Computer Vision

“Task 5”
Face Detection & Recognition

Introduction

This report will explain a Variety of algorithms implemented in C++ using Qt6 Creator. The algorithms aim to detect faces from a given image and recognize the team members' faces “4 members”. The implemented algorithm is PCA, SVM machine learning model, and cascade built-in library for face detection.

Team Members:

Magdy Nasr - Ahmed Emad - Youssef Kadry - Mohab Ghobashy - Mohammed Mostafa

Team No. 3

Tab 1 - Face Detection

Page Content

- On this page, the user uploads an image and the output image contains the faces surrounded by green squares.

The used library:

Here's a high-level overview of how cascade classifiers work for face detection:

1. Haar-like features: Haar-like features are simple rectangular patterns that are used to represent different visual patterns within an image. These features capture information about the contrast between adjacent regions of the image.

2. Training: Cascade classifiers are trained using a large set of positive and negative samples. Positive samples are images that contain the objects of interest (e.g., faces), while negative samples are images that do not contain the objects. During the training process, the cascade classifier distinguishes between samples by selecting the most informative Haar-like features.

3. AdaBoost algorithm: AdaBoost (Adaptive Boosting) is a machine learning algorithm that is used to construct a strong classifier by combining multiple weak classifiers. In the case of cascade classifiers, each stage of the cascade consists of a weak classifier, which is typically a decision tree based on a specific set of Haar-like features. The AdaBoost algorithm assigns weights to the weak classifiers, giving more importance to those that perform well on the training data.

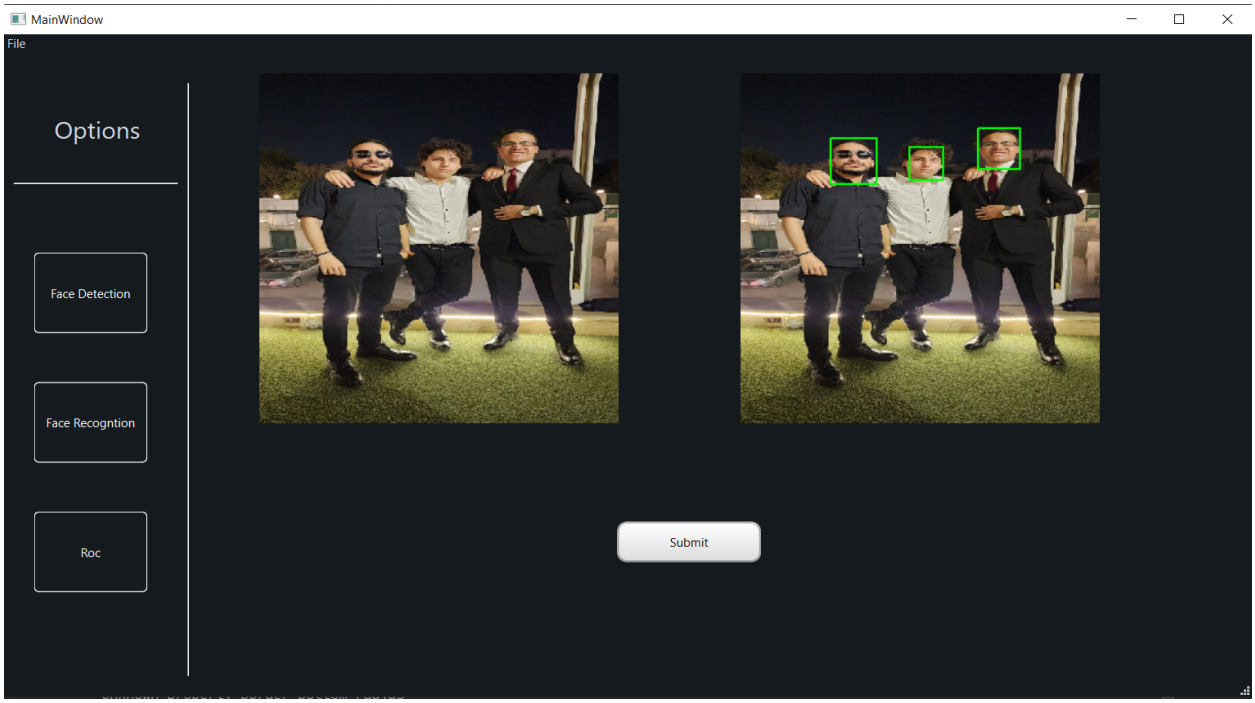
4. Cascade structure: A cascade classifier consists of multiple stages, where each stage consists of several weak classifiers. The stages are organized in a cascade structure, where each subsequent stage has a higher threshold for acceptance. This cascade structure allows for fast rejection of regions that are unlikely to contain the object of interest, reducing the computational load.

5. Detection process: During the detection process, the cascade classifier is applied to an input image. The image is divided into sub-windows of various sizes and each sub-window is evaluated by the cascade classifier. At each stage of the cascade, the sub-window is passed through the weak classifiers. If the sub-window passes all the stages, it is considered a positive detection (e.g., a face).

6. Sliding window and image pyramid: Since objects can appear at different scales in an image, the detection process often involves scanning the image at multiple scales. This is achieved by using a sliding window approach, where a fixed-size window is moved across the image at different positions and scales. Additionally, an image pyramid may be used to create multiple scaled versions of the original image, allowing detection at different resolutions.

7. Non-maximum suppression: After the detection process, multiple overlapping detections may occur. To eliminate redundant detections, a non-maximum suppression step is typically applied.

Example:



Tab 2 - Face Recognition

Page Content

- On this page, the user uploads an image and the output image contains the faces surrounded by green squares and the person's name.

PCA Algorithm for Feature Extraction:

Here's an overview of how PCA works for feature extraction in computer vision:

1. Data Preparation: The input dataset consists of a collection of high-dimensional data points or vectors, typically represented as an N x M matrix, where N is the number of samples and M is the number of features or dimensions.

2. Mean Centering: PCA begins by subtracting the mean of each feature from the data points. This process centers the data around the origin, ensuring that the mean of each feature becomes zero.

3. Covariance Matrix Calculation: The covariance matrix is computed from the mean-centered data. The covariance matrix provides information about the relationships and variances between different features.

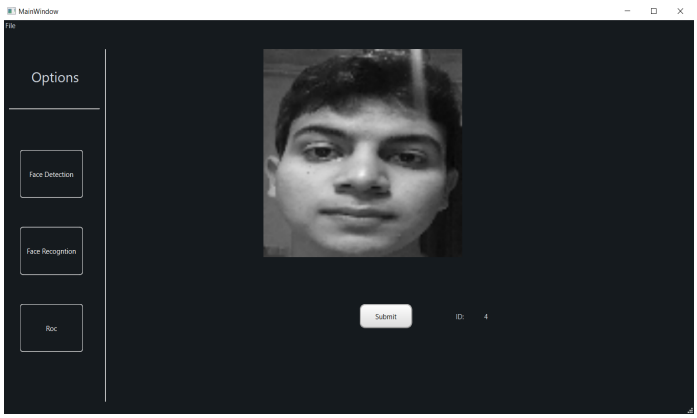
4. Eigenvalue Decomposition: The covariance matrix is then decomposed into its eigenvectors and eigenvalues. The eigenvectors represent the principal components of the data, and the eigenvalues indicate the amount of variance captured by each principal component.

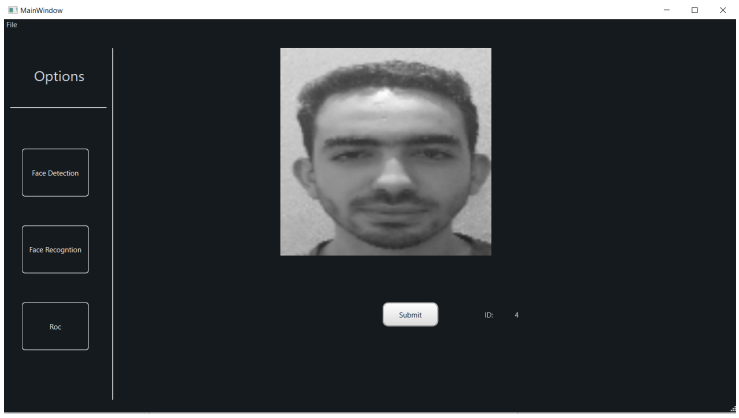
5. Eigenvalue Sorting: The eigenvectors are typically sorted in descending order based on their corresponding eigenvalues. This ensures that the principal components are arranged from the most significant to the least significant in terms of capturing the variance in the data.

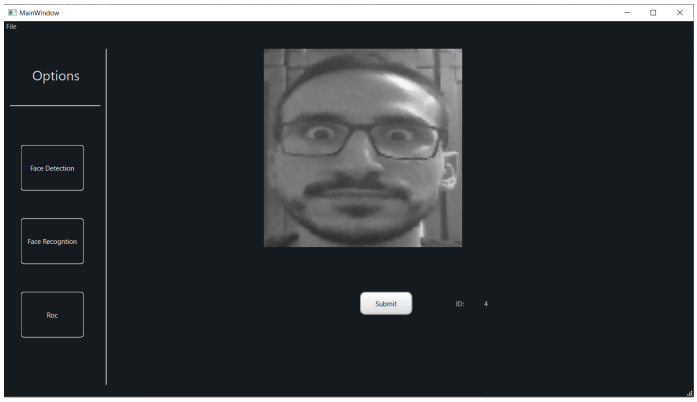
6. Dimensionality Reduction: A user-defined number of principal components (k) is selected to extract a subset of the most informative features. These principal components form a new lower-dimensional subspace that retains most of the variance in the original data.

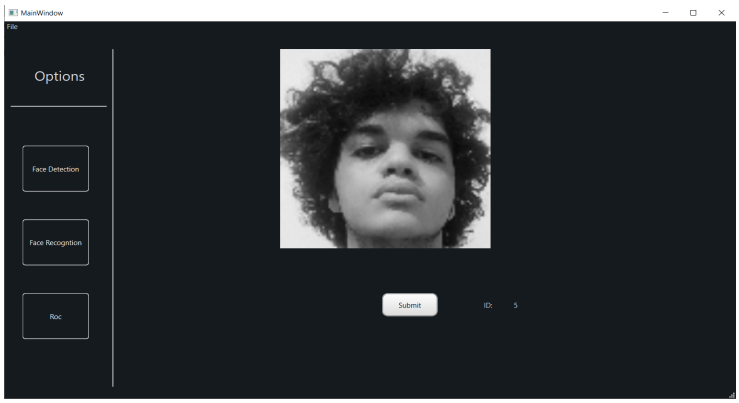
7. Feature Extraction: The original high-dimensional data is projected onto the lower-dimensional subspace spanned by the selected principal components. This projection results in a new representation of the data.

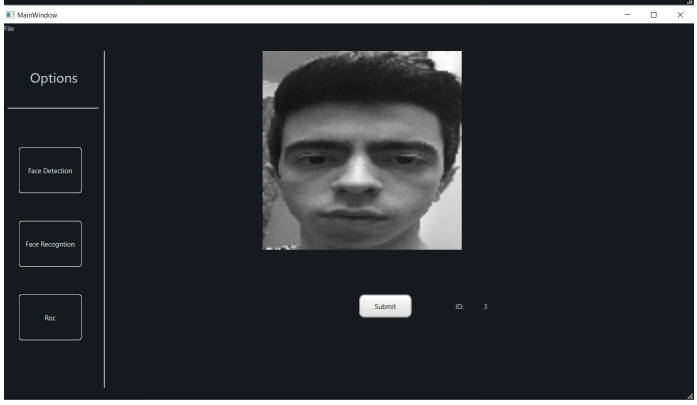
Example:

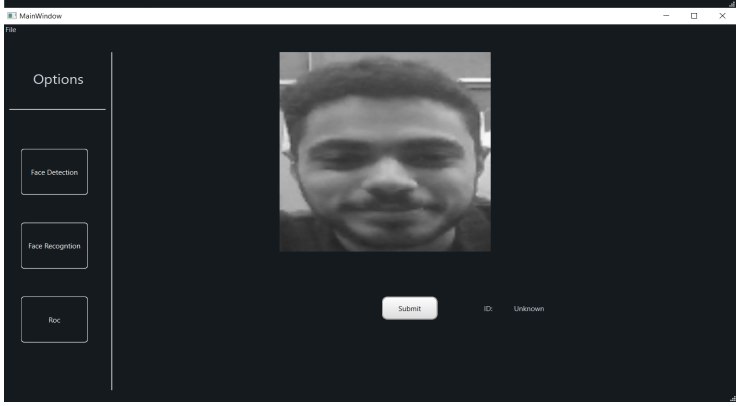












Tab 3 - ROC Curve

Page Content

- On this page, the user can see the ROC Curve for each class of the training data.

ROC Curve Implementation:

Here's an overview of how ROC Curve is implemented:

1. Firstly we calculate Euclidean distances between each test and training data to get the value of the minimum and maximum distance (The range of thresholds we're going to test).
2. For a specific class, we create a confusion matrix for each threshold we test, then retrieve the True Positives, False Positives, True Negatives, and False Negatives from it using a 1 vs all approach.
3. We calculate the True Positives Rate and the False Positives Rate and plot them on a graph.
4. We repeat the past 2 operations for each threshold we test.
5. We repeat the whole previous process again to plot the ROC Curve for each other class.

Example: