Small-area Fingerprint Recognition Based on Improved ORB Algorithm in Embedded Environment

Jianyu Xiao[1], Jiuan Liu[1], Huanhua Liu[2,*]

[1]School of Computer Science and Engineering, Central South University, Changsha 410075, China
[2]School of Information Technology and Management, Hunan University of Finance and Economics, Changsha 410205, China

Abstract

Most of the fingerprint matching algorithms were proposed for large area fingerprints, which can hardly work effectively in small-area fingerprints. In this work, an improved ORB algorithm is proposed for small-area fingerprint matching in embedded mobile devices. In feature descriptor design, we analyzed the characters of the fingerprint in the embedded mobile devices and discard the multi-scale feature process to reduce the amount of operations. Moreover, we proposed a fusion descriptor combing LBP and rBRIEF descriptor. In the key point matching process, we proposed a two-step (coarse and fine) matching method by using Hamming distance and cosine similarity, respectively. The experimental results show that the proposed method has a rejection rate of 6.4%, a false recognition rate of 0.1%, and an average matching time of 58ms. It can effectively improve the performance of small-area fingerprint matching and meet the application requirements of embedded mobile device authentication.

Keywords: Fingerprint recognition, Internet of thing, ORB, Feature point extraction and matching, Feature fusion.

Received on 21 February 2022, accepted on 08 March 2022, published on 06 April 2022

doi: 10.4108/eetiot.v7i27.297

*Corresponding author. Email: liuhuanhua@hufe.edu.cn

1. Introduction

As the most common biometric feature, fingerprint is widely used in various identification fields due to its better performance in uniqueness, stability and ease of collection. Its related products hold the number one market share in identity verification identification. Fingerprint identification is divided into registration phase and matching phase. In the registration phase, the fingerprint information of the user is recorded in advance and saved in the template database for using in the matching phase. In the matching phase, the fingerprint image to be matched is compared with the feature template in the database, and if the similarity is higher than the set threshold, the fingerprint can be identified as the same one. Depending on the matching method, the most common fingerprint recognition algorithms are: point pattern based algorithms[1][2][3], texture pattern based algorithms[4] and image pattern based algorithms[5]. Among them, the point pattern based fingerprint recognition algorithms are the most widely used.

The traditional fingerprint recognition algorithm based on the point pattern is based on the centroid and singularity for fingerprint image alignment, and then completes the fingerprint authentication by matching the detail feature points, such as endpoints and fork points on the fingerprint. With the increase in portability requirements of mobile devices, the size of their matching fingerprint acquisition sensors has been reduced a lot, and the small-area fingerprint images collected contain fewer detailed feature points[6], which may lead to missing centroids and singularities, and thus the fingerprint to be matched and the feature template may not be aligned. At the same time, when the overlapping area between the fingerprint to be matched and the feature template is too small, the number of detail feature points available for matching is very small, which makes the traditional fingerprint recognition algorithm based on point patterns have a high rejection rate when dealing with small-area fingerprint recognition[7].

Currently, the method of obtaining feature points and their feature descriptors in small-area fingerprint images is generally used to replace the detail feature points in traditional fingerprint recognition methods, and common feature point extraction algorithms include SIFT[8], SURF [9], FAST[10], and Oriented FAST and Rotated BRIEF (ORB)[11], among others. Among them, ORB algorithm has the advantages of small computation and fast running speed, which is suitable for embedded devices with limited computational resources. However, the correlation of the backward part of its descriptors is high, and it is easy to produce mis-matching when matching similarity by Hamming distance in a single way. Therefore, the original algorithm needs to be improved to ensure the real-time algorithm and to reduce its mis-matching rate. To address the above problems, this paper adopts the following three schemes to optimize the ORB algorithm. Firstly, we will not use image pyramids for multi-scale processing to reduce the computation of the ORB algorithm. Secondly, we introduce rotated LBP descriptors[12] to enrich the feature point neighbor information. Finally, the similarity is calculated by using Hamming distance[13] to select the four points to be matched with high similarity, and then the cosine similarity[14] is used to screen out the points to be matched with large disparity in directional features.

In this paper, feature points and their feature descriptors in small-area fingerprint images are extracted for feature point matching based on the improved ORB algorithm. To overcome the problem of small number of feature matching points due to small fingerprint area, multiple fingerprint images of the same finger are recorded in the registration stage. The relative position information is calculated based on their feature point pairs to generate a fusion template[15] to expand the effective area of the registered fingerprint template, enhance the number of trusted feature points in the template, and reduce the rejection rate of fingerprint recognition.

The rest of this paper is organized as follows. Feature point matching based on the improved ORB algorithm is described in Section II. The small-area fingerprint image registration and matching method based on the improved ORB algorithm is described in Section III. The proposed method is experimented and justified in Section IV, and finally the conclusion is given in Section V.

2. Feature Point Matching

Feature point matching is the core work of fingerprint recognition. In the registration phase of fingerprint recognition, it is necessary to match the feature points of the fingerprint image to be registered and perform template fusion according to the relative position information of the feature point pairs. In the matching phase, it is necessary to match the feature points in the fingerprint image to be matched and the registration template, and output the fingerprint recognition result according to the number of feature point pairs. The feature point matching process is shown in Figure 1.

The primary job of feature point matching is to select suitable feature points, and then to construct reliable descriptors for matching based on the neighbor information of feature points[16]. Due to the different acquisition directions and finger press positions, the collected fingerprint images may rotate and translation, and the images at different locations may have similar regions, which are prone to mis-matching based solely on the Hamming distance, in addition to the weak computational power in the embedded environment becomes a large constraint. Therefore, the selection and matching of fingerprint image feature points and feature descriptors need to satisfy the following conditions: 1) the feature points and descriptors are rotation invariant. 2) They require less computational resources and are suitable for mobile devices. 3) When matching based on descriptors, we need to pay attention to their orientation characteristics in addition to similarity

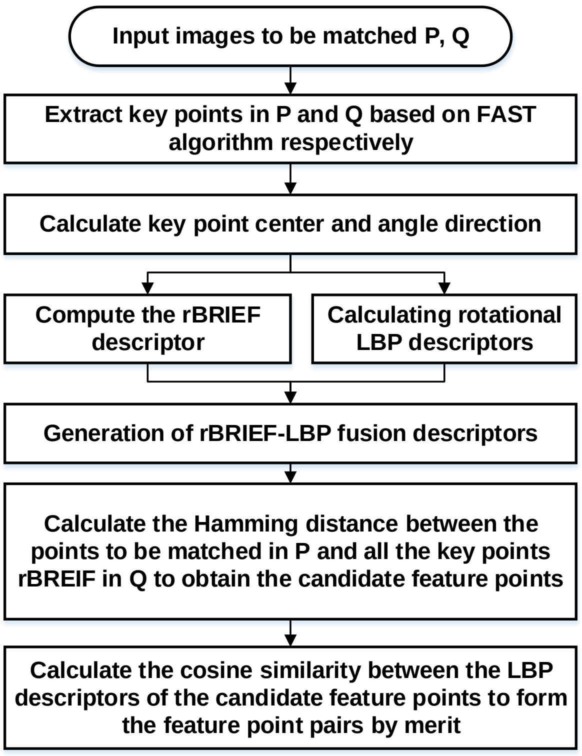

Input images to be matched P, Q
Extract key points in P and Q based on FAST algorithm respectively
Calculate key point center and angle direction
Compute the rBRIEF descriptor
Calculating rotational LBP descriptors
Generation of rBRIEF-LBP fusion descriptors
Calculate the Hamming distance between the points to be matched in P and all the key points rBREIF in Q to obtain the candidate feature points
Calculate the cosine similarity between the LBP descriptors of the candidate feature points to form the feature point pairs by merit

Figure 1. Feature point matching process

2.1. Feature Point Extraction

ORB is a fast feature point extraction and description algorithm. The ORB feature extraction algorithm is divided into two parts, feature point extraction and feature point description, which uses improved based on FAST feature point detection method (oFAST) as feature point extraction and improved based on BRIEF feature descriptor (rBRIEF) for feature point description[16].

The oFAST is an improvement based on the FAST operator, which determines whether a point is a feature point by detecting the gray value of 16 pixels on a circle with a pixel in the image as the center and a radius of 3 pixels and comparing it with the gray value of that point. As shown in

Figure 2, for a point *p* in the image, let its gray value be *Ip*, and if there are *N* consecutive pixel points on the circle composed of 16 pixels that are larger than *Ip*+*T* or smaller than *Ip*-*T*, it is considered a corner point, and *T* is the threshold value, and *N* is generally taken as 9 or 12. To improve efficiency, the difference between point p and the four positions of 1, 5, 9 and 13 can be calculated first. If three of them are larger than the threshold value, it continues to detect. If three of them are greater than the threshold, it continues to detect other pixel points on the circumference, otherwise discard point *p*.

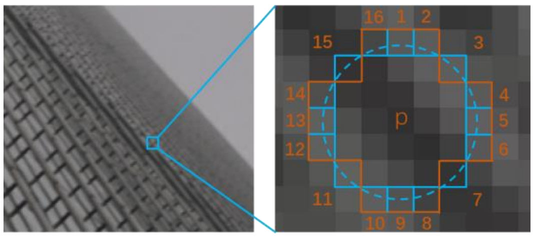


Figure 2. oFAST feature point extraction

Since the acquired feature points are not rotationally invariant, the ORB algorithm used the moment method to determine the orientation of the FAST feature points. The grayscale center of mass method is used to calculate the center of mass of the feature point within a radius of *r*. A vector is formed from the coordinates of the feature point to the center of mass as the direction of the feature point. The moments are defined as follows:

$$\mathrm{m}_{pq} = \sum_{x,y\in r} x^p y^q I(x,y) \quad P,Q = \{0,1\} \qquad (1)$$

where *I*(*x*,*y*) is the image grayscale expression, and when *p*, *q* take the values 0 or 1, the order 0 moments m_{00}, order 1 moments m_{10} and m_{01} are obtained, respectively. The center of mass of this moment is:

$$C = \left(\frac{\mathrm{m}_{10}}{m_{00}}, \frac{\mathrm{m}_{01}}{m_{00}} \right) \qquad (2)$$

Assuming that the corner point coordinates are 0, the angle of the vector is the direction of the feature point. The calculation formula is as follows:

$$\theta = \arctan(m_{01} / m_{10}) \qquad (3)$$

ORB in practical applications such as face recognition algorithm[17] and fast target detection[18] will achieve scale invariance by extracting feature points under different scale images through image pyramids[19]. After creating an image pyramid, the FAST algorithm is used to find feature points in images of different scales, which eventually forms a collection of feature points on images of different scales. In the fingerprint recognition algorithm, since the fingerprint image is not collected by pressing the finger with much force, most of the acquired small-area fingerprint images have only rotation and translation attributes, therefore, the scale basically does not change greatly. Although the number of feature points obtained is reduced, the missing feature points are all located in the low-scale image, which will not affect the matching of feature points in fingerprint images without scale changes. On the contrary, it can also effectively reduce some of the operations, and the subsequent computation of feature descriptors and performing feature matching are also greatly reduced due to the reduction in the number of feature points.

2.2. Feature Descriptors

Feature descriptors[20] are used to describe useful information in an image or image block. According to their preservation methods, they are generally classified into real-valued descriptors (floating-point preservation) and binary descriptors (consisting of 0 and 1). In the image matching process, the binary descriptor is slightly lower than the real descriptor in terms of matching accuracy, but it has the advantages of low time complexity and low memory consumption compared with the real descriptor, which is more suitable for operation in embedded devices with limited resources. rBRIEF descriptor used in the ORB algorithm is a binary descriptor, but since the correlation of the feature descriptors in the rBRIEF descriptor is stronger, it is easy to generate mis-matching. Therefore, this paper uses the LBP feature descriptor to replace the backward part of it to form a fused feature descriptor, which can describe the feature point neighborhood information more accurately.

2.2.1. rBRIEF Descriptors

The rBRIEF descriptor is generated by adding a rotation factor to the BRIEF feature descriptor[21] and reducing its relevance by a greedy search. The idea is to take a certain number of point pairs in the neighbor points detected by the oFAST algorithm and generate feature information based on their gray value magnitude. Its testing criterion is:

$$\tau\,(\mathrm{p}; x, y) = \begin{cases} 1 & p(x) < p(y) \\ 0 & p(x) \geq p(y) \end{cases} \qquad (4)$$

In Eq. (4), *p*(*x*), *p*(*y*) are the point gray values of the key point neighbor, x and y are the point pairs selected according to certain laws, n such point pairs form an n-dimensional binary vector.

$$\mathrm{f}_n(p) = \sum_{1\leq i\leq n} 2^{i-1} \tau(p; x_i, y_i) \qquad (5)$$

Since the BRIEF generated at this time does not have rotation invariance, of which the matching performance decreases sharply when the rotation angle is greater than 30°. The original ORB algorithm rotates all the sampled points in the main direction of the feature points extracted by oFAST, so that they have rotation invariance. First, *n* pairs of points (x_i,y_i) are selected in a certain neighbor center on the feature points in a certain way, forming a 2×*n* matrix as in Eq. (6).

$$S = \begin{pmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{pmatrix} \qquad (6)$$

From the rotation matrix R_θ corresponding to θ, construct $S_\theta = R_\theta S$, where R_θ is:

$$R_\theta = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{7}$$

The specific operation is shown in Figure 3. The coordinate axes are rotated by θ, and the matched point pairs with the main direction θ as the coordinate system are calculated, thus generating the streered BRIEF descriptors with rotational invariance.

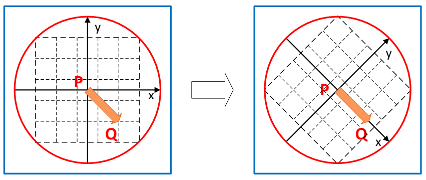


Figure 3. Schematic diagram of the rotation of the coordinate axes in the main direction

In the original ORB algorithm, the correlation between the streered BRIEF descriptors generated after the rotation of the point pairs selected using Gaussian distribution will be greatly increased, which is not conducive to feature matching. Therefore, a greedy search is proposed to replace the Gaussian distribution to select point pairs to reduce the correlation. The principle is to search all point pairs with larger variance and mean value close to 0.5 as the sampled point pairs. To form a sequence of n columns of global optimal point pairs, generate an n-dimensional binary vector, which is the rBRIEF descriptor, and n can generally be taken as 128, 256 or 512, and its form is shown in Eq. (8).

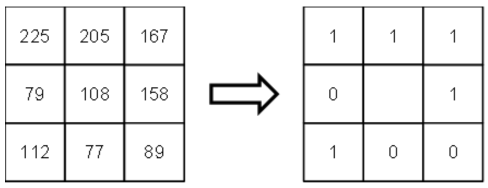
$$V = \underbrace{[1101\ldots0011]}_{n-\text{dimensional binary vector}} \tag{8}$$

2.2.2. LBP feature descriptors

Local Binary Pattern (LBP) feature is a powerful method to describe texture features, which can measure and extract texture information from local neighbor regions in an image, which has been widely used in face recognition. The main principle is to form a binary descriptor by comparing each gray value of a local image window with the central pixel gray value. The LBP descriptor operator is defined as:

$$\begin{cases} L(P,R) = \sum_{i=1}^{N} s(t_i - t_c)2^{i-1} \\ s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \end{cases} \tag{9}$$

In Eq. (9), P and R are the number of surrounding sampled pixels and the sampling radius, respectively. t_i is the gray value of the surrounding pixels, t_c is the gray value of the central pixel, $s(x)$ is a function of the gray value which is to quantify the relationship between the surrounding pixels and the central pixel. A sample calculation is shown in Figure 4.



225	205	167
79	108	158
112	77	89

1	1	1
0		1
1	0	0

Figure 4. Sample LBP descriptor taking values

The LBP feature descriptors generated in this way do not have rotational properties, so the LBP operator can be shifted and its minimum value can be taken as the final LBP descriptor with rotational properties. Here we take P as 16 and R as 2, then the final 16-bit LBP feature descriptor can be generated.

2.2.3. Fusion Descriptors

Since rBRIEF is taken according to its correlation size, the more forward descriptors have less correlation and the more backward descriptors have more correlation, which do not sufficiently utilize the neighbor information of feature points and lead to mis-matching when matching feature points. rBRIEF-LBP feature descriptors can effectively describe the neighbor information of feature points and have rotational invariance after shifting them to obtain the minimum value. Therefore, in this paper, the rBRIEF-LBP fusion descriptor[22] is formed by generating a 16-bit rotational LBP descriptor to replace the back 16 bits of the rBRIEF descriptor without changing its rotation-invariant property, which can enrich the neighbor information of feature points and reduce the feature point mis-matching rate with a very small operation cost. Figure 5 shows the fusion descriptor generation process.

2.3. Feature Point Matching

In the original ORB algorithm, the Hamming distance of the rBRIEF descriptor is directly calculated to determine whether it is the corresponding feature point. The Hamming distance is much more efficient than the similarity algorithms of Euclidean distance, Marxian distance[23] and information entropy [24] in terms of computational efficiency, but it also has a major drawback: the Hamming distance can only be used to measure the similarity between two pairs of feature points, and when there are more similar regions in the image, it is easy to produce a mis-matching[25]. In this paper, we use a combination of both Hamming distance and cosine similarity, using Hamming distance to reflect the difference of vectors in numerical features and cosine similarity to reflect the difference of vectors in directional features.

2.3.1. Hamming Distance

The Hamming distance represents the number of bits that are different between two strings, and we denote the Hamming distance between two strings *x*, *y* by *d*(*x*, *y*).

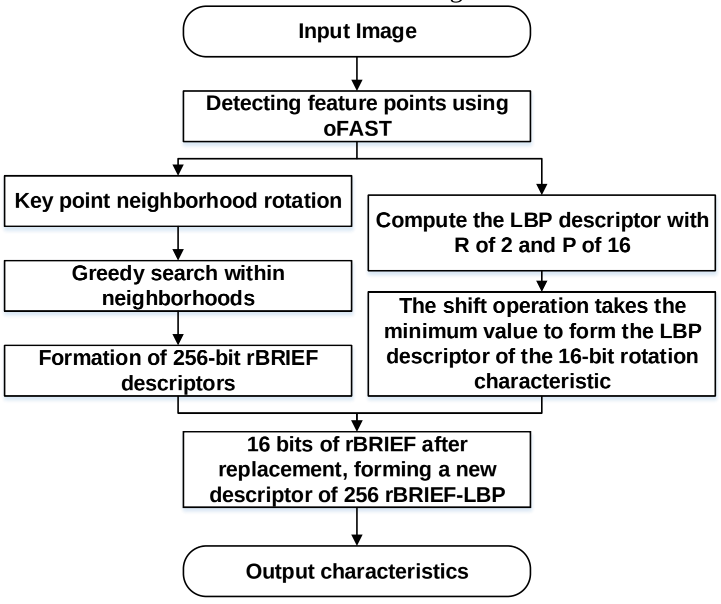

Input Image
Detecting feature points using oFAST
Key point neighborhood rotation
Greedy search within neighborhoods
Formation of 256-bit rBRIEF descriptors
Compute the LBP descriptor with R of 2 and P of 16
The shift operation takes the minimum value to form the LBP descriptor of the 16-bit rotation characteristic
16 bits of rBRIEF after replacement, forming a new descriptor of 256 rBRIEF-LBP
Output characteristics

Figure 5. rBRIEF-LBP fusion descriptor generation process

The Hamming distance is the number of times that two strings can be dissimilar and the number of times that the result is 1 is counted. The Hamming distance focuses on the similarity between two vectors, but does not reflect their directional characteristics

First, we set the threshold *Φ* and calculate the Hamming distance between two feature points, when the Hamming distance is less than *Φ*, it can be used as a candidate match. The formula is as follows:

$$\mathrm{ham}(U(a), U(B)) \leq \phi \qquad (10)$$

In Eq. (10), *a* denotes a feature point in the fingerprint image *P*. *B* denotes the set of feature points in the fingerprint image *Q*. ham(*) denotes the Hamming distance between rBRIEF descriptors, and *Φ* denotes the set threshold value. The experimental data show[26] that when the number of candidate matching points is 4, it can ensure the average recognition time and improve the correct matching rate, *i.e.*, when the number of candidate matching points is greater than 4, the top 4 arranged are selected as the candidate matching points.

2.3.2. Cosine similarity

The cosine similarity is calculated by calculating the cosine value between two vectors to measure the magnitude of their differences in spatial directional characteristics. The cosine similarity of two vectors in the 3D coordinate system is shown in Figure 6.

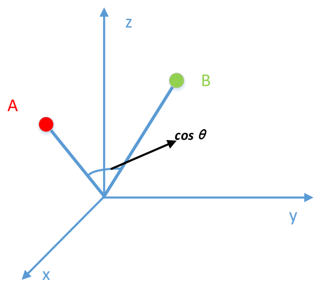


Figure 6. Cosine similarity between vectors in three-dimensional coordinate system

Compared with Hamming distance, cosine similarity is more concerned with the difference between two vectors in direction rather than in distance or length. Its calculation formula is as follows:

$$\cos(\theta) = \frac{a \cdot b}{\|a\| \times \|b\|} \tag{11}$$

In Eq. (11), *a*, *b* are the 2 vectors for which the cosine similarity is requested. The range of the cosine of the vectors is [-1, 1]. The larger the cosine value, the smaller the angle between the 2 vectors. The smaller the cosine value, the larger the angle between the 2 vectors.

Eq. (12) is the formula for calculating the cosine similarity of n-dimensional vectors, where A_i and B_i represent each component of vectors *A* and *B*, respectively.

$$\cos(\theta) = \frac{\sum_{i=1}^{n}(A_i \times B_i)}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}} \tag{12}$$

2.3.3. Feature Point Matching Program

By calculating the Hamming distance of the rBRIEF descriptor, the feature points with closer similarity are screened as the candidate feature points, and then the optimal feature point pair is derived by calculating the cosine similarity. The matching process of the specific feature point pair is as follows.

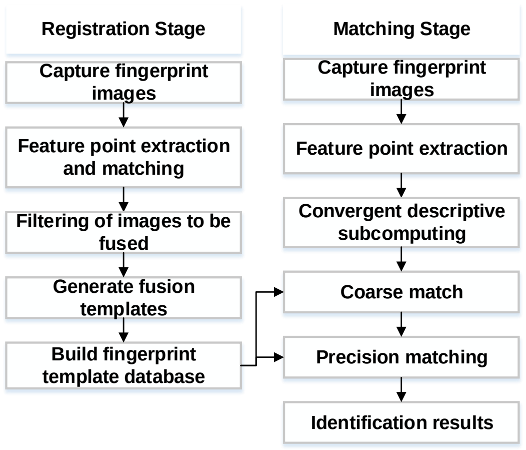
Step 1: Select a feature point from the set of feature points of the fingerprint image *P* as the feature point to be matched.

Step 2: Calculate the Hamming distance between the 240bits rBRIEF descriptors of the feature point to be matched and the rBRIEF descriptors of all feature points of the fingerprint image *Q*. The top 4 feature points in terms of similarity are selected as the candidate feature points[27].

Step 3: Calculate the cosine similarity of the feature points to be matched and the LBP feature descriptors of the candidate feature points, respectively. Set the threshold value, and if the cosine similarity appears to satisfy the set threshold value means that the feature point pair is matched successfully, if there are more than one feature point satisfying the condition, the feature point pair is formed by merit, and vice versa the feature point pair matching fails.

3. Small-area fingerprint recognition

Small-area fingerprint recognition[28] is divided into two processes, fingerprint registration and fingerprint matching, just like traditional fingerprint recognition, and the process is shown in Figure 7.



Registration Stage
Capture fingerprint images
Feature point extraction and matching
Filtering of images to be fused
Generate fusion templates
Build fingerprint template database
Matching Stage
Capture fingerprint images
Feature point extraction
Convergent descriptive subcomputing
Coarse match
Precision matching
Identification results

Figure 7. Flow chart of fingerprint recognition algorithm

In the registration stage, multiple fingerprint images of the same finger need to be captured to generate the fusion template. Under the premise of considering real-time and storage space, in this paper we take 5 images of the same finger to generate fusion templates. The process includes: 1) Selection of images to be fused and benchmark templates. 2) Template fusion[29].

In the fingerprint recognition stage, after obtaining the feature points and feature descriptors of the fingerprint image to be recognized, the feature points are compared with the feature points in the fusion template for coarse and fine comparison. When the number of feature pairs meets the given value, the fingerprint match is considered successful, otherwise the matching fails.

3.1. Fingerprint Registration

3.1.1. Selection of the image to be fused and the reference template

In order to overcome the influence of too small fingerprint area images and expand the number of reliable feature points in the registration template, we generate fusion templates by recording five images of the same finger and generate them based on their feature information. The selection of the images to be fused also has some limitations. If the overlap area between images is too large, the template fusion will result in too many overlapping feature points and cannot achieve the purpose of increasing feature points. If the overlap area between images is too small or no overlap at all, the number of feature point pairs is too small for template fusion. Therefore, after recording five fingerprint images, the similarity between images needs to be calculated separately and the threshold values *T* and *F* are set. If the similarity between the existing template and all other templates is less than *T*, the relative position of the image cannot be determined and a fingerprint needs to be re-entered to replace the template. Otherwise, the two images are the same location and also need to be re-entered.

Table 1. Example of template similarity

Sim.	F1	F2	F3	F4	F5	Average
F1	/	0.76	0.39	0.08	0.13	0.34
F2	0.76	/	0.44	0.14	0.18	0.38
F3	0.39	0.44	/	0.25	0.26	0.34
F4	0.08	0.14	0.25	/	0.65	0.28
F5	0.13	0.18	0.26	0.65	/	0.31

The benchmark template is selected in two ways: 1) The template with the highest average similarity is used as the benchmark template[30]. 2) The overall optimal strategy[31]. We use the overall optimal strategy to select the benchmark template. In addition, each splicing template is selected from an overall perspective. The template with the highest total number of alignable templates among the five templates is selected as the base template. If the similarity threshold is set to 0.2, although the average similarity of template *F3* is lower than that of *F2*, its total number of alignable templates is 4, which is higher than that of *F2*. Template *F3* has reference significance for the alignment of the rest of the templates. Obviously, it is more appropriate to select *F3* as the benchmark template.

3.1.2 Template Fusion

The user only needs to press gently to get a clear fingerprint image, and there are usually only finger translations and rotations. Therefore, the template fusion only needs to consider the rigid transformation to calculate the translation and rotation angles based on the relative position information of the feature point pairs, without considering the affine and projection transformations.

The rigid transformation is shown in Eq. (13) for the coordinate transformation in two dimensions.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & \Delta x \\ -\sin\theta & \cos\theta & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (13)$$

where Δx, Δy denote the horizontal and vertical translations, respectively, and θ denotes the angle of rotation of the fingerprint image. (x', y') denotes the coordinates of the original coordinates (x, y) after rigid transformation.

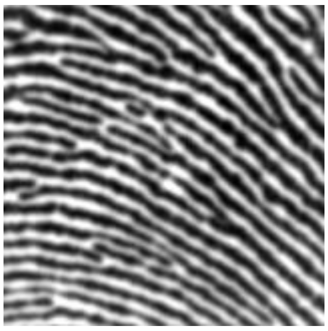
The specific implementation steps of template fusion are as follows.

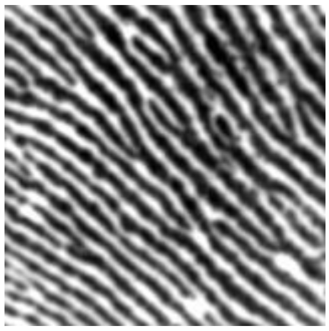
1) Input two fingerprint images *P* and *Q*, extract the corresponding point feature point sets, and the fusion descriptors of feature points. Generate feature point pairs according to the process of feature point comparison.

2) Calculate the corresponding rotation translation parameters based on the information of the feature point pairs.

3) Take the best rotation translation parameters. Each pair of matched feature points corresponds to a set of rotation translation parameters, and use *P* as a fixed template to calculate the coordinates (x, y) of the matched feature point pairs in *Q* after rotation. Then calculate the sum of the Euclidean distances[32] of the coordinates of the matched feature point pairs. After finding the Euclidean distances of the coordinate values corresponding to each pair of matched feature points, the rotation translation parameter with the smallest sum of Euclidean distances is selected as the optimal rotation translation parameter.

Figure 8 shows the two fingerprint images to be fused and the feature fusion, respectively.


（a）Fingerprint image to be fused *P*


（b）Fingerprint image to be fused *Q*

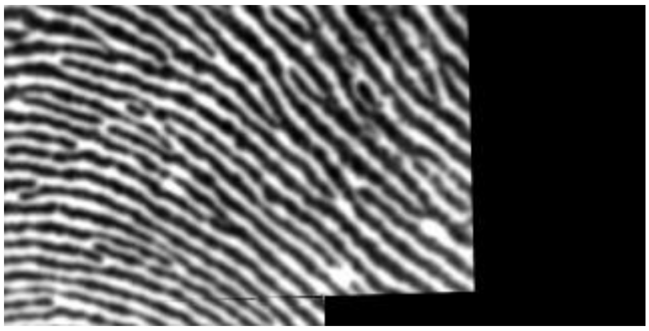

（c）Schematic diagram of template fusion

Figure 8. Template Fusion

3.2. Fingerprint Matching

In order to improve the efficiency of fingerprint matching, coarse matching and fine matching are still adopted in the matching stage. The specific steps of fingerprint matching are as follows.

1) Obtain all feature points and fusion descriptors in the fingerprint image to be matched.

2) For each feature point of the fingerprint image to be matched, calculate the Hamming distance between its rBRIEF descriptor and the rBRIEF of all feature points in the fusion template, and filter out 4 candidate feature points.

3) Calculate the cosine similarity between the candidate feature points and the feature points to be matched respectively, and take the best one to form a feature point pair. If both Hamming distance and cosine similarity meet the requirements, the number of feature pairs is added one, otherwise, the matching fails.

4) If the number of successful feature point matches exceeds the threshold, the finger is recognized as the same finger and the recognition is successful.

4. Experiments and analysis of results

4.1. Experimental environment and data set

The experiments in this paper use MediaTek's MT2503 processor with 32M bit of memory and Nucleus as the operating system. Since the current publicly available FVC series database is mainly for complete fingerprint images, there is no publicly available small-area fingerprint database, so the fingerprint images used in the experiments were acquired by the BF83160X_X fingerprint image collector, which has an acquisition area of 15.6×15.6 mm^2, 508 DPI, and an image size of 160×160 pixels. The database FP1 was entered by 10 volunteers in the laboratory, each volunteer entered 5 fingers, each finger collected 10 images, a total of 500 fingerprints. The acquisition process does not require the finger press position, angle. Two examples of self-built database fingerprint image are shown in Figure 9.

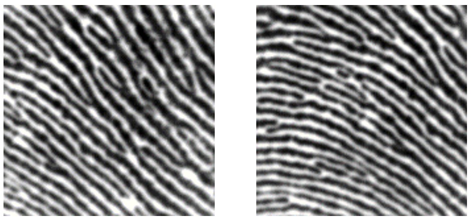
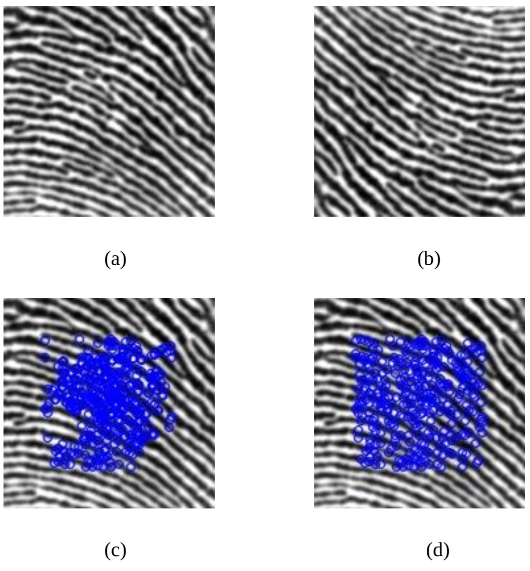


Figure 9. Self-built database fingerprint image (160×160 pixels)

4.2. Feature point matching effects

In order to verify the variation of the number of feature points extracted by the improved ORB algorithm in small-area fingerprint images and its rotation invariant property, the small-area fingerprint images in the database are taken for feature point extraction using two methods and rotated to match the feature points with the original image, and the results are shown in Fig.10. Fig.10 (a) is the original fingerprint image, Fig.10 (b) is formed by rotating Fig.10 (a) by 180°. In Fig.10 (c), the scale operation is performed on the original fingerprint image using the image pyramid, and the total number of feature points extracted on the 4-layer pyramid is 476. In Fig.10 (d), the scale operation is not performed on the fingerprint image, and the number of feature points extracted directly on the original image is 308. The number of feature points is greatly reduced, and the subsequent computation of feature point matching is also reduced. Fig.10 (e) shows the effect of feature point matching for (a) and its rotation by 180° to generate figure Fig.10 (b), which shows that the improved ORB algorithm still has better rotation invariance.



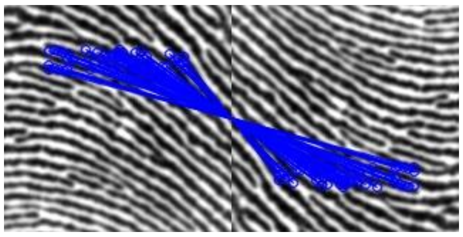


Figure 10. Feature point extraction and matching effect. (a) Fingerprint image a. (b) (a) rotated by 180°.(c) ORB extracts the feature points in (a). (d) Improved ORB to extract feature points in (a). (e) Improved ORB algorithm rotation change matching effect.

4.3. Accuracy Verification

In order to verify the accuracy of the proposed algorithms, the following four algorithms were tested on the self-built small-area fingerprint database, respectively.

1) Traditional fingerprint recognition method (*Trad*) based on detail feature points, which uses the traditional detail feature point method, any fingerprint image in FP1 is matched with the rest of the images in FP1.

2) Small-area fingerprint recognition method based on ORB algorithm with unspliced template (*ORB–N-SPL*), which directly use ORB algorithm to match any fingerprint image in FP1 with the rest of the images in FP1.

3) ORB algorithm based small area fingerprint recognition method with stitching template (*ORB- SPL*), which uses ORB algorithm to generate fusion templates for each finger in FP1 for registration before matching.

4) Small-area fingerprint recognition method based on improved ORB algorithm with template stitching (*IMP-ORB-SPL*), which use the improved ORB algorithm to generate fusion templates for each finger in FP1 for registration and then matching.

The above four algorithms are used to conduct comparison in the fingerprint self-buit database, and the experimental results are shown in Table 2, where FMR is the false recognition rate and FNMR is the rejection rate. From Table 2, it can be seen that: 1) The false recognition rates of all four algorithms are below 0.1%, and the rejection rates are 18.6%, 15.4%, 10.9% and 6.4%, respectively. Comparing with algorithm *ORB–N-SPL* and algorithm *ORB- SPL*, it shows that the template stitching method can improve the effective area of small-area fingerprint images and the number of feature points, thus reducing the rejection rate. 2) Algorithm *Trad* and algorithm *ORB–N-SPL* take 158 ms and 93 ms to match a single fingerprint respectively, which is acceptable. Because the first two algorithms do not perform fingerprint template stitching and need to compare the recorded fingerprint images one by one during matching, which will further increase with the increase of the fingerprint database. 3) Comparison of the rejection rate and single fingerprint matching time of algorithm *ORB-SPL* and algorithm *IMP-ORB-SPL*. The improved ORB algorithm can further reduce the rejection rate of fingerprint recognition. However, the time consumed increases due to the addition of LBP descriptors and cosine similarity calculation, but it can still meet the real-time performance in general.

Table 2. Experimental results.

Algorithm	FMR	FNMR	time(ms)
Trad	0.1%	18.6%	158
ORB-N-SPL	0.1%	15.4%	93
ORB-SPL	0.1%	10.9%	36
IMP-ORB-SPL	0.1%	6.4%	58

Funding. This work was supported by Special Funds for the Construction of an Innovative Province of Hunan, No. 2022GK4009, 2021 Central South University Graduate School-Enterprise Joint Innovation Project (Second Batch), No. 40, and Scientific research project of Hunan Provincial Department of Education, No. 21B0833.

5. Conclusion

In this work, we proposed a small-area fingerprint recognition method based on the improved ORB algorithm to generate fusion templates. In the improved ORB algorithm, the multi-scale feature process is removed to reduce the amount of operations. In addition, the LBP descriptor and the rBRIEF descriptor are introduced to generate the fusion descriptor. The coarse matching is realized by calculating the Hamming distance of the rBRIEF descriptor, and the cosine similarity of the LBP descriptor is calculated to complete the fine matching, which solved the problem of high mis-matching rate of the ORB algorithm to a certain extent. The experimental results show that the rejection rate of the small-area fingerprint recognition method based on the improved ORB algorithm is 6.4%. The elapsed time of single fingerprint matching is 58ms, which is better than the traditional fingerprint matching algorithm based on detailed feature points and can meet the application requirements of embedded mobile device terminal

References

[1] Ranade S , Rosenfeld A . Point pattern matching by relaxation[J]. Pattern Recognition, 1980, 12(4):269-275.
[2] Stockman G , Kopstein S , Benett S . Matching Images to Models for Registration and Object Detection via Clustering[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1982, 4(3):229-241.
[3] Zhu E, Yin JP, Zhang GM. A fingerprint matching method based on multiple reference nodes[J]. Computer Research and Development, 2005, 42(10):7.
[4] Liu H-Y, Liu Shu-Q, Ye Miao, et al. Fingerprint matching algorithm based on line pattern[J]. Computer Engineering and Design, 2017, 38(12):7.
[5] M Tico. Fingerprint recognition using wavelet features.[J]. IEEE Intl.symp.on Circuits & Systems Sydney Nsw Australia, 2001, 2.
[6] Fernandez-Saavedra B , Sanchez-Reillo R , Ros-Gomez R , et al. Small fingerprint scanners used in mobile devices: the impact on biometric performance[J]. Iet Biometrics, 2016, 5(1):28-36.
[7] Ratha, Nalini K , Karu, et al. A real-time matching system for large fingerprint databases.[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 1996.
[8] Gul S , Memon S , Naz B . Image Registration Model For Remote Sensing Images[J]. EAI Endorsed Transactions on Internet of Things, 2018, 4(16):159333.
[9] Bay H , Tuytelaars T , Gool L V . SURF: Speeded up robust features[C]// Proceedings of the 9th European conference on Computer Vision - Volume Part I. Springer-Verlag, 2006.
[10] Viswanathan D G . Features from Accelerated Segment Test (FAST).
[11] Rublee E , Rabaud V , Konolige K , et al. ORB: an efficient alternative to SIFT or SURF[C]// IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011. IEEE, 2011.
[12] Ojala T , Pietikainen M , Maenpaa T . Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns[C]// IEEE Transactions on Pattern Analysis and Machine Intelligence. IEEE, 2002:971-987.
[13] J Martínez-Bernal, Valencia-Bucio M A , Villarreal R H . Generalized Hamming weights of projective Reed–Muller-type codes over graphs[J]. Discrete Mathematics, 2019, 343(1):111639.
[14] Yang G , Zhao X , Fan G . A modification on the vector cosine algorithm of Similarity Analysis for improved discriminative capacity and its application to the quality control of Magnoliae Flos.[J]. Journal of Chromatography A, 2017:34.

[15] Maltoni D , Maio D , Jain A , et al. Handbook of Fingerprint Recognition[J]. Ch Synthetic Fingerprint Generation, 2005, 33(5-6):1314.

[16] Hou Shuwei, Guo Baolong. A fast algorithm for automatic image stitching [J]. Computer Engineering, 2005, 31(15):3.

[17] Chen, Zijia. Research on face recognition algorithm based on Android [D]. Ningbo University.

[18] Li S.H., Xie C.M., Jia Y.Z., et al. Fast target detection algorithm based on ORB features[J]. Journal of Electronic Measurement and Instrumentation, 2013, 27(5):6.

[19] Burt P J , Adelson E H . The Laplacian Pyramid as a Compact Image Code[J]. Readings in Computer Vision, 1987, 31(4):671-679.

[20] Song JF, Wen J. A review of image feature point descriptors [J]. Small and Medium Enterprise Management and Technology, 2016(21):2.

[21] Calonder M , Lepetit V , Strecha C , et al. BRIEF: Binary Robust Independent Elementary Features[C]// European Conference on Computer Vision. Springer, Berlin, Heidelberg, 2010.

[22] Wei W.L., Tan L.N., Lu L.B., et al. ORB-LBP feature matching algorithm with fused descriptors[J]. Electro-Optics and Control, 2020, 27(6):7.

[23] Maesschalck R D , D Jouan-Rimbaud, Massart D L. The Mahalanobis distance[J]. Chemometrics and Intelligent Laboratory Systems, 2000, 50(1):1-18.

[24] J. A. Núez, Cincotta P M , Wachlin F C . Information entropy[J]. Celestial Mechanics & Dynamical Astronomy, 1996, 64(1-2):43-53.

[25] Cheng Y, Zhu WK, Xu GW. Improved ORB matching algorithm based on cosine similarity [J]. Journal of Tianjin University of Technology, 2021, 40(1):7.

[26] Zhang S , Qi T , Huang Q , et al. USB: ultrashort binary descriptor for fast visual matching and retrieval.[J]. IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society, 2014, 23(8):3671.

[27] FAN Wenbo. Research on small area fingerprint recognition technology based on Android system [D]. Xi'an University of Science and Technology.

[28] T.T. Wang. Small-Area Fingerprint Image Recognition and Matching Algorithm[D]. Xi'an University of Electronic Science and Technology, 2020.

[29] Abro M , Talpur S , Soomro N , et al. Shape Based Image Retrieval Using Fused Features[J]. EAI Endorsed Transactions on Internet of Things, 2018, 5(17):159916.

[30] Yang C , Zhou J . A comparative study of combining multiple enrolled samples for fingerprint verification[J]. Pattern Recognition, 2006, 39(11):2115-2130.

[31] Liang K. Research and implementation of ARM-based small-size fingerprint recognition system.

[32] Danielsson P E . Euclidean distance mapping[J]. Computer Graphics and Image Processing, 1980, 14(3):227-248.