Iris recognition system in the spatial domain

Summary

Introduction

Scope of the system

The iris is the annular region of the eye bounded by the pupil and the sclera. Its intricate texture carries highly distinctive information useful for personal recognition. The accuracy and speed of currently implemented iris-based recognition systems are promising, supporting large-scale identification. Furthermore, recent systems have become more intuitive and user-friendly.

The unimodal iris-based recognition system has been designed to provide a reliable and efficient biometric identification method by analyzing and comparing the **keypoints** of the iris. This approach relies on identifying the unique characteristics of the iris, comparing common keypoints across different captures to determine an individual's identity with precision. The system aims to enhance security and efficiency in applications requiring fast and accurate identification.

The primary objective is to develop a robust solution capable of analyzing and comparing irises with high precision, even under variations caused by lighting, tilt, or partial obstructions of the eye. Additionally, the system must ensure high performance with limited computational resources, adapting to complex operational scenarios.

Context

The domain of iris-based comparison focuses on the analysis of keypoints, that are distinctive reference points that describe the unique structure of the iris. These keypoints are extracted using advanced image processing algorithms and represent a feature map essential for recognition. This approach is particularly valuable in contexts where other biometric modalities may be less reliable.

The keypoint comparison process enables:

- Identification of matches between unique iris features;
- Minimization of environmental variation impacts, such as light intensity or eye movement;
- Intrinsic robustness compared to other biometric techniques, like facial or fingerprint recognition.

Keypoint-based analysis is especially significant in environments requiring high levels of accuracy and speed, such as security and access control applications.

Importance of iris-based recognition

Iris-based recognition through keypoint comparison represents one of the most reliable and secure methods of personal identification. The complex and unique structure of the iris contains a set of details that can be accurately mapped and compared.

Using keypoints allows for:

- Reduced dependency on high-resolution images by focusing on essential information;
- Rapid and precise matching, even under suboptimal capture conditions;
- Leveraging precise segmentation techniques and matching algorithms to enhance system performance.

This approach has numerous practical applications, including:

- **Biometric verification**: quick and secure individual identification;
- **Access control**: applications in critical sectors like airport security or sensitive system access;
- **Identity management**: monitoring users in digital or physical environments.

Keypoint-based comparison stands out for its ability to minimize false positives and negatives, ensuring the highest degree of accuracy and reliability in identity verification.

System description

General architecture

The iris-based recognition system is divided into functional modules that work in synergy to ensure the accurate detection, processing, and comparison of iris features. These modules include:

1. **Dataset loading**: responsible for importing iris images from a pre-existing dataset, specifically the “CASIA (version 1.0)” dataset;
2. **Preprocessing**: includes techniques to enhance image quality;
3. **Keypoint extraction**: applies advanced algorithms to identify and map the distinctive keypoints of the iris;
4. **Matching**: compares the keypoints extracted from one image with those obtained from another image, using similarity metrics to determine correspondence;
5. **Decision**: analyzes matching results and generates a final response, determining whether the identity matches or not.

File structure

The system, implemented in Python, is organized into four files:

1. “config.py”;
2. “main.py”;
3. “segmentation.py”;
4. “utils.py”.

File “config.py”

The “**config.py**” file contains a series of essential configurations for the operation of the iris recognition system. These parameters govern various aspects of the detection, processing, and image comparison process. The file structure is organized into key-value dictionaries for easy customization.

Main configurations

The “**main**” dictionary defines the primary parameters related to data and the dataset:

- **DATA**: the name of the data file generated during processing;
- **DATASET**: the path to the dataset used for recognition. In this case, the “CASIA (version 1.0)” dataset is specified;
- **DEV_ANGLE**: tolerance for angular variations in matching keypoints;
- **DEV_DISTANCE**: tolerance for variation in normalized distances between keypoints relative to feature dimensions;
- **DEV_RATIO**: threshold for the ratio of matching distances;

- **EXTENSION**: File extension for images, set to ".bmp";
- **SHOW_ACCEPTANCES_CURVE**: boolean that determines whether to display the curve with the trend of the false acceptances number as the threshold varies;
- **SHOW_REJECTIONS_CURVE**: boolean that determines whether to display the curve with the trend of the false rejections number as the threshold varies;
- **SHOW_ROC_CURVE**: boolean that determines whether to display the Receiver Operating Characteristic (ROC) curve for performance evaluation.

Segmentation configurations

The "**segmentation**" dictionary defines the parameters used for iris and pupil localization, as well as optional visual settings:

- **CIRCLES_LENGTH**: specifies the maximum number of circles that can be detected during analysis;
- **IRIS**: parameters specific to iris segmentation:
  - **MEDIAN_1** and **MEDIAN_2**: lists of kernel sizes used for applying median filters to enhance image quality;
  - **PARAM_2_LIMIT**: upper limit for threshold parameter used in iris analysis;
  - **THRESHOLD**: a list of threshold values for localization.
- **MULTIPLIER**: a multiplier affecting the scale of measurements during localization;
- **PARAMS**: general parameters used during processing:
  - **PARAM_1** and **PARAM_2**: constants for adjusting algorithms, such as edge detection in Hough transforms.
- **PUPIL**: parameters specific to pupil segmentation:
  - **LENGTH**: maximum length for localization;
  - **MEDIAN**: median filter for enhancing pupil image quality;
  - **PARAM_2_LIMIT**: threshold parameter limit for pupil detection;
  - **THRESHOLD**: a list of threshold values for pupil identification.
- **RATIO**: geometric ratio used to compare relative dimensions of the iris and pupil;
- **SHOW**: visual configurations for debugging:
  - **CONTOURS**: show or hide detected contours;
  - **EQUALIZATION**: enable or disable histogram equalization;
  - **KEYPOINTS**: display or hide detected keypoints.
- **USE_HOUGH_LINES_METHOD**: boolean determining whether to use the standard or probabilistic Hough Line Transform method for localization.

Utility configurazions

The "**utils**" dictionary includes auxiliary parameters for managing and visualizing data:

- **DATA**: used for loading and saving files with data generated during processing;
- **KEYPOINTS**: file containing detected keypoints;
- **SHOW_MATCHES**: boolean specifying whether to display keypoint matches during comparison.

Conditional logic for detection method (USE_HOUGH_LINES_METHOD): conditional logic determines the output files based on the chosen detection method, either standard ("HoughLines") or probabilistic ("HoughLinesP"):

- If "USE_HOUGH_LINES_METHOD" is "True", specific files for the "HoughLines" method are used;
- If "False", file for the "HoughLinesP" method are applied.

This centralized configuration allows the system to be easily adapted to different operational scenarios without directly modifying the core code.

File "main.py"

The "**main.py**" file implements a system for analyzing and comparing iris images. It uses a series of functions defined in external modules and configuration variables. The goal is to analyze pairs of images to verify whether they belong to the same person, calculate comparison metrics, and display detailed results.

Function "compare_images"

This function compares two iris images given their paths. The main steps include:

- **Data loading**: if pre-calculated ".pkl" and ".txt" files exist for the images, it loads the saved information. Otherwise, it analyzes the images and saves the results;
- **Area comparison**: determines the number of matches between the areas of the two images;
- **Result evaluation**: returns the number of matches for subsequent analysis.

Function "find_optimal_threshold_and_error_metrics"

This function aims to calculate the optimal threshold for a binary classifier and evaluate False Reject Rate (FRR) and False Accept Rate (FAR) using the provided data. The main steps include:

- **Optimal threshold calculation**: the threshold is determined by iterating over a range of values from 1 to the maximum calculated score. The optimal threshold minimizes the total sum of false rejections number and false acceptances number;
- **Calculation and visualization of FRR and FAR**: for each threshold, the FRR and FAR values are identified and displayed. Additionally, FRR and FAR values for all explored thresholds are recorded, allowing an analysis of how they change with the threshold;
- **Result visualization**: returns the explored threshold values, the number of false rejections, and false acceptances for each threshold. It also enables the generation and observation of the ROC curve, if enabled.

Function "load_and_compare_dataset_images"

This function loads a dataset of images and compares pairs of images belonging to the same subject or different subjects. The main steps include:

- **Folder and image traversal**: iterates through subject folders and analyzes the images;
- **Smart comparison**: avoids redundant comparisons by saving already analyzed pairs;
- **Match count calculation**: for each pair of images compared, it finds the number of shared keypoints;
- **Verification system error calculation**: computes the number of false rejections in comparisons between images belonging to the same subject and the number of false acceptances in comparisons between images belonging to different subjects.

File "segmentation.py"

The "**segmentation.py**" file implements advanced functions for locating and analyzing circular areas in grayscale images, such as the pupil and iris in an eye image. It leverages image processing techniques provided by the OpenCV library, including Hough transforms, contour detection, and Gaussian and median filters.

Function "find_areas"

This function identifies and isolates the pupil and iris in an image. The main steps include:

- **Image loading**: the image is loaded in grayscale;
- **Pupil detection**: using median filters, threshold levels, and the Hough transform, the function aims to identify circles that meet the criteria for a pupil;
- **Iris detection**: once the pupil is detected, the center and radius of the iris are estimated, again using adaptive Hough transforms;
- **Optional visualization**: if configured, the program displays the contours of the pupil and iris overlaid on the image;
- **Iris area equalization**: enhances the contrast of the iris area to improve the detection of fine details;
- **Keypoint detection**: using the SIFT method, distinctive points within the iris area are identified, excluding those inside the pupil and outside the iris, as well as removing eyelashes;
- **Filtering**: keypoints are filtered to ensure they are valid and relevant.

Function "find_iris"

This function focuses exclusively on detecting the iris surrounding a previously identified pupil. The main steps include:

- **Image preparation**: applies median filters and edge detection to highlight relevant contours;
- **Hough transform usage**: detects potential circles that meet the criteria for representing an iris;
- **Circle filtering**: implements statistical criteria to remove anomalous circles based on standard deviations and mean distance;
- **Result averaging**: returns a single average circle representing the iris.

File "utils.py"

The "**utils.py**" file contains a collection of auxiliary functions used in the system for analyzing and comparing iris images. These functions support data management, match calculations, result visualization, and other essential operations.

Function "get_circle_means"

This function calculates the average values of coordinates (x, y) and the radii of circles. It takes a list of circles as input, where each circle is represented by a tuple (x, y, radius), and returns a tuple containing the averages (x_mean, y_mean, radius_mean).

Function "get_matches"

This function determines the number of matches between keypoints from two iris images. The main steps include:

- **Input data check**: if there are no keypoints in a set, it returns None;
- **Keypoint matching**: uses a brute force matcher, called BFMatcher, with the k-NN method to find matches between keypoints;
- **Initial filtering**: filters matches based on a distance ratio;
- **Angle and distance calculations**: calculates normalized angles and distances between keypoints and pupil centers;
- **Advanced filtering**: applies additional filters based on median values of angles and distances to exclude invalid matches;
- **Output**: draws valid matches on the image and displays them in a window if configured.

Function "get_mean_and_std"

This function calculates the mean and standard deviation of a set of values. It takes a list of numbers as input and returns the mean and standard deviation.

Function "load_from_pkl"

This function loads image data from a ".pkl" file and, if available, reads the associated keypoints from a separate ".txt" file. The main steps include:

- **Data loading**: loads relevant image data from the pickle file;
- **Keypoint loading**: loads the associated keypoints from a text file, if it exists.

Function “**plot_roc**”

This function plots a ROC curve to visualize system performance. The main steps include:

- **TPR and FPR calculation**: calculates the True Positive Rate (TPR) and False Positive Rate (FPR) required to represent the ROC curve;
- **AUC calculation**: computes the Area Under the Curve (AUC);
- **ROC curve composition**: plots the ROC curve based on TPR and FPR values.

Funzione “**save_to_pkl**”

This function saves image data in a “.pkl” file and keypoints in a separate “.txt” file because the “KeyPoint” object generated by the “OpenCV” library is incompatible with the pickle format. The main steps include:

- **Data filtering**: filters the dictionary containing the data to remove keypoints;
- **Saving filtered data to** “**.pkl**”: saves the filtered dictionary in a pickle file;
- **Saving keypoints to** “**.txt**”: saves keypoints as formatted strings in a text file.

Funzione “**show_curve**”

This function generates a graph illustrating the trend of false rejections or false acceptances number as the threshold used in the biometric authentication system changes. The main steps include:

- **Graph preparation**: draws a graph with the threshold on the x-axis and the number of errors, i.e., false rejections or false acceptances, on the y-axis;
- **Result visualization**: displays the graph to analyze the trend of errors concerning the threshold.

Funzione “**show_FRR_and_FAR_results**”

This function calculates and displays the FRR and FAR rates. The main steps include:

- **Actual FRR calculation**: computes FRR as the ratio of the number of false rejections to the total number of intra-class comparisons;
- **Actual FAR calculation**: computes FAR as the ratio of the number of false acceptances to the total number of inter-class comparisons;
- **System Results visualization**: displays the results in a readable format with actual and percentage values.

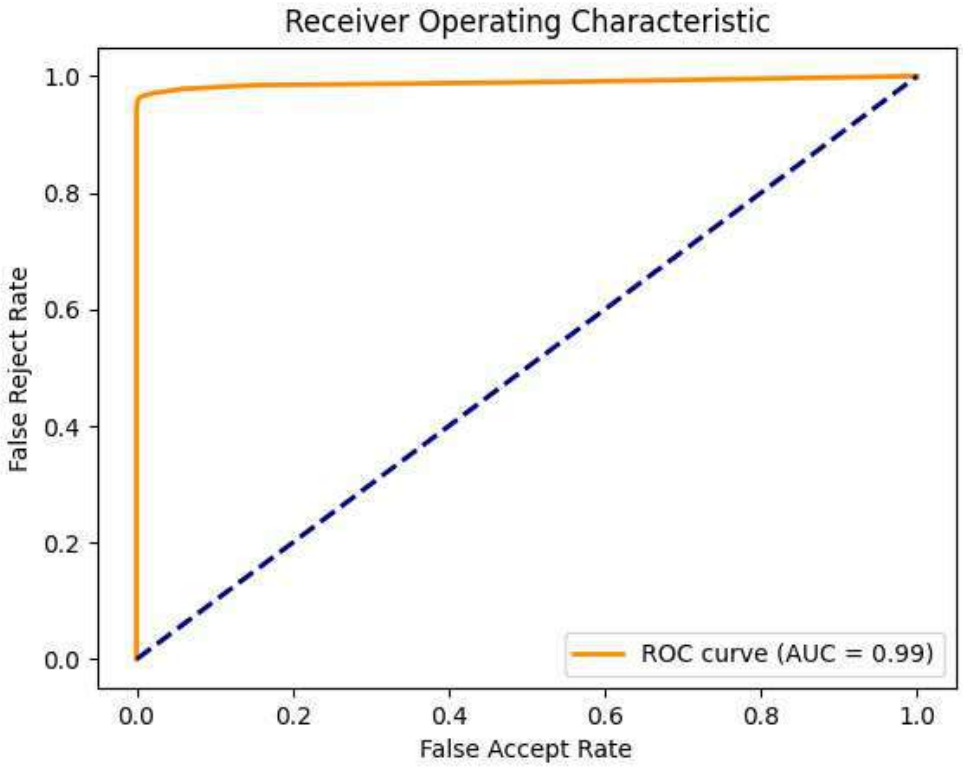
Results and performance

To evaluate the effectiveness of the iris recognition system, intra-class and inter-class comparisons were performed, with the computation of FRR and FAR metrics. These comparisons were conducted by varying only the "**USE_HOUGH_LINES_METHOD**" parameter in the configuration file to alternate between the "HoughLines" and "HoughLinesP" methods.

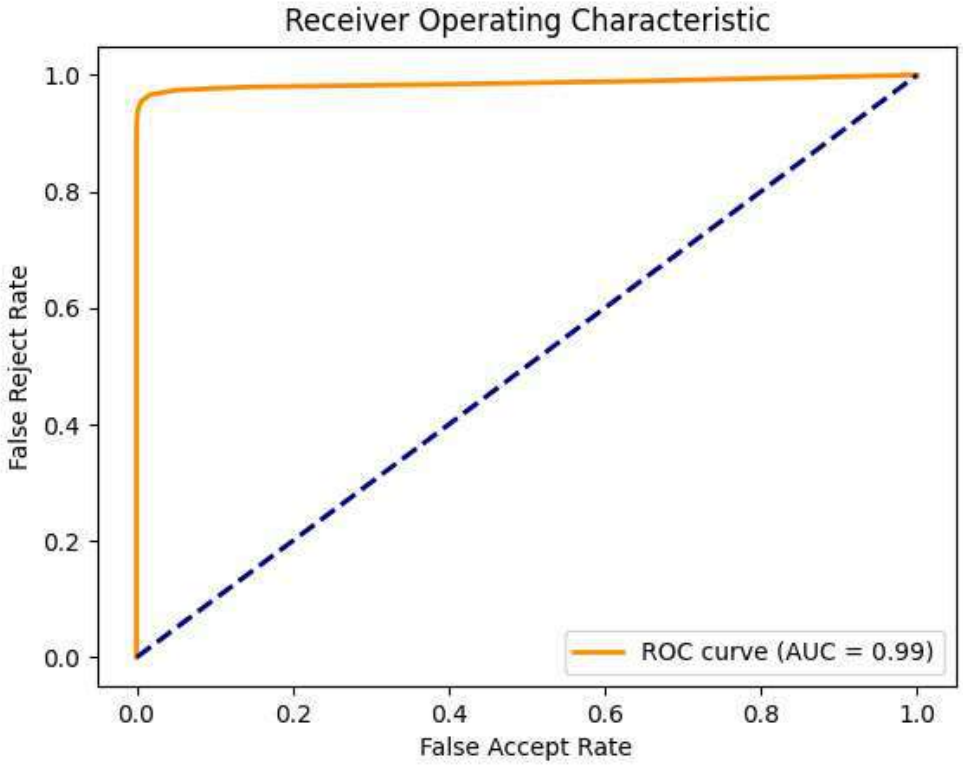
1. **Intra-class comparisons**: intra-class comparisons assessed the system's ability to recognize images belonging to the same class, such as images of the same iris. A low FRR in this context indicates that the system effectively avoids mistakenly rejecting images correctly associated with the same class;
2. **Inter-class comparisons**: inter-class comparisons analyzed the system's ability to distinguish between images from different classes, such as irises of different individuals. A low FAR in this case reflects the system's capacity to avoid false matches between images of different classes;
3. **Line detection techniques**:
   - **HoughLines**: this traditional technique detects global lines within the image based on the analysis of the parametric space. It was used to identify and remove linear structures that obstruct the description of iris features;
   - **HoughLinesP**: an optimized and probabilistic version of the Hough method that reduces the number of calculations and allows the detection of segmented lines instead of global lines. This approach proved useful in identifying and eliminating partial or fragmented obstructive structures in the image.
4. **Calculation of FRR and FAR metrics**:
   - **False Reject Rate (FRR)** is calculated as the ratio of the number of erroneously rejected intra-class comparisons to the total number of intra-class comparisons performed;
   - **False Accept Rate (FAR)** is determined as the ratio of the number of erroneously accepted inter-class comparisons to the total number of inter-class comparisons conducted.
5. **Comparison between HoughLines and HoughLinesP**: using both techniques allowed for a comparison of how the system's accuracy varies. Below are the results obtained using each technique:

Applied technique	Optimal threshold	False rejections number	Total intra-class comparisons	False acceptances number	Total inter-class comparisons	FRR (%)	FAR (%)
HoughLines	10	134	2268	28	283122	5,91%	0,01%
HoughLinesP	10	225	2268	28	283122	9,92%	0,01%

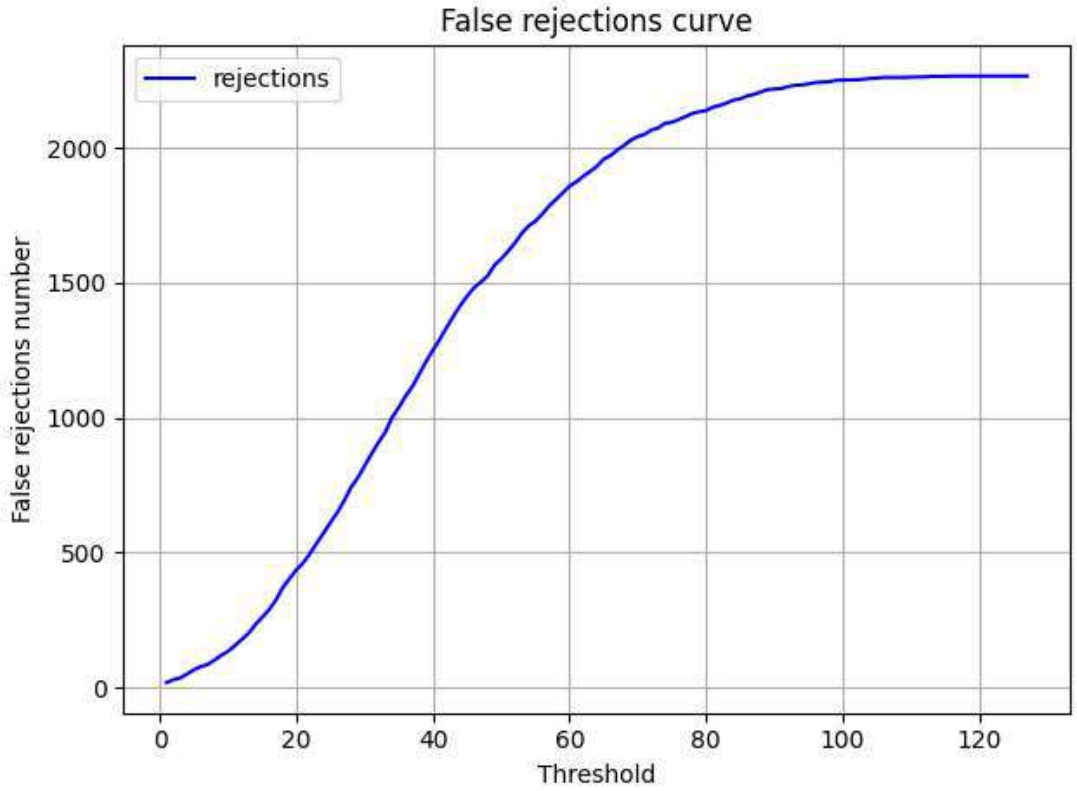
ROC curve and corresponding AUC with “HoughLines” method:



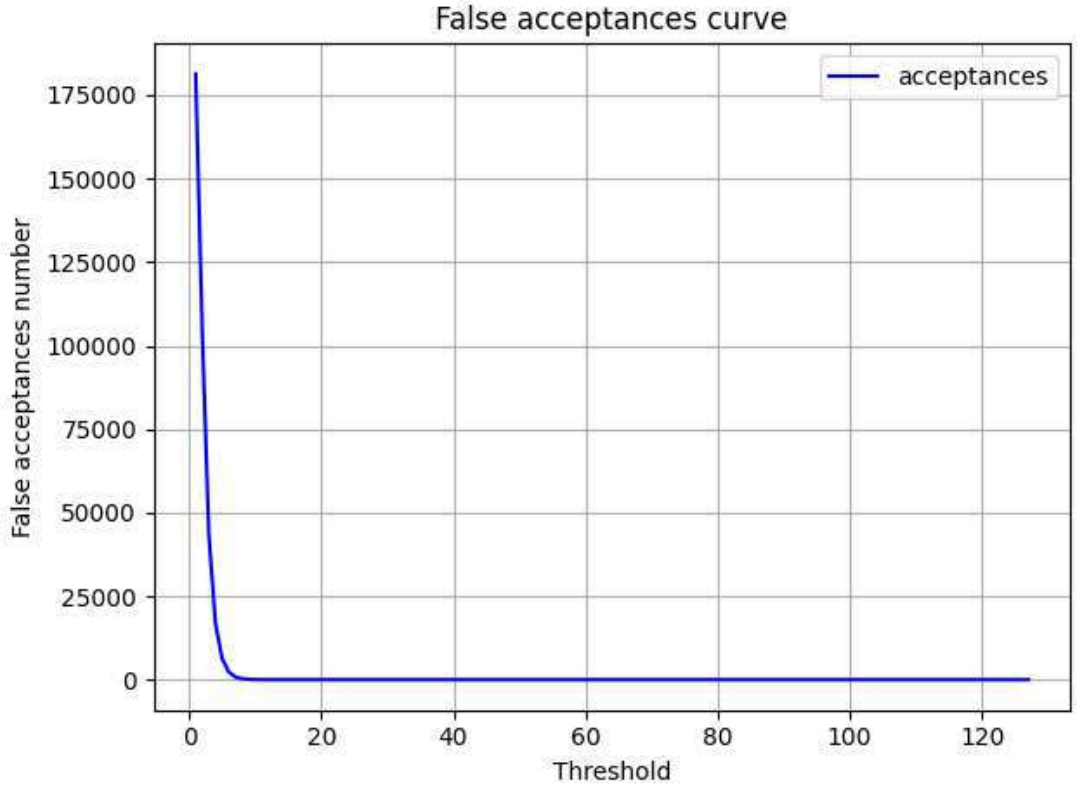
ROC curve and corresponding AUC with “HoughLinesP” method:



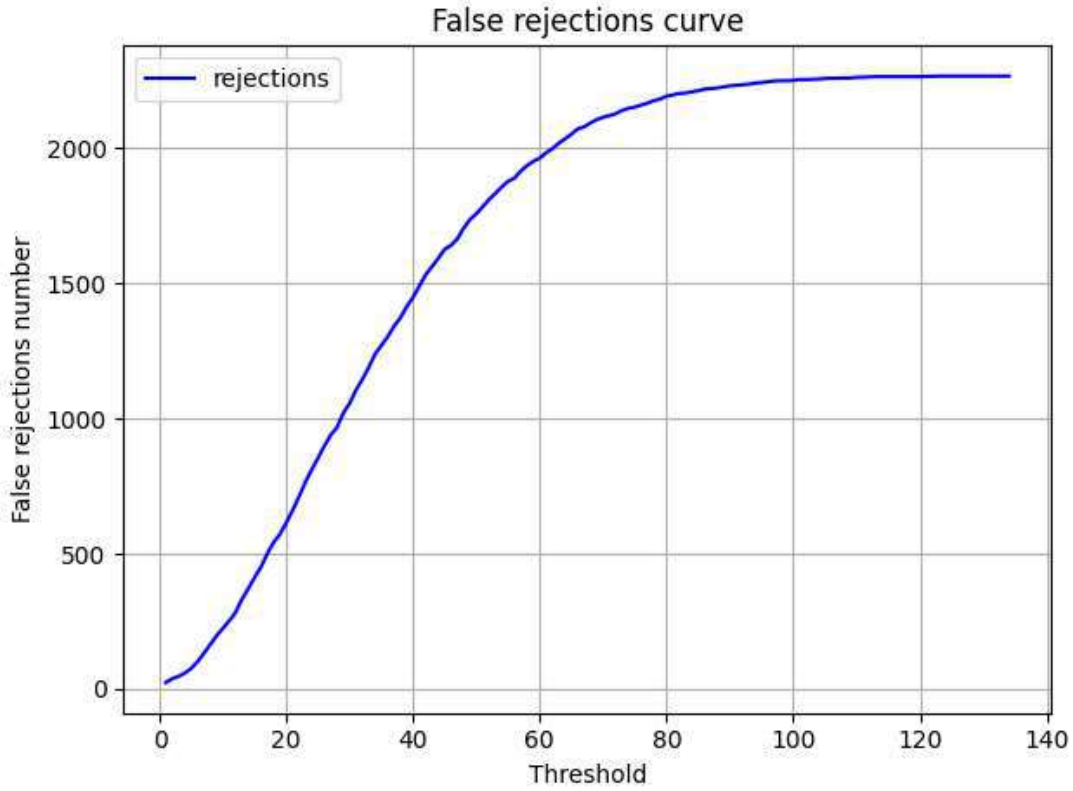
False rejections curve as the threshold varies with the “HoughLines” method:



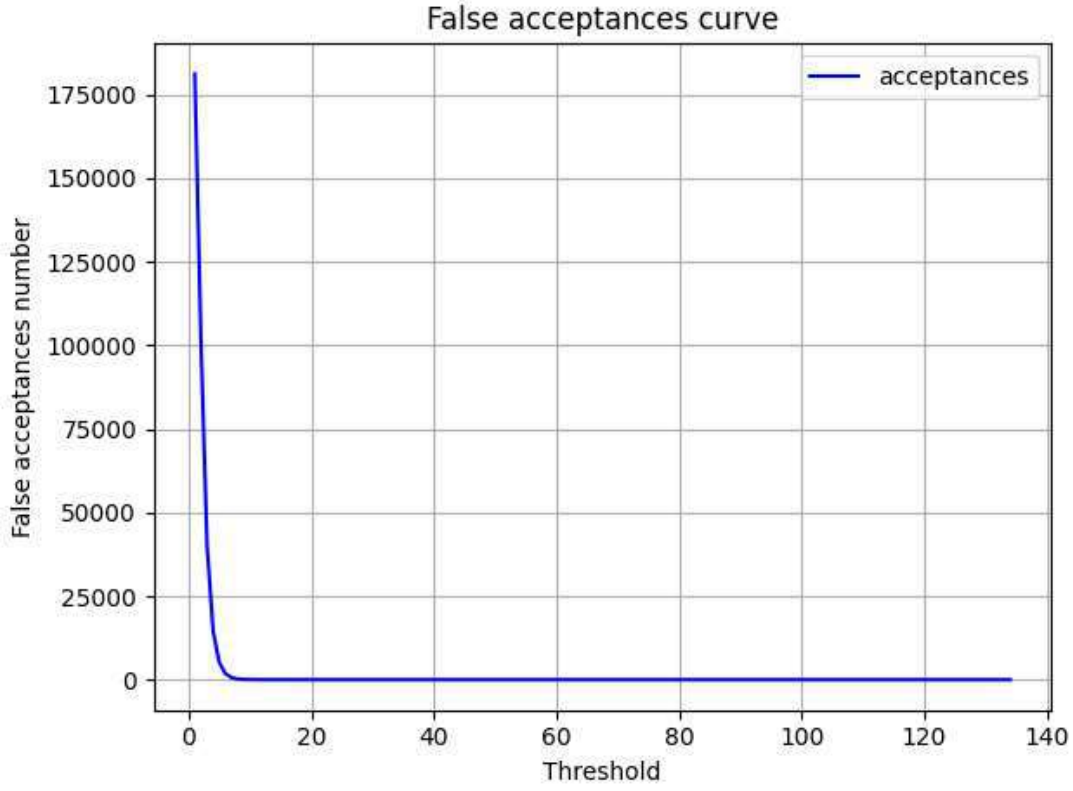
False acceptances curve as the threshold varies with the “HoughLines” method:



False rejections curve as the threshold varies with the “HoughLinesP” method:



False acceptances curve as the threshold varies with the “HoughLinesP” method:



Conclusions

The developed iris recognition system has proven to be effective and promising, achieving high accuracy in identification and verification stages, as confirmed by tests conducted on the CASIA dataset. The results highlight the robustness of the algorithm in recognizing the unique features of the iris while maintaining reliable performance even under slight variations in acquisition conditions, such as moderate lighting changes.

However, certain limitations have emerged, including higher sensitivity to the quality of input images and the presence of occlusions, such as reflections, eyelids, or eyelashes. These aspects represent opportunities for improvement in future developments.

The potential practical applications of this system are numerous, ranging from airport security and access control in critical settings to protecting sensitive data in digital environments. Furthermore, adopting multimodal approaches by integrating iris recognition with other biometric technologies, such as facial or fingerprint recognition, could further enhance the system's effectiveness and versatility.

In conclusion, this work represents a significant step forward in the field of biometrics, demonstrating how iris-based authentication can provide secure, precise, and scalable solutions. With further optimizations, the system could find concrete applications in a wide range of scenarios, contributing to improved security and efficiency in authentication processes.