

Fingerprint Recognition

Ashish Kumar and Karan Sehgal



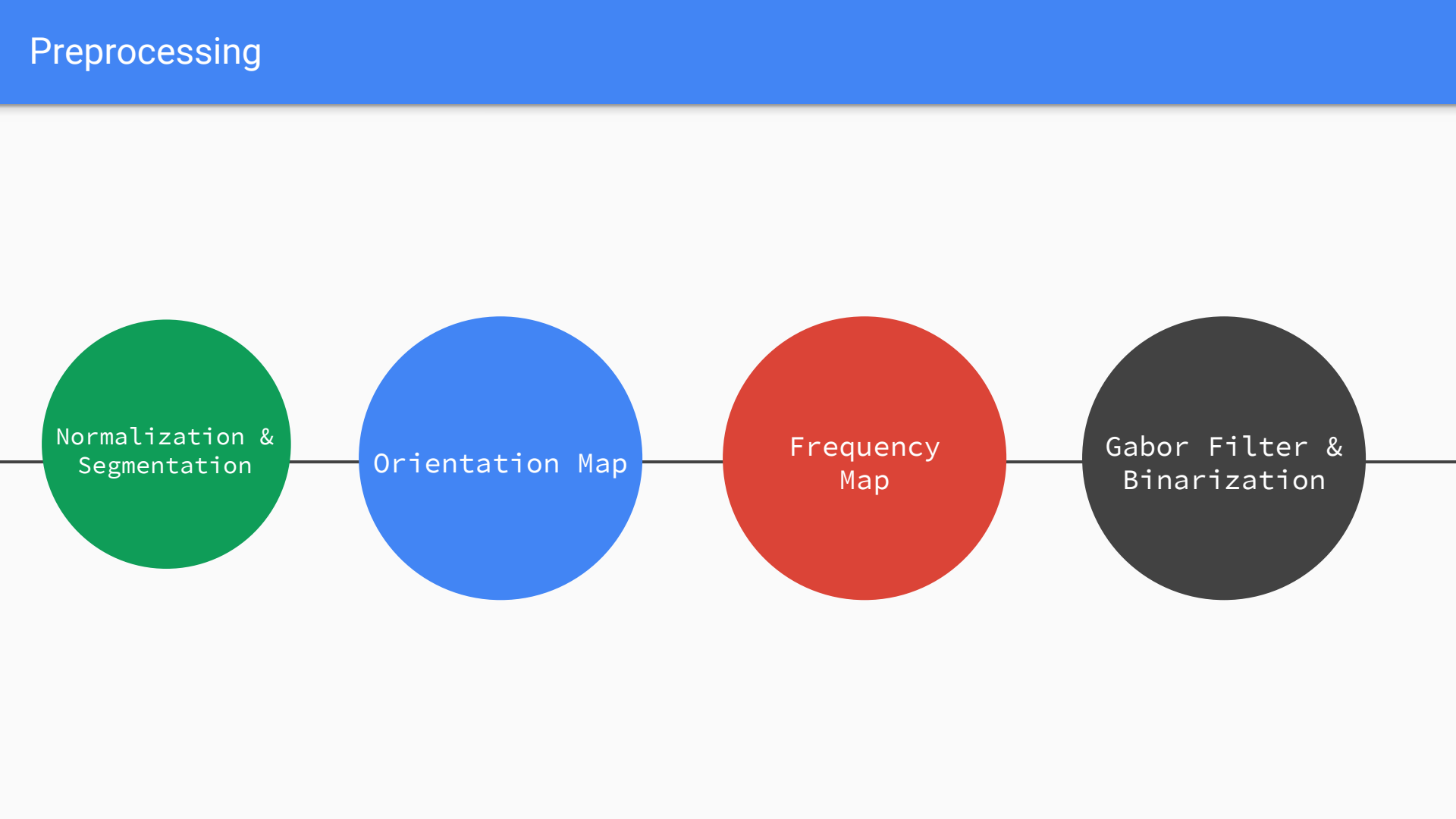
Few Terms

- 1) Ridges : A Ridge is a curved line in a fingerprint image.
- 2) Valleys : Valleys are the lines between ridges.
- 3) Minutiae: Points where either a ridge ends or bifurcates.

References

- Hong, Lin, Yifei Wan, and Anil Jain. "Fingerprint image enhancement: Algorithm and performance evaluation." IEEE transactions on pattern analysis and machine intelligence 20.8 (1998): 777-789.
- Khan, Muhammad Ali. "Fingerprint image enhancement and minutiae extraction." (2011).
- Jiang, Xudong, and Wei-Yun Yau. "Fingerprint minutiae matching based on the local and global structures." Pattern recognition, 2000. Proceedings. 15th international conference on. Vol. 2. IEEE, 2000

Preprocessing



```
graph LR; A((Normalization & Segmentation)) --- B((Orientation Map)); B --- C((Frequency Map)); C --- D((Gabor Filter & Binarization))
```

Normalization &
Segmentation

Orientation Map

Frequency
Map

Gabor Filter &
Binarization

Preprocessing

```
graph LR; A((Thinning)) --- B((Minutiae Extraction)); B --- C((False Minutiae Elimination)); C --- D((-))
```

Thinning

Minutiae
Extraction

False Minutiae
Elimination

-

Normalization & Segmentation

Normalization

First the image is normalized to have 0 mean and 1 standard deviation.

Segmentation

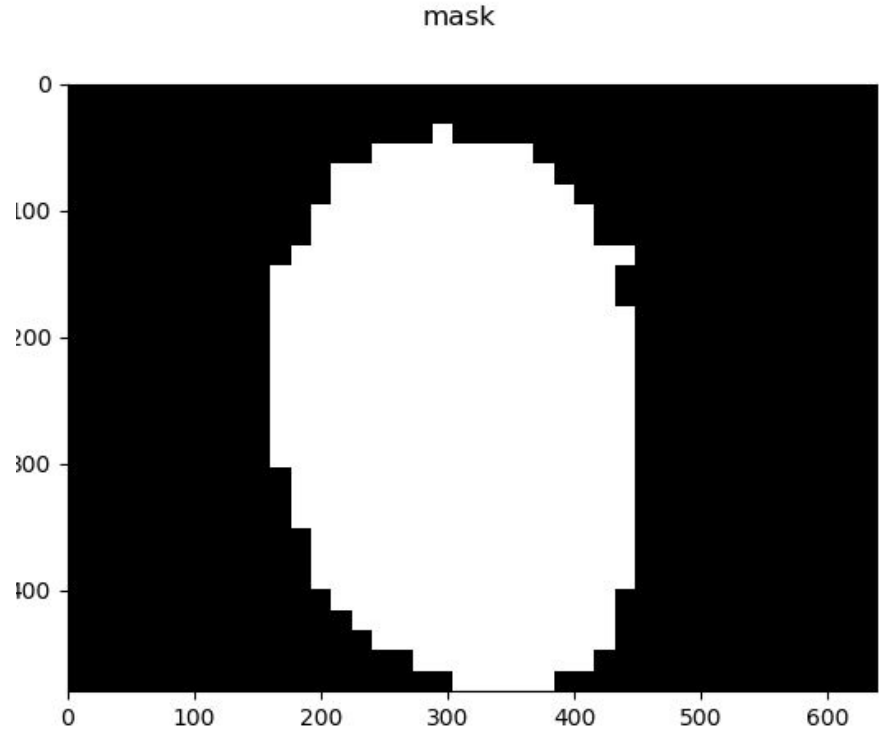
With a threshold of 0.1 and block size of 16x16, standard deviation is calculated of each non-overlapping block of the image, if the deviation is more than the threshold, the block is a part of the segmented mask.

This is because, foreground tends to have larger variance than the background.

Normalization & Segmentation



Original Image



Orientation Map

Fingerprint Enhancement

Fingerprint enhancement is done using a gabor filter followed by thresholding. The gabor filter requires local ridge orientation and local ridge frequency as arguments, hence before applying the filter, these two parameters have to be calculated

Local Ridge Orientation

An image is divided into sets of $W \times W$ non-overlapping blocks and a single orientation is calculated for each block. To calculate the orientation, first the image is smoothened to remove noise, then both horizontal and vertical sobel operators are used to calculate the gradients. The orientation of the block is defined by the equation:

$$\theta_o = \frac{1}{2} \tan^{-1} \left(\frac{\sum_{i=1}^W \sum_{j=1}^W 2G_x(i, j)G_y(i, j)}{\sum_{i=1}^W \sum_{j=1}^W (G_x^2(i, j) - G_y^2(i, j))} \right)$$

Where G_x is the gradient in the x-direction and G_y is the gradient in the y-direction.

Local Ridge Orientation

However, presence of high-curvature ridges, noise, smudges, and breaks in ridges leads to a poor estimate of the local orientation field. Hence to tackle this problem a low-pass filter can be used to modify the incorrect local ridge orientation. In order to perform the low-pass filtering, the orientation image needs to be converted into a continuous vector field, which is defined as follows:

$$\Phi_x(i, j) = \cos(2\theta(i, j)), \quad \Phi_y(i, j) = \sin(2\theta(i, j)),$$

Later perform low-pass filtering as follows:

$$\Phi'_x(i, j) = \sum_{u=-w_\Phi/2}^{w_\Phi/2} \sum_{v=-w_\Phi/2}^{w_\Phi/2} W(u, v) \Phi_x(i - uw, j - vw)$$

$$\Phi'_y(i, j) = \sum_{u=-w_\Phi/2}^{w_\Phi/2} \sum_{v=-w_\Phi/2}^{w_\Phi/2} W(u, v) \Phi_y(i - uw, j - vw),$$

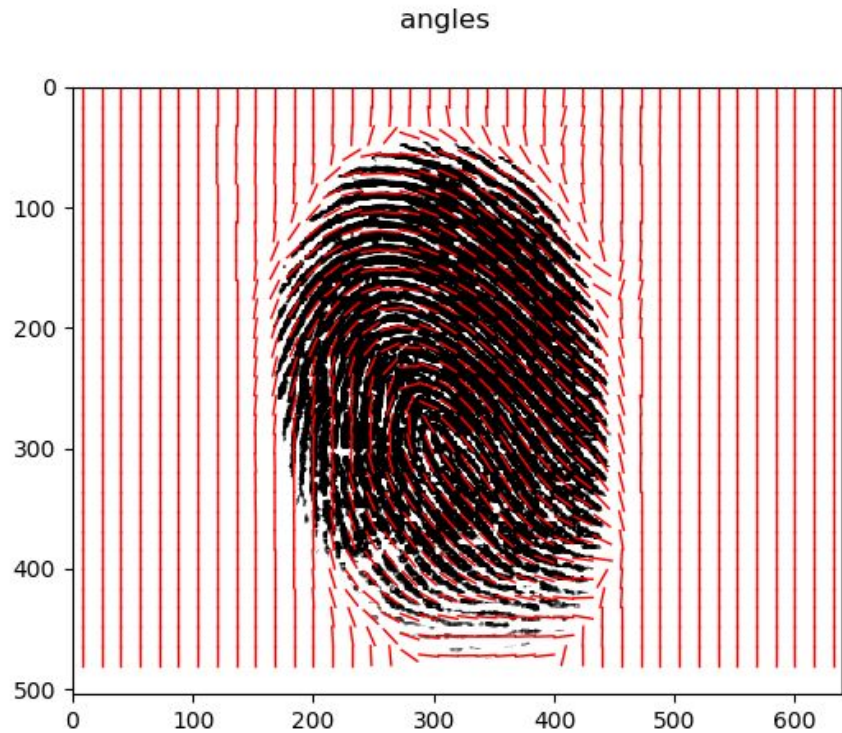
Final orientation is:

$$O(i, j) = \frac{1}{2} \tan^{-1} \left(\frac{\Phi'_y(i, j)}{\Phi'_x(i, j)} \right).$$

Local Ridge Orientation



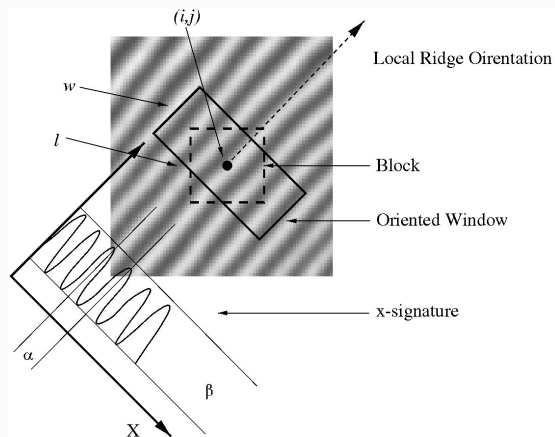
Original Image



Orientation Map

Local Ridge Frequency

The gray levels along ridges and valleys can be modeled as a sinusoidal-shaped wave along a direction normal to the local ridge orientation.



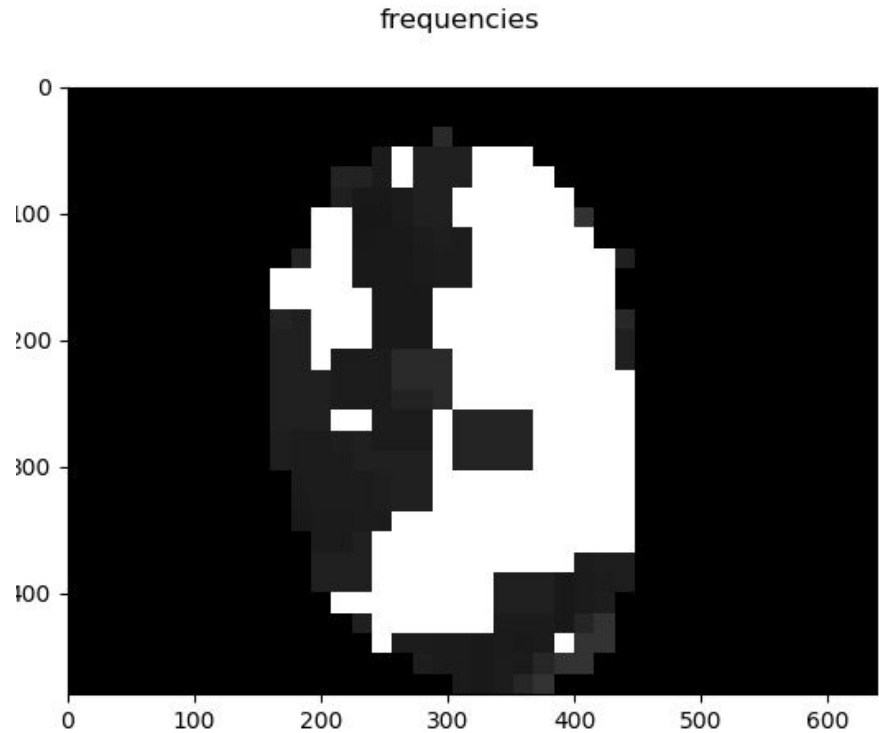
For each block, a window of size 32x16 normal to the block was created. Elements along the orientation direction were summed and peaks were found. The frequency of the block were $1/(\text{average number of pixels between two peaks})$.

This was followed by median filtering for better results

Local Ridge Frequency



Original Image



Frequency Map

Gabor Filter

Gabor filters have both frequency-selective and orientation-selective properties and have optimal joint resolution in both spatial and frequency domains. Currently we are using this [code](#), to apply the gabor filters, which takes our calculated parameters and apply it to the image.

Gabor Filter has the following formula:

$$h(x, y: \phi, f) = \exp \left\{ -\frac{1}{2} \left[\frac{x_{\phi}^2}{\delta_x^2} + \frac{y_{\phi}^2}{\delta_y^2} \right] \right\} \cos(2\pi f x_{\phi}),$$

$$x_{\phi} = x \cos \phi + y \sin \phi,$$

$$y_{\phi} = -x \sin \phi + y \cos \phi,$$

Gabor Filter



Original Image



Enhanced Image

Thinning

Extracting minutiae points require the image to be first thinned. For thinning we are using Zhang Suen algorithm. The steps of the algorithm are the following :

These steps are applied to all pixels:

$A(i,j)$ = the number of transitions from white to black, in the sequence of the eight neighbors around pixel (i,j) , where the sequence starts and ends at the same neighbor (making a complete circle)

$B(i,j)$ = the number of black pixels among the eight neighbors around pixel (i,j) .

Step1 : Following conditions are checked in step 1 :

- 1) The pixel is black and has eight neighbours
- 2) $2 \leq B(i,j) \leq 6$
- 3) $A(i,j) = 1$
- 4) At least one of the north, east, and south neighbors is white
- 5) At least one of east, south, and west neighbors is *white*

Thinning

Step 2:

- 1) The pixel is black and has eight neighbours
- 2) $2 \leq B(i,j) \leq 6$
- 3) $A(i,j) = 1$
- 4) At least one of the north, east, and west neighbors is *white*
- 5) At least one of north, south, and west neighbors is *white*

If a pixel satisfies any of the two steps conditions, than it is removed (set to background color). These steps are iteratively repeated on the image till no pixels is changed in both step 1 and step 2.

The result of the above algorithm is the thinned image.

The intuition of the algorithm is to repeatedly remove certain neighbours of the image.

Thinning Results



Enhanced Image



Thinned Image

Black fingerprint on white image was not visible as clear, so we have shown the opposite

Minutiae Extraction

There are two major steps in minutiae extraction :

1) Extracting Minutiae : Minutiae extraction is done using the following technique:

he CN (Crossing Number) is calculated for each pixel. Following is the definition of CN:

$$CN = 0.5 \sum_{i=1}^8 |P_i - P_{i+1}|$$

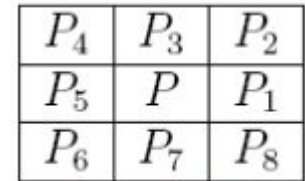


Fig. 11 3X3 neighbourhood

CN = 1 means a ridge ending and CN = 3 means a ridge bifurcation. Both of which are minutiae points.

2) Removal of false minutiae points from corners: The segmentation filter used previously is used here. A 11x11 region corresponding to each minutiae point is considered in segmentation filter (with the minutiae point a center). If all the values in segmentation filter are 1 then it is kept, otherwise the point is not considered a minutiae point. This helps in removal of false minutiae points at the corner.

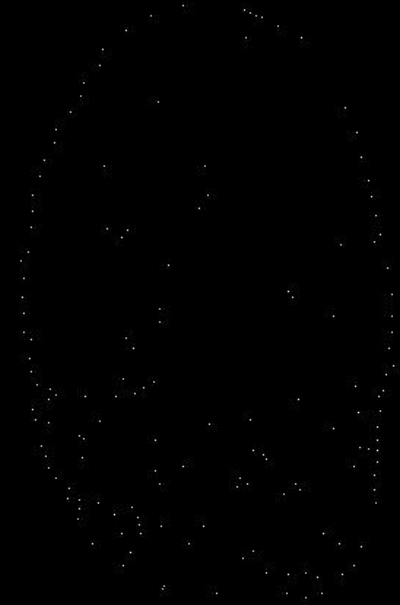
Minutiae Extraction

3) Removal of false minutiae points from other areas : We loop through each pixel in the fingerprint image. If it is a minutiae point, we consider a grid of 17×17 around this point, if there is any other minutiae point in this grid we remove all minutiae points, otherwise we move to the next pixel. This removal is in order to avoid false minutiae due to ridge breaks.

Minutiae Extraction



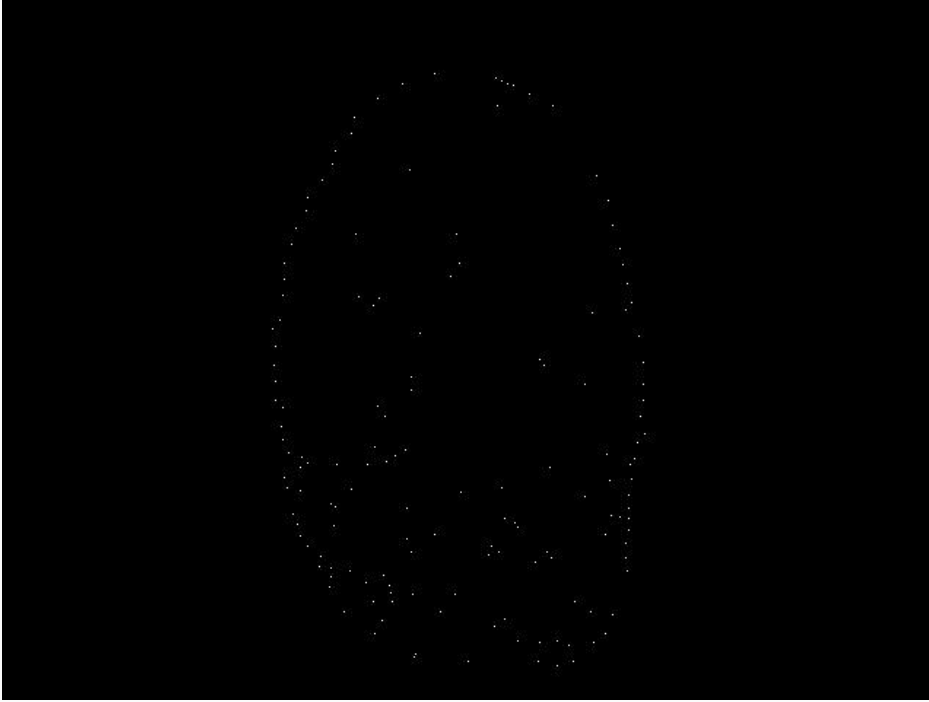
Thinned Image



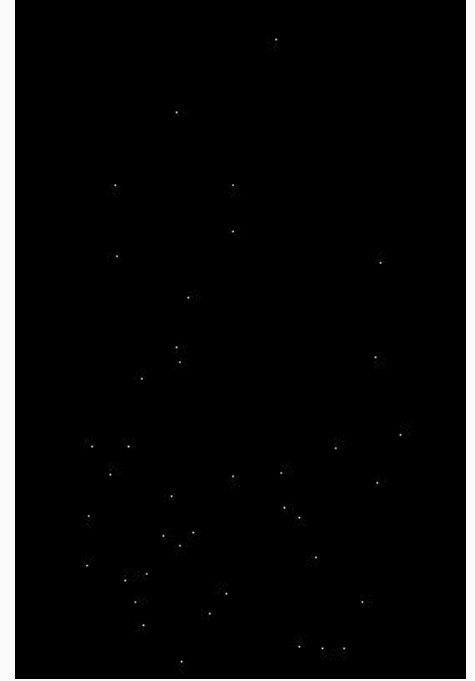
Extracted Minutiae

Black fingerprint on white image was not visible as clear, so we have shown the opposite

Minutiae Extraction



Extracted Minutiae



Minutiae after false removal

Black fingerprint on white image was not visible as clear, so we have shown the opposite

Local Structure Matching

Feature vector of a minutia point is defined by $F_k: (x_k y_k \phi_k t_k)$

$(x_k y_k)$: coordinates of the minutia, ϕ_k orientation of the minutia, t_k type of the minutiae (ridge ending or bifurcation)

For local structure matching, we take a minutia along with its 2-nearest neighbors to form one group of minutiae. Each group of a minutia M_k , with M_i and M_j as its nearest neighbors, is represented by a feature vector:

$$Fl_k = (d_{ki} \ d_{kj} \ \theta_{ki} \ \theta_{kj} \ \varphi_{ki} \ \varphi_{kj} \ n_{ki} \ n_{kj} \ t_k \ t_i \ t_j)^T, \quad (6)$$

Local Structure Matching

In the equation:

$$Fl_k = (d_{ki} \ d_{kj} \ \theta_{ki} \ \theta_{kj} \ \varphi_{ki} \ \varphi_{kj} \ n_{ki} \ n_{kj} \ t_k \ t_i \ t_j)^T, \quad (6)$$

where:

$$\begin{aligned} d_{ki} &= \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}, \\ \theta_{ki} &= d\phi \left(\tan^{-1} \left(\frac{y_k - y_i}{x_k - x_i} \right), \varphi_k \right) \\ \varphi_{ki} &= d\phi(\varphi_k, \varphi_i) \end{aligned}$$

n_{ki} and n_{kj} are the number of ridges between the minutiae M_k and M_i , and M_k and M_j respectively

Local Structure Matching

Finding ridge count between minutiae: To find the ridge counts, we iterate through each minutiae point and use the K-nearest neighbour algorithm to find the two closest minutiae points.

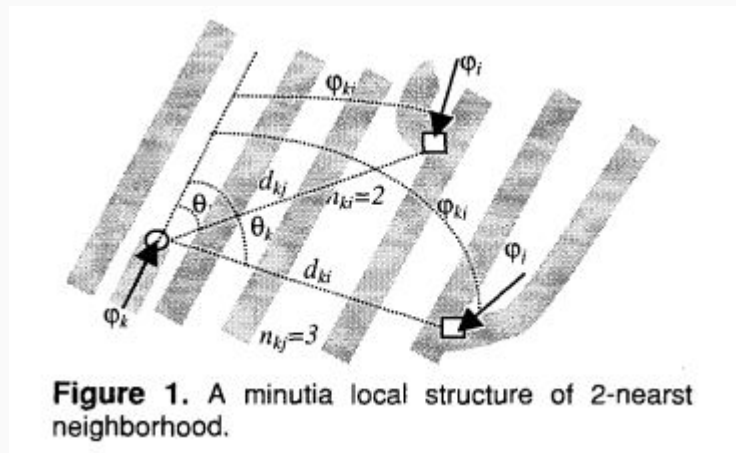
Then, we iterate through the two lines joining the reference minutiae point and the two nearest minutiae points and count the number of ridges encountered. We check the number of transitions from 0 to 1 and then 1 to 0 to count the number of ridges.

The initial and the last one's are ignored since those are the ridges to which the minutiae points belong and hence should not be counted.

Local Structure Matching

We use one base minutia as a reference and find the positions of the nearest two minutiae through their euclidean distance from the base, difference in orientation, number of ridge crossings and type, to create a feature vector representing a local structure that might match with some different fingerprint of the same finger.

Since all the features in the feature vector are differences or constants, this makes these local structures rotation and translation invariant, in theory.



Local Structure Matching

Similarity Level:

Let Fl_i^I be the feature vector of an input image and Fl_j^T be the feature vector of a template fingerprint. A similarity function $sl(i,j)$ can be defined as:

$$sl(i, j) = \begin{cases} \frac{bl - W|Fl_i^I - Fl_j^T|}{bl}, & \text{if } W|Fl_i^I - Fl_j^T| < bl \\ 0, & \text{Others} \end{cases} \quad (7)$$

$$W = (w_d \ w_d \ w_\theta \ w_\theta \ w_\phi \ w_\phi \ w_n \ w_n \ w_t \ w_t \ w_t). \quad (8)$$

Where $bl=36$, $w_d=0.5$, $w_{\theta}=0.3*180/\pi$, $w_t=1$

Global Structure Matching

Because only a local structure is used to describe a minutia, two fingerprints from different fingers may have quite a few similar minutia local structures. Hence to make our matching more reliable we use global structure matching.

To avoid orientation and translation effects, we change our coordinate system to polar coordinates. Each minutiae would be described by its polar coordinates with respect to a origin. This origin should be our best matched local structure. Therefore our origin **b** would be described by:

$$sl(b1,b2) = \max(sl(i,j)) \quad \forall i,j$$

Global Structure Matching

Feature Vector:

$$Fg_k = \begin{pmatrix} r_{kb} \\ \theta_{kb} \\ \varphi_{kb} \end{pmatrix} = \begin{pmatrix} \sqrt{(x_k - x_b)^2 + (y_k - y_b)^2} \\ d\phi \left(\tan^{-1} \left(\frac{y_k - y_b}{x_k - x_b} \right), \varphi_b \right) \\ d\phi(\varphi_k, \varphi_b) \end{pmatrix}$$

Global Structure Matching

Matching Level:

$$Bg = (8 \pi / 6 \pi / 6)$$

$$ml(i, j) = \begin{cases} 0.5 + 0.5sl(i, j), & \text{if } |Fg_i^I - Fg_j^T| < Bg \\ 0, & \text{Others} \end{cases}$$

Matching Score:

M and N are the number of minutiae between
two images

$$Ms = 100 \times \frac{\sum_{i,j} ml(i, j)}{\max\{M, N\}},$$

Since even unmatched minutiae would give a score of 0.5, hence compressing the range of detection, we only took the matched ml scores.

Experimentation

Due to noisy and damaged images in the dataset, we tried to experiment with few things in order to get better results.

The following are the things we tried:

- Noise Reduction
- Orientation Correction
- Cropping/ROI selection
- Morphologic Processing

Noise Reduction

Both gaussian and median filtering degraded the quality of enhanced image. Since might be due to the fact that fingerprints are very sensitive to details, hence any kind of blurring operations hampers its quality.

Orientation Correction

K-means with $K=2$ was used to cluster the minutiae into 2 clusters. Then the angle of the line joining the two centroids of the clusters was calculated and the image was rotated about that angle so as to align the clusters at 0 angle level.

This was done for both the images, hence in theory both the images would be aligned in the same orientation.

This improved the accuracy for the images where proper clusters were forming, however it is possible that the desired clusters are not being formed, which would make the orientation worse. Hence this was not used.

We found two methods for finding the ROI for our image:

1. Cropping: Since the background is white and our fingerprint is black, we just found the minimum and maximum (x,y) coordinates for the black pixel and that is our cropped image. This improved our accuracy by a little margin
2. Variance: The core of the fingerprint carries the most information about the fingerprint, hence we should use that as our ROI. We know our orientation map would change a lot more near the core area than other areas, hence a region with maximum variance should give us that area. However, the size of the ROI cannot be fixed and corners of the fingerprint were sometimes being selected instead, hence this approach was not used.

Morphological Processes

Since the images in our dataset are withered and noisy, we tried to use morphological processes like dilation, erosion, opening and closing. However, since the ridges are very near to each other, morphological operations, even with a 3x3 structuring element did not help in improving the quality of the fingerprints.

Closing followed by Opening did give good results for few fingerprints since it joined few breaks in the damaged image, however for smudged fingerprints, it drastically degraded the enhanced image.

Results



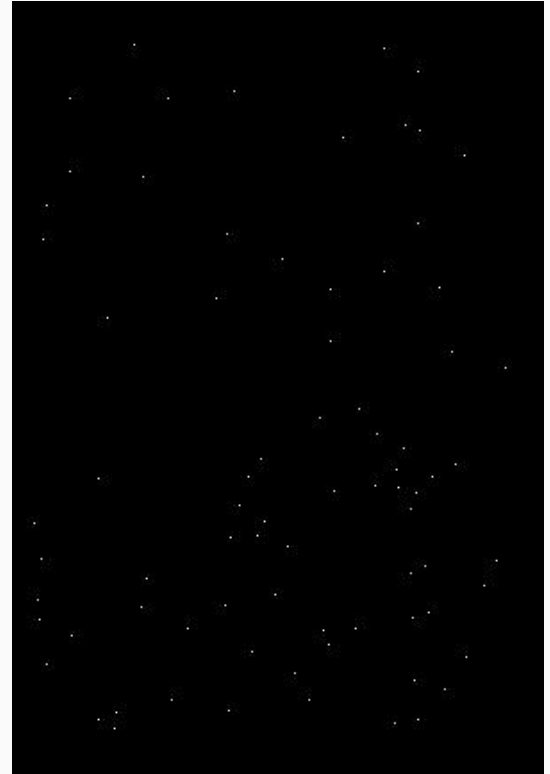
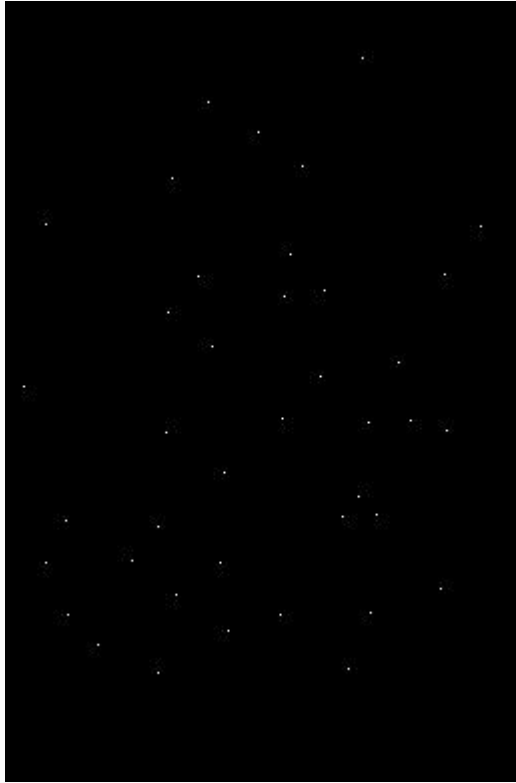
Two images of same fingerprint

Results



Enhanced and cropped Images

Results



Extracted Minutiae in both fingerprints. Final Result: Matched with matching score 43.8%

Results



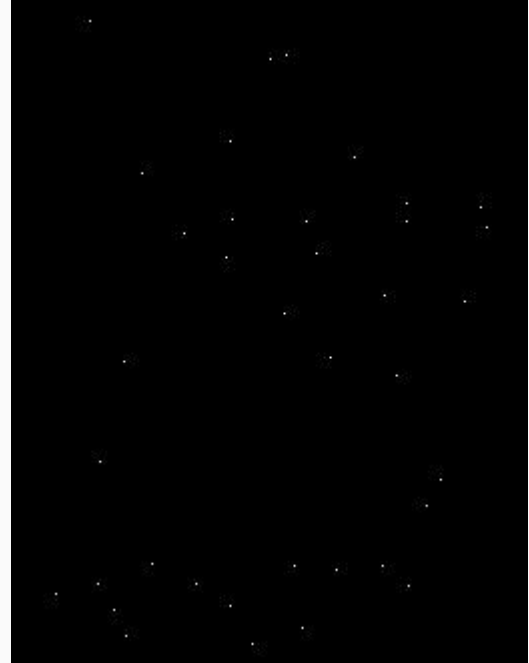
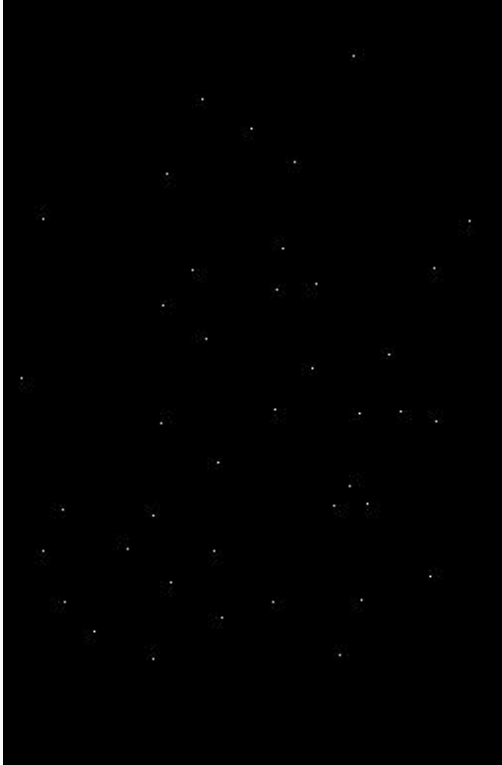
Two images of different fingerprint

Results



Enhanced and cropped Images

Results



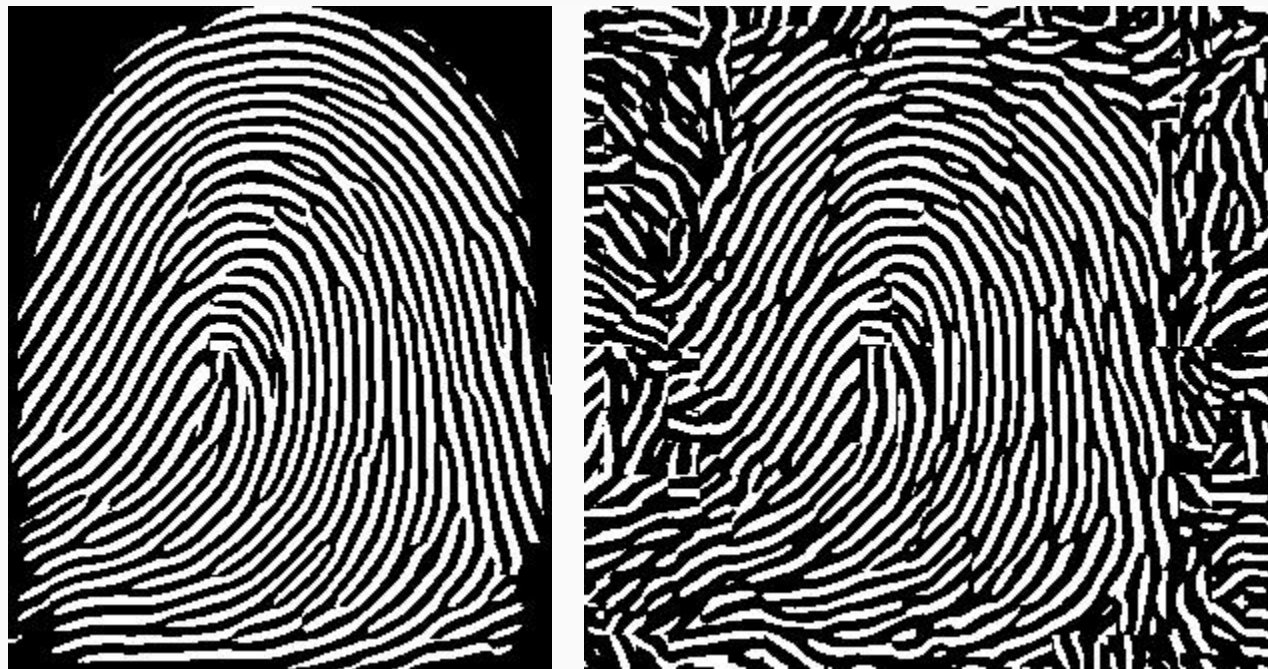
Extracted Minutiae in both fingerprints. Final Result: Unmatched with matching score 13.1%

Results



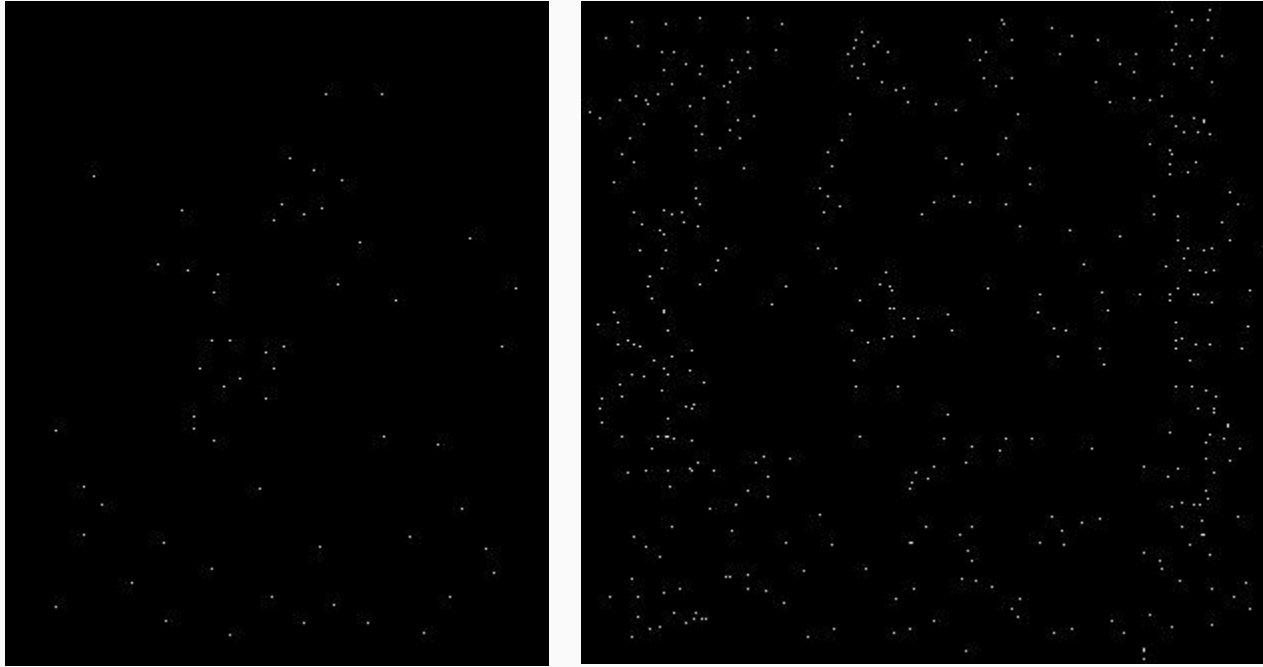
Image and speckle noise image (We did not include rotation in tampered image because due to noise, KMeans algorithm did not give any good outputs which is to be expected since KMeans depends on pixel intensities).

Results



Enhanced and cropped Images

Results



Extracted Minutiae in both fingerprints. Final Result: Matched with matching score 38.04%

Experimentation for results

We tried to test the fingerprint algorithm on our own fingerprints but we could not find any suitable way of taking the fingerprints from the fingerprint scanner. The scanner only gives fingerprints to authorized websites.

We also tried to work out of this by used stamp pad and taking our fingerprint on paper. But that also did not work out due to very poor image quality and degradation of fingerprint due to ink.