Fingerprint and Iris-based identification

Soheila Hesaraki -r0866128: The report focuses on implementing and testing a keypoint-based fingerprint and iris recognition system. It includes an evaluation of local and global similarity, a hybrid system, and a fusion of the two, as well as an analysis of an iris recognition neural network.

1. Data and Context

Our goal is to solve a murder case by implementing a recognition system based on an iris or fingerprint. Both are unique physiological characteristics that can be used for forensics. For fingerprints, we analyze the ridges and valleys on the fingertip surface, often using the right index or thumb. For the iris, we analyze the texture pattern. The fingerprint database has 100 images of size 480x320, and the iris database has 100 images of size 280x320. Sample images are in the notebook.

2. Creating a system for recognizing fingerprints

Data analysis will be conducted to determine the optimal similarity function for fingerprint and iris recognition, including a hybrid approach, through a series of sequential questions.

Q1: Now that we have some results, is the given similarity function a good metric to quantify the distance between two fingerprints? Is it reliable enough to incriminate a suspect? What are its limitations?

The current similarity function is comparing the raw pixel values using mean squared difference which is not reliable because as we have seen in our database, fingerprints have varying scales, orientations, translations, and other factors that can affect their appearance. To accurately assess if two fingerprints match, we must first align them. Aligning the fingerprints involves adjusting them to a common standard, allowing us to compare their pixel values accurately. then can we generate a score that represents the degree of similarity between two fingerprints. Figure 1 demonstrates that different distance metrics, such as 'Euclidean', Braycurtis, and 'Chebyshev', detect high score ID differently. This observation raises concerns about the reliability of these metrics for forensic purposes without proper alignment. The lack of a clear pattern in these plots does not indicate the similarity of the perpetrator's fingerprint with others in the database.

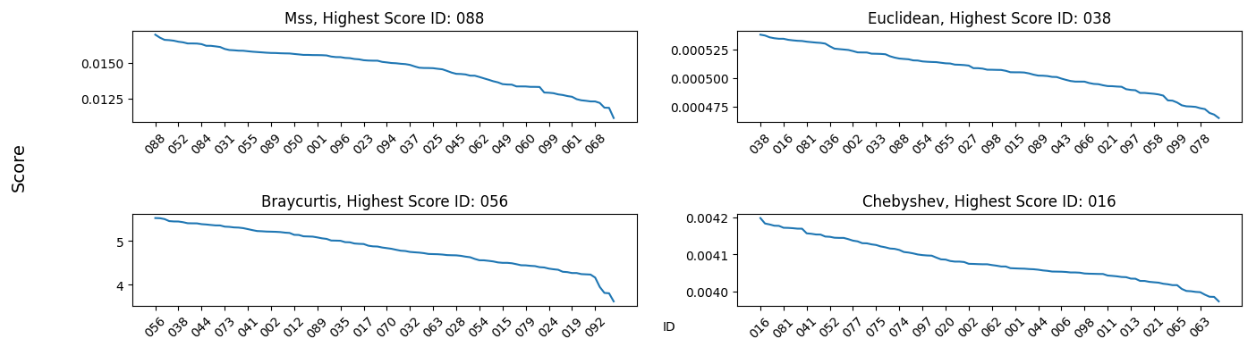


Figure.1. Comparison of 'MSS', ’Euclidean’, ’Braycurtis’ and ’Chebyshev’ pairwise distance function from the scikit-learn library

2.2 Model Extension: Local Similarity

To prepare our fingerprint database for analysis, we need to preprocess it by segmenting and enhancing the images. This involves using Gabor filter banks to estimate orientation and frequency and calculating a mask based on standard deviation to separate the fingerprint from the background. This method is based on the work of Ukarsh Deshmukh and Peter Kovesi.

2.2.1 Feature extraction

Comparing images solely based on pixels is not sufficient for fingerprint matching. Instead, established methods in computer vision and in this report rely on using keypoint features to compare images on local feature descriptors, global features, and hybrid features. Matching the key points between two fingerprints involves finding matches using brute force and returning the set of distances between the matched pairs, based on the normalized Hamming distance.

Q2: Are all the key point matches accurate? Are they expected to be? Explain why.

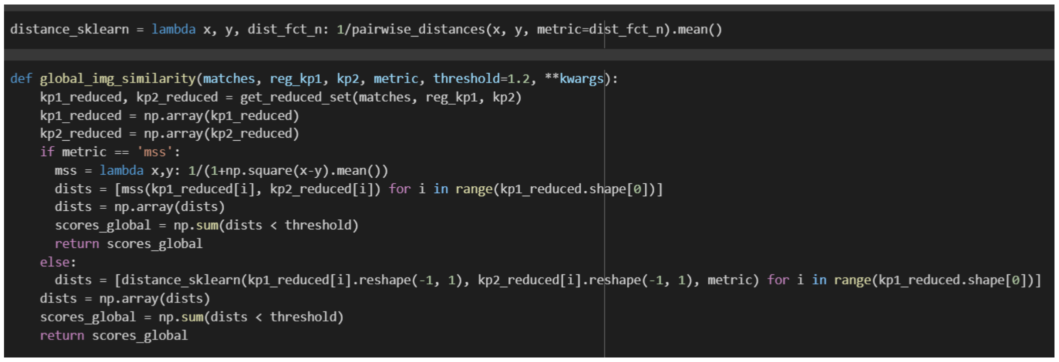
No, Key point matching on non-aligned fingerprints is inaccurate due to challenges in aligning them beforehand, leading to potential false matches and mismatches caused by lighting, image orientation, or other factors that may affect the appearance of key points in the images.

2.3 Model Extension: Global Similarity

Another approach for computing similarity scores is to use global features, which involves aligning the fingerprints based on local matches and then comparing the key points from a geometrical view. This is an alternative approach to comparing fingerprints on a pixel level or using local feature descriptors.

Q3: Choose a global feature similarity function, (e.g. you can start from the Euclidean distance between the reduced sets of KeyPoints and count the values above a threshold).

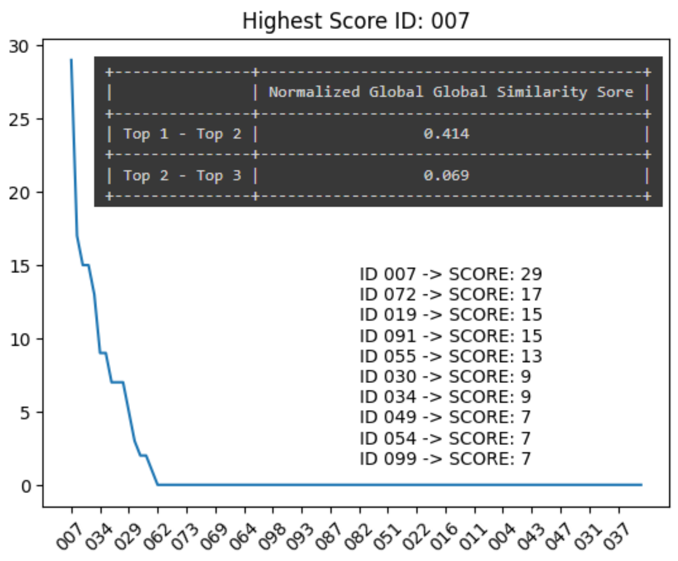
Once the local and global matches have been computed, the next step is to define an image similarity distance in order to calculate scores from the global features. I used 'Euclidean', Braycurtis, and 'Chebyshev' distance to get similarity score.



```
distance_sklearn = lambda x, y, dist_fct_n: 1/pairwise_distances(x, y, metric=dist_fct_n).mean()

def global_img_similarity(matches, reg_kp1, kp2, metric, threshold=1.2, **kwargs):
    kp1_reduced, kp2_reduced = get_reduced_set(matches, reg_kp1, kp2)
    kp1_reduced = np.array(kp1_reduced)
    kp2_reduced = np.array(kp2_reduced)
    if metric == 'mss':
      mss = lambda x,y: 1/(1+np.square(x-y).mean())
      dists = [mss(kp1_reduced[i], kp2_reduced[i]) for i in range(kp1_reduced.shape[0])]
      dists = np.array(dists)
      scores_global = np.sum(dists < threshold)
      return scores_global
    else:
      dists = [distance_sklearn(kp1_reduced[i].reshape(-1, 1), kp2_reduced[i].reshape(-1, 1), metric) for i in range(kp1_reduced.shape[0])]
    dists = np.array(dists)
    scores_global = np.sum(dists < threshold)
    return scores_global
```

Figure.2.Global imgae similarity function



```
+--------------+-----------------------------------------+
|              | Normalized Global Global Similarity Sore |
+--------------+-----------------------------------------+
| Top 1 - Top 2 |                 0.414                   |
+--------------+-----------------------------------------+
| Top 2 - Top 3 |                 0.069                   |
+--------------+-----------------------------------------+
```

```
ID 007 -> SCORE: 29
ID 072 -> SCORE: 17
ID 019 -> SCORE: 15
ID 091 -> SCORE: 15
ID 055 -> SCORE: 13
ID 030 -> SCORE: 9
ID 034 -> SCORE: 9
ID 049 -> SCORE: 7
ID 054 -> SCORE: 7
ID 099 -> SCORE: 7
```

Figure.3 Fingerprints Global similarity scores

2.3.1. Threshold and Distance Metric Effect

The threshold has an effect on the False Match Rate. Setting a high threshold will result in a lower False Match Rate because we only consider key points to be similar if their distance is below the threshold. However, this may increase the False Rejection Rate by rejecting key points that are actually similar.

In order to choose the best distance metric and threshold, we need to consider both of these factors simultaneously. We can do this by analyzing the top matching points for each metric and threshold combination and selecting the one that separates them the most. This will help us find the optimal balance between the False Match Rate and False Rejection Rate.

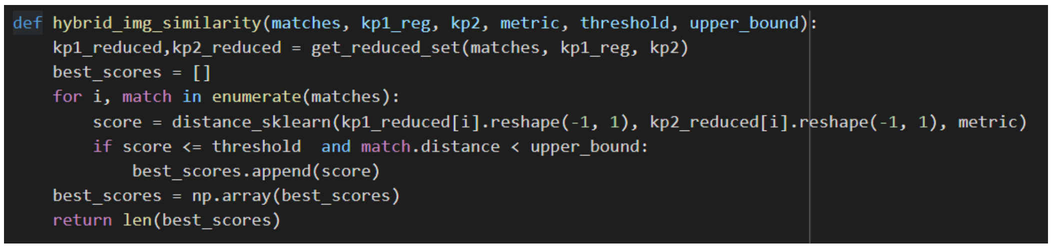
Q4: Visualize the scores and determine a score threshold to discriminate the matching fingerprints. Explain how you determine the threshold.

I calculated the normalized similarity scores in different thresholds and similarity functions. Based on the result in the ipynb file, the metric that separates the most top matching points is braycurtis with a threshold of 3.5, since it has the highest difference between Top 1 - Top 2 (0.414) compared to the other metrics. Threfor, I used this metric and teh threshold.

2.4 Model Extension: Hybrid similarity

Q5: Choose a hybrid feature similarity function that makes use of both the geometric distance and the feature distance of the key points. Using this hybrid function, visualize and assess the matches.

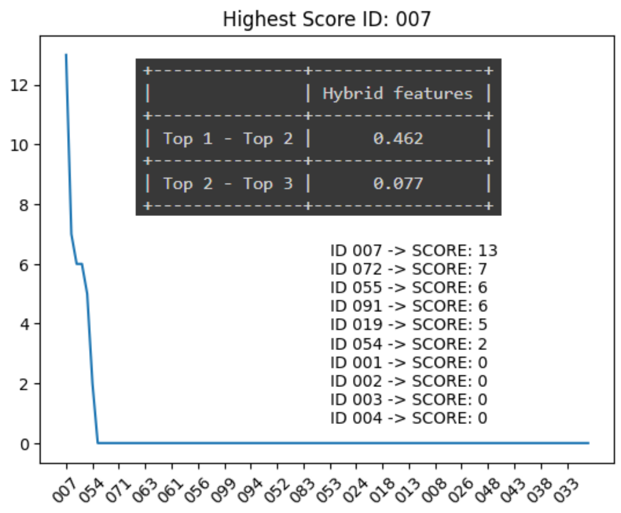
In the hybrid_img_similarity function, match.distance is used as a threshold to filter out poor matches where the descriptors are not similar enough. Using an appropriate value for value for upper_bound can improve the accuracy of feature matching by filtering out poor matches. If the upper_bound value is too small it may not use enough number of key points and if it is too high it cannot filter out poor matches.



```python
def hybrid_img_similarity(matches, kp1_reg, kp2, metric, threshold, upper_bound):
    kp1_reduced,kp2_reduced = get_reduced_set(matches, kp1_reg, kp2)
    best_scores = []
    for i, match in enumerate(matches):
        score = distance_sklearn(kp1_reduced[i].reshape(-1, 1), kp2_reduced[i].reshape(-1, 1), metric)
        if score <= threshold  and match.distance < upper_bound:
            best_scores.append(score)
    best_scores = np.array(best_scores)
    return len(best_scores)
```

Figure.4.Hybrid imgae similarity function

To find the optimal value for upper_bound, we can analyze the top matching points for a range of upper_bound values and select the one that provides the greatest separation between them. However, we should avoid selecting an upper_bound of 1 since this score indicates that only one score exists and few matching points were used, which is not reliable due to the limited number of key points used. In our case, we chose an upper_bound of 41 since it resulted in the highest difference between the top scores. Using Hybrid similarity scored helped and improved slightly the difference value of Top1 and Top 2.



```
+---------------+-----------------+
|               | Hybrid features |
+---------------+-----------------+
| Top 1 - Top 2 |      0.462      |
+---------------+-----------------+
| Top 2 - Top 3 |      0.077      |
+---------------+-----------------+
```

```
ID 007 -> SCORE: 13
ID 072 -> SCORE: 7
ID 055 -> SCORE: 6
ID 091 -> SCORE: 6
ID 019 -> SCORE: 5
ID 054 -> SCORE: 2
ID 001 -> SCORE: 0
ID 002 -> SCORE: 0
ID 003 -> SCORE: 0
ID 004 -> SCORE: 0
```

Figure.5 Fingerprints Hybrid similarity scores

3. Creating a system for recognizing Iris

The iris biometric analysis involves examining the texture patterns to identify a person's identity, and since it focuses on texture, the images can be grayscale.

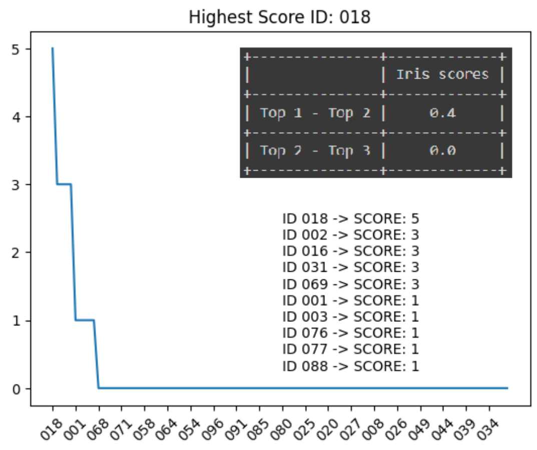
Q6: Check out iris_perpetrator.png. Where do you see difficulties? What kind of similarity measures do you expect to work best?

To analyze the perpetrator's iris scan, the first challenge would be to extract the iris part of the image by detecting and segmenting the flat donut-shaped region, which requires detecting the iris boundary and removing unwanted regions like eyelids. The other challenge can be the Pupil size. The size of the pupil can vary depending on lighting conditions, which can affect the accuracy of iris detection. In terms of similarity measures, I expect the hybrid similarity function using Euclidean, Braycurtis, and Chebyshev distances between the extracted features of Iris images to work well.

3.1. Determine the best similarity table for the iris

Q7: Construct a similarity table with the iris images. Use local or global matches in order to get scores. Use the scores you get to determine the perpetrator.

Earlier, we have discussed various similarity approaches to generate a similarity table. Similarly, I applied the same process to the iris images and found that using the hybrid similarity function based on the braycurtis distance metric with a threshold of 2.5 worked best (the highest difference between Top 1 - Top 2 (0.4)) .



```
+---------------+-------------+
|               | Iris scores |
+---------------+-------------+
| Top 1 - Top 2 |     0.4     |
+---------------+-------------+
| Top 2 - Top 3 |     0.0     |
+---------------+-------------+
```

```
ID 018 -> SCORE: 5
ID 002 -> SCORE: 3
ID 016 -> SCORE: 3
ID 031 -> SCORE: 3
ID 069 -> SCORE: 3
ID 001 -> SCORE: 1
ID 003 -> SCORE: 1
ID 076 -> SCORE: 1
ID 077 -> SCORE: 1
ID 088 -> SCORE: 1
```

Figure.6 Iris similarity scores

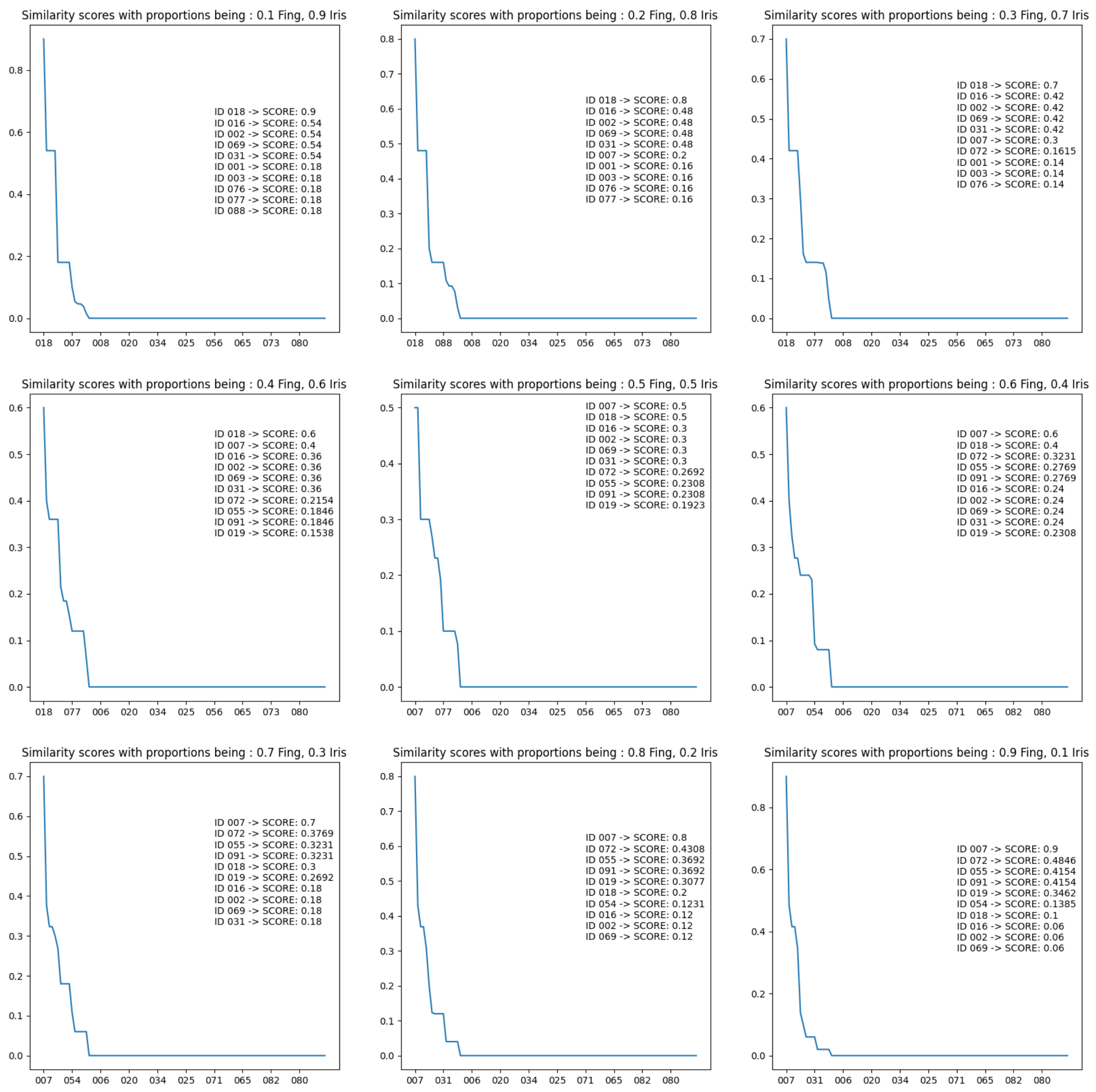
4. Multimodel system

Q8: Fuse your iris and fingerprint biometric system on the score level to solve the murder case! Do you feel confident in your prediction? How do you fuse the scores? Why?

Up until now, we have observed that the perpetrator had the ability to manipulate both the fingerprint and iris databases. However, there is still a possible solution to identify the killer. We can create a multimodal similarity table by combining the fingerprint similarity table and the iris similarity table. To achieve this, we must normalize both tables and may choose to assign a higher weight to a particular model if it produces more reliable results. Table 1 shows the difference between top scores with different value of weight model. Althouh the Top1 - Top 2 value is the same for some combinations, we choose the one that has higher top score. This greater top score makes us more confident in the results of this final model, although there is still some degree of uncertainty involved. As shown in Figure7, person 7 obtained the highest similarity score of 0.9, which is 0.416 higher than the second-highest score

Table.1. The difference between top scores with different value of weight model

(Fingerprint system weight, Iris system weight)	(0.1, 0.9)	(0.2, 0.8)	(0.3 ,0.7)	(0.4, 0.6)	(0.5, 0.5)	(0.6, 0.4)	(0.7, 0.3)	(0.8, 0.2)	(0.9 ,0.1)
Top1 - Top 2	0.4	0.4	0.4	0.333	0	0.333	0.462	0.462	0.462
Top2 - Top 3	0	0	0	0.067	0.4	0.128	0.077	0.077	0.077



```
ID 018 -> SCORE: 0.9
ID 016 -> SCORE: 0.54
ID 002 -> SCORE: 0.54
ID 069 -> SCORE: 0.54
ID 031 -> SCORE: 0.54
ID 001 -> SCORE: 0.18
ID 003 -> SCORE: 0.18
ID 076 -> SCORE: 0.18
ID 077 -> SCORE: 0.18
ID 088 -> SCORE: 0.18
```

```
ID 018 -> SCORE: 0.8
ID 016 -> SCORE: 0.48
ID 002 -> SCORE: 0.48
ID 069 -> SCORE: 0.48
ID 031 -> SCORE: 0.48
ID 007 -> SCORE: 0.2
ID 001 -> SCORE: 0.16
ID 003 -> SCORE: 0.16
ID 076 -> SCORE: 0.16
ID 077 -> SCORE: 0.16
```

```
ID 018 -> SCORE: 0.7
ID 016 -> SCORE: 0.42
ID 002 -> SCORE: 0.42
ID 069 -> SCORE: 0.42
ID 031 -> SCORE: 0.42
ID 007 -> SCORE: 0.3
ID 072 -> SCORE: 0.1615
ID 001 -> SCORE: 0.14
ID 003 -> SCORE: 0.14
ID 076 -> SCORE: 0.14
```

```
ID 018 -> SCORE: 0.6
ID 007 -> SCORE: 0.4
ID 016 -> SCORE: 0.36
ID 002 -> SCORE: 0.36
ID 069 -> SCORE: 0.36
ID 031 -> SCORE: 0.36
ID 072 -> SCORE: 0.2154
ID 055 -> SCORE: 0.1846
ID 091 -> SCORE: 0.1846
ID 019 -> SCORE: 0.1538
```

```
ID 007 -> SCORE: 0.5
ID 018 -> SCORE: 0.5
ID 016 -> SCORE: 0.3
ID 002 -> SCORE: 0.3
ID 069 -> SCORE: 0.3
ID 031 -> SCORE: 0.3
ID 072 -> SCORE: 0.2692
ID 055 -> SCORE: 0.2308
ID 091 -> SCORE: 0.2308
ID 019 -> SCORE: 0.1923
```

```
ID 007 -> SCORE: 0.6
ID 018 -> SCORE: 0.4
ID 072 -> SCORE: 0.3231
ID 055 -> SCORE: 0.2769
ID 091 -> SCORE: 0.2769
ID 016 -> SCORE: 0.24
ID 002 -> SCORE: 0.24
ID 069 -> SCORE: 0.24
ID 031 -> SCORE: 0.24
ID 019 -> SCORE: 0.2308
```

```
ID 007 -> SCORE: 0.7
ID 072 -> SCORE: 0.3769
ID 055 -> SCORE: 0.3231
ID 091 -> SCORE: 0.3231
ID 018 -> SCORE: 0.3
ID 019 -> SCORE: 0.2692
ID 016 -> SCORE: 0.18
ID 002 -> SCORE: 0.18
ID 069 -> SCORE: 0.18
ID 031 -> SCORE: 0.18
```

```
ID 007 -> SCORE: 0.8
ID 072 -> SCORE: 0.4308
ID 055 -> SCORE: 0.3692
ID 091 -> SCORE: 0.3692
ID 019 -> SCORE: 0.3077
ID 018 -> SCORE: 0.2
ID 054 -> SCORE: 0.1231
ID 016 -> SCORE: 0.12
ID 002 -> SCORE: 0.12
ID 069 -> SCORE: 0.12
```

```
ID 007 -> SCORE: 0.9
ID 072 -> SCORE: 0.4846
ID 055 -> SCORE: 0.4154
ID 091 -> SCORE: 0.4154
ID 019 -> SCORE: 0.3462
ID 054 -> SCORE: 0.1385
ID 018 -> SCORE: 0.1
ID 016 -> SCORE: 0.06
ID 002 -> SCORE: 0.06
ID 069 -> SCORE: 0.06
```

Figure.7 Multimodal system similarity scores

5. Additional Task

Create a biometric identification or authentication system on a modality of choice and evaluate it (6pt)

In this section, we aim to develop a non-contact palmprint recognition system using the ROI dataset. Palmprint refers to the skin patterns on the inner palm surface, comprising mainly two kinds of features, the palmar friction ridges (the ridge and valley structures like the fingerprint) and the palmar flexion creases (discontinuities in the epidermal ridge patterns). The authors have made available two datasets: the original dataset containing raw images and the ROI dataset, which already includes palm detection and alignment. For simplicity, we will be using the ROI dataset. However, for those interested in extracting ROIs from raw images, the authors have provided a comprehensive explanation in their paper.

5.1 Preparing the data

To develop this system, we will also utilize some functions that were previously used for fingerprint and iris recognition. The first step is to download the dataset from the authors' Google Drive, which includes two subfolders, session 1 and session 2. Both subfolders contain 6000 images, corresponding to 600 different palms. Images from 00001.bmp to 00010.bmp represent identity 1, images from 00011.bmp to 00020.bmp represent identity 2, and so on until images from 05991.bmp to 06000.bmp correspond to identity 600. Due to limited computational capacity, we will only use the first 200 identities, containing images from 00001.bmp to 02000.bmp, from both sessions. Therefore, our dataset will consist of 200 identities, each having 10 images from each session, totaling 40 images for each identity. The size of the resulting dataset will be 4000.

For splitting the data into training and validation sets, I used train_test_split of sklearn.model_selection module. This function is used to randomly split a given dataset into two subsets training (80% of data) and validation (20% of data).

In Figure 8, you can see the palm print of four different people.

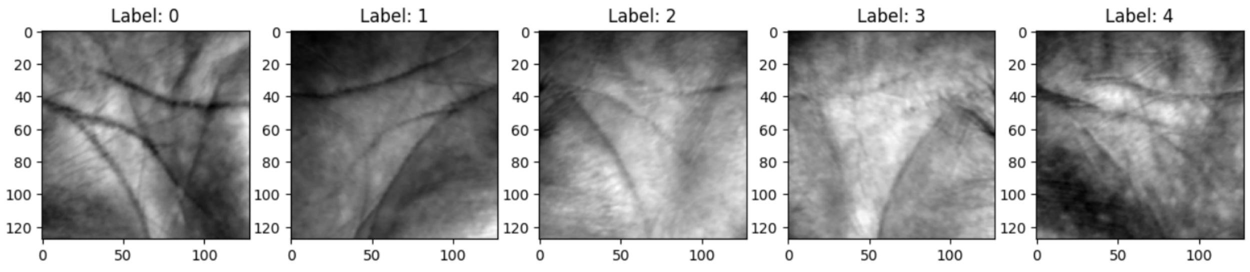


Figure.8 Palm figures, label number is reffering to people's identity

5.2 Feature Extraction

Extracting features from palmprints is a crucial step in recognizing them. Different methods, such as PalmCode and CompCode, have been developed for palmprint recognition, inspired by Daugman's IrisCode. Some authors have also used more traditional feature extraction techniques, like SIFT. SIFT features are usually combined with the scores of other methods to create a more robust recognition system. Our strategy involves implementing a feature extraction pipeline founded on deep learning principles.

The feature extraction approach I opted for in this project is Deep Learning. While CNN models with residual layers and triplet loss have demonstrated success in palmprint recognition, my pipeline takes a different approach. I excluded residual layers and instead utilized an SVM to classify the palmprint input. It is important to note that this is an identification-based setup, meaning we aim to classify the input palmprint into one of the N identities or assign it to the "unknown" identity, requiring the classifier to have N+1 classes.

5.3 Network Design

In order to create a robust network for palmprint recognition, we need a network that can learn image embeddings effectively. The embeddings should be able to handle different image transformations such as translations, rotations, and changes in illumination without compromising the accuracy of the recognition system. A Convolutional Neural Network (CNN) is a suitable choice for this task. The CNN will generate a feature vector, or embedding, that represents the input image. I decided to use the Triplet Loss function to ensure that the embeddings of images from different classes are well separated in the feature space. Figure 9 illustrates the complete pipeline.

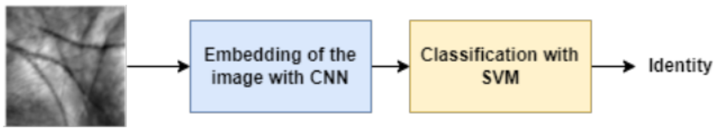


Figure.9 Training of the embedding extraction step

In the realm of Face Recognition systems, Triplet Loss is a frequently used method. Its simplicity is matched only by its elegance. At Figure 9, we present the training process for CNNs, using Triplet Loss. We begin with an Anchor image from identity "A," Positive images from the same identity, and Negative images that do not belong to identity "A." Our objective is to train the network to increase the distance between images from different classes and decrease the distance between images from the same class, in the feature space. The Triplet Loss function is defined as:

$$\sum_{i}^{N} \|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha$$

Here, xi represents the input sample (image), f(xi) represents the embedding computed for that sample, and α is the threshold value between positive and negative pairs. The exponents "a, p, n" signify "anchor, positive, and negative," respectively. By minimizing the Triplet Loss function, we aim to maximize our objective, ensuring that the "distance anchor → positive" is minimal and the "distance anchor → negative" is maximal.

5.4. Results and Evaluation

5.4.1. CNN Training

The evolution of Triplet Loss over 115 epochs is illustrated in Figure 10 a). Interestingly, the validation loss is observed to be lower than the training loss, possibly due to the samples in the validation set being easier to separate than those in the training set. We further assessed the CNN's ability to distinguish between classes by conducting a Principal Component Analysis (PCA) before and after applying the CNN. The results are presented in Figure 10. While the class separation was not optimal after 115 epochs, clusters were still visible. However, it should be noted that the data was visualized in two dimensions and a non-linear classifier will be utilized for classification.

To determine the effectiveness of our triplet loss embeddings, we compared the performance of SVM on the data before applying our pipeline and after. The results in Table 2 indicate that the CNN-embedded palmprint yielded a Rank 1 Recognition Rate of 0.9875, whereas using only SVM resulted in a rate of 0.89. Thus, CNN embedding with triplet loss improved the classification performance.

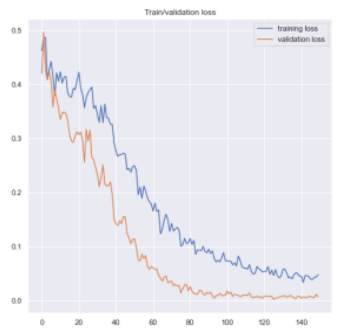


Figure 10 (a) Triplet Loss Training

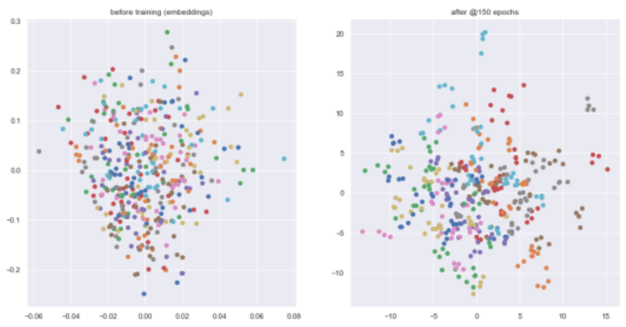


Figure 10 (b) PCA, without CNN and with CNN

Table2. Rank 1 Recognition Rate

SVM	0.89
CNN + SVM	0.9875

5.4.2. CMC:

We plotted the CMC curve, as shown in Figure 11, which demonstrates the steepness of our system's curve and indicates its high performance.

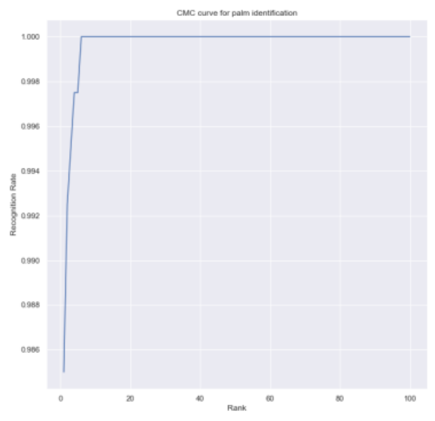


Figure.11 CMC curve