IRIS RECOGNITION vs FACE RECOGNITION

Biometric systems course – cybersecurity master degree at Sapienza University

Group members: Noemi Giustini, Filippo Guerra, Filippo Olimpieri, Ludovica Garufi

Table of Contents

Table of figures

Table of tables

1. Introduction

In our study, we aim to conduct a comprehensive comparison between two biometric recognition methods: face recognition and iris recognition. These two modalities exhibit distinct characteristics, such as:

- usage,
- approaches,
- algorithms,
- use cases,
- performance metrics.

Iris recognition is a cutting-edge biometric technology that uses the unique and complex patterns of the human iris to identify individuals. This innovative approach to biometric security has gained significant attention for its high accuracy and reliability. In this project, we delve into the realm of iris recognition, exploring its fundamental principles and implementing the iris recognition using the Daugman method. The human iris exhibits intricate patterns that are highly distinctive among individuals; these patterns are formed by the arrangement of pigments, trabecular meshwork, and various other biological structures. Iris recognition focuses on capturing and analysing these unique patterns to establish a person's identity, offering a non-intrusive and efficient means of biometric authentication.

One of the key figures in the development of iris recognition technology is Dr. John Daugman, who Daugman method is named after. This method, introduced in the 1990s, involves encoding the iris patterns using a mathematical model known as the Gabor wavelet transformation. This method has proven to be highly accurate and robust, capable of handling variations in lighting conditions and other environmental factors. In our project, in order to obtain a better result, we decided to use LBF to extract the feature vectors.

Face recognition, has gained significant attention due to its wide-ranging applications in security and human-computer interaction. The ability to automatically identify and verify individuals based on face features has become an important technology in various industries.

Among the various solutions available, the one used in this project is the face_recognition Python library (Geitgey, 2017). This library leverages the powerful machine learning framework “dlib”, providing advanced tools for detecting, aligning, and recognizing faces in images. By utilizing the Histogram of Oriented Gradients (HOG) model, this library enables accurate detection of faces in images even under varying lighting and positional conditions.

2. Dataset

2.1. Dataset for Iris Recognition

The CASIA Iris Image Database (CASIA-Iris), developed by the CASIA research group, has been a valuable resource since 2002, evolving from CASIA-IrisV1 to CASIA-IrisV3. With over 3’000 users across 70 countries, this database has facilitated substantial advancements in iris recognition research. CASIA-IrisV4 is an extension that comprises six subsets, including CASIA-Iris-Interval, each tailored to address specific challenges.

The CASIA-Iris-Interval dataset offers a comprehensive collection of iris images, captured using our specialized CASIA close-up iris camera within indoor environments (Figure 1). This unique dataset consists of images taken during two distinct sessions for most subjects (Figure 2).

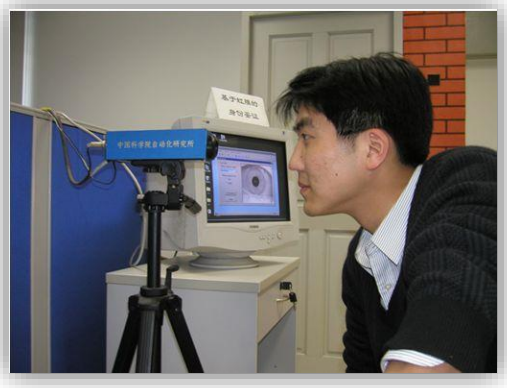


Figure 1: Iris capture method

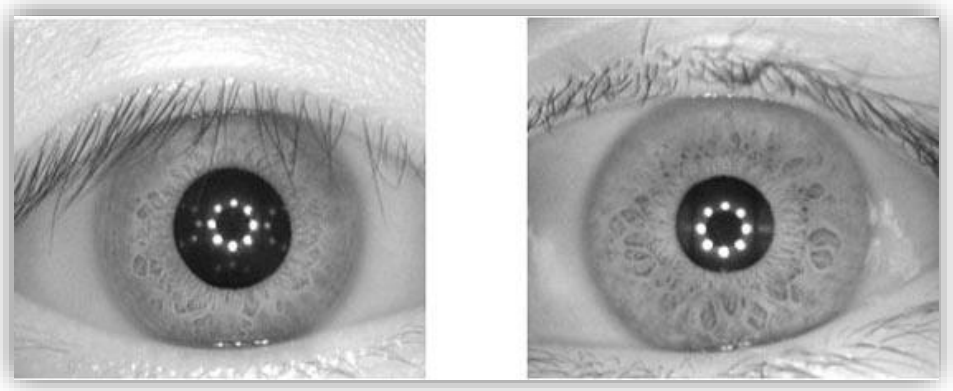


Figure 2: Eye capture samples

With a total of 249 subjects and a diverse array of 395 classes represented, the dataset provides a rich and fundamental resource for iris recognition research. In total, it encompasses 2,639 high-resolution images, each measuring 320 x 280 pixels. Notably, the dataset's key feature lies in its provision of cross-session iris images exhibiting remarkably clear texture details.

For CASIA-Iris-Interval, the images are stored with the following structure:

```
Datasets/CASIA-Iris- Interval/YYY/E/S2YYYENN.jpg
```

Where:

- **YYY** represents the unique identifier of the subject within the dataset,
- **E** indicates the captured eye, where **L** stands for the left eye and **R** for the right one,
- **S2** denotes the subset category,
- **YYYENN** specifies the subject class and image IDs. **YYY** is the subject ID and **NN** is the image index within the class.

This naming convention provides a structured and informative way to organize and identify individual images within the CASIA-Iris-Interval subset.

2.2. Dataset for Face Recognition

The CASIA Face Image Database Version 5.0 (CASIA-FaceV5) is a dataset comprising 2,500 coloured face images collected from 500 different subjects. These images were captured using a Logitech USB camera during a single session.

Each face image in the CASIA-FaceV5 dataset is stored as a 16-bit colour BMP file with a resolution of 640x480 pixels. This resolution allows for sufficient detail to be captured for face recognition tasks while maintaining manageable file sizes.

One of the key features of CASIA-FaceV5 is the inclusion of various intra-class variations commonly encountered in real-world scenarios. These variations include factors such as illumination, pose, expression, presence of eyeglasses, and other environmental conditions.

Figure 3: Face capture samples

The presence of intra-class variations enables researchers and developers to assess the robustness and generalization capabilities of face recognition algorithms. It facilitates the evaluation of algorithms under realistic conditions.

In CASIA-FaceV5, each image is uniquely named, and it is stored with the following structure:

```
/CASIA-FaceV5/SubjectID/SubjectID_ImageNumber.bmp
```

- '**CASIA-FaceV5**' represents the root directory where the CASIA-FaceV5 dataset is stored.
- '**SubjectID**' refers to the unique identifier assigned to each subject in the dataset. Subjects are numbered from 1 to 500.
- '**ImageNumber**' is the identifier assigned to each individual image captured for a particular subject. Images are numbered from 1 to 5, corresponding to the five images captured per subject during the single session.
- '**.bmp**' indicates that the images files are stored in BMP (Bitmap) format.

3. Approach

3.1. Approach for Iris Recognition

John Daugman is a researcher renowned for his significant contribution to iris recognition, particularly for developing the phase-coding method. Daugman's approach to iris recognition comprises several key phases, each playing a crucial role in the identification process. The main phases are as follows:

3.1.1. Segmentation

Daugman's approach incorporates a circular edge detector, an integro-differential operator, to localize both the pupil and the iris. This operator exploits the convolution of the image with a Gaussian smoothing function centered at R_0 with a standard deviation σ. It searches for a circular path where pixel variation is maximized, varying the center R and radius (x_0, y_0) of a candidate circular contour. A peak is expected when the candidate circle has the same radius and center as the iris.

3.1.2. Normalization:

After completing segmentation, it is necessary to normalize the iris image to account for scale and rotation variations. Daugman employs polar coordinates, transforming circular bands into horizontal stripes and the overall iris annulus into a rectangle. Determining the correct center for polar coordinates is crucial since the pupil and iris are not perfectly concentric. Additionally, the size of the pupil can change due to various factors, and gaze direction can alter the relative positions of the sclera, iris, and pupil. To address these challenges, Daugman introduced a normalization procedure known as the Rubber Sheet Model. The Rubber Sheet Model maps each iris point onto polar coordinates (r, θ) where $r \in [0, 1]$ and $\theta \in [0, 2\pi]$. It compensates for pupil's dilation and size variations, producing an invariant representation. The model does not compensate for rotations; however, during matching in polar coordinates, alignment is achieved by translating the obtained iris template. The transformation is defined by formulas involving a linear combination of points representing the pupil and external iris contour.

3.1.3. Coding:

Local Binary Pattern (LBP) is a popular texture descriptor for feature representation. The basic idea is as follows: for each pixel (p), its n-neighborhood pixels (usually n is 8) are thresholded into 1 or 0 by comparing them with the center pixel (pc). Then the binary sequence of the n-neighborhoods is transferred into a decimal number (bit pattern states with the upper left corner moving clockwise around the center pixel) and the histogram with 256 bins of the processed image is used as image feature. Furthermore, a kind of contrast measure referred to the central pixel can be computed, subtracting the average value of neighbors with a higher or equal value from the average value of neighbors with a lower value.

A pattern is called uniform when, if considered in a circular fashion, it contains at most two transitions 0-1 or 1-0. For example, the patterns 00011000, 00001111 are uniform. Uniform patterns are useful to save memory since they are only P×(P-1) +2 over a total of P^2.

In order to compute the histogram mentioned before the following steps are needed:

1. The image is partitioned into subwindows through a grid of k*k elements.
2. For each sub-window, a histogram is constructed in which each bin is associated with a specific pattern.
3. The final feature vector is obtained by concatenating the histograms calculated for all subwindows.

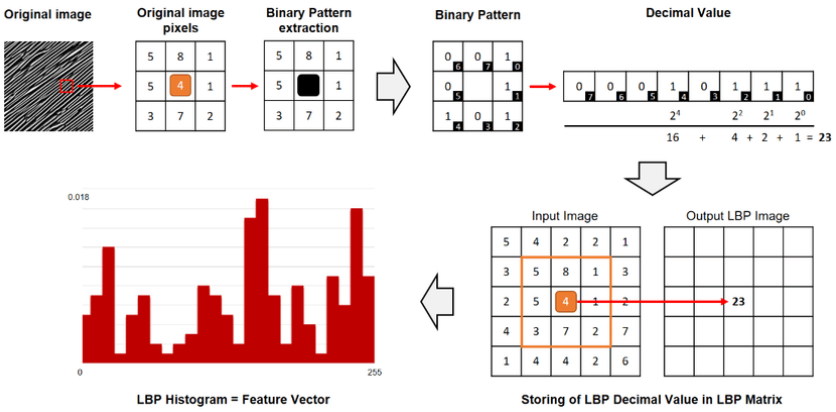


Figure 4: LBP calculus

3.1.4. Matching

The final phase involves comparing the acquired iris coding with previously stored encodings in the database. The comparison is made using the distance between feature vectors, where a smaller distance indicates greater similarity. Matching can be performed using various methods, such as Euclidean distance or Hamming distance.

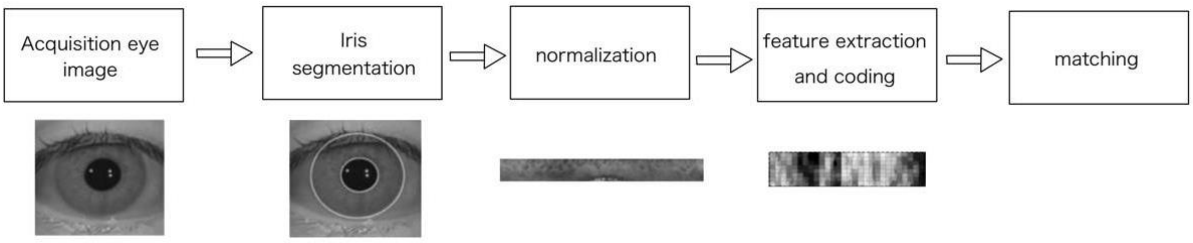


Figure 5: Phases for iris recognition

3.2. Approach for Face Recognition

In our project that implements face recognition, we utilize the face_recognition library in Python (Geitgey, 2017). This library performs face detection, face alignment, and recognition tasks. It's built

upon dlib, a high-performance machine learning library, making it efficient and accurate for identifying faces in images.

The face_recognition library, which indeed utilizes the Histogram of Oriented Gradients (HOG) based model for face detection, is a powerful tool in the field of computer vision. It provides functionalities for face detection and recognition. The Histogram of Oriented Gradients (HOG) is a feature descriptor used in computer vision and image processing for object detection. It counts occurrences of gradient orientation in localized portions of an image. Here's how it works in detail:

The image is divided into small connected regions called cells, and for the pixels within each cell, an histogram of gradient directions is compiled.

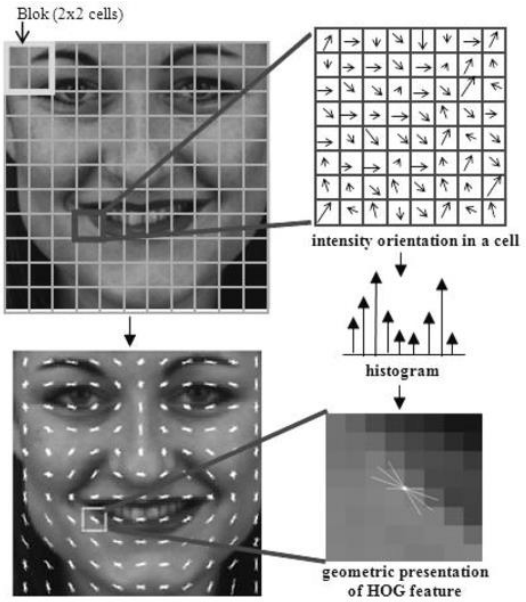


Figure 6: Histogram of Oriented Gradients

The descriptor is the concatenation of these histograms.

For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block. The HOG descriptor has some key advantages over other descriptors. Since it operates on local cells, it is invariant to geometric and photometric transformations, except for object orientation. Such changes would only appear in larger spatial regions.

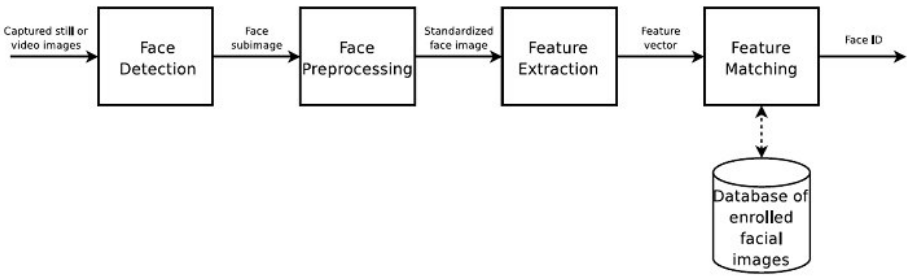


Figure 7: Phases for face recognition

4. Realization

4.1. Iris Recognition

To successfully recognize an iris multiple steps are required:

1. Acquisition of iris image,
2. Iris location and unwrapping,
3. Iris feature extraction,
4. Matching with a template database.

4.1.1. Acquisition

The image database used has been developed since 2002 by CASIA. In particular, CASIA-Iris-Interval was collected using specialized CASIA close-up iris camera within indoor environments. This unique dataset consists of images taken during two distinct sessions for most subjects.

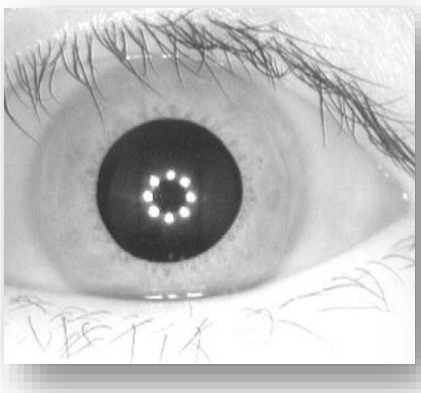


Figure 8: original image

4.1.2. Iris location

The first step to obtain a usable image of the iris is “iris location”. This process aims to find the pupil in a picture by preprocessing the said image, given as input. To find the pupil a threshold is used and an Erosion filter is applied to obtain (Figure 9).

These techniques are used to achieve a clear image of the pupil, needed to obtain the approximate pupil’s circle using the function HoughCircles. With this approach we are able to find both the pupil’s center and radius. Afterwards, the iris radius and its center are calculated based on the pupil’s ones. Finally, the iris circular crown is cut from the image with a mask (Figure 10).

Figure 9: Preprocessed image for pupil

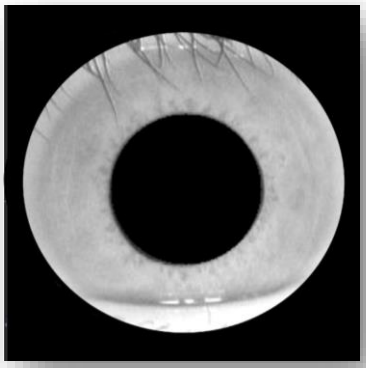


Figure 10: Iris localization

4.1.3. Iris unwrapping

With normalization, we ensure that, regardless of the original conditions of the iris image, the output will always have the same dimensions. We implement this using a polar coordinate model or a Daugman's Rubber sheet model, where the iris is "unrolled" onto a rectangle.

The variable angles represent the angle theta and is varied from 0 to 2π with 360 points, corresponding to a full circle around the iris, this determines the width of the unrolled image, while the variable radii represent the radius and is varied from the inner edge of the iris to the outer edge of the iris with 100 points, this determines the height of the unrolled image. Therefore, the unrolled image will have a dimension of 100x360 pixels.

The transformation of the iris image from a circular shape to a rectangular shape is done by converting the polar coordinates (radii, angles) into Cartesian coordinates and mapping the pixel values from the original image to the unrolled image (Figure 11).

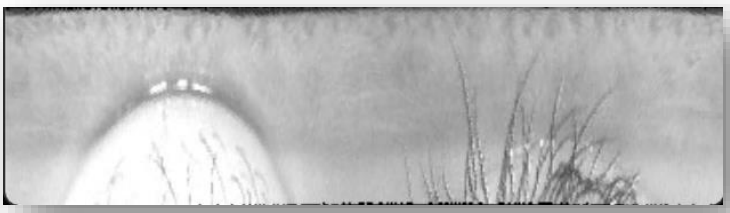


Figure 11: Unwrapped Iris

4.1.4. Iris imperfection's removal

After the normalization phase, we need to remove the eyelashes and the eyelid from the iris image. A white mask is generated to eliminate the eyelashes. This is accomplished by setting a threshold that only marks points significantly darker than the image's mean value as black. Subsequently, this mask is applied to the normalized image.

Then, we follow a similar process to remove the eyelashes, but this time we mark as black only those points that are significantly brighter than the image's mean value. Ultimately, this mask is applied to the normalized image.

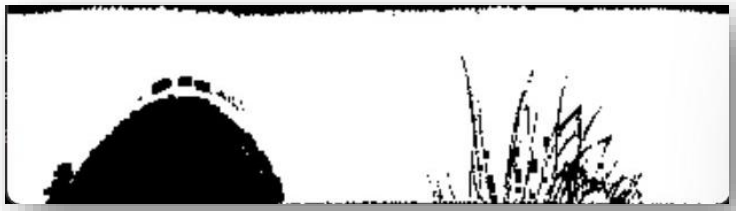


Figure 12: Eyelid and eyelashes mask

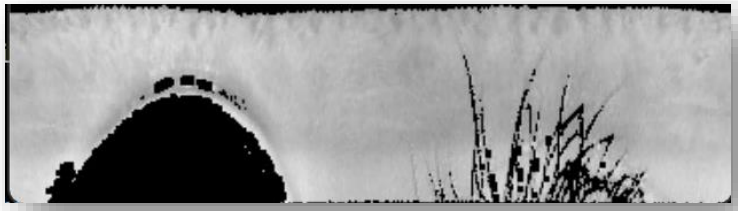


Figure 13: Normalized iris with mask

Finally, in order to emphasize the patter of the iris, an equalizeHist filter is used producing the following result (Figure 14):

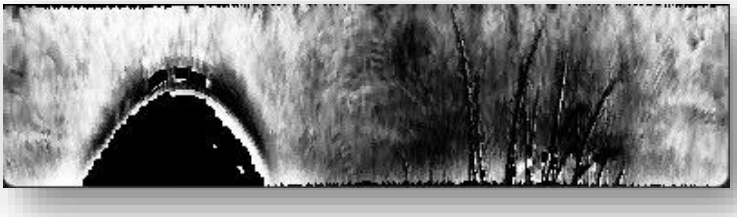


Figure 14: Normalized iris with equalizeHist filter

4.1.5. Iris feature extraction

Local Binary Pattern (LBP) is a widely used method for describing textures and features. It has been adapted for iris recognition by converting pixel neighbourhoods into binary sequences, then into decimal numbers. These numbers form histograms representing image features.

In our implementation, we divided the normalized iris into 20 horizontal bands and for each of them are calculated the LBP values and the corresponding histogram. Finally, all the histograms are concatenated together in order to create a representative feature vector.

4.1.6. Matching

To compare feature vectors extracted from iris images we use Euclidean Distance. Each dimension in these vectors corresponds to a feature, and the Euclidean distance gives a measure of how "close" the two feature vectors are to each other. A smaller Euclidean distance indicates a closer match between the feature vectors. In the context of iris recognition, this would mean that the two iris images are more likely to be from the same individual.

4.2. Face Recognition

The 'face_recognition' library provides high-level face recognition functionalities, allowing face features to be encoded into numerical vectors. The process begins by loading a dataset containing face images from a specified directory. The 'processDataset' function subdivides the dataset into two sets: the gallery set and the probe set. Subjects for the probe set are randomly chosen, ensuring a set of images unknown to the system (open set identification).

In the "main" method, for each individual in the gallery, the function encodes their face using the face_recognition.face_encodings function. This function takes an image as input and returns a 128-dimensional face encoding (a numerical representation of the face) for each face in the image. These encodings are then stored in the known_encodings list, and the corresponding names of the individuals are stored in the known_names list.

Then, for each individual in the probe set, the function encodes their face and then compares this encoding to the known encodings using the face_recognition.face_distance function. This function takes a list of known face encodings and a candidate encoding and returns a numpy array with the distance between the candidate encoding and each known encoding. Smaller the distance, more similar the faces are. Using a predefined threshold, the system decides whether the probe image matches a known subject in the gallery or not.

At the end, for each value of the threshold, some performance metrics are computed based on the results of face identification.

5. Comparison

To compare the biometrics system discussed in the previous sections we base our considerations on a series of parameters, such as: **Universality**, **Uniqueness**, **Permanence**, **Collectability**, **Acceptability**, **Performance**, **Speed**, **Template size**, **Accuracy**, **Hygienic level** and **Cost**. We present the said parameters in the following table.

Biometrics comparison	Face	Iris
Universality	**H**	**H**
Uniqueness	**L**	**H**
Permanence	**M**	**H**
Collectability	**H**	**M**
Acceptability	**H**	**L**
Performance	**L**	**H**
Speed	**M**	**M**
Template size	**H**	**M**
Accuracy	**M**	**H**
Hygienic level	**H**	**H**
Cost	**H**	**H**

Table 1: Iris vs Face recognition

As we can see Iris recognition has the greatest number of high positive values, in particular performance, universality and uniqueness. It however suffers from the lack of techniques that simplify the process of data collection, making collectability and acceptability major drawbacks. This is the reason why Iris is difficult to implement and roll to the public in great scale. Face, on the other hand, is easy to use and scale on large datasets, making it the obvious choice in scenarios where rigorous data collectability is not available. Furthermore, it is possible to reduce the data used in face recognition thanks to efficient compression algorithms; this however can only be used as long as face recognition is running on high performance machines or data centres.

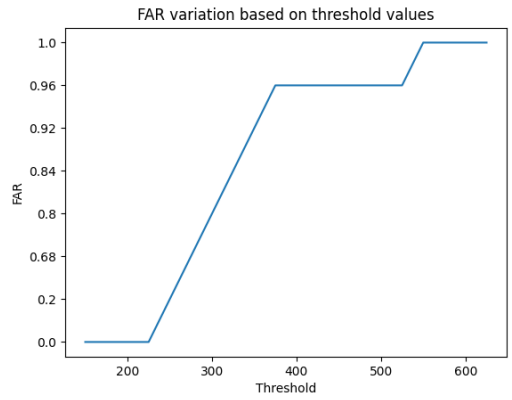
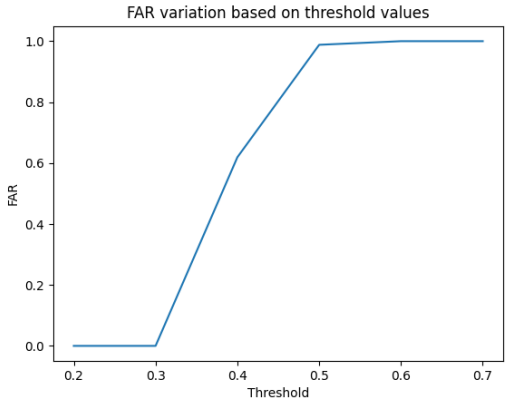
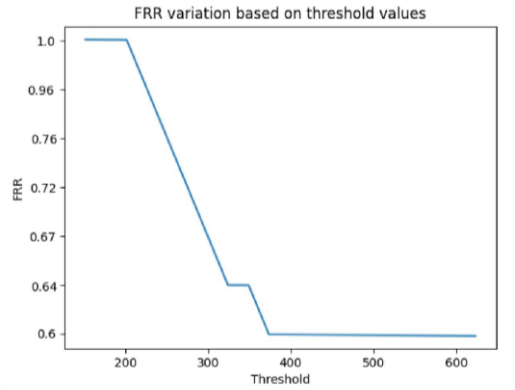
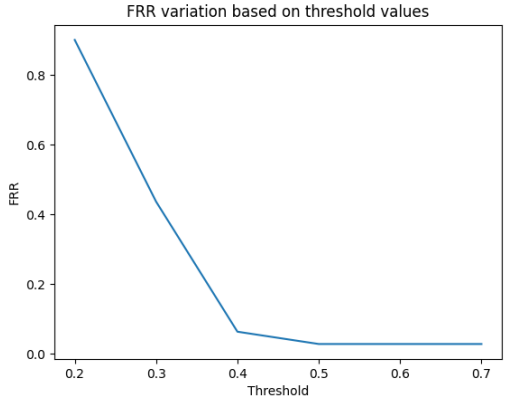
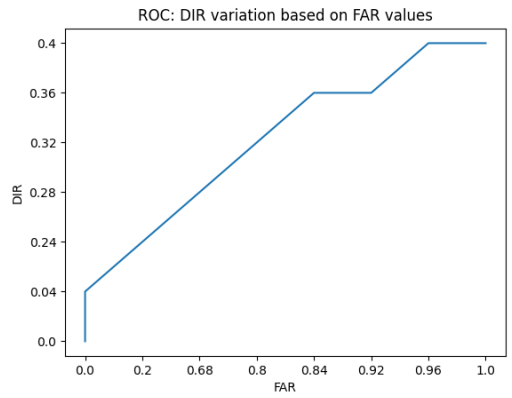
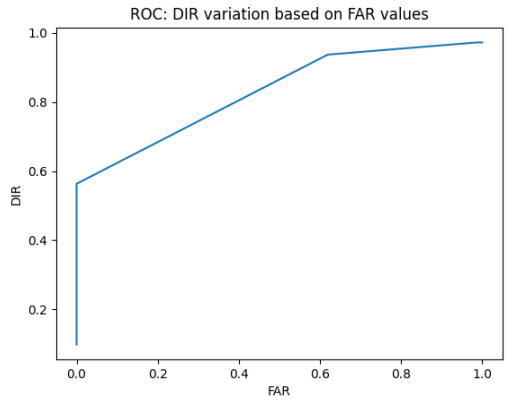
5.1. Performance evaluation

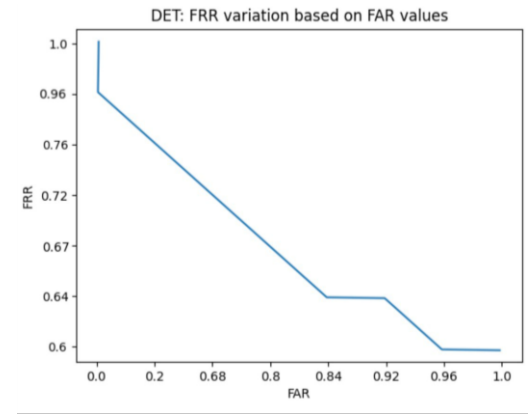
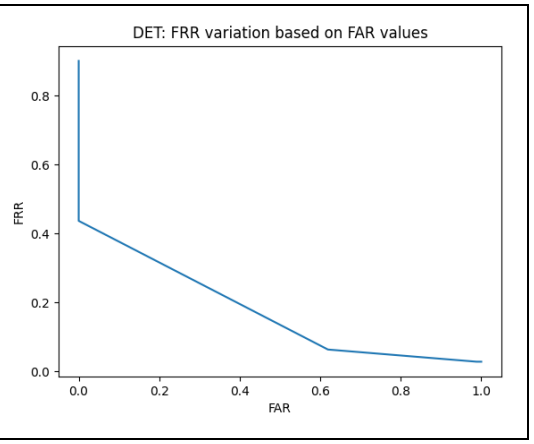
To evaluate the biometric system's performance, it has been decided to use crucial parameters on a large number of samples. These parameters are:

- DIR, acronym for detect and identification rate, represents the probability that there will be a correct identification within the k-th rank;
- FAR, acronym for false acceptance rate, represents the probability that an impostor will erroneously be accepted;
- FRR, acronym for false rejection rate, is the probability that a genuine sample will be wrongly rejected;
- GRR, acronym for good rejection rate, is the probability that an impostor is correctly rejected;
- GAR, acronym for good acceptance rate, is the probability that a genuine sample is correctly accepted.
- ROC curve, is the curve that shows the probability of DIR[0] versus FAR variations.

- DET curve, is the curve that shows the probability of FRR versus FAR variations.

Furthermore, for each of those parameters, different thresholds are used, obtaining the following results:

EVALUATION	IRIS RECOGNITION	FACE RECOGNITION
FAR		
FRR		
ROC curve		

DET curve		

All the previous charts are plotted thanks to the "matplotlib" (Hunter, 2007) python library.

```
------------------------------ Finished Image encoding -------------------------------
---------------------------- Started performance evaluation ----------------------------
DIR - (Detect Identification Rate) - is the probability that the subject is identified at rank i
FRR - (False Rejection Rate) - is the probability that a genuine subject is wrongly rejected
FAR - (False Acceptance Rate) - is the probability that an impostor is wrongly accepted
GRR - (Good Rejection Rate) - is the probability that an impostor is correctly rejected
GAR - (Genuine Acceptance Rate) - is the probability that a genuine subject is correctly accepted
DIR_List: [0.09848484848484848, 0.5631313131313131, 0.9368686868686869, 0.9722222222222222, 0.9722222222222222, 0.9722222222222222]
FRR_List: [0.9015151515151515, 0.43686868686868685, 0.06313131313131315, 0.02777777777777779, 0.02777777777777779, 0.02777777777777779]
FAR_List: [0.0, 0.0, 0.6190476190476191, 0.9880952380952381, 1.0, 1.0]
GRR_List: [1.0, 1.0, 0.38095238095238093, 0.011904761904761904, 0.0, 0.0]
GAR_List: [0.09848484848484851, 0.5631313131313131, 0.9368686868686869, 0.9722222222222222, 0.9722222222222222, 0.9722222222222222]
---------------------------- Performance evaluation finished ----------------------------
```

Figure 15: Face Identification performance evaluation

```
------------------------------ Finished Image encoding -------------------------------
---------------------------- Started performance evaluation ----------------------------
DIR - (Detect Identification Rate) - is the probability that the subject is identified at rank i
FRR - (False Rejection Rate) - is the probability that a genuine subject is wrongly rejected
FAR - (False Acceptance Rate) - is the probability that an impostor is wrongly accepted
GRR - (Good Rejection Rate) - is the probability that an impostor is correctly rejected
GAR - (Genuine Acceptance Rate) - is the probability that a genuine subject is correctly accepted
DIR_List: ['0.0', '0.0', '0.0', '0.04', '0.24', '0.28', '0.32', '0.36', '0.36', '0.4', '0.4', '0.4', '0.4', '0.4', '0.4', '0.4', '0.4', '0.4', '0.4', '0.4']
FRR_List: ['1.0', '1.0', '1.0', '0.96', '0.76', '0.72', '0.67', '0.64', '0.64', '0.6', '0.6', '0.6', '0.6', '0.6', '0.6', '0.6', '0.6', '0.6', '0.6', '0.6']
FAR_List: ['0.0', '0.0', '0.0', '0.0', '0.2', '0.68', '0.8', '0.84', '0.92', '0.96', '0.96', '0.96', '0.96', '0.96', '0.96', '0.96', '1.0', '1.0', '1.0', '1.0']
GRR_List: ['1.0', '1.0', '1.0', '1.0', '0.8', '0.31', '0.19', '0.16', '0.07', '0.04', '0.04', '0.04', '0.04', '0.04', '0.04', '0.04', '0.0', '0.0', '0.0', '0.0']

---------------------------- Performance evaluation finished ----------------------------
```

Figure 16: Iris Identification performance evaluation

A comparative analysis has revealed better performance in face recognition compared to iris recognition. This discrepancy can be attributed to various factors:

- *Utilization of Machine Learning Models*: Face recognition employs machine learning models, including deep learning algorithms developed and optimized by experts. These approaches integrate advanced algorithms that effectively address challenges such as intra-class variation and lighting conditions.
- *Intra-class Variation in the Iris Dataset*: The dataset utilized for iris recognition exhibits considerable intra-class variation, as the iris images of the same individual may significantly differ due to various environmental factors such as lighting, eye position, and acquisition quality. This is confirmed considering that the dataset contains iris captures from different sessions for the same user. Such additional variation complicates the task of iris recognition algorithms in accurately distinguishing individuals.
- *Variability in Iris Pattern and Texture*: The iris can demonstrate notable variation in both pattern and texture, even within the same individual.

- Iris recognition necessitates consideration of *different factors* such as pupil dilation or eye position, which may be more challenging to measure and standardize compared to face features.
- *Availability and Diversity of the Dataset*: The dataset used for face recognition often encompasses a wide variety of images from different individuals, allowing models to learn a vast range of distinctive patterns and features, unlike in iris recognition. The iris recognition dataset contains only 250 individuals, whereas the face recognition dataset contains 500 individuals.
- *Complexity of Iris Segmentation*: Iris identification and segmentation can be particularly complex due to occlusion, caused by eyelids and eyelashes, requiring additional efforts to mitigate such interferences.
- *Distinctiveness of Face Features*: face features such as eyes, nose, and mouth tend to be more distinctive compared to iris characteristics. This means that face recognition systems may have access to a greater amount of information to distinguish with between individuals compared to iris recognition. In fact the iris, unique to each person, may not be equally distinctive in terms of visible characteristics.

It is noteworthy that, despite the efforts made in iris recognition through the utilization of various distance metrics, such as Hamming and Euclidean distance, and feature extraction methods like PCA, LBP, Gabor kernel, and wavelet transform, the obtained results showed only marginal variations in performance. Moreover, while face recognition may be influenced by factors such as aging, pose, illumination, and expression (A-PIE), such challenges are managed more effectively compared to the specific complexities of iris recognition.

6. Demo of Face Re-identification

Face recognition is one of the most interesting and widely used technologies in the field of computer vision. As part of our project, we have developed a face recognition demo that utilizes the computer's built-in webcam as the input source. Re-identification, often shorted as ‘re-ID’, aims to match a person’s identity across different video capture hardware, such as video cameras, and different locations in a video stream or an image sequence. This process involves detecting and tracking an individual and its actions. Features such as appearance, body shape or behaviour are used to determine the presence the same *identity* in different frame at different time. The final goal is to associate the same person across multiple non-overlapping time slices.

In the following chapter, we will delve into the implementation details of this demo, examining the techniques and algorithms used to identify and recognize human faces in real-time.

6.1. Approach

The Viola-Jones algorithm is a seminal image-based face recognition approach that has gained significant traction for its efficiency and accuracy. Developed by Paul Viola and Michael Jones in 2001, this algorithm is particularly renowned for its rapid face detection capabilities, making it a milestone in computer vision applications. The Viola-Jones algorithm operates by utilizing a cascade of classifiers to identify regions of interest within an image. It leverages Haar-like features, integral images, and an adaptive boosting (AdaBoost) technique to create an effective face detection system.

Haar-like features are simple rectangular filters that can be applied to an image. These features capture variations in pixel intensities and are instrumental in identifying distinct patterns associated with face features. These features are difficult to calculate, for this reason, Viola-Jones employs integral images. These are precomputed representations of the original image, facilitating rapid calculation of Haar-like features at any given location with constant time complexity. During training, Haar-like features are evaluated on each image to determine which features are most discriminative for face detection.

AdaBoost is a machine learning technique used for training a strong classifier by combining multiple weak classifiers. In the context of Viola-Jones, weak classifiers are trained to evaluate specific Haar-like features, and AdaBoost assigns weights to these classifiers based on their performance. The resulting classifiers are organized into a cascade, where each stage aims to efficiently reject non-face regions thanks to a threshold. This cascade structure enables faster processing, as non-relevant regions are eliminated at each stage. If a region passes all stages, it is classified as a face.

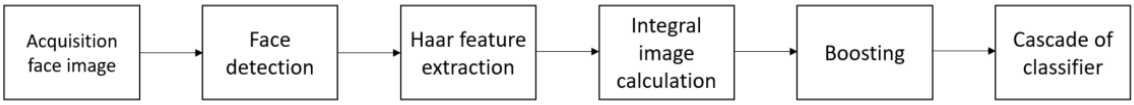


Figure 17: Phases for face recognition

6.2. Dataset for demo of Face Recognition

In our project, the dataset for training the face recognition model is automatically generated through the enrolment phase. The process can be described as follow:

The enrolment phase is a crucial step in building a face recognition system. During this phase, individuals who will be part of the system are registered or enrolled. The primary purpose is to capture face data and create a personalized template or representation for each individual.

The enrolment process involves capturing face images of individuals using the webcam of the computer. As individuals go through the enrolment phase, the system accumulates a dataset. Each entry in this dataset corresponds to a unique individual and includes their personalized face template. The dataset is automatically structured and labelled, associating each template with the respective individual's identity tanks to a unique ID.

In our project, the dataset located in the folder “dataset” contains photos that are renamed according to a specific format during the enrolment phase. The naming convention is structured as follows:

```
datasets/User.<id>.<count>.<name>.jpg
```

- “User” is the prefix that helps identify that the file corresponds to a user within the dataset;
- “id” represents the unique identifier associated with the user's identity. In real-world scenarios, this could be a unique identification number like a social security number, an employee ID, or any other identifier that distinguishes one individual from another;
- “count” indicates the sequence number of the photo for a specific user. It helps differentiate between multiple photos associated with the same user;
- “name” is the name of the user, and it may not necessarily be unique. It provides additional context information about the individual.

6.3. Realization

The Python code which we propose, utilizes the OpenCV library to capture video from the webcam and gather data for creating a dataset for face recognition. It can be divided in the following steps:

It starts by opening the video stream from the webcam using ‘cv2.VideoCapture(0)’. The video stream is captured frame by frame in the main program loop. Next, a ‘CascadeClassifier’ object is instantiated with the pre-trained XML file “haarcascade_frontalface_default.xml”. This classifier is used to detect faces in the current frame.

The user is prompted to enter an ID and a name associated with the face about to be captured. These values are used to name the dataset files. The while loop continues indefinitely until a sufficient number of faces, 500 by default, are captured to constitute a reliable dataset. For each detected face, the grayscale image is cropped and saved in the 'datasets' folder with a unique name based on the ID, count and user's name.

The detected faces are also outlined by a red rectangle in the real-time displayed frame to provide visual feedback to the user. At this point, the video is released, and all windows are closed. A message confirming the completion of the data collection is displayed. The “training.py” code serves the purpose of training a face recognition model utilizing the LBPH (Local Binary Pattern Histogram) algorithm through the OpenCV library.

The ‘getImageID’ function is defined to extract image IDs and face data from the dataset. It reads each image file from the specified directory, converts the image to grayscale, and extracts the ID and name information from the file name. The function also displays the training images using OpenCV. This function is called and the obtained data, composed by the said images, are used to train the LBPH face recognizer with the ‘recognizer.train’ method. The trained model is saved as a YAML file named “Trainer.yml”, using the function ‘recognizer.write()’.

After completing the training process, all OpenCV windows are closed ‘cv2.destroyAllWindows()’, and a message is printed to indicate the completion of the training.

The file “testmodel.py” Python code implements a real-time face recognition system using OpenCV and the LBPH (Local Binary Pattern Histograms) face recognizer.

At the beginning the video capture is initialized from the default camera, an LBPH face recognizer is created, and a pre-trained model is loaded from a file named “Trainer.yml”.

The code enters a loop that continuously captures video frames, performs face detection, and recognizes faces. Faces are detected in the current video frame using the Haar Cascade classifier. The code iterates through the detected faces and uses the LBPH recognizer to predict the identity of each face. If the confidence is above 60, the face is considered recognized, and the person's name is displayed along with a blue rectangle around the face. If the confidence is 60 or below, the face is labeled as “Unknown” and displayed with a red rectangle.

At the end the video capture resources are released ‘video.release()’, and all OpenCV windows are closed ‘cv2.destroyAllWindows()’

Bibliography

Aswis, A. E., Morsy, M., & Abo-Elsoud, M. E. (2015, June). Face Recognition Based on PCA and DCT Combination Technique. *International Journal of Engineering Research and Technology (IJERT)*. Retrieved from https://www.ijert.org/research/face-recognition-based-on-pca-and-dct-combination-technique-IJERTV4IS060723.pdf

Boisberranger, D. J., Bossche, J. V., Estève, L., Fan, T. J., Gramfort, A., & Grisel, O. a. (2010, February 1). Scikit-learn. Retrieved from https://scikit-learn.org

CASIA. (2016, October 26). CASIA-FaceV5. Retrieved from http://english.ia.cas.cn/db/201610/t20161026_169405.html

CASIA. (2020, 07 13). *CASIA-IrisV4.* Retrieved from hycasia.github.io: https://hycasia.github.io/dataset/casia-irisv4/

CASIA. (n.d.). CASIA Dataset. Beijing, China: CASIA. Retrieved from http://biometrics.idealtest.org/downloadDB.do

CASIA. (n.d.). CASIA-IrisV4 Dataset. *CASIA-IrisV4*. Beijing, China: CASIA. Retrieved from http://www.cbsr.ia.ac.cn/china/Iris%20Databases%20CH.asp

Dixit, A. J., & Kazi, K. S. (2015). IRIS Recognition by Daugman‟s Method. *International Journal of Latest Technology in Engineering,*, 93.

Egorov, A., Shtanko, A. N., & Minin, P. (2015). Selection of Viola–Jones algorithm parameters for specific conditions. *Bulletin of the Lebedev Physics Institute*. doi:10.3103/S1068335615080060

GeeksForGeeks. (2015, May). *geeksforgeeks*. Retrieved from Boosting in Machine Learning | Boosting and AdaBoost: https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/

Geitgey, A. (2017, March 4). Face Recognition. Retrieved from https://pypi.org/project/face-recognition

Greche, L., & Es-Sbai, N. (2016, March). Automatic system for facial expression recognition based histogram of oriented gradient and normalized cross correlation. *International Conference on Information Technology for Organizations Development (IT4OD)*, 5. doi:10.1109/IT4OD.2016.7479316

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering, 9*, 95. doi:10.1109/MCSE.2007.55

Müller, M., Britz, D., Ulrich, L., Staudt, T., & Mücklich, F. (2020, May 12). Classification of Bainitic Structures Using Textural Parameters and Machine Learning Techniques. p. 19. doi:10.3390/met10050630

OpenCV. (2022, January). *Face Detection using Haar Cascades*. Retrieved from Open Source Computer Vision: https://docs.opencv.org/3.4/d2/d99/tutorial_js_face_detection.html

Sari, Y., Alkaff, M., & Pramunendar, R. A. (2018, June 26). Iris recognition based on distance similarity and PCA. *AIP Conference Proceedings, 1977*. doi:10.1063/1.5042900

Tyagi, M. (2021, July 4). *HOG (Histogram of Oriented Gradients): An Overview*. Retrieved from Towards Data Science: https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f

Wikipedia. (2008, May). *Histogram of oriented gradients*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients