Instituto Politécnico de Lisboa
Instituto Superior de Engenharia de Lisboa

Biometric Systems
Project 2: Face Recognition Biometric System

January 2025

Authors:
Anna Ghiotto 53021
Adam Szokalski 53153
Adrian Osędowski 53015

Contents

1 Introduction

The goal of this project is to develop a comprehensive face recognition system capable of processing raw images, extracting features, and using machine learning models to achieve accurate recognition and authentication of individuals. Despite variations in appearance, such as lighting, facial expressions, and accessories, the system employs eigenfaces, a technique derived from Principal Component Analysis (PCA), to reduce the dimensionality of facial images while preserving critical features for recognition.

The project includes three guides embedded within the code as comments to facilitate understanding and implementation.

2 Dataset

The ATT Database of Faces was utilized for this project. This dataset comprises 40 individuals, each with ten images. The images exhibit variations in lighting, facial expressions (e.g., open or closed eyes, smiling or not smiling), and facial details (e.g., wearing glasses or not). All images are captured against a uniform dark background, with subjects positioned upright and facing forward, allowing slight side movement.

3 Code structure

This system is an adaptation of the ECG Biometric System developed in a previous project. The structure remains similar but has been adjusted for image processing specific to face recognition. The primary goal of this structure is to create a pipeline that processes raw images, extracts essential features, classifies individuals based on these features, and evaluates the identification and authentication systems' performance. The Preprocessing class is left for prospective improvements, but is not used in this project.

The code is divided into several key components.

3.1 Data Import

The `GetFaces` class is responsible for loading structured images from the chosen dataset and applying preprocessing and feature extraction steps to maintain consistency. Built on top of `TqdmIteratorBase`, this class iterates through each individual in the dataset, retrieves their images, and automatically creates a Person object for each individual. This structure simplifies data management and ensures compatibility with other system components. Additionally, it allows new datasets to be added with minimal effort.

3.2 Feature Extraction

The system includes a single extractor that suffices for building an accurate recognition system. The `EigenFacesExtractor` class encodes each face by projecting it onto the eigenfaces space, leveraging the `FaceEncoder` class. Pre-computed eigenfaces are loaded and utilized by this class to encode the images efficiently.

Eigenfaces play a central role in the feature extraction process of this face recognition system. They are derived using Principal Component Analysis (PCA), which reduces the dimensionality of facial images while preserving the most significant features necessary for recognition. The `EigenFacesExtractor` class extends the abstract `FeatureExtractor` base class and implements the process of encoding faces into their corresponding eigenfaces representation. Each image is passed through the `extract` method, which leverages an `encoder` to project the image onto the eigenfaces space. This encoding process ensures that only the essential components of the facial structure are retained, enabling efficient and accurate classification and authentication. By representing images in this reduced feature space, eigenfaces mitigate issues related to data redundancy and computational inefficiency, while maintaining the system's recognition capability.

3.3 Classification

The `Classifier` class serves as an abstract base class that defines the core functionality for all classifiers. It includes methods for:

- `fit`: Training the model using provided training data.
- `identify`: Predicting an individual's identity based on their data.
- `authenticate`: Verifying an individual's identity by checking if their data meets a predefined confidence threshold.

The `XGBoostClassifier` class extends the `Classifier` and implements these functionalities using the XGBoost algorithm. It employs a multi-class objective to classify individuals and monitors performance on an evaluation dataset during training.

The `identify` method predicts a person's identity by analyzing the probability distribution of templates across all classes and selecting the most frequent prediction meeting the confidence threshold.

The `authenticate` method verifies a `Person`'s identity by randomly picking one of their templates and checking if the model's confidence in identifying them exceeds the defined threshold.

3.4 Evaluation

The system's performance was evaluated using two methods: a simple train-test split and k-fold cross-validation. These approaches assess model stability by averaging results across different data divisions.

The `train_test_split` method in the `Person` class divides each individual's images into training and testing sets. Similarly, the `k_fold_split` method in the `Person` class and the related function in `utils.py` divide the signals into multiple parts (folds).

The main performance metrics used for evaluation include:

- **Accuracy**: The proportion of correct predictions made by the system.
- **Equal Error Rate (EER)**: The rate at which false acceptances equal false rejections.
- **EER Threshold**: The threshold value at which the EER occurs.
- **Area Under the Curve (AUC)**: A measure of how well the model distinguishes between genuine profiles and impostors.

The `evaluate` function in the `Classifier` class is responsible for calculating these metrics, offering a detailed view of the system's performance. Additionally, the system provides basic counts of successful authentications and correct identifications.

4 Results

- **Accuracy:** The model correctly predicts the identity 80% of the time, which indicates good classification performance, but there is a possibility for improvements.

Table 1: Performance Metrics for Train-Test Split and K-Fold Validation

Metric	**Train-Test Split**	**K-Fold Validation**
Accuracy (%)	80.00	81.11
Equal Error Rate	0.400	0.417
Area Under Curve	0.867	0.603
Correct Identifications (%)	75.00	91.67
Successful Authentications (%)	60.00	58.33

- **Equal Error Rate (EER):** The EER is relatively high, suggesting that the system has difficulty balancing false acceptances and false rejections.
- **Area Under Curve (AUC):** The model has good discriminatory power, as the AUC is close to 1, but it is much lower in k-fold validation, what suggests inconsistent performance across the folds, potentially due to data variability. However, the variance between AUC and EER suggests that the system's performance might vary under different thresholds. It could be caused by too small dataset used.
- **Correct Identifications:** There is a high number of correct identifications, but in k-fold validation is even higher, what could indicate that the model benefits from more training data in each fold. It corresponds to the accuracy calculated before.
- **Successful Authentications:** The authentication success rate remains similar in the train-test split and k-fold validation. Difference between AUC and EER suggesting that the threshold value might need adjustment to improve this metric.

5 Conclusion

In summary, the facial recognition system developed for this project demonstrates good performance, achieving an accuracy of approximately 80%. Despite this, areas for improvement exist, particularly in balancing false acceptances and rejections, as evidenced by the EER. The results suggest that increasing the dataset size and adjusting thresholds could enhance accuracy and authentication rates. Future work should focus on optimizing thresholds and exploring alternative methods to improve system robustness.