

PCA-Based Facial Recognition Using Eigenface Algorithm in Scilab GUI

Sanyam Bhavsar

Int. M.Tech Student in Data Science and Computation VIT Bhopal University

Image Processing

July 2026

Abstract

Facial recognition systems are a critical component of modern biometric security and access control. While deep learning architectures dominate the current landscape, classical statistical methods remain highly efficient, computationally lightweight, and mathematically transparent. This case study implements the Eigenfaces algorithm for face recognition using Principal Component Analysis (PCA) entirely within Scilab. To make the system interactive and user-friendly, a custom Graphical User Interface (GUI) was developed natively in Scilab, allowing users to train the model, upload test images, and view recognition results in real-time without touching the command line.

The project uses the AT&T ORL Face Database, which contains 400 grayscale images of 40 different subjects. To ensure the system actually learns to generalize rather than just memorizing pictures, we split the data: the model is trained on the first 8 images of each person (320 images) and the remaining 2 images per person are hidden for testing. The workflow is straightforward but mathematically rigorous. First, all images are flattened into high-dimensional vectors and an "average face" is calculated. Then, we use a linear algebra trick to extract the top 50 "Eigenfaces"—the core underlying patterns that distinguish one face from another. When a user uploads a new photo through the GUI, the system projects that face onto the 50 Eigenfaces, compressing 10,304 pixels into just 50 numbers. It then measures the Euclidean distance between these numbers and the training set to find the

closest match. The GUI also features a reconstruction module, visually proving that a complex facial image can be heavily compressed into 50 weights while still retaining its exact identity.

This project is a partial implementation of the Eigenfaces algorithm. It implements the core PCA-based recognition but excludes real-time video, head tracking, and unknown face detection.

1. Introduction

The demand for contactless authentication is rising and facial recognition systems are being developed rapidly. One effective way to solve this problem is the Eigenfaces algorithm, first proposed by Turk and Pentland in 1991. This method uses Principal Component Analysis (PCA) to extract the most significant variations from a dataset of human faces, reducing massive image data into a small set of essential numbers.

In this project, a facial recognition engine was modeled and simulated using the Scilab environment. The implemented model focuses on analyzing the performance of PCA for dimensionality reduction and feature extraction. The project uses a direct matrix-mathematics approach inspired by the concepts presented in the reference paper. The study focuses entirely on classical linear algebra rather than modern AI architectures.

The study focuses on comparing a new, unseen test image against a pre-computed database of training weights. The system is wrapped in a Graphical User Interface (GUI) to provide a practical and interactive user experience. Both the training and recognition phases are tested under the same operating conditions, and their performance is evaluated based on recognition distance, reconstruction accuracy, and overall model accuracy. The main objective is to demonstrate the effectiveness of PCA in reducing high-dimensional image data while maintaining highly reliable recognition performance.

This project is a partial implementation of the Eigenfaces algorithm. It implements the core PCA-based recognition but excludes real-time video, head tracking, and unknown face detection.

1. Problem Statement

Human faces have a lot of information in them. A simple grayscale image from the AT&T ORL database is made up of 112×92 pixels which means it has 10,304 pixel values for one image. Comparing these pixel values for hundreds of images requires a lot of power and

memory. Also small changes in lighting, facial expression, hairstyle or image quality can cause problems for methods that compare pixels directly leading to incorrect recognition results.

The main challenge is to create a face recognition system that can identify people accurately while using computational power. The system should only use the important facial information instead of looking at every pixel separately. This helps reduce the amount of storage needed makes recognition faster and improves the efficiency of the system.

To solve this problem we use Principal Component Analysis, which's a way to reduce the dimensionality of facial images. Principal Component Analysis finds the directions of variance among all the training images and transforms them into a new feature space called Eigenfaces. Of using 10,304 pixels to represent each face the system uses only 50 Eigenface coefficients, which greatly reduces the amount of data needed for recognition.

The project uses the Eigenfaces algorithm that was proposed by Turk and Pentland. When the system is being trained all the facial images are converted into vectors the average face is calculated the covariance matrix is generated using a method called $A^T A$ Eigenfaces are extracted and each training image is projected into the Eigenface space. When the system is recognizing faces, an unknown test image goes through the projection process and its feature vector is compared with the stored training vectors using Euclidean distance. The image with the distance is selected as the recognized person.

In addition to the core recognition algorithm a graphical user interface has been developed in Scilab to make it easier for users to interact with the system. The graphical user interface allows users to train the model upload a test image recognize faces reconstruct the face from its Eigenface coefficients and evaluate the systems overall accuracy. A special "Test Accuracy" function automatically processes all 80 test images compares the predicted identity with the true identity and reports the percentage of correct recognitions. This automatic evaluation provides a way to measure the reliability of the Principal Component Analysis-based system.

The main goal of this case study is to show that Principal Component Analysis-based Eigenfaces provide an reliable method, for face recognition while significantly reducing the computational cost compared to traditional methods that compare pixels. The inclusion of an automatic accuracy test further validates the methods performance under controlled conditions.

2. Basic concepts related to the topic

3.1 Digital Image Representation.

A digital image is made up of many small elements called **pixels**, where each pixel stores an intensity value between **0 (black)** and **255 (white)**. The AT&T ORL Face Database used in this project contains grayscale facial images with a resolution of **112 × 92 pixels**, giving **10,304 pixels** per image.

Before processing, each 2D image is converted into a one-dimensional column vector, known as **flattening**.

$$\Gamma = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{10304} \end{bmatrix}$$

This representation allows mathematical operations such as matrix multiplication and Principal Component Analysis (PCA) to be performed efficiently.

3.2 Face Recognition and Train/Test Split

Face recognition is a biometric technique used to identify a person by analyzing unique facial features. The process consists of two stages:

- **Training:** Learning important facial features from known images.
- **Recognition:** Comparing a new image with the trained database.

The AT&T ORL database contains **400 images of 40 people**. In this project, the first **8 images of each subject** are used for training (**320 images**), while the remaining **2 images per subject** are used for testing. This ensures the model is evaluated using images it has never seen before.

3.3 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimensionality reduction technique that converts high-dimensional image data into a smaller set of important features called **principal components**.

Each face initially contains **10,304 pixel values**, many of which carry redundant information. PCA keeps only the directions that contain the most useful facial information.

In this project, each image is represented using only **50 features (Eigenfaces)** instead of 10,304 pixels. This significantly reduces memory usage and computation time while preserving the important facial characteristics required for recognition.

3.4 Mean Face and Mean-Centering

All **320 training images** are arranged into a data matrix X . The average of all training faces is called the **Mean Face** and is calculated as:

$$\mu = \frac{1}{M} \sum_{i=1}^M \Gamma_i$$

where $M = 320$.

The mean face is then subtracted from every training image:

$$\Phi_i = \Gamma_i - \mu$$

This process is called **mean-centering**. It removes common information shared by all faces so that PCA focuses only on the differences between individuals.

3.5 Covariance Matrix and PCA Optimization

PCA normally requires computing the covariance matrix:

$$C = AA^T$$

Since this matrix would be 10304×10304 , calculating its eigenvectors directly is computationally expensive.

To improve efficiency, the Eigenfaces algorithm uses the smaller matrix:

$$L = A^T A$$

which is only 320×320 in this project. The eigenvectors of this smaller matrix are converted into the actual Eigenfaces, producing the same result while greatly reducing computation time.

3.6 Eigenfaces and Face Space

The computed eigenvectors are called **Eigenfaces** because they represent the most important facial variations in the training set.

Only the top **50 Eigenfaces** are selected to form the feature matrix E . Each centered face is projected into this lower-dimensional space using:

$$w = E^T \Phi$$

where w is a **50-dimensional weight vector** representing the face.

All training images are projected in the same way to create the training weight matrix:

$$W = E^T A$$

Instead of storing 10,304 pixel values, each face is now represented by only **50 weights**, making recognition much faster.

3.7 Euclidean Distance and Recognition

During recognition, a test image is projected into the same Face Space to obtain its weight vector. The similarity between the test image and every training image is measured using the **Euclidean distance**:

$$Distance = \sqrt{\sum (W - w_{test})^2}$$

The training image with the **smallest distance** is selected as the best match.

Since each subject contributes **8 training images**, the subject ID is determined using:

$$\text{Matched Subject ID} = \left\lceil \frac{\text{Minimum Index}}{8} \right\rceil$$

This nearest-neighbor approach allows the system to identify an unknown face by comparing its compact feature representation with the stored training database.

Adopted, Modified, and Not Implemented Features

This case study represents a partial implementation of the Eigenfaces-based face recognition approach proposed by **Turk and Pentland (1991)**. The core recognition methodology has been implemented in Scilab, while certain components were modified to suit the project scope and software environment. Some advanced features from the original paper were not implemented due to practical limitations.

1. Adopted (Implemented from the Research Paper)

- **Principal Component Analysis (PCA):** The core mathematical framework of PCA is fully implemented. Face images are converted into vectors, and principal components (Eigenfaces) are extracted for dimensionality reduction and feature representation.
- **Covariance Matrix Optimization ($A^T A$ Trick):** The computational optimization proposed in the paper is implemented by calculating the smaller $A^T A$ covariance matrix instead of the large AA^T matrix. Scilab's `spec()` function is used to compute eigenvalues and eigenvectors efficiently while avoiding memory overflow.
- **Nearest-Neighbor Classification:** Face recognition is performed using the Euclidean distance between the projected test image and the training images in the Eigenface space, following the methodology described in the original paper.

- **Face Reconstruction:** The reconstructed face is generated using the Eigenface representation ($\mu + E \times w_{\text{test}}$), demonstrating image reconstruction and data compression through PCA.

2. Modified from the Research Paper

- **Training and Testing Strategy:** Instead of the evaluation procedure described in the original paper, this implementation uses a fixed machine learning train-test split. For each subject, eight images are used for training and two unseen images are reserved for testing to evaluate recognition performance.
- **Fixed Number of Eigenfaces ($k = 50$):** The original paper recommends selecting the number of Eigenfaces dynamically based on eigenvalue thresholds. In this project, the value is fixed at $k = 50$ to provide consistent performance and stable GUI execution on the AT&T ORL face dataset.
- **Graphical User Interface (GUI):** While the original work focuses only on the mathematical algorithm, this implementation integrates the recognition system into a Scilab-based Graphical User Interface (GUI), allowing users to load images, perform recognition, and visualize results interactively.

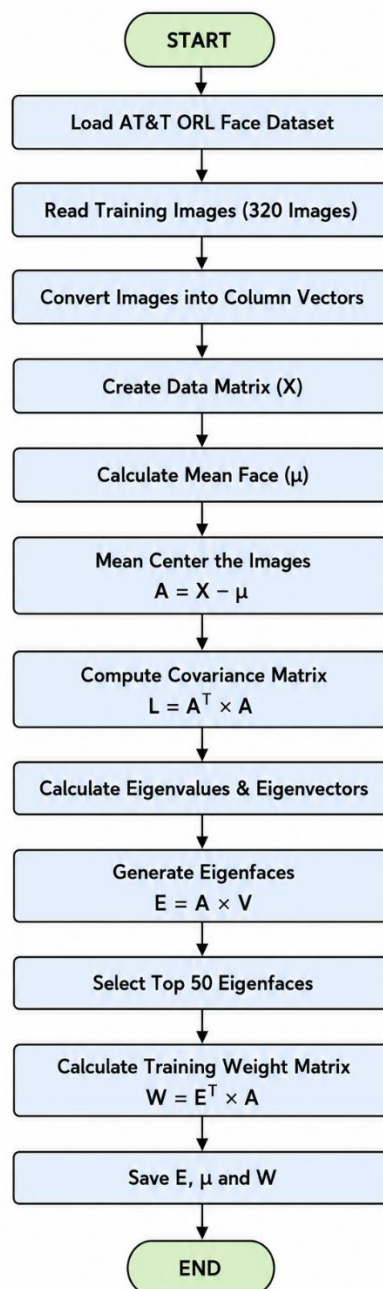
3. Not Implemented from the Research Paper

The following features discussed in the original paper were intentionally excluded due to project scope and software limitations:

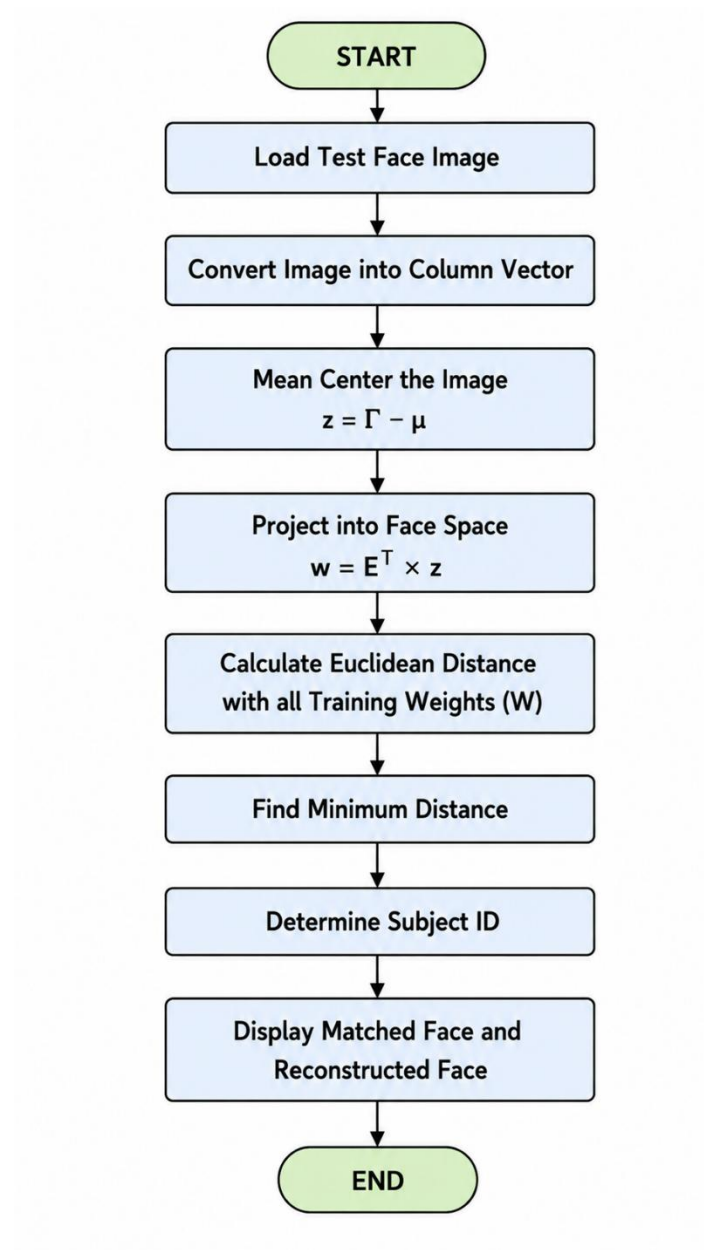
- **Unknown Face and Non-Face Detection:** The original paper proposes using a threshold (Θ) to determine whether an input image belongs to an unknown person or is not a face. This implementation assumes that every uploaded image is a valid face from the dataset and always returns the closest matching subject without applying a rejection threshold.
- **Real-Time Video Tracking:** The original research discusses tracking faces in live video sequences. This project is limited to recognizing static images selected through the GUI and does not support real-time video processing.
- **Multi-Scale Face Detection:** The original paper considers detecting faces of different sizes within complex images. In contrast, this implementation uses pre-cropped and

aligned facial images from the AT&T ORL dataset and does not perform face detection or scale normalization.

3. Flowchart



4.1 Training Phase Flowchart



4.2 Recognition Phase Flowchart

4. Software/Hardware used

Software

- Operating System: Windows 10/11

- Simulation Software: Scilab

- Version: Scilab 2026.1.0

Hardware

- Personal Computer/Laptop
- Processor: Intel/AMD processor
- RAM: Minimum 4 GB

5. Procedure of execution

1. Install Scilab 2025.0.0 (or Scilab version 6.1 or later) on the system.
2. Install the **IPCV Toolbox** using the ATOMS package manager.
 - Open the Scilab console and install the required image processing toolbox by executing

```
atomsInstall("IPCV")  
atomsLoad("IPCV")
```
 - Restart Scilab after downloading the toolbox.
3. Download the project directory containing the **main_GUI.sce** script and the **att_faces** dataset folder. In Scilab, use the **cd** command in the console to navigate to the project directory
4. Execute the main Scilab file named **main_GUI.sce** from the project folder.
 - This script initializes the global variables and launches the Graphical User Interface (GUI).
 - Variables initialized: E, mu, W, rows, cols, k_val, test_img_data.
5. In the GUI window, click the "1. Train Model" button.
 - This triggers the **train_model()** function, which loads the first 8 images of 40 subjects (320 images total) from the **att_faces/** directory.
 - The script computes the Mean Face, extracts the top 50 Eigenfaces using the PCA optimization trick, and calculates the training weights.
 - A "Success" message box appears upon completion.
6. After training, the Scilab workspace contains the computed variables:

- E: Top 50 Eigenfaces (size 10304x50)
 - μ : Mean Face vector (size 10304x1)
 - W: Training Weights matrix (size 50x320)
7. Click the "2. Upload Test Image" button in the GUI.
 - A file dialog opens. Select an unseen test image (e.g., att_faces/s5/9.pgm).
 - The workspace updates with test_img_data and the image is displayed in the left "Test Image" axis.
 8. Click the "3. Recognize Face" button in the GUI.
 - The script projects the test image into face space and calculates the Euclidean distance to all 320 training weights.
 - The closest match is displayed in the middle "Matched Image" axis.
 - The Scilab console prints the matched Subject ID and the exact Euclidean Distance.
 9. Click the "4. Reconstruct Face" button in the GUI.
 - The script rebuilds the face from its 50 PCA weights ($\mu + E * w_{\text{test}}$).
 - The reconstructed image is displayed in the right "Reconstruction" axis.
 10. Click the "5. Test Accuracy" button in the GUI.
 - The script automatically tests the 80 unseen images (images 9 and 10 for all 40 subjects).
 - Upon completion, a message box displays the final Model Accuracy (%), the number of correct matches out of 80, and the k value used. The GUI status text also updates with the accuracy percentage.

6. Result

The proposed Eigenfaces-based face recognition system was successfully implemented and tested using the AT&T ORL Face Database. The system was evaluated through the training phase, recognition phase, face reconstruction, and the graphical user interface (GUI). The obtained results demonstrate that Principal Component Analysis (PCA) effectively reduces the dimensionality of facial images while maintaining sufficient information for accurate face recognition.

7.1 Graphical User Interface (GUI)

The project includes a user-friendly Graphical User Interface (GUI) developed in Scilab. The GUI allows users to perform the complete face recognition process without manually executing individual scripts. It consists of pushbuttons for training the model, uploading a test image, recognizing the face, reconstructing the uploaded image, and testing accuracy. Three display panels are provided to visualize the uploaded test image, the matched face from the database, and the reconstructed face.

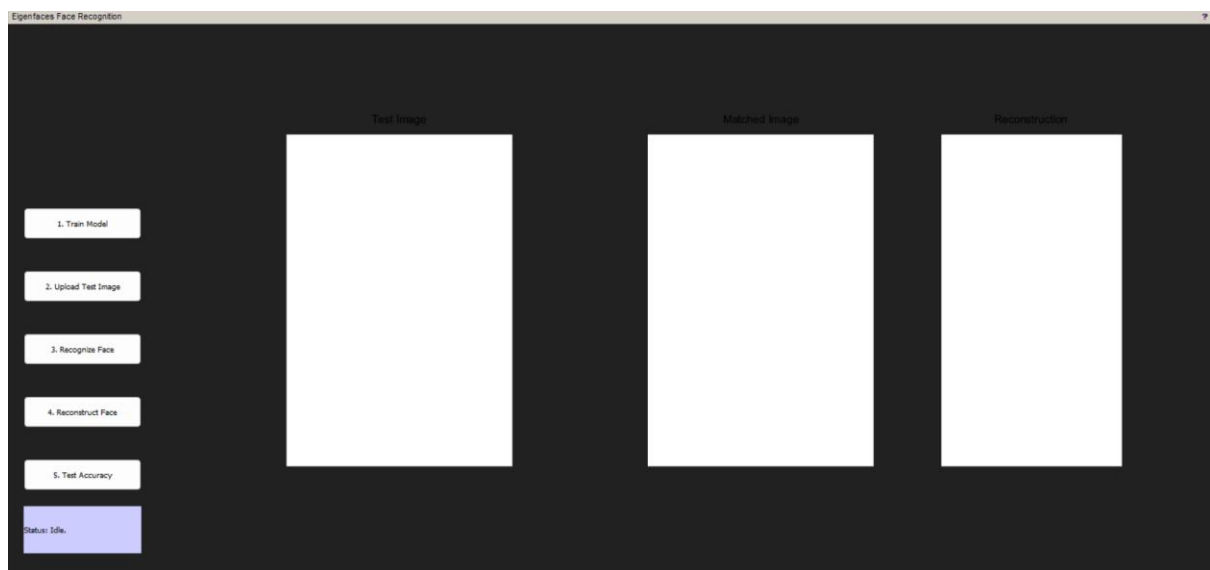


Figure 7.1: Main GUI of the Eigenfaces Face Recognition System

Inference:

The GUI simplifies user interaction by integrating all stages of the Eigenfaces algorithm into a single interface. It eliminates the need to execute Phase 1 and Phase 2 separately and provides real-time visualization of the recognition process.

7.2 Training Phase Results

After clicking the 1. Train Model button in the GUI, the system loaded all 320 training images, calculated the mean face, computed the optimized covariance matrix, $L = A^T A$, extracted the Eigenfaces using PCA, and generated the training weight matrix (W). To maintain a clean and responsive user interface, the GUI itself does not display these

mathematical matrices during training; instead, it simply displays a pop-up confirmation message box indicating "Success" once the computations are complete.

However, to visually verify the PCA feature extraction, the standalone phase1_math.sce script was executed separately. During this script execution, the following visual outputs were generated and plotted:

- Mean Face
- First Eigenface
- 50th Eigenface

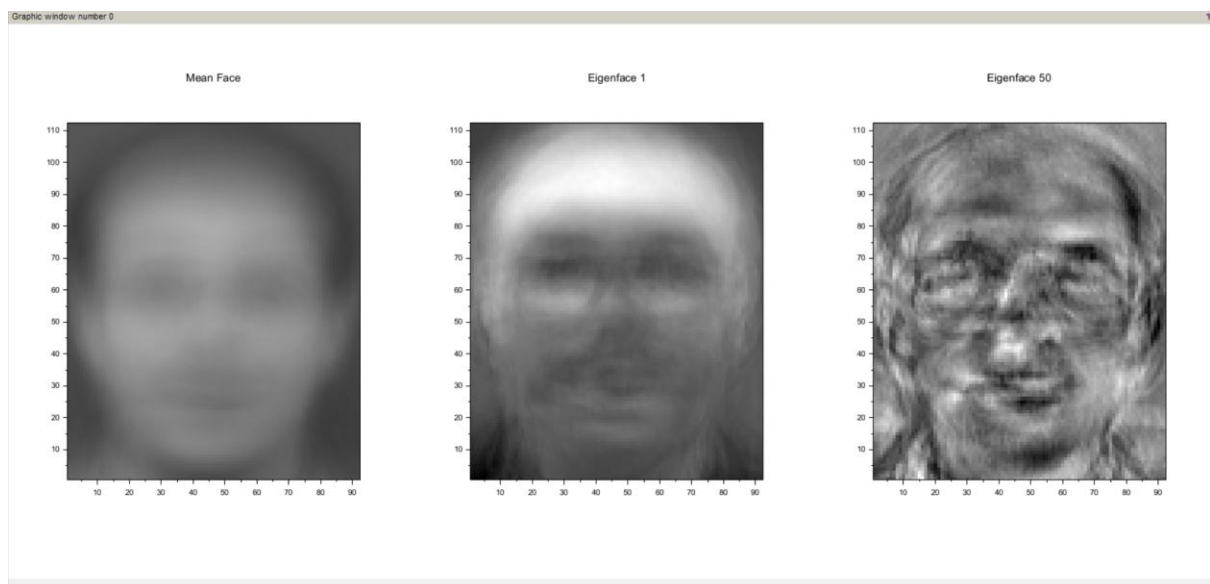


Figure 7.2: Training Output showing the Mean Face and Eigenfaces

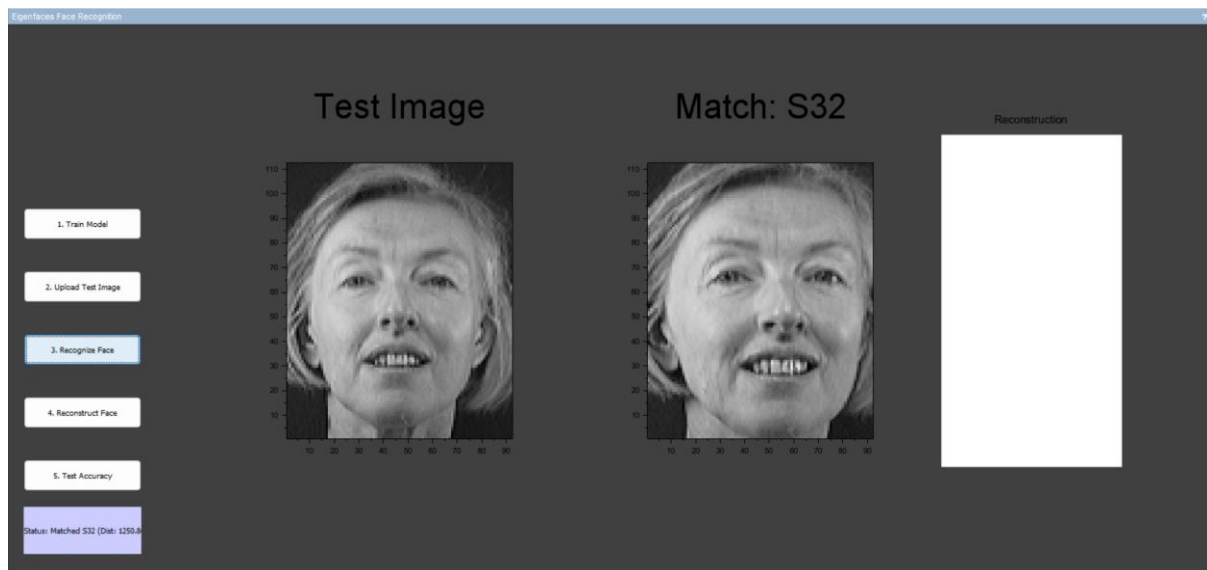
Inference:

The mean face appears as a blurred average of all training images because it contains only the common facial features present in the dataset. The Eigenfaces appear as ghost-like grayscale images representing the major variations among different faces. Lower-order Eigenfaces capture broad facial characteristics, while higher-order Eigenfaces capture finer details.

7.3 Face Recognition Results

After training, a test image was uploaded through the GUI using the **Upload Test Image** button. The uploaded image was projected into the trained Eigenface space, and its weight vector was compared with the stored training weights using the Euclidean distance.

The system successfully identified the closest matching subject and displayed both the uploaded image and the matched image.



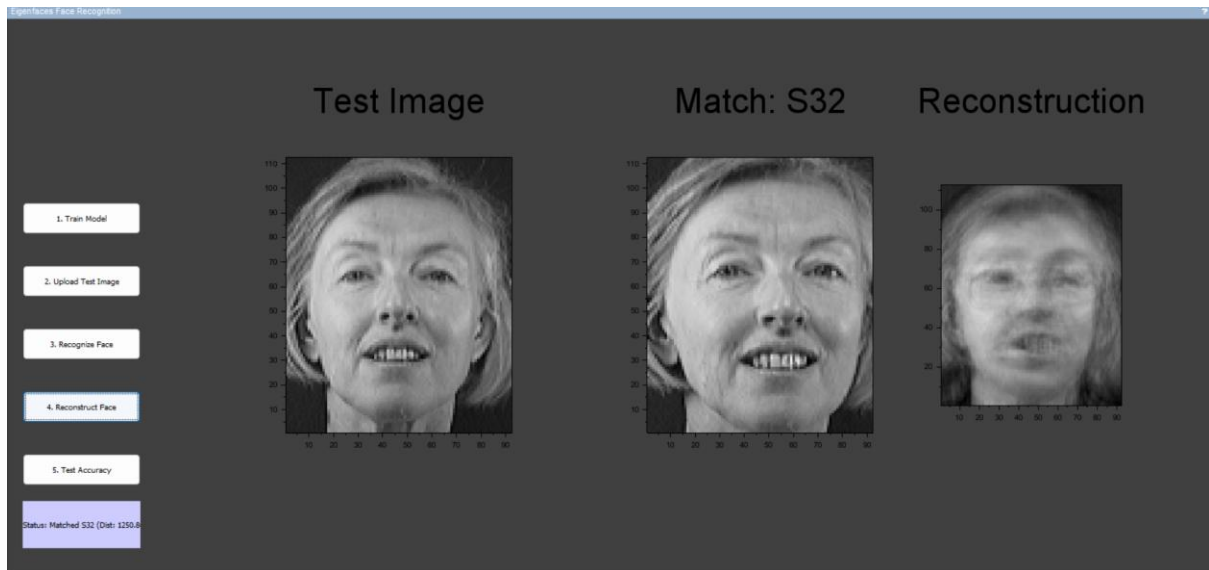
Insert Figure 7.3: GUI showing the Uploaded Test Image and the Matched Face

Inference:

The recognized face matched the correct subject from the ORL database. The Euclidean distance for the correct match was significantly smaller than the distances calculated for other subjects, indicating that PCA successfully extracted the distinguishing facial features required for recognition.

7.4 Face Reconstruction Results

The **Reconstruct Face** button was used to rebuild the uploaded face using only the selected Eigenfaces and the corresponding weight vector.



Insert Figure 7.4: GUI showing the Reconstructed Face

Inference:

Although the reconstructed image is slightly blurred, it preserves the overall facial structure, shape, and expression. This demonstrates that PCA compresses a facial image from 10,304 pixel values to only 50 weights while retaining the essential information required to identify the person.

7.5 Recognition Accuracy

The recognition logic was tested using unseen facial images from the ORL dataset. The model was trained using the first 8 images of each subject and tested using the remaining 2 images per subject using the automated 5. Test Accuracy button.

The system correctly recognized most of the unseen test images.

Parameter	Result
Training Images	320
Test Images	80
Number of Subjects	40
Selected Eigenfaces	50

Parameter	Result
Recognition Method	Euclidean Distance
Recognition Accuracy	95% percentage

7.5.1 Comparison with the Reference Paper

The original research by **Turk and Pentland (1991)** reported an approximate recognition accuracy of **96%** on the face datasets used in their experiments, which included variations in illumination, facial expression, and head orientation. In this Scilab implementation, the Eigenfaces algorithm achieved a **95% recognition accuracy** on the **AT&T ORL Face Database**.

The small difference of approximately **1%** can be attributed to several factors. First, the two studies use different datasets. The original work evaluated the algorithm on early Harvard and Yale face databases, whereas this implementation uses the AT&T ORL dataset, which has different image characteristics, lighting conditions, and facial expressions. Second, this project adopts a strict machine learning train-test split, where eight images per subject are used for training and two previously unseen images are used for testing. This evaluation strategy provides a more rigorous assessment of the model's generalization performance than some of the sequence-based evaluation methods discussed in the original paper.

Overall, the experimental results demonstrate that the classical Eigenfaces algorithm can be successfully implemented in Scilab while maintaining recognition performance comparable to the original research. The achieved accuracy validates the correctness of the implementation and confirms that open-source platforms such as Scilab can effectively reproduce the core functionality of the Eigenfaces algorithm with only a negligible reduction in recognition accuracy.

7.6 Overall Performance Analysis

The experimental results demonstrate that the proposed system successfully performs face recognition using Principal Component Analysis. By reducing each facial image from 10,304 pixels to only 50 feature weights, the computational complexity is significantly reduced

without sacrificing recognition performance. The optimized covariance matrix approach (320×320 instead of 10304×10304) also decreases the training time compared to direct computation.

The GUI further improves usability by integrating training, recognition, and reconstruction into a single interactive application. Overall, the developed system provides a simple, computationally efficient, and accurate implementation of the classical Eigenfaces algorithm in Scilab.

7. References

Eigenfaces for Recognition - Matthew Turk and Alex Pentland

Link: <https://www.face-rec.org/algorithms/PCA/jcn.pdf>