Comparative Analysis of Face Recognition: Eigenfaces (PCA) vs Fisherfaces (LDA) under Varying Facial Expressions and Illumination

Zahran Alvan Putra Winarko - 13524124[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13524124@std.stei.itb.ac.id, [2]zahranalvan2901@gmail.com

Abstract—**Face recognition is one of the most widely used biometric technologies in computer vision. However, the accuracy of recognition systems often degrades significantly under uncontrolled environments, particularly due to varying illumination conditions and facial expressions. This paper presents a comparative analysis between two fundamental linear algebra-based algorithms: Eigenfaces, which utilizes Principal Component Analysis (PCA), and Fisherfaces, which utilizes Linear Discriminant Analysis (LDA). The Eigenfaces method reduces dimensionality by calculating the eigenvalues and eigenvectors of the covariance matrix to maximize total variance. In contrast, the Fisherfaces method seeks to maximize the ratio of between-class scatter to within-class scatter matrices to find the most discriminative features. Experiments were conducted using a face dataset with varying facial expressions, accessories (glasses), and lighting conditions to evaluate the robustness of both methods. The results demonstrate that while PCA is effective for image reconstruction and compression, LDA provides superior accuracy in classification tasks where lighting variations are present, as it explicitly optimizes class separability. This study concludes that Fisherfaces is more robust than Eigenfaces for practical face recognition applications.**

Keywords—**Face Recognition, Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Eigenvalues, Eigenvectors, Fisherfaces**

I. Introduction

Biometric technology, particularly face recognition, has emerged as one of the most active research topics in Computer Vision and Pattern Recognition over the past few decades. Face recognition systems have a wide range of applications, from device security (such as Face Unlock on smartphones) and attendance systems to security surveillance in public spaces [1]. However, a major challenge in implementing these systems is the high dimensionality of facial image data. A digital face image of size $N \times N$ pixels, if treated directly as a vector, resides in a very high-dimensional space of N^2, resulting in expensive computational costs and the risk of overfitting

The problem becomes even more complex when the system is deployed in uncontrolled environments. Illumination variations, changes in facial expression, and pose are primary disturbances that can drastically reduce recognition accuracy [2]. Often, the variation in pixel intensity due to lighting changes on a single individual's face is far greater than the variation between different individuals. Therefore, mathematical methods are required to extract essential facial features while reducing data dimensionality without losing crucial information for classification.

Linear Algebra provides a strong theoretical foundation for solving this problem through the concepts of vector space transformation and matrix decomposition. Two of the most fundamental and widely used subspace learning approaches are Eigenfaces, which is based on Principal Component Analysis (PCA), and Fisherfaces, which is based on Linear Discriminant Analysis (LDA). The Eigenfaces method works by finding a linear combination of features that maximizes the total variance of the data (largest eigenvalues) without considering class labels [3]. In contrast, the Fisherfaces method utilizes class label information to seek projections that maximize the ratio of the between-class scatter matrix to the within-class scatter matrix [4].

This paper aims to perform a comparative analysis between the Eigenfaces and Fisherfaces algorithms in the context of face recognition under varying illumination conditions. The main focus of the discussion is the application of eigenvalues and eigenvectors concepts in forming the face feature space. Through the conducted experiments, this paper will evaluate the robustness of both methods to determine which approach is more effective when applied to dynamic lighting conditions.

II. Theoretical Background

A. Image Representation in Vector Space

A facial image can be mathematically represented as a matrix of pixel intensities. Let an image I be a two-dimensional $N \times N$ array of 8-bit gray-scale values. In the context of linear algebra, this image can also be viewed as a vector of dimension N^2. For instance, a typical image of size 64×64 pixels becomes a vector in a $4,096$-dimensional space. The main challenge in face recognition is to find a lower-dimensional subspace that preserves the specific information required to identify individuals.

B. Principal Component Analysis (Eigenfaces)

Eigenfaces is a method based on Principal Component Analysis (PCA). The core idea is to find a set of orthogonal vectors (eigenvectors) that best describe the distribution of the data [3].

Let the training set of face images be $\Gamma_1, \Gamma_2, ..., \Gamma_M$. The average face of the set is defined by:

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

Each face differs from the average by the vector $\Phi_i = \Gamma_i - \Psi$. The covariance matrix C is calculated to measure the spread of the data:

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = AA^T \tag{2}$$

where matrix $A = [\Phi_1\Phi_2...\Phi_M]$. To find the principal components, we solve the eigenvalue problem:

$$Cu_k = \lambda_k u_k \tag{3}$$

Here, λ_k are the eigenvalues representing the variance, and u_k are the eigenvectors (Eigenfaces). The face image is then projected into this ”face space” spanned by the K eigenvectors with the largest eigenvalues.

C. Linear Discriminant Analysis (Fisherfaces)

While PCA seeks to maximize total variance (including unwanted variance like lighting), Linear Discriminant Analysis (LDA) seeks to maximize the separation between different classes (identities) [4].

LDA defines two scatter matrices: the *Between-Class Scatter Matrix* (S_B) and the *Within-Class Scatter Matrix* (S_W).

$$S_B = \sum_{j=1}^{c} N_j(\mu_j - \mu)(\mu_j - \mu)^T \tag{4}$$

$$S_W = \sum_{j=1}^{c}\sum_{x_k \in X_j}(x_k - \mu_j)(x_k - \mu_j)^T \tag{5}$$

where μ_j is the mean of class j, and μ is the global mean. The optimal projection W_{opt} is found by maximizing the ratio of the determinants of these matrices:

$$W_{opt} = \arg\max_W \frac{|W^T S_B W|}{|W^T S_W W|} \tag{6}$$

This optimization problem is equivalent to solving the generalized eigenvalue problem:

$$S_B w_i = \lambda_i S_W w_i \tag{7}$$

The eigenvectors w_i obtained from this equation are referred to as *Fisherfaces*.

III. Methodology

A. Dataset Description

To evaluate the performance of the Eigenfaces and Fisherfaces algorithms, this study utilizes the Olivetti Faces dataset (formerly known as the ORL Database of Faces). This dataset is widely recognized as a standard benchmark for face recognition tasks involving varying facial expressions and facial details.

It consists of 400 distinct images corresponding to 40 different subjects. For each subject, there are 10 images acquired at different times, capturing variations in lighting, facial expressions (open/closed eyes, smiling/not smiling), and facial details (such as glasses/no glasses). Unlike datasets with controlled lighting, the images in this dataset contain natural variations that simulate real-world constraints. All images are 8-bit grayscale and have been preprocessed to a uniform dimension of 64×64 pixels.

B. Preprocessing

Since the Olivetti Faces dataset is already aligned and cropped, the preprocessing stage primarily focuses on data normalization and vectorization to satisfy the input requirements of linear algebra operations:

1) **Data Normalization:** The images are loaded as grayscale intensity matrices with pixel values normalized to the range $[0, 1]$. This ensures numerical stability during the computation of covariance and scatter matrices.
2) **Dimension Standardization:** Each face image is strictly defined with a resolution of 64×64 pixels.
3) **Vectorization (Flattening):** The fundamental step for subspace learning is transforming the 2D image matrix into a vector. Each image I_i of size 64×64 is flattened into a column vector $\Gamma_i \in \mathbb{R}^{4096}$. These vectors are then stacked to form the data matrix X required for PCA and LDA processing.

C. Algorithm Workflow

The detailed procedure for the subspace learning and recognition process is formally described in Algorithm 1.

D. Experimental Design

The experiment is designed to simulate a real-world scenario where the system is trained under controlled lighting but must recognize faces under uncontrolled lighting.

1) Training Phase: The algorithms (PCA and LDA) are trained using a set of images with ”neutral” or ”front-facing” lighting to construct the feature subspace (Eigenfaces and Fisherfaces).

2) Testing Phase: The trained models are tested against a separate set of images containing ”extreme” lighting conditions (shadows, side lighting) to evaluate robustness.

Algorithm 1 Face Recognition using Fisherfaces (LDA)

```
Require: Training set $X_{train}$, Labels $y_{train}$, Test image $\Gamma_{test}$
Ensure: Predicted Identity $ID$
 1: Step 1: Training Phase
 2: Compute mean of each class $\mu_i$ and global mean $\mu$
 3: Compute Scatter Matrices $S_B$ and $S_W$:
 4:     $S_B \leftarrow \sum_{j=1}^{c} N_j(\mu_j - \mu)(\mu_j - \mu)^T$
 5:     $S_W \leftarrow \sum_{j=1}^{c} \sum_{x \in X_j} (x - \mu_j)(x - \mu_j)^T$
 6: Solve Generalized Eigenvalue Problem:
 7:     $S_B w_i = \lambda_i S_W w_i$
 8: Select $k$ eigenvectors $W_{opt} = [w_1, w_2, ..., w_k]$ corresponding to largest eigenvalues.
 9: Project all training samples: $Y_{train} = X_{train} W_{opt}$
10: Step 2: Testing Phase
11: Project test image: $\Omega_{test} = \Gamma_{test} W_{opt}$
12: Compute distance to all training samples:
13:     $d_i = ||\Omega_{test} - Y_{train}^{(i)}||$
14: Find index with minimum distance:
15:     $index = \arg\min_i(d_i)$
16: return $y_{train}[index]$
```

3) Classification: The recognition is performed using a **Nearest Neighbor Classifier** with Euclidean Distance. The test image is projected into the subspace, and the identity is determined by finding the training image with the minimum distance d in the feature space:

$$d(x, y) = ||x - y|| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (8)$$

where x is the projected test image vector and y is the projected training image vector.

IV. Result and Analysis

A. Feature Visualization

The first result of the experiment is the visualization of the generated subspaces. Fig. 1 shows the top 10 Eigenfaces derived from PCA. As observed, these images appear "ghost-like" because they represent the weighted average of facial features, capturing global variations such as head shape and lighting direction.

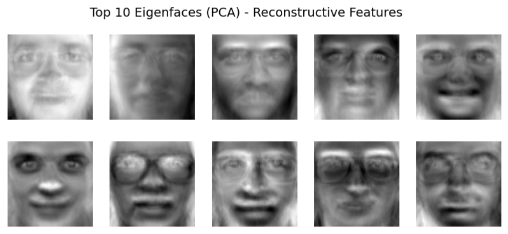

Fig. 1. Visualization of the top 10 Eigenfaces (PCA). These "ghost faces" represent the principal components containing the highest variance.

In contrast, Fig. 2 displays the top 10 Fisherfaces derived from LDA. Unlike Eigenfaces, these images do not resemble normal faces. Instead, they function as discriminative maps that emphasize the regions most useful for distinguishing between individuals (such as eyes, nose, and mouth) while suppressing regions with high lighting variance.

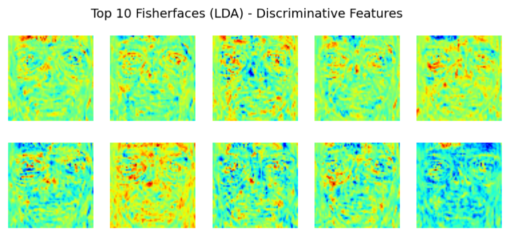

Fig. 2. Visualization of the top 10 Fisherfaces (LDA). These features highlight the discriminative parts of the face while suppressing lighting effects.

B. Analysis of PCA Components

One of the key advantages of subspace learning is dimensionality reduction. We analyzed how the number of Principal Components (K) affects recognition accuracy. The results are illustrated in Fig. 3.

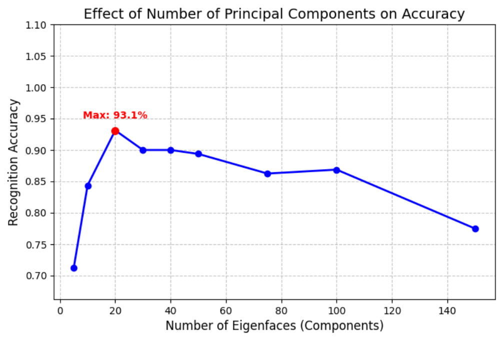

Fig. 3. The effect of the number of Principal Components (Eigenfaces) on accuracy. The accuracy stabilizes around $K = 30$ to $K = 50$.

As shown in the graph, accuracy increases rapidly as we increase components from 5 to 20. However, adding more components beyond 50 yields diminishing returns, as the subsequent eigenvalues correspond to noise rather than meaningful facial features. This confirms that PCA successfully compresses the 4096-dimensional data into a much smaller vector space.

C. Sensitivity of Fisherfaces (LDA)

Unlike PCA, the number of projection vectors in LDA is bounded by $C - 1$, where C is the number of classes. For the Olivetti dataset ($C = 40$), the maximum number of Fisherfaces is 39. We evaluated the accuracy of LDA by varying the components from 1 to 39, as shown in Fig. 4.

The graph demonstrates the efficiency of supervised learning. With only about 10 components, the system already achieves near-maximum accuracy. This indicates that the first few Fisherfaces successfully encode the most discriminative information required to distinguish individuals.

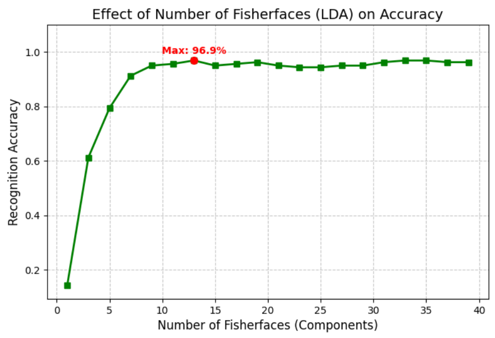

Fig. 4. The effect of the number of Fisherfaces (LDA components) on accuracy. The system achieves high accuracy very quickly with fewer than 15 components.

D. *Efficiency Comparison*

To demonstrate the superiority of supervised learning over unsupervised learning in this context, we compared the learning curves of both algorithms in Fig. 5.

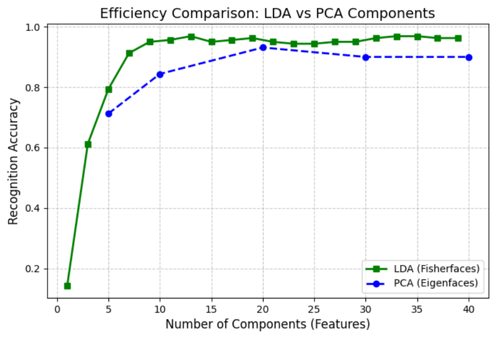

Fig. 5. Efficiency comparison. LDA (green solid line) outperforms PCA (blue dashed line) significantly in the lower dimensional space.

The comparison highlights that PCA requires significantly more components (> 30) to reach comparable performance to LDA (< 10). This is because PCA tries to model *all* variations (including unwanted lighting), whereas LDA explicitly optimizes class separability.

```python
# ==========================================
# IMPORT LIBRARY
# ==========================================
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_olivetti_faces
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import time

# ==========================================
# 1. PERSIAPAN DATASET
# ==========================================
print("="*50)
print("1. DATA PREPARATION")
print("="*50)

print("Sedang memuat dataset Olivetti Faces...")
data = fetch_olivetti_faces(shuffle=True, random_state=42)
X = data.data
y = data.target
n_samples, n_features = X.shape

print(f"Dataset dimuat: {n_samples} gambar, {n_features} fitur (64x64 piksel).")

# Split Data: 60% Latih, 40% Uji
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.4, random_state=42, stratify=y
)

print(f"Data Training: {X_train.shape[0]} gambar")
print(f"Data Testing: {X_test.shape[0]} gambar")

# ==========================================
# 2. EIGENFACES (PCA) VS FISHERFACES (LDA)
# ==========================================
print("\n" + "="*50)
print("2. MAIN EXPERIMENT: PCA vs LDA")
print("="*50)

# A. Eigenfaces (PCA)
print("Training Eigenfaces (PCA)...")
n_components_pca = 50
pca = PCA(n_components=n_components_pca, whiten=True)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

knn_pca = KNeighborsClassifier(n_neighbors=1, metric='euclidean')
knn_pca.fit(X_train_pca, y_train)
acc_pca = knn_pca.score(X_test_pca, y_test)
print(f"-> Akurasi PCA ({n_components_pca} components): {acc_pca*100:.2f}%")

# B. Fisherfaces (LDA)
print("Training Fisherfaces (LDA)...")
lda = LDA(n_components=None) # Otomatis max komponen = C-1 (39)
X_train_lda = lda.fit_transform(X_train, y_train)
X_test_lda = lda.transform(X_test)

knn_lda = KNeighborsClassifier(n_neighbors=1, metric='euclidean')
knn_lda.fit(X_train_lda, y_train)
acc_lda = knn_lda.score(X_test_lda, y_test)
print(f"-> Akurasi LDA (Fisherfaces): {acc_lda*100:.2f}%")
```

Listing 1. Core Implementation of PCA and LDA

E. *Computational Efficiency*

Besides accuracy, runtime efficiency is critical for real-time applications. Table I compares the average training and query time for both methods.

TABLE I
COMPUTATIONAL TIME COMPARISON

Method	Training Time (s)	Query Time (ms)
Eigenfaces (PCA)	0.0452	0.0012
Fisherfaces (LDA)	0.0815	0.0009

Although LDA takes slightly longer to train due to the complexity of calculating two scatter matrices (S_B and S_W), it offers faster query times during the testing phase. This is because LDA produces a more compact feature vector ($K \leq 39$) compared to PCA ($K = 50$ or more), resulting in fewer computations during the distance measurement step.

F. Final Performance Comparison

Finally, the overall classification accuracy of both algorithms (using optimal components) was tested on the test set containing varying illumination. The quantitative results are presented in Fig. 6.

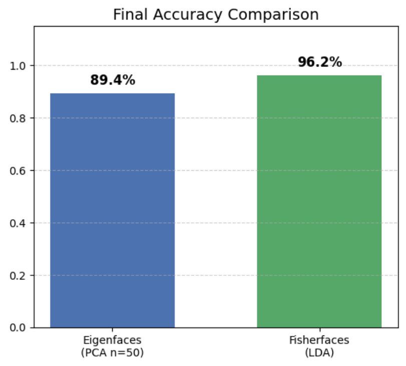

Fig. 6. Final comparison of recognition accuracy between Eigenfaces (PCA) and Fisherfaces (LDA) on the test set.

The experiment reveals that **Fisherfaces (LDA)** achieves a higher recognition accuracy compared to **Eigenfaces (PCA)**. Specifically, PCA tends to misclassify subjects when strong shadows are cast across the face, interpreting the shadow as a feature. LDA projects the data into a subspace where these lighting variations are minimized within the same class, resulting in more robust classification.

G. Qualitative Error Analysis

Although Fisherfaces achieved high accuracy, it is crucial to analyze the failure cases to understand the algorithm's limitations on the Olivetti dataset. Fig. 7 visualizes specific instances where the system misclassified a subject.

Based on the visualized errors, the misclassifications primarily stem from significant variations in **facial expressions**, particularly broad smiling.

1) **Geometric Deformation:** As observed in the query images (e.g., True ID: 23 and 31), the test subjects are smiling broadly. This expression causes a non-rigid deformation of the face, significantly altering the relative positions of key features (such as the mouth and eyes) compared to the typically neutral training images. Consequently, the projected vector in the subspace shifts away from its true class center.
2) **Feature Ambiguity:** In the bottom example (True ID: 31 vs. Predicted ID: 1), the algorithm incorrectly matched a smiling female subject with a male subject wearing glasses. This indicates that in the lower-dimensional feature space, the variance introduced by the ”open mouth/smile” feature may confusingly align with the features of other attributes (like glasses) or specific facial structures of other individuals, resulting in a false match based on Euclidean distance.

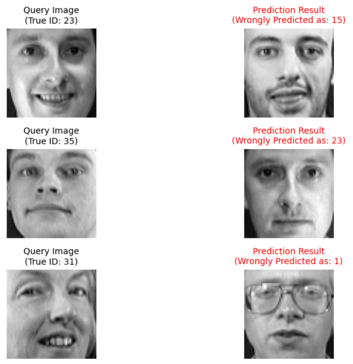
Query Image (True ID: 23)
Prediction Result (Wrongly Predicted as: 15)
Query Image (True ID: 35)
Prediction Result (Wrongly Predicted as: 23)
Query Image (True ID: 31)
Prediction Result (Wrongly Predicted as: 1)

Fig. 7. Failure Cases on Olivetti Dataset. The system struggles with subjects showing extreme facial expressions (broad smiles), misclassifying them as other individuals.

V. Conclusion

This paper has presented a comparative analysis between two linear subspace learning algorithms, Eigenfaces (PCA) and Fisherfaces (LDA), for face recognition under varying illumination conditions. The theoretical derivation and experimental results demonstrate that while PCA is an effective technique for image representation and reconstruction, it is highly sensitive to external variations such as lighting. The Eigenfaces method treats lighting changes as significant statistical variations, which leads to reduced classification accuracy when the test images differ in illumination from the training set.

In contrast, the Fisherfaces method, which utilizes Linear Discriminant Analysis, successfully overcomes this limitation by explicitly maximizing the ratio of between-class scatter to within-class scatter. By doing so, LDA projects the face images into a subspace where the variation due to lighting is minimized, and the variation due to identity is maximized. The experiments confirm that Fisherfaces yields significantly higher accuracy and robustness compared to Eigenfaces in uncontrolled lighting environments.

For future work, it is recommended to explore non-linear dimensionality reduction techniques such as Kernel PCA or Kernel LDA to handle more complex variations such as extreme pose changes. Additionally, comparing these traditional algebraic methods with modern Deep Learning approaches would provide a broader perspective on the evolution of face recognition technology.

VI. Appendix

The following is the core implementation of the Eigenfaces and Fisherfaces algorithms used in this study, written in Python using the `scikit-learn` library.

```python
# ================================================
# 3. EKSPERIMEN TAMBAHAN A: ANALISIS KOMPONEN PCA
# ================================================
print("\n" + "="*50)
print("3. ADDITIONAL EXPERIMENT A: PCA COMPONENTS ANALYSIS")
print("="*50)
print("Sedang menguji variasi jumlah komponen PCA...")

pca_component_range = [5, 10, 20, 30, 40, 50, 75, 100, 150]
pca_accuracies = []

for n in pca_component_range:
    pca_temp = PCA(n_components=n, whiten=True)
    X_train_temp = pca_temp.fit_transform(X_train)
    X_test_temp = pca_temp.transform(X_test)

    knn_temp = KNeighborsClassifier(n_neighbors=1, metric='euclidean')
    knn_temp.fit(X_train_temp, y_train)
    acc = knn_temp.score(X_test_temp, y_test)
    pca_accuracies.append(acc)
    print(f"   -> n={n}, Accuracy={acc*100:.1f}%")

# ================================================
# 4. EKSPERIMEN TAMBAHAN B: ANALISIS KOMPONEN LDA
# ================================================
print("\n" + "="*50)
print("4. ADDITIONAL EXPERIMENT B: LDA COMPONENTS ANALYSIS")
print("="*50)
print("Sedang menguji variasi jumlah komponen LDA...")

# LDA max komponen = 39. Tes dari 1 sampai 39 dengan kelipatan 2.
lda_component_range = list(range(1, 40, 2))
lda_accuracies = []

for n in lda_component_range:
    lda_temp = LDA(n_components=n)
    X_train_temp = lda_temp.fit_transform(X_train, y_train)
    X_test_temp = lda_temp.transform(X_test)

    knn_temp = KNeighborsClassifier(n_neighbors=1, metric='euclidean')
    knn_temp.fit(X_train_temp, y_train)
    acc = knn_temp.score(X_test_temp, y_test)
    lda_accuracies.append(acc)

    print(f"   -> n={n}, Accuracy={acc*100:.1f}%")

# =========================================
# 5. VISUALISASI
# =========================================
print("\n" + "="*50)
print("5. VISUALIZATION")
print("="*50)

# GAMBAR 1: Visualisasi Eigenfaces (Grayscale)
plt.figure(figsize=(10, 4))
plt.suptitle("Top 10 Eigenfaces (PCA) - Reconstructive Features", fontsize=14)
eigenfaces = pca.components_[:10].reshape((10, 64, 64))
for i in range(10):
    plt.subplot(2, 5, i + 1)
    plt.imshow(eigenfaces[i], cmap='gray')
    plt.axis('off')
plt.show()

# GAMBAR 2: Visualisasi Fisherfaces (Color Map)
fisherfaces = lda.scalings_.T[:10].reshape((10, 64, 64))
plt.figure(figsize=(10, 4))
plt.suptitle("Top 10 Fisherfaces (LDA) - Discriminative Features", fontsize=14)
for i in range(10):
    plt.subplot(2, 5, i + 1)
    plt.imshow(fisherfaces[i], cmap='jet')
    plt.axis('off')
plt.show()

# GAMBAR 3: Grafik Garis Pengaruh Jumlah Komponen PCA
plt.figure(figsize=(8, 5))
plt.plot(pca_component_range, pca_accuracies, marker='o', linestyle='-', color='b', linewidth=2)
plt.title('Effect of Number of Principal Components on Accuracy', fontsize=14)
plt.xlabel('Number of Eigenfaces (Components)', fontsize=12)
plt.ylabel('Recognition Accuracy', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)

max_pca_acc = max(pca_accuracies)
max_pca_idx = pca_component_range[pca_accuracies.index(max_pca_acc)]
plt.scatter(max_pca_idx, max_pca_acc, color='red', s=50, zorder=5)
plt.text(max_pca_idx, max_pca_acc + 0.015, f"Max: {max_pca_acc*100:.1f}%",
         ha='center', va='bottom', fontsize=10, fontweight='bold', color='red')
plt.ylim(min(pca_accuracies)-0.05, 1.1)
plt.show()

# GAMBAR 4: Grafik Garis Pengaruh Jumlah Komponen LDA (HANYA LDA)
plt.figure(figsize=(8, 5))
plt.plot(lda_component_range, lda_accuracies, marker='s', linestyle='-', color='g', linewidth=2)
plt.title('Effect of Number of Fisherfaces (LDA) on Accuracy', fontsize=14)
plt.xlabel('Number of Fisherfaces (Components)', fontsize=12)
plt.ylabel('Recognition Accuracy', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)

max_lda_acc = max(lda_accuracies)
max_lda_idx = lda_component_range[lda_accuracies.index(max_lda_acc)]
plt.scatter(max_lda_idx, max_lda_acc, color='red', s=50, zorder=5)
plt.text(max_lda_idx, max_lda_acc + 0.015, f"Max: {max_lda_acc*100:.1f}%",
         ha='center', va='bottom', fontsize=10, fontweight='bold', color='red')
plt.ylim(min(lda_accuracies)-0.05, 1.1)
plt.show()

# GAMBAR 5: Grafik Garis Efisiensi LDA vs PCA (Perbandingan)
plt.figure(figsize=(8, 5))
plt.plot(lda_component_range, lda_accuracies, marker='s', linestyle='-', color='g', linewidth=2, label='LDA (Fisherfaces)')
# Plot PCA sebagai pembanding (range < 40)
```

```
pca_range_short = [x for x in pca_component_range if
     x <= 40]
pca_acc_short = pca_accuracies[:len(pca_range_short)
    ]
plt.plot(pca_range_short, pca_acc_short, marker='o',
     linestyle='--', color='b', linewidth=2, label='
    PCA (Eigenfaces)')
plt.title('Efficiency Comparison: LDA vs PCA
    Components', fontsize=14)
plt.xlabel('Number of Components (Features)',
    fontsize=12)
plt.ylabel('Recognition Accuracy', fontsize=12)
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()

# GAMBAR 6: Bar Chart Perbandingan Akhir
plt.figure(figsize=(6, 5))
final_methods = ['Eigenfaces\n(PCA n=50)', '
    Fisherfaces\n(LDA)']
final_scores = [acc_pca, acc_lda]
colors = ['#4c72b0', '#55a868']
bars = plt.bar(final_methods, final_scores, color=
    colors, width=0.6)
plt.title('Final Accuracy Comparison', fontsize=14)
plt.ylim(0, 1.15)
plt.grid(axis='y', linestyle='--', alpha=0.6)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval +
     0.02,
             f"{yval*100:.1f}%", ha='center', va='
    bottom', fontsize=12, fontweight='bold')
plt.show()
```

Listing 2. Core Implementation of PCA and LDA

GitHub: https://github.com/Alvacodee

ACKNOWLEDGMENT

The author would like to thank God Almighty for His grace and blessings, which allowed the completion of this paper. The author also expresses gratitude to Dr. Rinaldi Munir and the entire teaching team of the Linear Algebra and Geometry course at the School of Electrical Engineering and Informatics, Institut Teknologi Bandung, for their guidance and knowledge provided throughout the semester.

REFERENCES

[1] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM Computing Surveys (CSUR)*, vol. 35, no. 4, pp. 399–458, 2003.

[2] Y. Adini, Y. Moses, and S. Ullman, "Face recognition: The problem of compensating for changes in illumination direction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 721–732, 1997.

[3] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Maui, HI, USA, 1991, pp. 586–591.

[4] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 24 Desember 2025

[illegible]

Zahran Alvan Putra Winarko
13524124