Computer Vision
13016370

Edited by:
Dr. Ukrit Watchareeruetai
International College
King Mongkut's Institute of Technology Ladkrabang

Problem-based learning:
Fingerprint verification

Fingerprint



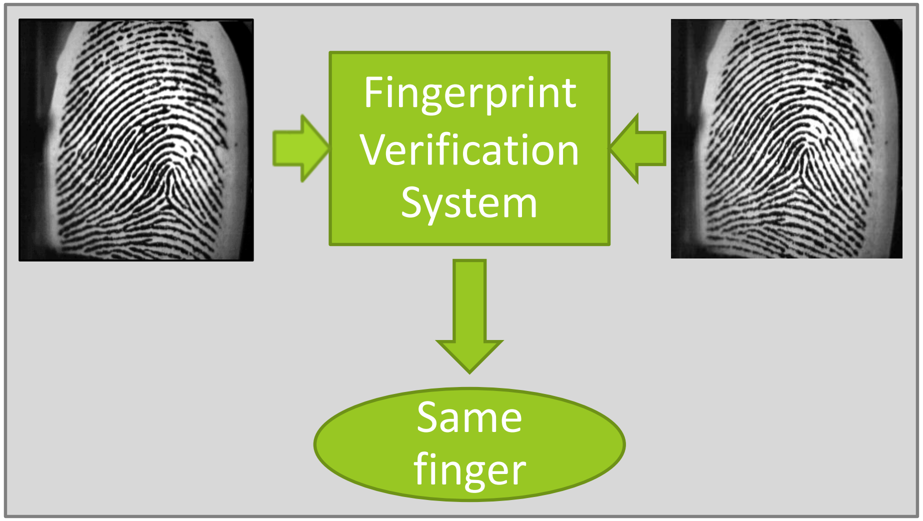


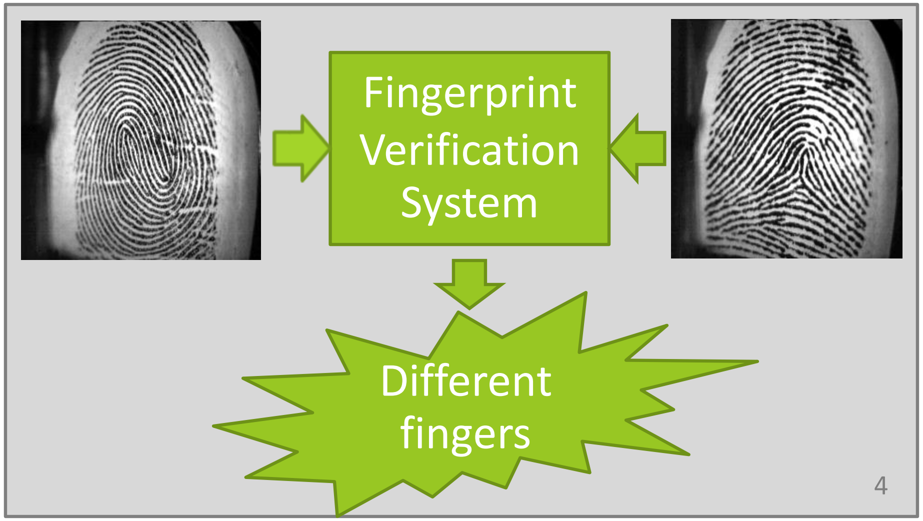



Images: Fingerprint Verification Competition (FVC2000, FVC2002, FVC2004, FVC2006)

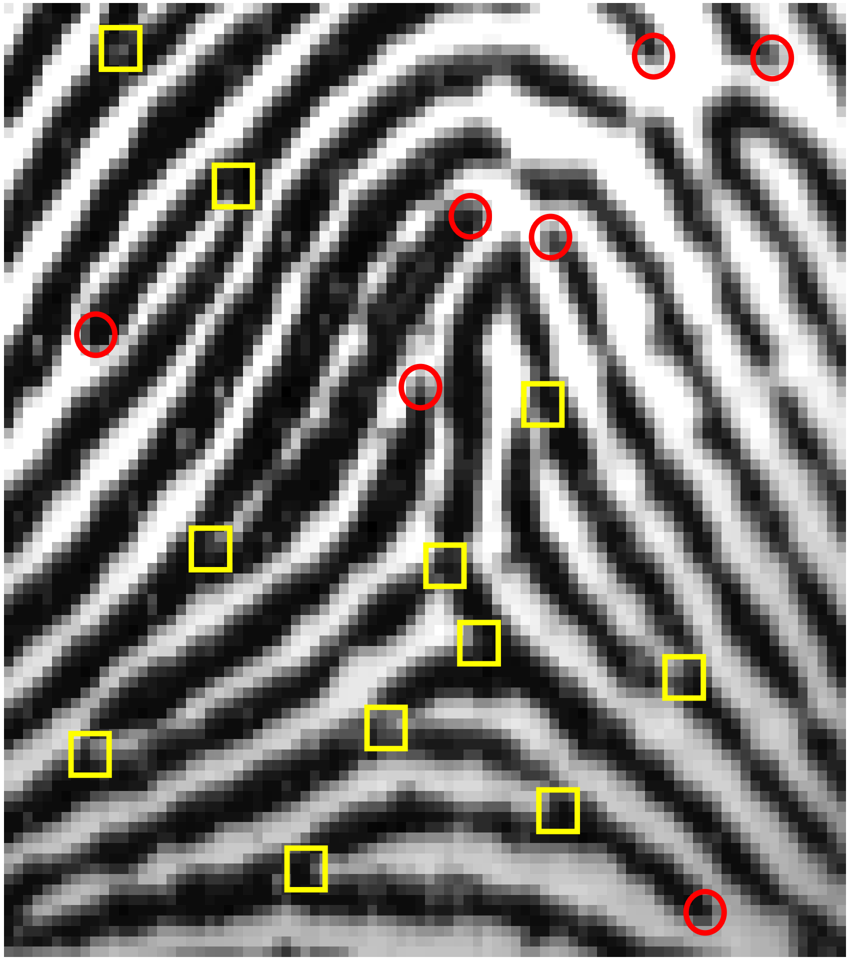
Fingerprint verification

- **Task:** to verify if two given fingerprints are from the same finger
- **Input:** two fingerprint images
- **Output:** a boolean answer (yes/no)





How to recognize fingerprints?



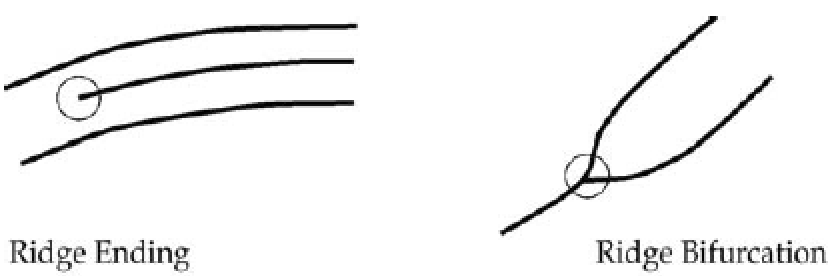


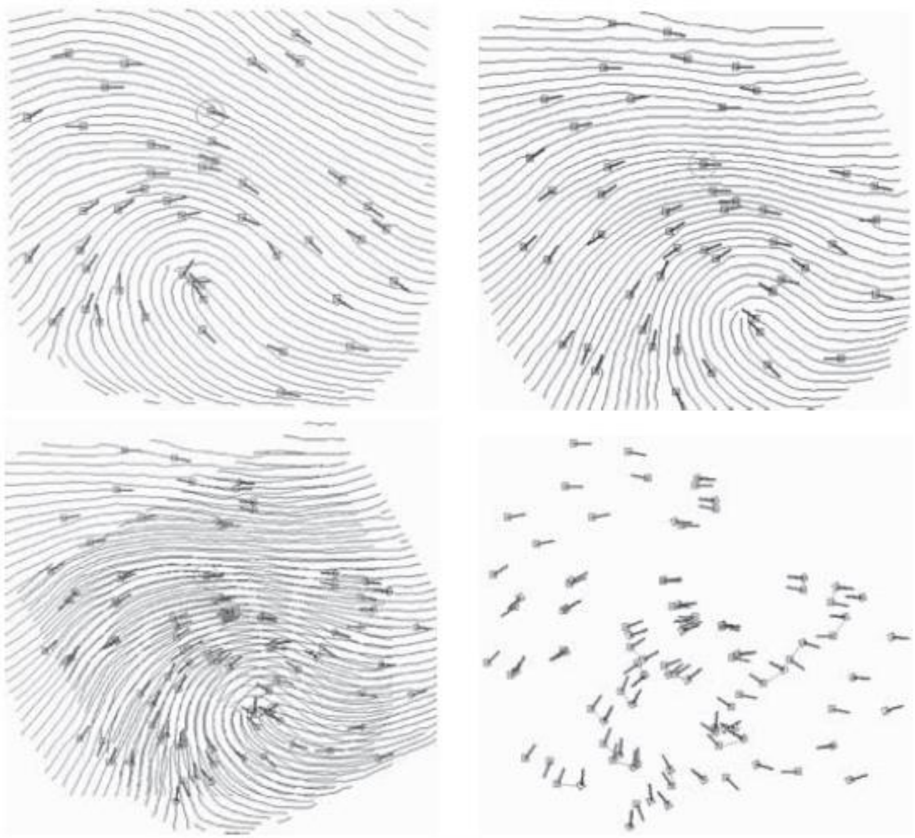
Fig. 4. Ridge ending and ridge bifurcation.

A.K. Jain et al, "On-line fingerprint verification," PAMI, vol.19(4), 1997

Minutiae

- A **minutia** is a point on fingerprint ridge(s) that has a certain characteristic.
  - **End point**
  - **Bifurcation**
- The distribution pattern of minutiae is one of features often used for fingerprint recognition.

Minutia matching



A.K. Jain et al, “On-line fingerprint verification,” PAMI, vol.19(4), 1997

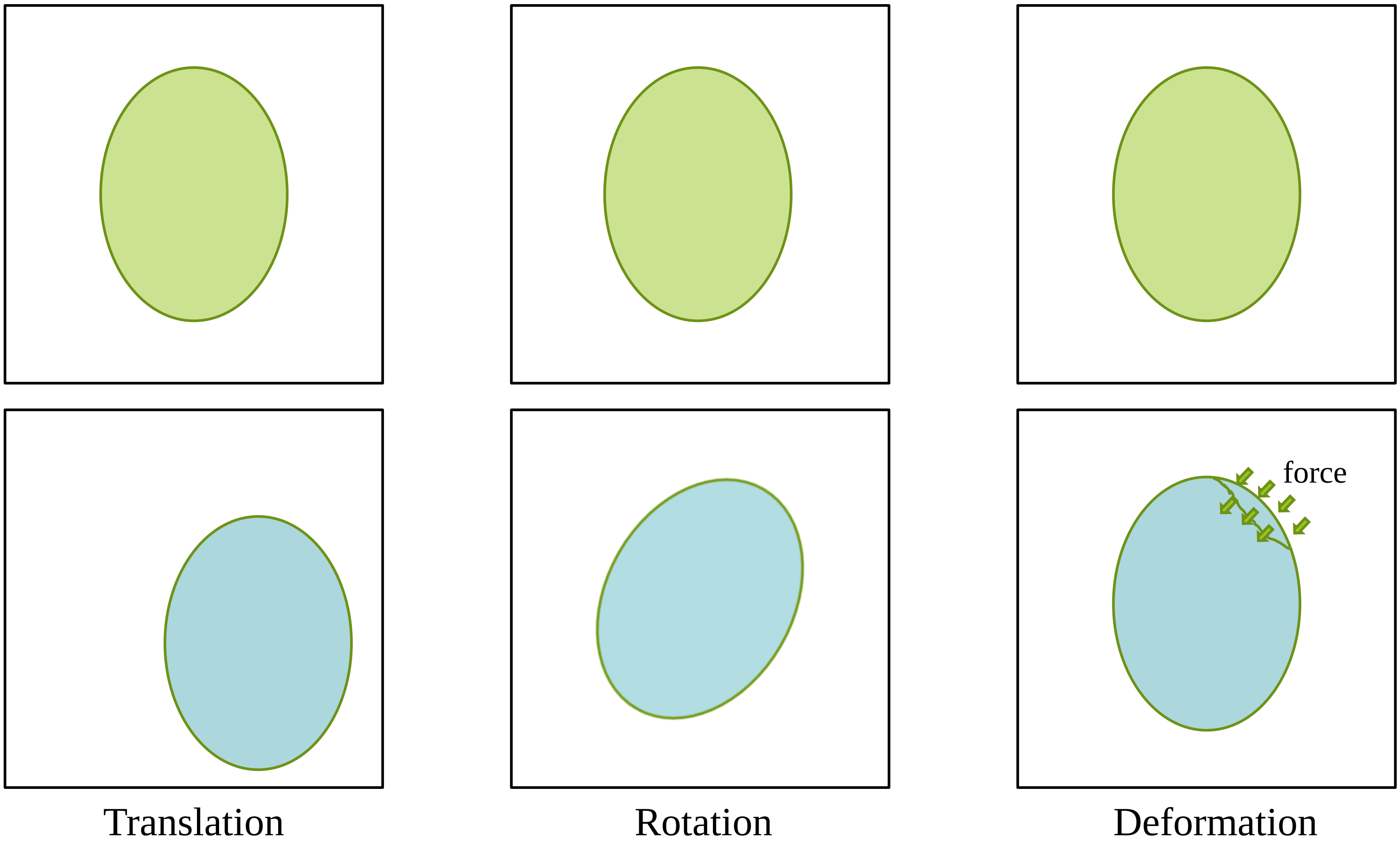
- By representing a set of minutiae as a point pattern, a fingerprint recognition problem can be reduced to a **point pattern matching** problem.
  - If two fingerprints are from the same finger, the point patterns formed by minutiae would be the same.
  - Their minutiae details (features around each minutiae) would match each other topologically.

Minutia matching

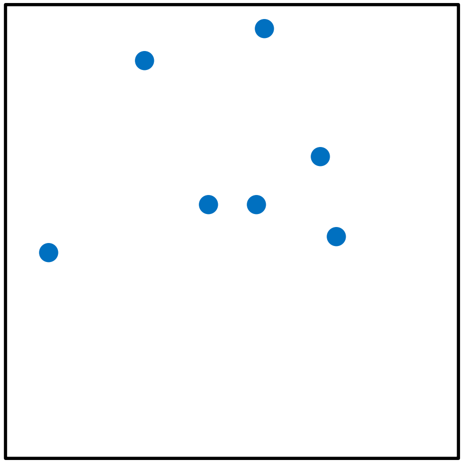
- The difficulties in point pattern matching for fingerprint recognition are caused by the followings:
  - The correspondences between the template and input fingerprint are not known beforehand.
    - There are relative translation, rotation, and non-linear deformations between template minutiae and input minutiae.
  - Some minutiae are missed from both templates and inputs.
  - Spurious minutiae normally present in both templates and inputs

- Minutia matching is an elastic matching of point patterns without knowing their correspondence beforehand.

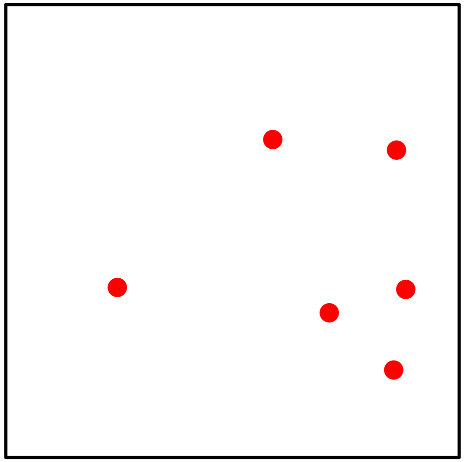
Common problems
in fingerprint recognition



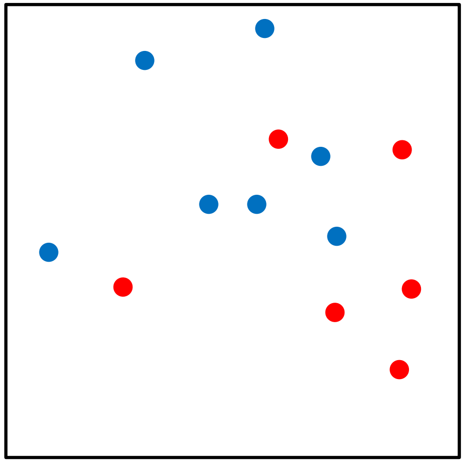
Point pattern matching



P



Q



Matching P and Q

- Let $P = \left\{ \mathbf{m}_1^P, \mathbf{m}_2^P, \mathbf{m}_3^P, \ldots, \mathbf{m}_M^P \right\}$ denote a set of M minutiae in the template and

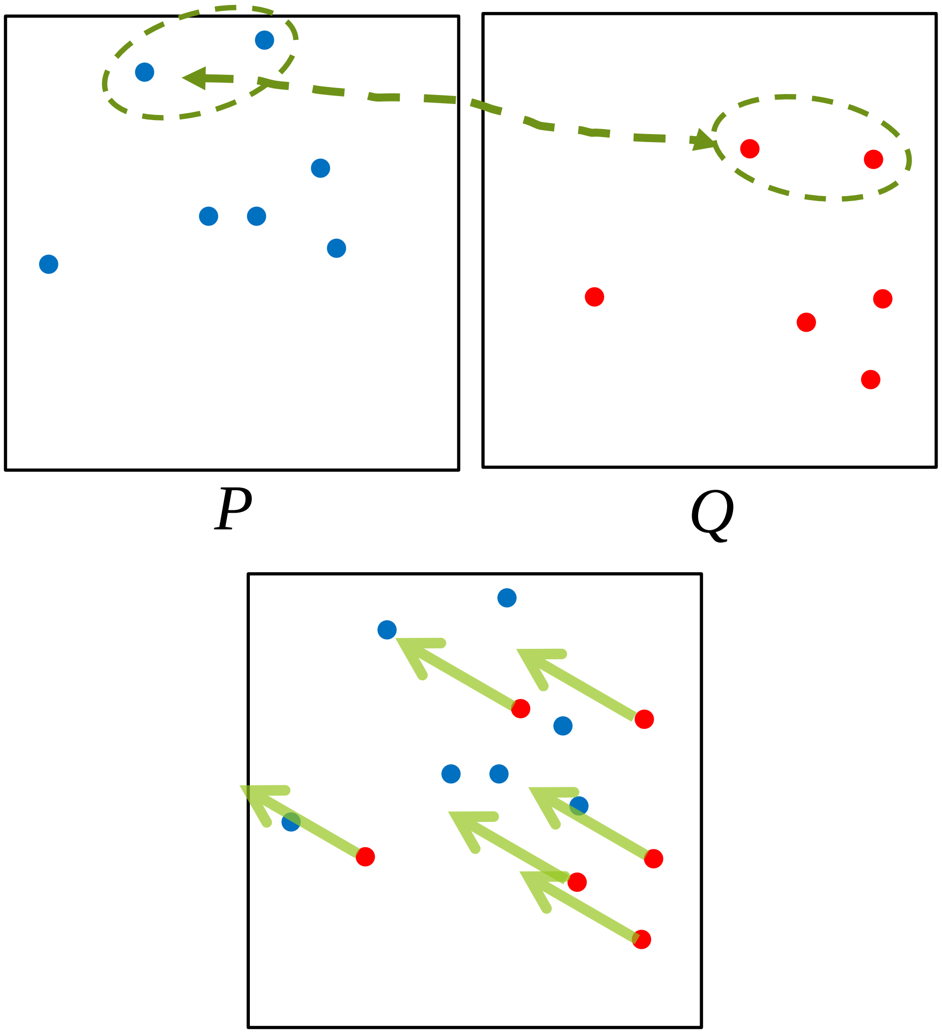
$$Q = \left\{ \mathbf{m}_1^Q, \mathbf{m}_2^Q, \mathbf{m}_3^Q, \ldots, \mathbf{m}_N^Q \right\}$$

denote a set of N minutiae in the input, where $\mathbf{m}_i$ is a vector describing a minutia point

  - For example, it may contain the position of the minutia, its type (end point or bifurcation), and its direction.

$$\mathbf{m}_i = \left(x_i, y_i, t_i, \theta_i \right)^T$$

Point pattern matching



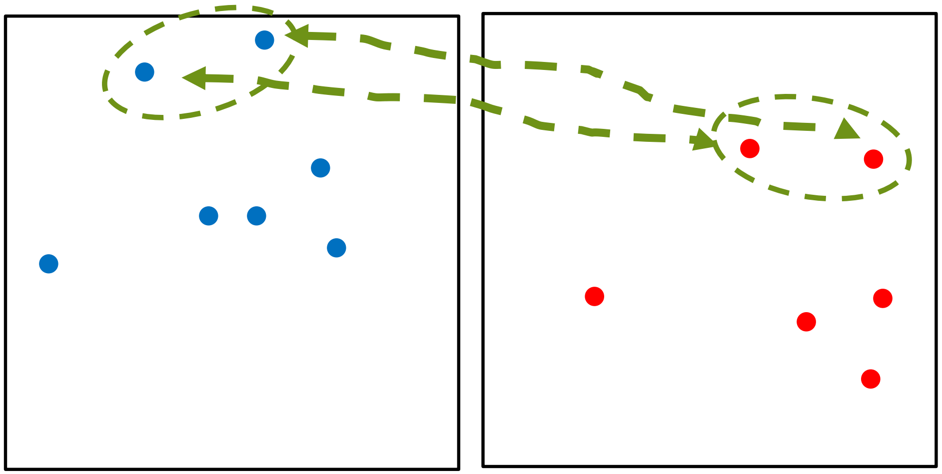
Matching P and Q

- A simple point pattern matching can be described as follows:

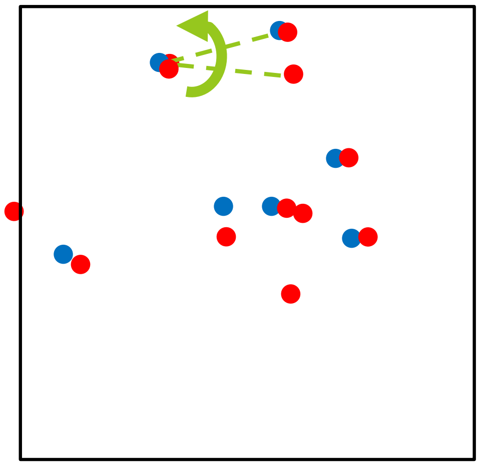
1. Choose a pair of minutiae in the template ($\mathbf{m}_i{}^P$, $\mathbf{m}_j{}^P$) and a pair of minutiae in the input ($\mathbf{m}_k{}^Q$, $\mathbf{m}_l{}^Q$)
2. Suppose that $\mathbf{m}_i{}^P$ corresponds to $\mathbf{m}_k{}^Q$, then the translation vector between the two fingerprints is computed by

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} x_i^P \\ y_i^P \end{bmatrix} - \begin{bmatrix} x_k^Q \\ y_k^Q \end{bmatrix}$$

Point pattern matching



P Q



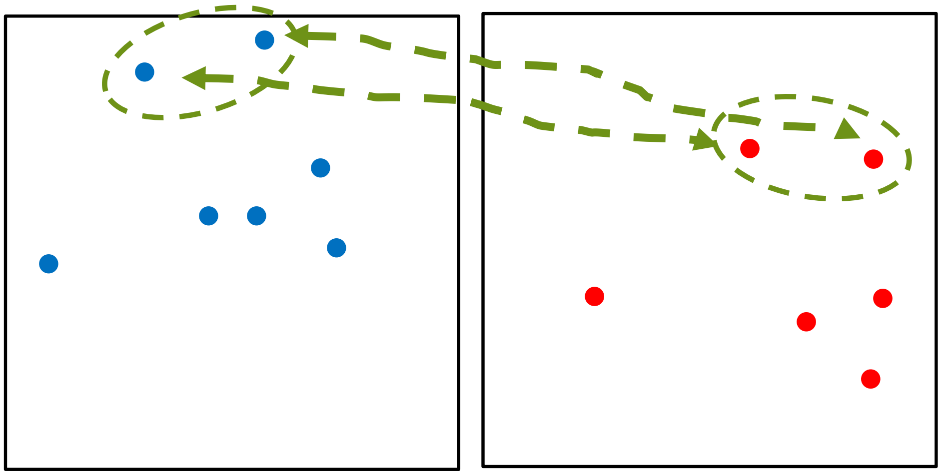
Matching P and Q

3. Suppose that $\mathbf{m}_j^P$ corresponds to $\mathbf{m}_l^Q$, then the rotation angle between the two fingerprints is computed by

$$\Delta\theta = \theta_{ij}^{P} - \theta_{kl}^{Q}$$

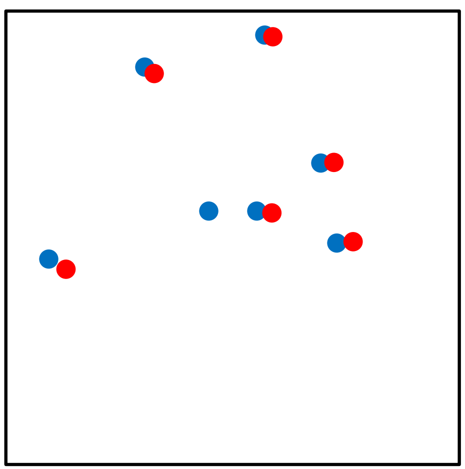
4. Translate and rotate all minutiae in the input by the translation vector and rotation angle from steps 2 and 3.

Point pattern matching



P *Q*

5. For each point in the template, find its correspondence in the input. Count the number of correspondences C_{ijkl}.



Matching *P* and *Q*

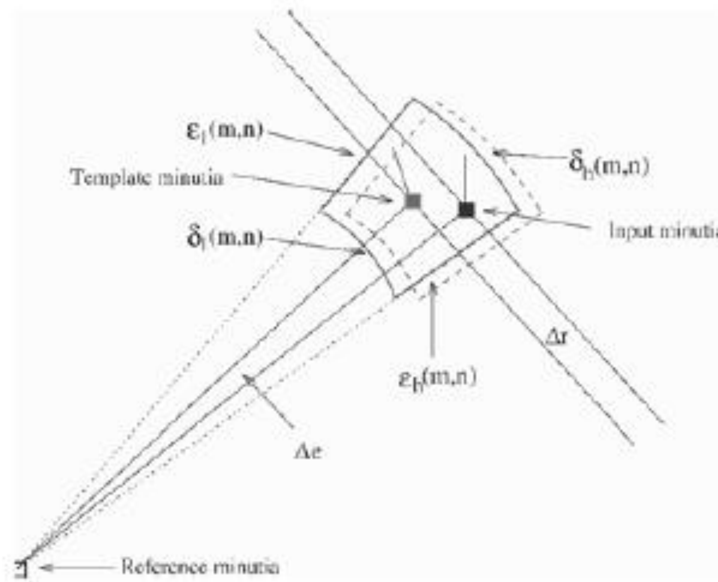
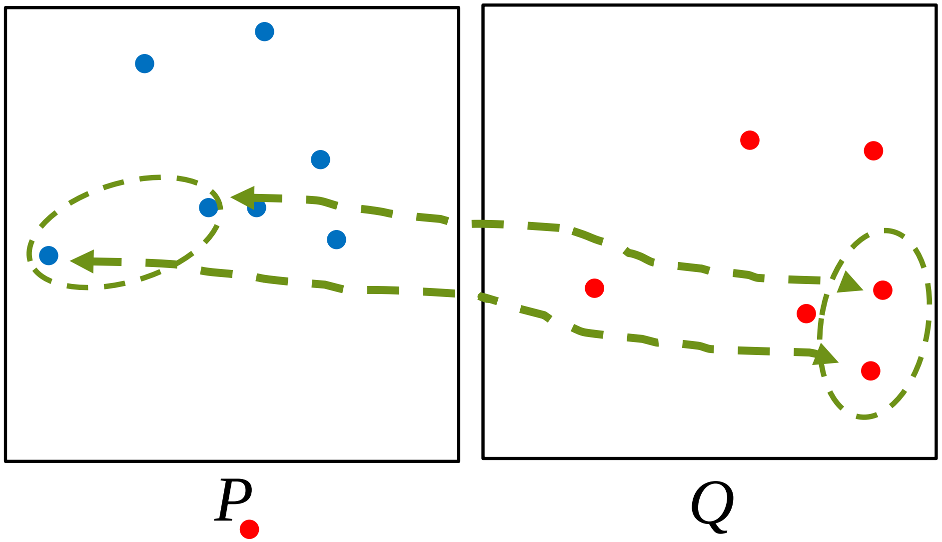
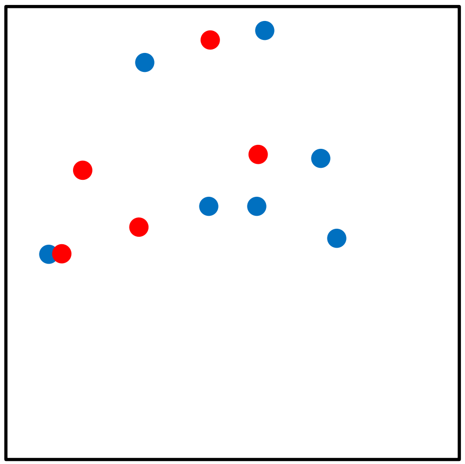


Fig. 11. Bounding box and its adjustment.

A.K. Jain et al, "On-line fingerprint verification," PAMI, vol.19(4), 1997

Point pattern matching





Matching P and Q

6. Repeat the process until each pair in the template has been matched with each pair in the input. Find the maximum correspondences C_{max}.
7. Compute the similarity S between the two fingerprints as follows:

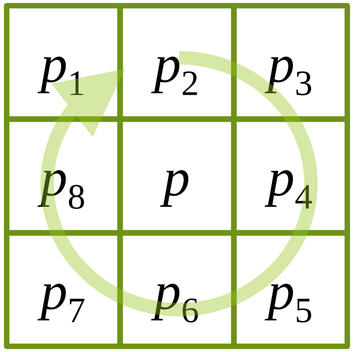
$$S = \frac{C_{\max}}{\max(M, N)}$$

- If S is larger than a threshold value T, they will be considered as fingerprints from the same finger (match); otherwise, different fingers (non-match).

How to detect minutiae?

How to detect minutiae?

Minutiae can be easily detected from a skeleton image



Minutia detection

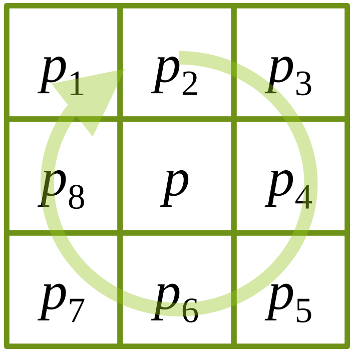
- Pixels corresponding to minutiae are characterized by a **crossing number** different from 2.
- The crossing number $cn(p)$ of a pixel p in a skeleton image is defined as half the sum of the differences between pairs of adjacent pixels in the 8-neighborhood of p.

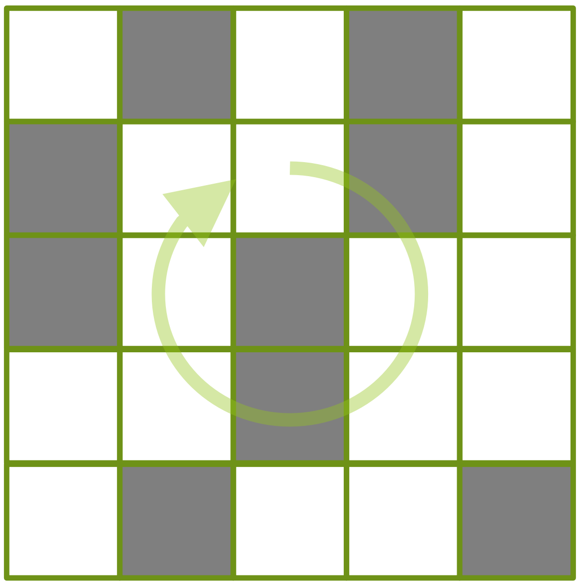
$$cn(p) = \frac{1}{2} \sum_{i=1..8} \left| val(p_i) - val(p_{i+1}) \right|$$

  - $p_1, p_2, \ldots, p_8$ are the pixels belonging to an ordered sequence of pixels defining the 8-neighborhood of p
  - $p_9 = p_1$
  - $val(p) \in \{0,1\}$.

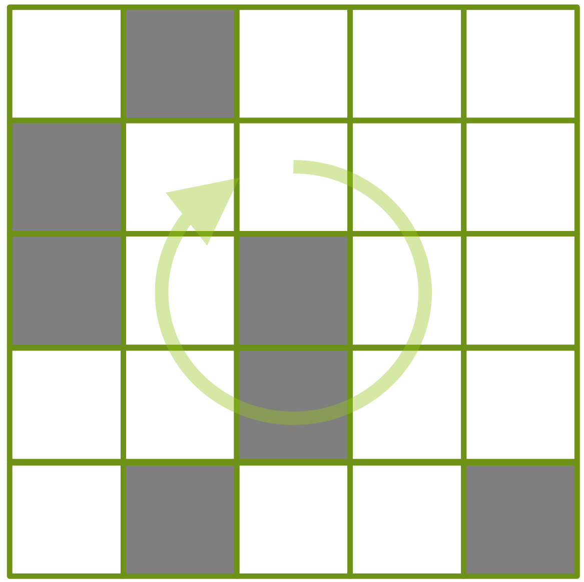
D. Maltoni et al., Handbook of Fingerprint Recognition, Springer, 2003

Minutia detection

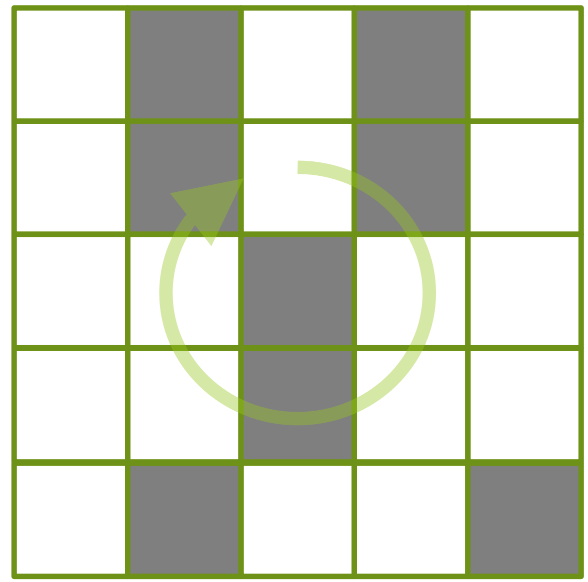




$cn(p) = 2$



$cn(p) = 1$

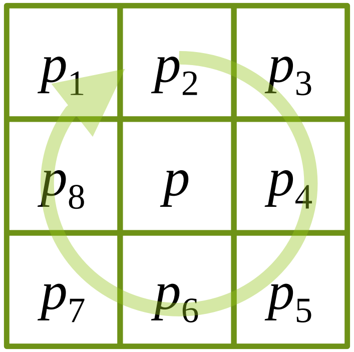


$cn(p) = 3$

How to obtain a skeleton image?

- **Thinning** or **skeletonization** is a process that reduces the thickness of ridges to one pixel.

- An image obtained from this process is called a **skeleton image**.



Thinning

- Thinning algorithm can be described as follows:
  1. For each black pixel in the image (except the borders), check whether the pixel satisfies all of the following conditions:
     - The number of black neighboring pixels $\in [2, 6]$.
     - The number of transitions from white to black $= 1$.
     - p_2, p_4, or p_6 is white.
     - p_4, p_6, or p_8 is white.
  2. Change the gray intensity of all pixels satisfying the conditions to white.
  3. Repeat the process until there is no pixel satisfying the conditions. In each iteration, the last two conditions may be switched to the followings:
     - p_2, p_4, or p_8 is white.
     - p_2, p_6, or p_8 is white.

Binarization

- Before a thinning process can be done, a binary fingerprint image is required.
- A simple global thresholding can be used to convert a fingerprint image into a binary image.

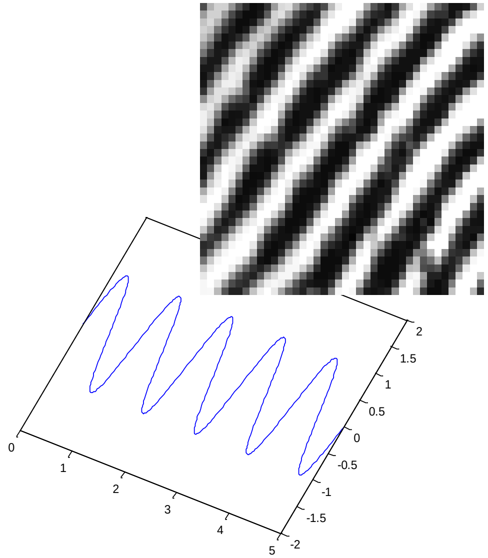
$$g(x, y) = \begin{cases} 255 & f(x, y) > th \\ 0 & \text{otherwise} \end{cases}$$

- A threshold selection method such as the iterative algorithm or Otsu's threshold selection can be used to obtain an appropriate threshold value.

How to more accurately detect minutiae?

Fingerprint enhancement

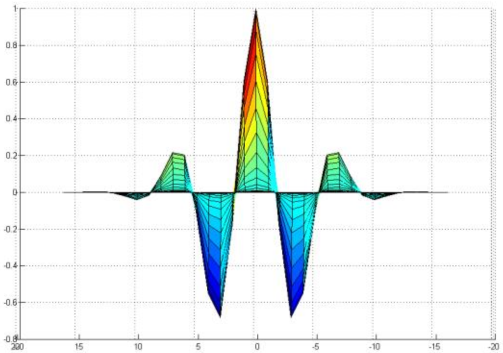
- The quality of fingerprint image strongly affects minutiae extraction algorithms.
- The goal of **fingerprint enhancement** process is to improve the clarity between ridges and valleys.
  - Resulting in a more reliable extraction of minutiae.
- **Gabor filter** is often used to enhance fingerprint images.
  - Local characteristics of fingerprint images:
    - One fingerprint direction
    - Constant frequency
    - Can be approximated as a sinusoidal signal

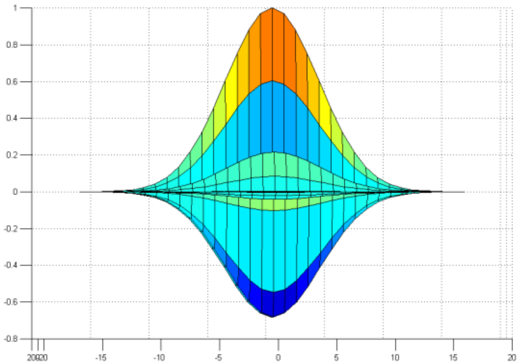


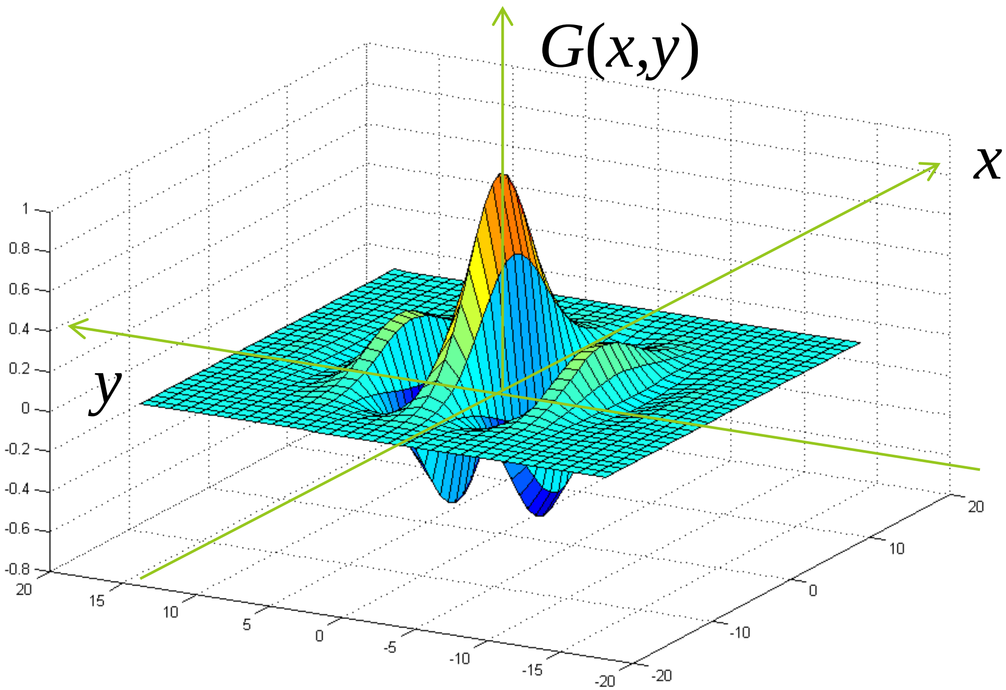
2D Gabor filter

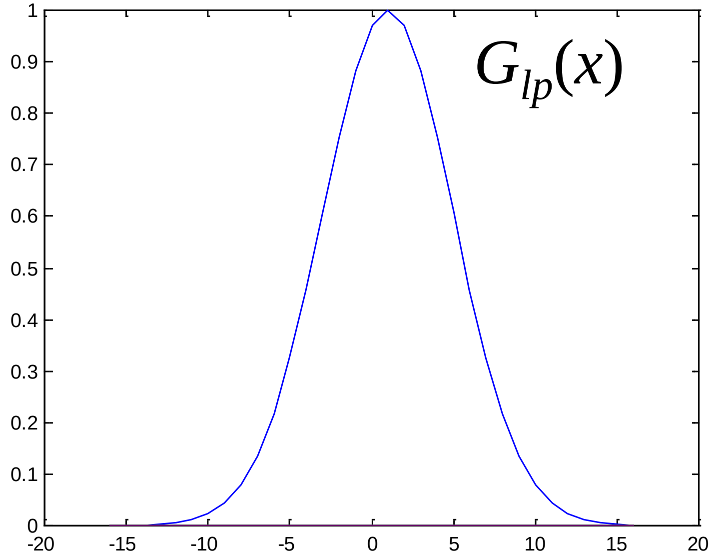
- 2D Gabor filter (with orientation of 0°):

$$G(x,y) = \exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)\right] \times \cos(2\pi f y)$$









2D Gabor filter

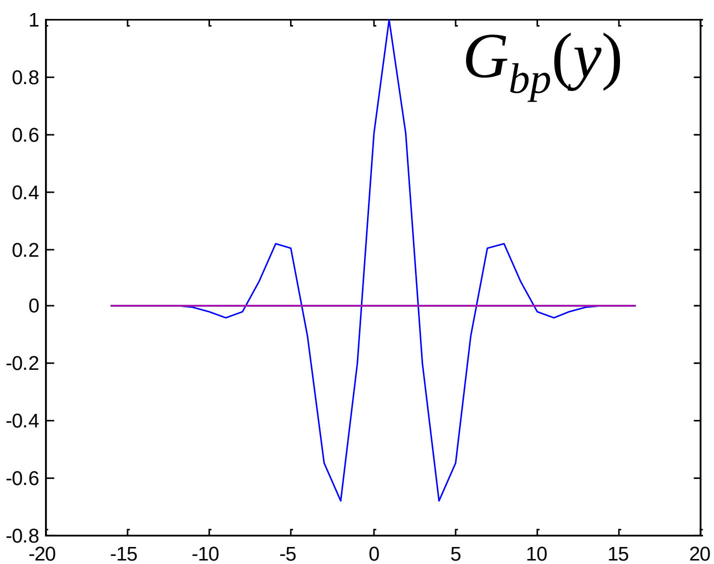
- 2D Gabor filter (with orientation of 0° or 90°) can be decomposed into two 1D-filters:

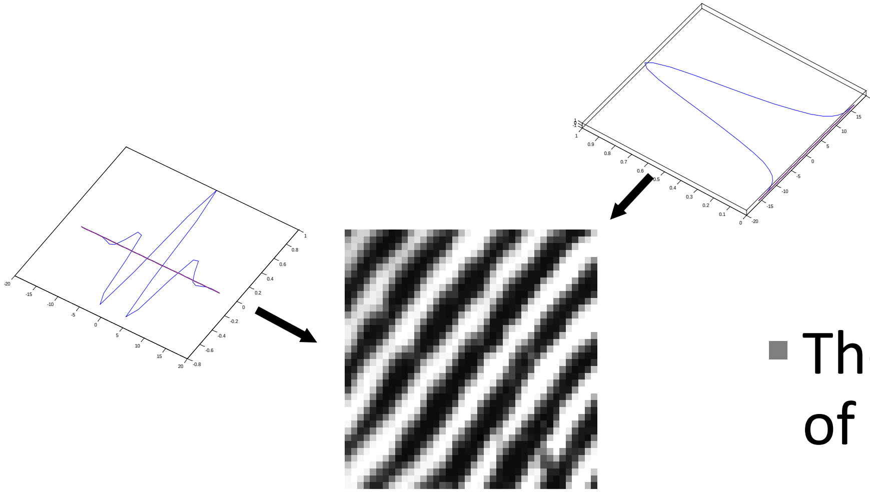
$$G(x,y) = \exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)\right]\times\cos(2\pi f y)$$

$$= G_{lp}(x)G_{bp}(y)$$

$$G_{lp}(x) = \exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}\right)\right]$$

$$G_{bp}(y) = \exp\left[-\frac{1}{2}\left(\frac{y^2}{\sigma_y^2}\right)\right]\times\cos(2\pi f y)$$



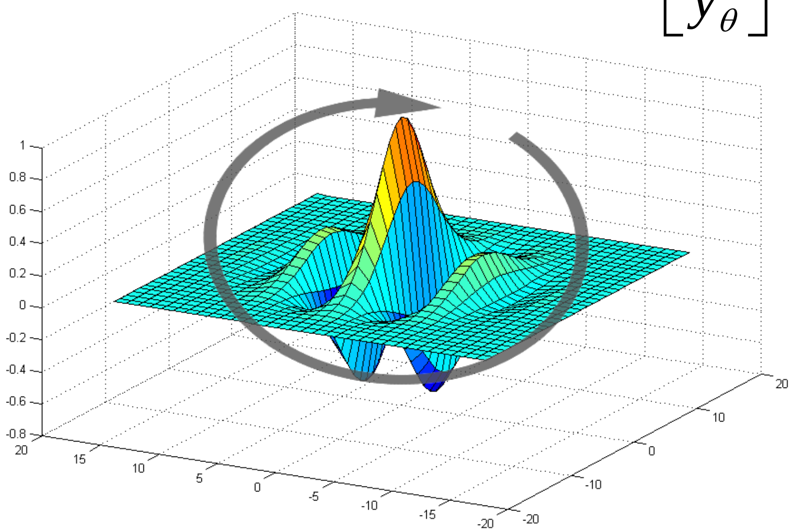


Rotating a Gabor filter

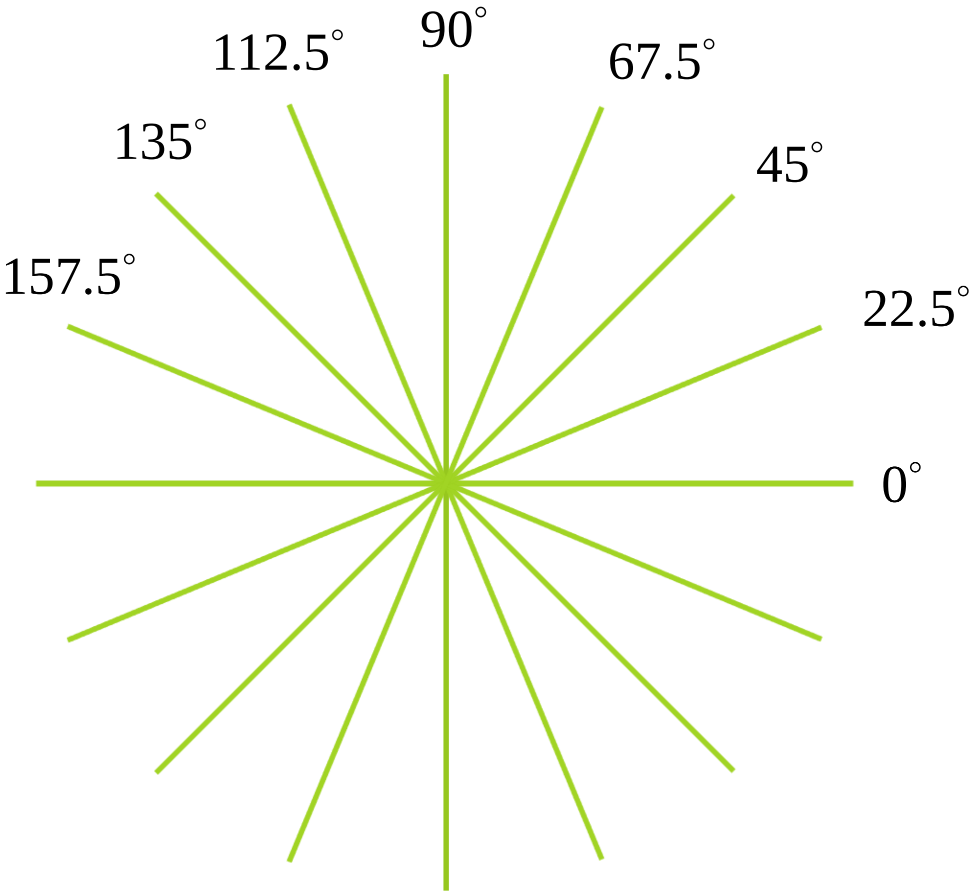
- The following is a general formula of Gabor filter with orientation θ:

$$G(x, y : \theta) = \exp\left[-\frac{1}{2}\left(\frac{x_\theta^2}{\sigma_x^2} + \frac{y_\theta^2}{\sigma_y^2}\right)\right] \times \cos(2\pi f y_\theta)$$

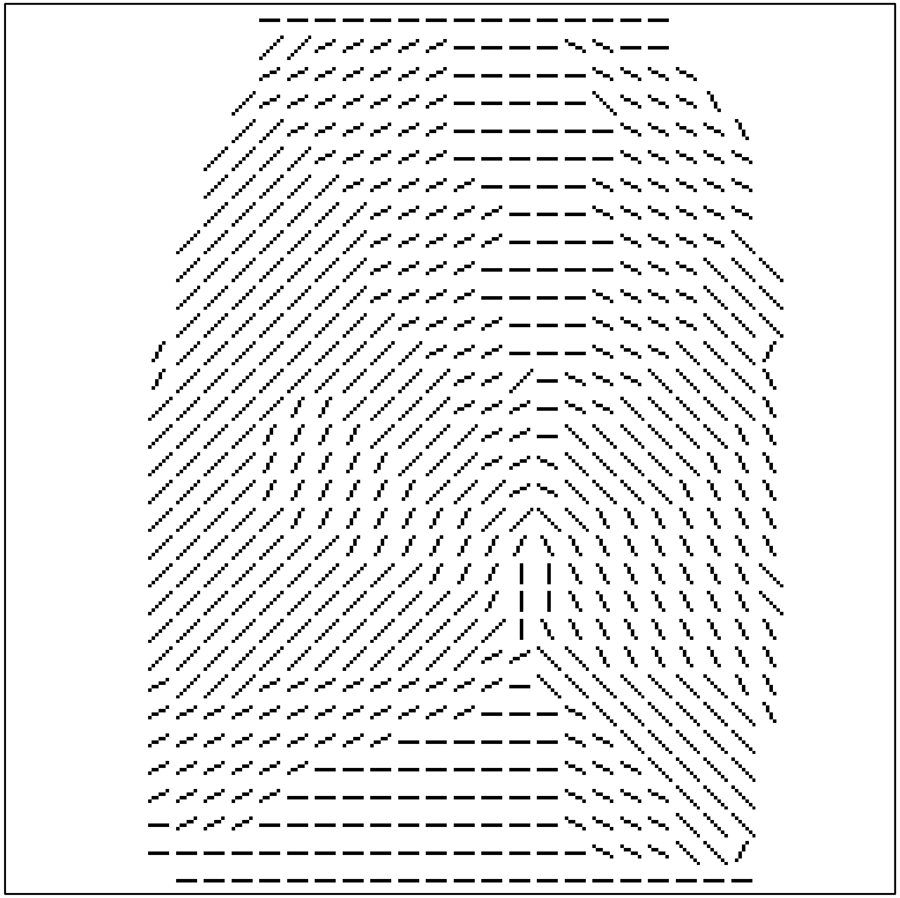
$$\begin{bmatrix} x_\theta \\ y_\theta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



Applying Gabor filters



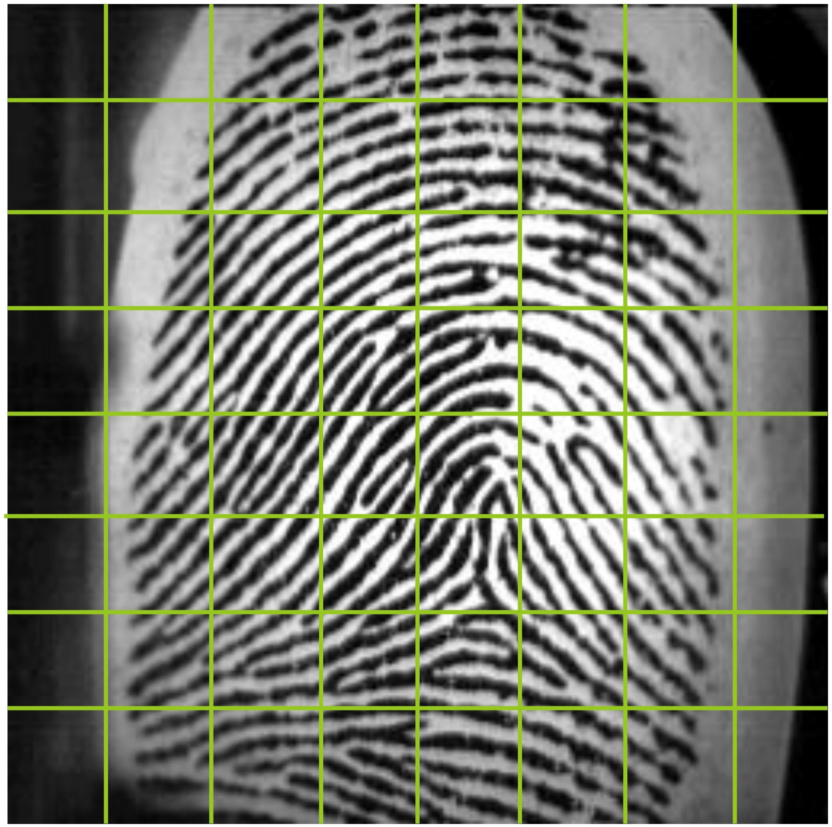
- Generate a bank of Gabor filters in different orientations
  - 8 orientations are normally used
- Compute the **orientation field (OF)**, which describes the local ridge orientation, for each block and quantize it into 8 orientations
- For each block, apply a Gabor filter corresponding the quantized orientation the block



How to estimate local ridge orientation?

A.K. Jain et al, “On-line fingerprint verification,” PAMI, vol.19(4), 1997

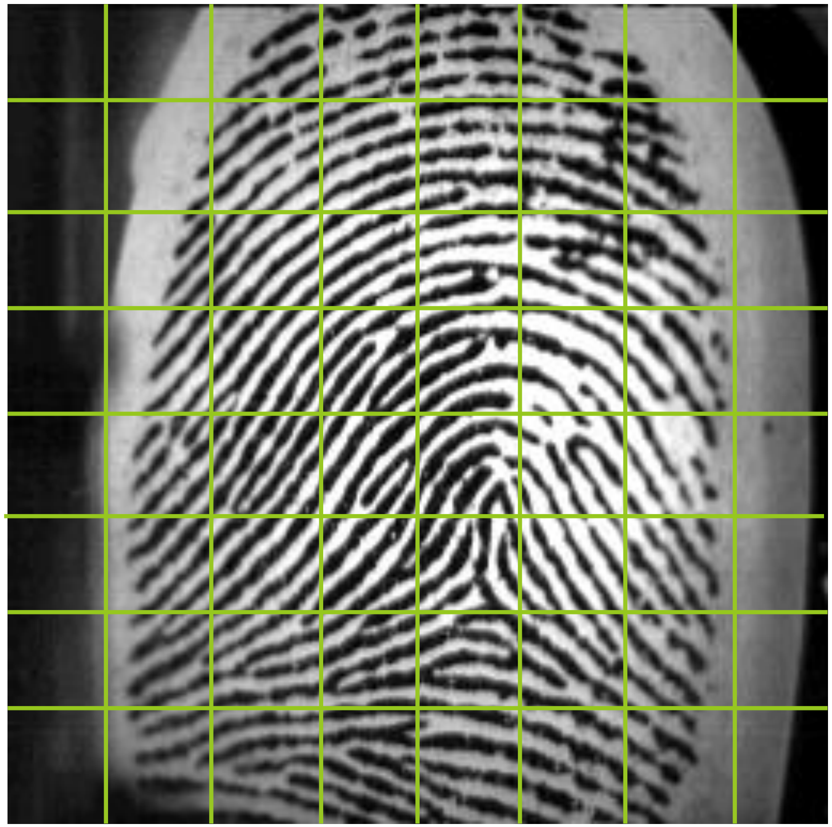
Block-wise ridge orientation estimation (method 1)



- Divide an input fingerprint image into blocks of size $N \times N$ pixels.
- Compute the gradient magnitude g_x and g_y for each pixel in a block.
- Estimate the local ridge orientation of a block using the following formula:

$$\theta = \frac{1}{2} tan^{-1}\left(\frac{\sum_{i=1..N} \sum_{j=1..N} 2 g_x(i,j) g_y(i,j)}{\sum_{i=1..N} \sum_{j=1..N} \left(g_x^2(i,j) - g_y^2(i,j) \right)} \right)$$

Block-wise ridge orientation estimation (method 2)



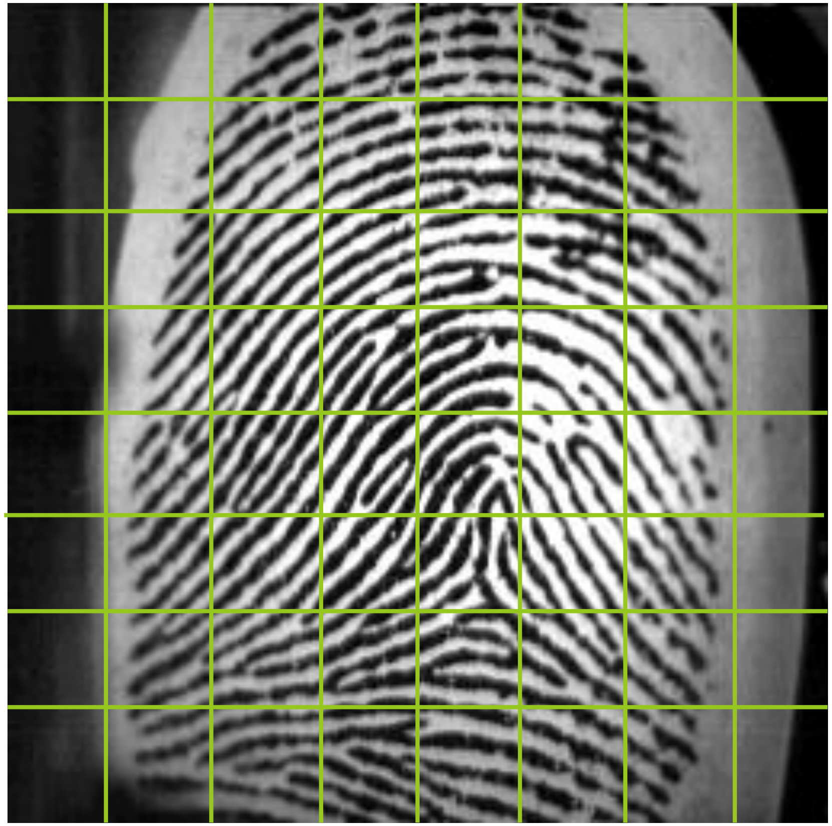
- Divide an input fingerprint image into blocks of size $N \times N$ pixels.
- Perform DFT to each block
- Detect two strongest peaks (except the center) in the block
- The local ridge orientation can be estimated as the direction perpendicular to a line segment passing the two peaks.

How to increase the speed of computation?

Fingerprint segmentation

- **Fingerprint segmentation** is process that separates the area of fingerprint from background.
- This makes us be able to avoid further processing applied on non-fingerprint areas.
- This also helps reduce the number of spurious minutiae that might appear in non-fingerprint areas.

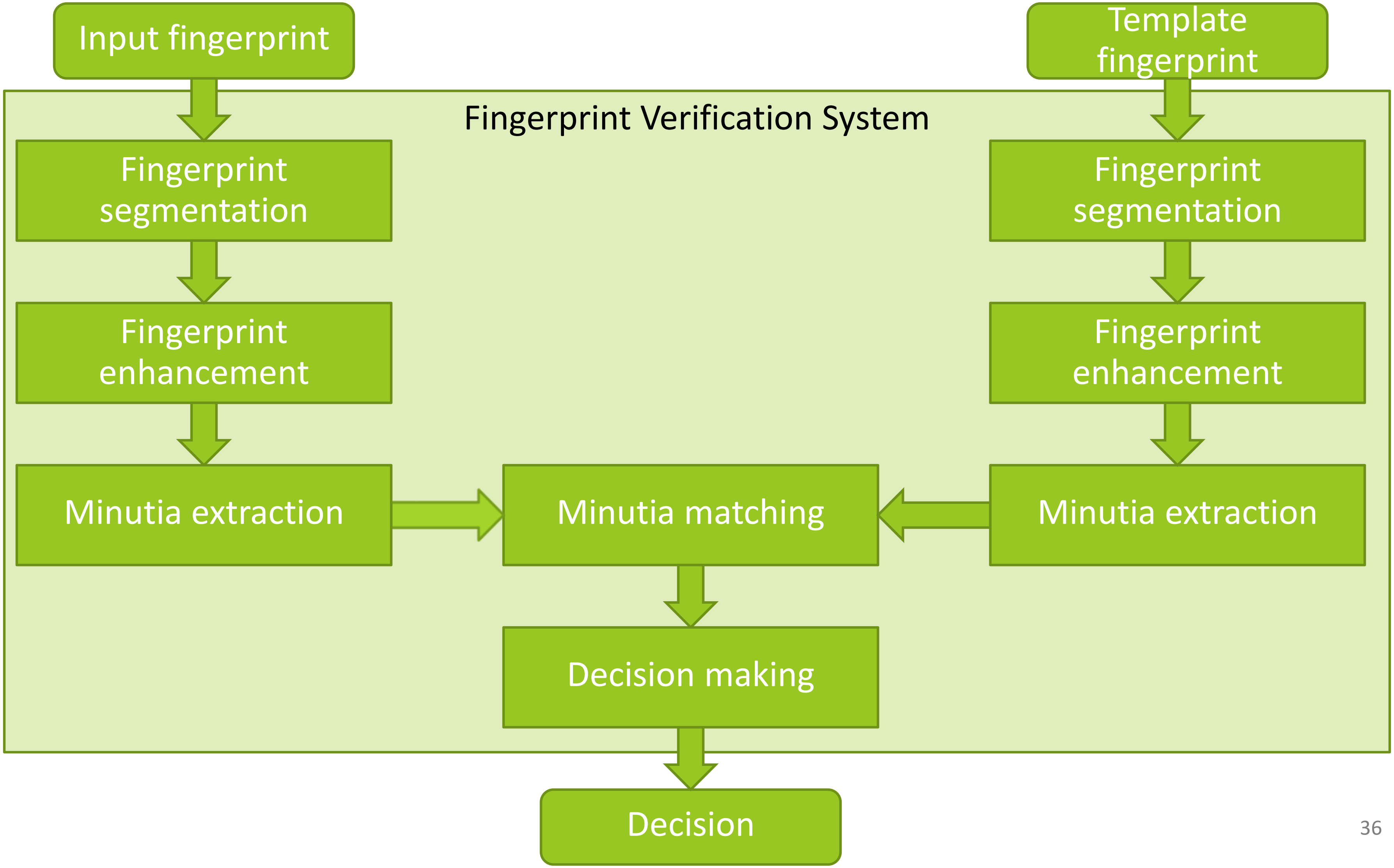
Fingerprint segmentation



- A simple algorithm for fingerprint segmentation consists of three steps:
  - Divide an input fingerprint image into blocks of size $N \times N$ pixels.
  - Compute some statistical features, e.g., the average and variance of gray intensity in each block
  - Perform decision (thresholding) based on the computed features

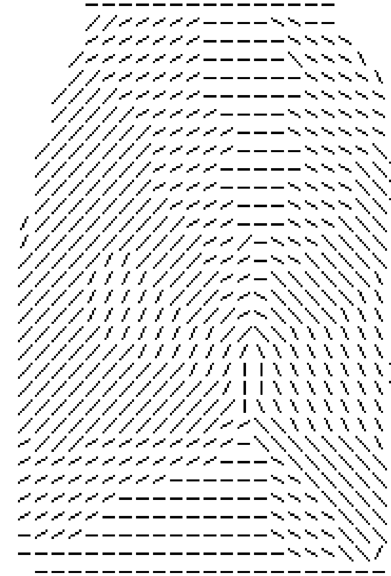
Summary of fingerprint verification algorithm

Fingerprint verification system



Input fingerprint image

Segmented image

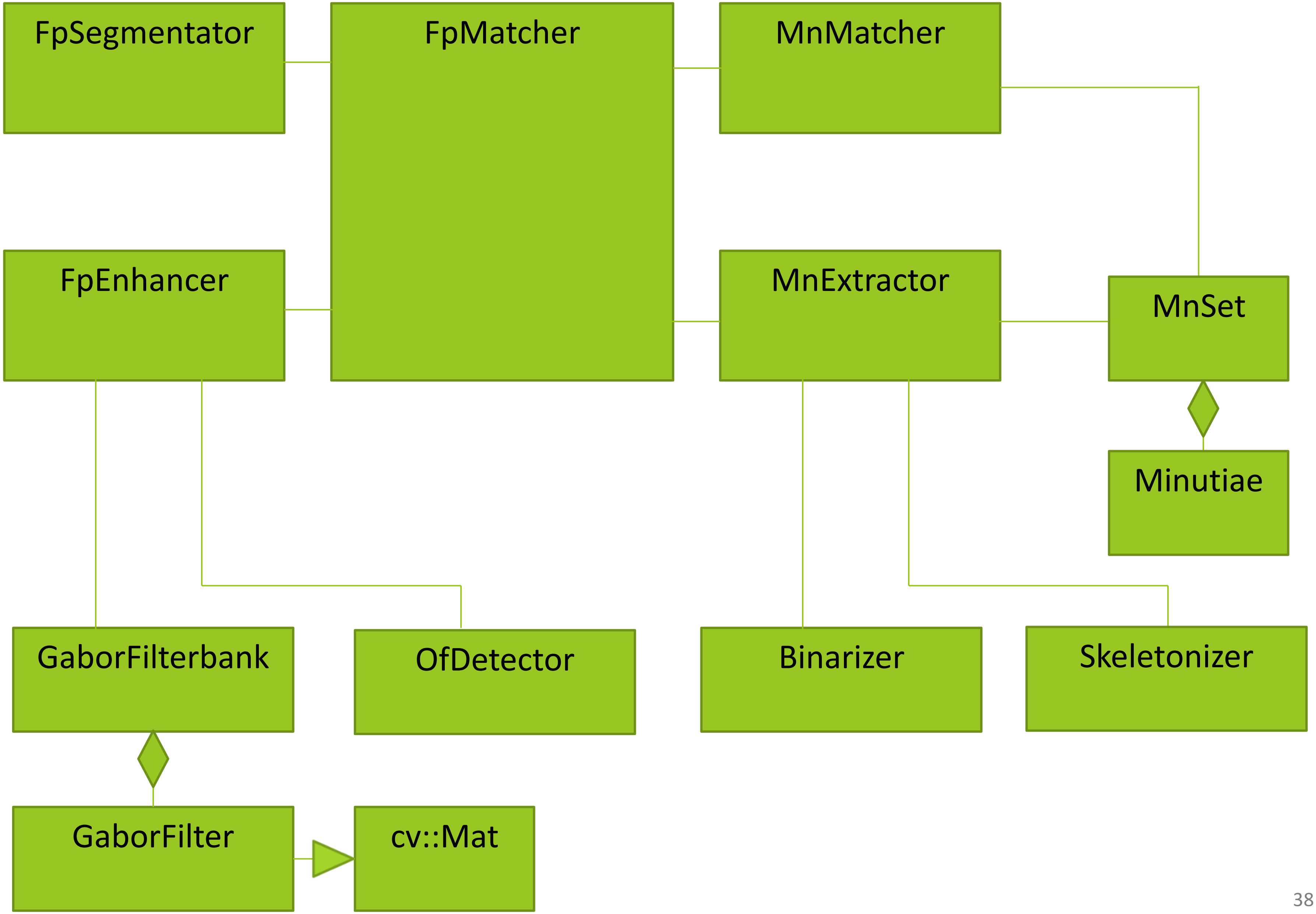


Orientation field

Filtered by Gabor filters

Binarized image

Skeleton image

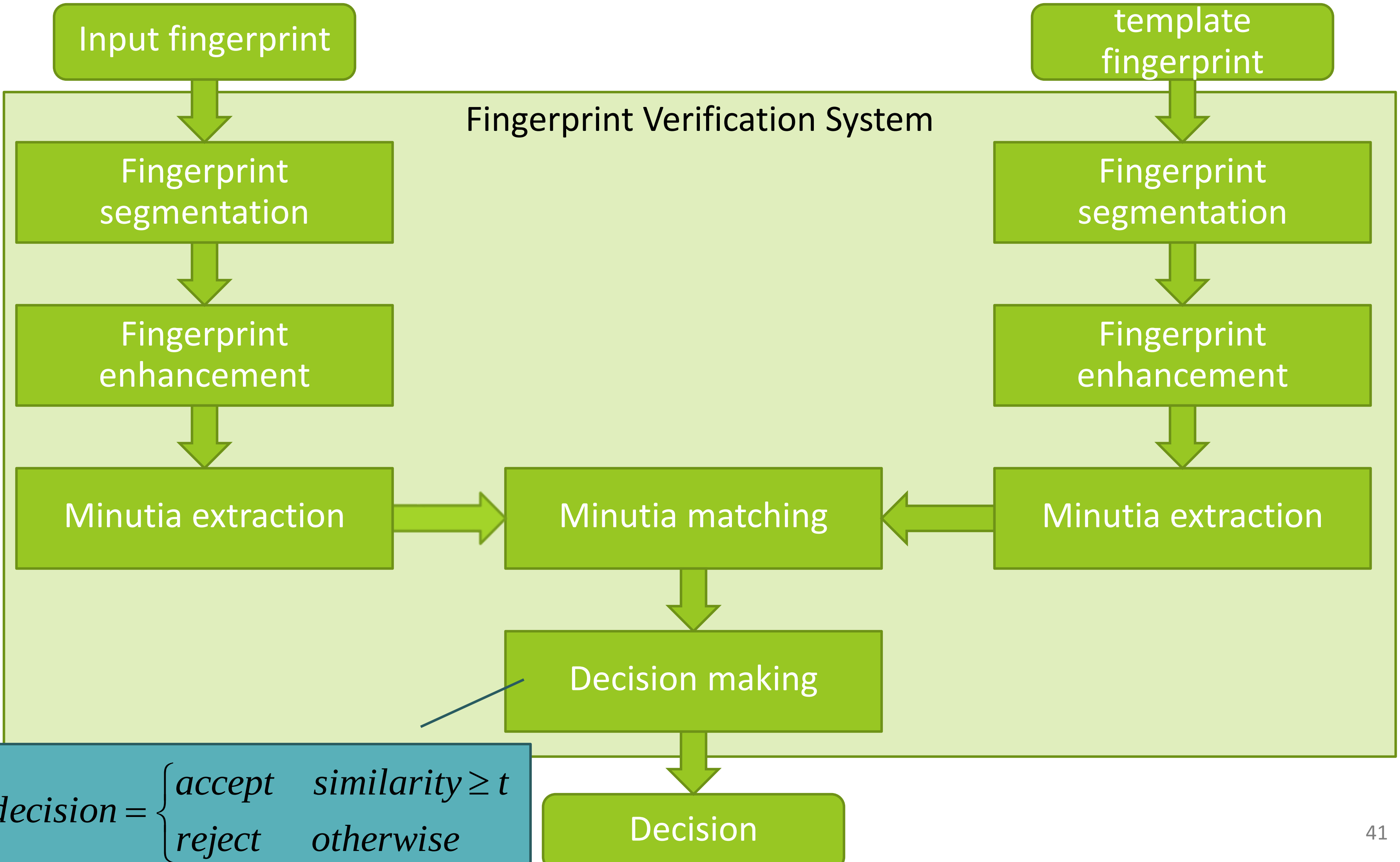

FpSegmentator
FpMatcher
MnMatcher
FpEnhancer
MnExtractor
MnSet
Minutiae
GaborFilterbank
OfDetector
Binarizer
Skeletonizer
GaborFilter
cv::Mat

```
import cv2
import numpy as np
import FpMatcher

fpMatcher = FpMatcher.FpMatcher()
fpEvl = FpMatchingEvaluator()
fpEvl.evaluate("../FP DB1 (test subset)/DB_Info.txt", fpMatcher)
```

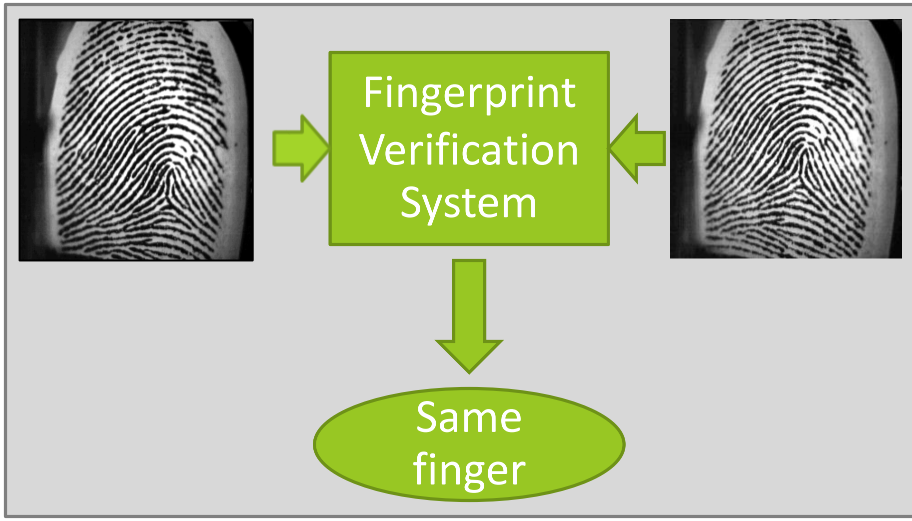
Performance evaluation

Fingerprint verification system

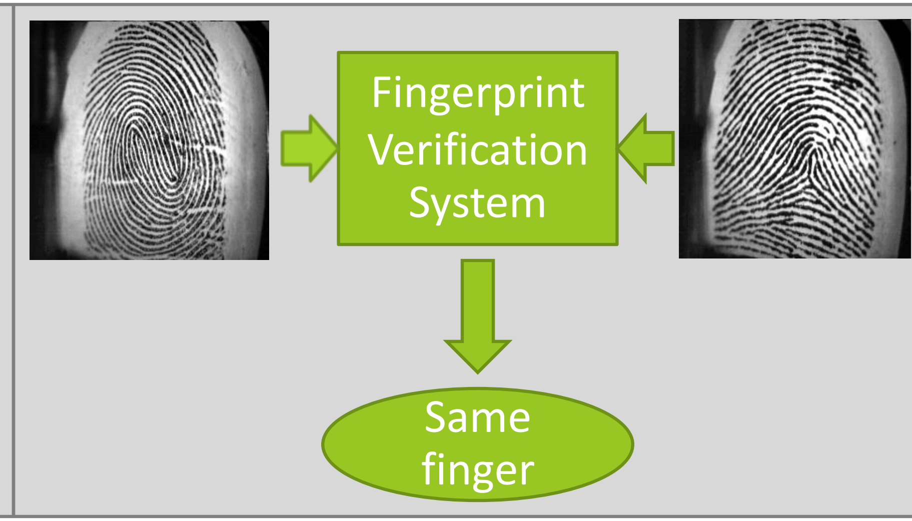


$$decision = \begin{cases} accept & similarity \geq t \\ reject & otherwise \end{cases}$$

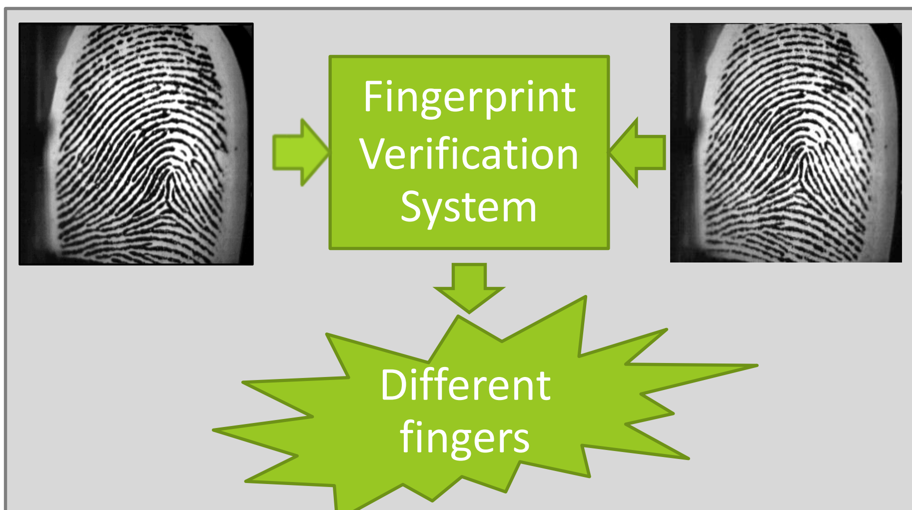
Genuine matching attempt **Imposter matching attempt**



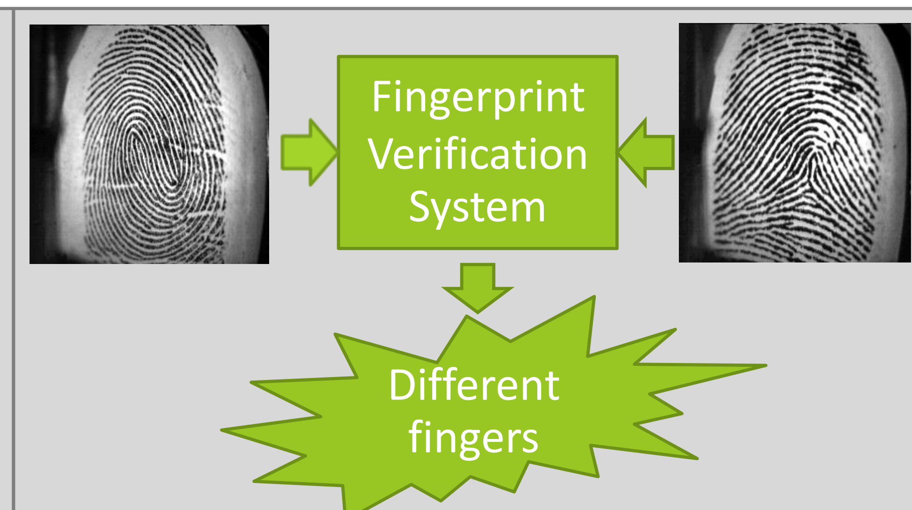
True acceptance (TA)



False acceptance (FA)



False rejection (FR)



True rejection (TR)

How to evaluate a fingerprint verification system?

- Given a set of fingerprint images
  - Suppose there are M fingers, N images per finger.
  - Use two index values to represent an image
    - E.g., 4_1.bmp means this is 1st image of 4th finger.

- Perform matching between all pairs of fingerprint images in the set and calculate their similarity (0-100)

- For each threshold value t $(0 \leq t \leq 100)$
  - Perform decision making
  - Find the total numbers of true/false acceptance/rejection
    - Denoted by N_{TA}, N_{TR}, N_{FA}, N_{FR}.
    - These are functions of threshold value t.
  - Find the total numbers of genuine/imposter matching attempts
    - Denoted by N_G and N_I

How to evaluate a fingerprint verification system?

- False acceptance rate $FAR(t)$
  - Also known as false match rate (FMR)

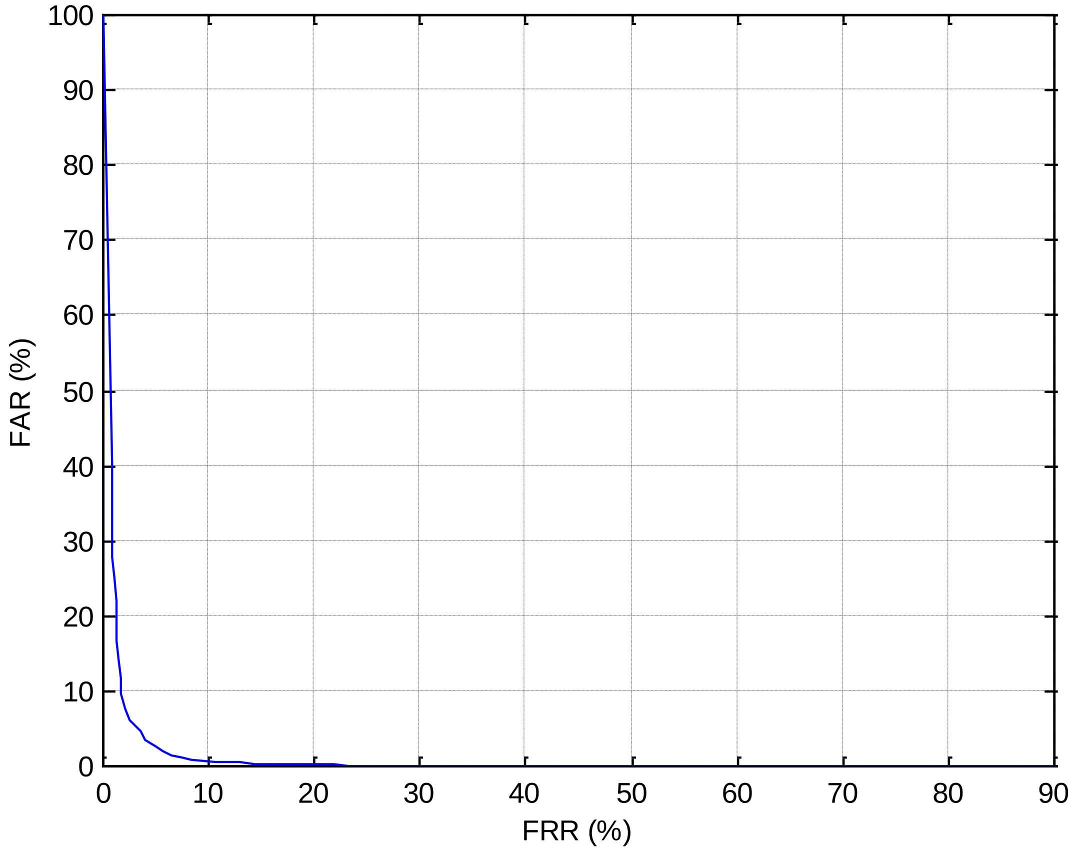
$$FAR(t) = \frac{N_{FA}(t)}{N_I}$$

- False rejection rate $FRR(t)$
  - Also known as false non-match rate (FNMR)

$$FRR(t) = \frac{N_{FR}(t)}{N_G}$$

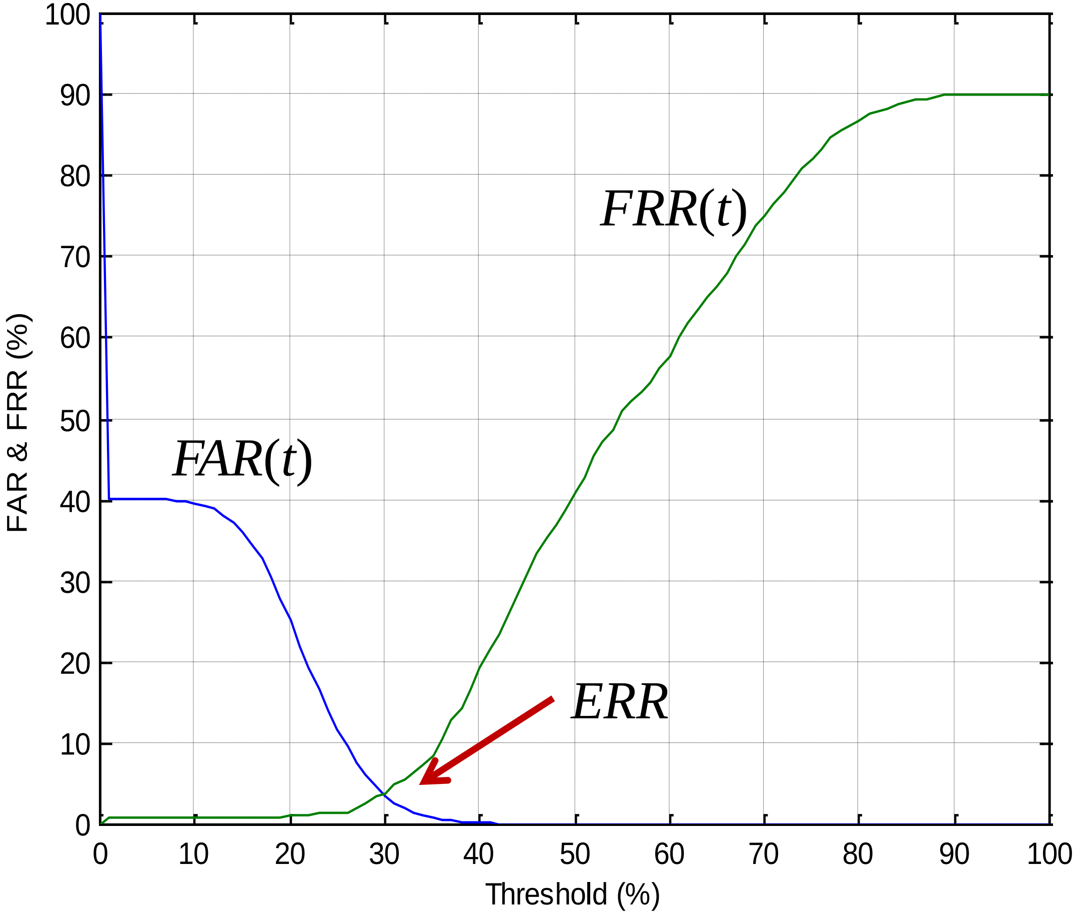
  - These two functions depend on the threshold value t.
    - It controls the security level of the system.
    - Normally, when t increases, $FRR(t)$ increases but $FAR(t)$ decreases.

ROC curve



- Receiver operating characteristics (ROC) curve is the plot of $FAR(t)$ against $FRR(t)$ (or sometimes against $1-FRR(t)$).
- The closer to the origin, the better performance

Equal error rate



- Equal error rate (EER) is the error rate at the threshold value t in which $FAR(t) = FRR(t)$.
- In practical, this might not exist because of the quantization of threshold value t.
  - It might be approximated or described as a range instead.

Other performance indicators

- FAR_{100} – the lowest FRR for $FAR \leq 1\%$ (1/100)
- FAR_{1000} – the lowest FRR for $FAR \leq 0.1\%$ (1/1000)
- FAR_{10000} – the lowest FRR for $FAR \leq 0.01\%$ (1/10000)
- $Zero_{FAR}$ – the lowest FRR at which no false acceptance occurs ($FAR = 0\%$)
- $Zero_{FRR}$ – the lowest FAR at which no false rejection occurs ($FRR = 0\%$)

https://biolab.csr.unibo.it/fvcongoing/UI/Form/BenchmarkAreas/BenchmarkAreaFV.aspx

References

References

- A. K. Jain, L. Hong, and R. Bolle, "On-line fingerprint verification," IEEE Trans. PAMI, vol.19, no.4, pp.302-314, April 1997.
- L. Hong, Y. Wan, and A. K. Jain, "Fingerprint image enhancement: algorithm and performance evaluation," IEEE Trans. PAMI, vol.20, no.8, pp.777-789, August 1998.
- D. Maltoni et al., Handbook of Fingerprint Recognition, Springer, 2003.
- Biometric System Laboratory, University of Bologna
  (URL: http://biolab.csr.unibo.it/home.asp)
- The First Fingerprint Verification Competition (FVC2000)
  (URL: http://bias.csr.unibo.it/fvc2000/)
- The Second Fingerprint Verification Competition (FVC2002)
  (URL: http://bias.csr.unibo.it/fvc2002/)
- The Third Fingerprint Verification Competition (FVC2004)
  (URL: http://bias.csr.unibo.it/fvc2004/)
- The Forth Fingerprint Verification Competition (FVC2006)
  (URL: http://bias.csr.unibo.it/fvc2006/)
- FVC-onGoing: On-line evaluation of fingerprint recognition algorithms
  (URL: https://biolab.csr.unibo.it/FVCOnGoing/UI/Form/Home.aspx)