Visual odometer method based on improved ORB feature

Xiaonan Wang 1,a, Feifei Liu 1,b* and Yaxin Xue1,c

1 School of Electrical Engineering and Automation, Jiangxi University of Science and Technology, Ganzhou, Jiangxi, China

a e-mail: 987858882@qq.com
b* corresponding author’s e-mail: liufeifei@jxust.edu.cn
c e-mail: 1335033575@qq.com

Abstract: The original ORB feature detection algorithm has many problems, such as uneven distribution of feature points, many feature mismatches and poor robustness. A visual odometer method based on improved ORB is proposed. First of all, in order to solve the problem of low extraction efficiency and uneven distribution in the process of key points extraction, an improved quadtree algorithm is proposed which set an appropriate quadtree depth for the image pyramid layer to improve calculation efficiency and filter key points. Secondly, aiming at the problem of mismatching in the feature matching process of adjacent frames, the PROSAC algorithm improved based on RANSAC is used to sort all the sampling points according to the matching quality, and the matching point pairs with good quality are selected to improve the matching accuracy. The experimental results shows that the improved visual odometer method has higher accuracy and less calculation time than the original algorithm.

1. Introduction

Visual SLAM (Simultaneous Localization And Mapping) [1], as an important branch of robotics, has always been a hotspot of research. Visual SLAM is divided into four parts, front-end, back-end, loop detection and mapping. Among them, the front-end visual odometer is mainly used for feature extraction and matching of images, and estimation of camera pose. In the process of image information processing, it is a crucial step to extract and match feature points quickly and efficiently. In this regard, many classic algorithms have been proposed. For example, David Lowe proposed the SIFT Scale-Invariant Feature Transform (SIFT Scale-Invariant Feature Transform) [2], which detects features at different scales of the image and has stability against scale changes; Herbert Bay proposed an accelerated robustness algorithm SURF [3] (Speed-Up Robust Feature) algorithm. On the basis of SIFT, the integral map is effectively used, and filters and image convolutions of different scales are used, which greatly reduces the calculation consumption and improves the robustness and calculation speed.

The above two algorithms have high accuracy but poor real-time performance. To meet the high real-time requirements of SLAM, the ORB (Oriented BRIEF (Binary Robust Independent Elementary Features)) feature detection algorithm proposed by Ethan Rublee et al. [4]. It uses the FAST operator to quickly extract features, and the BRIEF two-dimensional descriptor for feature description, greatly reduce the amount of calculation. The ORB feature detection algorithm also sacrifices a certain accuracy while accelerating the detection speed, and the ORB feature points are easily concentrated in the texture-rich area, which has a great impact on the matching of points between adjacent frames.
In view of the uneven distribution of feature points and many mismatches in the original ORB visual odometer, this paper proposes a visual odometer based on the improved ORB.

2. Improved ORB feature detection algorithm

2.1. FAST corner detection
ORB adopts FAST algorithm for corner detection, and uses the feature of fast corner detection by FAST to improve the speed of the entire algorithm. The FAST corner detection algorithm is very sensitive to image scale and rotation, so the original ORB algorithm improves robustness by constructing multi-scale image pyramids and calculating feature main directions. The principle of FAST corner detection algorithm is shown in Figure 1.

![Figure 1: Schematic diagram of FAST corner detection](image)

The specific operations are as follows:
1) Randomly select a pixel $P$ from the constructed multi-scale image pyramid layer, and its gray value is $I_{P}$;
2) Take the $P$ as the center and a radius of 3 to generate a circle with 16 pixels;
3) Set an appropriate initial gray threshold $t$;
4) Perform detection on the circle. If the absolute value of the gray difference between more than 12 pixels and the pixel $P$ is greater than the initial threshold $t$, the point $P$ is determined to be a feature point; otherwise, it is discarded, and other pixels are reselected, and repeat the above steps.

The original ORB algorithm is more sensitive to rotation. To solve this problem, an improvement has been made. The gray value heart method is used to add a main direction to the detected feature points:

a) In image block $B$, the moments of the image block are defined as:

$$m_{pq} = \sum_{x,y \in B} x^p y^q I(x,y), \quad p, q = \{0, 1\} \tag{1}$$

$I(x,y)$ is the gray value of the image. The first and zero order moments of the image:

$$\begin{align*}
m_{00} &= \sum_{x,y} I(x,y) \\
m_{10} &= \sum_{x,y} xI(x,y) \\
m_{01} &= \sum_{x,y} yI(x,y) \tag{2}
\end{align*}$$

b) Find the centroid of the image block by the moment:

$$C = \left( \frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \tag{3}$$

c) Construct a direction vector from the feature point $P$ to the intensity center $C$. The main direction of the feature point can be defined as:

$$\theta = \arctan \left( \frac{m_{01}}{m_{10}} \right) \tag{4}$$
After obtaining the feature points, use the BRIEF description algorithm to describe the feature points in two dimensions. The BRIEF descriptor uses binary calculations. Compared with the floating-point calculations used by algorithms such as SIFT, it has faster calculation speed and higher real-time performance. BRIEF uses Gaussian distribution sampling to randomly select A group of point pairs in the image blocks around the feature points. In general, A=256. Binarize the A group of point pairs, and then compare, store each comparison result bit by bit to form a 256-bit binary coded descriptor. Calculate the Hamming distance based on the descriptor and perform feature matching.

2.2. Improved quadtree algorithm

The initial threshold of the traditional quadtree algorithm is manually set, which is not applicable to each area in the division. Therefore, it is improved by setting an adaptive threshold to replace the initial artificial threshold, reducing the amount of calculation and enhancing the robustness of the algorithm. The formula is as follows:

$$\text{iniT} = \frac{\sum_{i=1}^{n} (I(x_i) - \bar{I}(x))^2}{n \bar{I}(x)}$$

In the formula, $I(x_i)$ represents the gray value of the pixel, $\bar{I}(x)$ represents the average gray value, and iniT is the initial threshold of the algorithm.

By constructing a multi-scale image pyramid, generally 8 layers are constructed, and the expected feature points are calculated according to the scale factor. Denote the total number of feature points as $m$ and the scale factor as $s$, then the number of features in the first layer is:

$$m = a + \frac{1}{s^2}a + \frac{1}{s^4}a + \frac{1}{s^6}a + \frac{1}{s^4}a + \frac{1}{s^2}a + 1$$

By dividing the image into a grid, the corner points in the entire image are evenly distributed, and the grid is initialized to a square of 30×30 pixels. Then calculate the number of rows $L$ and the number of columns $R$ according to the resolution. Take the width as an example, record the width of the image as width, then the number of divided columns $R$ is:

$$R = \frac{\text{width}}{30}$$

Find the width $w$ of the actual grid by the number of columns:

$$w = \text{round} \left( \frac{\text{width}}{R} \right)$$

After the grid is divided, the corner points will be extracted as the initial extraction threshold.

In the image pyramid, the corner points obtained are filtered through the quadtree. Take the entire area of the image as a node, divide the area into four quadrants, count the number of corner points detected in each quadrant, if the number of corner points is greater than 1, then use the area as a node and continue to split into four areas for detection. Until the number of corner points in the area is 1 or the number of feature points meets expectations, the quadtree split is completed.

In different image layers of the image pyramid, adaptively set unequal expected number of feature points. Let $D_{\text{max}}$ be the maximum depth of the pyramid, and $\text{Num}_j$ is the number of storage nodes:

$$4^{D_{\text{max}}} \geq \text{Num}_j$$

The feature point distribution before and after the improved quadtree homogenization is shown in Figure 2:
2.3. PROSAC matching algorithm

The fast and uniform extraction of feature points lays a good foundation for feature point pair matching. Feature matching is based on the Hamming distance of similar feature points between adjacent frames. In the matching process, under-matching or mismatching often occurs, which has a great impact on the pose estimation. Traditional methods to remove mismatches include RANSAC, relaxation iteration, minimum median method, and parallax-based filtering algorithms, but they are time-consuming and insufficient in accuracy. This paper uses the improved PROSAC algorithm based on RANSAC for feature matching.

The PROSAC algorithm is improved on the basis of the RANSAC algorithm. The RANSAC algorithm randomly selects samples without considering the differences between samples, that is, the degree of excellence. In actual situations, the probability of points within the sample varies from high to low. PROSAC assumes that the number of correct point sets in the sampling set is greater than the number of wrong point sets, that is, the number of interior points is greater than the number of exterior points. The PROSAC algorithm is as follows:

Find the ratio $R$ of the closest and next closest Hamming distance of the matching pair of feature points:

$$R = \frac{D(V_p, V_{aq})}{D(V_p, V_{mq})}$$

(10)

Among them, $V_p$ is the feature vector of the feature point $p$; $V_{aq}$ is the feature vector of the closest point $q$ in a picture; $V_{mq}$ is the feature vector of the next adjacent feature point $q$ in a picture; $D$ is the distance between the vectors.

The Hamming distance between the matching point and the non-matching point is obviously different, so 0.7 times the maximum Hamming distance is set as the threshold. When $R$ is less than the threshold $T$, the sampling point sets are arranged according to the matching quality from good to bad. The mass function is $M$, as shown in formula (11):

$$R_i > R_j \Rightarrow M(i) > M(j)$$

(11)

The matching steps of PROSAC algorithm are as follows:

1. Set the initial value of the iteration and the maximum number of iterations.
2. Determine whether the number of iterations is greater than the maximum number of iterations. If it is, no suitable mathematical model is found and a prompt is given; otherwise, proceed to the next step.
3. Arrange the set of sampling points in descending order of matching quality $M$, and select $n$ data with higher quality.
4. Randomly select $K$ from $n$ data, calculate the model parameters, and calculate the number of data whose error is less than the threshold of the interior point error obtained by using this model parameter, and determine whether the number of data is greater than the set threshold, if so, return the interior point and Model parameters; if not, add 1 to the number of iterations and return to step 2.
Since the improved quadtree algorithm has a good limit on the number of feature points, this will greatly reduce the number of iterations in the PROSAC matching stage, thereby reducing time-consuming.

3. Analysis of experimental results
This experiment is carried out on a personal PC, the programming environment is Ubuntu 16.04 (Intel(R) Core(TM) i5-9300h CPU @ 2.4GHz), 8 G memory, and OpenCV version 3.4.

3.1. Feature matching experiment
The improved algorithm in this paper is compared with SURF and the original ORB algorithm for feature matching experiments. As can be seen from Table 1. Although SURF has high matching accuracy, it takes too much time to calculate, which is not suitable for high real-time systems such as SLAM. The original ORB has a short calculation time, but there are many mismatches, and the correct matching rate is low. Improved ORB has a good performance in matching accuracy and calculation time. Compared with the original algorithm, the feature point distribution after the improved quadtree homogenization is more uniform, which reduces the matching error and matching time, and the PROSAC algorithm is used to increase the matching accuracy. In general, the accuracy of the improved algorithm is increased by 10%, and the time required is reduced by 20%.

| Algorithm   | Match point pair/error point pair | Extraction time/s | Matching time/s | Total time/s | Correct matching rate/% |
|-------------|----------------------------------|-------------------|----------------|--------------|-------------------------|
| SURF        | 54/3                             | 0.47              | 0.19           | 0.66         | 94.4                    |
| ORB         | 79/18                            | 0.24              | 0.16           | 0.40         | 77.2                    |
| Improved    | 67/7                             | 0.27              | 0.05           | 0.32         | 89.6                    |

3.2. Visual odometer experiment
This experiment selects two video sequences in TUM data for experiment, and compares the visual odometer based on the improved algorithm with the original ORB visual odometer. The experimental results are evaluated by calculating the absolute error.

Figure 3 Absolute error comparison
Figure 3 shows a bar graph showing the absolute error of the improved ORB visual odometer and the original ORB visual odometer, as well as the standard deviation, mean, maximum, and minimum of the absolute error in fr1_xyz and fr2_xyz video sequences. The blue bar graph in the figure represents the improved algorithm, and the green bar graph represents the original ORB-SLAM2. On the whole, the error of the improved algorithm is smaller than that of the original algorithm.
Figure 4 shows the visualized trajectories of the estimated and true values of the improved algorithm and the original ORB algorithm in the fr1_xyz and fr2_xyz video sequences in the x and y-axis planes. The dotted line represents the real trajectory, the blue line represents the original ORB algorithm movement trajectory, and the green line represents the movement trajectory of the improved algorithm in this chapter. It can be seen from Figure 5 that the movement trajectory of the improved algorithm in this chapter is basically the same as the real trajectory.

4. Conclusions
Aiming at the problems of uneven distribution of the original ORB algorithm, obvious aggregation phenomenon, and many mismatches, a visual odometer based on improved ORB features is proposed. By designing the desired number of feature points required for different pyramid levels and setting the depth of the quadtree for each level, the uneven distribution of feature points is solved; the PROSAC algorithm is used to improve the matching accuracy of feature points. Finally, multiple sets of TUM data sets are selected for experimental verification. The experimental results of feature matching show that the algorithm is significantly improved compared to the traditional ORB algorithm. Since adaptive depth detection is added to different pyramid layers, the calculation of redundant feature points is reduced, and the feature extraction time is reduced by more than 20% on average. Due to the use of the PROSAC algorithm, the matching accuracy rate is increased by more than 10%. The visual odometer experiment shows that the absolute error of the improved algorithm is smaller than that of the original algorithm, it has higher robustness, and can be well applied in the SLAM algorithm.

Acknowledgments
Foundation items: National Natural Science Foundation of China (61364014); Key Laboratory of Machine Vision and Intelligent System, No.20181BCD40009;

References
[1] Jianjun Gui, Dongbing Gu, Sen Wang, Huosheng Hu. A review of visual inertial odometry from filtering and optimisation perspectives [J]. Advanced Robotics, 2015, 29(20).
[2] Lowe D G. Distinctive image features from scale-invariant keypoints [J]. International Journal of Computer Vision, 2004, 60(2): 91−110.
[3] Bay H., Tuytelaars T., Gool L V. SURF: Speeded up robust feature [C]. Proceedings of European Conference on Computer Vision, 2006, 1: 404−417.
[4] Rublee E., Rabaud V., Konolige K., et al. ORB: An efficient alternative to SIFT or SURF [C]. Proceedings of IEEE International Conference on Computer Vision, Washington, USA, 2011: 2564−2571.
[5] Mur-Artal R., Tardos J D. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras [J]. IEEE Transactions on Robotics, 2016, 33(5):1255-1262.