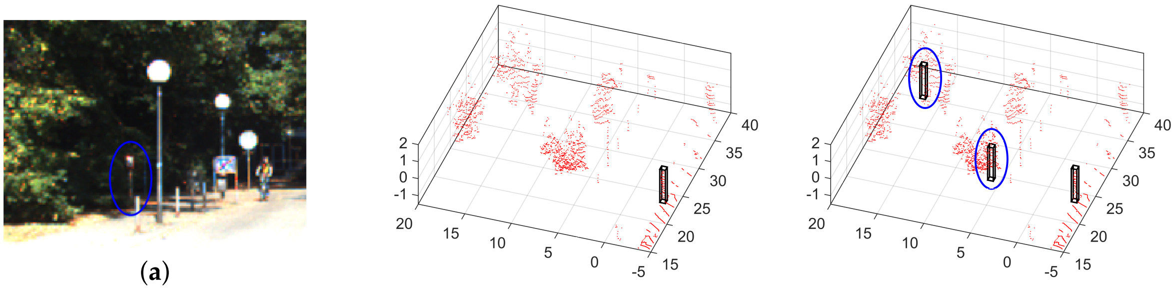


Figure 9. Receiver operating characteristic (ROC) curve in the range of 50 m.



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

Figure 10. Detection results of objects that may cause false alarms. (**a**) Scenario optical image. (**b**) Detection results with single template matching. (**c**) Detection result with the KDE-based method.

In Figure 9, it can also be concluded that the detection rate of the STM-RBNN algorithm is higher than that of STM-KDE in the case of low false alarms. The reason is that RBNN clustering results for most of the objects in the environment were good, though undersegmentation may occur when objects are close to each other. In KDE clustering, some point clouds of walls and street signs in special angles may be accumulated and extracted as candidate pedestrians, which produces false alarms. However, with threshold decrease, the STM-KDE algorithm achieves a higher detection rate. The reason that limits the detection rate with RBNN clustering is that there were many gathering pedestrians in the scenario, which made them undersegmented with RBNN clustering and filtered them out.

According to the ROC curve, appropriate thresholds were selected to show the statistics of pedestrian-detection results in different distances. Equations (16) and (17) were used to evaluate detection performance, in which the precision P indicates the ratio of the number of correct detections to the total number of detections and the recall R indicates the ratio of the number of correct detections to the total number of targets.

$$P = \frac{TP}{TP+FP}, \ R = \frac{TP}{TP+FN}, \tag{16}$$

$$F_1 = \frac{2PR}{P+R}, \tag{17}$$

where TP, number of true detections; FP, number of false alarms, FN, number of missing detections. F_1 score is the harmonic average of precision P and recall R and it is a comprehensive measure of detection results. Generally, the higher the F_1 score is, the better the detection performance of the algorithm is. The probability threshold in the KDE-based detection method was set to $\eta_{det}^{KDE} = 0.6$. The cosine similarity thresholds in STM with RBNN and KDE clustering were set to $\eta_{det}^{ST} = 0.6$. Detection results in the range of 15 m, 25 m and 50 m are listed in Tables 1–3, respectively.

Table 1. Pedestrian detection in the range of 15 m.

Algorithm	Datasets	Total Number	Truely Detected	False Alarms	Precision	Recall	F_1 Score
KDE-based method	0016	852	579	23	96.2%	68.0%	0.80
	0047	30	22	11	66.7%	73.3%	0.70
	Total	882	601	34	94.6%	68.1%	0.79
STM-RBNN	0016	852	427	77	84.7%	50.1%	0.63
	0047	30	29	10	74.4%	96.7%	0.84
	Total	882	456	87	84.0%	51.7%	0.64
STM-KDE	0016	852	629	34	94.9%	73.8%	0.83
	0047	30	28	14	66.7%	93.3%	0.78
	Total	882	657	48	93.2%	74.5%	0.83

Table 2. Pedestrian detection in the range of 25 m.

Algorithm	Datasets	Total Number	Truely Detected	False Alarms	Precision	Recall	F_1 Score
KDE-based method	0016	1591	890	27	97.1%	55.9%	0.71
	0047	100	90	33	73.2%	90.0%	0.81
	Total	1691	980	60	94.2%	58.0%	0.72
STM-RBNN	0016	1591	711	27	96.3%	44.7%	0.61
	0047	100	97	18	84.3%	97.0%	0.90
	Total	1691	808	45	94.7%	47.8%	0.64
STM-KDE	0016	1591	945	38	96.1%	59.4%	0.73
	0047	100	97	23	80.8%	97.0%	0.88
	Total	1691	1042	61	94.4%	61.6%	0.75

Table 3. Pedestrian detection in the range of 50 m.

Algorithm	Datasets	Total Number	Truely Detected	False Alarms	Precision	Recall	F_1 Score
KDE-based method	0016	2174	962	1106	46.5%	44.3%	0.45
	0047	118	97	59	62.2%	82.2%	0.71
	Total	2292	1059	1165	47.6%	46.2%	0.47
STM-RBNN	0016	2174	731	196	78.9%	33.6%	0.47
	0047	118	98	21	82.4%	83.1%	0.83
	Total	2292	829	217	79.3%	36.2%	0.50
STM-KDE	0016	2174	965	313	75.5%	44.3%	0.56
	0047	118	102	26	79.7%	86.4%	0.83
	Total	2292	1067	339	75.9%	46.6%	0.58

As can be seen in Tables 1–3, as the distance of the Lidar and the objects increases, the F_1 score of the three detection algorithms decreases to different degrees, which shows that pedestrian-detection efficiency has a decreasing trend. As distance increases, the number of point clouds returned by the pedestrian decreases and the contour of pedestrian gradually blurs, which causes a lot of miss alarms. At the same time, due to sparse point clouds in distant areas, the contour difference between pedestrians and other columnar objects may be reduced, resulting in many false alarms.

In addition, as can be seen from the results, in the range of 25–50 m, the detection performance of the KDE-based method obviously increases. The number of detections increases from 980 to 1059, while the detection number of STM-KDE only increased from 1042 to 1067. From Figures 7 and 8, it can be seen that cosine similarity decreases as the number of point clouds decreases, which makes detection more difficult. In the 50 m range, the number of false alarms of the two STM-based algorithms was 217 and 339, respectively, while the false-alarm number of the KDE-based number was increased to 1165. Therefore, compared with the KDE-based algorithm, the STM-based algorithm has a significant suppression effect on false alarms.

Since the moving speed of pedestrians is slow, and the safety of pedestrians who are closer to vehicles is more threatened by vehicles, pedestrian detection in close areas is more significant for environmental perception. Figure 11 shows the ROC curve in the range of 15 m. The KDE-based method performs well in the case of low false alarms. However, it cannot distinguish between upright pedestrians and columnar objects. The low detection threshold leads to a high detection rate but at the same time brings many false alarms, which limits the improvement of the detection rate. It can be seen from Figure 5 that RBNN clustering cannot accurately segment gathering pedestrians, so improvement of the detection rate is also limited. The STM-KDE algorithm could solve the problem of point-cloud segmentation and it has good detection performance for pedestrians in close distances.

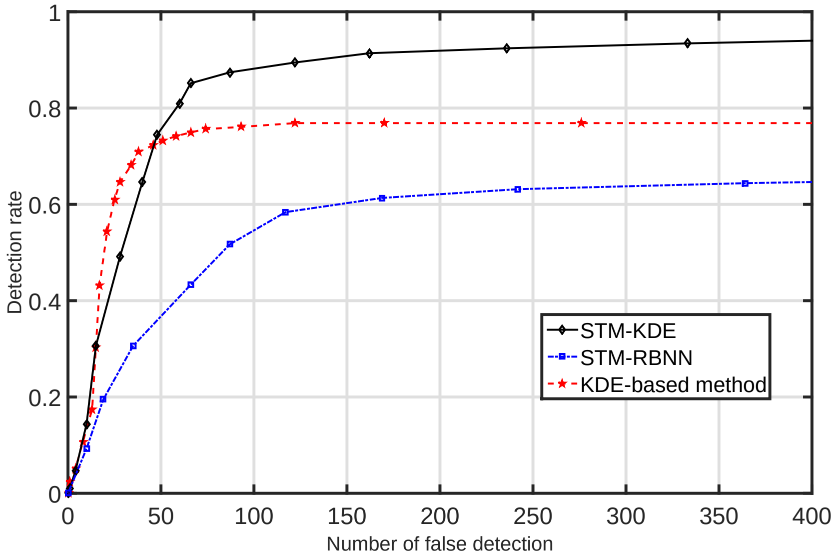


Figure 11. ROC curve in the range of 15 m.

3.4. Operation Time

All experiments in this work are implemented on a MacBook Pro with a 2.9 GHz Inter Core i5 CPU and 8 GB main memory. The MATLAB R2017b is employed to process the pedestrian detection with KITTI datasets. The operation time in 0016 dataset is shown in Table 4. Each process is calculated seprately. There are totally 31 frames of point clouds in 0016 dataset. The algorithm will detect a total of 583 candidate pedestrians in the scenarios, from which the real pedestrians are finally determined.

Table 4. Operation time of the pedestrian detection.

Total Time	Time/ Frame	Read Data	Preprocessing	Scan Layer Mark	Cluster	Template Match	Cluster Number	Detection/ Cluster
28.4 s	916 ms	248 ms	1791 ms	242 ms	11231 ms	11831 ms	583	48 ms

As shown in Table 4, in the consumed 28.4 s, there are 248 ms to read the point clouds and 242 ms to mark the layer index, which is caused due to processing the data in MATLAB platform. After removing the consuming time of data reading and scan layer index marking, it takes an average of 48 ms to detect one cluster. With the advanced hardware platform and parallel computing, the detection method can get real-time performance.

4. Conclusions

In this paper, a pedestrian-detection algorithm based on STM-KDE was proposed. The experiment results show that the proposed algorithm can cluster gathering pedestrians well and effectively distinguish pedestrians from other columnar objects. The main contributions of the paper are listed as follows.

First, considering the size of pedestrians and gathering characteristics, the candidate-pedestrian clustering method based on hierarchical segmentation and multilayer fusion can solve the segmentation problem of gathering pedestrians. The KDE-based method can distinguish between columnar objects and other structural objects in the scenario and reduce the searching region of pedestrian detection.

Second, the proposed single template-matching pedestrian-detection method can distinguish upright pedestrians from other columnar objects in the scenario. The contour feature matrix is established by projecting the point clouds on the main plane and constructing a projection image.

The proposed method applies the idea of template matching to pedestrian detection with point clouds, which avoids model training with a large number of data.

Third, the proposed detection method does not improve the computation complexity on the basis of improving the detection efficiency compared with existing detection method. With the advanced hardware platform and parallel computing, the proposed detection will perform well in operation time.

The proposed algorithm still has the problem of a decrease in detection rate when point clouds are sparse or occluded. In the future, point-cloud density enhancement and sensor fusion could be applied to obtain more object information for better detection.

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

Funding: This research was funded by the National Natural Science Foundation of China under grant No. 61771028.

Acknowledgments: The authors would like to thank the Jianqiang Wang of Tsinghua University for his constructive suggestions on our manuscript. The authors are also grateful to the reviewers and editor for their valuable comments and remarks to improve the quality of the contributions.

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

References

1. Ren, R.; Fu, H.; Wu, M. Large-Scale Outdoor SLAM Based on 2D Lidar. *Electronics* **2019**, *8*, 613. [CrossRef]
2. Dalal, N.; Triggs, B. Histograms of Oriented Gradients for Human Detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; Volume 1, pp. 886–893.
3. Ouyang, W.; Zeng, X.; Wang, X. Single-Pedestrian Detection Aided by Two-Pedestrian Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1875–1889. [CrossRef] [PubMed]
4. Hurney, P.; Waldron, P.; Morgan, F.; Jones, E.; Glavin, M. Review of Pedestrian Detection Techniques in Automotive Far-Infrared Video. *IET Intell. Transp. Syst.* **2015**, *9*, 824–832. [CrossRef]
5. Inan, T.; Halici, U. 3-D Face Recognition with Local Shape Descriptors. *IEEE Trans. Inf. Forensics Secur.* **2012**, *7*, 577–587. [CrossRef]
6. Wang, R.; Xu, Y.; Sotelo, M.A.; Ma, Y.; Sarkodie-Gyan, T.; Li, Z.; Li, W. A Robust Registration Method for Autonomous Driving Pose Estimation in Urban Dynamic Environment Using LiDAR. *Electronics* **2019**, *8*, 43. [CrossRef]
7. Goodin, C.; Carruth, D.; Doude, M.; Hudson, C. Predicting the Influence of Rain on LIDAR in ADAS. *Electronics* **2019**, *8*, 89. [CrossRef]
8. Zhao, J.; Xu, H.; Wu, J.; Zheng, Y.; Liu, H. Trajectory Tracking and Prediction of Pedestrian's Crossing Intention Using Roadside LiDAR. *IET Intell. Transp. Syst.* **2019**, *13*, 789–795. [CrossRef]
9. Arras, K.O.; Mozos, O.M.; Burgard, W. Using Boosted Features for the Detection of People in 2D Range Data. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 3402–3407.
10. Premebida, C.; Ludwig, O.; Nunes, U. Exploiting LIDAR-based Features on Pedestrian Detection in Urban Scenarios. In Proceedings of the 2009 12th International IEEE Conference on Intelligent Transportation Systems, St. Louis, MO, USA, 4–7 October 2009; pp. 1–6.
11. Kim, B.; Choi, B.; Yoo, M.; Kim, H.; Kim, E. Robust Object Segmentation Using A Multi-layer Laser Scanner. *Sensors* **2014**, *14*, 20400–20418. [CrossRef]
12. Kim, B.; Choi, B.; Park, S.; Kim, H.; Kim, E. Pedestrian/Vehicle Detection Using a 2.5-D Multi-Layer Laser Scanner. *IEEE Sens. J.* **2016**, *16*, 400–408. [CrossRef]
13. Tang, H.L.; Chien, S.C.; Cheng, W.H.; Chen, Y.Y.; Hua, K.L. Multi-cue Pedestrian Detection from 3D Point Cloud Data. In Proceedings of the 2017 IEEE International Conference on Multimedia and Expo (ICME), Hong Kong, China, 10–14 July 2017; pp. 1279–1284.
14. Li, K.; Wang, X.; Xu, Y.; Wang, J. Density Enhancement-Based Long-Range Pedestrian Detection Using 3-D Range Data. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 1368–1380. [CrossRef]

15. Stauffer, C.; Grimson, W.E.L. Learning Patterns of Activity Using Real-time Tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 747–757. [CrossRef]
16. Benedek, C.; Gálai, B.; Nagy, B.; Jankó, Z. Lidar-Based Gait Analysis and Activity Recognition in a 4D Surveillance System. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *28*, 101–113. [CrossRef]
17. Cui, J.; Zha, H.; Zhao, H.; Shibasaki, R. Laser-Based Detection and Tracking of Multiple People in Crowds. *Comput. Vis. Image Underst.* **2007**, *106*, 300–312. [CrossRef]
18. Gidel, S.; Checchin, P.; Blanc, C.; Chateau, T.; Trassoudaine, L. Pedestrian Detection and Tracking in an Urban Environment Using a Multilayer Laser Scanner. *IEEE Trans. Intell. Transp. Syst.* **2010**, *11*, 579–588. [CrossRef]
19. Liu, K.; Wang, W.; Tharmarasa, R.; Wang, J. Dynamic Vehicle Detection With Sparse Point Clouds Based on PE-CPD. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 1964–1977. [CrossRef]
20. Börcs, A.; Nagy, B.; Benedek, C. Instant Object Detection in LiDAR Point Clouds. *IEEE Geosci. Remote Sens. Lett.* **2017**, *14*, 992–996. [CrossRef]
21. Liu, K.; Wang, W.; Tharmarasa, R.; Wang, J.; Zuo, Y. Ground Surface Filtering of 3D Point Clouds Based on Hybrid Regression Technique. *IEEE Access* **2019**, *7*, 23270–23284. [CrossRef]
22. Giorgini, M.; Barbieri, F.; Aleotti, J. Ground Segmentation from Large-Scale Terrestrial Laser Scanner Data of Industrial Environments. *IEEE Robot. Autom. Lett.* **2017**, *2*, 1948–1955. [CrossRef]
23. Wei, P.; Cagle, L.; Reza, T.; Ball, J.; Gafford, J. LiDAR and Camera Detection Fusion in a Real-Time Industrial Multi-Sensor Collision Avoidance System. *Electronics* **2018**, *7*, 84. [CrossRef]
24. Jun, W.; Wu, T.; Zheng, Z. LIDAR and Vision Based Pedestrian Detection and Tracking System. In Proceedings of the 2015 IEEE International Conference on Progress in Informatics and Computing (PIC), Nanjing, China, 18–20 December 2015; pp. 118–122.
25. Klasing, K.; Wollherr, D.; Buss, M. A Clustering Method for Efficient Segmentation of 3D Laser Data. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; pp. 4043–4048.
26. Chavez-Garcia, R.O.; Aycard, O. Multiple sensor fusion and classification for moving object detection and tracking. *IEEE Trans. Intell. Transp. Syst.* **2015**, *17*, 525–534. [CrossRef]
27. Cheng, Y. Mean Shift, Mode Seeking, and Clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* **1995**, *17*, 790–799. [CrossRef]
28. Navarro-Serment, L.E.; Mertz, C.; Hebert, M. Pedestrian Detection and Tracking Using Three-dimensional Ladar Data. *Int. J. Robot. Res.* **2010**, *29*, 1516–1528. [CrossRef]
29. Seo, H.J.; Milanfar, P. Training-Free, Generic Object Detection Using Locally Adaptive Regression Kernels. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 1688–1704. [PubMed]
30. Wang, Y.; Wang, W. Face Detection Using Skin Color and Locally Adaptive Regression Kernels. In Proceedings of the 2017 20th International Conference on Information Fusion (Fusion), Xi'an, China, 10–13 July 2017; pp. 1–8.
31. Takeda, H.; Farsiu, S.; Milanfar, P. Kernel Regression for Image Processing and Reconstruction. *IEEE Trans. Image Process.* **2007**, *16*, 349–366. [CrossRef] [PubMed]
32. Wang, D.; Lu, H.; Bo, C. Visual Tracking via Weighted Local Cosine Similarity. *IEEE Trans. Cybern.* **2015**, *45*, 1838–1850. [CrossRef] [PubMed]
33. Pirlo, G.; Impedovo, D. Cosine Similarity for Analysis and Verification of Static Signatures. *IET Biom.* **2013**, *2*, 151–158. [CrossRef]
34. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision Meets Robotics: The KITTI Dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [CrossRef]