Assignment 2

Rojina Kashefi (r1018183)

April 2025

Question 1

The MSS similarity metric is a simple way to compare fingerprint images. It performs well when the images are clean and perfectly aligned. However, it has notable limitations when applied to real-world fingerprint recognition. From the output of this metric, we can see that the image scores are very close together, suggesting the possibility of false positives. Additionally, as shown in the database fingerprint images in Figure 1, the images are not aligned, and the MSS function does not account for this. A major issue with this method is its sensitivity to small shifts, rotations, or changes in scale between images. It also ignores important fingerprint features such as minutiae points, ridge patterns, and pores, which are essential for accurate identification. Furthermore, the method is affected by degradations such as pressure variations and skin conditions during fingerprint acquisition, all of which can reduce the reliability of the score. Therefore, while this approach may be suitable for basic comparisons or controlled environments, it is not adequate for critical tasks such as suspect identification. More advanced methods, such as those based on minutiae matching or deep learning, are better suited for real-world applications.

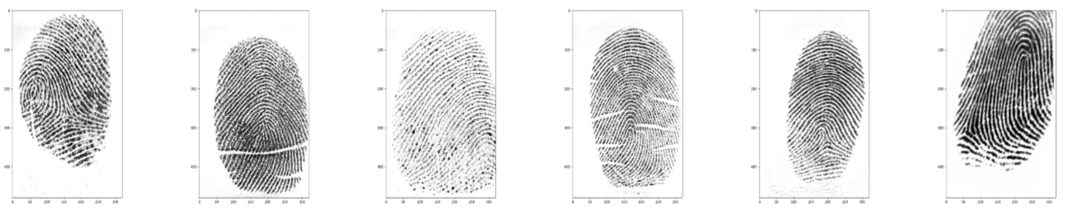

Figure 1: Fingerprints in dataset

Question 2

As we can see in Figure 2, the ridges are not continuous, there is poor separation of parallel ridges due to noise, and there are cuts, creases, or bruises. These issues result in the extraction of many spurious minutiae, as the gaps may be incorrectly identified as terminations, and a large number of genuine minutiae may be missed. This cause problem in position and orientation matching in minutiae detection.

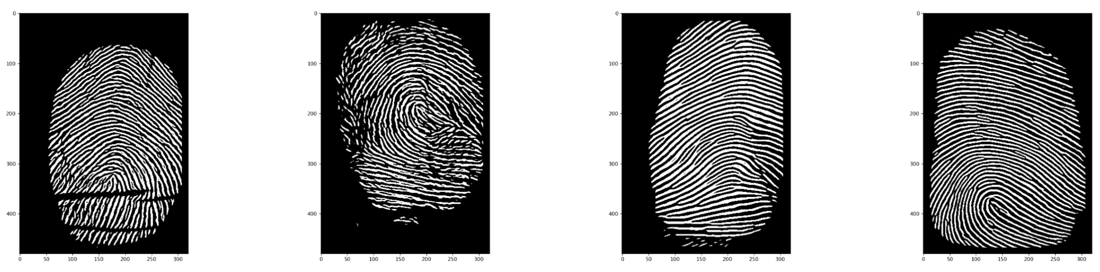

Figure 2: Fingerprints enhanced using Gabor filters

Question 3

The algorithm takes a skeletonized fingerprint image as input, where the ridges have been reduced to 1-pixel-wide lines. It then checks every pixel to find white pixels, which represent the ridges, based on their value. For each white pixel, it inspects its 3×3 neighborhood, which includes the pixel itself and its eight surrounding pixels. It then counts how many of these neighbors are also white. This count determine the type of minutiae: if the pixel has one white neighbor, it is considered a termination, indicating the end of a ridge. If the pixel has three white neighbors, it is identified as a bifurcation, where a single ridge splits into two branches. As shown in Figure 3, the ridge endings are detected as blue points, indicating terminations, while the two branching ridges in the skeleton are marked in red, representing bifurcations. However, there are problems with eye minutiae being detected as bifurcations, and borders being incorrectly identified as ridge endings, resulting in false detections. These issues will be addressed in the next section.

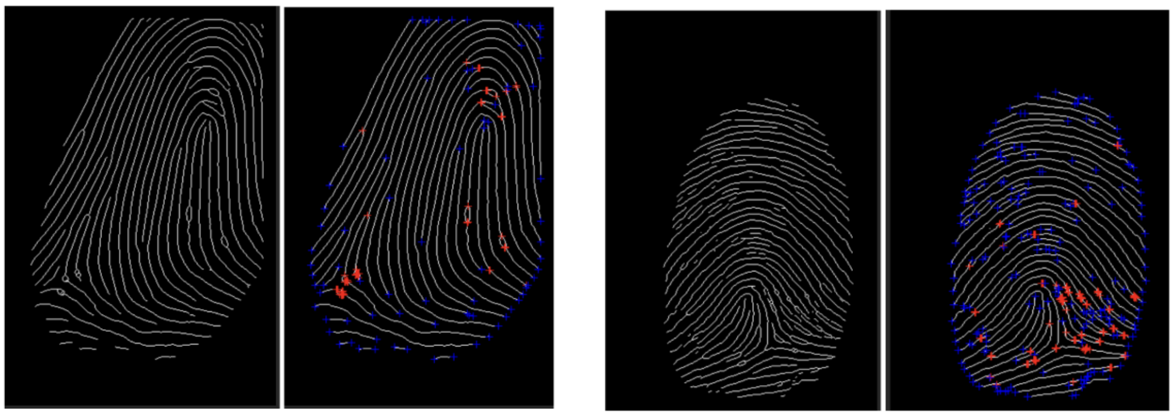
Figure 3: Minutiae detection based on counting white pixel neighborhood

Question 4

As plotted the fingerprints in our database, we observed that each can be rotated, scaled, or slightly distorted. This variation makes matching fingerprints more challenging. In such cases, having information about the orientation can be very helpful. While the location tells us what kind of minutiae it is, the orientation shows the direction of the fingerprint ridges at that point. This additional detail improves matching accuracy even when the fingerprint is not in perfect condition, since both the position and the direction of the minutiae are aligned. As we can see in Figure 4, the left image shows detection based only on location, the middle image includes valid minutiae detection on the skeleton based on both orientation and location, and the right image shows the valid minutiae detected on the perpetrator's image applied.

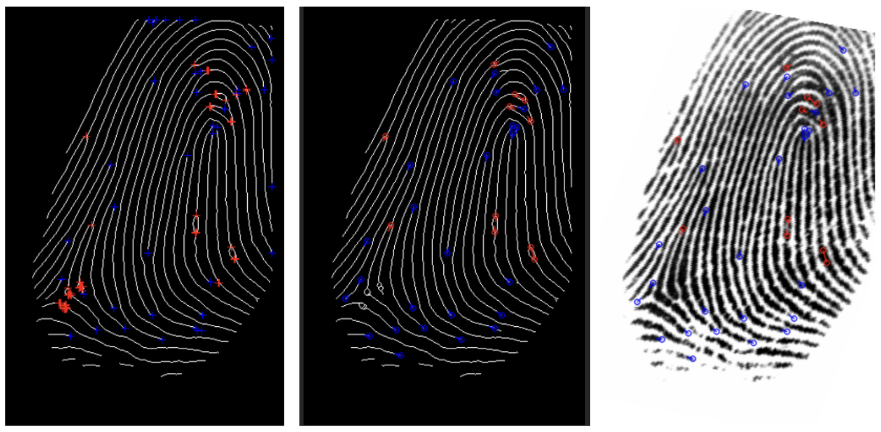
Figure 4: Valid minutiae detection based on orientation and location

Question 5

No, they are not matching, and it is not expected that they would, as they are different fingerprints. As shown in Figure 5, the fingerprints in the left image align well because they are from the same finger. However, in the middle and right images, they do not. In the middle image, the alignment is incorrect due to mismatched minutiae. This occurs because minutiae are local features, and some may appear similar even when they belong to different fingerprints. In the right image, the fingerprint has been scaled to align it with the original, but it fails to capture all keypoints. This is why we need global matching.

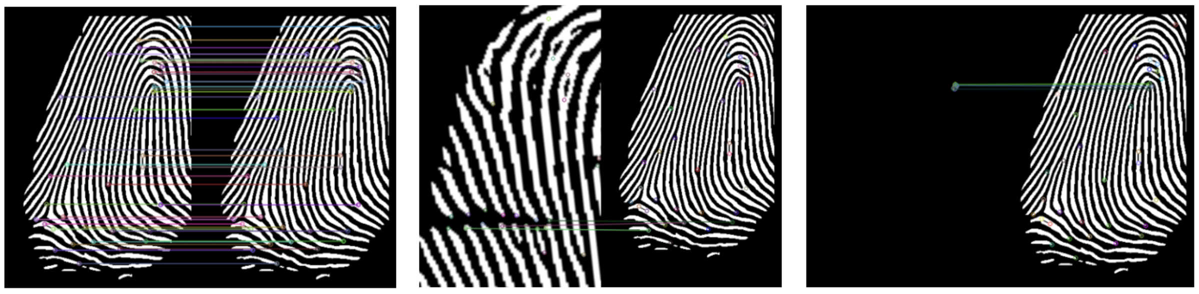

Figure 5: Minutiae matching using Brute-force and RANSAC for image alignment

Question 6

The global similarity score is calculated based on how well the matched keypoints line up between two images. For each local match, it measures the Euclidean distance between the matching keypoints in the input images. If this distance is greater than a threshold of 8, the match is ignored. As we know, the smaller the distance between matches, the higher the score should be. Therefore, I assign a score of $\frac{1}{1+d}$ to each match, where d is the distance. This approach rewards matches that are closer together more.

Question 7

Normalized similarity scores and the knee locator result are displayed in Figure 6. The Knee Locator algorithm identifies a knee point in the curve of sorted scores, where the values begin to drop off significantly and show low matches. In this case, the algorithm suggests a knee at index 7 and a threshold of 0.89. As shown in the score visualization, the first six images of the seven have high similarity. Any score above this threshold can be considered a potential match. Additionally, the top seven matching fingerprints are plotted in the figure.

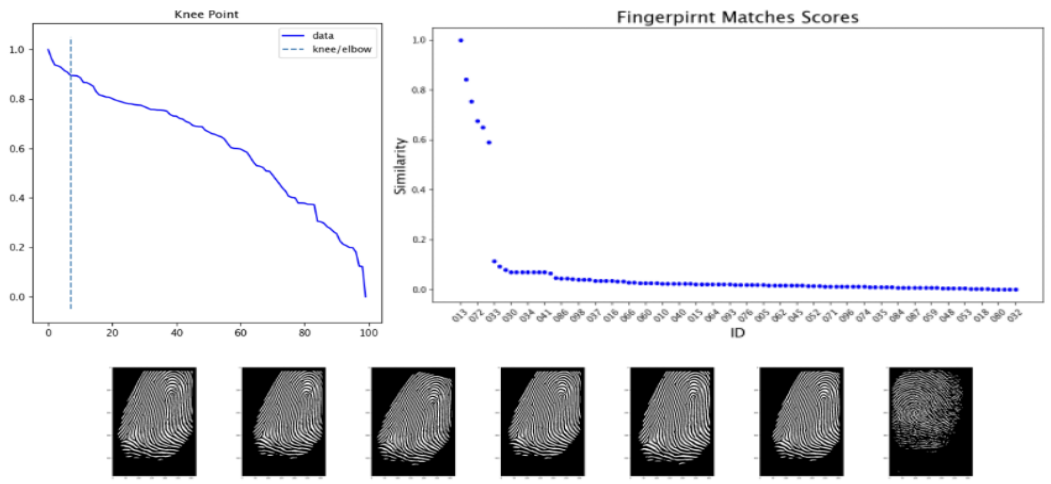

Figure 6: Score visualization and knee locator plot for fingerprint matching

Question 8

As shown in Figure 7, the iris is partially occluded by the upper eyelid, and the image is overly zoomed in, causing a significant portion of the frame to be taken up by the upper eyelid area. Additionally, very little of the lower eyelid is visible. This can create challenges during iris segmentation. The issue is likely caused by an improper distance or angle between the eye and the camera. Moreover, compared to other images in the dataset, the iris in this image appears larger in size. In general, the main challenges in iris recognition include specular reflection, eyelash and eyelid occlusion, and lighting variation. Therefore, it is important to use similarity measures that are invariant to these factors. To address such issues, robust similarity measures, such as the Hamming distance or deep feature embeddings, are commonly employed.

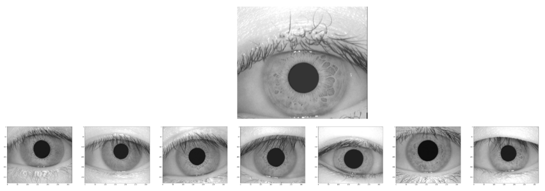

Figure 7: Iris image of the perpetrator and iris images from the database

Question 9

As shown in Figure 8, the iris images are enhanced. Edge detection and the Hough transform are first applied to isolate the circular iris region. Then, Daugman Normalization is used to unwrap the iris into a rectangular form, making it easier to compare across images. The darker oval shapes represent the pupil or occluded areas, such as the eyelids or eyelashes. In some pictures, we can see that a significant portion of the iris is occluded by both the eyelids and eyelashes. In some cases, the pupil segmentation appears inaccurate, which may affect the quality of further iris recognition.

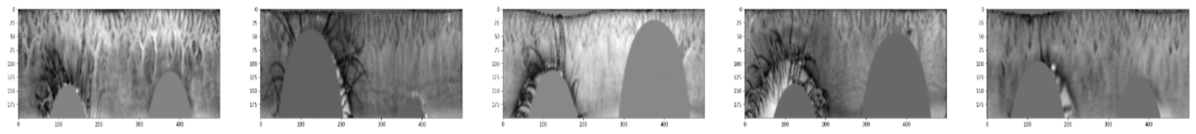

Figure 8: Enhanced iris images using Hough Transform and Daugman Normalization

Question 10

As shown in Figure 8, the t-SNE plot presents a 2D view of the features learned by the model for 100 different classes. Each dot represents a sample, and the colors indicate the class it belongs to. The model performs well by placing samples from the same class close together, showing strong intra-class separation. Most clusters are clearly separated, demonstrating good inter-class distinction, although some overlap, suggesting the model may confuse a few similar classes. The points are also evenly distributed across clusters, indicating balanced class representation in the data.

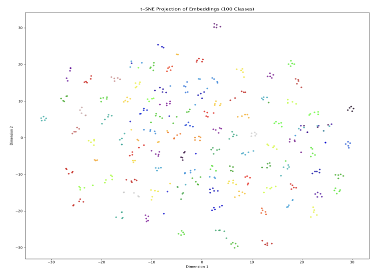

Figure 9: t-SNE plot of iris encoder embeddings

Question 11

The normalized similarity scores and the result from the Knee Locator are shown in Figure 10. For the similarity metric, I used the Euclidean distance between embeddings, where a smaller distance indicates a higher similarity. To determine an appropriate threshold, I applied the Knee Locator, which suggested a threshold of 0.086 at the 11th ranked sample. With longer training, this threshold become more accurate. As illustrated in the score visualization, the distances around this threshold are densely packed, making it difficult to achieve a clear separation between the most similar and less similar samples, thereby we fused analysis in the next section.

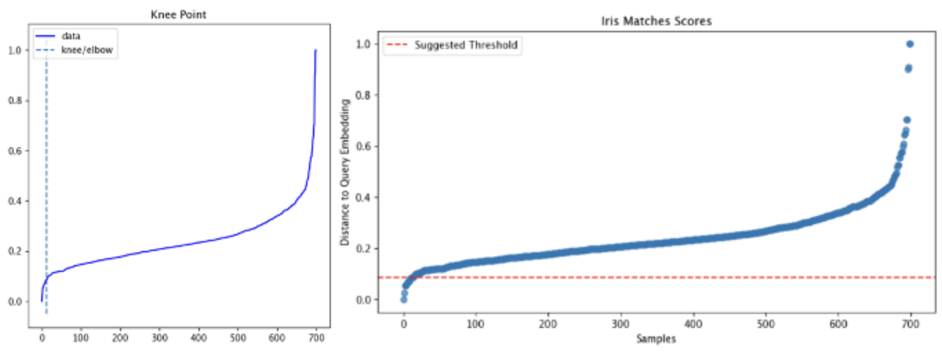

Figure 10: Score visualization and Knee Locator plot for iris matching

Question 12

For score fusion, the similarity metrics from iris matching are computed per sample. First, I averaged the scores belonging to the same person. Then, for fusion, we can use a weighted average, which allows us to adjust the contribution of each modality depending on which one we want to emphasize more. However, in this case, the weights are similar, so the fusion is a simple average of the scores from iris and fingerprint matching. In Figure 11, I plotted the fused scores from highest to lowest. The highest score corresponds to image 13, indicating that image 13 has the strongest match. As observed earlier, the database contains images similar to the perpetrator's fingerprint, and these images exhibit the highest fused scores with the greatest similarity. Since this is an identification system, we can use a CMC curve for evaluation, however drawing CMC curve, requires having ground truth. Another evaluation approach would be to pass every example in the database as a input image and check whether its highest similarity is with itself. This would help verify that the system is robust and that the results are accurate.

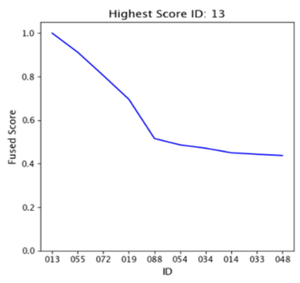

Figure 11: Fused score of fingerprint and iris similarity

Question 13

Neural network for similarity score on fingerprints

In this section, I matched fingerprint images by learning an embedding space where similar fingerprints are close together and dissimilar ones are farther apart. Since we have no labels, a dataset of triplets was created. As shown in Figure 12, each containing an anchor image, a positive image obtained through augmentation of the anchor, and a different negative image. Then, a convolutional neural network was used to extract 128-dimensional embeddings from the fingerprints. The model was trained using triplet loss, encouraging anchors to be closer to positives than to negatives by a set margin. After training, the embedding of a perpetrator's fingerprint was compared to the database embeddings by calculating Euclidean distances and ranking matches from closest to farthest. As shown in Figure 13, all of the most similar images are exactly those associated with the murderer and closely match his fingerprint. This figure demonstrates that the model has accurately learned fingerprint features and can detect similarities without the need for feature extraction, such as minutiae detection.

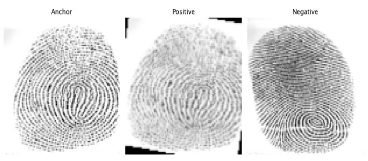

Figure 12: Anchor, Positive (augmented of anchor), and Negative Images for Triplet Loss Training

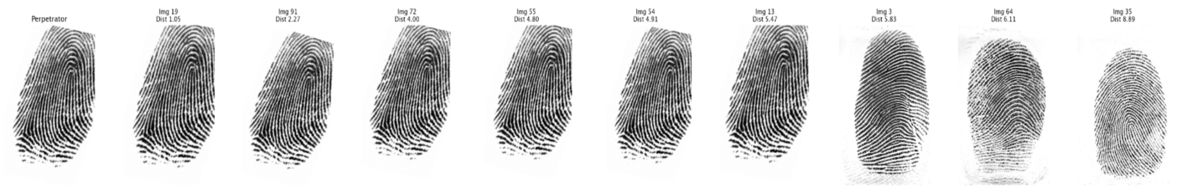

Figure 13: The fingerprints with most similarity to the perpetrator's after training the neural network

Improving feature extraction on fingerprint

In this assignment, we focused on extracting minutiae features. During the lectures, it was noted that non-minutiae features have been receiving increasing attention. New methods based on local texture, ridge geometry, ridge spatial relationships, and pores have been proposed alongside minutiae matching. In this section, I explored a local texture technique known as Local Binary Patterns (LBP). LBP is a simple method for describing the local texture of an image by comparing each pixel's intensity with the intensities of its surrounding neighbors. In my code, I used 24 sampling points at a radius of 3 pixels. For each neighbor, if its intensity is greater than or equal to the center pixel's intensity, it is assigned a value of 1; otherwise, it is assigned a 0. These binary results are then combined into a binary number and converted into a decimal value for that pixel. After computing the LBP codes for all pixels, I constructed a histogram to summarize the local texture distribution across the image. I used the uniform LBP variant, which focuses on patterns with a small number of transitions. I then normalized the histogram so that it could be used as a feature vector for comparing different images. As shown in Figure 14, the transformed image and the corresponding histogram shows which patterns occur most frequently. In this case, the pattern corresponding to a dark or bright spot has the highest frequency, followed by corner patterns. LBP is particularly useful because it is fast to compute, robust to changes, and compact in size. These properties make it ideal for applications such as fingerprint recognition. After extracting the features, I applied the Euclidean distance for matching. As shown the images with features that have a low distance from the murderer's fingerprint are similar to it.

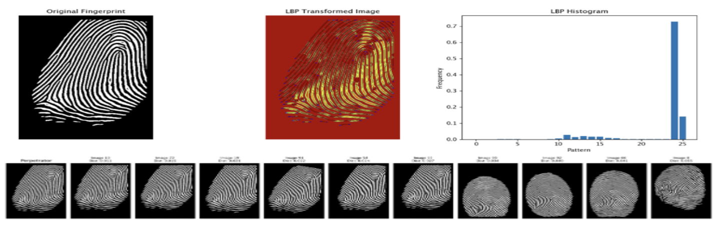

Figure 14: Feature extraction using the local texture method of LBP