SIL775: Biometric Security
Assignment 1
Contact to Contactless Fingerprint Matching

Anand Sharma
[2024JCS2049]
Submission Date: 12-02-2025

---

1. Introduction

Overview

Traditional fingerprint recognition systems rely on contact-based sensors, which may introduce hygiene concerns, sensor wear, and inconsistencies due to skin deformations. Contactless fingerprint recognition offers a solution by capturing fingerprints without direct contact, reducing distortions while improving user convenience and security. However, due to variations in imaging conditions, direct matching of contactless and contact-based fingerprints is a challenging task.

This report explores the contact-to-contactless fingerprint matching process, covering essential preprocessing steps such as segmentation, enhancement, distortion correction, orientation field estimation, and minutiae extraction. The methodology follows a structured approach to ensure accurate fingerprint recognition, leveraging deep learning models, image processing techniques, and feature extraction algorithms.

Motivation

- The increasing adoption of contactless biometric systems necessitates effective methods to match fingerprints across different acquisition techniques.
- Contact-based sensors introduce hygiene concerns and distort fingerprints due to skin elasticity.
- Ensuring compatibility between contact-based and contactless fingerprints is essential for robust biometric authentication.

Objective

- Develop an efficient contactless fingerprint preprocessing pipeline to extract meaningful features.
- Implement segmentation, enhancement, distortion correction, and feature extraction techniques for improved recognition.
- Analyze the effectiveness of the proposed approach through visual and quantitative results.

Methodology Overview

The workflow follows a systematic C2CL (Contact to Contactless) approach, involving:

1. Segmentation: U-Net-based model to isolate the fingerprint region from the background.
2. Enhancement: Adaptive histogram equalization and inversion techniques for ridge enhancement.
3. Distortion Correction and Scaling: Spatial Transformer Networks (STN) to align fingerprint patterns.
4. Orientation Field Estimation: Gradient-based analysis to determine ridge flow directions.
5. Minutiae Extraction: Skeletonization and crossing number methods to identify unique ridge features.

---

2. Preprocessing of Fingerprints

2.1 Segmentation

- Introduction

  Fingerprint segmentation is the process of isolating the distal phalange (fingerprint region) from the background in contactless fingerprint images. This step is essential to improve feature extraction accuracy and eliminate unwanted noise.

A U-Net-based segmentation model is employed to generate a segmentation mask, which is used to:

1. Eliminate background noise from the input image.
2. Crop the fingerprint region to focus only on the relevant area.
3. Resize the segmented fingerprint to a standard resolution while maintaining aspect ratio.

- Methodology

The segmentation pipeline consists of the following steps:

1. U-Net Architecture for Fingerprint Segmentation

A U-Net deep learning model is trained for automatic fingerprint segmentation. The model consists of:

- Contracting Path (Encoder):
    - Extracts hierarchical features using multiple convolutional layers and max-pooling operations.
    - Downsamples the input to capture global context.
- Expanding Path (Decoder):
    - Uses transposed convolution layers to upsample the feature maps.
    - Concatenates encoder outputs via skip connections to retain spatial information.
    - Produces a segmentation mask where fingerprint regions are labeled as foreground (1) and background as 0.

2. Training Process

- The model is trained on a dataset of fingerprint images with corresponding segmentation masks.
- The loss function used is Binary Cross Entropy with Logits Loss (BCEWithLogitsLoss).
- The Adam optimizer is used with a learning rate of 1e-4.
- Training runs for 50 epochs, and a checkpoint is saved every 10 epochs.

- Fingerprint Segmentation and Preprocessing

  A. Segmentation Mask Generation

  1. The trained U-Net model takes an input fingerprint image and generates a binary mask where:
     a. White (1) represents the fingerprint region.
     b. Black (0) represents the background.
  2. The segmentation mask is resized to 480×480 pixels to standardize processing.

  B. Fingerprint Cropping using Mask

  3. Bounding box extraction is applied to detect the largest connected component (fingerprint region).
  4. The detected fingerprint region is cropped while maintaining its original aspect ratio.

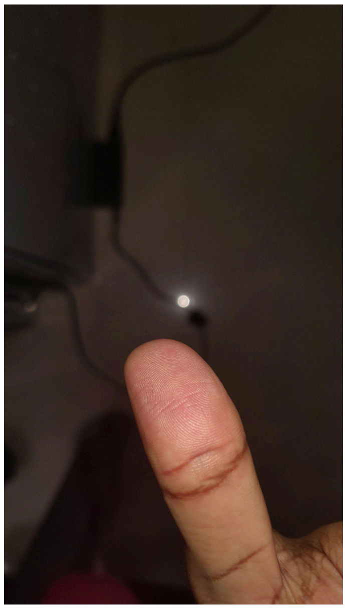
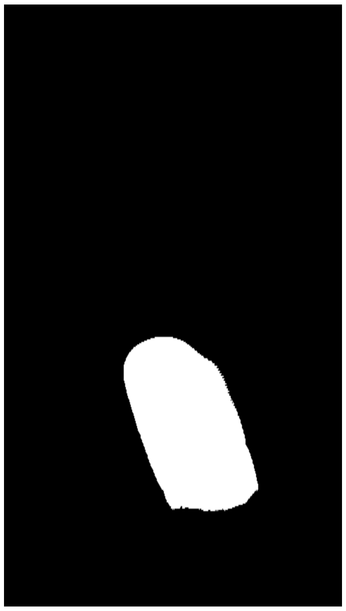
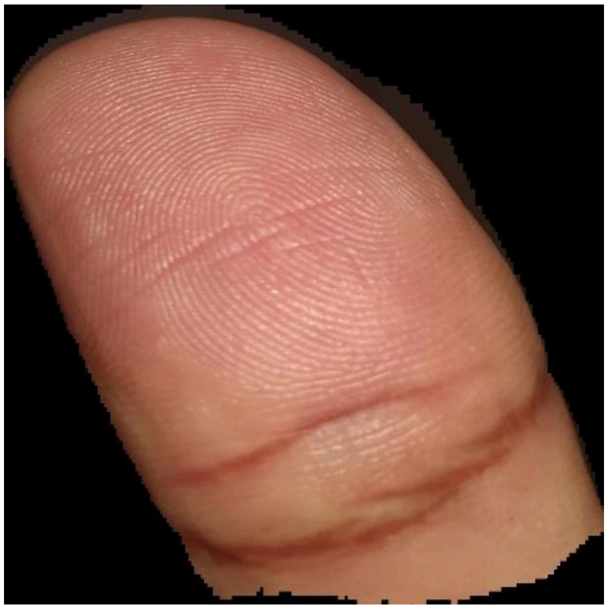
  C. Post-Processing for Standardization

  5. The cropped fingerprint is resized to 480×480 pixels using bilinear interpolation to ensure consistency across all images.
  6. The final fingerprint is stored in three formats:
     a. Resized cropped fingerprint
     b. Binary segmentation mask
     c. Overlayed segmentation result

- Implementation Details
  - Step 1: Image Preprocessing for Segmentation
    - The input fingerprint image is converted to RGB format and resized to 480×480 for U-Net processing.
    - The pixel values are normalized between 0 and 1.
  - Step 2: Applying U-Net for Segmentation
    - The trained U-Net model is loaded.
    - The model performs a forward pass to generate the segmentation mask.

        - A thresholding operation is applied to convert soft probabilities into a binary mask.
    - Step 3: Cropping the Fingerprint Region
        - The largest connected component in the mask is extracted using cv2.boundingRect().
        - The image is cropped to remove unnecessary background regions.
    - Step 4: Resizing and Saving Output
        - The cropped fingerprint is resized to 480×480 pixels while maintaining its aspect ratio.
        - The segmented fingerprint, binary mask, and overlapped result are saved in an output directory.

- Results
    - Visual Comparisons:
        - Before Segmentation: The fingerprint image contains background noise and varying illumination.
        - After Segmentation: The fingerprint region is cleanly extracted, removing unnecessary areas.
    - Effectiveness Analysis:
        - U-Net-based segmentation effectively isolates the fingerprint with high accuracy.
        - Bounding box-based cropping removes unwanted background elements.
        - Resizing ensures consistency, enabling reliable fingerprint feature extraction.
    - Sample Outputs:
        - Raw image vs. Segmented mask vs. Cropped fingerprint

BEFORE (mask) AFTER

2.2 Enhancement

- Methodology:

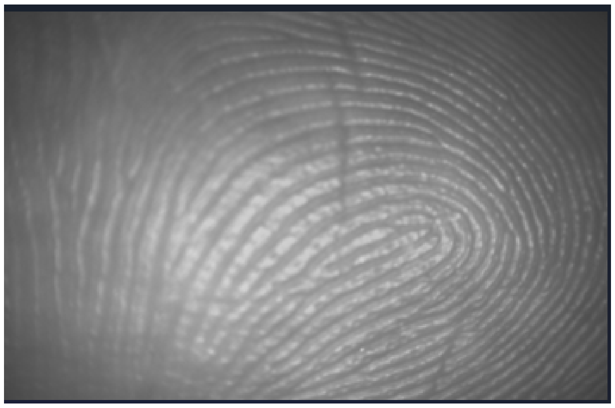
  Enhancement plays a crucial role in improving fingerprint quality for better feature extraction and matching. The following techniques are applied:

  1. Contrast Limited Adaptive Histogram Equalization (CLAHE):
     - Traditional histogram equalization often over-enhances noise, leading to unwanted distortions.
     - CLAHE improves contrast by applying localized histogram equalization while limiting noise amplification.
     - A tile grid size of (12,12) is used to balance local contrast enhancement while preserving fine fingerprint details.
     - The clipLimit is set to 2.0 to prevent excessive contrast enhancement.
  2. Pixel Gray-Level Inversion:
     - Contactless fingerprint images often exhibit inverse ridge-valley structures compared to contact-based fingerprints.

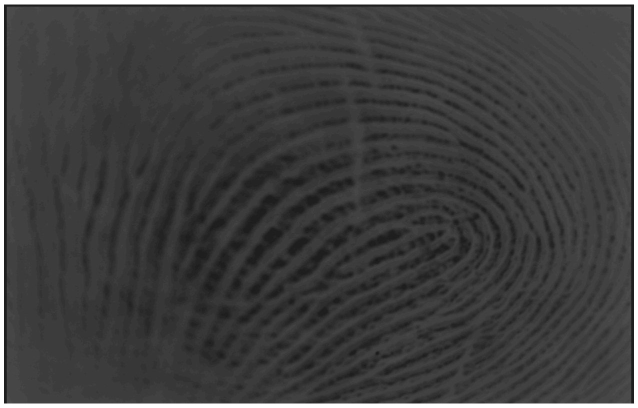
    - To ensure compatibility with traditional databases, pixel-wise inversion is performed using OpenCV's bitwise_not().
    - This operation flips grayscale intensity values, making ridges darker and valleys lighter, aligning them with contact-based fingerprints.
3. Normalization:
    - Fingerprint images often have varying intensity distributions.
    - The final step involves normalizing pixel intensities to a standard range (e.g., mean 60, standard deviation 60) using the normalize() function.
    - Normalization ensures uniform lighting conditions across images and facilitates better minutiae extraction.

- Implementation Details
    1. Input Processing:
        - The fingerprint images are loaded in grayscale using cv.imread().
        - The images are iteratively processed from the dataset while maintaining directory structure.
    2. Processing Steps:
        - CLAHE is applied to enhance ridge visibility.
        - Pixel inversion is performed to align ridge contrast with contact-based fingerprints.
        - The image is normalized to ensure consistent intensity distribution.
    3. Output Storage:
        - Enhanced images are saved while preserving the original dataset's directory structure.
        - The processed images are stored separately in a designated output directory.
- Results
    1. Visual Comparisons:
        - Before Enhancement: Contactless fingerprints exhibit lower contrast, making ridge structures difficult to distinguish.
        - After Enhancement: The enhanced images show improved ridge contrast, clearer ridge-valley separations, and better resemblance to contact-based fingerprints.

2. Effectiveness Analysis:
    - The application of CLAHE significantly improves ridge clarity without introducing excessive noise.
    - Inverted images better match the ridge orientation of contact-based fingerprints, improving matching accuracy.
    - Normalization ensures a uniform brightness level across all images, reducing variations in lighting conditions.
3. Sample Outputs:
    - Side-by-side comparisons of original vs. enhanced images are included in the appendix for verification.



BEFORE



AFTER

2.3 Distortion Correction and Scaling

- Methodology

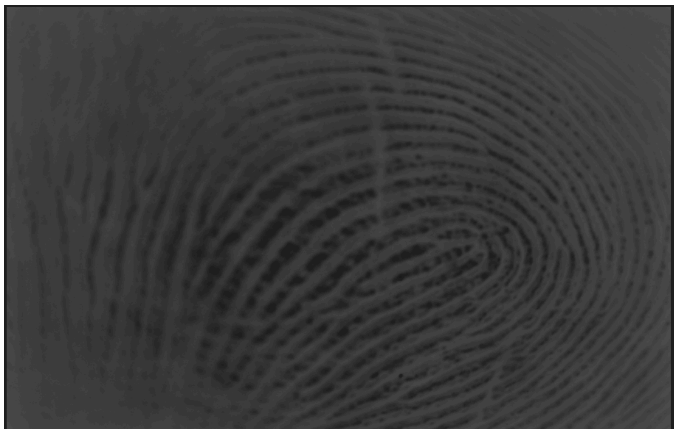
  To align contactless fingerprints with contact-based ones, we employ the Spatial Transformer Network (STN), a deep learning module designed to correct geometric and nonlinear distortions. The STN ensures that features from contactless fingerprint images are mapped to a form compatible with traditional contact-based fingerprint databases.

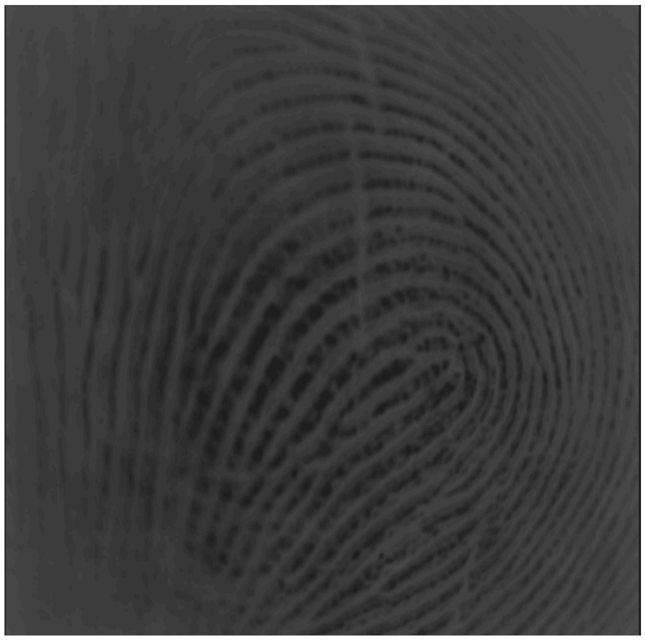
The STN performs two key transformations:

1. Affine Transformation:
    - This corrects perspective distortions caused by varying distances between the finger and the camera.
    - The transformation matrix learns scaling, rotation, and translation parameters to normalize positioning variations.
    - It ensures that the overall structure of the fingerprint is preserved without introducing excessive deformation.
2. Thin-Plate Spline (TPS) Warping:
    - Contact-based fingerprints are captured by pressing the finger against a rigid surface, leading to skin deformation.
    - TPS warping corrects these nonlinear distortions by interpolating key ridge structures in the fingerprint.
    - This transformation ensures that ridges and valleys from contactless images align accurately with their contact-based counterparts.

- Implementation Details
    - Input Processing:
        - Each fingerprint image is first converted to grayscale and resized to 480×480 for uniformity.
        - The images are transformed into tensors for processing in a deep learning model.
    - Neural Network Processing:
        - A convolutional network extracts local features to estimate transformation parameters.
        - The STN module predicts a 2×3 affine transformation matrix and applies it to correct perspective distortions.
        - TPS warping is then performed to fine-tune alignment by correcting local deformations.
    - Output Storage:
        - The corrected images are saved in a structured directory, preserving dataset hierarchy for further analysis.

- Results
    1. Visual Comparisons:
        - Before Correction: Contactless fingerprints exhibit distortions due to varying angles and skin elasticity.
        - After Correction: The STN output shows better ridge alignment, with minimized distortions and improved feature preservation.
    2. Effectiveness Analysis:
        - Affine transformation effectively handles global misalignment caused by camera positioning variations.
        - TPS warping improves local ridge alignment, crucial for fingerprint matching algorithms.
        - The transformed images resemble contact-based fingerprints, leading to higher matching accuracy.
    3. Sample Outputs:
        - A comparison of raw vs. STN-corrected fingerprints





BEFORE AFTER

---

3. Identifying Fingerprint Orientation Fields Using a Gradient-Based Method

- Methodology

  Fingerprint orientation fields provide crucial information about the directionality of ridges across different regions of the fingerprint image. This information is essential for minutiae extraction, fingerprint alignment, and matching. The orientation field captures the local ridge flow at each point and helps in noise reduction, enhancement, and feature extraction for contact-to-contactless fingerprint matching.

  The gradient-based approach is a widely used method for computing fingerprint orientation fields. It relies on analyzing the variations in grayscale intensity to determine local ridge directions. This process is broken down into four key steps:

  1. Gradient Computation using Sobel Filters
  2. Block-wise Dominant Orientation Calculation
  3. Gaussian Smoothing for Refinement
  4. Visualization of the Orientation Field

Step 1: Gradient Computation using Sobel Filters

The Sobel operator is applied to detect horizontal (Gx) and vertical (Gy) gradients in the fingerprint image. The gradient magnitude is calculated as:

And the gradient direction is given by:

Step 2: Block-wise Dominant Orientation Calculation

The fingerprint image is divided into fixed-size blocks (W × W pixels). The dominant orientation for each block is computed as:

Step 3: Gaussian Smoothing for Refinement

To ensure smooth transitions between neighboring blocks, a Gaussian filter is applied:

This step reduces abrupt changes in orientation and improves ridge alignment accuracy.

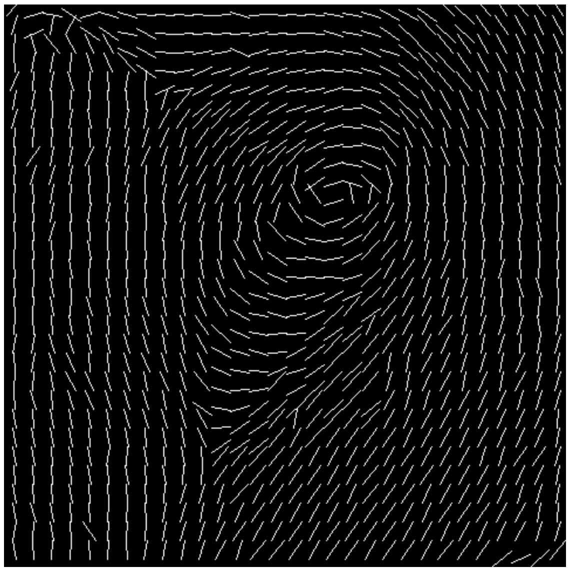
Step 4: Visualization of the Orientation Field

To represent the computed orientations visually, a grid of white lines is drawn over the fingerprint image, where each line segment represents the dominant ridge direction for that region.

- Implementation Details
  - Input Processing:
    - Grayscale images are loaded from the STN-corrected output folder.
    - Images are converted to NumPy arrays for processing.
  - Processing Steps:
    - Step 1: Apply Sobel filters to compute Gx and Gy.
    - Step 2: Compute block-wise dominant orientation (16×16 pixel blocks).
    - Step 3: Apply Gaussian smoothing to refine orientation estimates.
    - Step 4: Visualize the orientation field by overlaying directional lines.
  - Output Storage:
    - The computed orientation field images are saved in a dedicated directory for further processing.

- Results
  - Visual Comparisons:
    - Before Orientation Field Computation: The fingerprint image contains ridges, but their orientations are not explicitly visible.
    - After Orientation Field Computation: The computed orientation field overlays ridge directions with short white line segments, making ridge flow clear.
  - Effectiveness Analysis:
    - Sobel gradient computation extracts ridge directionality effectively.
    - Block-wise estimation captures the dominant ridge flow, reducing sensitivity to noise.
    - Gaussian smoothing eliminates abrupt orientation shifts, ensuring smooth ridge continuity.
  - Sample Outputs:

        - Raw fingerprint image vs. computed orientation field comparisons are provided in the appendix.



---

4. Minutiae Extraction

- Introduction

Minutiae points are the unique ridge characteristics in a fingerprint that help in biometric identification. The two primary minutiae types are:

- Ridge endings: Where a ridge terminates.
- Bifurcations: Where a ridge splits into two.

While extracting these minutiae points is crucial for fingerprint recognition, false minutiae may appear due to noise, broken ridges, and segmentation artifacts. Therefore, a filtering mechanism is essential to eliminate unreliable minutiae points and ensure accuracy.

- Methodology

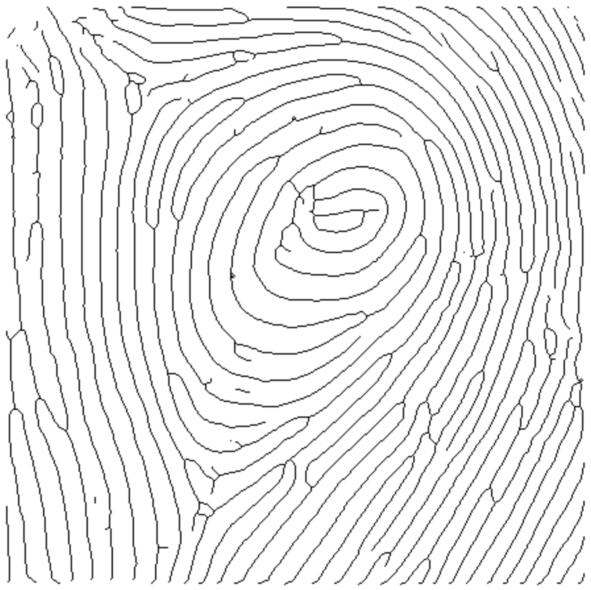
The minutiae extraction process is divided into three major steps:

1. Ridge Thinning (Skeletonization)
    - Converts the fingerprint image into a single-pixel-wide ridge structure using skimage.morphology.skeletonize().
    - Ensures that ridge endings and bifurcations are extracted precisely.
2. Minutiae Detection using the Crossing Number Method
    - A 3×3 local neighborhood is analyzed around each ridge pixel.
    - The crossing number (CN) formula determines the type of minutia: which represents pixel values in a circular neighborhood.
    - Minutiae classification:
        - CN = 1 → Ridge Ending
        - CN = 3 → Bifurcation
3. Computation of Minutiae Orientation
    - A Sobel operator is applied over a small window around each minutia point to determine gradient-based orientation:
    - This ensures that minutiae points are aligned correctly for matching.
4. False Minutiae Removal
    - Several types of false minutiae arise due to noise, segmentation errors, or ridge distortions. The implemented filtering techniques include:
      A. Border Removal
        - Minutiae too close to the fingerprint boundary (within 25 pixels) are discarded to avoid unreliable detections.
    - B. Cluster Reduction
        - If multiple minutiae exist within a 15-pixel radius, only the one nearest to the cluster center is retained.
    - C. Pairwise Removal Based on Proximity and Type
        - If two ridge endings face each other within 20 pixels and their angles are nearly opposite, they are removed.
        - If a ridge ending and a bifurcation are within 15 pixels, both are discarded.
        - If two bifurcations are within 15 pixels, they are removed.

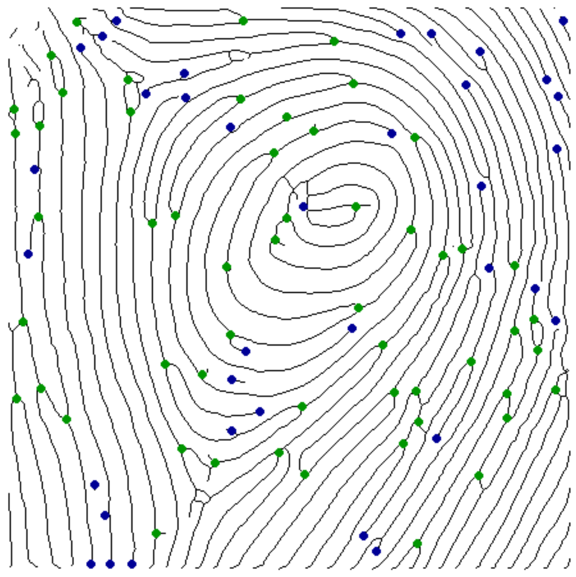
- Implementation Details
    - Step 1: Image Preprocessing
        - Load the grayscale fingerprint image.

        - Convert it into a binary image where ridges = 1, background = 0.
    - Step 2: Ridge Thinning (Skeletonization)
        - Apply morphological skeletonization to reduce ridge thickness while preserving structure.
    - Step 3: Minutiae Detection
        - Iterate through the skeletonized fingerprint image, applying the crossing number method to identify ridge endings and bifurcations.
        - Store each minutia's (x, y) coordinates, type, and orientation in a structured dataset.
    - Step 4: False Minutiae Removal
        - Apply border removal, cluster reduction, and pairwise filtering using Euclidean distance and angular difference constraints to refine the minutiae dataset.
    - Step 5: Visualization and Storage
        - Annotate ridge endings (red) and bifurcations (green) on the fingerprint image.
        - Store extracted minutiae in a CSV file for later verification.

- Results
    - Visual Comparisons:
        - Before Minutiae Extraction: The skeletonized fingerprint image contains ridges but lacks feature markings.
        - After Minutiae Extraction: Minutiae points are annotated with color-coded markers based on type.
    - Effectiveness Analysis:
        - The crossing number method accurately identifies minutiae points.
        - Gradient-based orientation computation ensures correct alignment of extracted minutiae.
        - The false minutiae removal step significantly improves the reliability of extracted minutiae by reducing noise-induced errors.
    - Sample Outputs:
        - Annotated fingerprint images with minutiae points



THINNED IMAGE



FILTERED MINUTIAE POINTS IMAGE

5. Matching Methods

Fingerprint matching is the final stage in the Contact to Contactless (C2CL) fingerprint recognition pipeline. After minutiae extraction, the goal is to compare two sets of minutiae points—one from a contact-based fingerprint and the other from a contactless fingerprint—to determine their similarity.

<u>Implemented Matching Methods</u>

This assignment implements and evaluates three different fingerprint-matching techniques:

1. Generalized Hough Transform-Based Matching
    - Aligns fingerprints by estimating rotation and translation before performing minutiae matching.
2. Genetic Algorithm-Based Matching
    - Uses an evolutionary approach to optimize alignment and maximize fingerprint-matching accuracy.

3. Core Point-Based Matching
    - Locates the fingerprint's core point and uses it as a reference for minutiae alignment and matching.

Method 1: Generalized Hough Transform-Based Matching

This method utilizes the Generalized Hough Transform (GHT) to align fingerprints before performing minutiae matching.

Key Steps:

1. Fingerprint Alignment Using Hough Transform
    - Estimates relative rotation and translation between the contact and contactless fingerprints.
    - Uses a Hough-based voting scheme to find the best transformation parameters.
2. Minutiae Matching After Alignment
    - The query fingerprint is transformed using computed rotation and translation values.
    - Minutiae points from both fingerprints are compared using Euclidean distance and orientation similarity.
3. Similarity Score Calculation
    - Computes a matching score based on the number of matched minutiae relative to the total detected minutiae in both images.

Advantages:

Robust against translation and rotation differences.
Handles noisy or slightly distorted fingerprints.
Computationally efficient compared to feature-learning-based approaches.

Method 2: Genetic Algorithm-Based Matching

This method applies Genetic Algorithms (GA) to optimize fingerprint alignment and improve matching performance.

Key Steps:

1. Chromosome Representation
    - A possible fingerprint alignment is encoded as a chromosome containing transformation parameters (rotation, translation).
2. Fitness Function Evaluation
    - A fitness function evaluates how well the transformed query fingerprint aligns with the template fingerprint based on minutiae overlap.
3. Genetic Operators
    - Uses selection, crossover, and mutation to generate better alignment configurations in each iteration.
4. Optimal Alignment Selection
    - The best transformation parameters are selected based on the highest fitness score, ensuring the best possible minutiae alignment.

Advantages:

Automatically optimizes fingerprint alignment without manual adjustments.
More robust against distortions and nonlinear variations in fingerprints.
High accuracy, especially for fingerprints with warped or stretched features.

Method 3: Core Point-Based Matching

This method aligns fingerprints by using the fingerprint's core point (central reference point) as a fixed reference.

Key Steps:

1. Core Point Detection
    - Identifies the core point of the fingerprint using orientation field analysis.
    - This serves as a reference for fingerprint alignment.
2. Alignment Using Core Point
    - The query fingerprint is translated and rotated to align with the template fingerprint’s core point.
3. Minutiae Matching

    - Compares aligned minutiae points using Euclidean distance and orientation similarity.
4. Similarity Score Calculation
    - Computes a final similarity score based on the number of matched minutiae points.

Advantages:

Fast and computationally efficient since alignment is based only on the core point. Works well when the core point is accurately detected.

---

6. Dataset and Experimental Setup

- Dataset Used: The Hong Kong Polytechnic University Contactless 2D to Contact-based 2D Fingerprint Images Database Version 1.0
- Implementation Details:
    - Programming language - Python
    - Libraries used (OpenCV, NumPy, math)
    - System specifications
    - RAM - 16GB
    - GPU - RTX 3050

---

7. Limitations

Even if a high-quality fingerprint image is captured, extracting the correct features is a major challenge. Contact and contactless fingerprints often have differences in structure and appearance, making it difficult to extract and align minutiae points accurately.

1. Issues with Feature Extraction

- Contactless fingerprint images are captured under different conditions than contact-based images, leading to inconsistent ridge spacing and partial ridge loss due to improper segmentation.

- Traditional minutiae extraction algorithms rely on clear, well-defined ridge patterns. However, contactless images often exhibit distortions, making it difficult to locate bifurcations and ridge endings with high accuracy.
- The natural curvature of a finger is slightly altered when pressed against a contact-based scanner. In contrast, contactless images maintain the finger's natural shape, leading to differences in ridge structure that complicate matching.

2. Challenges in Matching Contactless to Contact-Based Fingerprints

- Contact-based fingerprint images tend to compress the ridges due to pressure, whereas contactless fingerprints appear slightly stretched, leading to discrepancies in ridge alignment.
- If a finger is not positioned at the correct angle during capture, the extracted minutiae points may not align well, making matching difficult.
- The segmentation process may incorrectly detect or miss minutiae points, further reducing the number of overlapping features between contact and contactless fingerprints.

3. Why the Matching Algorithms Struggle

- Generalized Hough Transform: Since this method assumes that extracted minutiae points will maintain a predictable spatial relationship, it struggles when the ridge structure is altered due to stretching or compression.
- Core Point-Based Matching: If the core point is not accurately detected due to segmentation errors, the alignment process fails entirely. This is particularly problematic for contactless fingerprints, where ridges may be less defined.
- Genetic Algorithm-Based Matching: This method works well for slight distortions but requires sufficient overlapping minutiae points. If the initial minutiae extraction fails to capture key ridge features, the algorithm cannot compensate.

4. Potential Improvements

- Instead of using conventional minutiae extraction, deep learning-based feature extraction could be used to identify and adapt to variations in ridge structure.
- Implementing dynamic ridge alignment techniques could help correct distortions caused by differences in pressure and perspective.

- Using a fusion-based approach that incorporates both minutiae-based and texture-based matching could improve overall accuracy.

THANK YOU