Fingerprint Matching and Verification

Anjali Pankan
Matriculation Number: 11016821
Email Id: 11016821@stud.hochschule-heidelberg.de
Under the Guidance Of:
Prof. Dr.-Ing Christof Joneitz
Professor, SRH Hochschule Heidelberg

***Abstract*—In this project, a mechanism for fingerprint matching using SIFT keypoints is discussed. The project aims at detecting the SIFT keypoints from a query as well as a reference images of fingerprint, and matching and verification of these fingerprint images are done by considering these SIFT keypoints. For demonstrating the project, preprocessed Sokoto Coventry Fingerprint Dataset (SOCOFing) has been considered. In order to implement the project, OpenCV library has been used.**

***Index Terms*—Fingerprint Verification, SIFT keypoints, OpenCV**

I. Introduction

Due to the uniqueness and persistence nature of fingerprints, they are considered one of the significant biometric information worldwide. The application of fingerprint detection and verification is not only limited to forensic and law enforcement sections but also other scientific areas such as automatic border control [1] [2]. Fingerprint extraction process can be done either using touch based capturing system or touchless image processing mechanism. Once the fingerprint image is available, the next important step is to match this fingerprint with a known collection of fingerprints stored in a database.

One of the well known fingerprint verification method is by using minutiae features. In a fingerprint, minutiae are ridge terminations and ridge bifurcations. Ridge terminations are determined by a pixel with only one neighbor and ridge bifurcations are identified by pixels with three neighbors. Usually, there will be around 40 to 100 minutiae points in a fingerprint. Since they do not change over time are more suitable for fingerprint matching. One of the state-of-the-art minutiae based algorithm is BOZORTH3 developed by NIST [3]. Even though they are well suited for fingerprint verification, minutiae based approach is not suitable for situations where low quality or partial fingerprint images are provided.

The Scale Invariant Feature Transform or SIFT is an extensively used algorithm to extract invariant feature points from images. It has many applications in the area of computer vision such as image retrieval, object recognition, pose estimation etc. [4]. There has been hand full of works related to SIFT algorithm discussed in the literature, such as, face recognition [5]. In a similar way SIFT can be used for fingerprint matching and verification. As SIFT feature points are determined using texture analysis by finding blobs on entire scale space of an image, it is anticipated that these points will be robust irrespective of deformation and quality [6]. In this project, the SIFT algorithm is used to match a deformed query fingerprint image with a set of reference fingerprint images by calculating keypoints and descriptors and finding the best match by determining matching score.

II. Keypoint Determination using SIFT

In this section, the theory behind keypoint determination using SIFT algorithm is discussed. It was introduced by D.Lowe in 2004. Unlike other methods like Harris [7] algorithm, which is a corner detector algorithm with rotation-invariant property, SIFT is both rotation and scale invariant. A rotation-invariant algorithm can match to another image, which is a rotated version of the previous image. As a result, Harris algorithm can detect the same corner of an image from a corresponding rotated version of it. But, in the case of scaling an image, where the size of the image is increased, the corner may not be conceived as corner in the second image. In such cases, a scale-invariant algorithm, such as, SIFT is necessary.

SIFT algorithm is used to detect keypoints and their corresponding descriptors from an image, which is scale, rotation and transform invariant. The algorithm is shown to be fast and robust in image recognition. There are mainly four steps involved in SIFT algorithm [8] as explained below:

A. Scale-space Extrema Detection

As discussed above, the scaling effect of an image should be considered with the use of different window sizes. As a result, SIFT uses scale-space filtering. When considering a 2D image, Laplacian of Gaussian (LoG) of the image is calculated for different values of σ. Here, σ acts as a scaling parameter. The LoG calculated above can be used as blob detector in an image, which detects blobs of different sizes for different values of σ. For a measurement with lower value of σ of a Gaussian kernel, gives higher value for smaller blobs, whereas, higher value of σ of a Gaussian kernel gives higher value for wider blobs. In this way, the local maxima across the scale and space can be determined. This in effect gives a list of (x, y, σ) indicating there is a local maxima at point (x, y) for a value of σ.

The calculation of Laplacian of Gaussian of an image is expensive. As a result, the SIFT algorithm uses Difference of Gaussian (DoG) to determine local maxiam. The DoG is calculated for difference of Gaussian blurring for an image for different values of σ. Lets say, the σ values taken are σ and

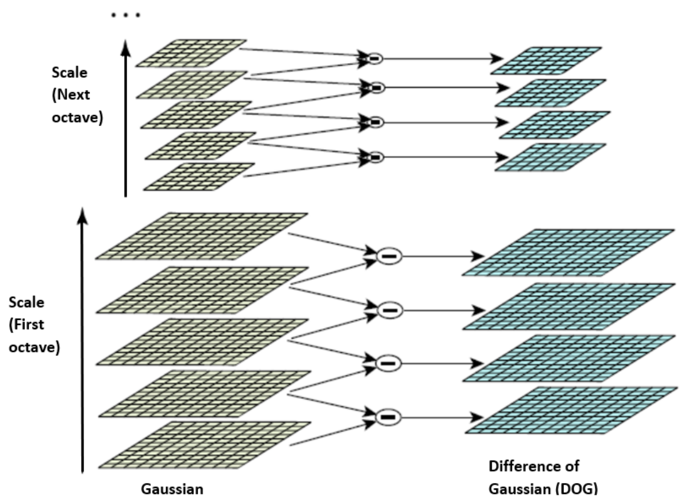


Fig. 1: Computation of the Difference of Gaussian for blurred images at different scales

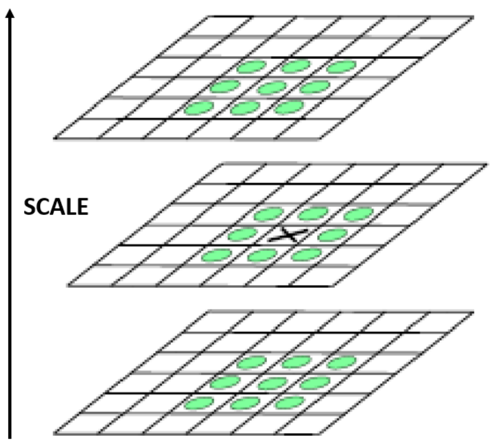


Fig. 2: Local maxima detection, the pixel marked with x in the central scale is compared with 8 neighbor pixels and 9 pixels in the adjacent scales in a $3 \times 3 \times 3$ neighborhood

$k\sigma$, for an image $I(x,y)$, the Gaussian blurred image can be represented as in Equation 1 [5].

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \tag{1}$$

Where:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}} \tag{2}$$

The DoG convolved with image $I(x,y)$ computed for two nearby scales separated by a multiplicative factor k can be determined as in Equation 3.

$$\begin{aligned} D(x,y,\sigma) &= (G(x,y,\sigma) - G(x,y,\sigma)) * I(x,y) \\ &= L(x,y,k\sigma) - L(x,y,\sigma) \end{aligned} \tag{3}$$

This process of determining DoG can be done for different octaves of Gaussian Pyramid of the image. This is represented in Figure 1.

After calculating the DoG, the images are searched to find local maxima with scale and space. In a scale, a pixel is compared with its eight neighbors and nine neighbors of its next scale and previous scale. If that pixel value is a local maxima, it is taken as a potential keypoint. That means, the identified keypoint is best determined in that scale. The above procedure is represented in Figure 2.

B. Keypoint Localization

The next step in SIFT algorithm is the keypoint localization. Once the keypoints are determined as in the previous step, it is required to refine them to get accurate result. By using Taylor series expansion over scale space gives more precise location of maxima. The intensity of the maxima at this location is compared with a threshold to decide on rejection or acceptance of the point. In OpenCV library, this threshold is given as $contrastThreshold$.

Like intensity based refinement, it is also required to identify edge corners to eliminate them from the keypoints as DoG shows higher response for edges. In order to do this, a technique similar to Harris corner detector is employed. According to [8] this step uses a 2×2 Hessian matrix (H) to determine the principal curvature. As per Harris corner detector, for edges, one eigen value produced is more than other. Similarly, a simple function is used in SIFT algorithm. The ratio calculated is compared with a threshold and if it is greater than the threshold it is discarded. In OpenCV library the threshold is called $edgeThreshold$. Finally, the edge points and low contrast points are removed from the keypoints to obtain strong interest points.

C. Orientation Assignment

Once the keypoints are refined to get the interest points, the algorithm finds the orientation of these points. This is done to make the keypoints rotation-invariant due of image rotation. For a given keypoint, a neighborhood is considered around its location depending on the scale and the gradient operation is applied. The gradient operation gives magnitude and direction of that area. A histogram of orientation covering 360 degree with 36 bins (each 10 degrees) is created. Suppose that a pixel in the orientation collection region has value 18.759, then it belongs to the $10 - 19$ degree bin. The orientation contributed by each neighboring pixel is weighed by the gradient magnitude and a Gaussian window with a σ that is 1.5 times the scale of the keypoint [5]. The highest peak from the histogram is selected and any other directions greater than 80% of this also considered for calculating final orientation. As a result, the keypoints are assigned with different orientations, although the scale and location remains the same. This value provide support for stability of matching. A sample histogram representing the histogram is shown in Figure 3 [9].

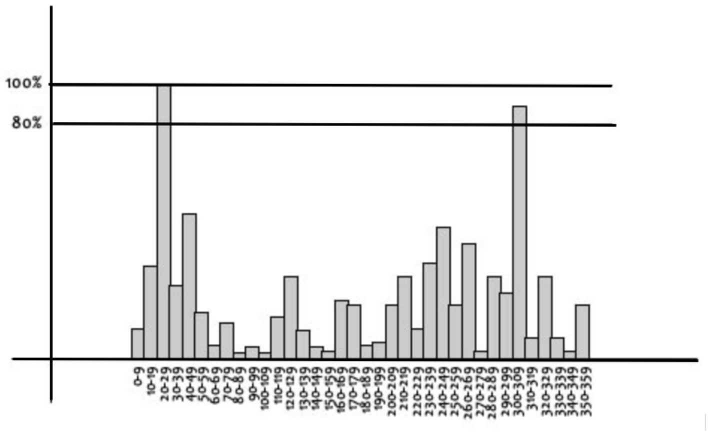


Fig. 3: Gradient orientation histogram

D. Keypoint Descriptor

The last step executed in the SIFT algorithm is the creation of key point descriptors. Here, a 16×16 pixel neighborhood around the keypoint is considered. This region is then divided into 4×4 sub-blocks. An 8 bin histogram is calculated for each sub-blocks relative to the keypoint orientation. As a result, a total of 128 bin values are obtained. These values are arranged in a vector to form keypoint descriptor. This vector is normalized against changes due to illumination, rotation etc. to provide robustness.

III. SIFT Feature for Fingerprint Matching

The fingerprint verification procedure employed in this project is depicted in Figure 4. The procedure consists of five steps as explained below, in which, a query image is expected as an input and is verified against reference fingerprint images stored in a database. The query fingerprint is selected as matched during the decision phase based on a threshold value. Figure 5 represents the schematic diagram of the matching process. Furthermore, the proposed system also provides means for manually checking two fingerprint images. This provided a way to check unsuccessful matching cases.

A. Feature Extraction using SIFT

As explained in the previous section, the keypoints and their descriptors are computed for both query and reference fingerprint images. For SIFT detector, the first step is to create the SIFT detector object using OpenCV function $cv :: xfeatures2d :: SIFT :: create()$. Once it is created, the function $detectAndCompute()$ can be used to detect keypoints and compute descriptors. Listing 1 shows the corresponding code snippet.

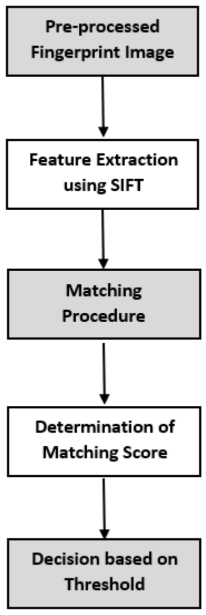

Pre-processed Fingerprint Image
Feature Extraction using SIFT
Matching Procedure
Determination of Matching Score
Decision based on Threshold

Fig. 4: Fingerprint verification procedure

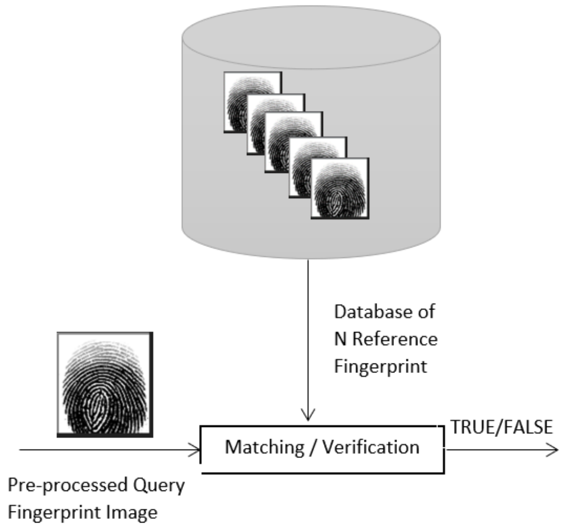

Database of N Reference Fingerprint
Matching / Verification
TRUE/FALSE
Pre-processed Query Fingerprint Image

Fig. 5: Schematic diagram of the matching process

```
// 1. Step: Detect keypoints using SIFT
   algorithm, compute the descriptors
std::vector< cv::KeyPoint > keypoints_1,
   keypoints_2;
cv::Mat descriptors_1, descriptors_2;
cv::Ptr< cv::xfeatures2d::SIFT > detector =
   cv::xfeatures2d::SIFT::create();
detector->detectAndCompute( img_1,
   cv::Mat(), keypoints_1, descriptors_1 );
detector->detectAndCompute( img_2,
   cv::Mat(), keypoints_2, descriptors_2 );
```

Listing 1: Detect keypoints and compute descriptors using SIFT algorithm

B. Matching Procedure

Once the descriptors of the query and reference images are available matching can be done to verify similarity. In this project, a Brute-Force Matcher is used. In this case, a descriptor of a feature in the first image is matched against all other features in the second image using a distance calculation. The closest among them is returned as the best match. Here, the first step is to create a BFMatcher object of type $cv :: BFMatcher$. The match function of the matcher object can be called to create the list of best matches. This function takes three parameters, the query descriptors, train descriptors and a match vector. A mask vector can be passed as the forth parameter to specify permissible matches between query and train matrices of descriptors. Listing 2 shows the corresponding code snippet.

```
// 2. Step: Matching the descriptors to
   determine point correspondences
cv::BFMatcher matcher;
std::vector< cv::DMatch > matches;
matcher.match(descriptors_1, descriptors_2,
   matches);
```

Listing 2: Matching the descriptors to determine point correspondences

C. Determination of Matching Score

For a given query fingerprint image $I_q(x, y)$ and reference image $I_r(x, y)$ number of good matches needs to be identified. This is done by checking for each match obtained in the previous step, if the distance value of the match is less than or equal to a threshold value. Experimentally it is found that a threshold value of 120 gives better result. The output of these check is a list of good matches.

Once the number of good matches p is available, the matching score is calculated using the Equation 4.

$$m(I_q(x, y), I_r(x, y)) = \frac{p}{max\{k_r, k_q\}} \tag{4}$$

Where k_q is the number of keypoints in $I_q(x, y)$ and k_r is the number of keypoints in $I_r(x, y)$.

D. Decision based on Threshold

Th final step towards fingerprint verification is the process of taking decision based on matching score with respect to a threshold t_d. The decision d is made based on the Equation 5.

$$d = \begin{cases} 1, & \text{for } \mathrm{m}I_q(x, y), I_r(x, y) \ \geq t_d \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Here, if the matching score is greater than or equal to the threshold, the query fingerprint is considered matching to the reference fingerprint ($d = 1 = TRUE$), otherwise they are not matched to each other ($d = 0 = FALSE$).

IV. Results

In order to verify the proposed system, fingerprints from Sokoto Coventry Fingerprint Dataset (SOCOFing) has been used [10]. The dataset consist of a collection of real (unaltered) fingerprints as well as three sets of altered fingerprints. The altered fingerprints are provided as Easy, Medium and Hard sets. For every altered fingerprint there is a corresponding unaltered fingerprint in the real set.

In the testing process, images from altered Easy, Medium, Hard set are evaluated against images in the real set. Experiment shows that for a threshold value t_d of 0.1, all altered Easy images (299) are correctly matched to the corresponding images in real set (100%). In the case of altered Medium images, with t_d value 0.1, 290 out off 291 images are correctly matched against the real set (99.65%). For the same t_d in the altered Hard set, 237 out off 247 images are correctly matched against real set (95.95%). Figures 6 and 7 shows an example of successful case with fingerprint good match lines and terminal output of matched values respectively.

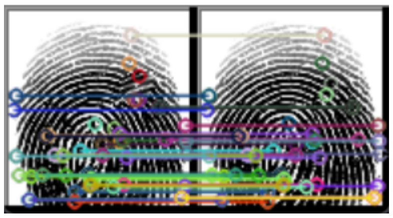
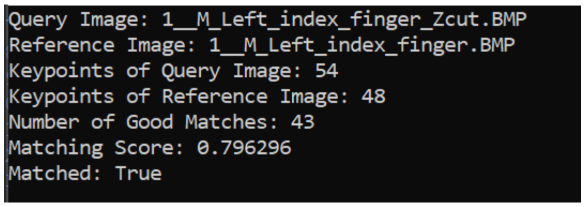


Fig. 6: Successful matching fingerprint images



```
Query Image: 1__M_Left_index_finger_Zcut.BMP
Reference Image: 1__M_Left_index_finger.BMP
Keypoints of Query Image: 54
Keypoints of Reference Image: 48
Number of Good Matches: 43
Matching Score: 0.796296
Matched: True
```

Fig. 7: Successful matching values

Figures 8 and 9 shows an unsuccessful example with fingerprint good match lines and terminal output of matched values respectively.

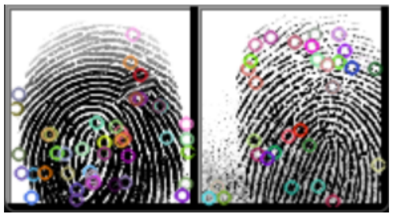
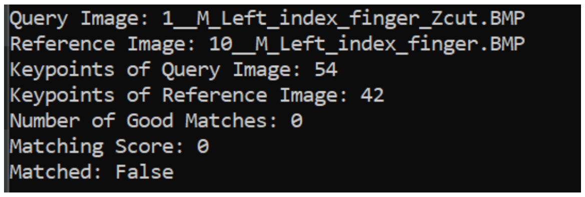


Fig. 8: Non-successful matching fingerprint images



```
Query Image: 1__M_Left_index_finger_Zcut.BMP
Reference Image: 10__M_Left_index_finger.BMP
Keypoints of Query Image: 54
Keypoints of Reference Image: 42
Number of Good Matches: 0
Matching Score: 0
Matched: False
```

Fig. 9: Non-successful matching values

V. Conclusion

In this project, a fingerprint matching and verification mechanism using SIFT algorithm is successfully implemented. Using the SOCOFing fingerprint dataset the matching procedure is evaluated. Results shows that SIFT based approach is quite applicable for fingerprint matching as the algorithm produces large number of keypoints for matching. It is noted that the threshold for distance to consider good matches as well as threshold for matching decision needs to be taken carefully to obtain better results. Experimentally it is showed that a decision threshold value of 0.1 provides best matching decision.

References

[1] Priesnitz, J.; Huesmann, R.; Rathgeb, C.; Buchmann, N.; Busch, C. *Mobile Contactless Fingerprint Recognition: Implementation, Performance and Usability Aspects*. Sensors 2022, 22, 792. https://doi.org/10.3390/s22030792

[2] Christof Jonietz, Eduardo Monari, Chengchao Qu. *Towards Touchless Palm and Finger Detection for Fingerprint Extraction with Mobile Devices*. Vision and Fusion Laboratory (IES), Karlsruhe Institute of Technology (KIT)

[3] *NIST Biometric Image Software (NBIS)*. www.nist.gov. [Online]. Available: https://www.nist.gov/services-resources/software/nist-biometric-image-software-nbis

[4] Faraj Alhwarin, Chao Wang, Danijela Risti -Durrant, Axel Grser. *Improved SIFT-Features Matching for Object Recognition*. BCS International Aca demic Conference 2008 -Visions of Computer Science

[5] M. Bicego, A. Lagorio, E. Grosso, and M. Tistarelli. *On the Use of SIFT Features for Face Authentication*. Computer Vision and Pattern Recognition Workshop (CVPRW'06), 35, 2006.

[6] Ms. S.Malathi, Dr.C.Meena. *Partial fingerprint matching based on SIFT Features*. (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 04, 2010, 1411-1414

[7] *Harris Corner Detection*. docs.opencv.org. [Online]. Available: https://docs.opencv.org/4.x/dc/d0d/tutorial_py_features_harris.html

[8] *Introduction to SIFT (Scale-Invariant Feature Transform)*. docs.opencv.org. [Online]. Available: https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html

[9] *SIFT: Theory and Practice*. aishack.in. [Online]. Available: https://aishack.in/tutorials/sift-scale-invariant-feature-transform-keypoint-orientation/

[10] *Sokoto Coventry Fingerprint Dataset*. pureportal.coventry.ac.uk. [Online]. Available: https://pureportal.coventry.ac.uk/en/publications/sokoto-coventry-fingerprint-dataset