Localization of Mobile Robots Based on Depth Camera

Zuoliang Yin 1, Huaizhi Wen 2, Wei Nie 2,* and Mu Zhou 2

1 College of Electronics and Information, Qingdao University, Qingdao 266071, China; yinzuoliang@qdu.edu.cn
2 School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China; s200131183@stu.cqupt.edu.cn (H.W.); zhoumu@cqupt.edu.cn (M.Z.)

* Correspondence: niewei@cqupt.edu.cn

Abstract: In scenarios of indoor localization of mobile robots, Global Positioning System (GPS) signals are prone to loss due to interference from urban building environments and cannot meet the needs of robot localization. On the other hand, traditional indoor localization methods based on wireless signals such as Bluetooth and WiFi often require the deployment of multiple devices in advance, and these methods can only obtain distance information and are unable to obtain the attitude of the positioning target in space. This paper proposes a method for the indoor localization of mobile robots based on a depth camera. Firstly, we extracted ORB feature points from images captured by a depth camera and performed homogenization processing. Then, we performed feature matching between adjacent two frames of images, and the mismatched points are eliminated to improve the accuracy of feature matching. Finally, we used the Iterative Closest Point (ICP) algorithm to estimate the camera’s pose, thus achieving the localization of mobile robots in indoor environments. In addition, an experimental evaluation was conducted on the TUM dataset of the Technical University of Munich to validate the feasibility of the proposed depth-camera-based indoor localization system for mobile robots. The experimental results show that the average localization accuracy of our algorithm on three datasets is 0.027 m, which can meet the needs of indoor localization for mobile robots.

Keywords: indoor localization; mobile robot; visual SLAM

1. Introduction

Outdoor mobile robots can accurately obtain their own positioning information through localization technologies such as GPS [1]. However, satellite signals are prone to be blocked by the reinforced concrete structure in buildings for indoor localization scenarios. GPS is susceptible to interference, and the signals are weak and unstable, making it unable to provide real-time and accurate positioning information to mobile robots. This will greatly limit the application of mobile robots in indoor environments, such as indoor security prevention and control and autonomous inspection. As is well known, indoor positioning methods include LiDAR [2], motion capture systems [3], and localization technologies such as WiFi and Bluetooth [4]. Some common localization schemes based on laser radar, such as Gmapping [5] and Hector [6], have high localization accuracy; in addition to camera-assisted localization, laser radar is also an alternative sensor to help robots locate. The authors of [7] show a data acquisition and analysis platform for an auto drive system, hydro 3d, which uses 3D laser radar to detect and track hybrid objects with cooperative perception. The authors of [8] propose an improved yolov5 algorithm based on transfer learning to detect tassels in RGB UAV images, but the equipment is often heavy and expensive, which is not suitable for some lightweight mobile robots. A motion capture system (MCS) obtains the relative position of a target in space by deploying multiple high-speed cameras. The disadvantage of this approach is that it requires the deployment of equipment in advance, and the range of areas that can achieve localization is limited. On the other hand, it is unrealistic to deploy devices in advance in mobile robot application scenarios such as disaster prevention and rescue and hazardous area...
detection. Ultra-wideband (UWB) localization mainly adopts Bluetooth and WiFi methods. However, this type of method is greatly influenced by the environment and often requires the deployment of multiple devices. In addition, this method can only obtain distance information and cannot determine the posture of the target in space. In today’s diverse sensor world, relying solely on pure vision to achieve autonomous localization may face performance bottlenecks and algorithm robustness issues in extreme scenarios, such as environments with few or no textures. In this regard, other sensors can be integrated for positioning, such as inertial sensors [9]. The latest research on the visual SLAM algorithm for integrating inertial sensors is presented in [10]. The disadvantage of this approach is that it increases the complexity of the algorithm and hardware usage costs.

The visual Simultaneous Localization and Mapping (SLAM) method provides a solution for the localization of mobile robots without GPS signals [11]. It does not require the deployment of additional devices; relying solely on its own camera and utilizing the SLAM algorithm, it can achieve its own localization. The SLAM algorithm is mainly divided into two parts: localization and mapping [12]. The localization function mainly calculates the position and posture of the robot in the current environment based on the camera input image and the principle of multi-view geometry [13]. The purpose of creating a map is to establish a corresponding relationship with the surrounding environment. The visual SLAM algorithm can be divided into two categories based on the methods of motion estimation and scene structure restoration: the feature point method and the direct method [14]. The feature point method establishes the spatial geometric relationship between adjacent frames of images through the extraction and matching steps of image feature points [15], thereby performing motion estimation. On the basis of the assumption that the grayscale remains unchanged, the direct method performs motion estimation by minimizing the grayscale difference between adjacent frames of images [16]. Compared with the direct method, the feature point method has better adaptability to the environment due to the presence of robust image feature points. Therefore, we adopt the feature point method.

The concept of SLAM was first proposed by Dr. Smith in 1986. He modeled the SLAM problem in the form of filtering, used motion and observation equations to describe the entire SLAM process, and finally solved the camera’s pose by optimizing noise [17]. In 2007, Davison proposed a real-time-running monocular SLAM system. It uses extended Kalman filtering as the backend to track very sparse feature points in the front end, but there is a situation where feature points are easily lost, leading to unstable or even interrupted system operation [18]. In 2008, reference proposed a miniaturized SLAM system that parallelized the localization and mapping processes using two threads, with high running speed. However, its drawbacks are also quite obvious: small scene size and easy tracking loss [19]. In 2014, a reference proposed a form of SLAM based on the monocular direct method, which does not require feature point extraction and can construct semi-dense maps. However, it requires GPU acceleration, increases costs, and is time-consuming [20].

In summary, we propose a depth-camera-based indoor localization system for mobile robots in application scenarios. The authors’ contribution in this paper is to upgrade the accuracy of the localization algorithm by improving the distribution uniformity of ORB feature points and feature-matching accuracy. The improved algorithm module was tested to demonstrate the performance of the improved method. Finally, the entire localization algorithm was tested on a public dataset. This system not only achieves autonomous localization and map-building functions for mobile robots in unknown environments, but also has the advantages of low cost and easy deployment.

2. Methods

This section introduces a solution for indoor localization of mobile robots based on depth cameras, mainly including ORB feature extraction and homogenization processing, feature matching and removal of mismatching feature points, and pose estimation algorithms.
2. Methods

This section introduces a solution for indoor localization of mobile robots based on depth cameras. The system mainly includes four parts, and the system diagram of the algorithm is shown in Figure 1. The first step is to extract the ORB feature points of the current frame image. In order to prevent the distribution of feature points in local areas of the image from being too dense and improve the ability of the image to extract feature points in homogeneous areas, we adopt a feature point homogenization algorithm based on Quadtree structure. Then there is the ORB feature matching between adjacent two frames of images, and the presence of a large number of mismatches will affect the final localization accuracy. Therefore, we adopt a feature point mismatch removal algorithm based on cosine similarity to remove the mismatched points. Finally, real-time pose estimation of the camera is performed using pose estimation algorithms.

![Figure 1. Overall process of localization algorithm.](image)

2.1. System Framework

Our proposed indoor localization algorithm for mobile robots based on depth cameras mainly includes four parts, and the system diagram of the algorithm is shown in Figure 1. The first step is to extract the ORB feature points of the current frame image. In order to prevent the distribution of feature points in local areas of the image from being too dense and improve the ability of the image to extract feature points in homogeneous areas, we adopt a feature point homogenization algorithm based on Quadtree structure. Then there is the ORB feature matching between adjacent two frames of images, and the presence of a large number of mismatches will affect the final localization accuracy. Therefore, we adopt a feature point mismatch removal algorithm based on cosine similarity to remove the mismatched points. Finally, real-time pose estimation of the camera is performed using pose estimation algorithms.

2.2. ORB Feature Extraction

Feature points are some unique features in an image that can represent local features such as corners, blocks, and edges. Among them, corner and edge blocks have a higher degree of differentiation between different images than pixel blocks and perform more stably in small-scale image perspective transformations, but they do not meet the needs of image feature points in visual SLAM. In order to obtain more stable local features of images, scholars in the field of computer vision have conducted a lot of work and designed many image features that remain stable despite significant changes in perspective. The classic ones include Smallest Univalue Segment Assisting Nucleus (SUSAN), Harris, Features From Accelerated Segment Test (FAST), Shi Tomasi, Scale Invariant Feature Transform (SIFT) [21], Speeded-Up Robust Feature (SURF) [22], and Oriented FAST and BRIEF (ORB) [23]. Among them, the Harris and SIFT algorithms, as well as variations of these two methods, are widely used in traditional visual SLAM algorithms. The Harris algorithm adopts differential operations that are not sensitive to changes in image brightness, as well as grayscale second-order matrices. The principle of the SUSAN algorithm is similar to that of the Harris algorithm, and it performs well in corner and edge detection. During the operation of the visual SLAM algorithm, there are often changes in the distance of the camera's perspective, and SUSAN and Harris have weaker support for scale invariance. SIFT, also known as a scale-invariant feature algorithm, uses Gaussian differential functions and feature descriptors to ensure that images perform well in scenarios with changes in luminosity, rotation, and scale. However, the drawback of SIFT, namely its high computational complexity, is also evident, and SIFT is generally used in high-performance computing platforms and applications that do not consider real-time performance. SURF is an improved version of SIFT, but it still has difficulty meeting the real-time requirements in visual SLAM application scenarios without using GPU acceleration.

In this regard, the ORB feature points used in the visual SLAM system constructed in this paper are a compromise between accuracy and real-time performance. While appropriately reducing the accuracy of image feature extraction, it has resulted in a significant increase in computational speed to meet the real-time requirements of the visual SLAM
algorithm. The ORB feature points consist of an improved FAST corner point and a Binary Robust Independent Element Features (BRIEF) descriptor. The principle of FAST is shown in Figure 2.

![Figure 2. FAST corner diagram. The parameter \( p \) represents a pixel in the image.](image)

Assuming that there is a pixel \( P \) in the image with a brightness of \( I \), based on engineering experience, we set a brightness threshold \( T \) and select 16 pixels on the circumference with a radius of 3 pixels with \( a \) as the center. If there are consecutive \( N \) pixels with grayscale values greater than \( I + T \) or less than \( I - T \), pixel \( P \) is considered a corner. The overall FAST corner extraction algorithm process is shown in Figure 3. The ORB algorithm uses an image pyramid to downsample the image and then uses the gray centroid method to solve the problem of image rotation. The centroid refers to taking a gray value as the center of weight. Firstly, the moment of an image is defined as follows:

\[
m_{pq} = \sum_{x,y} x^p y^q I(x,y) \quad p, q = \{0, 1\}
\]  

(1)

![Figure 3. FAST corner extraction algorithm flow. Select a pixel \( p \) in the image with brightness \( I \) and \( T \) as the comparison threshold for grayscale values.](image)

The centroid of this image can be found through the following equation:

\[
C = \left( \begin{array}{c}
m_{10} \\
m_{01} \\
m_{00}
\end{array} \right)
\]  

(2)

We connect the centroid and geometric center of the image and then use a vector to represent it. Therefore, the direction of a feature point can be defined as:

\[
\theta = \arctan(m_{01}/m_{10})
\]  

(3)

After extracting ORB feature points from the image, the step is feature matching. In cases where the number of feature points is not large, brute force matching can be directly used. However, this method requires a lot of time when the number of feature points is
large. Therefore, a fast approximate nearest neighbor algorithm can be used to improve the matching speed.

2.3. Feature Point Homogenization

The dense distribution of image feature points in local areas can affect the accuracy of visual odometry, as using only local area data to represent the global area often results in errors. We adopt an improved ORB feature uniform distribution algorithm based on Quadtree for this purpose. This paper mainly improves the feature point extraction and feature description stages. Firstly, the threshold is calculated based on the grayscale values of different pixel neighborhoods, rather than manually setting the threshold to improve the algorithm’s ability to extract feature points in homogeneous image regions. Secondly, an improved Quadtree structure is adopted for feature point management and optimization to improve the distribution uniformity of detected feature points. To prevent excessive node splitting, threshold values are set in each layer of the pyramid image. In the feature description stage, image information included in the grayscale difference is used to obtain more information support and enhance the discriminative power of the feature description. The overall process of the algorithm is shown in Figure 4. The first step is to construct an eight-layer image pyramid. Assuming that the required number of ORB feature points is \( m \), the scale factor is \( s \), and the number of feature points in the first layer is \( a \),

\[
a + \frac{1}{s}a + \frac{1}{s^2}a + \frac{1}{s^3}a + \ldots + \frac{1}{s^7}a = m
\]  

(4)

![Diagram](image)

Figure 4. Overall process of feature point homogenization algorithm.

Based on the resolution of the image, each layer of the pyramid image is divided into grids, in order to make the FAST corner distribution more uniform. The next step is to calculate the feature point extraction threshold in each grid. Due to the fixed threshold of the ORB algorithm, it does not consider the specific situation of the local neighborhood of pixels. Therefore, the feature points detected by the ORB algorithm are concentrated in areas with significant changes in the image, and there are a large number of redundant feature points. For uniform areas of the image, due to the similarity of the grayscale features, the FAST algorithm can only extract a small number of feature points.
In this regard, this paper adopts a dynamic local threshold method. We set a separate threshold \( t \) within each grid, and the calculation formula is as follows:

\[
t = a \times \left( \frac{1}{m} \sum_{i=1}^{m} |f(Q_i) - \overline{f(Q)}| \right)
\]  

(5)

Here, \( a \) is the scaling factor, usually set to 1.2 based on engineering experience, \( m \) is the number of pixels in the grid, \( f(Q_i) \) is the grayscale value of pixels in the grid, and \( \overline{f(Q)} \) is the average grayscale value of all pixels in the grid.

After using dynamic thresholding to extract FAST corners, a relatively uniform distribution of feature points can be obtained, but there is a large number of redundant feature points. Therefore, for this paper, a Quadtree structure is used to further screen feature points. Quadtree, also known as Q-tree, has a maximum of four subnodes per node. Using a Quadtree structure, a two-dimensional space can be divided into four quadrants, and the information in a quadrant can be stored in a subnode. The Quadtree structure is shown in Figure 5. If the number of feature points in a subregion is greater than 1, the splitting will continue in each subregion, while regions without feature points will delete a subnode until the number of nodes equals 1 or reaches the required number of feature points for the system.

![Schematic diagram of Quadtree structure](image)

**Figure 5.** Schematic diagram of Quadtree structure. (a) Regional quadrant division; (b) Quadtree node distribution.

The original ORB feature descriptor only uses the size of grayscale values, without using the difference information between grayscale values, resulting in the loss of some image information. In this regard, this paper adopts a feature description method that integrates grayscale difference information, while using grayscale value size and grayscale difference information to describe feature points. Suppose there is a feature point \( Q \); according to the rules of BRIEF algorithm, we select the \( T \) group of pixel block pairs to compare the gray value, obtain a binary code string \( G_a \), and then record the gray difference of each pixel block pair to generate another binary code string. Assuming \( D_{w} \) is the dataset for recording grayscale differences,

\[
G = \{G_{i}\}_{i=1...T} = \{|f(A_i) - f(B_i)|\}_{i=1...T}
\]

(6)

Here, \( f \) is the average grayscale value of the pixel block. Then, we calculate the average value \( G_a \) of these \( T \) differences as the threshold for binary encoding these grayscale differences, and the calculation formula is (7). We compare these \( T \) grayscale differences with \( G_a \), as shown in Formula (8), and then obtain a binary string \( D_{w}^{b} \), as shown in Formula (9). Finally, we combine \( D_{w} \) and \( D_{w}^{b} \) to form a new feature descriptor \( D_{w}^{b} \), as shown in Formula (10).

\[
G_a = \frac{G}{T} = \frac{1}{T} \sum_{1 \leq i \leq T} G_i
\]

(7)
\[ h(Q, G_i) = \begin{cases} 1 & G_i > G_a \\ 0 & G_i < G_a \end{cases} \]  

\[ D_w^\wedge = \sum_{1 \leq t \leq T} 2^t h(Q, G_t) \]  

\[ D_k = D_w + D_w^\wedge \]  

2.4. Feature Matching

In traditional ORB feature-matching methods, feature vectors are matched by comparing binary strings without considering the correlation between the components of the feature vectors. Therefore, when there are similar regions in the image, there will be more feature mismatches, such as two document images with multiple identical characters. The Hamming distance represents the difference in numerical features between two vectors, which can effectively determine whether two vectors are similar but does not have the ability to describe the directional features of two vectors. We adopt an ORB feature-matching algorithm that combines Hamming distance and cosine similarity. Firstly, we use Hamming distance for coarse matching, then we use cosine similarity for further filtering, and finally we use the Random Sample Consistency (RANSAC) algorithm to further eliminate mismatches. The overall framework of the algorithm is shown in Figure 6.

![Figure 6. Overall process of Random Sample Consistency (RANSAC) algorithm.](image)

The algorithm first performs a preliminary screening of ORB feature points waiting for matching based on Hamming distance and then calculates cosine similarity based on this. Assuming there are two ORB feature description vectors \( a \) and \( b \), the cosine similarity between them can be obtained:

\[ \cos(\theta) = \frac{a \cdot b}{\|a\| \times \|b\|} \]  

Here, the larger the cosine similarity value, the closer the directions of the two vectors are. The next step is to calculate the comparison threshold for cosine similarity. The specific method is to subtract the cosine similarity of all feature point pairs and then find the value with the highest occurrence frequency from it as the cosine similarity comparison threshold for this feature matching. Considering the existence of errors, a floating value is usually
set to 0.3 based on engineering experience. Next, the ORB feature points can be re-filtered based on the cosine similarity comparison threshold, and those feature point pairs that are not within the threshold range can be removed.

Finally, we use the RANSAC algorithm to remove the mismatched feature pairs. Although the previous steps can ensure that most of the matches are correct, there may still be some cases of mismatches. Because these two methods are based on the local characteristics of the image, it is necessary to remove the mismatched feature pairs from the perspective of the entire image. The RANSAC algorithm can eliminate outliers in a dataset by calculating the best model of the data which are also known as outer points, and it can retain normal values that conform to this model, which are also known as inner points.

The algorithm first randomly selects four pairs of feature points from the dataset and then calculates the homography matrix $H$, which is the initialization of the optimal model. We use the homography matrix $H$ to test all data. If the error between the coordinate point and the actual result obtained through homography matrix transformation is small enough, then the point is considered an interior point. If there are not enough internal points, we select new random points and recalculate the new homography matrix model until the internal points are sufficient. The minimum number of iterations $k$ of the RANSAC algorithm should meet the following:

$$
1 - P_m = (1 - \eta^m)^k \quad \quad k = \frac{\ln(1 - P_m)}{\ln(1 - \eta^m)}
$$

(12)

Here, the minimum amount of data required to calculate the homography matrix $H$ is $m$; $P_m$ is the confidence level, representing the probability of at least one inner point in $m$ ORB feature point pairs; and $\eta$ represents the probability of the selected ORB feature point pair being an outer point.

2.5. Camera Pose Estimation

After the ORB feature extraction and matching steps described in the previous section, a correctly matched and evenly distributed pair of feature points can be obtained, and then the camera motion can be estimated based on this. The Iterative Closest Point (ICP) algorithm can calculate the camera’s pose changes based on two RGB-D images. Assuming that after the feature-matching step, there is a set of paired 3D points,

$$
P = \{p_1, \ldots, p_n\}
$$

$$
P' = \{p'_1, \ldots, p'_n\}
$$

(13)

According to the camera rigid body motion theory, the camera motion can be represented by the rotation matrix $R$ and the translation vector $t$:

$$
\forall i, p_i = Rp'_i + t
$$

(14)

The ICP algorithm constructs the solution process of camera motion into a least-square form, assuming that the error between the matched ORB feature points is

$$
e_i = p_i - (Rp'_i + t)
$$

(15)

This error term is constructed into a least-square problem, and the camera position and pose at this time can be obtained by minimizing the sum of error squares:

$$
\min f(R, t) = \frac{1}{2} \sum_{i=1}^{n} \|p_i - (Rp'_i + t)\|^2
$$

(16)
Before solving Formula (16), we define the centroids of these two matched ORB feature points and then simplify the formula through the centroids:

\[
\begin{align*}
    p &= \frac{1}{n} \sum_{i=1}^{n} p_i \\
    p' &= \frac{1}{n} \sum_{i=1}^{n} p'_i
\end{align*}
\]  

(17)

\[
\min f(R, t) = \frac{1}{2} \sum_{i=1}^{n} \| p_i - p - R(p'_i - p') \|^2 + \| p - (Rp'_i + t) \|^2
\]  

(18)

where the left term on the right side of the equation is only related to the rotation matrix \( R \). If the rotation matrix \( R \) is known, we order the right term on the right side of the equation to be equal to 0 to obtain the translation vector \( t \). So, ICP can be solved in three steps, first calculating the centroid coordinates of these two matched ORB feature points:

\[
\begin{align*}
    q_i &= p_i - p \\
    q'_i &= p'_i - p'
\end{align*}
\]  

(19)

Then, the second step is to calculate the rotation matrix \( R \) according to Formula (20), and the last step is to obtain the translation vector \( t \) according to Formula (21):

\[
R = \underset{R}{\arg\min} \frac{1}{2} \sum_{i=1}^{n} \| q_i - Rq'_i \|^2
\]  

(20)

\[
t = p - Rp'
\]  

(21)

3. Experimental Environment and Datasets

The data processing platform for this experiment was an Intel (R) Core (TM) i3-8100M CPU @ 2.60GHz main frequency, 16GB memory laptop, running the Ubuntu 16.04 operating system. The development software was Microsoft Visual Studio 2019 and the image algorithm library OpenCV 2.4.10, and the programming was implemented in C++ language.

In the experiments of feature extraction, homogenization, and feature matching, the test image data were obtained using the Oxford Optical Image Dataset provided by Schmid, which includes multiple image categories [24], including images with different lighting intensities, perspectives, blurring levels, and resolutions, as shown in Figure 7. The Trees dataset contains a set of images with complex textures. The images in the Bikes dataset have varying degrees of blur, the compression of the Ubc dataset is different, the contrast of the Leuven dataset is different, and the images in the Boat dataset are rotated to a certain extent.

The TUM dataset released by the Computer Vision Laboratory of the Technical University of Munich as experimental data [25] was used in the localization and mapping experiment. This dataset is composed of multiple image sequences collected by a depth camera in two different indoor environments. Each image sequence contains color images and depth images, as well as the real camera motion track collected by the motion capture system, which is the GroundTruth.txt file in the dataset folder. In addition, the dataset is time-synchronized with the motion capture system based on camera calibration. This experiment uses three image sequences from the TUM dataset, namely fr1/xyz, fr1/desk, and fr1/desk2. Among them, fr1/xyz contains a set of image sequences recorded by a handheld depth camera, which only performs slow translation motion in the main axis direction, making it more suitable for evaluating trajectory accuracy. In the fr1/desk scene, an office environment is set up, which includes two desks with computer monitors, keyboards, books, and other objects. The environment texture is relatively rich, and it describes the scene where a handheld depth camera surrounds the desk for a week. Compared to fr1/desk, although the scene is the same, fr1/desk2 contains more objects, and the camera
moves faster. The detailed information on the three datasets selected for the experiment is shown in Table 1.

![Figure 7. Oxford Optical Dataset. (a) Bikes; (b) Boat; (c) Graf; (d) Leuven; (e) Trees; (f) Ubc.](image)

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Trajectory Length</th>
<th>Number of Frames</th>
<th>Average Angular Velocity</th>
<th>Translational Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>fr1/xyz</td>
<td>7.11 m</td>
<td>798</td>
<td>8.920 deg/s</td>
<td>0.244 m/s</td>
</tr>
<tr>
<td>fr1/desk</td>
<td>18.88 m</td>
<td>2965</td>
<td>23.33 deg/s</td>
<td>0.293 m/s</td>
</tr>
<tr>
<td>fr1/desk2</td>
<td>10.16 m</td>
<td>640</td>
<td>29.31 deg/s</td>
<td>0.430 m/s</td>
</tr>
</tbody>
</table>

Table 1. Test Dataset Details.

4. Technical Implementation and Results
4.1. Feature Point Extraction and Homogenization

In order to verify the effectiveness of the uniform ORB algorithm based on a dynamic threshold and improved Quadtree proposed in Section 2 of this paper, an experimental analysis was carried out from the two aspects of feature point distribution uniformity and calculation efficiency. Six image groups were selected from the Oxford dataset for experimentation, namely Bikes (blur), Boat (rotation and scaling), Graf (viewpoint), Leuven (illumination), Trees (blur), and Ubc (JPG compression). As a comparison, two types of feature point algorithms were used for comparative experiments, namely the feature point algorithm based on floating-point feature description and the feature point algorithm based on binary feature description.

The algorithms based on floating-point feature description include SIFT and SURF, while the algorithms based on binary feature description include ORB and the methods used in ORB-SLAM (represented by the symbol MRA in this paper). Finally, to ensure the generality of the test results, 30 experiments were conducted on each set of image datasets, with the average value as the final experimental result.

Firstly, the performance differences in the uniform distribution of feature points among these five algorithms can be intuitively observed through observation. Figure 8 shows the feature point extraction results of the Bikes image dataset using the Oxford dataset. It can be seen that the feature points extracted by SIFT, SURF, and the original ORB algorithm are less evenly distributed than those extracted by MRA and the algorithm in this paper, and they are more concentrated in areas with obvious local areal feature characteristics of the image, such as doors and windows, which are more prominent edge image areas; however, there are not many feature points extracted in homogeneous areas such as walls. The feature points extracted by MRA and our algorithm are significantly more evenly distributed. 
distributed than those extracted by other algorithms. Compared with the MRA algorithm, the method proposed in this paper can extract feature points that MRA cannot extract in areas where the image features are not obvious, such as the wall area in the upper right corner of the image.

![Image](image_url)

**Figure 8.** The feature point extraction results of five algorithms on the Bikes image dataset of Oxford dataset. (a) SIFT; (b) SURF; (c) ORB; (d) MRA; (e) algorithm proposed in this paper.

In order to quantify the distribution of feature points, a feature point distribution uniformity function is used, where a smaller uniformity value indicates a more uniform distribution of feature points. The experimental results are shown in Table 2.

**Table 2.** Average uniformity of characteristic points of five algorithms on Oxford dataset.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SIFT</th>
<th>SURF</th>
<th>ORB</th>
<th>MRA</th>
<th>This Paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bikes</td>
<td>189.54</td>
<td>254.09</td>
<td>171.85</td>
<td>141.27</td>
<td>140.64</td>
</tr>
<tr>
<td>Boat</td>
<td>201.34</td>
<td>201.34</td>
<td>148.83</td>
<td>111.05</td>
<td>92.43</td>
</tr>
<tr>
<td>Graf</td>
<td>158.66</td>
<td>159.55</td>
<td>120.23</td>
<td>84.45</td>
<td>84.37</td>
</tr>
<tr>
<td>Leuven</td>
<td>234.04</td>
<td>273.04</td>
<td>175.45</td>
<td>158.66</td>
<td>168.16</td>
</tr>
<tr>
<td>Trees</td>
<td>206.39</td>
<td>231.24</td>
<td>146.66</td>
<td>112.15</td>
<td>113.32</td>
</tr>
<tr>
<td>Ubc</td>
<td>253.76</td>
<td>269.88</td>
<td>180.09</td>
<td>168.16</td>
<td>166.01</td>
</tr>
</tbody>
</table>

Among the six scenes in the Oxford image dataset, the uniformity values of MRA and our algorithm are smaller, indicating a more uniform distribution of feature points. Meanwhile, compared to MRA, our algorithm can be applied to more scenarios. For example, in the boat image group, our algorithm has improved uniformity by about 17% compared to MRA, indicating that our algorithm has a higher tolerance for image rotation and scaling. Compared to the original ORB algorithm, our algorithm has improved uniformity by 53%.

4.2. Feature Matching and Mismatching Elimination

The content of this section is the verification of the effectiveness of the improved ORB matching algorithm based on cosine similarity proposed in Section 2.4 of this paper. The evaluation criterion is feature-matching accuracy.

The experiment used three algorithms for comparison, namely the original ORB algorithm, the method used in ORB-SLAM (represented by the symbol MRA in this paper), and the improved algorithm proposed by the authors. Figure 9 shows the ORB feature point matching results of the Bikes image dataset using the Oxford dataset. The feature points
connected by green lines represent correct feature matching, while the red lines represent mismatching. From the feature point matching results of the three algorithms, it can be seen that MRA and our algorithm have fewer red lines, indicating fewer mismatching phenomena.

![Feature matching results](image_url)

**Figure 9.** ORB feature point matching results of three algorithms on Bikes image dataset of Oxford dataset. (a) ORB; (b) MRA; (c) proposed algorithm.

To ensure the generality of the test results, 30 experiments were conducted on each set of image datasets, with the average value as the final experimental result, and the average accuracy is shown in Table 3. From the experimental results, it can be seen the matching accuracy of MRA and our algorithm on these six image sets is higher than that of the original ORB algorithm. Compared with MRA, our algorithm does not have much difference in feature-matching accuracy on the Bikes, Graf, Leuven, and Ubc datasets, but it has significantly improved matching accuracy on the Boat dataset. The reason is that our algorithm uses cosine similarity to screen the matching point pairs after the initial matching of Hamming distance, removing some mismatched point pairs caused by image rotation. After calculation, compared with the original ORB algorithm, the average feature-matching accuracy of our algorithm on six datasets has been improved by 33%. Compared with MRA, the average feature-matching accuracy of our algorithm on six datasets has been improved by 6.5%, and the ORB feature-matching algorithm proposed in this thesis has a higher tolerance for image rotation.

**Table 3.** Average matching accuracy of feature points of three algorithms on different datasets.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Bikes</th>
<th>Boat</th>
<th>Graf</th>
<th>Leuven</th>
<th>Trees</th>
<th>Ubc</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORB</td>
<td>58.04</td>
<td>50.95</td>
<td>39.46</td>
<td>63.90</td>
<td>73.64</td>
<td>68.66</td>
</tr>
<tr>
<td>MRA</td>
<td>70.55</td>
<td>64.31</td>
<td>46.45</td>
<td>86.84</td>
<td>90.43</td>
<td>85.56</td>
</tr>
<tr>
<td>Our algorithm</td>
<td>76.52</td>
<td>76.39</td>
<td>47.18</td>
<td>88.90</td>
<td>94.37</td>
<td>90.45</td>
</tr>
</tbody>
</table>

4.3. Localization and Map Construction

In order to verify the performance of the visual-information-based autonomous localization system constructed in this paper, simulation experiments were conducted using partial sequences from the TUM dataset. For the convenience of representation, the algorithm in this paper was named My-SLAM. As the algorithm in this paper was improved on the ORB-SLAM algorithm framework, comparative simulation experiments were conducted using the ORB-SLAM algorithm. The results of the algorithm proposed in this
paper running on the fr1/xyz sequence are shown in Figure 10. The red points in Figure 10a represent the ORB feature points projected onto the map, and the green lines and blue boxes represent the camera’s pose changes and the camera’s position in three-dimensional space, respectively. Figure 10b shows the real-time ORB feature point extraction results of the visual odometer module.

![Figure 10](image_url)

**Figure 10.** The running results of this algorithm on the fr1/xyz sequence of TUM. (a) The change in camera position and posture and its motion track; (b) ORB feature extraction.

In order to evaluate the trajectory accuracy estimated by the algorithm in this paper, a comparative experimental analysis was conducted with ORB-SLAM on the fr1/xyz sequence and fr1/desk sequence, and the results are shown in Figures 11 and 12. From Figures 11 and 12a, it can be intuitively seen that the camera motion trajectory estimated by the algorithm in this paper is closer to the real trajectory than the ORB-SLAM algorithm. For further data analysis, this paper will decompose the camera’s motion from two angles: the translational motion in the main axis direction and the azimuth angle. Firstly, the translational motion in the main axis direction is considered. The experimental results are shown in Figures 11 and 12b. For example, in the fr1/desk sequence, although there is not much difference between the two algorithms in the X-axis and Y-axis directions, the algorithm in this paper has higher accuracy than ORB-SLAM in the Z-axis direction. From the comparison of azimuth angles shown in Figures 11 and 12c, it can be seen that although the motion trajectories of the two algorithms are basically the same at the beginning of a period of time, as the running time increases, the ORB-SLAM algorithm begins to experience trajectory errors. In contrast, the trajectory errors observed in the proposed method are smaller, indicating better stability of the proposed algorithm.
Figure 10. The running results of this algorithm on the fr1/xyz sequence of TUM. (a) The change in camera position and posture and its motion track; (b) ORB feature extraction.

In order to evaluate the trajectory accuracy estimated by the algorithm in this paper, a comparative experimental analysis was conducted with ORB-SLAM on the fr1/xyz sequence and fr1/desk sequence, and the results are shown in Figures 11 and 12. From Figures 11 and 12a, it can be intuitively seen that the camera motion trajectory estimated by the algorithm in this paper is closer to the real trajectory than the ORB-SLAM algorithm. For further data analysis, this paper will decompose the camera’s motion from two angles: the translational motion in the main axis direction and the azimuth angle. Firstly, the translational motion in the main axis direction is considered. The experimental results are shown in Figures 11 and 12b. For example, in the fr1/desk sequence, although there is not much difference between the two algorithms in the X-axis and Y-axis directions, the algorithm in this paper has higher accuracy than ORB-SLAM in the Z-axis direction. From the comparison of azimuth angles shown in Figures 11 and 12c, it can be seen that although the motion trajectories of the two algorithms are basically the same at the beginning of a period of time, as the running time increases, the ORB-SLAM algorithm begins to experience trajectory errors. In contrast, the trajectory errors observed in the proposed method are smaller, indicating better stability of the proposed algorithm.

Figure 11. Experimental results of two algorithms on the fr1/xyz sequence of TUM. (a) Overall track map; (b) track change in spindle direction; (c) change in azimuth.

The algorithms in this paper are evaluated from a quantitative perspective. The APE and RPE of the two algorithms on three dataset sequences are calculated, and then their root-mean-square error (RMSE) is calculated. The experimental results are shown in Tables 4 and 5. Firstly, there is a comparison of the RMSE of the RPE of the two algorithms.
The algorithms in this paper are evaluated from a quantitative perspective. The APE and RPE of the two algorithms on three dataset sequences are calculated, and then their root-mean-square error (RMSE) is calculated. The experimental results are shown in Tables 4 and 5. Firstly, there is a comparison of the RMSE of the RPE of the two algorithms. In the fr1/xyz sequence where the camera only moves slowly along the main axis, the RMSE of our algorithm is not significantly different from that of the original ORB-SLAM algorithm. However, in the fr1/desk series dataset where the camera moves faster and the scene is more complex, the error of our algorithm is significantly smaller than that of the ORB-SLAM algorithm. It can also be seen from the RMSE comparison experiment results of APE of the two algorithms that with the increase in the complexity of dataset scenes and the change in camera attitude, the increase in the RMSE of the algorithm in this paper is smaller than that of ORB-SLAM. In order to further investigate the relationship between the localization errors of the two algorithms in different datasets, the percentage reduction in localization errors of our algorithm compared to ORB-SLAM in different datasets was calculated. After calculation, the average RPE of our algorithm on the three datasets was reduced by 15.4% compared to ORB-SLAM, while the average APE was reduced by 13.9% compared to ORB-SLAM.

Table 4. RMSE comparison of RPE of two algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>fr1/xyz</th>
<th>fr1/Desk2</th>
<th>fr1/Desk</th>
</tr>
</thead>
<tbody>
<tr>
<td>My-SLAM</td>
<td>0.0126 m</td>
<td>0.0582 m</td>
<td>0.0191 m</td>
</tr>
<tr>
<td>ORB-SLAM</td>
<td>0.0139 m</td>
<td>0.0752 m</td>
<td>0.0223 m</td>
</tr>
</tbody>
</table>
Table 5. RMSE comparison of APE of two algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>fr1/xyz</th>
<th>fr1/Desk2</th>
<th>Fr1/Desk</th>
</tr>
</thead>
<tbody>
<tr>
<td>My-SLAM</td>
<td>0.0109 m</td>
<td>0.0439 m</td>
<td>0.0247 m</td>
</tr>
<tr>
<td>ORB-SLAM</td>
<td>0.0121 m</td>
<td>0.0533 m</td>
<td>0.0296 m</td>
</tr>
</tbody>
</table>

5. Discussion and Future Work

In the indoor environment where GPS signals are missing, mobile robots cannot use satellite signals to determine their own positions, which greatly limits the application scenarios for mobile robots. This article proposes a method for indoor positioning of mobile robots based on depth cameras. The localization algorithm is mainly divided into three parts, namely ORB feature extraction, feature matching, and pose estimation.

In the ORB feature extraction section, due to the uneven distribution of feature points extracted by the original ORB algorithm, the dense distribution of feature points in local areas of the image will degrade the accuracy of subsequent pose estimation. In this regard, this paper proposes an improved Quadtree feature extraction algorithm based on a dynamic threshold, which uses the Quadtree structure to divide the image into regions and then calculates the FAST corner comparison threshold in each sub-region to achieve a uniform distribution of ORB feature points. The comparative experimental results are shown in Table 2. For example, in the boat image group, our algorithm improved uniformity by about 17% compared to MRA, indicating that our algorithm has a higher tolerance for image rotation and scaling. Compared with the original ORB algorithm, our algorithm improves consistency by 53%.

In the ORB feature-matching part, the mismatching between feature points will affect the final positioning accuracy. We combined the feature-matching method based on Hamming distance with cosine similarity and conducted a secondary screening of the results after the initial feature matching, and then we used the RANSAC algorithm to eliminate the external points of the feature points without matching from the global perspective. Figure 9c shows the improved feature-matching results. Compared with the original ORB algorithm, our algorithm improves the average feature-matching accuracy by 33% on six datasets. Compared with ORB-SLAM, our algorithm has an average feature-matching accuracy improvement of 6.5% on six datasets, and the ORB feature-matching algorithm proposed in this paper has a higher tolerance for image rotation. All experimental results are shown in Table 3.

The improvement of feature extraction and matching is the main reason for the improved accuracy of the overall positioning algorithm. We compared the algorithm proposed in this paper with ORB-SLAM on three TUM datasets and calculated their average positioning accuracy after multiple experiments. The results are shown in Tables 4 and 5. Compared with ORB-SLAM, our algorithm achieved an average RPE reduction of 15.4% and an average APE reduction of 13.9% on the three datasets.

Through experiments, it was found that although the algorithm in this paper runs relatively smoothly most of the time, its positioning accuracy is poor during intense camera motion, which is a disadvantage of all pure visual SLAM methods. Therefore, in our future work, we will consider integrating inertial sensors, which can adapt to intense movements in a short period of time. Combining the two approaches can effectively improve the accuracy and robustness of the positioning algorithm.

6. Conclusions

This paper presents a depth-camera-based positioning system for the indoor positioning of mobile robots as the application background. The algorithm is mainly divided into four parts, namely the feature extraction module, feature point homogenization module, feature-matching module, and pose estimation module. In the feature extraction module, in order to balance accuracy and real-time performance, ORB feature points are used. In the ORB feature extraction module of the algorithm, in view of the poor uniformity of the
traditional ORB algorithm, an improved ORB feature uniform distribution algorithm based on Quadtree is used for this paper. In the feature-matching module of the algorithm, an improved ORB algorithm based on cosine similarity is proposed to address the problem of a high mismatch rate when ORB matches images with multiple similar regions. With the foundation of the first three steps, the ICP algorithm is used in the pose estimation module to estimate the camera’s pose. The final experimental results indicate that the average positioning accuracy of our algorithm in three indoor scenes on the TUM dataset is 0.027 m, which can meet the needs of autonomous positioning for mobile robots.

Author Contributions: Conceptualization, Z.Y. and W.N.; methodology, H.W.; software, Z.Y. and H.W.; validation, Z.Y.; formal analysis, Z.Y. and H.W.; investigation, Z.Y.; resources, M.Z. and W.N.; data curation, Z.Y.; writing—original draft preparation, Z.Y.; writing—review and editing, Z.Y. and H.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare that they have no conflict of interest regarding the publication of this manuscript.

References


Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.