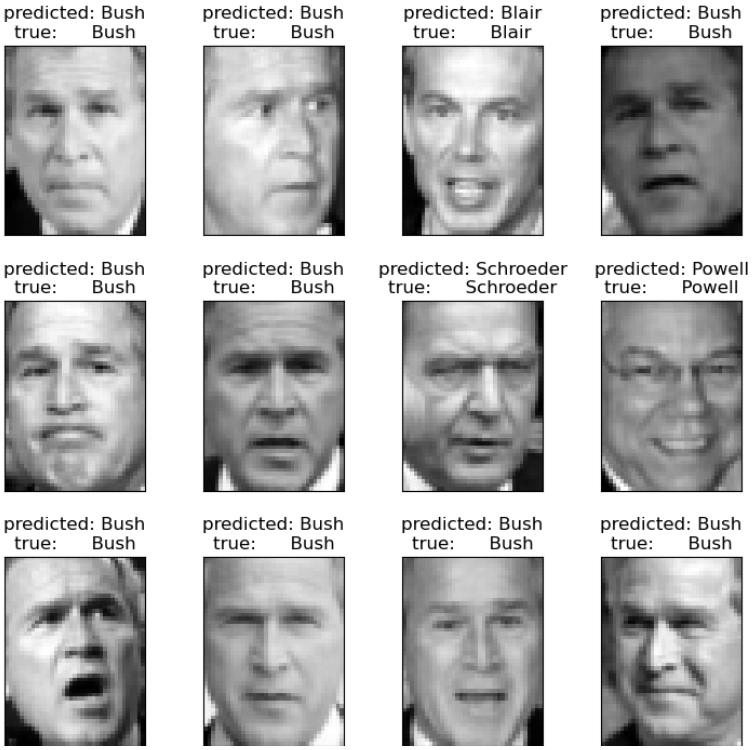
SMAI Project Report

Eigen Faces



LINK TO PROJECT REPOSITORY:

https://github.com/aravind-3105/SMAI-Family-Project

Team Family (8)

Aravind Narayanan - 20192014

Abhayram A Nair - 2019102017

Vaibhav Bhushan - 2019112019

Rishin Chakraborty - 2019112008

In this project, we study some of the most popular recognition techniques namely, Eigenfaces and Fischerfaces. Eigenfaces incorporate Principal Component Analysis (PCA), i.e., it uses a small subset of features to describe facial features for recognition. Fischer faces on the other hand uses Linear Discriminant Analysis (LDA) for dimensionality reduction.

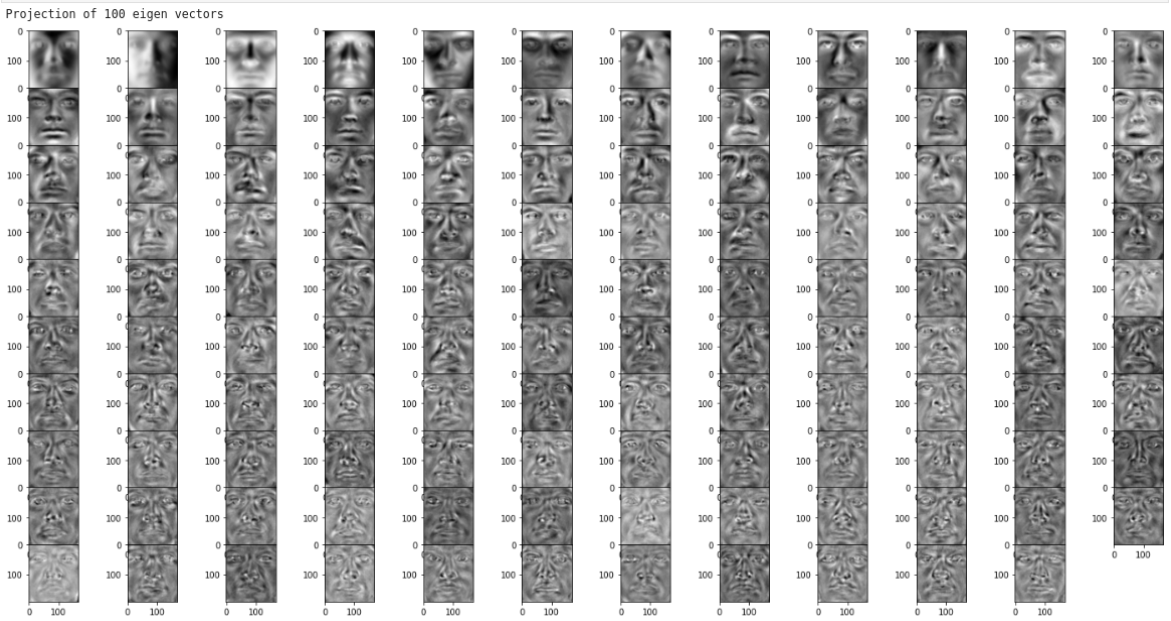
Overall Goals:

The primary goal of this project was to implement the method of Eigenfaces for face recognition by projecting the face images on the feature space (face space) which best represents the variations among distinct faces. The face space is defined as the “Eigenfaces ", which are the eigenvectors of the set of faces. The model will have the ability to learn new faces in an unsupervised manner. We also implement Fischerfaces. This was done using Linear Discriminant Analysis. We have also compared the results of these two algorithms with another well-known method (LBPH).

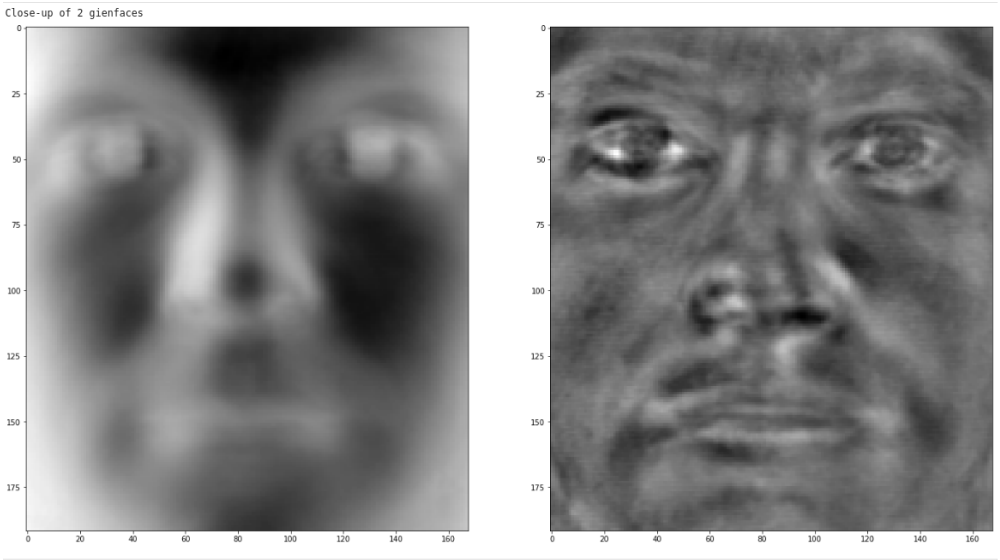
The approach used for Eigenfaces:

The gist of the approach is that the faces are reduced to vectors that preserve only those characteristics of the face which are useful for recognition. This set of reduced vectors are called Eigenfaces and all other faces are expressed as a linear combination of these faces.

1. Obtain all images given in the dataset and store them into vector X.
2. An average ‘face’ is calculated. This average is then subtracted from each of the vectors and the resulting difference is stored in an N^2xM matrix. This matrix is storing the variation from the average for each face vector. We call this matrix A.
3. We need to find eigenvalues and eigenvectors of the covariance matrix AA^T. The issue with this is that the matrix will have size N^2xN^2 and will be computationally very expensive. Here we can apply a useful relation to simplify the computation. The eigenvalues for AA^T will be the same as the ones for A^TA.
4. Another useful relation is that the eigenvectors v of the original covariance matrix are related to the eigenvectors v' of the reduced covariance matrix as $v = Av'$. We calculate the eigenvalues and eigenvectors for the smaller covariance matrix and use this relation to map them to the original covariance matrix. This gives us M eigenvectors.



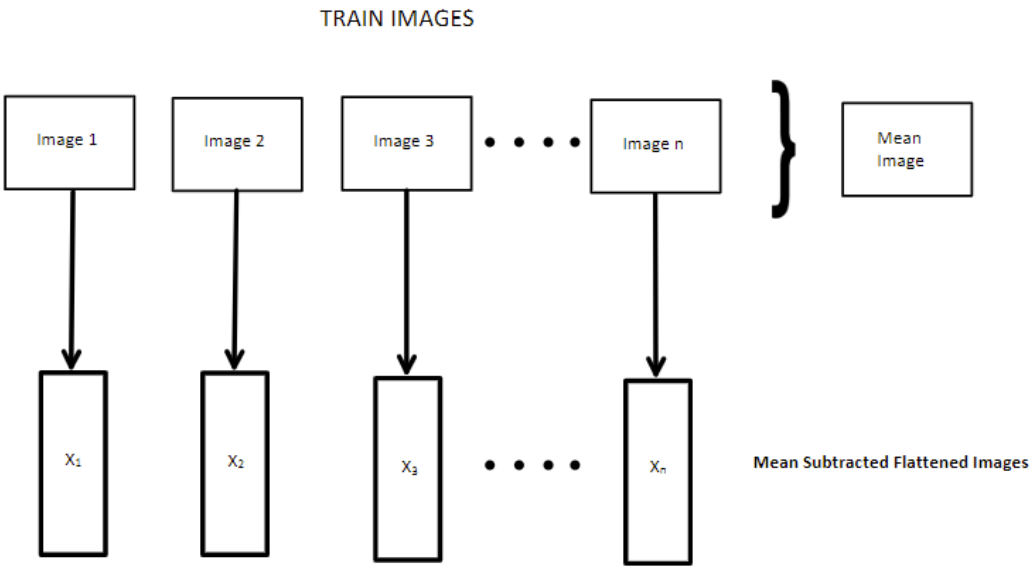
5. Optionally, we reduce these *M* vectors to the *K* best eigenvectors. These are just the *K* largest eigenvectors. These will be able to capture maximum variations from the average face. These resultant eigenvectors are called ***eigenfaces***.

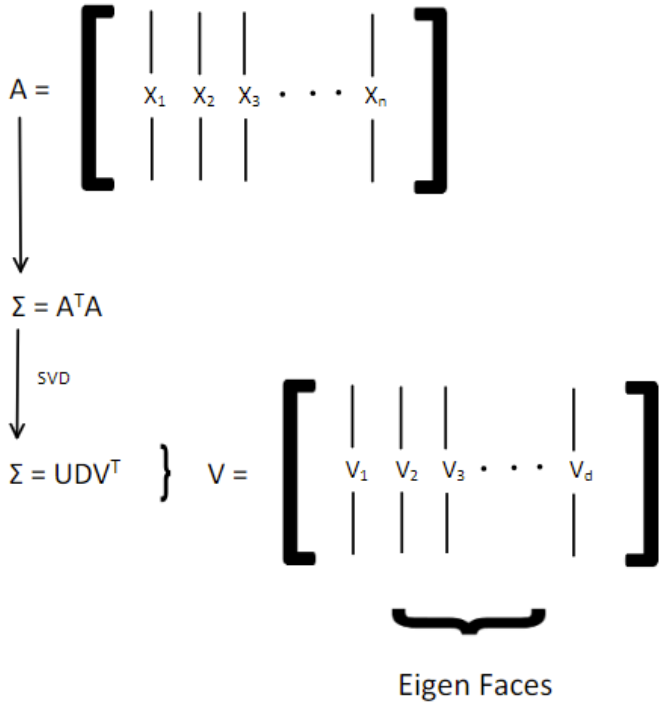


6. Different combinations of K(10,15,20) are shown for two different images in the jupyter notebook.

7. All the training set faces can now be represented as a linear combination of the eigenfaces. Each face is now stored as a vector containing the coefficients of this linear combination.
8. Whenever a new image is received, we convert this image into a linear combination of eigenfaces and do a nearest neighbor analysis with existing training set faces in this space to get the final prediction.

Visualizing the approach:



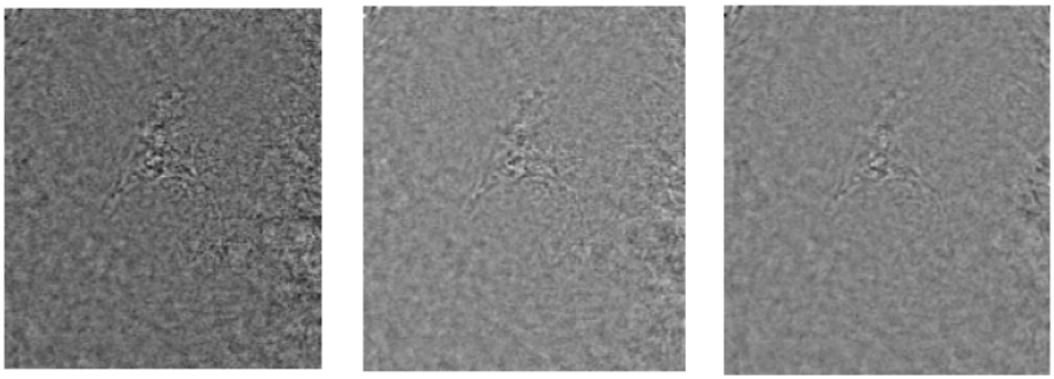


The approach used for Fischer faces:

In the previous section, we saw PCA is used to extract features that are useful in facial recognition. Now, we extract the best feature vectors to discriminate between classes using Linear Discriminant Analysis which are called Fischer faces. For prediction of the class given a face image, the distance between the projections of training data and input image is taken, the class with minimum distance is predicted as the result.

1. We start with converting the images into matrix form (X), where it can be arranged row-wise or column-wise. Each image has its corresponding class stored in a vector (y).
2. Let's say we have c number of classes, N is total number of samples, next we project X into (N-c) dimensions using PCA and obtain eigenvectors W_{PCA} .
3. We now calculate S_w and S_b , within class & between class scatter respectively, and apply LDA and maximize the Fischer index which is the ratio of the determinant of between class and within class scatter.
4. The solution obtained is a set of eigenvectors W_{FLD} , and hence we get $W = W_{PCA} * W_{FLD}$, where W is called Fischer Faces.

Closeup example of Fischer faces (yaleB face database used)

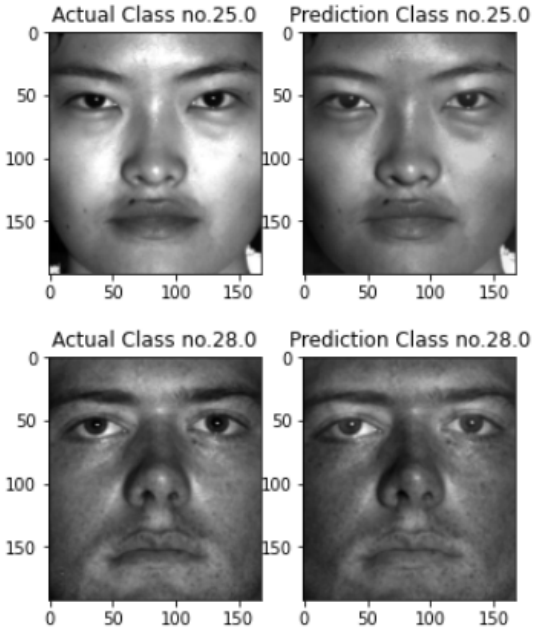


We then trained the model and tested using 80% training/20% testing. Following accuracy values were obtained in 10 trials:

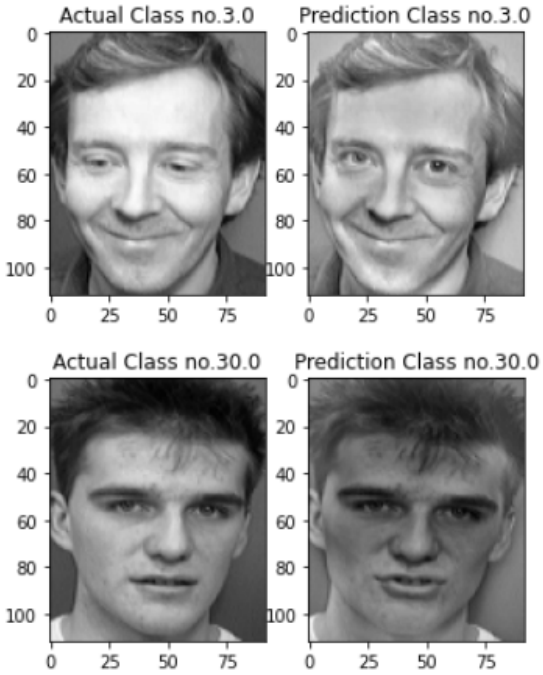
Trial Number	Accuracy
1	81.70
2	81.73
3	73.93
4	71.43
5	72.93
6	68.35
7	79.37
8	77.48
9	86.17
10	67.79

We get average accuracy of 76.08%

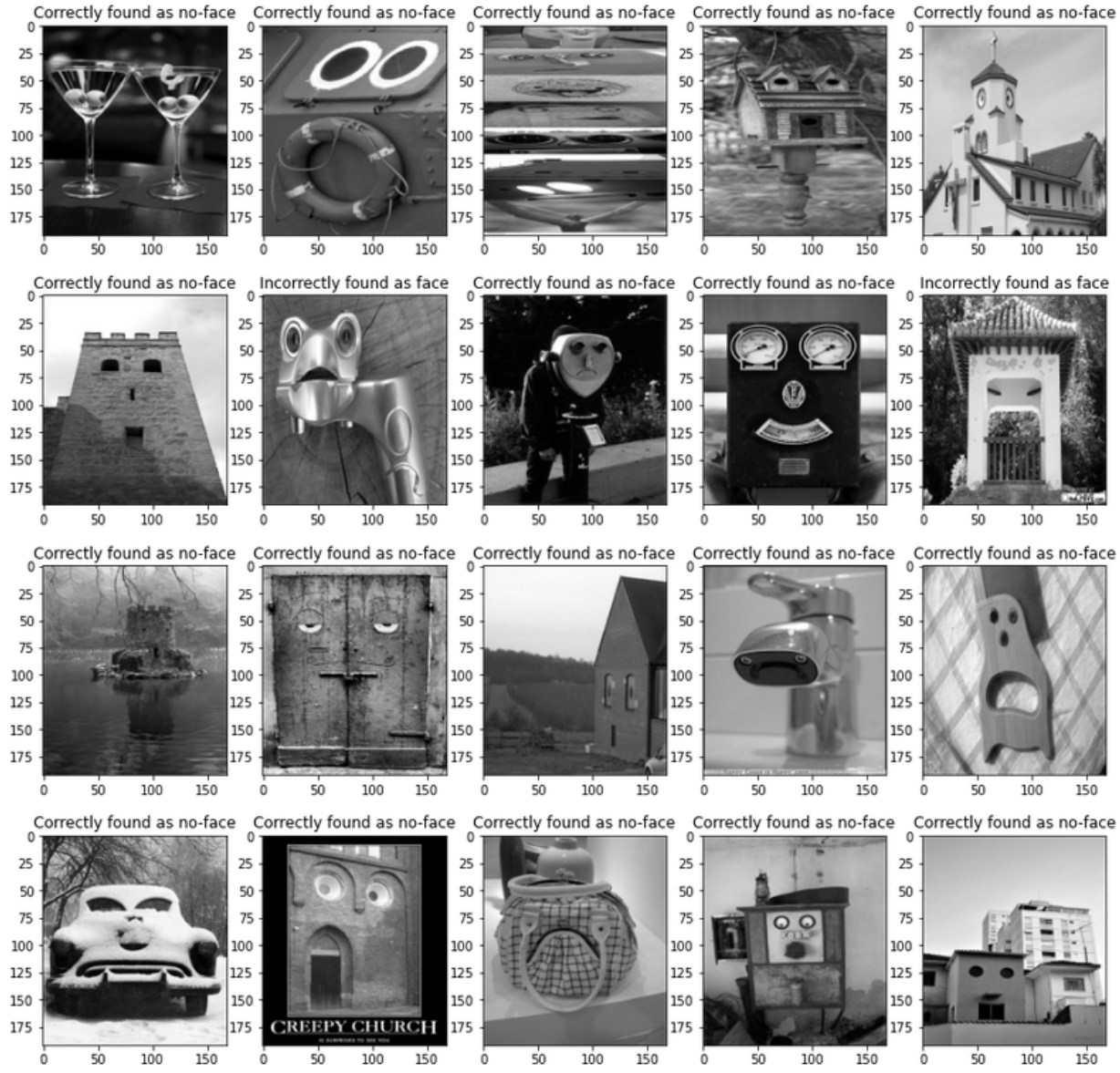
Classification by eigenfaces in YaleB dataset:



Classification by eigenfaces in AT&T dataset:



Checking by eigenfaces for non-faces:



Performance table on different datasets and operations:

Dataset and Operation	Accuracy
AT&T Face, Face Recognition	95%
Yale B Face, Face Recognition	84.21%
No-Face Dataset, Face Detection	90%
Combination of No-Face and AT&T, Face Detection	81.25%

Other Methods of Face Recognition Used:

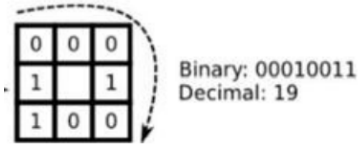
For comparison, we train and test two other algorithms on the same dataset and compare the results. We have used pre-built implementations for these two algorithms. As eigenfaces and fisherfaces are a little old, we chose two more recent face recognition methods.

1. The first algorithm is the Local Binary pattern Histogram algorithm described in Ahmed et.al, 2018[3]. It involves dividing the image into blocks of a certain size, then applying a Local Binary Pattern operator to its subset pixels, which gives us a histogram for the features seen in that block. A basic implementation of LBP is as follows:

   Suppose we have a set of 9 pixels with grayscale values as follows:

   When LBP operator is applied to it, we use the central pixel as threshold(here, 5) and we assign 1 to all pixels with a greater or equal value and 0 to all pixels with a lower value. As a result, we get the following:

Reading the pixel values in clockwise order we get a binary number. This number will differ based on the shape of the feature in this set of pixels, so it gives an idea of the feature. The paper uses a more advanced operator called circular LBPH. This operator has the advantage of not being limited to a fixed number of pixels. It generates an eigenvalue for each pixel and the histogram of these values is used as the feature vector for classification.

We used the implementation from [4]. This method had an accuracy of **96.25%** on the AT&T dataset.

2. We have used an OpenCV version [5] of Eigenfaces to compare our implementation with, we tested it on the AT&T database, and obtained an accuracy of **97.5%**.

Final Deliverables:

1. Analysis of eigenfaces and Fischer faces accuracy
2. A prediction system that tells us whether a face is detected
3. A prediction system that tells us which person was detected in the database, if a face is detected
4. Comparison with other well-known methods for face recognition
5. Final presentation and description of our project along with instructions to run the code.

Timeline and Milestones:

Timeline	Milestones
26th October 2021	**Project Allocation**
7th November 2021	**Project Proposal**
Week - 1	**Assigned paper reading and Fischer faces paper formalized the structure of basic implementation**

Week - 2	• Implementation of both the core methods from scratch. • Reading papers about the other comparison methods.
19th November 2021	Mid-Evaluation
Week - 3	Comparison of the two methods and tabulating them
Week - 4	Comparison with other well-known methods
8th December 2021	Final Evaluation

Work Distribution:

- **Abhayram -** implementation of PCA, generating eigenfaces, testing on YaleB dataset
- **Aravind-** implementation of eigenfaces on AT&T dataset, comparison of faces, no faces
- **Rishin-** implementation of Fischer faces functions and implementation of the inbuilt functions for eigenfaces
- **Vaibhav -** implement the Fischer faces on the YaleB, LBPH on YaleB

References:

1. Face recognition using Eigenfaces (https://ieeexplore.ieee.org/document/139758)
2. Eigenfaces vs. Fisherfaces: recognition using class specific linear projection (https://ieeexplore.ieee.org/document/598228)
3. LBPH Based Improved Face Recognition At Low Resolution (https://ieeexplore.ieee.org/document/8396183)
4. LPBH Implementation (https://github.com/opencv/opencv_contrib/blob/4.x/modules/face/src/lbph_faces.cpp)
5. Eigenfaces using OpenCV (https://github.com/opencv/opencv_contrib/blob/4.x/modules/face/src/eigen_faces.cpp)