Eigenfaces Based on Recognition System

A Novel approach for Facial Identification

GRATITUDE

SINCERE THANKS TO OUR PROJECT GUIDE

DR. DEBASMITA MUKHERJEE FOR HER

INVALUABLE GUIDANCE AND SUPPORT

THROUGHOUT THE COURSE OF THIS PROJECT.

PROJECT PARTICIPANTS

SAP ID	ROLL NO.	NAME
86062500023	A031	CAROL LOPES
86062500038	A018	AMAN GUPTA
86062500029	A007	VEDANT CHAUGULE
86062500020	A012	SHRUSHTI DAVE
86062500064	A041	SHRIPAD PATIL

TABLE OF CONTENTS

ABSTRACT

- Based on Turk and Pentland's Eigenfaces approach (1991).
- Uses Principal Component Analysis (PCA) for dimensionality reduction.
- Represents faces using key components (eigenfaces) capturing major variations.
- Recognizes new faces by projecting them into a reduced "face space."
- Demonstrates a non–deep learning approach to classical face recognition.

AIM

- Implement and analyze the Eigenfaces approach for facial recognition using PCA.

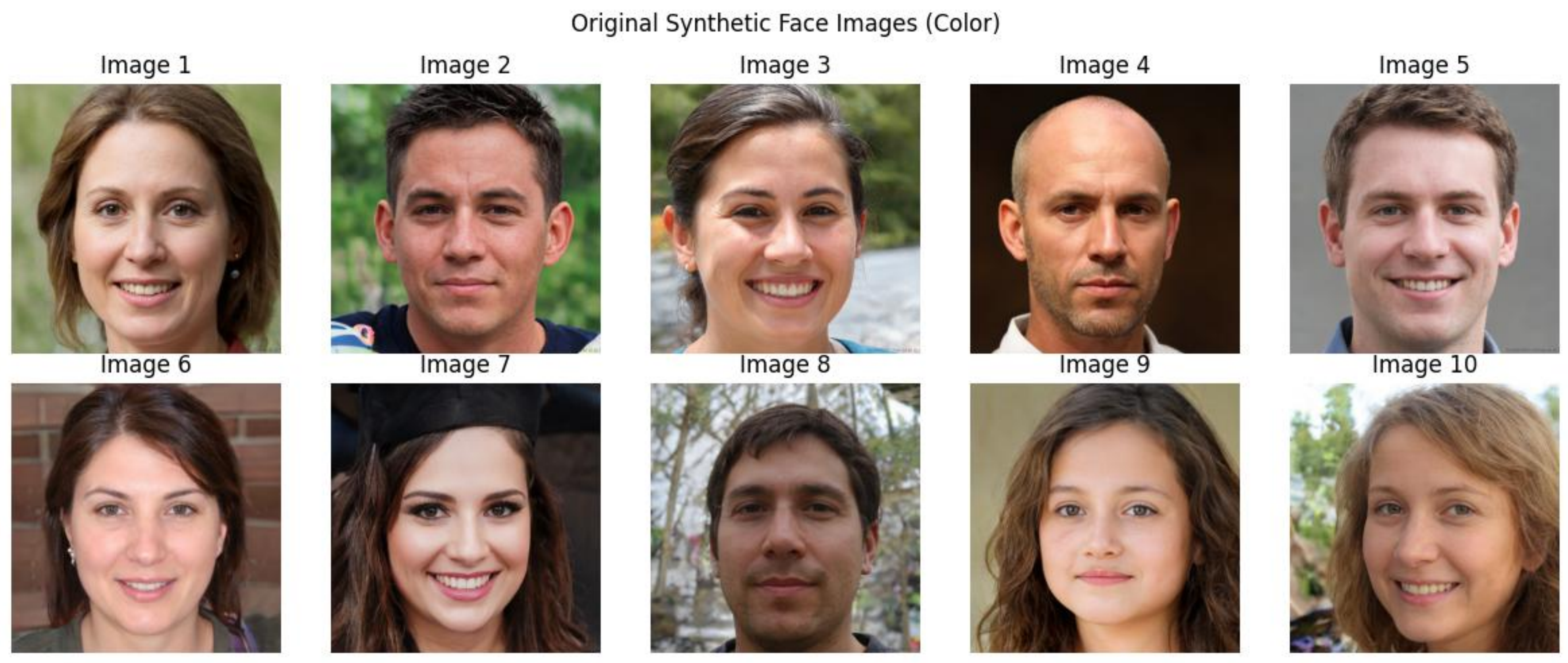
OBJECTIVES

- Understand PCA theory and Eigenfaces algorithm.
- Preprocess and normalize images.
- Compute mean face and eigenfaces.
- Project faces into lower-dimensional space.
- Implement recognition using Python + KNN.
- Discuss limitations and improvements.

DATA DESCRIPTION

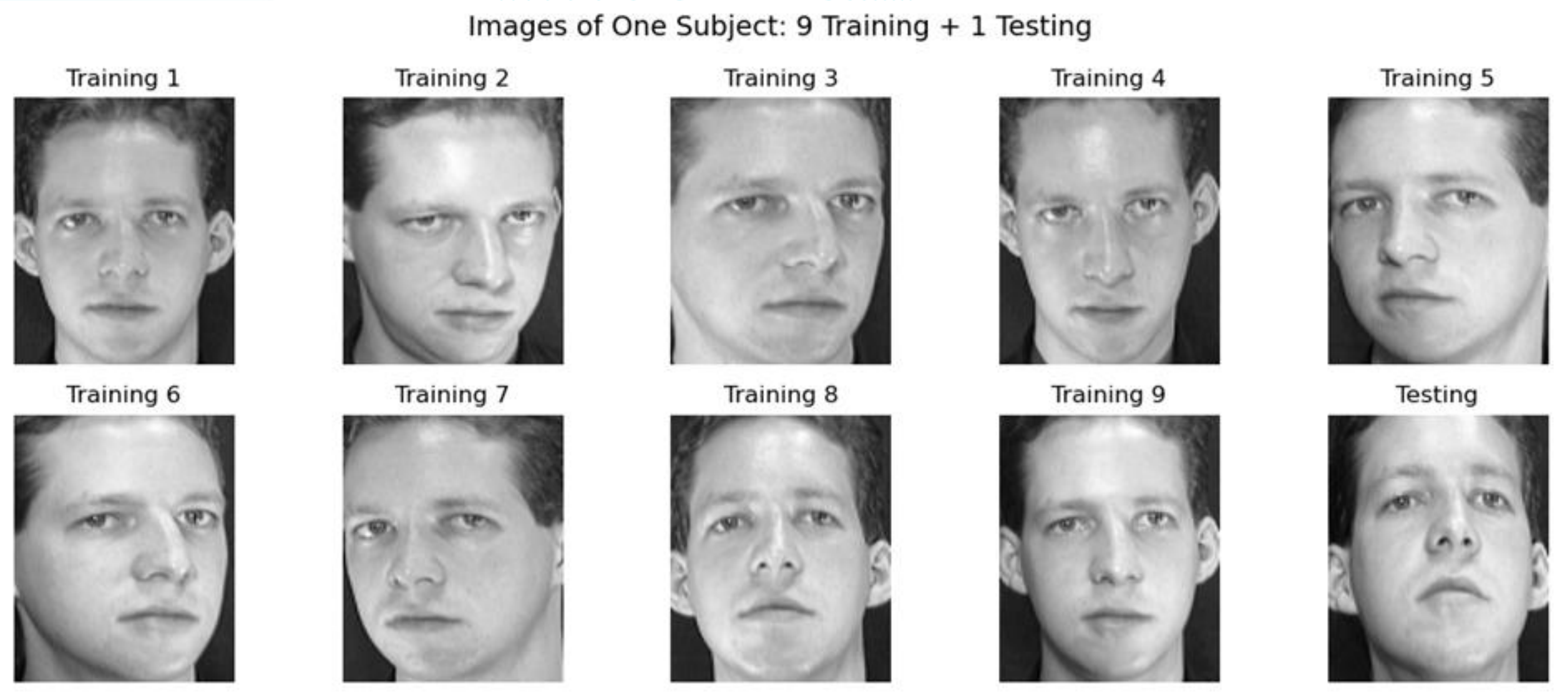
Synthetic Face Dataset (AI-Generated)

- 10 synthetically generated face images used for mathematical demonstration and visualization. Images were resized to 100 x 100 pixels and converted to grayscale.



AT&T (ORL) Database

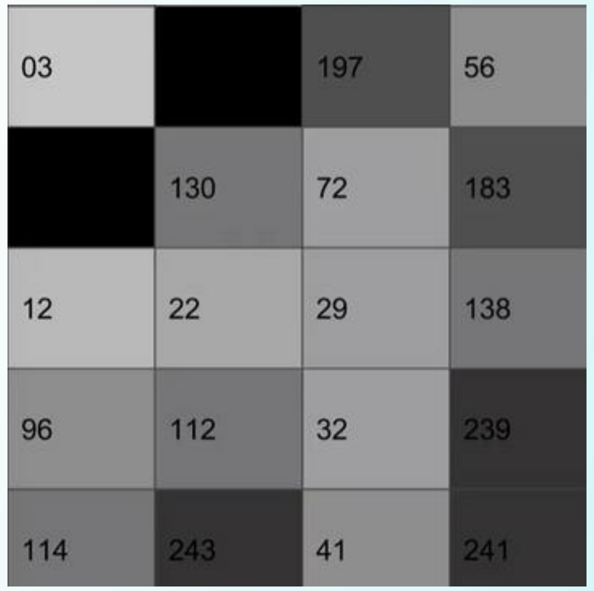
- 20 subjects × 10 images each
- Resolution: 92×112 pixels
- Grayscale, standardized format
- 9 images for training, 1 for testing



10 Images of one of the person where 9 images are used for training and 1 is used for testing

DATA PROCESSING

Synthetically generated image → **CONVERT TO GRAYSCALE RESIZE IMAGE TO CONSISTENT SIZE** → **EVERY PIXEL GETS A NUMBER** → **CONVERT INTO 1D VECTOR** → **STACK INTO A DATA MATRIX**



03
250
197
56
251
130
72
183
12
22

03	69	125	36	159
250	36	95	53	222
56	45	85	6	245
251	02	4	254	103
130	35	5	63	78
72	45	23	20	20

METHODOLOGY

- **Methodology is divided into two main parts**

1. Mathematical implementation - to understand how the Eigenfaces and PCA actually work.

2. System Implementation – to apply the same process on a real dataset for facial recognition.

Together, these stages helped us connect the theory of PCA with its practical use in face recognition.

WHAT IS PCA ?

PCA (Principal Component Analysis)is used to reduce the dimensionality of the face image data while retaining the components that account for the most variance.
It helps remove redundancy and highlight the most important pattern in the dataset

WHAT IS EIGENFACES ?

Eigenfaces are the Principal Component derived from a set of face images using **PCA**
They represent the key feature or variation among different faces-such as shape, lighting, or expression.
In short, Eigenfaces are like building blocks or characteristic patterns of human faces

1. Mathematical Implementation – Data Representation

- Every image was converted into grayscale and resized to a fixed dimension
- Image Vectors: A $N×N$ face image is represented as a vector Γ of dimension $N2$.
- Training Set: The initial database consists of M sample images, $\{\Gamma 1,\Gamma 2,\ldots,\Gamma M\}$.
- Compute the mean face (Ψ):

$$\Psi = \frac{1}{M}\sum_{n=1}^{M} \Gamma_n$$

- Each face image is then mean-adjusted to form the difference vector Φi: $\Phi_i = \Gamma_i - \Psi$

Eigenface Calculation

- Covariance Matrix: The goal is to find the eigenvectors of the $N2 \times N2$ covariance matrix C:

$$C = \frac{1}{M}\sum_{n=1}^{M} \Phi_n \Phi_n^T = AA^T$$

where $A = [\Phi_1, \Phi_2, \ldots, \Phi_M]$.

- To simplify, compute eigenvectors of smaller matrix:

$$L = A^T A$$

$$L v_l = \mu_l v_l$$

- Eigenface Construction: The first M non-zero eigenvectors of C (the Eigenfaces, $u1$) can be constructed from the eigenvectors of L:

$$u_l = \sum_{k=1}^{M} v_{lk} \Phi_k, \quad l = 1, \ldots, M$$

Face Space Representation

- Only top M′ eigenfaces (largest eigenvalues) are kept.
- Any face image Γ can be projected onto the Face Space, resulting in a low-dimensional pattern vector Ω (also called the weight vector):

$$\Omega^T = [\omega_1, \omega_2, \ldots, \omega_{M'}]$$

where each weight ωk is the projection onto the k-th Eigenface:

$$\omega_k = u_k^T(\Gamma - \Psi)$$

- Face Class Characterization

For each known person k, multiple training images are projected to yield multiple pattern vectors.

Class Vector: The final face class vector Ωk for individual k is the average of the pattern vectors derived from their training images

Recognition & Classification

Reconstruct face using:

$$\Gamma_f = \sum_{k=1}^{M'} \omega_k u_k + \Psi$$

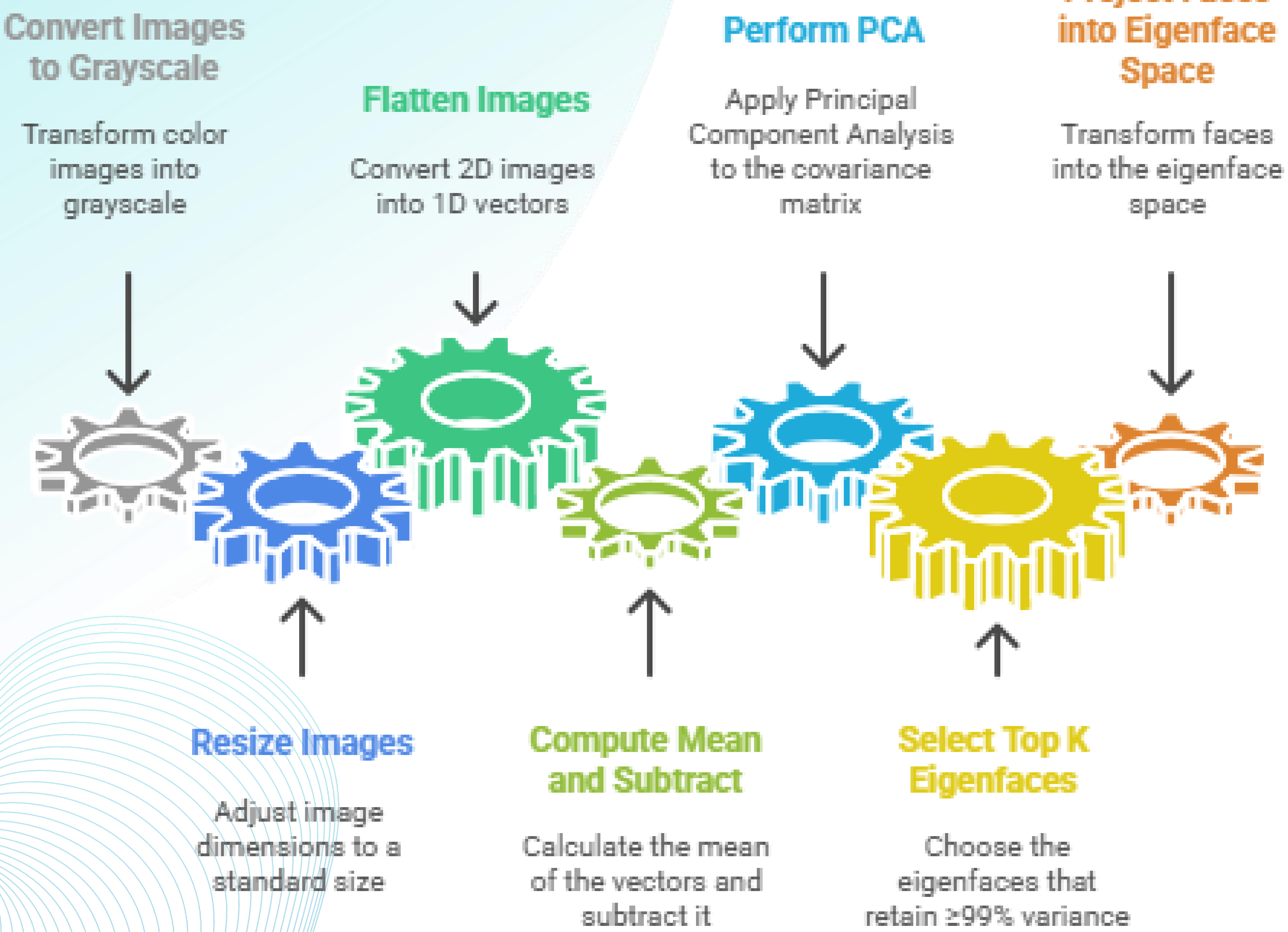
Decision rules:

If ε > threshold → not a face.

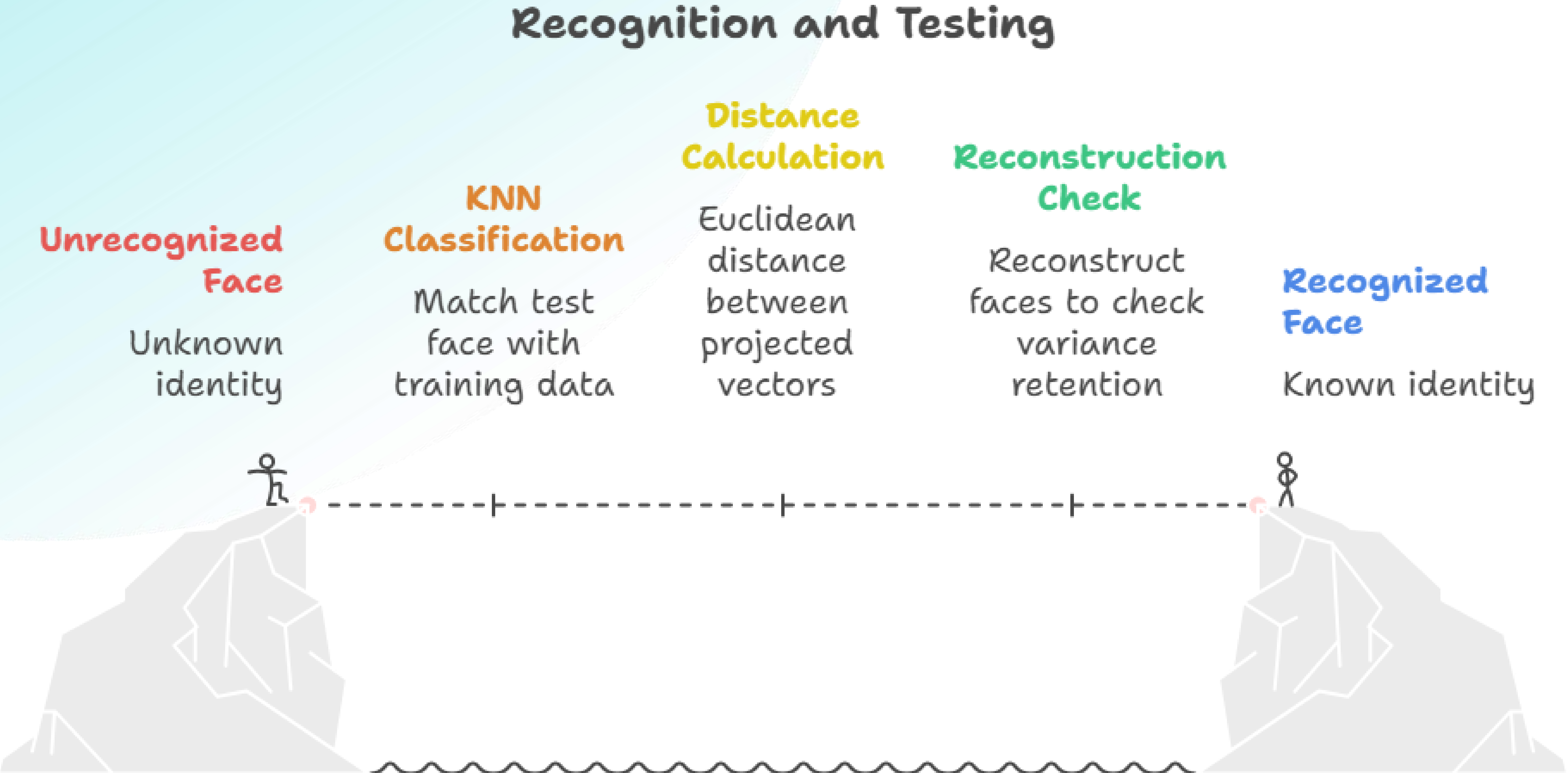
If ε < threshold → compute Euclidean distance between Ω and Ω_k.

Smallest distance → recognized face.

2. System Implementation



Recognition and Testing



Summary of Methodology

- PCA reduces dimensionality while preserving facial features.
- Eigenfaces form basis vectors representing variations.
- Recognition achieved via projection + nearest neighbor in face space.
- Achieved ~98–100% precision for 20 subjects with 35 eigenfaces.

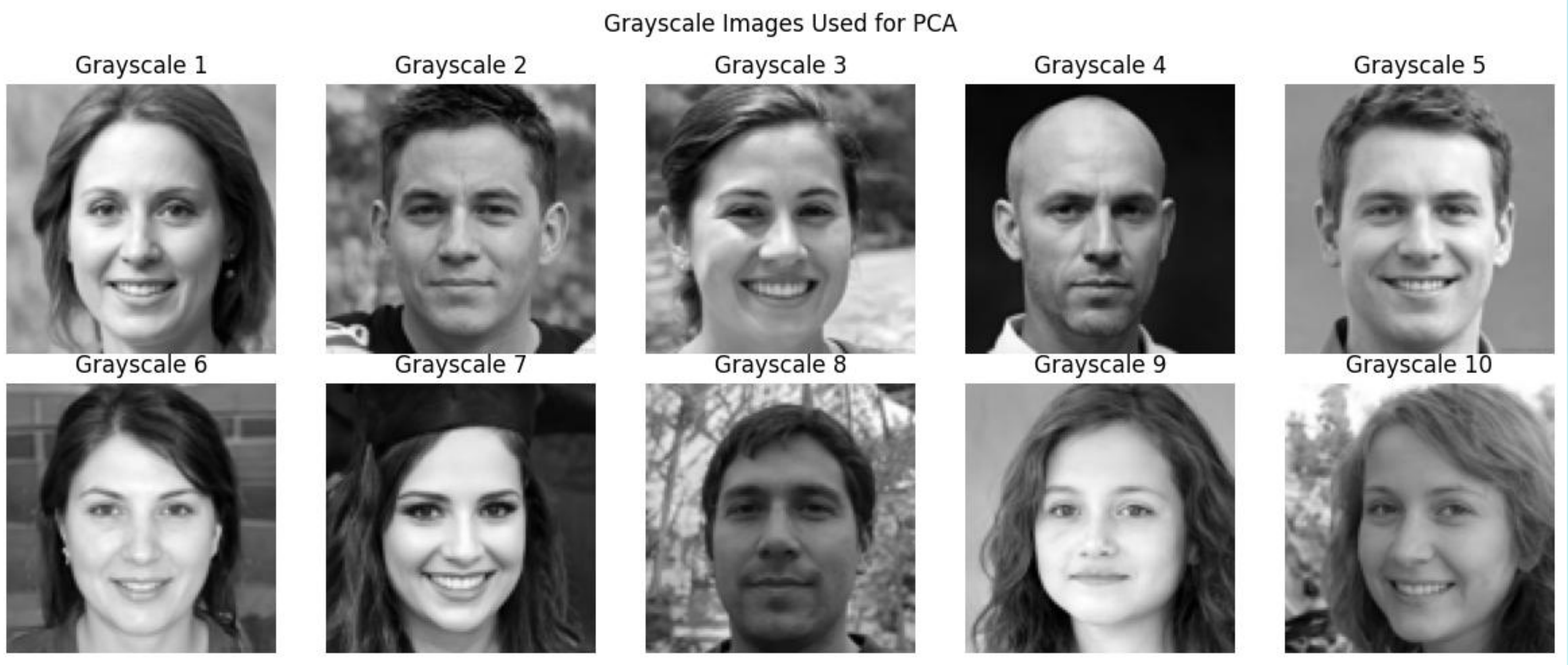
RESULT AND ANALYSIS

1. MATHEMATICAL IMPLEMENTATION

Step 1 : Image Matrix (X)

Process:

- Each Image resized to 100x100 pixels and flattened to 10000 elements.
- Data matrix X formed with 10 images of size (10 x 10,000).




```
X (image matrix)
 Shape: (10, 10000)  dtype: float64
 Stats: min=0.000000, max=255.000000, mean=132.275350, std=60.682283
 Row/col sample (first 3 rows):
[[ 95.  96.  96.  96.  97.  97. 100. 103. 105. 106. 107. 108. 107. 108.
  109. 109. 110. 111. 111. 110.]
 [ 74.  74.  74.  74.  74.  74.  74.  74.  73.  73.  73.  73.  73.  73.
   73.  73.  73.  73.  73.  72.]
 [144. 145. 144. 144. 144. 144. 144. 145. 143. 142. 141. 142. 142. 142.
  142. 142. 141. 140. 141. 141.]]
```

Step 2 : Mean Face Computation

Process:

- Compute average intensity across all faces ⟶ mean Face

$$\mu = \frac{1}{M}\sum_{n=1}^{M} \Gamma_n$$

Interpretation:

- Represents the average appearance of all faces.
- Serves as the baseline from which individual variations are measured.

```
mean_face (vector)
 Shape: (10000,)  dtype: float64
 Stats: min=69.400000, max=192.700000, mean=132.275350, std=22.342433
 First elements: [156.6 157.5 156.2 156.7 155.9 156.6 158.3 156.9 156.9 154.8 156.  164.
 162.1 158.9 158.9 159.4 152.8 156.5 161.4 161. ]
 Last elements:  [125.5 126.2 131.5 140.4 142.7 141.7 159.4 166.5 162.7 164.4 165.3 172.6
 176.4 173.6 173.6 181.4 190.2 192.2 189.  187.3]
```

Mean Face (μ)

Step 3 : Mean Subtraction

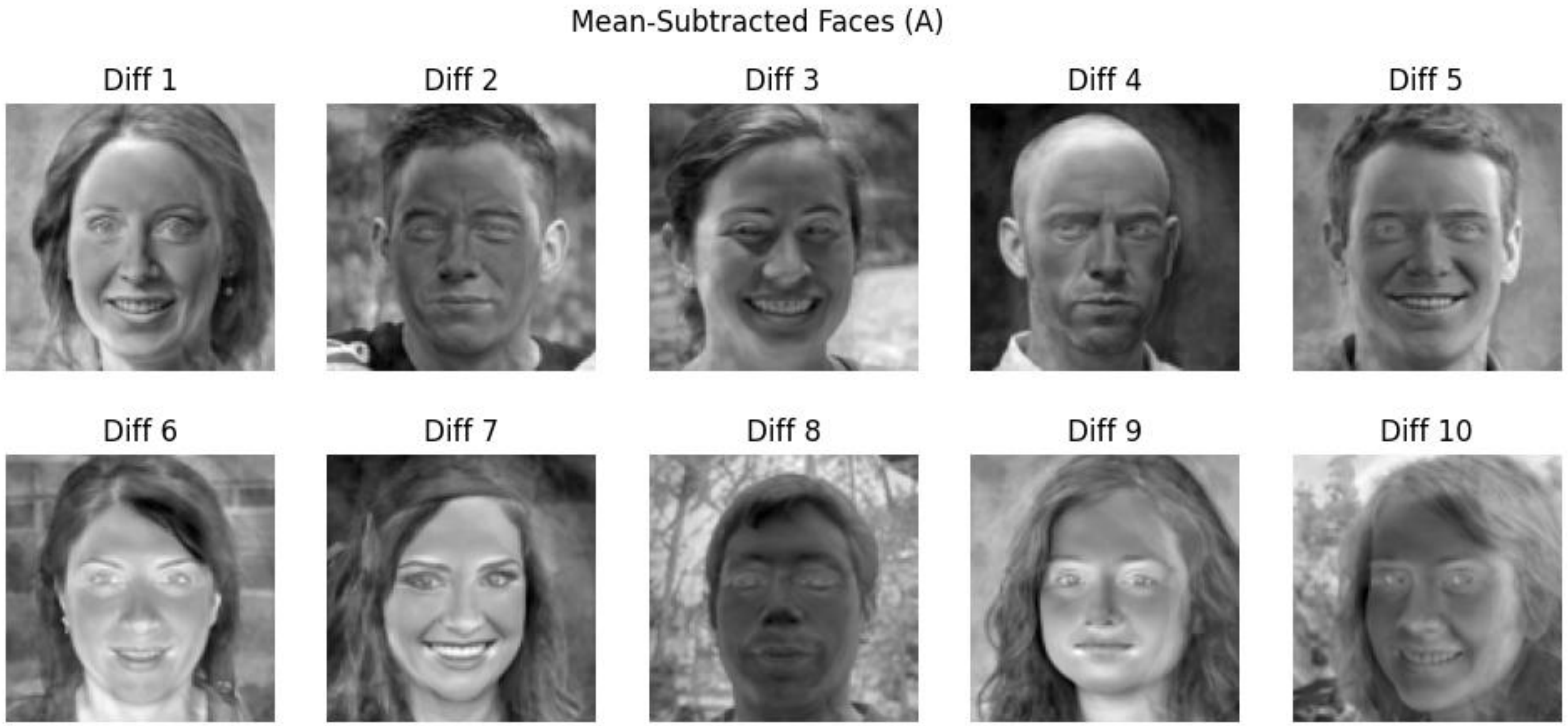
Process:

- Subtracted mean face from each image inorder to obtain a mean - centered faces.

$A_i = \Gamma_i - \mu$

Interpretation:

- Centers data around zero mean.
- Highlights deviation from the average, making facial differences visible.




```
A (X - mean_face)
 Shape: (10, 10000)  dtype: float64
 Stats: min=-169.700000, max=167.200000, mean=-0.000000, std=56.419457
 Row/col sample (first 3 rows):
[[-61.6 -61.5 -60.2 -60.7 -58.9 -59.6 -58.3 -53.9 -51.9 -48.8 -49.  -56.
  -55.1 -50.9 -49.9 -50.4 -42.8 -45.5 -50.4 -51. ]
 [-82.6 -83.5 -82.2 -82.7 -81.9 -82.6 -84.3 -82.9 -83.9 -81.8 -83.  -91.
  -89.1 -85.9 -85.9 -86.4 -79.8 -83.5 -88.4 -89. ]
 [-12.6 -12.5 -12.2 -12.7 -11.9 -12.6 -14.3 -11.9 -13.9 -12.8 -15.  -22.
  -20.1 -16.9 -16.9 -17.4 -11.8 -16.5 -20.4 -20. ]]
```

Step 4 : Covariance Matrix

Process:

- Computing covariance matrix $L = AA^T$, where A is mean-subtracted data.
- Using the trick to reduce the covariance matrix from 10000 x 10000 to 10x10 for efficiency. $L = A^TA$

	0	1	2	3	4	5
0	19831070.07	-878830.63	425172.27	-2280125.13	8113895.87	-2071527.23
1	-878830.63	62133414.67	22186325.57	-25206811.83	-7173687.83	-26807239.93
2	425172.27	22186325.57	31589387.47	-14002847.93	-7616982.93	-17293134.03
3	-2280125.13	-25206811.83	-14002847.93	21549765.67	-987969.33	15376951.57
4	8113895.87	-7173687.83	-7616982.93	-987969.33	40603362.67	-3841472.43
5	-2071527.23	-26807239.93	-17293134.03	15376951.57	-3841472.43	26389320.47

Interpretation:

- Captures how faces vary relative to one another.
- A smaller covariance matrix enables faster eigen decomposition.

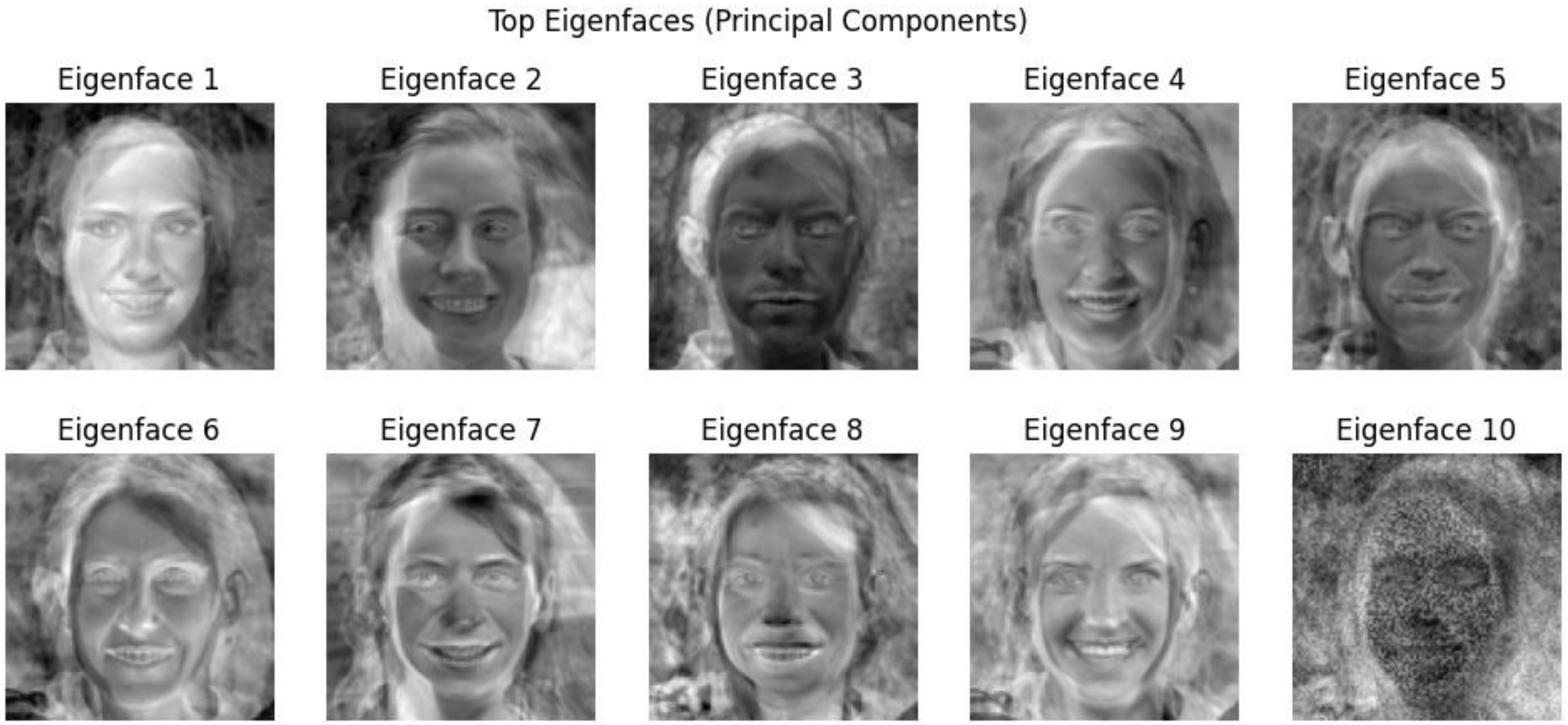
Step 5 : Eigenface Extraction

Process:

- Perform eigen decomposition on covariance matrix to obtain eigenvectors. $Lv_i = \lambda_i v_i$
- Project eigenvectors into image space giving top 10 eigenfaces $u_i = A^T v_i$

Interpretation:

- Each eigenface captures a distinct variation (lighting, eyes, nose, etc.
- They form a basis for representing all faces.




```
eigenfaces (matrix)
 Shape: (10000, 10)  dtype: float64
 Stats: min=-0.053105, max=0.050928, mean=-0.001143, std=0.009934
 Row/col sample (first 3 rows):
[[-0.01446534  0.02242147 -0.00504411  0.00554871  0.00513582 -0.00060653
  -0.00204736 -0.0077305   0.00164527 -0.005654  ]
 [-0.01434982  0.02231571 -0.00440658  0.00501397  0.00561905 -0.00138408
  -0.00129623 -0.00673116  0.00221857 -0.00650636]
 [-0.01460789  0.02188016 -0.00489341  0.00506178  0.00530836 -0.00240467
  -0.00302949 -0.00686637  0.00242175 -0.00288568]]
```

Step 6 : Projection and Recognition

Process:

- Project each face image into Eigenface space to obtain vectors.
- Compare Euclidean distances to identify closest match.

$$\omega_k = u_k^{\top}(\Gamma - \mu)$$

Interpretation:

- The smaller the distance the more similar the face.
- Perfect match when test image = training image (distance = 0.0)

	0	1	2	3	4	5
0	580.597999	7328.846956	3720.343022	-3600.607281	-210.293407	-3983.049440
1	-2740.576147	1236.724990	688.053000	-28.975834	-5637.780355	658.170602
2	-147.438868	934.068139	-3432.077517	714.604836	1101.501488	1774.825188
3	2526.486611	-337.479556	608.125161	1270.322109	-2386.754066	1344.183759
4	830.346300	-1276.069136	1068.106471	-343.188704	-200.866963	1073.893402
5	941.221705	-311.966337	-1753.411015	-744.301585	-1205.572189	-726.547447

Step 7 : Reconstruction

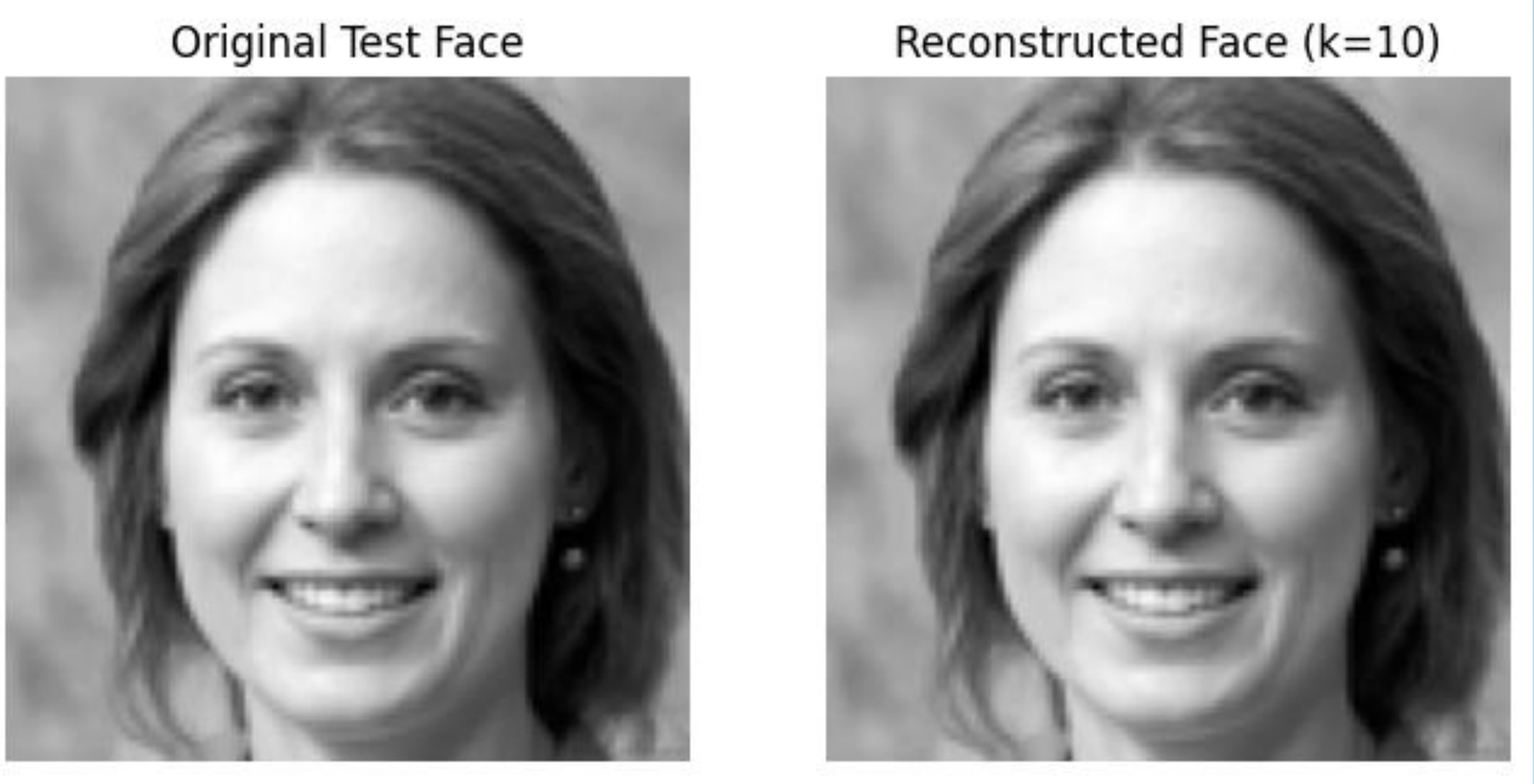
Process:

- Reconstruct original image using top k eigenfaces.

$$\hat{\Gamma} = \mu + \sum \omega_k \mu_k$$

Interpretation:

- Shows that most facial information can be recovered from few components.
- Demonstrates PCA’s data compression and reconstruction capability.



2. SYSTEM IMPLEMENTATION

AT&T (ORL) Database

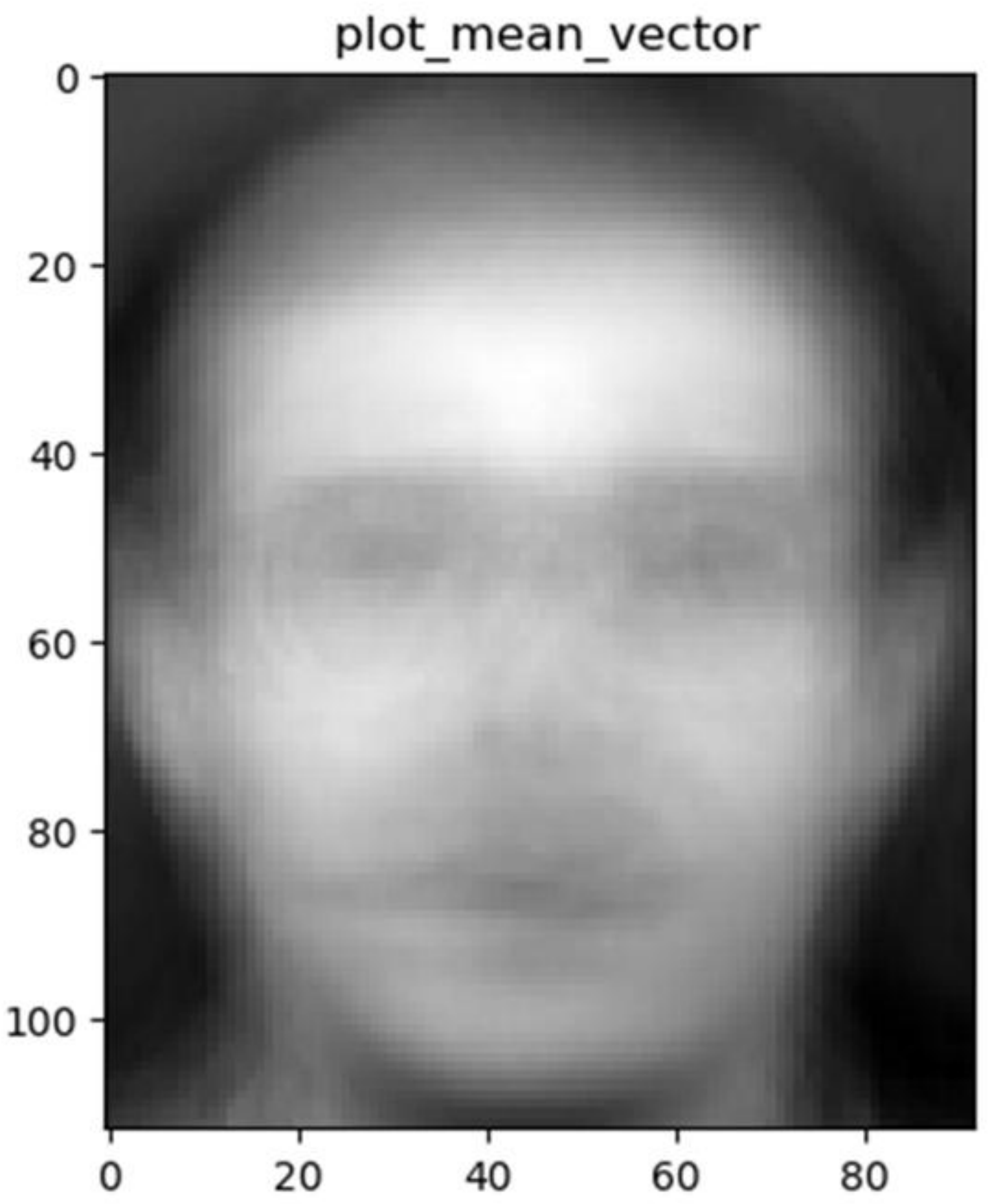
Mean Face (System level)

Process:

- Mean computed across all training images.

Interpretation:

- Represents the average of all subjects.
- Smoother and more generalized than the small demo dataset.



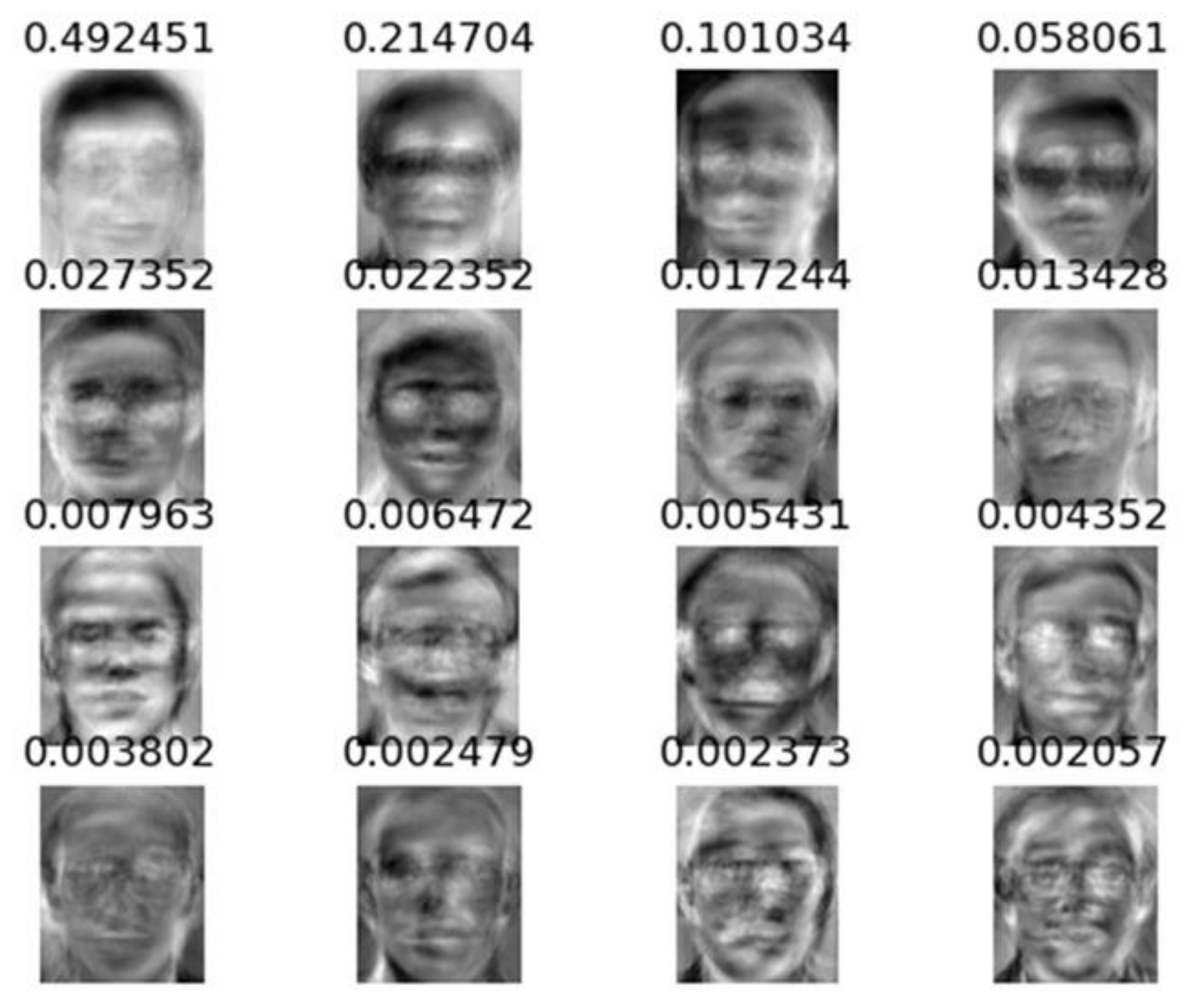
Top Eigenfaces Visualization

Process:

- Computed PCA to obtain eigenfaces.
- Here we are using top 35 eigenfaces, but for visualization we are only using top 16 eigenfaces.

Interpretation:

- Capture key identity variations such as facial structure, lighting, and expression.
- Each eigenface contributes differently to variance representation.



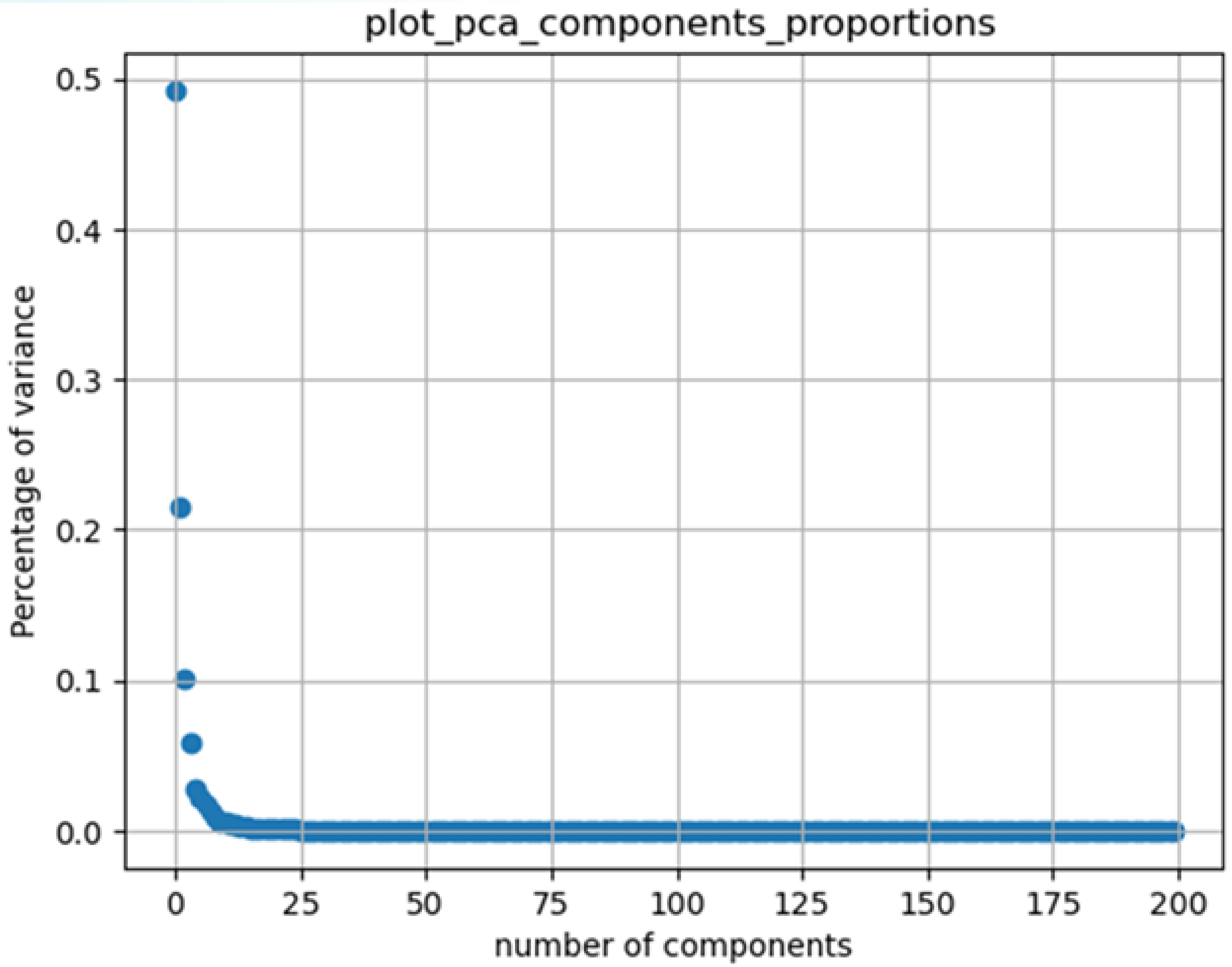
PCA Variance & Cumulative Distribution

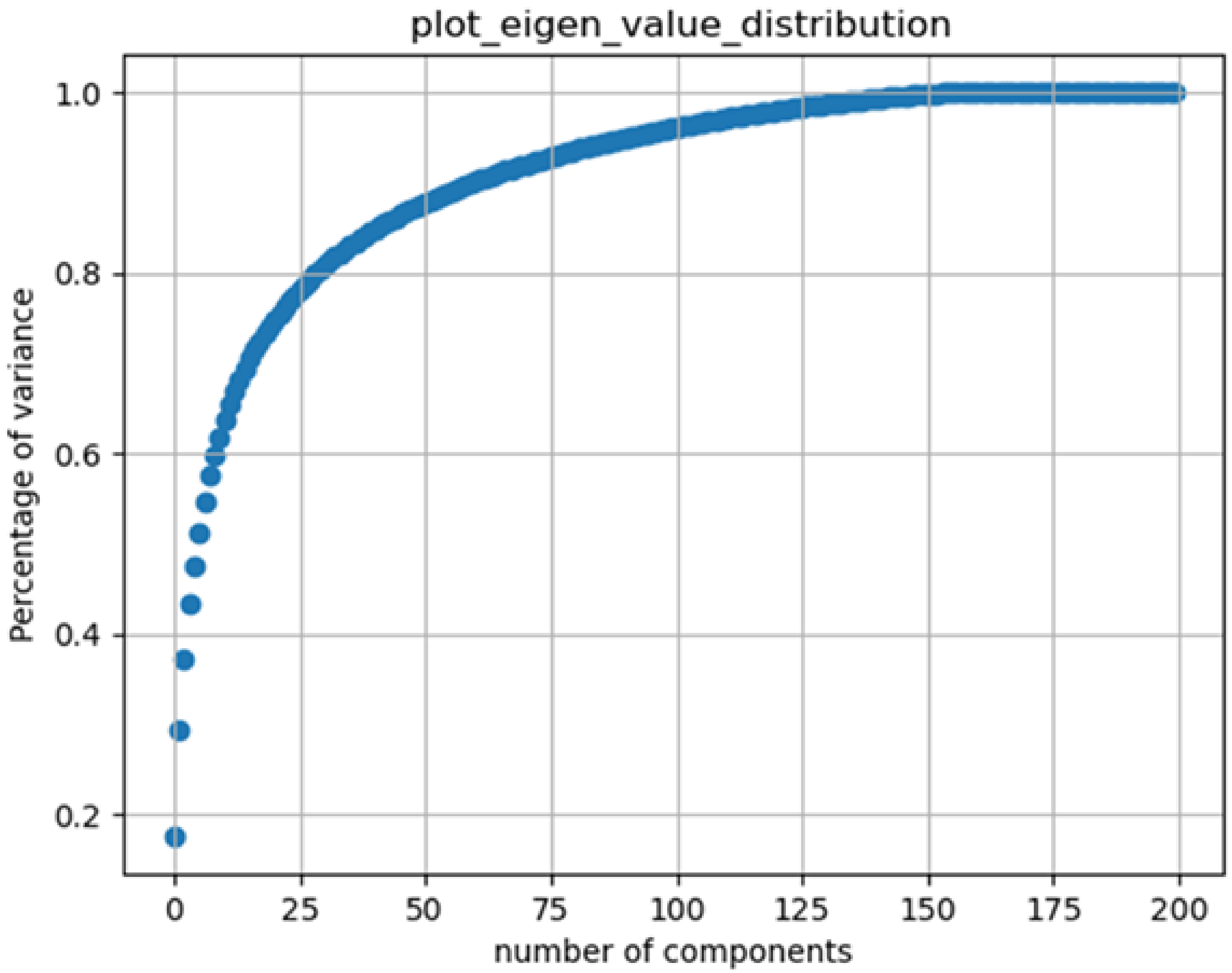
Process:

- Calculate variance explained by each principal component.
- Plot cumulative variance curve.

Interpretation:

- ~99% variance retained with k = 35 eigenfaces.
- Confirms that a few components capture most facial information.





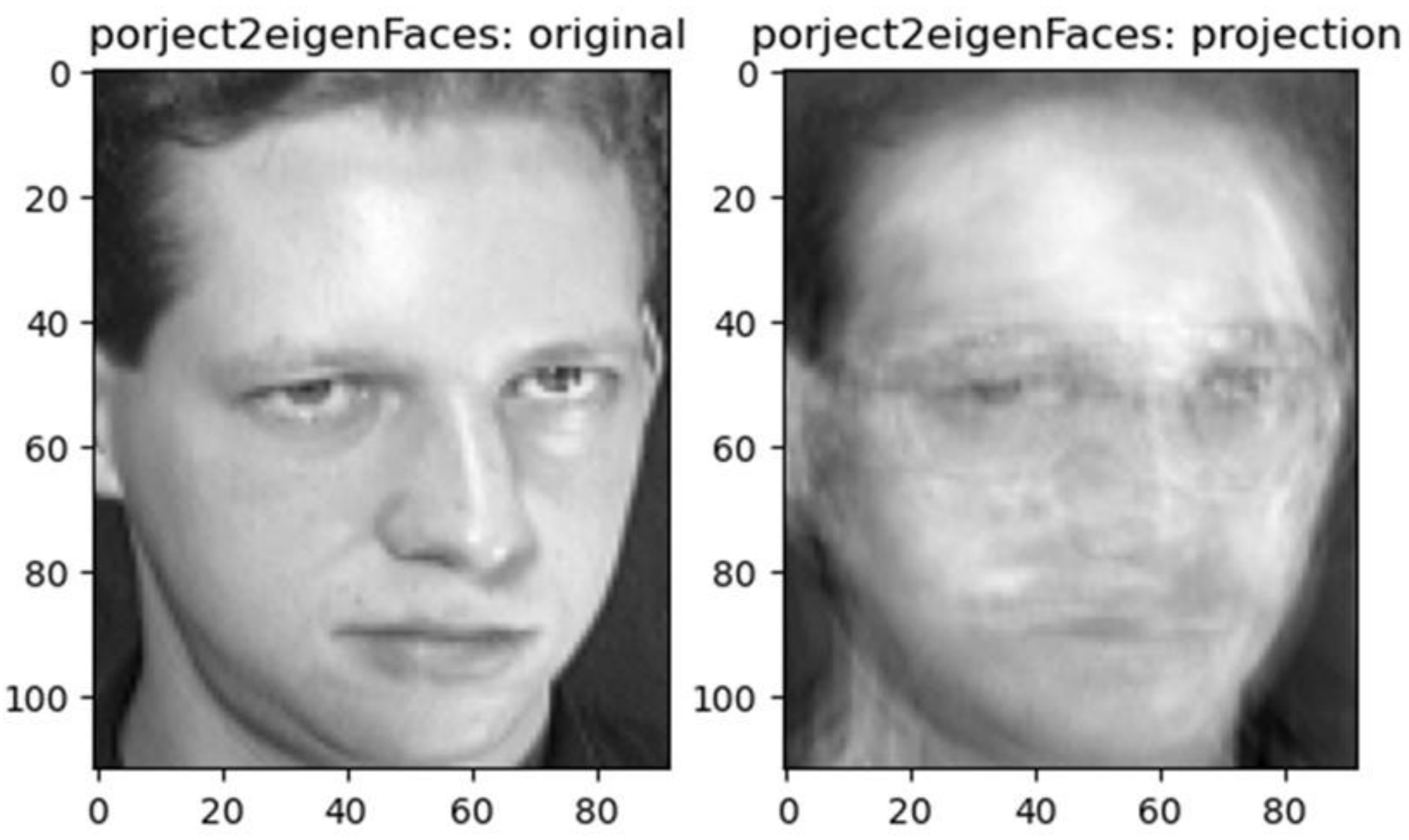
Face Projection & Reconstruction

Process:

- Project test face into Eigenface space.
- Reconstruction it using selected components.

Interpretation:

- Reconstructed image closely matches original giving high information retention.
- Confirms PCA's efficiency in compressing face data.



Precision vs KNN (Recongnition Performance)

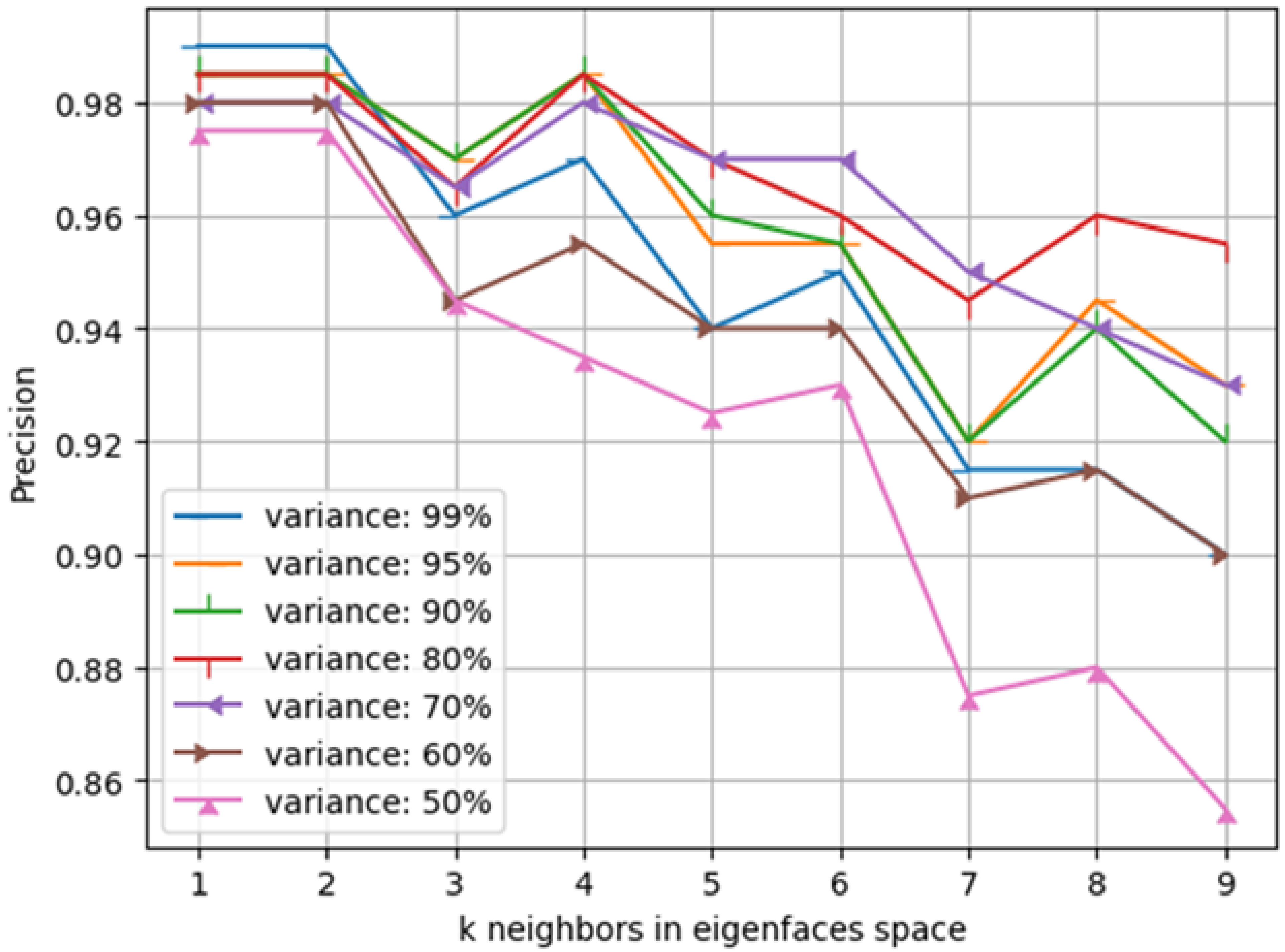
Process:

- Tested different variance retention levels (50-99)% and KNN (1-10).

Interpretation:

- Accuracy remains ~ 100% up to 70% variance.
- Recognition drops sharply below 60%.

Variance	Components (K)	Precision (avg)
99%	35	100%
95%	27	100%
90%	20	100%
80%	12	100%
70%	7	97.50%
50%	3	70-82%



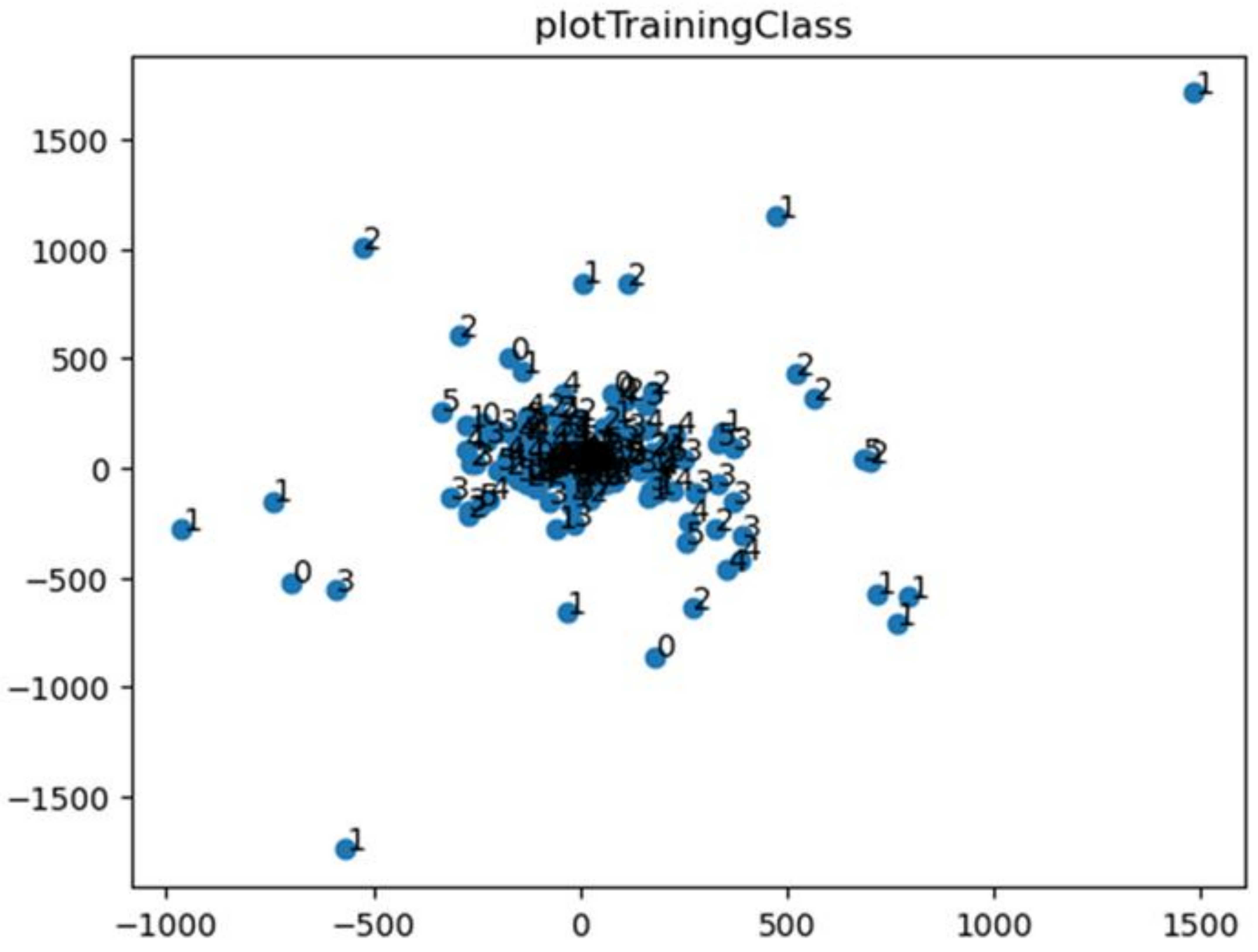
Face Space Visualization

Process:

- Each image projected into 2D PCA Space.

Interpretation:

- Distinct clusters represent individual subjects.
- Confirms successful feature separation and robust class distinction.



DISCUSSION

- PCA correctly identifies principal variations (lighting, expression).
- Higher variance retention → higher accuracy.
- Recognition performance robust under controlled conditions.
- Declines under pose or illumination changes.
- Reinforces interpretability and efficiency of PCA-based methods.

LIMITATIONS

1.

Works best under controlled lighting and pose.

Assumes same scale and alignment for all faces.

2.

Linear PCA cannot capture complex facial variation

3.

Not robust to occlusion or expression change.

4.

Requires retraining when adding new faces.

Performance drops on large/uncontrolled datasets.

Future Scope

Use a larger and more varied dataset for better accuracy.

Implement Fisherfaces (LDA) to improve class separation.

Add preprocessing like lighting and pose correction.

Optimize PCA components for speed and precision.

Build a real-time face recognition system using OpenCV

Thank You!!

Synthetic images generated for academic demonstration purposes only.