electronics

MDPI

Article

A Hierarchical Clustering Obstacle Detection Method Applied to RGB-D Cameras

Chunyang Liu [1,2], Saibao Xie [1,*], Xiqiang Ma [1,2], Yan Huang [1], Xin Sui [1,3], Nan Guo [1], Fang Yang [1,2] and Xiaokang Yang [1]

[1] School of Mechatronics Engineering, Henan University of Science and Technology, Luoyang 471003, China; chunyangliu@126.com (C.L.)
[2] Longmen Laboratory, Luoyang 471000, China
[3] Key Laboratory of Mechanical Design and Transmission System of Henan Province, Luoyang 471000, China
* Correspondence: xiesaibao@163.com

Abstract: Environment perception is a key part of robot self-controlled motion. When using vision to accomplish obstacle detection tasks, it is difficult for deep learning methods to detect all obstacles due to complex environment and vision limitations, and it is difficult for traditional methods to meet real-time requirements when applied to embedded platforms. In this paper, a fast obstacle-detection process applied to RGB-D cameras is proposed. The process has three main steps, feature point extraction, noise removal, and obstacle clustering. Using Canny and Shi–Tomasi algorithms to complete the pre-processing and feature point extraction, filtering noise based on geometry, grouping obstacles with different depths based on the basic principle that the feature points on the same object contour must be continuous or within the same depth in the view of RGB-D camera, and then doing further segmentation from the horizontal direction to complete the obstacle clustering work. The method omits the iterative computation process required by traditional methods and greatly reduces the memory and time overhead. After experimental verification, the proposed method has a comprehensive recognition accuracy of 82.41%, which is 4.13% and 19.34% higher than that of RSC and traditional methods, respectively, and recognition accuracy of 91.72% under normal illumination, with a recognition speed of more than 20 FPS on the embedded platform; at the same time, all detections can be achieved within 1 m under normal illumination, and the detection error is no more than 2 cm within 3 m.

Keywords: vision; detection process; depth continuity judgment; clustering method

Citation: Liu, C.; Xie, S.; Ma, X.; Huang, Y.; Sui, X.; Guo, N.; Yang, F.; Yang, X. A Hierarchical Clustering Obstacle Detection Method Applied to RGB-D Cameras. *Electronics* **2023**, *12*, 2316. https://doi.org/10.3390/electronics12102316

Academic Editor: Silvia Liberata Ullo

Received: 28 April 2023
Revised: 18 May 2023
Accepted: 19 May 2023
Published: 21 May 2023

1. Introduction

The selection of different sensors can significantly impact a robot navigation system's operational functionality and efficiency. Typically, the selection of sensors depends on the task requirements and the working environment. In office areas, for example, irregularly stacked obstacles and dynamic environments place higher application requirements on sensors. In this type of scenario, the versatility of the vision system has a more special status than other sensors. Vision systems are feasible in all aspects of specific target detec-tion, such as target tracking, scene analysis or even robot path generation [1]. Thus, vision systems are preferable in providing critical functions for robot navigation, such as obstacle detection [2], area detection [3], road detection [4], map updating, body localization, and state estimation, and crop row guidance [5].

Our research aims to enable a robot system to implement unconstrained movements in any random space. Currently, a robot already has an a priori map indicating immovable obstacles (e.g., large tables, walls) generated by LIDAR to assist a path planning module to efficiently generate feasible paths, where LIDAR and inertial navigation continuously conduct localization and state estimation even for an identical scene. For scenarios containing several immovable obstacles, the scale of data generated by LI-DAR could oversize the processing

capacity of a dynamic obstacle detection system. Therefore, several candidate sensor types are selected to aid obstacle detection, where vision systems outweigh other types of sensors by providing rich 3D information [6] and efficient data processing capabilities.

In practice, both monocular and binocular cameras are feasible for vision systems. Monocular cameras generally produce significant errors in calculating targets' spatial position, which requires an additional map zoom-in operation after projecting an identified obstacle onto the map [1,7]. This additional operation may produce a more extensive coverage of the identified object than the actual one and can impact the robot's pathfinding or even produce an infeasible path. Conversely, RGB-D binocular cameras are preferable for obstacle detection since they can provide more accurate spatial information.

This paper conducts research on obstacle detection using RGB-D cameras, and the main contributions are:

1. an obstacle detection process applied to an embedded platform is proposed;
2. a new hierarchical clustering method that improves on the traditional hierarchical clustering method.

The main steps of the process are data alignment, feature point extraction, ground point elimination, feature point clustering and 2D mapping of obstacles. For the feature extraction and ground point elimination steps, we use existing methods and make adaptive changes to ensure the reliability of the feature points. Finally, we test and compare the recognition results of the whole process using an artificial data set to verify its practicality.

The paper's organization is as follows: Section 2 comprehensively reviews related work. Section 3 shows the algorithmic process for obstacle detection, which comprises data acquisition, image processing, feature extraction, noise reduction, and features clustering. Section 4 uses an artificial dataset for initial testing and a real dataset to compare the performance of different methods, demonstrating an obstacle generation approach for SLAM maps. Sections 5 and 6 conclude the highlights and summaries the drawbacks of the proposed method.

2. Related Work

Obstacle detection algorithms can be divided into two main categories: deep learning methods and traditional methods. Deep learning methods often suffer from problems such as the need for large data sets to train the network itself and the inability to recognize obstacles when they are untrained or partially present in the field of view. Traditional methods, despite being less effective in classifying objects, possess higher adaptability than the former in scenarios that do not require analysis of specific categories. Clustering, as a classical data processing method that can distinguish data without explicit labels, has been successfully applied in a variety of fields.

There are various clustering methods, which can be classified into division methods, density-based methods, spectral clustering methods, affinity propagation algorithms, etc. according to the principles of the methods. However, these algorithms often have disadvantages, such as external determination of the number of clusters, fixed granularity, and unclear organization of the resulting clusters. In contrast, hierarchical clustering has the advantages of generating clustering trees that reveal inter-cluster relationships at different resolutions, such as clustering according to inter-cluster conditions, without the need for external determination of the number of clusters, etc.

2.1. Hierarchical Clustering

Hierarchical clustering can be divided into merging and splitting algorithms based on the difference between bottom-up and top-down. The former consists of two main steps: (1) calculating the distance between two clusters based on linkage criteria, and (2) iteratively merging the clusters that meet the requirements. Since the proposed method belongs to coalescent clustering, the related introduction of a splitting algorithm is omitted here.

The classical hierarchical clustering methods are cluster-averaged clustering methods [8], where the inter-cluster distance is defined as the average distance of all data point

pairs. The Sneath–Sokal method [9] considers that the inter-cluster distance is defined as the shortest distance from any member of one inter-cluster to any member of another cluster. The farthest distance method [10] considers that the inter-cluster distance should be the longest distance from any member of one inter-cluster to any member of another cluster.

The above algorithms are prone to produce slender clusters and usually consume much time [11]. Many other hierarchical algorithms have appeared to solve these problems. BIRCH [12] and PERCH [13] based on binary trees have very fast clustering speeds but are less robust and unsuitable for data with outliers. Sampling-based methods such as ROCK [14] and CURE [15] possess higher stability by computing the distance between clusters with a fixed number of cluster representations. The stepwise algorithm CHAMELEON [16] uses the nearest neighbor graph to divide the original data into small clusters to reduce the number of iterations and improve the clustering efficiency. The disadvantage of sampling-based and stepwise algorithms is that they are very sensitive to parameters. Wen-Bo et al. proposed the RSC method [17] based on the assumption that two reciprocal nearest data points should be grouped in one cluster and achieved faster and more accurate results than other methods.

2.2. Optimization Method of Clustering

When the amount of data or the number of dimensions is too large, clustering itself suffers from problems such as too long clustering time and difficulty in finding similarity. To solve this problem, the usual method is to perform data dimensionality reduction. Data dimensionality reduction can be divided into two kinds. One is by considering both clustering and feature learning in the clustering framework kind. For example, the depth density-based image clustering algorithm DDC [18] adds deep network feature learning to the clustering framework kind to extract object features for improving the clustering accuracy. The algorithm DCC [19], based on deep continuous clustering, introduces nonlinear dimensionality reduction in the clustering framework and then performs continuum optimization at the global level to improve the clustering accuracy. Such methods have good clustering performance and clustering speed after completion, the drawback is that the shallow learning network has limited capability, while the deep network itself requires a large amount of data and training process.

Another way is to introduce manual empirical extraction of object feature points or feature selection to reach the purpose of data dimensionality reduction. For example, in the literature [20], point cloud segmentation is used to chunk the point cloud and then cluster it, in the literature [21], multi-view feature descriptors are encoded as binary codes for computational optimization before clustering the data points, and in the literature [22], clustering is performed from a single channel, merging clusters with small boundary point spacing, and finally performing inter-channel clustering. Chu, Z. et al. processed high-dimensional data by extracting features mixed with clustering [23]. Such methods usually provide less improvement in terms of clustering performance but can greatly improve the speed of clustering detection while reducing upfront costs by eliminating the need for training.

In the process proposed in this paper, the data dimensionality is reduced by the traditional corner point extraction method to speed up the clustering.

2.3. Ground Point Removal

Since this paper uses the traditional corner point detection method to extract feature points, the influence of ground texture and illumination will lead to the appearance of non-obstruction feature points. After the feature points are endowed with depth information, their properties are similar to those of point clouds, so the ground point elimination methods can be borrowed from point clouds.

The common ground point elimination methods can be divided into geometric and neural network methods. Neural network methods are generally trained to segment images, and in-point cloud-type data usually convert dense point clouds into multi-channel

images before the segmentation process. Neural-network-based road detection methods are presented in [24,25]. Geometric methods generally reject ground points based on geometric relationships between themselves or obstacles, such as in [26], Chu et al. distinguish ground points from obstacle points by calculating the change in slope between consecutive points, and common geometric-like methods are also available in [27,28].

3. Method

In this section, we propose a process for obstacle detection in mobile robots based on clustering by RGB-D cameras. This method can be divided into four main stages. First, mapping the depth data to color images to form 4D data; second, extracting color image feature points using image processing and feature point extraction methods; third, re-moving interfering features; and fourth, using the depth data of the feature points to distinguish between different targets based on depth continuity. The fifth part is not the principal part of the algorithm, but the mapping of the map after the obstacle detection is completed. The overall scheme for implementing the proposed obstacle detection method is shown in Figure 1 and each step of the system is described in detail in the following sub-sections.

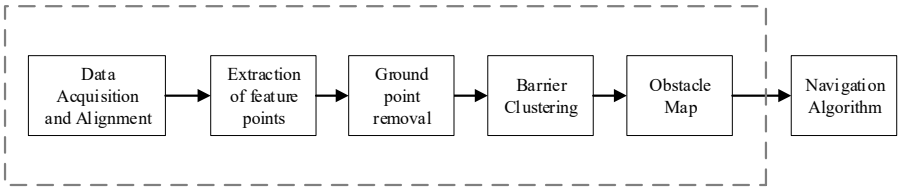


Figure 1. Obstacle detection system.

3.1. Data Acquisition and Alignment

The first step in the process is to get depth data for each pixel point on a color image. We achieve this by projecting points from the depth pixel coordinate system onto the color pixel coordinate system.

To align an RGB image to a depth image, the depth data of each pixel point in the depth image is assigned to the RGB image; that is, the data of the depth pixel coordinate system is mapped to the color pixel coordinate system. First, point $P^d_{u,v}$ in the depth pixel, the coordinate system is reduced to the depth coordinate system to obtain point $\dot{P}_{dc}$:

$$\dot{P}_{dc} = Z\boldsymbol{K_d^{-1}}P^d_{u,v} \tag{1}$$

where Z is the Z-axis information of the pixel point, and $\boldsymbol{K_d}$ is the internal reference matrix of the depth camera. Next, point $\dot{P}_{dc}$ is converted to P_ω by using the conversion matrix $\boldsymbol{T^{-1}_{\omega 2d}}$ between the depth coordinate system and the world coordinate system:

$$P_\omega = \boldsymbol{T^{-1}_{\omega 2d}}\dot{P}_{dc} \tag{2}$$

Then, point P_ω is converted to the color camera coordinate system:

$$\dot{P}_{cc} = \boldsymbol{T_{\omega 2c}}P_\omega \tag{3}$$

Finally, point $\dot{P}_{cc}$ is mapped to the colored plane with Z = 1:

$$P^c_{u,v} = \boldsymbol{K_c}\left(\dot{P}_{cc}/Z\right) \tag{4}$$

After processing all the depth pixel points, a PRB-D picture with depth channels is obtained.

3.2. Image Preprocessing and Feature Point Extraction

In the second step, we obtain the contour of the object by extracting the feature points on the contour of the object.

The corner points of an object are commonly used feature points, usually determined by the gradient change of pixel points on the image. The Shi–Tomasi algorithm is a strong corner point extraction method improved from Harris, which is faster to compute. However, when only the Shi–Tomasi algorithm is used, the intense color changes on the surface of some objects can cause the algorithm to ignore the gradient changes of the object edge contour points, resulting in sparse extraction of object edge contour points, as shown in Figure 2a.

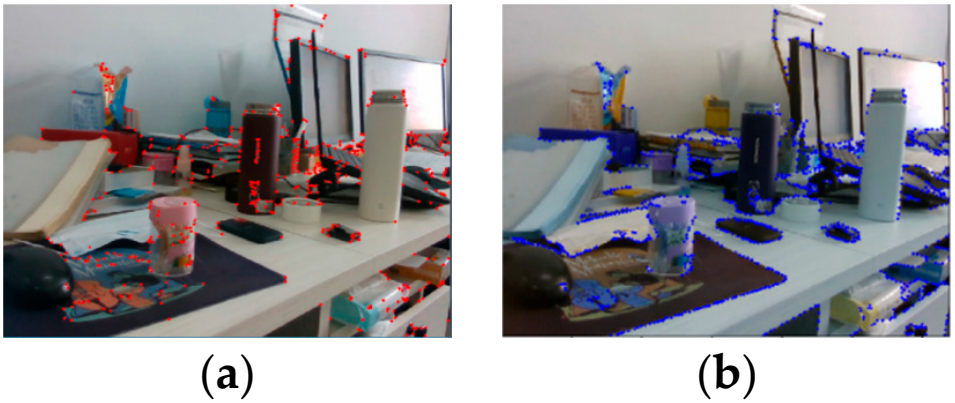


Figure 2. Comparison of extraction effects: (**a**) Shi–Tomasi algorithm; (**b**) Effect after the addition of the Canny operator.

The Canny operator is a commonly used object edge extraction algorithm to extract the edges of an object, ignoring the texture information on the object's surface. After extracting the object edges using the edge extraction algorithm, the Shi–Tomasi algorithm is then used to extract the feature points of the object edges, which can obtain contour points with high accuracy, as shown in Figure 2b.

3.3. Noise Removal

We have two conditions to judge whether the feature points are valid, one is that the feature points have the correct depth information, and the other is that the feature points are present on the obstacle. Then, invalid feature points come from two sources: feature points with abnormal depth data and noise on color images.

Feature points with depth anomalies usually occur in the hollow regions of a depth image, resulting in feature points with too-large or too-small depth attributes. Usually, the solution to this problem is to use edge information from the color image to repair the depth edges. However, we found that the absence of some feature points in our method does not affect the overall contour determination. Therefore, we increase the upper and lower thresholds to remove feature points whose depth information is out of range, depending on the effective distance of the camera and the required detection distance.

Noise in color images is usually found in ground textures, such as floor patterns or object shadows. This is because the points floating in the air are usually associated with objects after excluding points that are too far in depth from the application scenario. To remove the floor noise, we improve the method based on the one in [26] to be more applicable to the method in this paper. The image distortion is first corrected using the camera's internal reference, at which point the camera's optical center, pixel plane, and object are positioned as shown in Figure 3. The blue area at the bottom of the figure represents the horizontal ground.

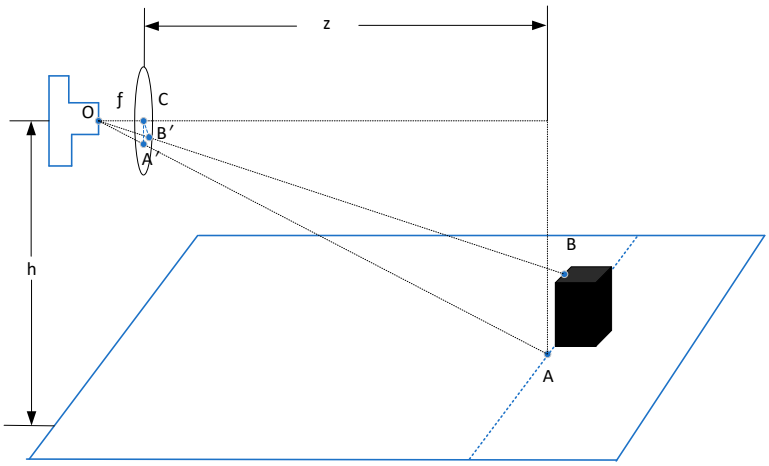


Figure 3. Using depth to determine ground noise.

In a condition where the ground is not significantly undulating, the ground point A is represented in the pixel coordinate system as A', and the distance between the camera optical center and the ground is determined by the height of the body as h. The distance between the pixel coordinate system and the object is identified by the depth camera as z. The vertical height of $A'C$ in the pixel plane can be calculated from the triangle similarity:

$$d_h = (f * h)/z \tag{5}$$

The depth z of different feature points is different, and the maximum height d_h is calculated according to the depth plane where the feature point is located. When the height of the image plane is less than the maximum height d_h, the point is considered a valid feature point, and the opposite is considered a noise point to be removed.

On the same plane, the vertical height $d_{B'C}$ of the corresponding point B' of object B on the pixel plane concerning the optical center is clearly less than d_h.Therefore, considering the obstacle height of 5 cm that can be spanned by the wheel diameter, the detection limit is set to $h_0 = (h - 0.05)$, and the vertical height of the corresponding pixel point to the optical center is $d_{h_0} = (f * h_0)/z$. For each feature point N, the corresponding d_n and d_{h_0} are calculated after obtaining its depth value z. If $d_n > d_{h_0}$, it is filtered out.

3.4. Barrier Clustering

After completing the above preparatory work, we get the set of feature points with deep data, and next, we need to find these objects by a new clustering method. Our approach is based on the simple fact that feature points on the same object contour must be continuous or within the same depth for obstacles within the RGB-D camera view. When we filter all the feature points according to this principle, we can get the alternative ones and filter out the noise simultaneously. Additionally, we can complete the clustering process by dividing the points with a large interval between them in the horizontal direction of the camera.

Whether the feature points are continuous or in the same plane is determined by the following equation:

$$|\mathrm{P_n} - \mathrm{P_{n+1}}| \leq \mathrm{d} \tag{6}$$

where $\mathrm{P_n}$ and $\mathrm{P_{n+1}}$ are two points adjacent to each other after sorting by individual axial data and d is the threshold distance, different distance thresholds are selected on different axes. For example, when distinguishing the transverse distance of an object, the threshold is determined by the body width. If the space between the two obstacles cannot pass through the robot, the obstacles can be classified as one class.

The specific process is shown in Figure 4. We can divide it into five steps.

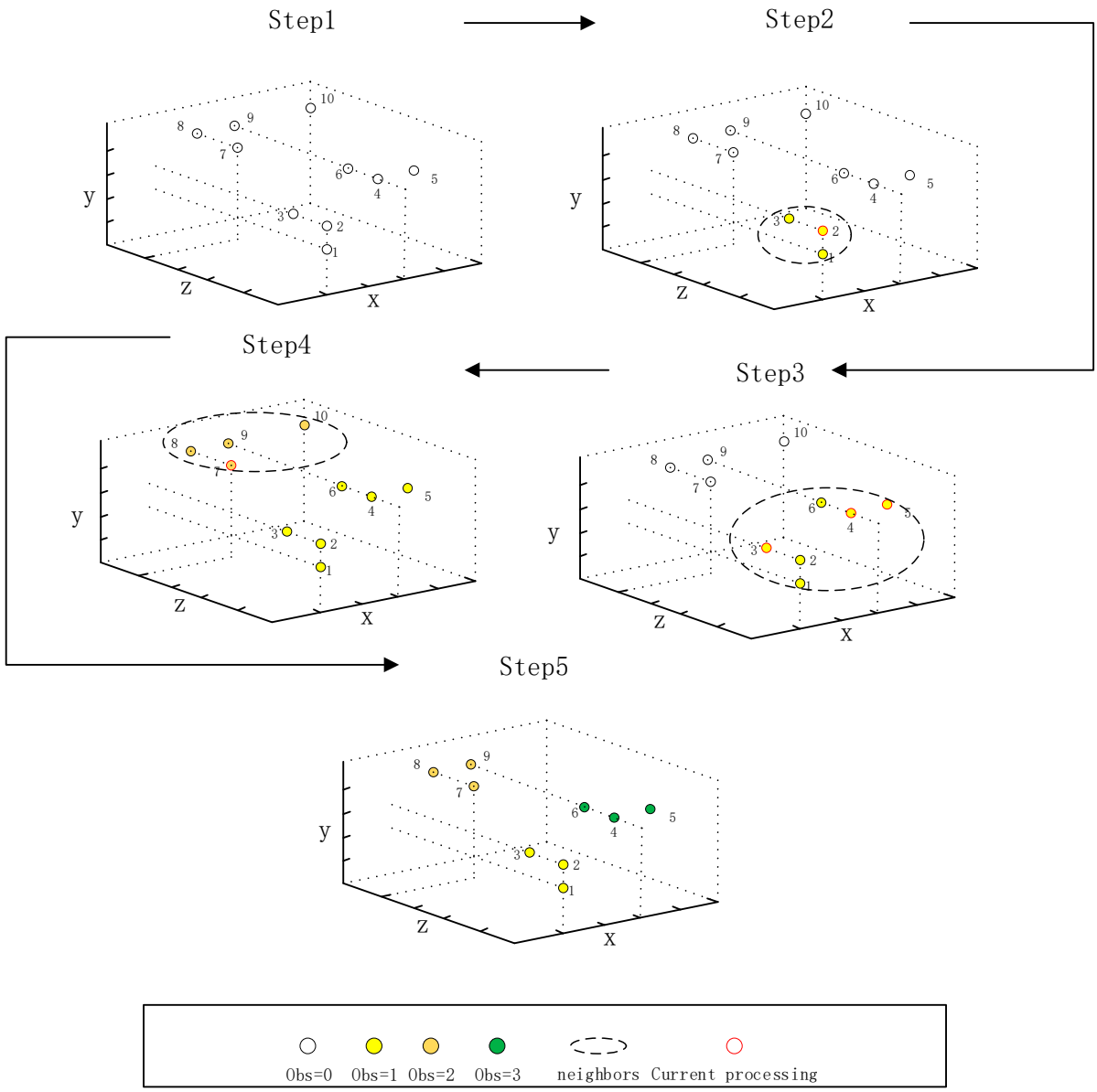


Figure 4. An example of the new clustering method for clustering feature points.

Step 1: The input feature points are sorted by *z*-axis from smallest to largest;

Step 2: The point with the smallest depth is set as the initial set of points, and the points smaller than the depth distance threshold are grouped in the same set;

Step 3: When the depth distance between other feature points and any point in the set is less than a threshold, these points are also put into the same set;

Step 4: After there are no feature points with similar depth distance, the points with the smallest depth value among the remaining feature points are reselected to form a new set. Until all feature points are divided;

Step 5: Repeat the above process for the feature points in the same set in the *x*-axis direction. Finally, remove the set containing fewer feature points.

The specific algorithm process Algorithm 1 is shown:

Algorithm 1. Depth continuous judgment

```
Input: Img,N,d,D
Output: obstacle
obstacle ←[], point_set ←[], v ←0
1. while len(Img):
2.       point_set ← Img(0)
3.             for i in range (1,len(Img))
4.                for j in point_set:
5.                   if D(Img(i))-D(j) < n:
6.                      point_setadd D(Img(i))
7.                   else:
8.                      break
9.                   end if
10.               end for
11.           end for
12.       if len(point_set)>N:
13.           Img = Img delete point_set
14.           obstacle[v] ← point_set
15.           v ←v + 1
16.       else:
17.           Img = Img delete point_set
18.       end if
19. end while
20. return obstacle
```

The Img is RGBD data after sorting by depth, the N is the threshold for determining the number of obstacles, and the d is the distance threshold to determine whether the depth direction is continuous or not. Additionally, the d() serves to obtain the depth value of the points inside the brackets.

3.5. Obstacle Mapping

Once the work of target recognition is complete what needs to be considered is how to translate the information about the obstacles onto the map to help the robot move. This requires the interpretation of the object information contained in the feature points. As the map is two-dimensional, we project the feature points onto the ground. Based on the experimental process, there are three main object models, as shown in Figure 5.

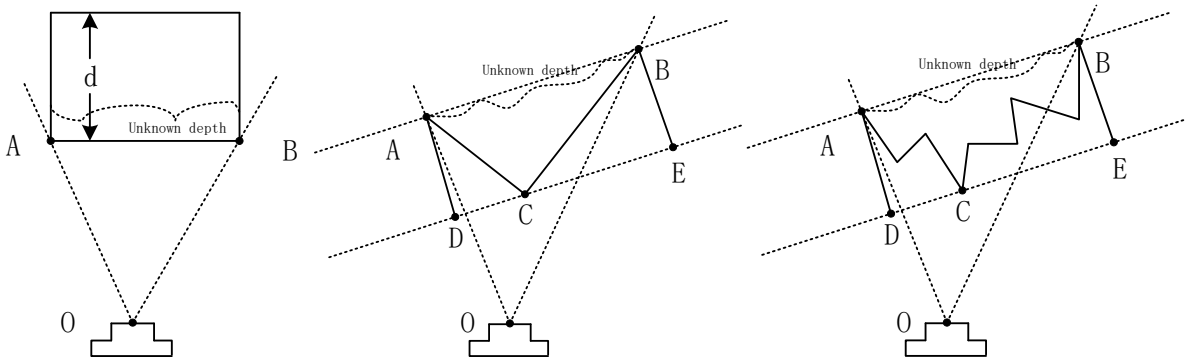


Figure 5. Model and obstacle estimation from the camera's perspective.

Due to the errors in the depth camera itself, it is difficult for the multi-vertex model to count its vertex information, so we take a simple approach to build the obstacle map. Firstly, the feature points are arranged by lateral coordinates, where the two points with the smallest and largest lateral coordinates are used as the two points *A* and *B* in the map;

the point *C* with the smallest depth value is obtained and the depth difference between its depth value and the smaller of the two points *AB* is calculated, if there is:

$$|d_c| - \min(d_a, d_b) < d_{\text{threshold}} \tag{7}$$

The value of $d_{\text{threshold}}$ is usually determined by the sensor error.

Then, the measured barrier has no vertices. If greater than d, the minimum point is the vertex. In the vertex-less model, the line connecting *AB* is used as an edge and extended backward a fixed distance to form a rectangle. In other models, the *AB* line is used as an edge and a parallel line of *AB* is made through point *C* to intersect the perpendicular line through *AB* at two points *DE* to form a rectangle, as shown in Figure 5, ensuring absolute safety for the robot despite some puffing up of the obstacle.

In practical applications, the individual models are transformed into each other. For example, if a robot detects a bin that does not exist in the map while moving, it is considered as a vertex-free model when facing forward, and we assign a width d to it as an imaginary width. As the robot moves, the viewpoint changes and the bin become a single-vertex model, and the mapping of the bin in the map is calculated according to the three vertices of the bin. Another example is that a person is considered a vertex-free model when facing the camera head-on, becomes a single-vertex model when facing the camera side-on, and is mapped to a large rectangle as a multi-vertex model when facing a queue formed by multiple people.

After completing the 2D contour construction, the object in the camera coordinate system must be represented in the world coordinate system. As shown in Figure 6, For any point *P* on the object, the position in the camera coordinate system is *(x,y)*, the camera's position in the world coordinate system is (X_0, Y_0), and the angle between the coordinate systems is θ. The corresponding position in the world coordinate system (X,Y) is:

$$X = y\sin(\theta) + x\cos(\theta) + x_0 \tag{8}$$

$$Y = y\cos(\theta) - x\sin(\theta) + y_0 \tag{9}$$

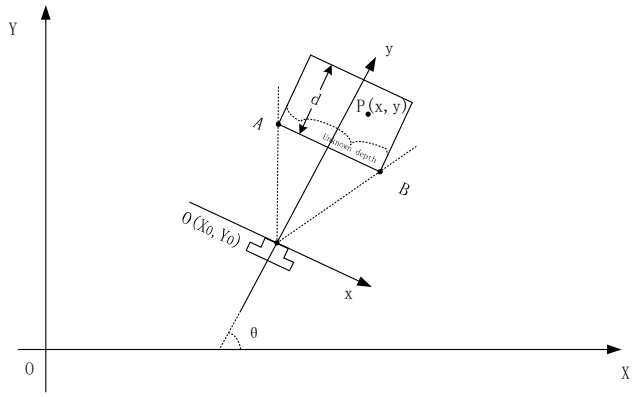


Figure 6. Obstacle space coordinates transformation.

The construction of the obstacle map is completed by adding the transformed contour points to the SLAM map one by one.

4. Experiment and Analysis

To demonstrate the feasibility of our proposed method, we will experimentally validate our method in this section in terms of clustering performance and clustering efficiency, target recognition performance, and target recognition accuracy. Section 4.1 shows the identification of the dataset and discusses the reasons why the open dataset is not suitable for evaluating the performance of the method. Section 4.2 compares clustering efficiency and performance between our approach and the advanced method RSC hierarchical clustering

algorithm. Section 4.3 compares our method with the RSC method under different scene datasets to investigate the difference in their recognition performance. Section 4.4 discusses the accuracy of the robot's recognition of objects. Section 4.5 briefly shows a scene where an object is recognized and projected onto a SLAM map.

The mobile robot used in this study is shown in Figure 7. An Intel RealSense D435i RGB-D camera was mounted in the center of the robot body to acquire color and depth images. NVIDIA TX2 was used as the main control computer for processing the data obtained by sensors, and 16-line LIDAR was used for map construction and navigation.

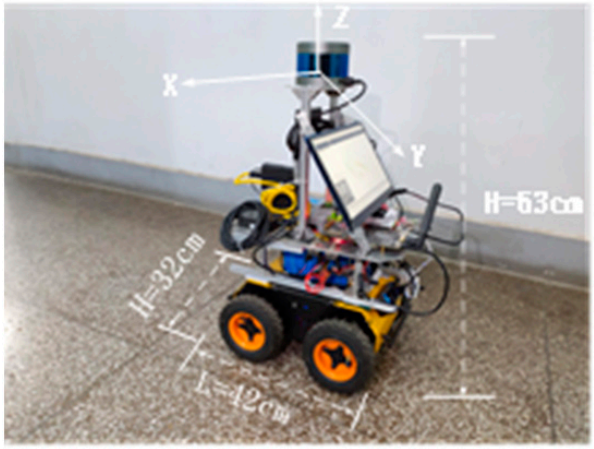


Figure 7. Mobile robot for testing.

4.1. Dataset Description

We tested the proposed method on two datasets: a public dataset (SUNRGBD multi-modal artificial dataset) and a real-world dataset containing live images and depth maps recorded by cameras mounted on a mobile robot.

4.1.1. SUNRGBD Dataset

We performed tests using the SUNRGBD dataset, which contains images of various indoor complex environments captured using cameras with different depths. This dataset includes various patterns; however, in this study, we used only RGB and depth images. Figure 8 shows the recognition in different scenes of the dataset. The whole obstacle could be recognized in different scenes. However, this data set has two shortcomings in the actual test. One is that the photographer does not obtain data from the wheeled robot's perspective; the other is that in terms of content, there are too many and too miscellaneous objects in the field of vision, which is inconvenient for data statistics and comparison. Thus, a dataset containing only complex obstacles does not demonstrate the performance of the proposed method.

Figure 8. Recognition effect on the SUNRGBD dataset.

4.1.2. Real-World Dataset

Real-world applications need a dataset of real-world application scenarios so that the strengths and weaknesses of the approach can be better evaluated. Therefore, we

constructed the second dataset by remotely controlling the wheeled robot to move and shoot to obtain different images. It includes streetlights, people, cars, buckets, tree trunks, tables, chairs, etc. There are 50 different kinds of common obstacles, containing more than 500 images. Figure 9 shows some examples.

Figure 9. Scenarios in the real-world dataset.

4.2. Clustering Efficiency Comparison

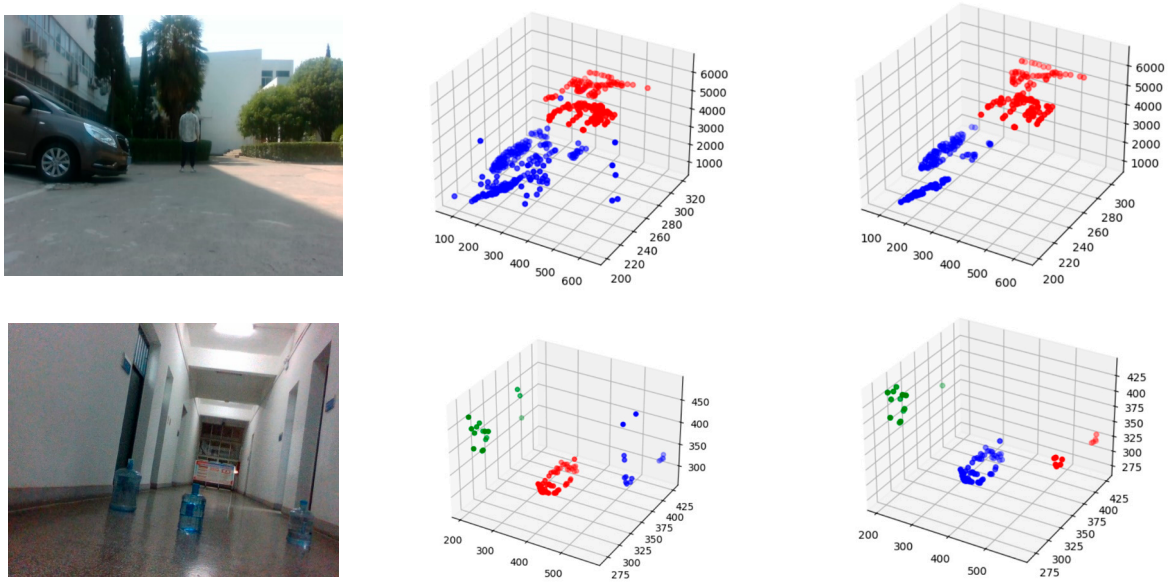
The ideal result of the clustering algorithm is that points within the same category have a high degree of similarity, while points between different categories have no similarity. Silhouette Coefficient (SC) and Davies–Bouldin Index (DB) are common internal indicators to evaluate the clustering performance, where SC is calculated between $[-1, 1]$, the larger the value means the higher the intra-class similarity and the larger the inter-class distance, and DB is calculated between $[0, 1]$, the smaller the value the better the clustering performance better.

In order to verify the clustering performance of our method, we choose the traditional hierarchical clustering method as well as the advanced RSC method to compare the clustering performance index results of the three methods under the same scenario and preprocessing method, and the calculation results are shown in Table 1, where the traditional hierarchical clustering method is from the sklearn library in Python.

Table 1. Clustering performance index.

	People and Vehicle		**Roadblocks**		**Bucket**		**Tree Trunk**	
	SC	**DB**	**SC**	**DB**	**SC**	**DB**	**SC**	**DB**
Traditional Method	0.6747	0.5467	0.8523	0.4037	0.7492	0.4127	0.5339	0.6531
RSC Method	0.7897	0.428	0.9005	0.1214	0.9423	**0.079**	**0.8791**	**0.3453**
Ours	**0.8782**	**0.3256**	**0.9209**	**0.0957**	**0.9477**	0.0833	0.8781	0.3459

In each column, the best results are highlighted in bold. From the table, we can see that (1) the indicators of the clustering results of the same method in different scenes are different. Additionally, the three methods show the same change trend, which may be the influence of the scene characteristics themselves. (2) The results of clustering by the method in this paper are generally similar to those of the RSC method and better than the traditional hierarchical clustering method, which indicates that in most cases, the method in this paper is competitive with the advanced methods. (3) Regarding metrics, our method outperforms RSC's method in scenario 1. Combined with the clustering images in Figure 10 that may illustrate the results, our method clusters fewer outlier points in the human-vehicle scenario, which indicates that our method has stronger noise resistance.



(**a**) Original scene (**b**) RSC method Clustering results (**c**) Our method Clustering results

Figure 10. Clustering results of different scenarios.

The clustering speed is also an important indicator of how good the clustering method is. The clustering speed of the three methods is shown in Figure 11, from which the method in this paper is at least two times faster than the RSC method and 4–5 times faster than the traditional method, which can better meet the working requirements of wheeled robots.

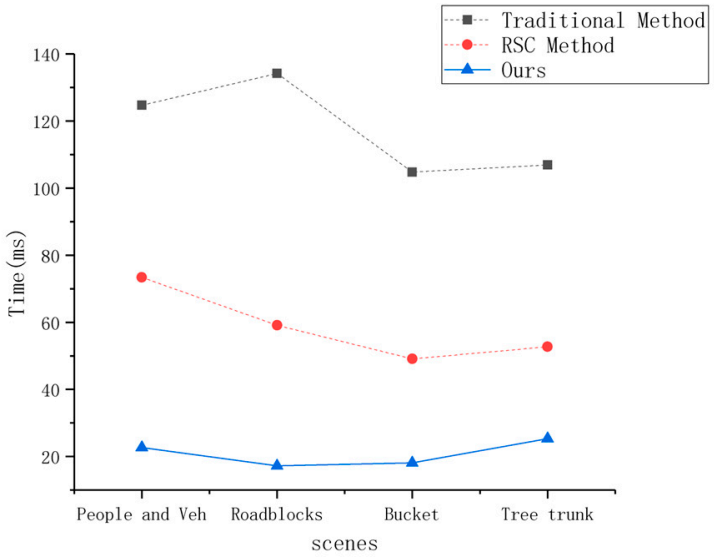


Figure 11. Clustering time comparison.

4.3. Image Recognition Results

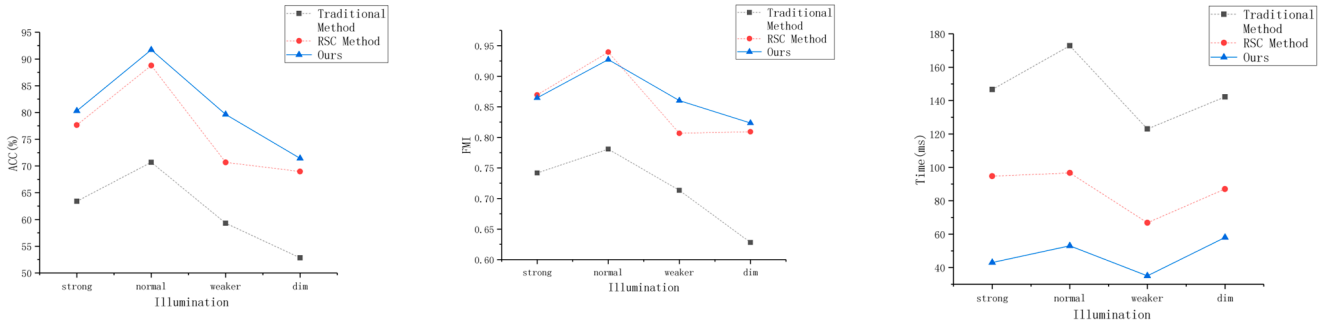
The internal metrics of clustering can only prove the clustering results internally, whether it can be applied to the actual scenario also needs to verify the obstacle detection performance. We label all obstacles as positive samples by manual labeling, and ground shadows, noise, or other non-existent situations as negative samples. We also evaluate the detection performance of different methods by the commonly used clustering performance metrics Accuracy (ACC) and Fowlkes and Mallows Index (FMI).

Table 2 shows the detection of each method under the test of 100 images. From the figure, we can find that the detection effect of the method in this paper is better, and the accuracy is 4.13% and 19.34% higher compared with RSC and traditional methods, respectively.

Table 2. Obstacle detection results.

	ACC	FMI
Traditional Method	63.07%	0.7287
RSC Method	78.28%	0.8679
Ours	82.41%	0.8773

In the proposed process of us, the feature point extraction method is seriously affected by illumination. We divide the samples into strong illumination, normal illumination, weaker illumination, and dim illumination according to the illumination condition, which is used to test the overall robustness of the process under different illumination. The detection results are shown in Figure 12. From the figure, we can see that (1) the method in this paper is influenced by the change of light, and the highest accuracy rate is 91.72% in normal light, while the accuracy rate is only 71.4% in dim light, which needs to consider the influence of environment when applying; (2) the recognition of the method in different environments in this paper is better compared with other methods. Although there are fluctuations in different environments, the overall difference is similar to the gap in the mixed sample. (3) In terms of specific detection time, the detection frame rate of the method in this paper can reach 20FPS, which can meet the real-time requirements of the application.



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

Figure 12. Barrier detection results under different lighting conditions. (**a**) ACC comparison results, (**b**) FMI comparison results, and (**c**) Comparison of time spent on the whole process.

Deep learning is a popular direction in the field of image recognition, and we compare our method in this paper with the deep learning methods YOLOv7 and Swin–Transformer YOLOv5 to verify the performance of our method. Deep learning methods are trained using only normally illuminated images, and the metrics are shown in the following Table 3.

Table 3. Comparison of performance between this method and other methods.

	ACC	Precision	Recall	F1	Time
YOLOv7	81.47%	83.60%	82.96	84.38	0.232 s
Swin-Transformer YOLOv5	85.11%	92.54%	80.921	86.87	0.224 s
Ours	81.72%	85.54%	90.22	86.72	0.053 s

From the data, the method in this paper still lags behind the deep learning method in terms of performance metrics but has four times the detection speed of the deep learning method in embedded.

4.4. Accuracy Test Results

The ultimate goal of obstacle detection applied to real-world scenarios is to ensure the robot's safety when it moves. Therefore, it is necessary to analyze whether the robot can completely detect obstacles and the accuracy of the obstacle information acquired by the robot. In the previous section, we discussed recognizing single-frame objects in different scenarios but did not discuss whether complete recognition can be guaranteed. In this section, we verify whether our method meets the safety requirements of the robot by varying the distance and lighting conditions. In our experiments, we found that the probability of successful recognition by the robot increases as the distance between the robot and the obstacle gradually decreases. This situation is shown in Figure 13.

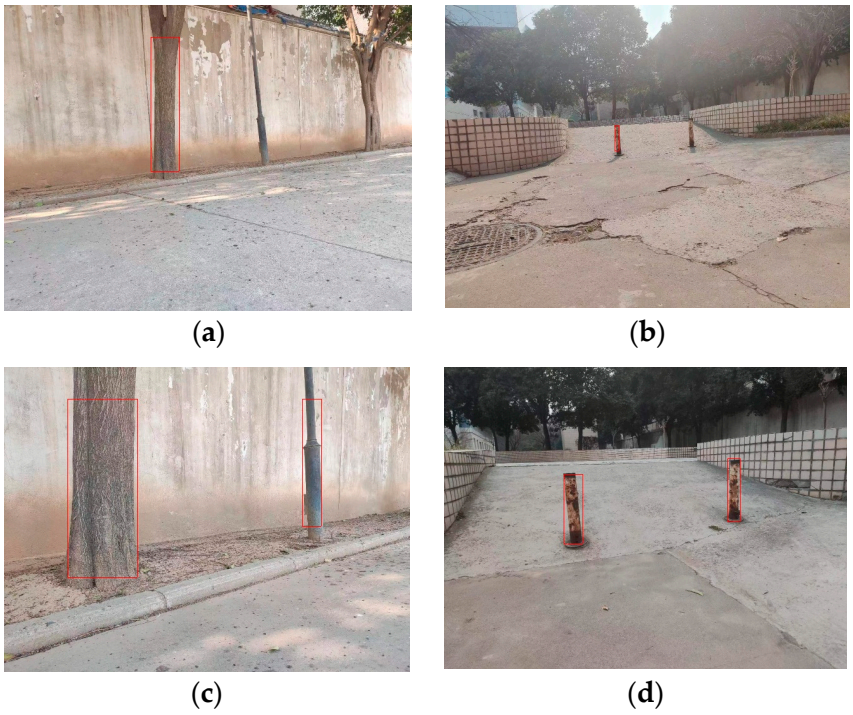


Figure 13. Different distance recognition effect. (**a**,**b**) Remote identification situation; (**c**,**d**) Close range recognition situation.

We conducted 20 sets of experiments in different scenarios to verify the relationship between the distance between the robot and the obstacle and the detection success rate, and the final results are shown in Table 4.

Table 4. Variation of detection success rate [%] with distance.

	3 [m]	**2.5 [m]**	**2 [m]**	**1.5 [m]**	**1.25 [m]**	**1 [m]**
Strong Illumination	78.2	88.9	94.6	100	100	100
Normal Illumination	87.3	93.3	97.1	100	100	100
Weaker Illumination	70.8	82.8	88.4	92.5	97.2	100
Dim Illumination	0	0	52.4	77.3	93.7	100

The success rate in the table is the number of detected samples as a percentage of the total number of samples and the distance in meters.

The results show that the recognition success rate of the proposed method varies with the distance of the obstacle. Under non-extreme conditions, the recognition success rate exceeds 90% at distances of up to two meters. When the obstacle is within 1 m, full recognition is possible even in dim light conditions, which proves that our method possesses the ability of full recognition.

The second issue to consider is the recognition accuracy. We placed a cylindrical obstacle with a diameter of 9 cm in the four scenes mentioned above to investigate the effect of illumination and distance on detection accuracy. The results are shown in Figure 14.

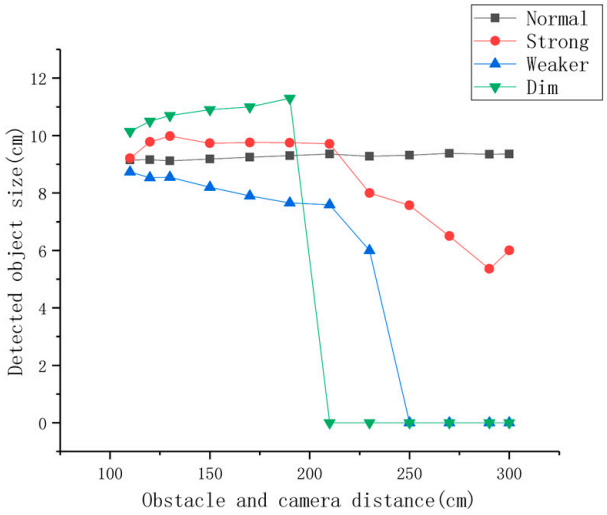


Figure 14. Recognition accuracy concerning illumination and distance.

From the results, our method guarantees the accuracy of obstacle detection up to a distance of three meters in normal light; when the light changes, it has reliable detection results within 1.5 m; between 1.5 m and 2 m large errors start to appear, with errors of around 2 cm; beyond 2 m huge errors start to appear. The results of the precision experiments show that the clustering algorithm proposed in this paper can accurately obtain the edges of the obstacle, further verifying that the clustering performance of the algorithm is excellent.

The above two experiments demonstrate that our method can ensure the robot's safety during its movement and can be applied to real work scenarios.

4.5. Map Mapping Experiment

To verify the practicality of our method, we arranged for a pedestrian to walk in front of the robot that was walking and tested whether the robot could detect the pedestrian and perform obstacle avoidance. The results are shown in Figure 15. In Figure 15a, the robot can quickly identify the obstacle and give the spatial coordinates of the obstacle, and in Figure 15b, the addition of the obstacle to the SLAM map is shown.

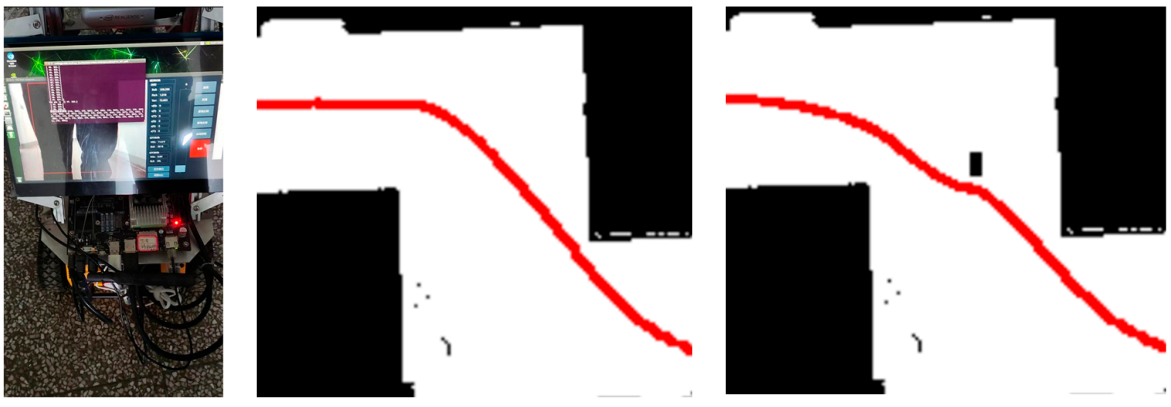


Figure 15. Obstacle detection and projection onto SLAM maps. (**a**) Obstacle detection; (**b**) Mapping obstacles to maps.

5. Conclusions

In conclusion, this paper presents a fast obstacle-detection process for RGB-D cameras mounted on a wheeled robot. In this process, the image pre-processing part adjusts the parameters according to the changes in the environment, and the noise screening part is modified based on the referenced method according to the wheeled robot platform which we used. The main contribution is to propose a new hierarchical clustering method for this flow. The basic principle of this method is that for obstacles in the field of view of the RGB-D camera, there are two relationships between feature points on the same object contour in depth, one is within the same depth, and the other is that the depth interval between adjacent feature points in depth are within a certain distance value, i.e., they are continuously distributed in depth, and the obstacles with different depths are grouped and then further segmented from the horizontal direction. To verify the performance of the proposed clustering method and detection process, we compare it with the existing methods, and experimentally prove that the proposed method has similar performance with the RSC method in terms of the internal index of clustering, which is much faster than the traditional method, but more than one time faster than the RSC method in terms of clustering speed. The overall detection results are 4% and 19% higher than those of RSC and traditional methods, respectively, and the average detection speed is more than 20 FPS. Regarding detection accuracy, it can reach 2 cm within 3 m under normal illumination. In summary, the proposed method can meet the detection needs of wheeled robots in low-speed scenarios.

6. Discuss

In this paper, we verify the feasibility of our method by conducting experiments in different scenarios. From the experimental results, our method shows competitive recognition performance in all types of environments. However, overall, the recognition success rate of our method drops sharply when the illumination level darkens. For this, regularized illumination optimization with depth noise suppression for flash image enhancement is a feasible solution [29]. In future work, we will further investigate detection methods in dim light environments.

In addition, our method needs to modify the parameters continuously according to the actual environment during practical use. The main parameters are the boundary value of the Canny operator and the depth interval parameter to determine whether the feature points of an obstacle are continuously distributed in depth. The Canny operator mainly affects the clustering time, Table 5 shows the detection time corresponding to different boundary values of the Canny operator under normal illumination without affecting the detection success rate.

Table 5. Relationship between Canny boundary values and detection time.

Lower boundary value	100	125	150
Upper boundary value	200	250	300
Detection time	92.81 ms	83.55 ms	56.25 ms

The greater the value of the depth interval parameter, the more likely different obstacles will be grouped into the same category, and a smaller parameter will lead to the same object being divided into multiple layers. Therefore, in practical use, the width that the robot can pass is often used as the parameter, and two obstacles that are impassable between them can be considered as the same obstacle.

To ensure detection efficiency, we tend to increase the boundary value of Canny, and to better distinguish obstacles, we decrease the depth interval parameter. However, when the light darkens, the obstacle contour becomes blurred, the feature points will become fewer, and the depth interval parameter will affect the detection success rate when it is too small. How to summarize the law of parameter change with the environment will be the focus of the next research.

Author Contributions: Conceptualization, C.L. and S.X.; Methodology, C.L. and S.X.; Software, S.X. and X.S.; Validation, S.X. and Y.H.; Formal analysis, F.Y. and X.Y.; Data curation, X.M.; Writing—original draft, S.X.; Writing—review & editing, C.L. and S.X.; Visualization, S.X.; Supervision, Y.H.; Project administration, C.L. and N.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (52105574), the Henan science and technology research plan project (222102220079), the Training plan for young backbone teachers in universities of Henan Province (2019GGJS082), the Basic research plan project of key scientific research projects of universities in Henan Province (17A460003), and Henan Science and Technology Project (212102210370).

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

References

1. Hongkun, T.; Tianhai, W.; Yadong, L.; Xi, Q.; Yanzhou, L. Computer vision technology in agricultural automation—A review. *Inf. Process. Agric.* **2020**, *7*, 1–19.
2. Norris, W.R.; Patterson, A.E. System-Level Testing and Evaluation Plan for Field Robots: A Tutorial with Test Course Layouts. *Robotics* **2019**, *8*, 83. [CrossRef]
3. Ilesanmi, D.; Vincent, B.; Adefemi, A.; Bankole, O.; Johnson, K.P.; Khumbulani, M. Development and Performance Evaluation of a Robot for Lawn Mowing. *Procedia Manuf.* **2020**, *49*, 42–48.
4. Ochman, M. Hybrid approach to road detection in front of the vehicle. *IFAC PapersOnLine* **2019**, *52*, 245–250. [CrossRef]
5. Bietresato, M.; Carabin, G.; D'Auria, D.; Gallo, R.; Ristorto, G.; Mazzetto, F.; Vidoni, R.; Gasparetto, A.; Scalera, L. A trackedmobile robotic lab for monitoring the plants volume and health. In Proceedings of the 2016 12th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), Auckland, New Zealand, 29–31 August 2016; pp. 1–6.
6. Lu, L.; Redondo, C.; Campoy, P. Optimal frontier-based autonomous exploration in unconstructed environment using RGB-D sensor. *Sensors* **2020**, *20*, 6507. [CrossRef] [PubMed]
7. Lu, D.V.; Hershberger, D.; Smart, W.D. Layered costmaps for context-sensitive navigation. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 709–715.
8. Ward, J.H., Jr. Hierarchical grouping to optimize an objective function. *J. Am. Stat. Assoc.* **1963**, *58*, 236–244. [CrossRef]
9. Sneath, P.H.A.; Sokal, R.R. *Numerical Taxonomy: The principles and Practice of Numerical Classification*; W.H. Freeman & Co. Ltd.: New York, NY, USA, 1973.
10. King, B. Step-wise clustering procedures. *J. Am. Stat. Assoc.* **1967**, *62*, 86–101. [CrossRef]
11. Rokach, L. Chapter 14. In *Data Mining and Knowledge Discovery Handbook*; Springer: New York, NY, USA, 2010.
12. Zhang, T.; Ramakrishnan, R.; Livny, M. BIRCH: A new data clustering algorithm and its applications. *Data Min. Knowl. Discov.* **1997**, *1*, 141–182. [CrossRef]
13. Kobren, A.; Monath, N.; Krishnamurthy, A.; McCallum, A. A hierarchical algorithm for extreme clustering. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 255–264.
14. Guha, S.; Rastogi, R.; Shim, K. ROCK: A robust clustering algorithm for categorical attributes. *Inf. Syst.* **2000**, *25*, 345–366. [CrossRef]
15. Guha, S.; Rastogi, R.; Shim, K. Cure: An efficient clustering algorithm for large databases. *Inf. Syst.* **2001**, *26*, 35–58. [CrossRef]
16. Karypis, G.; Han, E.-H.; Kumar, V. Chameleon: Hierarchical clustering using dynamic modeling. *Computer* **1999**, *32*, 68–75. [CrossRef]
17. Xie, W.-B.; Lee, Y.L.; Wang, C.; Chen, D.B.; Zhou, T. Hierarchical clustering supported by reciprocal nearest neighbors. *Inf. Sci.* **2020**, *527*, 279–292. [CrossRef]
18. Shah, S.A.; Koltun, V. Deep continuous clustering. *arXiv* **2018**, arXiv:1803.01449, 1–11.
19. Ren, Y.; Wang, N.; Li, M.; Xu, Z. Deep density-based image clustering. *Knowl. Based Syst.* **2020**, *197*, 105841. [CrossRef]
20. Sun, X.; Ma, H.; Sun, Y.; Liu, M. A Novel Point Cloud Compression Algorithm Based on Clustering. *IEEE Robot. Autom. Lett.* **2019**, *4*, 2132–2139. [CrossRef]
21. Zhang, Z.; Liu, L.; Shen, F.; Shen, H.T.; Shao, L. Binary multi-view clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *41*, 1774–1782. [CrossRef]
22. Dulău, M.; Oniga, F. Obstacle Detection Using a Facet-Based Representation from 3-D LiDAR Measurements. *Sensors* **2021**, *21*, 6861. [CrossRef]
23. Chu, Z.; He, J.; Zhang, X.; Zhang, X.; Zhu, N. Differential Privacy High-Dimensional Data Publishing Based on Feature Selection and Clustering. *Electronics* **2023**, *12*, 1959. [CrossRef]
24. Wang, H.; Fan, R.; Sun, Y.; Liu, M. Dynamic Fusion Module Evolves Drivable Area and Road Anomaly Detection: A Benchmark and Algorithms. *IEEE Trans. Cybern.* **2021**, *52*, 10750–10760. [CrossRef]
25. Caltagirone, L.; Bellone, M.; Svensson, L.; Wahde, M. LIDAR-Camera Fusion for Road Detection Using Fully Convolutional Neural Networks. *Robot. Auton. Syst.* **2019**, *111*, 125–131. [CrossRef]

26. Chu, P.; Cho, S.; Sim, S.; Kwak, K.; Cho, K. A Fast Ground Segmentation Method for 3D Point Cloud. *J. Inf. Process. Syst.* **2017**, *13*, 491–499.
27. Chen, L.; Yang, J.; Kong, H. Lidar-histogram for fast road and obstacle detection. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 1343–1348.
28. Asvadi, A.; Premebida, C.; Peixoto, P.; Nunes, U. 3D Lidar-based Static and Moving Obstacle Detection in Driving Environments. *Robot. Auton. Syst.* **2016**, *83*, 299–311. [CrossRef]
29. Guo, Y.; Lu, Y.; Liu, R.W.; Yang, M.; Chui, K.T. Low-light image enhancement with regularized illumination optimization and deep noise suppression. *IEEE Access* **2020**, *8*, 145297–145315. [CrossRef]