Front-End Optimization of ORB-SLAM3 Using Adaptive Thresholding and PROSAC

Qian Tang
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, China
E-mail: tangqian@st.xatu.edu.cn

Shifeng Zhao
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, China
E-mail: zhaoshifeng@xatu.edu.cn

Xiaojun Bai
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, China
E-mail: baixiaojun@xatu.edu.cn

Liuhua Di
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, China
E-mail: di.liuhua@technopro.com.cn

Yanfang Fu
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, China
E-mail: fuyanfang@xatu.edu.cn

***Abstract*—The performance of visual SLAM is strongly influenced by the quality of front-end feature detection and correspondence matching. To improve ORB-SLAM3 in weak-texture environments, under feature clustering, and in the presence of mismatches, this paper optimizes the front-end pipeline in three stages. First, an adaptive threshold is introduced into FAST detection to improve keypoint extraction in weak-texture regions. Second, an improved quadtree-based distribution strategy is adopted to reduce feature over-concentration in strongly textured areas while retaining more valid keypoints in weak-texture regions. Third, PROSAC is used for correspondence verification to remove mismatches with lower iterative cost than conventional random sampling. The improved front-end is integrated into ORB-SLAM3 and evaluated on the EuRoC Vicon Room1 03 sequence. Experimental results show clear gains in feature extraction and matching quality, reducing the mean Absolute Trajectory Error (ATE) by 81.8% and the mean Relative Pose Error (RPE) by 89.4% relative to the baseline, thereby improving localization and mapping accuracy.**

Keywords-Visual SLAM; Feature Extraction and Matching; Quad-Tree Coding; PROSAC Algorithm

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) is a core technology for autonomous robot navigation. It enables a robot to perceive its surroundings through on board sensors, estimate its own pose, and build a map of the environment at the same time [1]. With the development of computer vision, visual SLAM [2] has become an important research direction because cameras are low-cost sensors and can provide abundant scene information. Among visual SLAM methods, the ORB-SLAM series [3] offers strong real-time performance and practical engineering value, but its performance can still degrade under different scene conditions. In general, the system detects and describes image features from consecutive frames, estimates inter-frame motion through feature matching, and then combines mapping and loop-closure modules to realize real-time localization and mapping.

Within the ORB-SLAM algorithmic framework, feature point detection, extraction, and precise matching between adjacent frames are critical steps for ensuring positioning accuracy. However, the ORB-SLAM series faces challenges such as uneven feature point distribution across areas with differing textures—over concentration in highly textured regions and excessive sparsity in weakly textured areas — alongside mismatching issues caused by data noise [4]. These factors collectively impact positioning precision.

ORB-SLAM3, as the latest iteration in the ORB-SLAM series, employs the ORB (Oriented FAST and Rotated BRIEF) [5] algorithm for feature point extraction. ORB is a fast feature extraction and description algorithm that combines FAST [6] with the Binary Robust Independent Elementary Feature (BRIEF) [7], achieving exceptionally high matching speeds. Although this algorithm effectively addresses issues present in the ORB algorithm within the ORB-SLAM series — such as rotational variations, poor scale in variance, low first-match accuracy, and weak interference resistance—it still exhibits limitations in the feature point detection phase. The threshold in the feature response function remains a fixed value [8]. When detecting areas with weak texture, the number of detected feature points significantly decreases, resulting in poor performance adaptability under such conditions. Reference [9] employs an adaptive thresholding method to enhance feature detection in weakly textured regions. However, this approach calculates thresholds based on global grayscale information, increasing computational overhead. In strongly textured areas, the ORB algorithm may cause feature clustering. Reference [10] introduced a quadtree uniformization algorithm to address feature point clustering, but traditional quadtree algorithms retain too few feature points in weakly textured regions. For matching, ORB-SLAM3 relies on either brute-force matching or the FLANN algorithm, both of which produce some degree of mismatches, thereby reducing localization accuracy. Reference [11] employs the RANSAC (Randomized Algorithms for Normalizing and Selecting) algorithm to eliminate mismatches. However, this algorithm relies on fully random global sampling, requiring excessive iterations and increasing computational overhead.

To address the above limitations, this paper improves the ORB-SLAM3 front-end in detection, distribution, and matching verification. A local adaptive-threshold FAST strategy is designed to increase keypoint detectability in weak-texture areas while avoiding the overhead of global-gray statistics. An improved quadtree-based selection scheme is then applied to obtain a more balanced keypoint distribution by preserving more points in weak-texture regions and suppressing redundancy in highly textured regions. Finally, a PROSAC-based mismatch rejection module is introduced to prioritize high-quality correspondences during model estimation, thereby reducing iteration cost and improving matching reliability.

This paper focuses on optimizing the feature point extraction and matching stages within the ORB-SLAM3 algorithm's front-end. The main contributions of this paper include:

- Adaptive FAST thresholding based on local gray statistics is introduced to improve keypoint detection in weak-texture regions, increasing feature availability for subsequent tracking while keeping the computational burden controllable.
- A texture-aware quadtree feature distribution strategy is proposed. By adjusting the node stopping and retention mechanism, the method preserves more informative keypoints in weak-texture areas and reduces redundancy in strongly textured regions, producing a more balanced global distribution.
- A PROSAC-based [13] mismatch filtering scheme is integrated into the matching stage to exploit correspondence quality ranking during sampling, which improves outlier rejection efficiency and reduces iterative computation compared with random sampling strategies.

II. FEATURE DETECTION, EXTRACTION, AND MATCHING IN ORB-SLAM3

ORB-SLAM3 is the latest version of ORB-SLAM, featuring optimizations for maximum a posteriori estimation, multi-map systems, and novel position recognition methods [13]. The ORB-SLAM3 front-end workflow is illustrated in Fig.1.

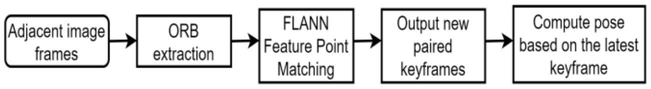

Adjacent image frames
ORB extraction
FLANN Feature Point Matching
Output new paired keyframes
Compute pose based on the latest keyframe

Figure 1. Feature Detection and Matching Stream in the ORB-SLAM3 Algorithm

A. Feature Point Extraction Algorithm

ORB-SLAM3 performs feature point detection and feature extraction using the ORB algorithm. First, an image pyramid is constructed to maintain image scale in variance. Next, the gray-level centroid method is employed to ensure image rotation in variance. Then, the FAST algorithm detects feature points. Finally, the BRIEF algorithm describes the features. These four steps complete the feature point extraction for the image. The feature point extraction algorithm flow is shown in Fig.2:

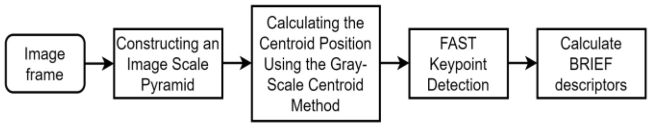


Image frame
Constructing an Image Scale Pyramid
Calculating the Centroid Position Using the Gray-Scale Centroid Method
FAST Keypoint Detection
Calculate BRIEF descriptors

Figure 2. Feature Point Extraction Process

FAST feature point detection is a critical step in the ORB feature extraction process. The FAST algorithm determines whether a point qualifies as a feature point by evaluating the gray-value differences between the candidate feature point and its surrounding pixels. If a consecutive sequence of surrounding pixels exhibits gray-value differences exceeding a preset threshold relative to the candidate feature point, that point is classified as a feature point. A Bresenham circle with a radius of 3 pixels (i.e., a circle with a circumference of 16 pixels) is used to determine whether its center pixel is a feature point. Pixels on the Bresenham circle are numbered sequentially. A schematic diagram of FAST feature point detection is shown in Fig.3.

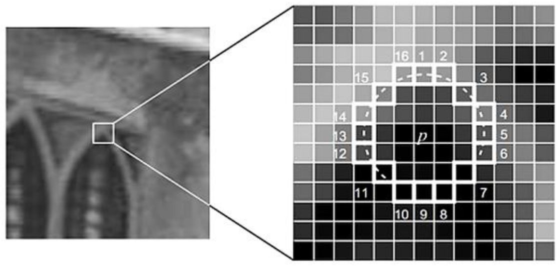


Figure 3. FATS Feature Detection Schematic

The specific process of the FAST algorithm is as follows:

a) First, select the pixel point p with brightness l(p)in the image.

b) Set a fixed threshold T, where T is taken as 0.2l(p).

c) On a circle centered at the candidate feature point p with a radius of 3, select 16 pixels, denoted as l(i).

d) If there are N consecutive points on the circle with brightness values between lp－T and lp+T, then point p is a feature point for that region. N is typically set to 12 (FAST12). The feature point response function is as follows:

$$V(i)=\begin{cases}1 & if\ \left|l(i)-l(p)\right|>T \quad 1\le i\le 16\\ 0 & else\end{cases} \tag{1}$$

$$N=\sum_{i=1}^{16}V(i) \tag{2}$$

When N⩾ 12, the point is considered a valid feature point.

e) Perform steps *a)* to *d)* on all pixels in the image to filter out all feature points.

B. Feature point matching

After extracting feature points, the next step involves matching them across adjacent keyframes. The paired feature points provide prior information for transforming the pose relationship between subsequent frames. ORB-SLAM3 employs the FLANN (Fastest Local Nearest Neighbor) matching algorithm [14]. The FLANN algorithm achieves rapid nearest neighbor search by constructing efficient data structures. Its principle involves searching the neighborhood around the location of the feature point to be matched, identifying the point with the smallest Euclidean distance to it. The smallest Euclidean distance indicates the closest distance between the matching feature point pairs, meaning this pair is considered the most similar, while the largest distance indicates the least similarity. While the Fastest Nearest Neighbor matching algorithm significantly enhances feature point matching efficiency, it may generate mismatches, leading to reduced accuracy and consequently affecting positioning precision.

C. Position and Orientation Estimation

This paper employs 3D-2D pose estimation using a stereo camera [15]. By leveraging the parallax between identical frames from the left and right cameras, depth information for spatial points can be determined. Subsequently, after obtaining feature point matching pairs across different frames via the ORB feature matching algorithm, the camera pose can be estimated using the PnP (Perspective-n-Point) algorithm based on the spatial point's 3D information and its projected position.

In PnP, n represents the number of points whose pixel positions require estimation. Common approaches for resolving projection errors include P3P, EPnP (Efficient Perspective-n-Point), and direct linear transformation. ORB-SLAM3 employs nonlinear optimization to formulate a least-squares problem, solving the camera pose by minimizing reprojection errors — a technique known as bundle adjustment (BA).

Given two consecutive image frames containing n pairs of matched feature points, the camera pose between the two images is defined as rotation R and translation t. Consider a point P in 3D space. Its projections in the two image frames are denoted as p1 and p2, respectively. The point p1 is reprojected into the subsequent frame via R and t, yielding $\hat{p}_1$. This reprojected point $\hat{p}_2$ does not perfectly coincide with the matched point p2, and the difference between $\hat{p}_2$ and p2 is denoted as e, as illustrated in Fig.4.

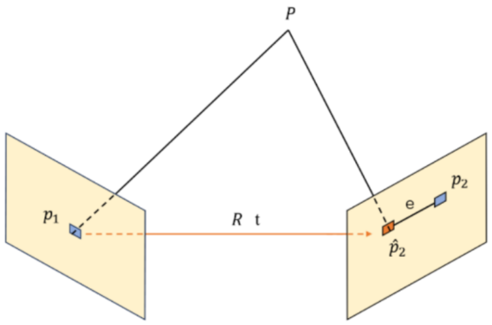


Figure 4. Schematic of Reprojection Error

Suppose there are i points $p_i = [X_i, Y_i, Z_i]^T$ and their projections $u_i = [u_i, v_i]^T$ in three-dimensional space. Using these to solve for the camera's R and t, the camera pose is represented by the Lie algebra ξ. The relationship between spatial points and pixels is expressed as:

$$S_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = K exp(\xi^n) \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \tag{3}$$

Matrix-forming the above equation yields:

$$s_i u_i = K exp(\xi^n) P_i \tag{4}$$

Due to the difference between the camera observation values and the projected positions, the above equation contains an error term, namely the reprojection error. Therefore, a least squares problem is formulated to minimize this error:

$$\xi^* = \left\| \arg min_\xi \frac{1}{2} \sum_{i=1-1}^{n} u_i - \frac{1}{s_i} K exp(\xi^n) P_i \right\|_2^2 \tag{5}$$

By incorporating the properties of Lie algebras, the above equation is formulated as an unconstrained problem. Optimization methods such as the Gauss-Newton (GN) method and the Levenberg-Marquardt (LM) method are employed to solve for the optimal solution. At this stage, the optimization derivatives for the error term are calculated:

$$e(x + \Delta x) \approx e(x) + J^T \Delta x \tag{6}$$

In equation (6), x represents the camera pose, Δx denotes the projection pixel coordinate error, and JT is the Jacobian matrix of the error term e(x)with respect to x. At this point, the Jacobian matrix JT is solved:

$$\frac{\partial e}{\partial \delta \xi} = -\begin{bmatrix} \frac{f_x}{Z'} & 0 & -\frac{f_x X'}{Z'^2} & -\frac{f_x X' Y'}{Z'^2} & f_x + \frac{f_x X^2}{Z'^2} & -\frac{f_x Y'}{Z'} \\ 0 & \frac{f_x}{Z'} & -\frac{f_x Y'}{Z'^2} & -f_y - \frac{f_y Y'^2}{Z'^2} & \frac{f_y X' Y'}{Z'^2} & \frac{f_y Y'}{Z'} \end{bmatrix} \quad (7)$$

Equation (7) represents the first-order change relationship of the Li algebra ξ for the camera's motion pose. Its solution corresponds to the change value of the relative pose between two image frames.

III. Algorithm Improvements

The improved feature point extraction and matching algorithm described in this paper is illustrated in Fig.5. First, after acquiring an image frame, the adaptive thresholding FAST algorithm is employed to detect feature points. These points are then normalized using an improved quad-tree algorithm. Subsequently, feature point matching is performed between two image frames. The PROSAC algorithm is applied to filter out mismatched points. Finally, valid candidate frames are selected to serve as prior information for subsequent localization. The algorithm flow is illustrated in Fig.5:

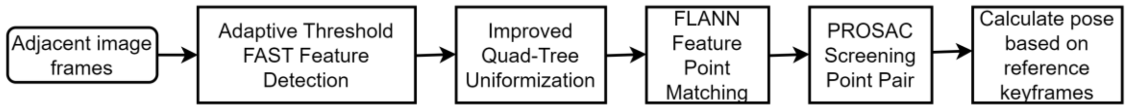

Adjacent image frames
Adaptive Threshold FAST Feature Detection
Improved Quad-Tree Uniformization
FLANN Feature Point Matching
PROSAC Screening Point Pair
Calculate pose based on reference keyframes

Figure 5. Feature Point Extraction and Matching Algorithm Workflow for This Paper .

A. Adaptive Threshold-Based FAST Algorithm

In the traditional ORB algorithm's feature point detection response function, the threshold T is a pre-set fixed value. The difference between pixels in weakly textured regions and their surrounding pixels often fails to reach the set threshold T, making it difficult to detect feature points. This hinders subsequent feature-based tracking tasks, ultimately reducing the robot's positioning accuracy. Therefore, this paper proposes an adaptive threshold FAST feature point detection method. This method adjusts the threshold T in step (b) of Section ⅡA based on texture variations, enabling the extraction of more feature points in weakly textured regions. The remaining steps remain consistent with Section ⅡA. The adaptive threshold T calculation process is as follows:

a) The calculation method for determining the average gray value $\overline{f(Q)}$ of all pixels within the Bresenham circle centered at the feature point p with a radius of 3, along with the 16 pixels on the circle, is as follows:

$$\overline{f(Q)} = \frac{\sum_{i=1}^{n} I(x,y)}{\frac{1}{2}\pi r^2} \quad (8)$$

Here, n represents the sum of pixels on and inside the circle, and r is set to 3.

b) Calculate the variance S between the grayscale values f(Q) of m pixels on the circle and the average grayscale value $\overline{f(Q)}$:

$$S = \frac{1}{m}\sum_{i=1}^{m}(f(Q) - \overline{f(Q)})^2 \quad (9)$$

c) Set another scaling factor a, where a typically ranges from 0 to 1. Then the adaptive threshold T is defined as:

$$T = \alpha(\frac{1}{m}\sum_{i=1}^{m}(f(Q) - \overline{f(Q)})^2) \quad (10)$$

B. Improved Quad-Tree Algorithm

After extracting feature points using improved algorithms, the issue of excessive concentration and overlap of feature points persists. The quad-tree algorithm is commonly employed to address feature point clustering [16]. This algorithm first sets the total number of tree nodes N, where each node represents a partitioned image region. It then traverses the image, dividing it into 4, 16, ... nodes until the total number reaches N. Each node is assigned a response value (typically 1). When the number of feature points within a node falls below this response value, the node stops splitting. The highest-quality feature point is then selected from each final node as the result of the uniformization process. The quad-tree algorithm workflow is illustrated in Fig.6, where: (a) shows the initial feature point distribution map; (b) displays the final feature point distribution image after splitting, with elliptical and circular regions indicating strong texture areas and rectangular regions representing weak texture areas; (c) presents the result after feature point domain filtering; (d) depicts the final result image.

Traditional quadtrees can reduce information redundancy by removing some feature points when handling densely clustered points in highly textured regions. However, they also mistakenly delete feature points in weakly textured areas, leading to insufficient feature point coverage.

To mitigate the above problem, this paper modifies the quadtree stopping and retention strategy instead of directly using the conventional node-selection rule. Specifically, when the feature response in a node is lower than a preset threshold, the texture strength of that region is further evaluated. If the region is classified as weak-texture, node splitting is terminated earlier. During the final feature selection stage, all keypoints inside weak-texture nodes are preserved, whereas strongly textured nodes still keep only representative high-quality keypoints. This design improves feature coverage in weak-texture regions and enhances the overall balance of the final distribution. The detailed procedure is described below:

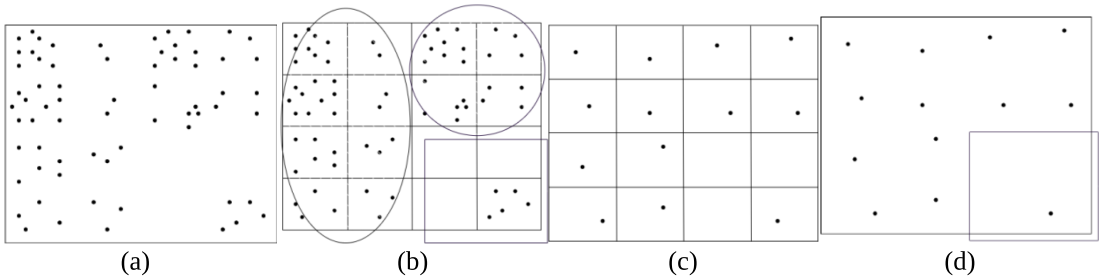


Figure 6. Quadtree Flowchart.（a）Initial Feature Point Distribution,(b)Feature point distribution after splitting,(c)Distributed Features after Screening,(d)Final Feature Point Distributio

a) Partition the image into quadtree nodes. For a node whose feature count is lower than the node response threshold, compute the mean grayscale value μ of that region and then calculate the grayscale variance with respect to μ. If the variance is smaller than the preset threshold Q (taken as the median of σ2), the node is regarded as a weak-texture region. The specific formula is as follows:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} I_i(x_i, y_i) \qquad (11)$$

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N} (I_i - \mu)^2 \qquad (12)$$

Among them, N is the number of feature points in each region.

b) When σ2<Q, the node region ceases division, while the remaining node regions continue to be divided.

c) When the number of nodes reaches the preset value and partitioning stops, all feature points are retained in weakly textured regions, while one high-quality feature point is retained in strongly textured regions.

Fig.7 illustrates the results of the improved quadtree strategy: subfigure (a) shows the initial partition result, (b) shows the final partition, and (c) shows the final screened feature distribution. Compared with the conventional strategy, the improved method retains more keypoints in weak-texture regions while maintaining a relatively uniform feature layout in strongly textured regions.

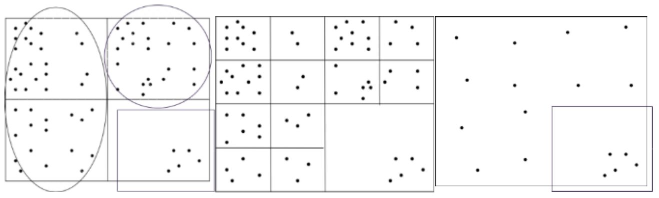


Figure 7. Improved Quad-Tree Flowchart. (a)After the feature points are segmented once, (b)Final Feature Distribution, (c)Final Screening Results

C. PROSAC Mismatch Removal Algorithm

After feature point matching, some mismatches may occur. Currently, mainstream visual SLAM systems primarily employ the RANSAC (Random Sample Consensus) algorithm to address mismatching issues.

RANSAC estimates a model through repeated random sampling and is widely used for outlier suppression. In a typical 2D line-fitting example, two points are randomly selected to form a candidate line. The distances from the remaining points to this line are then computed. Points whose distances are below a preset threshold are treated as inliers, while the others are treated as outliers. By iteratively repeating this process and updating the model according to the inlier set, RANSAC eventually obtains the model with the strongest support. Figure 8 shows the sampling and model-update process, where the red points denote retained inliers and the blue points denote rejected outliers. The iterative model progression is illustrated in Fig.8: (a) shows the initial iteration model, while (b) depicts the optimal iterative model. The red dots represent retained inliers, and the blue dots denote discarded outliers.

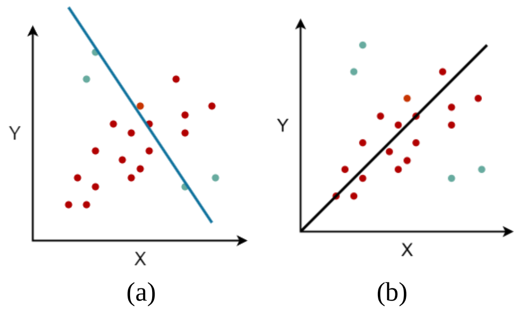


Figure 8. RANSAC Random Sampling Process. (a)Model after one iteration, (b)Optimal Iterative Model

The RANSAC algorithm in the image matching phase of ORB-SLAM3 can eliminate mismatched outliers while retaining correctly matched inliers. Its algorithmic steps are detailed in Table 1.

TABLE I. RANSAC ALGORITHM STEPS

Input:Maximum iteration count Im, error threshold δ for determining inliers, threshold T for the number of inliers Output:homography matrix H
(1) Randomly select four pairs of matching points and compute the homography matrix H; (2) After removing the aforementioned four pairs of points, the remaining points are calculated based on the homography matrix H to determine their corresponding projection points. (3) Calculate the error e between other points and the projection point, then compare it with the error threshold δ. If e<δ, the point is an inlier; otherwise, it is an outlier. (4) Compute the number of inliers t and compare it with the preset inlier threshold T. If t > T, update the inlier count to t; otherwise, increment the iteration count by 1 and repeat the procedure. (5) After updating the number of inliers t, recalculate the single-response matrix H and obtain new inliers. (6) If the number of iterations is less than Im, return the homography matrix H; otherwise, the model does not satisfy the conditions.

The maximum iteration count is calculated as:

$$I_m = \frac{\ln(1-p)}{\ln(1-\varepsilon^w)} \tag{13}$$

In the formula, ε denotes the number of interior points divided by the total number of sample points, representing the proportion of interior points ε^w. represents the probability of matching all t points correctly, where P is the confidence probability. Based on the above, the maximum iteration count can be calculated.

The sample points selected in the RANSAC algorithm are random, requiring continuous iteration to select as many interior points as possible. Consequently, the overall computation time increases, and the number of iterations becomes unstable.

To improve mismatch rejection efficiency, this paper adopts PROSAC (Progressive Sample Consensus), which can be regarded as a confidence-guided extension of RANSAC. Instead of sampling uniformly from the entire correspondence set, PROSAC first ranks candidate matches according to matching quality and then progressively expands the sampling pool during model estimation. By prioritizing high-confidence correspondences in early iterations, the algorithm increases the probability of generating reliable hypotheses and reduces unnecessary iterations. The corresponding sampling process is illustrated in Fig.9, where (a) shows the model after one iteration and (b) presents the optimal iterative model.

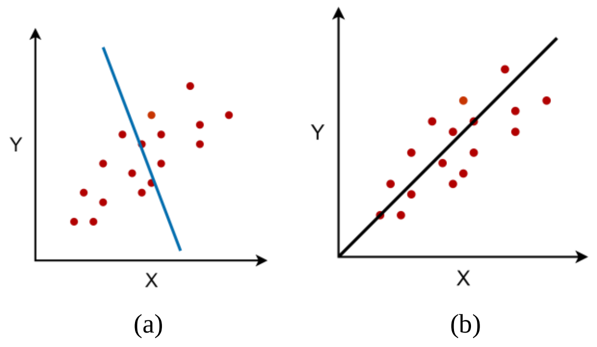


Figure 9. PROSAC Algorithm Sampling Process. (a)Model after one iteration, (b)Optimal Iterative Model

In the proposed correspondence verification stage, PROSAC is driven by a confidence-based ranking of candidate matches. For each correspondence, a distance ratio β is computed from the nearest and second-nearest descriptor distances:

$$\beta = \frac{d_{min}}{d_{min2}} \tag{14}$$

Where dmin is the minimum Euclidean distance and dmin2 is the second-smallest Euclidean distance. A smaller β indicates lower ambiguity and generally corresponds to a more reliable match. Based on this ratio, a quality factor γ is further defined to score and rank correspondences. The ranked correspondences are then used by PROSAC to guide hypothesis generation, which helps reduce iteration count and computational cost while maintaining effective mismatch rejection. A quality factor can be introduced to measure matching quality, defined as:

$$\gamma = \frac{1}{\beta d_{min}} \tag{15}$$

By introducing the quality score into correspondence ranking, the sampling process can prioritize more reliable matches, which helps improve verification efficiency and decrease iteration overhead. The implementation workflow is listed in Table II.

TABLE II. PROSAC ALGORITHM STEPS

Input: Maximum iteration count Im, error threshold δ for determining interior points, threshold T for the number of inliers Output: Homography matrix H
(1) Compute the minimum Euclidean distance dmin between feature points and determine the Euclidean distance ratio β. (2) Calculate the quality factor and measure the quality of the matching points. (3) Sort in descending order by matching quality. Select m points and group them into sets of four. Calculate the sum of quality for each group and sort the groups based on this sum. (4) From the sorted matching pairs, select the four highest-quality matching points and compute the homography matrix H; (5) After removing the aforementioned four pairs of points, the remaining points are calculated based on the homography matrix H to determine their corresponding projection points. (6) Calculate the error e between other points and the projection point, then compare it with the error threshold δ. If e<δ, the point is an interior point; otherwise, it is an outlier. (7) Compute the number of inliers t and compare it with the preset inlier threshold T. If t > T, update the inlier count to t; otherwise, increment the iteration count by 1 and repeat the procedure. (8) After updating the number of inliers t, recalculate the homography matrix H and obtain new inliers. If the number of iterations is less than Im, return the homography matrix H; otherwise, the model does not satisfy the conditions.

IV. Experimental Results and Analysis

A. Experimental Environment

Experiments were conducted on Ubuntu 20.04 using the EuRoC dataset (ETH Zurich) in stereo mode. The Vicon Room1 03 sequence was selected for evaluation, with a duration of 104.655 s and a trajectory length of 78.982 m. To assess localization performance, two commonly used trajectory metrics are employed: Absolute Trajectory Error (ATE) and Relative Pose Error (RPE). ATE evaluates global trajectory agreement between the estimated trajectory and the ground truth, whereas RPE reflects local pose drift by comparing relative motion over the same time interval.

1) Absolute Trajectory Error (ATE): This metric is obtained by comparing the SLAM-estimated trajectory with the reference trajectory provided by the datasets, and it reflects overall trajectory consistency.

2) Relative Pose Error (RPE): This metric measures the difference between relative pose changes in the estimated trajectory and the ground truth over a specified interval, and it is used to characterize local drift behavior.

B. Controlled trial

1) Feature Point Detection Experiment: Two consecutive frames of drone motion from the EuRoc datasets were selected. The ORB algorithm in OpenCV was first applied for initial feature point extraction, followed by the improved adaptive threshold ORB algorithm for feature point extraction. A total of 20 sets of control experiments were conducted. Due to similar results, one set was selected as the control experiment. The results are shown in Figure 7:

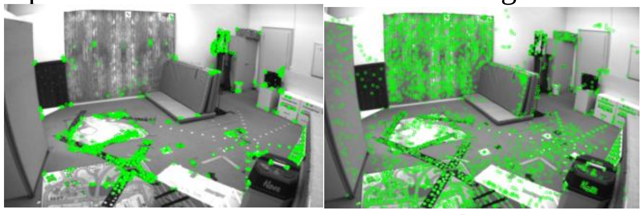


Figure 10. Experimental Results of the ORB Algorithm and the Improved Adaptive Thresholding Algorithm. (a)ORB Feature Detection Results,(b)Adaptive Threshold Feature Point Detection Results

The green circles in Fig.10 indicate detected keypoints. The conventional ORB detector mainly responds to highly textured regions such as edges and corners, while only a limited number of keypoints are found in weak-texture areas (e.g., walls, tabletops, and cabinet surfaces). In contrast, the adaptive-threshold method proposed in this paper increases keypoint detection in weak-texture regions by using the local grayscale variance as the threshold basis, resulting in a more balanced feature distribution. However, noticeable clustering still remains in strongly textured areas.

2) Feature Point Uniformization Experiment: To address the feature point clustering observed in Figure 10(b), a quad-tree algorithm is employed to optimize feature point distribution. Traditional quad-tree methods store only one high-quality feature point per node, resulting in excessive feature points retained in areas with strong texture while insufficient points are preserved in regions with weak texture. This leads to overall uneven feature point distribution. Therefore, an improved quad-tree method is adopted to achieve uniform distribution of feature points across the entire image. Experimental results are shown in Fig.11: (a) displays the feature extraction result, (b) presents the traditional quadtree uniformization result, and (c) illustrates the improved quadtree uniformization result.

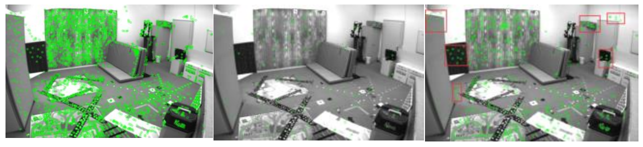
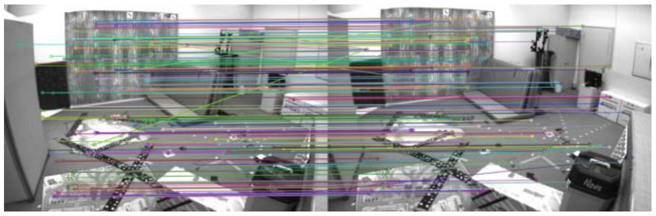


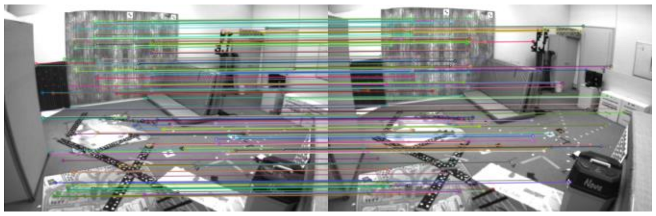
Figure 11. Experimental Results of Forked Tree Algorithm and Improved Quad-Tree Algorithm. (a)Feature point extraction results,(b)Quadtree Uniformization Results,(c)Improving Quad-Tree Uniformization Results

Fig.11 shows that the improved quadtree uniformization method retains more keypoints in the red-box region and other weak-texture areas than the conventional quadtree approach. At the same time, the feature distribution in strongly textured regions remains sufficiently uniform. These results indicate that the proposed strategy improves overall spatial balance without sacrificing coverage in weak-texture regions.

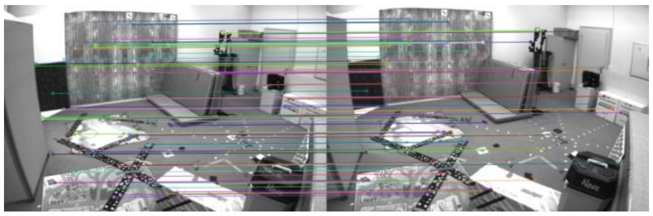
3) Feature Point Matching Experiment: To address mismatching issues, RANSAC and PROSAC are employed to sort and sample matching points, filtering out low-precision points to eliminate erroneous matches. Fig.12 shows the comparison results between FALNN algorithm matching and matching after introducing RANSAC and PROSAC algorithms respectively:



(a)



(b)



(c)

Figure 12. Feature Point Matching Results Comparison Experiment. (a)FLANN Algorithm Matching Results,(b)Matching results after RANSAC algorithm mismatches are discarded,(c)Matching Results After PROSAC Algorithm Mis-Matching Elimination

In Fig.12, colored line segments denote matched correspondences. Subfigure 12(a) presents the direct FLANN matching result, while 12(b) and 12(c) show the results after geometric verification using RANSAC and PROSAC, respectively. The quantitative comparison is listed in Table III:

TABLE III. COMPARATIVE ANALYSIS OF RANSAC AND PROSAC ALGORITHMS IN OUTLIER DETECTION PERFORMANCE

Algorithm	Number of matches	Inner point count	Inner Point Ratio	Matching time
FLANN+RANSAC	246	216	87.8%	15.45ms
FLANN+PROSAC	246	191	77.6%	6.22ms

As shown in Fig.12, the comparison results reveal that direct matching produces a certain number of erroneous matching pairs. Both the RANSAC and PROSAC algorithms exhibit similar numbers of interior points, and their performance surpasses that of the FLANN algorithm's direct matching. Furthermore, PROSAC performs matching calculations using points with high confidence levels rather than randomly selecting all feature points like RANSAC. This approach reduces the number of iterations, resulting in faster matching times compared to RANSAC and enhancing the real-time capability of the matching process.

C. Analysis of Positioning Results

The matching results of consecutive frame feature points provide initial information for subsequent calculations of motion relationships between key frames, directly impacting the accuracy of the drone's trajectory. This study uses the EVO toolkit to evaluate trajectory quality from both global and local perspectives. ATE is adopted to measure the overall consistency between the estimated trajectory and the ground truth, while RPE is used to assess short-interval pose drift between consecutive timestamps. To provide a more complete comparison, mean, median, RMSE, SSE, and standard deviation are reported.

All experiments were conducted using the Vicon Room 1 03 sequence from the EuRoc datasets in stereo camera mode. As shown in Fig.13: (a) depicts the ORB-SLAM3 trajectory plot in 3D space alongside the ground truth trajectory; (b) shows the trajectory of the improved adaptive threshold algorithm and the true trajectory; (c) presents the trajectory of the proposed algorithm after incorporating PROSAC mismatch removal based on b. The dashed line “True” denotes the true trajectory, while solid lines represent the corresponding algorithm's trajectory. The three algorithms in Fig.13(a), (b), and (c) correspond to ORB-SLAM3, Algorithm 1, and the proposed algorithm, respectively.

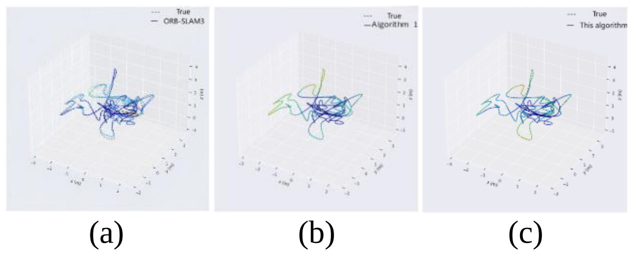


Figure 13. Three Algorithms for Absolute Trajectory vs. Actual Trajectory. (a)ORB-SLAM3 vs. Real Trajectory,(b)Algorithm 1 vs. Actual Trajectory,(c)Algorithm and Actual Trajectory

Fig.13 indicates that all three methods can reproduce the overall trend of the reference trajectory. Among them, the proposed method shows the closest agreement with the ground truth, suggesting better trajectory fitting accuracy.

To quantitatively compare this difference, the Absolute Trajectory Error (ATE) is further analyzed. Fig.14 presents the ATE results for the three methods, where subfigures (a), (b), and (c) correspond to ORB-SLAM3, Algorithm 1, and the proposed method, respectively.

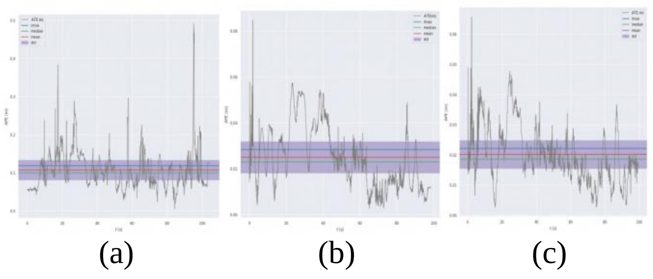


Figure 14. Analysis of Absolute Trajectory Error in Three Algorithms. (a)Absolute Trajectory Error of ORB-SLAM3,(b)Absolute Trajectory Error of Algorithm 1,(c)Absolute Trajectory Error of the Algorithm in This Paper

To further quantify trajectory consistency, the ATE distributions of ORB-SLAM3, Algorithm 1, and the proposed method are compared in Fig.14, and the corresponding statistics are summarized in Table IV.

TABLE IV. ABSOLUTE TRAJECTORY ERROR ANALYSIS

Algorithm	Mean	Median	Rmse	Sse	Std
ORB-SLAM3	0.11	0.10	0.11	0.94	0.051
Algorithm 1	0.024	0.022	0.028	0.79	0.013
Algorithm in This Paper	0.020	0.018	0.022	0.59	0.010

Absolute trajectory error is an error analysis method based on the global trajectory. Subsequently, relative pose error is used to analyze the error between two keyframes at the same time interval, yielding the local pose error. The experimental analysis results for relative pose error are shown in Fig. 15: a) stereo1, stereo2, and stereo3 represent the relative trajectory error comparisons for ORB-SLAM3, Algorithm 1, and the proposed algorithm, respectively; b) shows the comparison of six error metrics—max, min, std, median, mean, and rmse—among the three algorithms.

Since ATE mainly reflects global trajectory deviation, RPE is additionally analyzed to characterize local pose estimation errors between adjacent frames. Fig.15 and Table V present the comparative RPE results for the three methods under the same evaluation protocol.

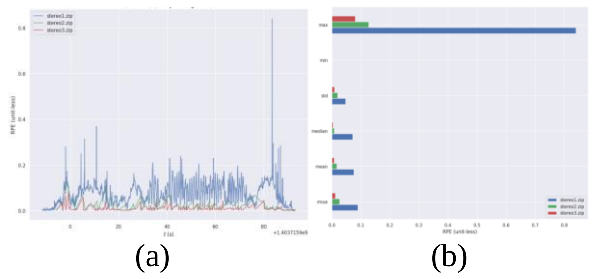


Figure 15. Analysis of Relative Pose Error Among Three Algorithms. (a)Relative Pose Error of Three Algorithms, (b)Relative Pose Error Metrics for Three Algorithms

TABLE V. RELATIVE POSE ERROR ANALYSIS

Algorithm	Mean	Median	Rmse	Sse	Std
ORB-SLAM3	0.076	0.071	0.089	6.94	0.047
Algorithm 1	0.026	0.065	0.026	0.092	0.020
Algorithm in This Paper	0.008	0.005	0.012	0.052	0.009

Based on experimental results, the proposed algorithm in this paper reduces the overall mean absolute trajectory error by 81.8% and the overall mean relative pose error by 89.4% on the Vicon Room 1 03 datasets. Consequently, the algorithm achieves higher precision in fitting real trajectories, providing more accurate initial information for subsequent localization and mapping tasks in visual SLAM systems.

V. Conclusions

This paper presents a front-end optimization method for ORB-SLAM3 targeting three practical issues: insufficient keypoints in weak-texture regions, uneven feature distribution, and mismatches in inter-frame correspondence. A local adaptive-threshold FAST detector is used to improve keypoint extraction in low-texture areas; a texture-aware quadtree strategy is introduced to balance feature distribution while preserving more valid points in weak-texture regions; and PROSAC is employed to improve mismatch rejection efficiency during feature matching. After integration into ORB-SLAM3 and evaluation in stereo mode on the EuRoC dataset, the proposed method maintains normal system operation while improving front-end feature quality and trajectory estimation accuracy. On the Vicon Room1 03 sequence, the mean ATE and mean RPE are reduced by 81.8% and 89.4%, respectively. These results demonstrate that the proposed front-end improvements enhance the robustness and localization accuracy of ORB-SLAM3 in indoor visual SLAM tasks.

References

[1] CHEN CL, HER, PENG CC. Development of an online adaptive parameter tuning VSLAM algorithm for UAV sin GPS-denied environments[J]. Sensors, 2022,22(20):8067.

[2] Macario Barros A, Michel M, Moline Y, et al. A comprehensive Survey of Visual Slam Algorithms[J]. Robotics,2022,11(1):24.

[3] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel and J. D. Tardós, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multi-map SLAM," in IEEE Transactions on Robotics, vol. 37, no. 6, pp. 1874-1890, Dec. 2021

[4] IDRISM.VSLAM analysis using various ORBSLAM parameters setting[J]. PRZEGLAD ELEKTRO TECHNIC ZNY ,2022,1(9):42-47.

[5] Mur-Artal R， Tardós J D. ORB-SLAM2： an opensource SLAM system for monocular， stereo， and RGBD cameras［J］. IEEE Transactions on Robotics， 2017， 33（5）：1255-1262.

[6] J. R. Kim and J. W. Jeon, "Real-time Implement ORB algorithm in FPGA: oFast with Harris corner and rBrief algorithm," 2021 IEEE Region 10 Symposium (TENSYMP), Jeju, Korea, Republic of, 2021, pp. 1-4.

[7] Xie, Y. Wang, Q. Chang, Y. Zhang, X. Fast Target Recognition Based on Improved ORB Feature. Appl. Sci. 2022, 12, 786.

[8] C. Ma, X. Hu, J. Xiao et al., "Improved ORB algorithm using three-patch method and local gray difference," Sensors (Basel, Switzerland), vol. 20, no. 4, 2020.

[9] Ma CQ, Hu XG, Xiao J, Zhang GF (2021) Homogenized ORB algorithm using dynamic threshold and improved quadtree. Math Probl Eng 2021:6693627.

[10] Perrolas G, Niknejad M, Ribeiro R, Bernardino A. Scalable Fire and Smoke Segmentation from Aerial Images Using Convolutional Neural Networks and Quad-Tree Search. Sensors. 2022; 22(5):1701.

[11] Wu K. Creating panoramic images using ORB feature detection and RANSAC-based image alignment[J]. Advances in Computer and Communication, 2023, 4(4): 220-224.

[12] Z. Li, D. Wu, W. Xu, Q. Wu, G. Yang and D. Zhou, "Feature Point Matching Method of Weak Texture Environment in Farmland Based on Improved GMS-PROSAC Fusion Algorithm," in IEEE Access, vol. 13, pp. 4158-4170, 2025.

[13] B. Han, T. Li, Z. Wang and C. Shi, "Improving the performance of the ORB-SLAM3 with low-light image enhancement," 2024 14th International Conference on Indoor Positioning and Indoor Navigation (IPIN), Kowloon, Hong Kong, 2024, pp. 1-7.

[14] LUO Z Y, ZHENG Y, ZHOU J L, et al. A class of augmented complex-value FLANN adaptive algorithms for nonlinear systems[J]. Neurocomputing, 2023, 520: 331-341.

[15] Urban S, Leitloff J, Hinz S. Mlpnp-a real-time maximum likelihood solution to the perspective-n-point problem[J]. arXiv preprint arXiv:1607.08112, 2016.

[16] Zhang Y, Zhang A, Gao M, Liang Y. Research on Three-Dimensional Electronic Navigation Chart Hybrid Spatial Index Structure Based on Quadtree and R-Tree. ISPRS International Journal of Geo-Information. 2022; 11(5):319.