SBE3230 - Computer Vision
Team#03 *Sun., May 12th, 2024*

Task 5 Report - Face Detection and Recognition

1) Face Detection Using Haar-Cascade

We will use haar cascade with pre-trained XML files. Haar Cascade is not very computation-heavy; hence it is popular for small devices with small computation power.

We can load haar-cascade XML files using **`cv2.CascadeClassifier`** function.

We can call the detector function once the XML file is loaded using **`detectMultiScale.`**

Parameters in **`detectMultiScale`**:

- **`scaleFactor`** – This tells how much the object's size is reduced to the original image.

  The line *scale_factor = max(scale_factor, 1.2)* in the code ensures that the scale_factor used for face detection is always greater than or equal to 1.2.

  In the ***detectMultiScale*** function of OpenCV, the ***scaleFactor*** parameter determines how much the image size is reduced at each image scale. A smaller ***scaleFactor*** will increase the chance of detecting faces but will also increase the computation time. Conversely, a larger ***scaleFactor*** will decrease the chance of detecting faces but will reduce computation time.

  By setting *scale_factor = max(scale_factor, 1.2)* , the code ensures that the scale factor used for face detection is not too small. If ***scale_factor*** provided by the user is less than 1.2, it's overridden with a value of 1.2. This ensures that the face detection process is not overly aggressive in scaling down the image, potentially missing smaller faces, and also prevents excessive computation time caused by very small scale factors.

- **`minNeighbors`** – This parameter tells how many neighbors should contribute in a single bounding box.

This parameter specifies how many neighbors each candidate rectangle should have to retain it. It's used to control the quality of detected faces. Higher values result in fewer detections but with higher quality.

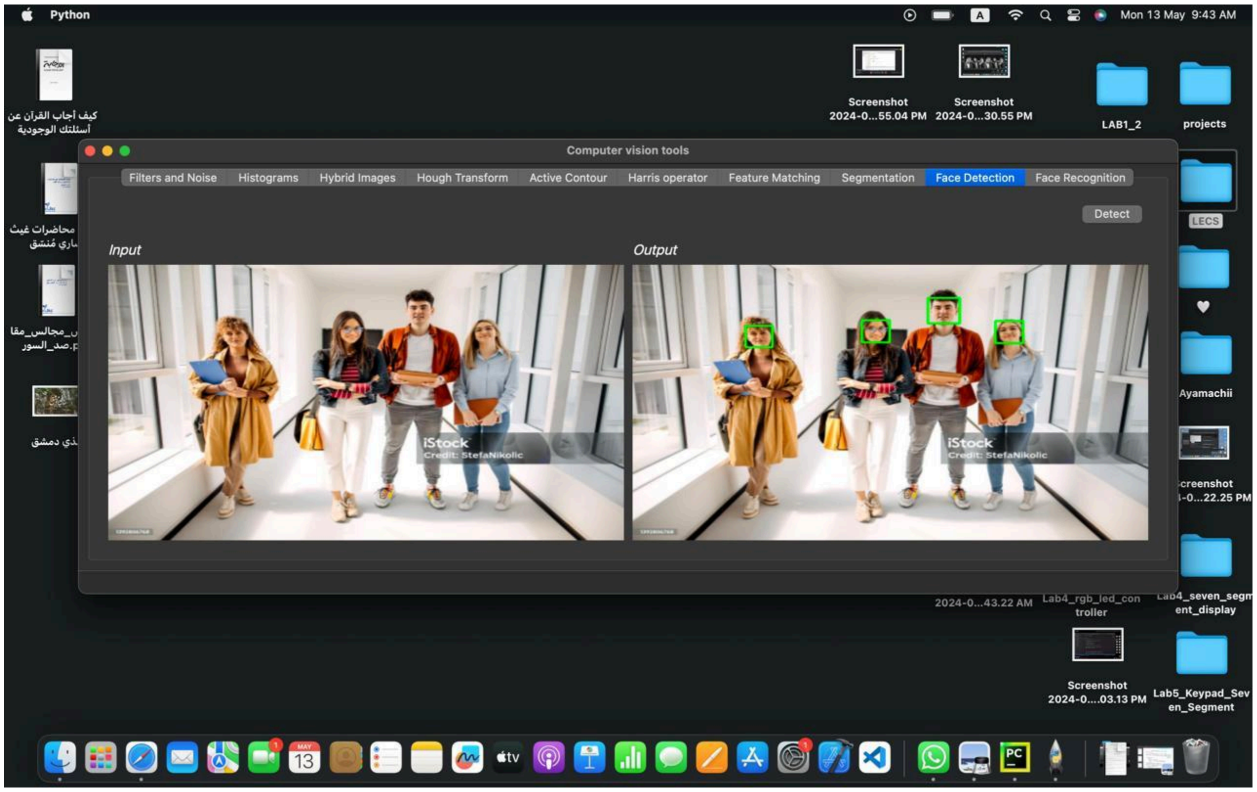
- `minSize`—This signifies the minimum possible size of the object in our image. if our object is smaller than the **minSize** it will be ignored.

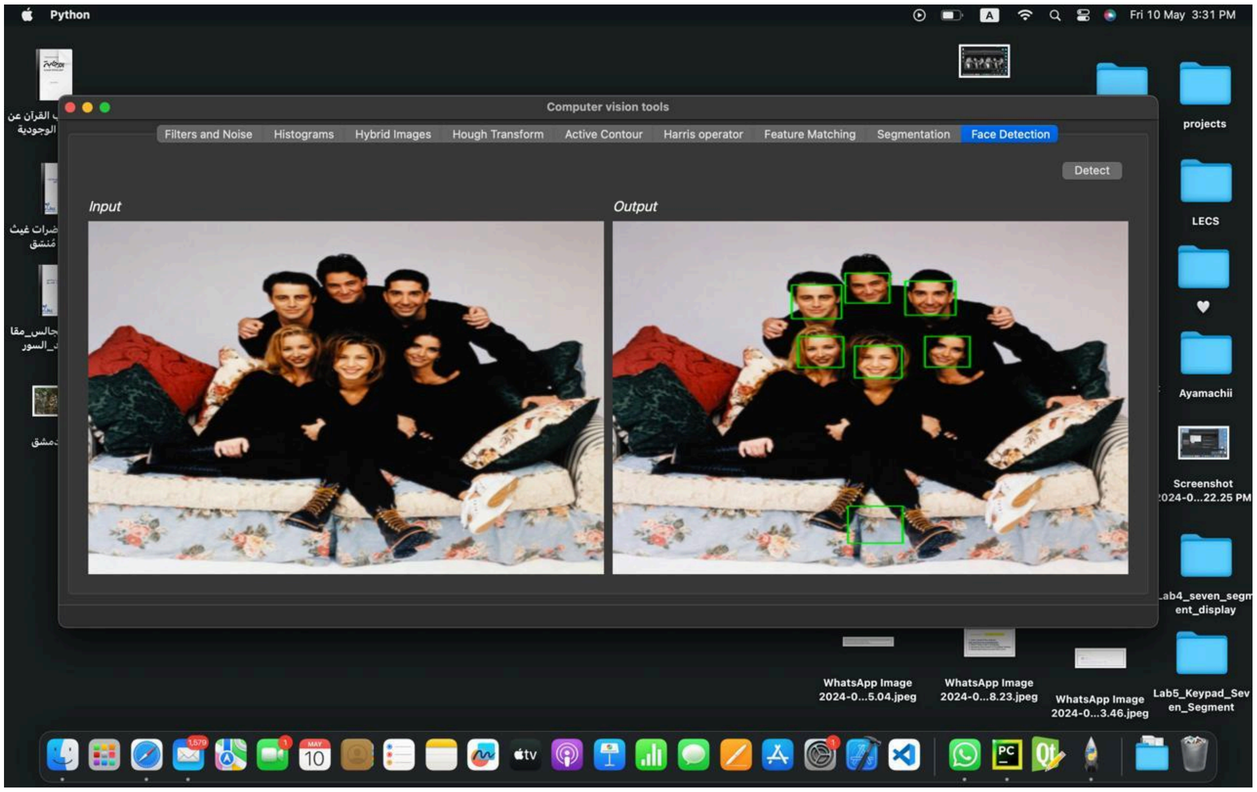
  This parameter specifies the minimum possible object size. Objects smaller than this size are ignored. It's a tuple (width, height) representing the minimum window size for face detection. This parameter helps in speeding up the detection process by ignoring regions that are smaller than the specified size.

  The ***detectMultiScale*** method returns a list of rectangles where it has detected faces. Each rectangle represents a detected face, defined by its ***(x, y)*** coordinates of the top-left corner, and its width ***w*** and height ***h***.

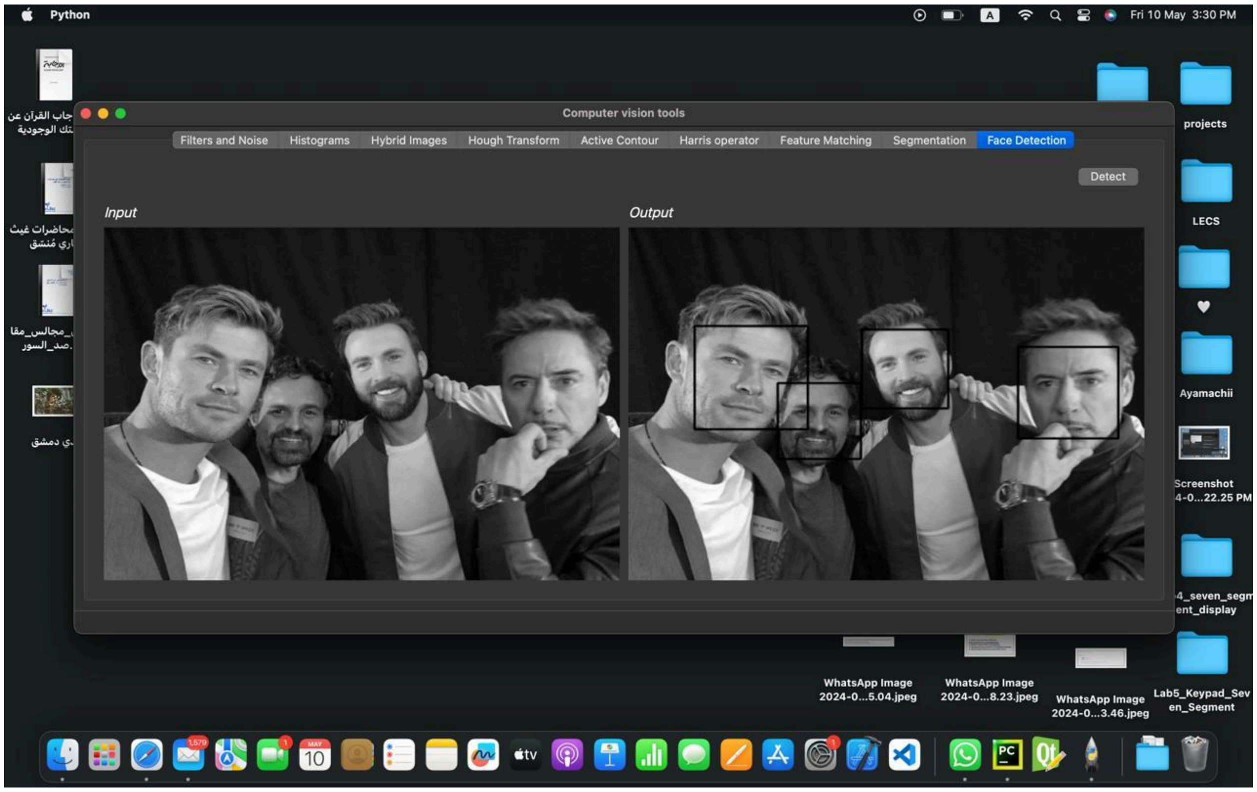
Output:

RGB image:


Computer vision tools
Filters and Noise
Histograms
Hybrid Images
Hough Transform
Active Contour
Harris operator
Feature Matching
Segmentation
Face Detection
Face Recognition
Detect
Input
Output


Computer vision tools
Filters and Noise
Histograms
Hybrid Images
Hough Transform
Active Contour
Harris operator
Feature Matching
Segmentation
Face Detection
Detect
Input
Output

Grayscale Image:



2) Face Recognition

Face recognition has numerous applications in various domains, including security systems, surveillance, and human-computer interaction. The PCA algorithm is widely used in face recognition due to its effectiveness in dimensionality reduction and feature extraction.

Dataset

The dataset comprises a total of 15 subjects. It is divided into training and test sets. Each subject within the dataset has 9 images assigned for training purposes and 2 images per subject are reserved exclusively for testing the trained model's performance.

Steps to perform face recognition using PCA

1. Data Preparation:

The training images are loaded. Each image is associated with a corresponding label indicating the person's identity. The training images are resized and reshaped to ensure consistent dimensions for further processing.

2. Mean Image Calculation:

Once the training images are reshaped, the mean image is calculated. This mean image represents the average facial appearance across the training set and serves as a reference for comparison.

3. Subtracting Mean Image:

Subtracting the mean image from each training image helps remove the global lighting conditions and highlights the local facial features. This step normalizes the images and reduces the influence of lighting variations.

4. Principal Component Analysis (PCA):

PCA is applied to the subtracted images. PCA decomposes the data matrix into eigenvectors and eigenvalues, capturing the most significant variations in the data. The top-k eigenvectors are selected based on their corresponding eigenvalues, representing the principal components.

5. Mapping Images to Components:

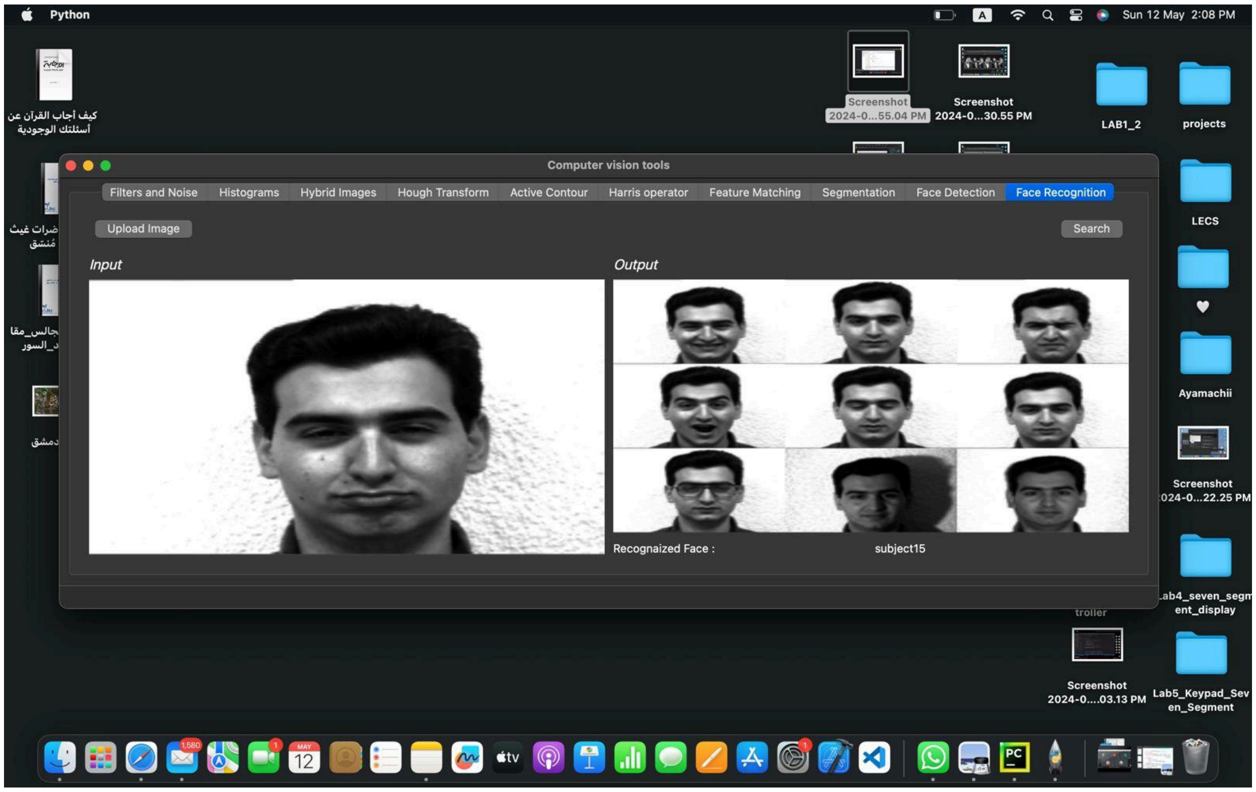
The subtracted training images are projected onto the selected eigenvectors This mapping process represents each image as a linear combination of the principal components, effectively reducing the dimensionality of the data.

6. **Face Recognition:**

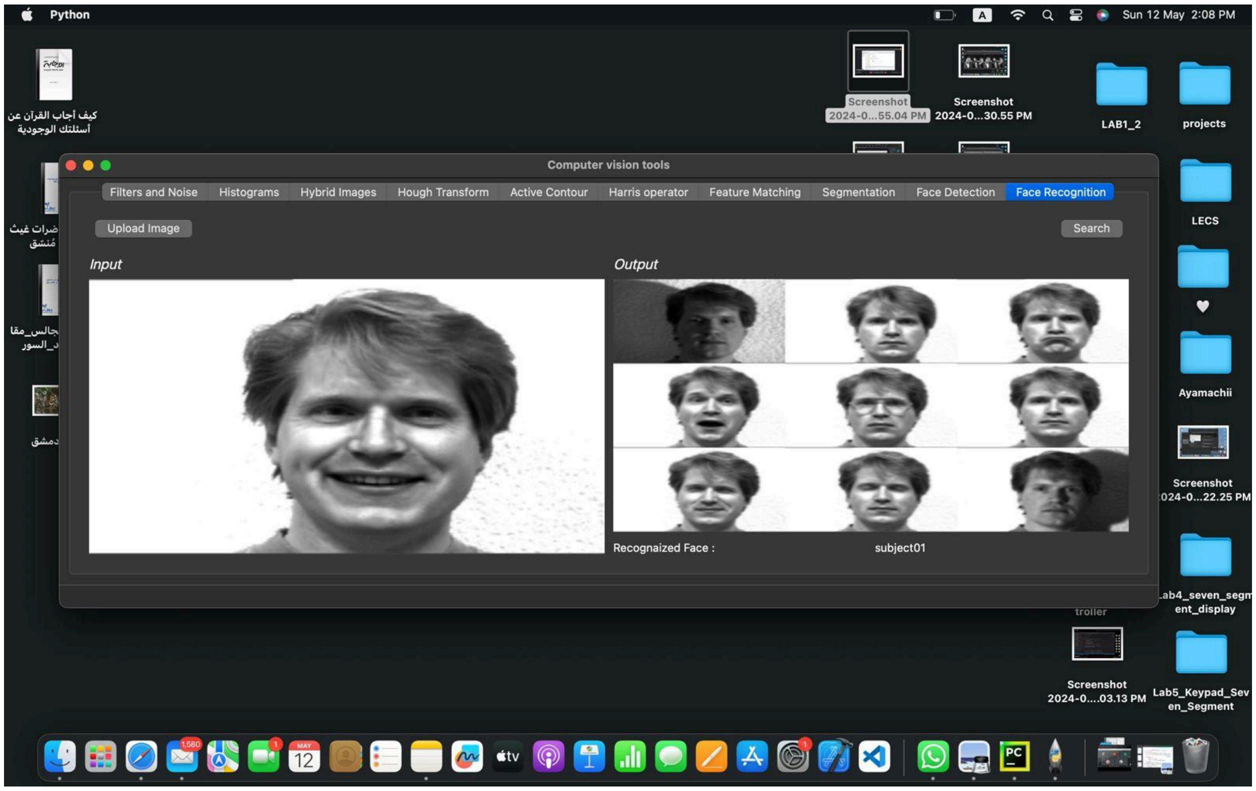
To recognize a face in a test image, the test image is compared with the training set by calculating the distance between the test image's projected components and the components of each training image. A minimum distance criterion is used to determine if a match is found.

Output:

Images Recognized for each test image:

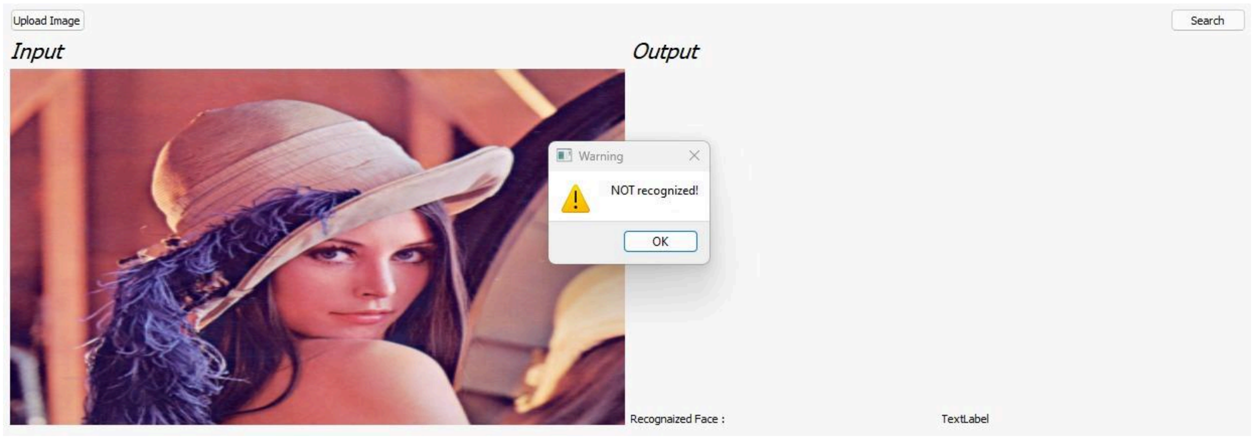


Recognaized Face : subject15

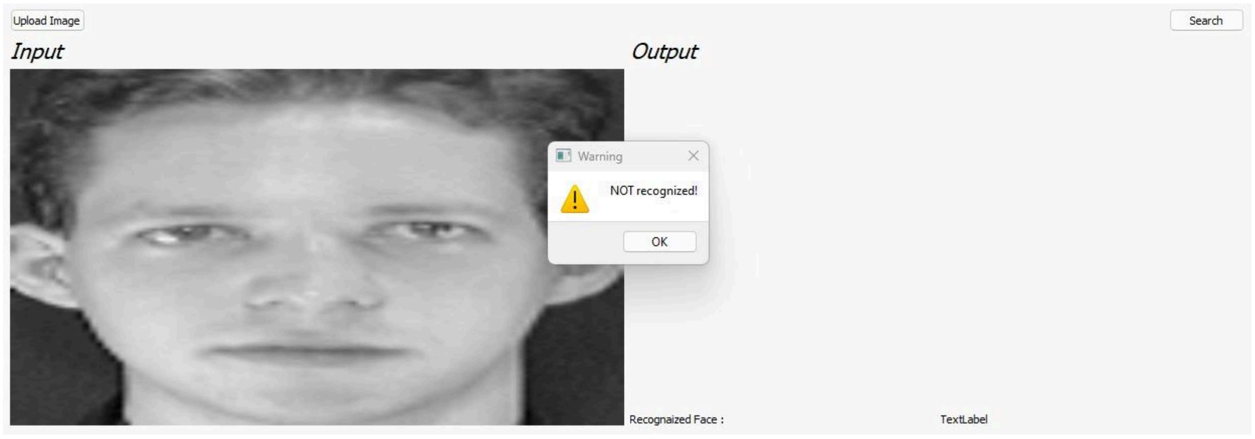


Recognaized Face : subject01

Images Not in Training Data:



NOT recognized!



NOT recognized!

3) Performance Evaluation (ROC and Accuracy)