

Computer Vision

Task 5

Face Detection and Recognition

**Team 3**

**Submitted to:**

**Eng. Laila Abbas**

**Contact Email: amira.gamal.ag22@gmail.com**

|  |  |  |
| --- | --- | --- |
| **BN** | **Section** | **Name** |
| **10** | **1** | **Esraa Mohamed Saeed** |
| **12** | **1** | **Alaa Tarek Samir** |
| **15** | **1** | **Amira Gamal Mohamed** |
| **8** | **2** | **Fatma Hussain Wageeh** |
| **26** | **2** | **Mariam Mohamed Osama** |

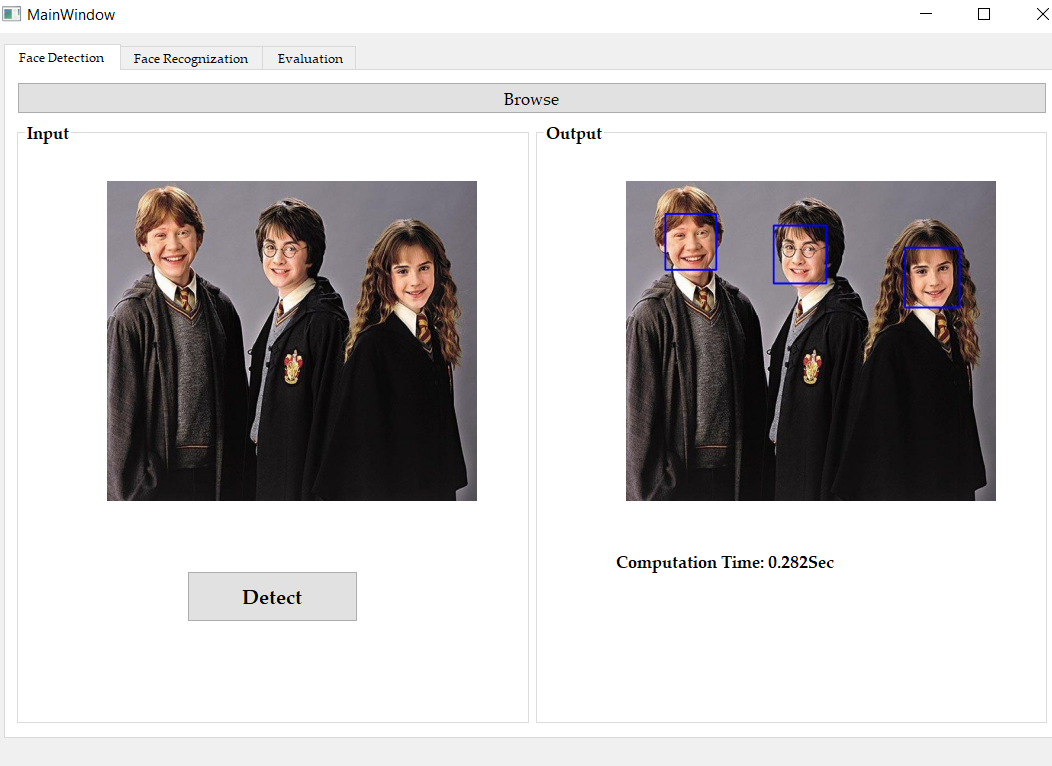
**Submitted by:**

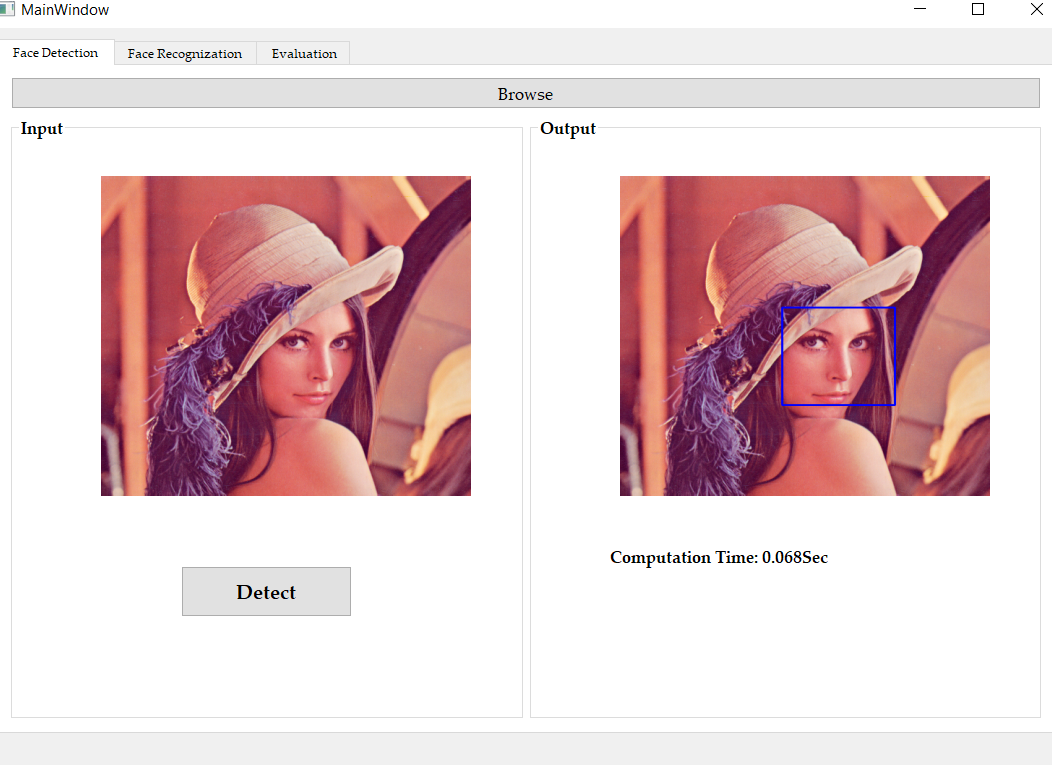
**Face Detection**

Face detection applications use algorithms and ML to find human faces within larger images, which often incorporate other non-face objects such as landscapes, buildings and other human body parts like feet or hands. Face detection algorithms typically start by searching for human eyes -- one of the easiest features to detect. The algorithm might then attempt to detect eyebrows, the mouth, nose, nostrils and the iris. Once the algorithm concludes that it has found a facial region, it applies additional tests to confirm that it has, in fact, detected a face.

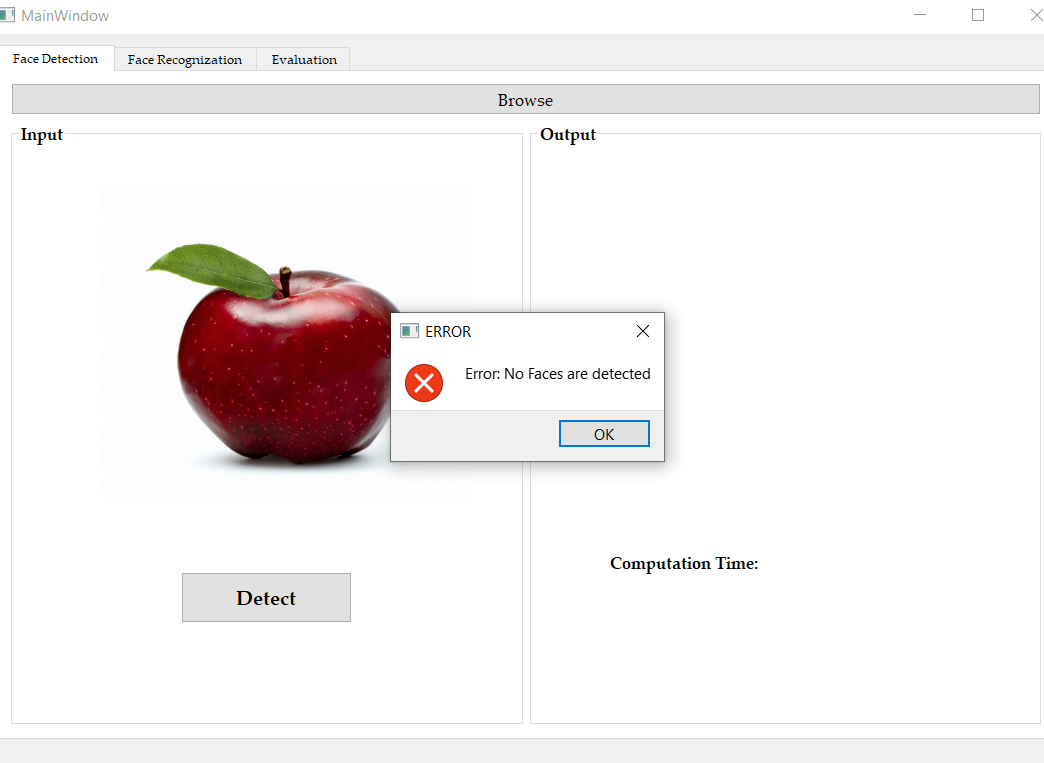
We load the face cascade \_which is just an XML file that contains the data to detect faces­­\_ into memory so it’s ready for use.

**Results and Computation Time**





If there is no face:



**Face Recognition by Eigen Faces**

**Eigen Faces**

An eigenface is the name given to a set of eigenvectors when used in the computer vision problem of human face recognition.

The eigenvectors are derived from the covariance matrix of the probability distribution over the high-dimensional vector space of face images. The eigenfaces themselves form a basis set of all images used to construct the covariance matrix. This produces dimension reduction by allowing the smaller set of basis images to represent the original training images. Classification can be achieved by comparing how faces are represented by the basis-set.

**How to Get Eigen Faces and Recognize the Face**

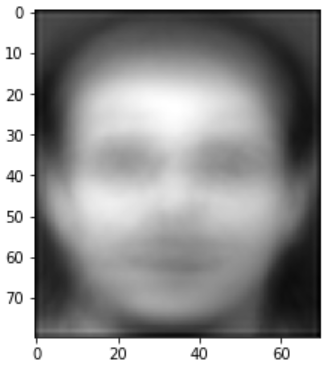
First, we trained the model:

1.-Flat the black and white images of the training set (from matrices (2D) to vectors (1D))



2.-Calculate the mean and get the mean face.

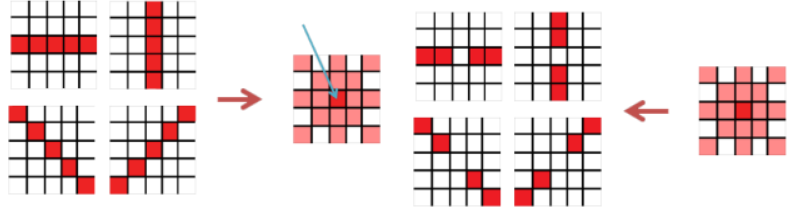
The mean is just the sum of all of the pictures divided by the number of pictures. As a result, we will have a “mean face”.



3.-Normalize the training set: for each image, subtract the mean.

To normalize the training set, we just simply need to subtract for each picture in the training set the mean that was calculated in the previous step.

The reason why this is necessary is because we want to create a system that is able to represent any face. Therefore, we calculated the elements that all faces have in common (the mean). If we extract this average from the pictures, the features that distinguish each picture from the rest of the set are visible.



4.-Calculate the covariance: multiply all images by themselves.

The covariance represents how two variables change together. After the previous step, we have a set of images that have different features, so now we want to see how these features for each individual picture change in relation to the rest of the pictures.

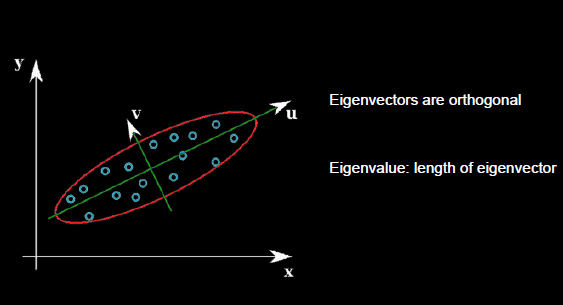
For this purpose, we put all the flat normalized pictures together in a vector. My training set consists of 328 pictures whose dimensions are 80×70 pixels. Therefore, the resulting matrix will be 5600×328. The covariance is the multiplication of this matrix by itself, and if we transpose it correctly, the resulting matrix will be 16×16:

328x5600 \* 5600x328 = 328x328

5.-Extract eigenvectors from the covariance.

From the covariance we can extract the eigenvectors.

Note: The first eigenvector will describe more information than the second and so on. For this reason, later we have to pick the first eigenvectors generated (avoiding noise).



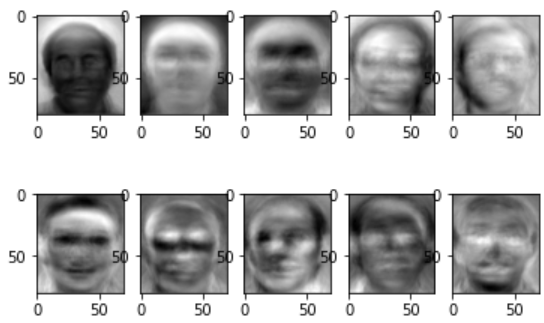
6.-Calculate eigenfaces: eigenvectors x normalized pictures.

Each eigenvector is multiplied by the whole normalized training set matrix and as a result, we will have the same amount of eigenfaces as images in our training set.

7.-Choose the most significant eigenfaces.

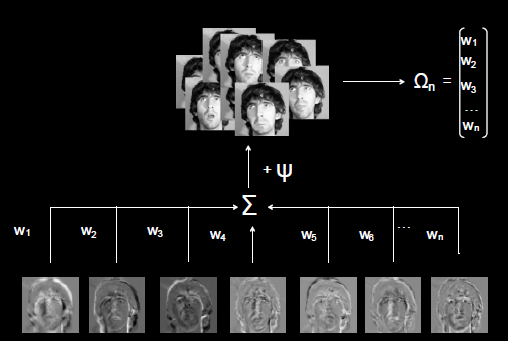
The first eigenfaces represent more information than the last eigenfaces. Actually, the last eigenfaces only add noise to the model, so it is necessary to avoid them.

Sample of them:



8.-Calculate weights: chosen eigenfaces x normalized pictures.

Each normalized face in the training set multiplies each eigenface. Consequently, there will be N set of weights with M elements (N = number of pictures in the training set, M = number of eigenfaces).



Second, we recognized the test face:

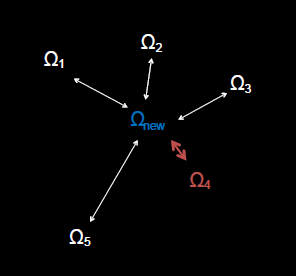
9.-Vectorize and normalize this picture: subtract the calculated mean from the picture.

Reshape the test picture into a vector and subtract the mean calculated in 2) from it.

10.-Calculate the weights: multiply the eigenfaces x normalized picture.

The same as 8) but with the test picture.

11.-Interpret the distance of this weight vector in the face space and get the matched one (the closest distance).



**Results ana Computation Time**

There are some images with their best matched one

