SAPIENZA
UNIVERSITÀ DI ROMA

Biometrics Systems project

Password manager secured by iris recognition

Teacher: Maria De Marsico

Student: Riccardo Scuto 2125102

Summary

1. Preface

This project is based on Iris-Recognition-Registration-Database-System by aaahmedms.[1]

- **Additions:** Iris Segmentation; Iris Unwrapping; Password Manager; Blink Counter System Verification.
- **Modifications:** User Interface; Matching Algorithm

1.1 Project Objective

This project aims to create a password manager protected by an iris recognition system. Utilizing advanced image processing and computer vision technologies, the system is capable of capturing, processing, and comparing iris images.

1.2 System Overview

The iris recognition system comprises two main components: registration and verification.

- **User Registration:** This process begins with the user entering personal information (such as name, ID, age) through a GUI. Following this, the system captures an image of the user's eye using a video input (in our case, a smartphone camera connected through Iruin Webcam) [2].

  Once the iris is identified according to the conditions defined by the code, the system performs the iris unwrap process and saves the normalized image in the database, which will then be used for verification purposes. After registration, an internal text editor within the program will open, allowing the user to write their passwords.

- **User Verification:** In the verification process, a new iris scan is performed, and it asks for the database ID for which a match is being sought. If a correct match rate is achieved, the correct correspondence is obtained, and access to the passwords entered during registration is granted.

2. Theoretical Foundations

2.1 Iris Recognition

The iris is characterized by its variable color, diaphragm-like shape, and function, it is pigmented and located between the cornea and the lens, with a pupil at its center. The iris comprises a layer of muscle fibers around the pupil, smooth muscle fibers for pupil dilation (controlling light entry), and two layers of pigmented epithelial cells. Its unique color, texture, and patterns (e.g., freckles and crypts) provide a high level of discrimination, akin to fingerprints.

Pros of using iris recognition include its visible yet protected location, time invariance after infancy. It's randotypic, so it is not affected by family inheritance (uniqueness even among twins). It can be captured using both near-infrared and visible wavelengths. Capture modalities involve visible light, which reveals the iris's intricate texture despite noisy information, and infrared light, which enhances texture visibility and is preferred in iris-based biometric systems, albeit requiring specialized equipment.

The iris recognition process entails pre-processing for noise reduction and segmentation to isolate iris pixels, followed by normalization (unwrapping), coding (feature extraction), and matching to verify identity.

2.2 Image Processing and Computer Vision

Image processing and computer vision are crucial for iris recognition, allowing the system to interpret and analyze eye images. Techniques such as edge detection and segmentation are used to isolate the iris from other parts of the eye. Normalization and transformation algorithms standardize iris images, enabling accurate comparison between different captures. These processes leverage advanced machine learning and pattern recognition methods to identify an individual's unique iris characteristics reliably and precisely.

Within the context of this project, the integration of these technologies enables the creation of a robust and versatile iris recognition system, suitable for a wide range of applications, from access security to identity verification in digital services.

3. Technologies and libraries

3.1 Python

Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. It's widely used for web development, data analysis, artificial intelligence, scientific computing, and automation due to its extensive standard library and the availability of a vast ecosystem of third-party packages.

3.2 PyQt5

PyQt5 is a set of Python bindings for Qt libraries that can be used to create cross-platform applications with graphical user interfaces (GUIs). In this project facilitates the creation of forms for user registration and verification, displaying live camera feeds, and presenting informational dialogs and messages to the users.

3.3 OpenCV

OpenCV is an open-source computer vision and machine learning software library. It provides a vast array of functions that are used for image processing, including capturing video, converting images to different color spaces, detecting edges, and more. In the context of iris recognition, OpenCV is utilized for the initial capture of the eye image, pre-processing steps such as resizing and normalization, and the extraction of iris features. Its powerful functions for image manipulation and analysis are instrumental in processing and preparing the iris images for recognition.

3.4 Dlib

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software to solve real-world problems. It includes a wide range of functionalities, notably for facial feature detection.

3.5 PIL

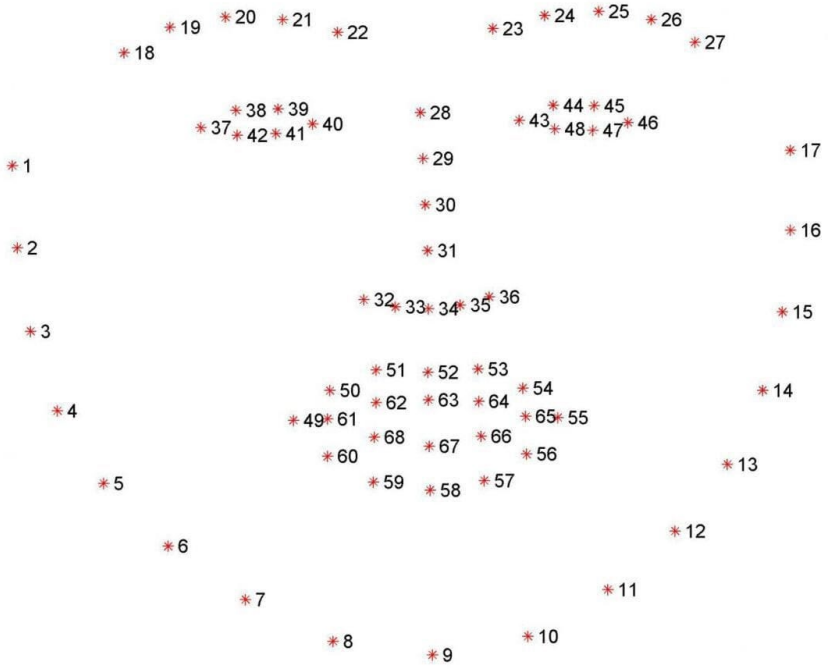
A Python library that adds support for opening and manipulating images in various formats. It's used for image operations like format conversions.

3.6 Numpy

A library for supporting large arrays and matrices, along with a collection of mathematical functions to operate on these data structures. It is extensively used for manipulating image data.

4. Dataset

The "shape_predictor_68_face_landmarks.dat"[3] file is a pre-trained model used in computer vision, particularly with the Dlib library, for detecting facial landmarks. Facial landmarks are key points on a face that represent significant areas, such as the contours of the eyes, eyebrows, nose, mouth, and jawline. This model identifies 68 specific landmarks that cover the face's main features, providing a detailed map of the facial structure. These landmarks are instrumental in various applications, including facial recognition, emotion analysis, and augmented reality, as well as for tasks like aligning faces in photographs.



Visualizing the 68 facial landmark coordinates from the iBUG 300-W dataset.[3]

4.1 Role in Iris Recognition Systems

In the context of an iris recognition system, the "shape_predictor_68_face_landmarks.dat" file plays a crucial role in the initial stages of iris detection. By accurately mapping the facial landmarks, the system can efficiently locate the eyes within a captured image. This precise eye detection is critical for isolating the iris, the next step in the recognition process. By knowing the exact position and boundaries of the eyes, the system can focus on the iris for further processing, such as segmentation, normalization, and feature extraction, enhancing the overall accuracy and reliability of the iris recognition.

4.2 Integration with Iris Recognition Workflow

The integration of the "shape_predictor_68_face_landmarks.dat" model into the iris recognition workflow involves several steps:

- **Face Detection**: The system first detects the face(s) in the captured image using Dlib's face detection capabilities.
- **Landmark Detection**: Once a face is detected, the system uses the "shape_predictor_68_face_landmarks.dat" model to identify the 68 facial landmarks, focusing particularly on those around the eyes.
- **Eye Region Isolation**: Using the landmarks corresponding to the eyes, the system isolates the eye regions, preparing them for iris detection and analysis.

5. System Architecture

5.1 User Interaction

Application Initialization: The graphical user interface is prepared with functionalities for user registration, verification, and password management.

User and Iris Registration: During registration, users provide personal details and undergo an iris scanning process. These details, along with a new password, are encrypted and stored locally.

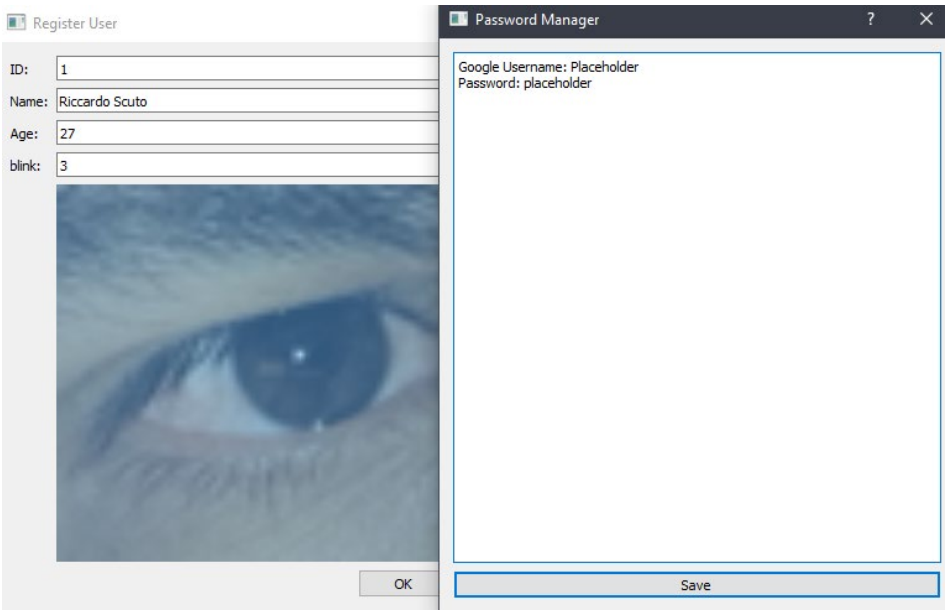
Blink definition: As part of the registration, there's a "blink" field that users must set, indicating the number of blinks required to access their passwords after successful iris match. This feature serves as an anti-spoofing measure, considering potential errors due to low image quality and attempts to deceive the system. After a positive iris match, users must perform the specified number of blinks to access their data.

User Verification

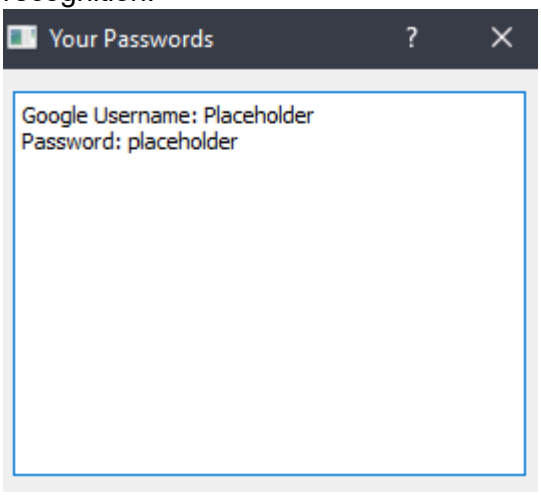
Users attempting to log in are verified through iris scanning. If verification is successful, they can access their stored passwords. The viewable passwords are those previously encrypted and stored, now decrypted for display.

Password Management

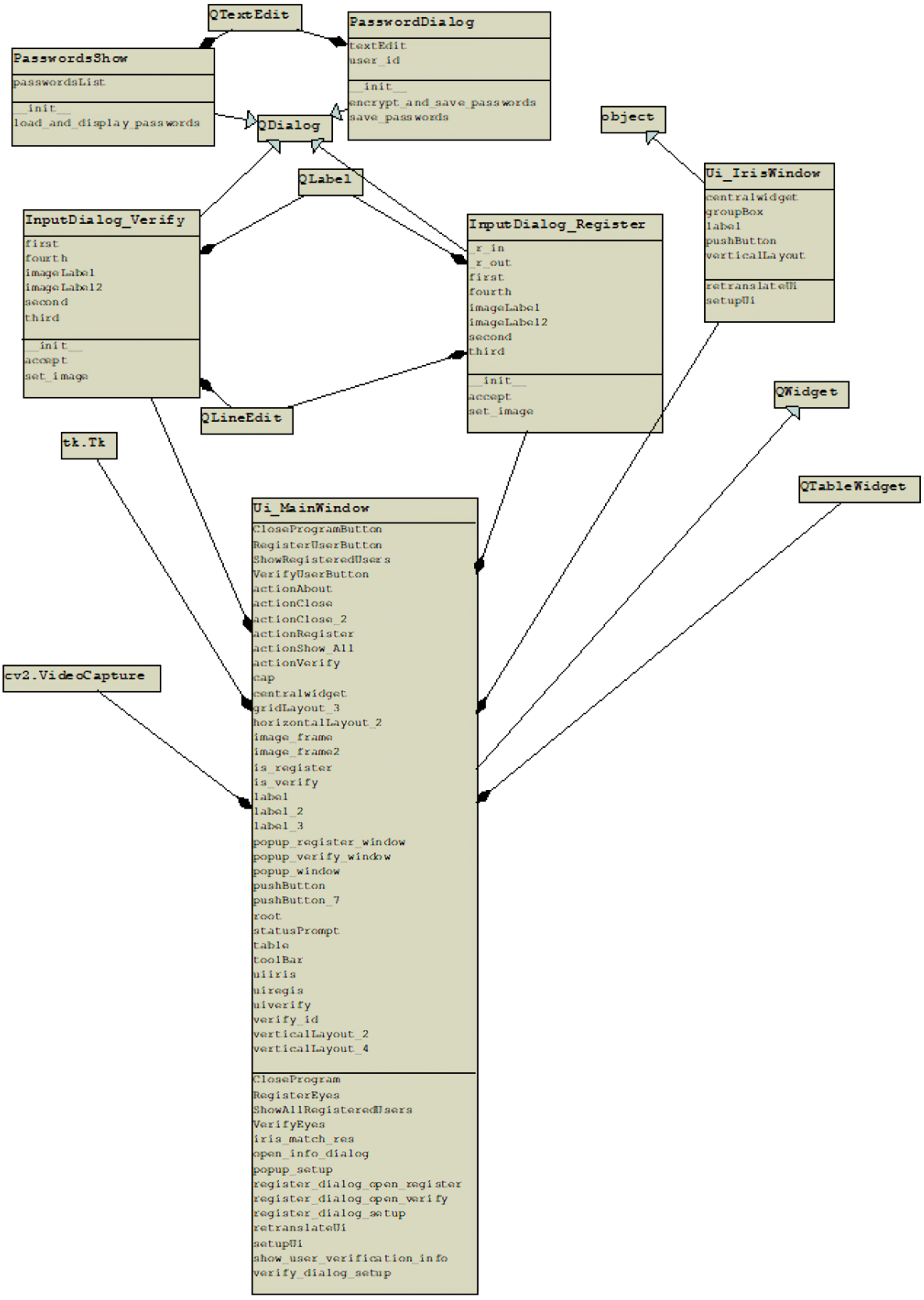
- **Saving**: After registration, users can save new passwords, which are encrypted and associated with their user ID.



- **Viewing**: Stored passwords can be viewed after authentication through iris recognition.



5.2 Code Structure



UML

6. Implementation

6.1 Circle Detection Using Hough Transform

The detection of the iris begins with identifying the circular edge of the iris itself. This is done using the Hough Transform for circles, an algorithm capable of detecting circular shapes in an image. Key parameters for the Hough Transform include the minimum and maximum radius of the circle to detect, which are set based on the expected size of the iris in the image. This step returns the circle's center coordinates (x, y) and radius (r) of the iris.

6.2 Iris Segmentation

Once the iris circle is identified, segmentation proceeds, which isolates the iris from the rest of the eye image. This is achieved by creating a binary mask with a filled (white) circle at the iris position and the rest of the image set to black. Applying this mask to the original eye image results in the isolated iris with the rest of the image darkened.

```python
def get_iris(frame):
        iris = []
        copy_img = frame.copy()
        res_img = frame.copy()
        gray_img = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
        mask = np.zeros_like(gray_img)
        edges = cv.Canny(gray_img, 5, 70, 3)
        circles = get_circles(edges)
        iris.append(res_img)
        for circle in circles:
            rad = int(circle[0][2])
            global radius
            radius = rad
            cv.circle(mask, centroid, rad, (255, 255, 255), cv.FILLED)
            mask = cv.bitwise_not(mask)
            cv.subtract(frame, copy_img, res_img, mask)
            x = int(centroid[0] - rad)
            y = int(centroid[1] - rad)
            w = int(rad * 2)
            h = w
            crop_img = res_img[y:y + h, x:x + w].copy()
            return crop_img
        return res_img
```

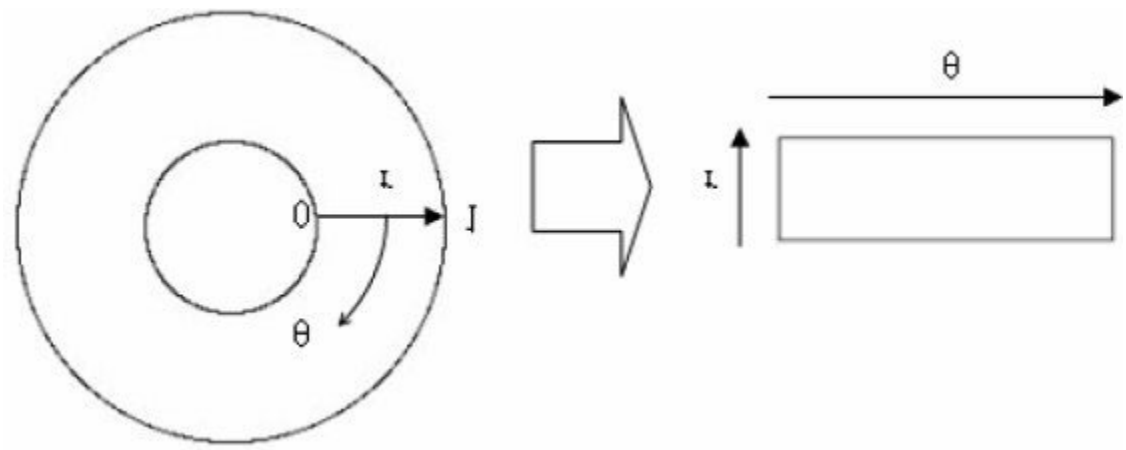
6.3 Preprocessing and Normalization

The segmentation of the iris is followed by a transformation from Cartesian to polar coordinates. This transformation is essential for normalizing the iris image, thereby enhancing the robustness of the recognition system against variations in iris scale,

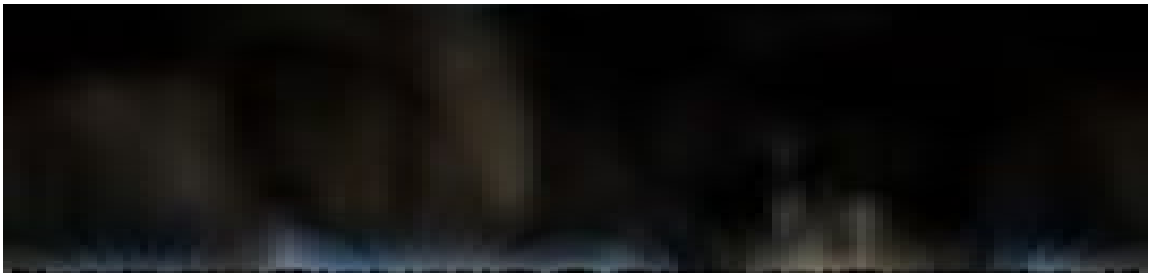
orientation, and camera angle. The conversion to polar coordinates significantly aids in the uniform extraction of features, which is pivotal for subsequent comparison processes.

Once in polar coordinates, the iris image undergoes normalization. This standardization is a critical procedure that resizes the iris to a consistent dimension, mitigating the discrepancies caused by different distances between the eye and the camera. Such normalization is instrumental in bolstering the accuracy and dependability of the iris recognition process."

```python
def daugman_normalization(image, center, r_in, r_out, width, height):
    thetas = np.linspace(0, 2 * np.pi, width)
    r = np.linspace(r_in, r_out, height)
    theta, radius = np.meshgrid(thetas, r)
    X = center[0] + radius * np.cos(theta)
    Y = center[1] + radius * np.sin(theta)
    polar_to_cartesian = cv.remap(image, X.astype(np.float32), Y.astype(np.float32), cv.INTER_LINEAR)
    return polar_to_cartesian
```



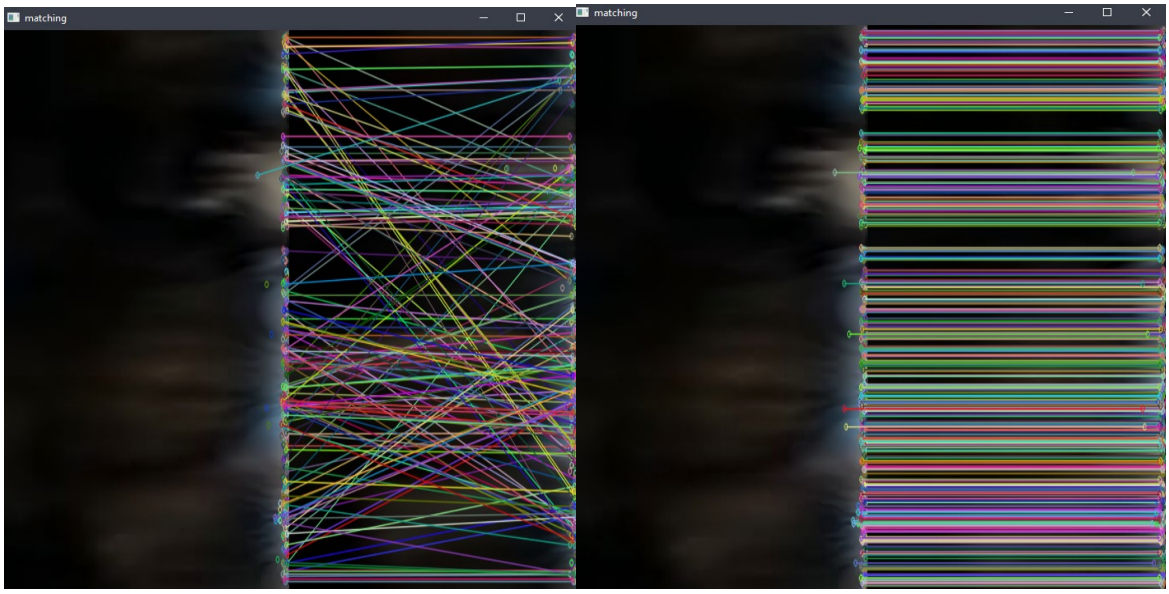
Rubber Sheet model



Normalized Iris

6.4 Matching

The verification process leverages the ORB feature matching algorithm. During verification, the user's iris image, captured in real-time, is processed using the `iris_match_res()` method alongside a previously registered iris image associated with the user ID provided. The ORB algorithm employs brute force matching to determine if there is a correspondence between the live and registered iris images. If a match is identified, the registered user's details are displayed in a dialog box. The threshold for determining a match may need adjustment based on the quality of the camera used. For higher-resolution cameras, a match rate of over 50% could be adequate, whereas for lower-quality devices, a lower threshold around 18-25% might be more appropriate.



Match Rate: 35.4% *Match Rate: 100%*

7.Blink check

Once a match between the iris in the database and the one captured during the verification process is found, the system proceeds to verify if the number of blinks performed for confirmation matches the number registered in the database. The script initializes variables

for counting blinks, sets a threshold for detecting blinks, and uses flags to track whether the eyes are open or closed. Additionally, it employs dlib's pre-trained face detector and facial landmark predictor to identify facial structures in video frames.

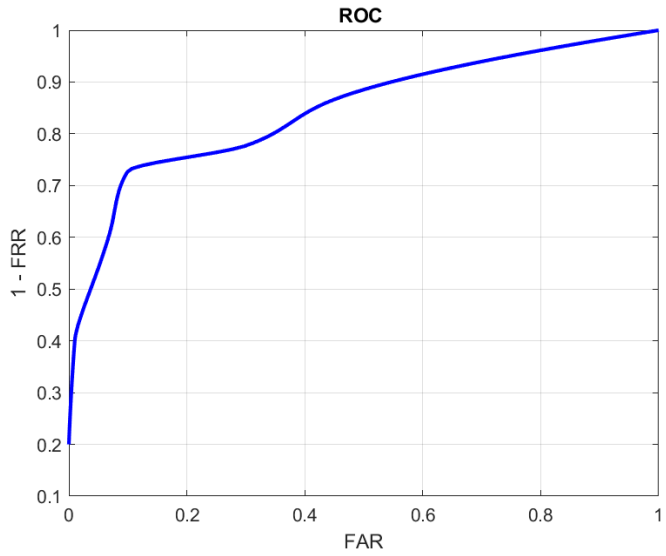
For each detected face, the script calculates the Eye Aspect Ratio (EAR) for both eyes. The EAR is a straightforward geometric measure that gauges the openness of the eyes. When the EAR drops below a predefined threshold, it is interpreted as a blink.

The number of detected blinks is displayed on the video frame, along with instructions for user interaction. Users can interact with the system through keyboard inputs:

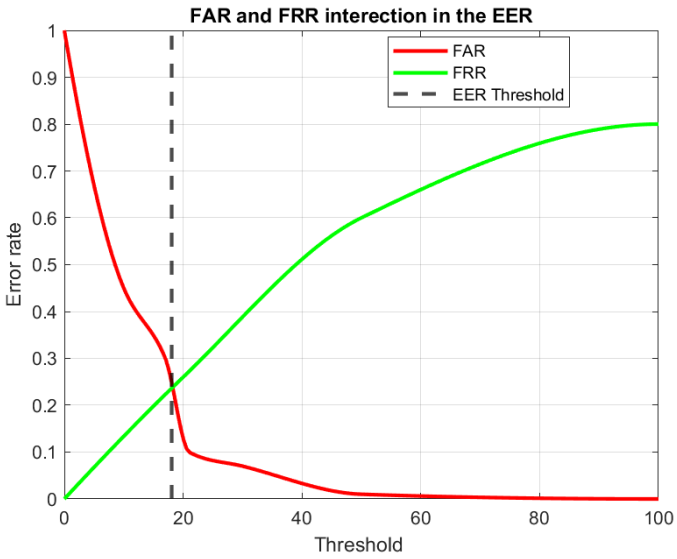
- Pressing "a" verifies whether the number of detected blinks matches the required number for a specific authentication or confirmation process, ending the loop and returning `True` or `False`.
- Pressing "w" resets the blink counter to zero.

8. Evaluation

The test results were extracted from an analysis conducted on a dataset of 20 images taken with a smartphone camera. To ensure data reliability, the lighting conditions were kept constant throughout the testing process.



The ROC curve demonstrated a sharp ascent at the beginning, indicating that a significant number of genuine images were accurately recognized while maintaining a low FAR. However, the curve showed a gradual flattening as the FAR increased, suggesting that the number of true positive identifications (1 - FRR) began to plateau. Ideally, for optimal performance, the curve would approach the upper left corner of the graph, signifying a high rate of correct identifications with minimal false acceptances.



At lower threshold settings, the system exhibited a marked inclination to erroneously authenticate non-matching iris images, thus raising security concerns. Raising the threshold enhances the system's security as evidenced by a falling FAR. This increase in security comes with a trade-off in terms of user convenience due to the rising FRR. The intersection of the FAR and FRR curves, which occurs between match rates of 0.2 and 0.3, represents the EER. This intersection indicates an optimal balance between maintaining system security and ensuring user accessibility.

9. Limitations and Challenges

Iris recognition technology, celebrated for its high accuracy and reliability in biometric authentication, does have its set of technical limitations and challenges. These constraints can affect its implementation and effectiveness in various contexts, from security systems to personal device access.

Environmental and Lighting Conditions: Iris recognition systems are highly sensitive to lighting conditions. Too much or too little light can lead to poor image quality, making iris pattern extraction difficult. Inconsistent lighting conditions can result in reflections, shadows, and glare, which may obscure the iris details necessary for accurate identification.

High-Quality Imaging Requirements: For iris recognition systems to be effective, high-resolution images are essential. This necessitates the use of specialized cameras capable of capturing detailed iris patterns, even from a distance. Such equipment can be costly and may not be practical for all applications, especially those requiring rapid scanning in public or busy environments.

Physical and Biological Factors: The effectiveness of iris recognition can be affected by biological and physical changes in the eyes. Ageing, ocular diseases, or injuries can alter

the iris structure. Furthermore, wearing glasses, contact lenses, or even eye makeup can interfere with the system's ability to capture a clear image of the iris.

Distance and Movement: The distance between the user and the scanner and any movement can significantly impact the system's ability to capture an accurate image. Users must be at an optimal distance, and any movement can blur the image, leading to recognition failures. This poses a challenge in environments where quick or on-the-move identification is needed.

Spoofing and Privacy Concerns: Like all biometric systems, iris recognition is not immune to spoofing attacks. High-quality fake eyes or images can potentially deceive scanners. Privacy concerns also arise, as the iris data, if compromised, can be misused, and unlike passwords, iris patterns cannot be changed.

Integration and Standardization: Integrating iris recognition technology with existing security systems can be challenging due to the lack of standardization across different devices and platforms. This can lead to additional costs and complexities in deploying iris recognition solutions on a large scale.

Acceptance and Comfort: Some users may find the process of scanning their iris invasive or uncomfortable, particularly in applications that require proximity to the scanning device. User acceptance is crucial for the widespread adoption of any biometric technology, and discomfort or privacy concerns can be significant barriers.

10. Conclusions and future remarks

Improve the database and the data storage within it since it currently functions as a simple local database. Additionally, a self-destruct condition should be implemented: to prevent someone with malicious intentions from attempting trial and error to guess the required number of blinks for data access, the record will be deleted from the database after a number of N incorrect attempts.

11. References

[1] https://github.com/aaahmedms/Iris-Recognition-Registration-Database-System

[2] https://iriun.com/

[3] https://pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

[4] https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/

The code was extracted from Visual Studio Code (https://code.visualstudio.com) using the CodeSnap extension (https://github.com/kufii/CodeSnap).