**Cairo University**

**Faculty of Engineering**

**Systems & Biomedical Engineering**

**Computer Vision**

**(SBE 3230)**

**Under the Supervision of:**

Professor: Ahmed M. Badawi

Eng. Laila Abbas

Eng. Omar Hesham

**Team Members:**

1. Lama Zakaria
2. Lamees Mohee
3. Camellia Marwan
4. Rana Ibrahim
5. Fady Mohsen Magdy

**Face Detection Using Haar Cascade Algorithm**

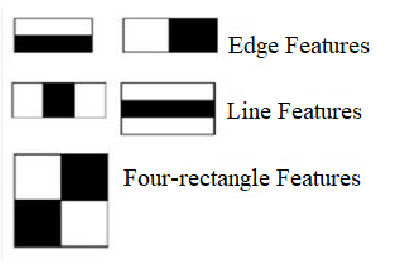
Among the various approaches to face detection, the **Haar Cascade algorithm** has gained significant attention due to its efficiency and accuracy. Developed by Viola and Jones in 2001, the Haar Cascade algorithm revolutionized face detection by introducing a robust method that can detect faces in real-time.

The key innovation of the Haar Cascade algorithm lies in its use of **Haar-like features** and a **cascaded classifier**. Haar-like features are simple rectangular patterns that capture local image intensity variations, while the cascaded classifier combines multiple weak classifiers to achieve high detection rates with low false positives. This combination makes the Haar Cascade algorithm particularly well-suited for real-time face detection in various environments.

**Algorithm Steps**

1. Haar-Like Features
2. Integral Image
3. AdaBoost Training
4. Cascade Classifier
5. Detection
6. **Haar-Like Features**

Haar-like features are simple rectangular patterns that capture local image intensity variations. These features act as basic filters that can detect edges, lines, and other patterns in an image.



The algorithm calculates the difference between the sum of pixel intensities in the white and black rectangles of each feature. By analyzing these features across different parts of the face, the algorithm can distinguish between facial features and the background.

1. **Integral Image**

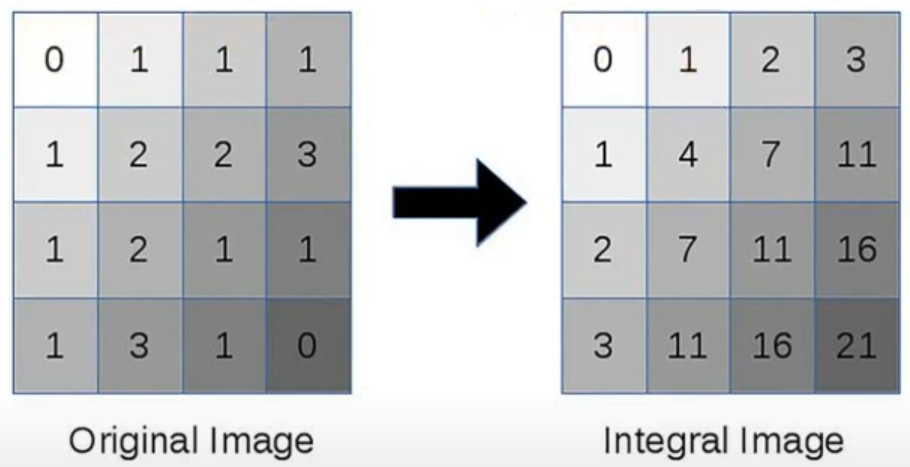
The integral image is a two-dimensional array that contains the sum of all pixels above and to the left of each pixel in the original image. This representation enables fast computation of Haar-like features over rectangular regions of the image.

A screenshot of a grid

Description automatically generated

When we add the pixels in the blue box, we get 8 as the sum of all pixels and here we had six elements involved in your calculation. Now to calculate the sum of these same pixels using the integral image, you just need to find the corners of the rectangle and then add the vertices which are green and subtract the vertices in the red boxes. Now doing that here

We get the same answer and only four numbers are involved in calculations. No matter how many pixels are in the rectangle box, we will just need to compute on these 4 vertices.



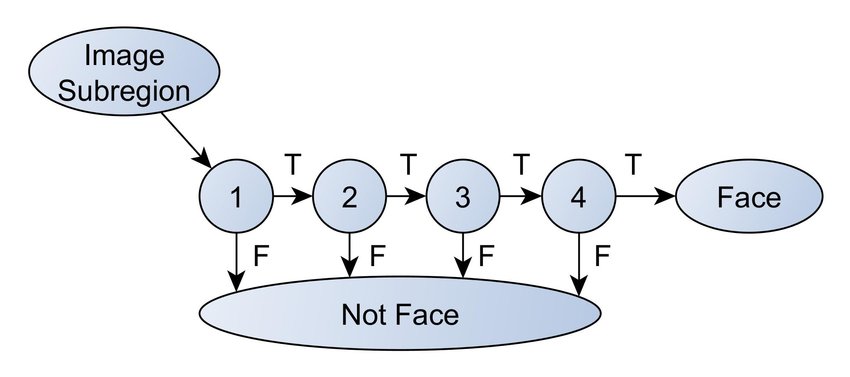
By using the integral image, the algorithm can calculate the sum of pixel intensities within any rectangular region in constant time, regardless of the size of the region. This significantly speeds up the feature calculation process.

1. **AdaBoost Training**

Adaboost is a machine learning algorithm used to create a strong classifier by combining multiple weak classifiers. In the context of Haar Cascade, Adaboost selects a small number of critical Haar-like features and their thresholds that best classify faces and non-faces. It iteratively trains a series of weak classifiers, with each classifier focusing on the mistakes made by the previous ones. This process selects a set of features that collectively provide high accuracy in face detection.

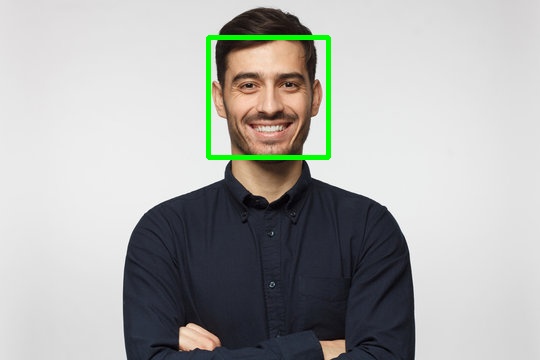
1. **Cascade Classifier**

The cascade classifier is a series of stages, each containing multiple weak classifiers. Its purpose is to efficiently reject non-face regions early in the detection process, reducing the computational load. Each stage consists of a set of weak classifiers, and an image region is rejected at a stage if it fails to pass any of the weak classifiers. This cascaded structure allows for fast rejection of non-face regions, making the algorithm suitable for real-time applications.



1. **Detection**

Detection involves sliding a window across the image and applying the cascade classifier to each window.

** **

**Face Recognition**

Principal Component Analysis (**PCA**) is a popular technique for dimensionality reduction and feature extraction in face recognition systems.

**PCA for Dimensionality Reduction:**

* PCA is a statistical technique used to reduce the dimensionality of data while preserving its essential features.
* It identifies orthogonal axes (principal components) that capture the maximum variance in the data.
* By projecting data onto a lower-dimensional subspace spanned by the principal components, PCA facilitates efficient representation and visualization of high-dimensional data.

**Methodology:**

1. **Data Preparation:**

* Training and testing images() are preprocessed through resizing images (50,50) and normalization (By subtracting mean of all trained images from image) to ensure uniformity in data representation.

1. **Principal Component Analysis (PCA):**

* PCA is applied to the preprocessed training images to extract the most significant features while reducing dimensionality.
* Eigenvalues and eigenvectors of the covariance matrix are computed to determine the principal components.
* The number of principal components is selected based on a threshold of explained variance is 39, typically set to retain a certain percentage of the total variance (e.g., 98%).

1. **Face Reconstruction:**

* Faces are reconstructed using the selected principal components to visualize the impact of dimensionality reduction and understand the features captured by PCA.

1. **Testing and Evaluation:**

* Testing images are projected onto the PCA space to obtain their representations.
* Distance metrics such as Euclidean distance and cosine similarity are employed to measure the similarity between the representations of testing and training images. (**Euclidean distance gets the best Performance**)
* **K Nearest neighbor (K=1)** classification is used to predict the identity of the test images based on the closest match in the training set.

**Results**:

The training set consists **of 9 people.**

** **

**Reconstructed image by 39 eigen faces**

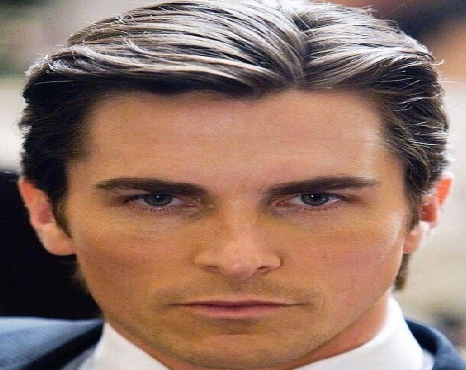
**Test image**

**.**

**Predicted person:**

**A black and white text

Description automatically generated**

** A close up of a person's face

Description automatically generated**

**Reconstructed image by 39 eigen faces**

**Test image**

**Predicted Person:**

**A white text on a black background

Description automatically generated**

**A person with blue eyes and red lipstick

Description automatically generated A close-up of a person's face

Description automatically generated**

**Reconstructed image by 39 eigen faces**

**Test image**

**Predicted person:**

**A black background with white text

Description automatically generated**

** A close up of a person's face

Description automatically generated**

**Reconstructed image by 39 eigen faces**

**Test image**

**Predicated person:**

**A white text on a black background

Description automatically generated**

**A close up of a person's face

Description automatically generated **

**Reconstructed image by all eigen faces**

**Test image**

**Roc curves for face recognition :**

ROC curves, serve as a vital tool for evaluating binary classification systems by illustrating the trade-off between true positive rate (TPR) and false positive rate (FPR) at different classification thresholds. TPR, or sensitivity, measures the proportion of positive instances correctly identified by the model, while FPR indicates the proportion of negative instances incorrectly classified as positive.

By plotting TPR against FPR, ROC curves provide a visual representation of a classifier's discrimination ability, with a curve above the diagonal representing better-than-random performance. The area under the ROC curve (AUC-ROC) summarizes the classifier's performance, with higher values indicating superior discrimination. In face recognition, ROC curves offer insights into the system's ability to distinguish between faces and non-faces, aiding in algorithm evaluation, threshold optimization, and model comparison.

**Use of ROC Curves:**

1. **Evaluating Binary Classification Systems:**
   * ROC curves are useful for evaluating binary classification systems, such as face recognition, where the output is either positive or negative.
2. **Handling Class Imbalance:**
   * ROC curves are robust to class imbalance, making them suitable for evaluating classifiers trained on imbalanced datasets.
3. **Threshold Selection:**
   * ROC curves visualize the trade-off between TPR and FPR at different classification thresholds, aiding in selecting an optimal threshold based on application requirements.
4. **Comparing Models:**
   * ROC curves facilitate the comparison of multiple models by comparing their AUC-ROC scores. A higher AUC-ROC score indicates better discrimination between positive and negative instances.

**Importance of TPR and FPR:**

1. **True Positive Rate (TPR):**
   * TPR measures the proportion of positive instances correctly identified by the model. In face recognition, TPR indicates the system's ability to correctly identify faces.
2. **False Positive Rate (FPR):**
   * FPR measures the proportion of negative instances incorrectly classified as positive by the model. In face recognition, FPR indicates the likelihood of incorrectly identifying non-face images as faces.

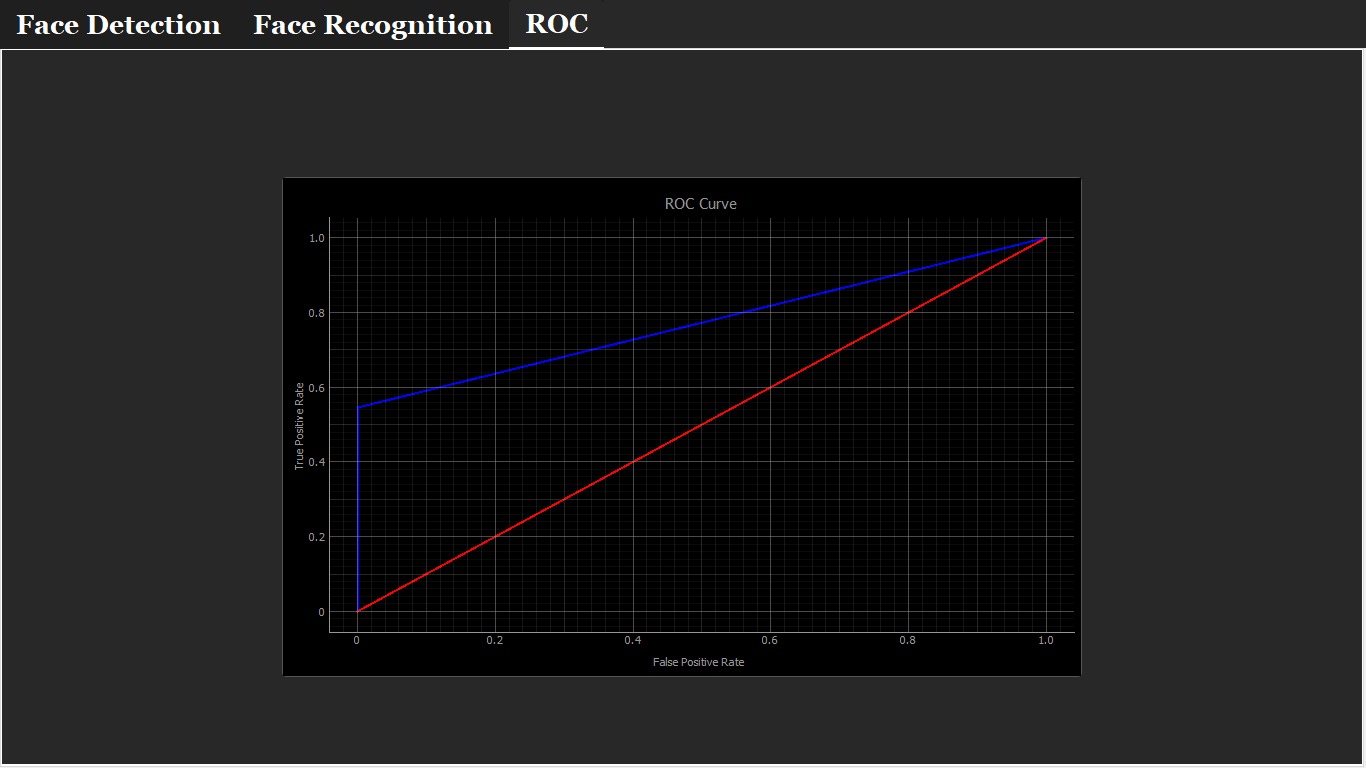
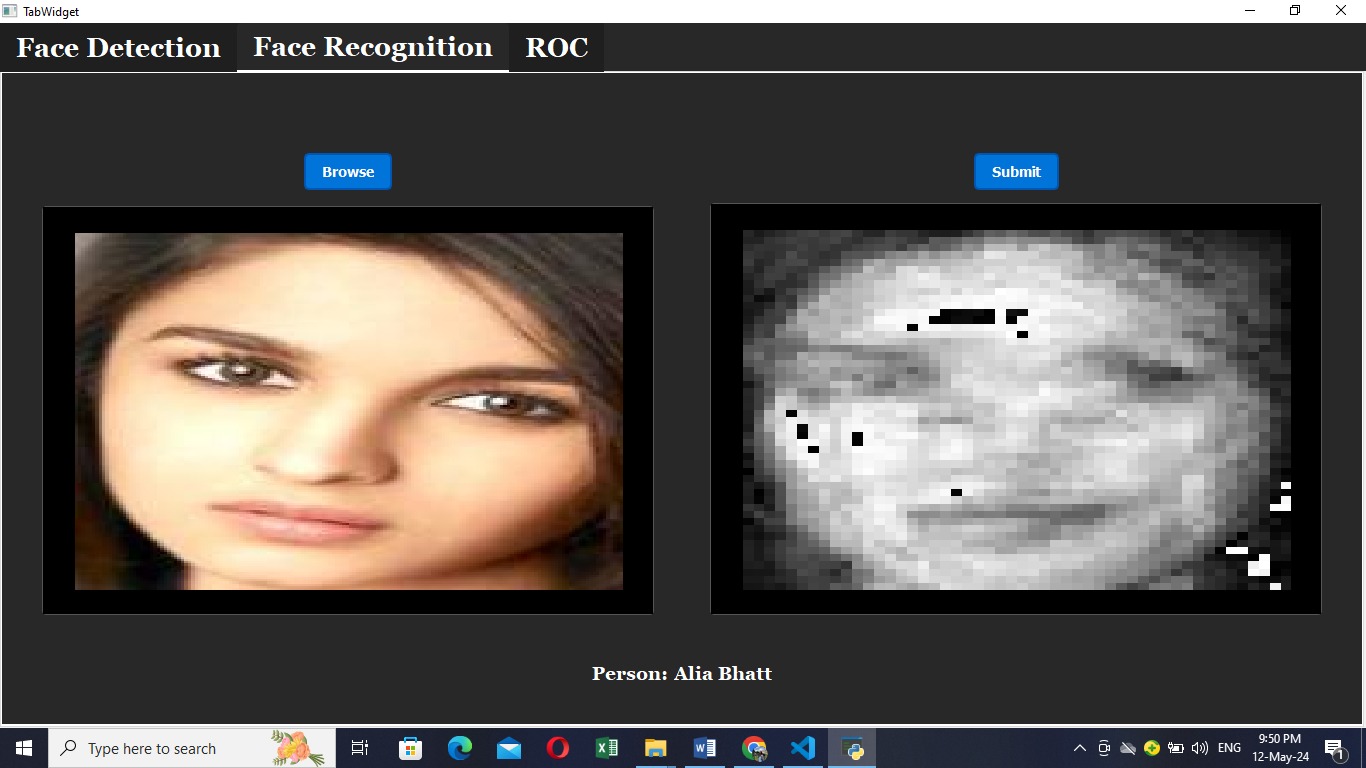
**Calculating TPR and FPR:**

1. **True Positive Rate (TPR):**
   * TPR = TP / (TP + FN), where TP is the number of true positives and FN is the number of false negatives.
2. **False Positive Rate (FPR):**
   * FPR = FP / (FP + TN), where FP is the number of false positives and TN is the number of true negatives.

**Plotting the ROC Curve:**

1. **Gather Predicted Labels and True Labels:**
   * Calculate predicted labels and true labels for the test dataset.
2. **Calculate TPR and FPR:**
   * Vary the classification threshold and calculate TPR and FPR at each threshold.
3. **Plot the ROC Curve:**
   * Plot TPR (y-axis) against FPR (x-axis) to create the ROC curve. A diagonal line represents random guessing, while the ROC curve above the diagonal indicates better-than-random performance.
   * Calculate the AUC-ROC score to quantify the model's performance.

**Results of calculating roc :**

****

