LAPORAN PROJECT FACE RECOGNITION
DENGAN EIGEN FACE

Anggota kelompok :

1. Marleyn Laura O. V. T. (L0124023)
2. Nasywa Rifqia R. (L0124027)
3. Wan Nayyara Y. (L0124033)
4. Agischa Nur A. (L0124035)

BAB I
DESKRIPSI MASALAH

Pengenalan wajah (Face Recognition) adalah teknologi biometrik yang bisa dipakai untuk mengidentifikasi wajah seseorang untuk berbagai kepentingan khususnya keamanan. Program pengenalan wajah melibatkan kumpulan citra wajah yang sudah disimpan pada database lalu berdasarkan kumpulan citra wajah tersebut, program dapat mempelajari bentuk wajah lalu mencocokkan antara kumpulan citra wajah yang sudah dipelajari dengan citra yang akan diidentifikasi.

Terdapat berbagai teknik untuk memeriksa citra wajah dari kumpulan citra yang sudah diketahui seperti jarak Euclidean dan cosine similarity, principal component analysis (PCA), serta Eigenface.

Pada Tugas ini, akan dibuat sebuah program pengenalan wajah menggunakan Eigenface. Sekumpulan citra wajah akan digunakan dengan representasi matriks. Dari representasi matriks tersebut akan dihitung sebuah matriks Eigenface. Program pengenalan wajah dapat dibagi menjadi 2 tahap berbeda yaitu tahap training dan pencocokkan. Pada tahap training, akan diberikan kumpulan data set berupa citra wajah. Citra wajah tersebut akan dinormalisasi dari RGB ke Grayscale (matriks), hasil normalisasi akan digunakan dalam perhitungan eigenface. Seperti namanya, matriks eigenface menggunakan eigenvector dalam pembentukannya.

BAB II
DASAR TEORI

1. Perkalian Matriks

Perkalian matriks adalah operasi matematika yang melibatkan dua matriks, di mana elemen-elemen dari satu matriks dikalikan dengan elemen-elemen dari matriks lainnya dan dijumlahkan untuk menghasilkan elemen-elemen baru dalam matriks hasil perkalian.

Jika A adalah matriks ukuran $m \times n$ dan B adalah matriks ukuran $n \times p$, maka hasil perkalian A × B adalah matriks C berukuran $m \times p$, yang dihitung dengan cara:

$$C_{ij} = \sum_{k=1}^{n} A_{ik} \times B_{kj}$$

Perkalian matriks ini digunakan dalam banyak algoritma, termasuk PCA, di mana matriks data dikalikan dengan matriks eigenfaces untuk melakukan proyeksi.

2. Nilai Eigen

Nilai eigen adalah skalar yang mengukur seberapa besar pengaruh vektor eigen terhadap transformasi linier yang diberikan oleh suatu matriks. Dalam konteks PCA (*Principal Component Analysis*) atau Eigenfaces, nilai eigen mengindikasikan seberapa besar kontribusi dari suatu vektor eigen dalam representasi data. Secara matematis, jika **A** adalah matriks dan **v** adalah vektor eigen, maka nilai eigen λ adalah solusi dari persamaan:

$$A \cdot v = \lambda \cdot v$$

Di sini, A adalah matriks yang merepresentasikan transformasi linier, v adalah vektor eigen, dan λ adalah nilai eigen. Nilai eigen ini menentukan seberapa besar vektor eigen "merepresentasikan" data setelah transformasi.

3. Vektor Eigen

Vektor eigen adalah vektor non-nol yang, ketika diterapkan transformasi matriks, hanya mengalami perubahan dalam skala dan bukan dalam arah. Vektor eigen ini mengidentifikasi arah utama data, yaitu arah yang memiliki varians terbesar.

Secara matematis, vektor eigen **v** adalah vektor yang memenuhi persamaan:

$$A \cdot v = \lambda \cdot v$$

Di mana **A** adalah matriks, λ adalah nilai eigen, dan **v** adalah vektor eigen. Dalam konteks pengenalan wajah menggunakan Eigenfaces, vektor eigen ini berfungsi untuk mengidentifikasi pola utama dalam gambar wajah yang dapat digunakan untuk membedakan wajah satu dengan yang lainnya.

4. **Eigenface**

Eigenface adalah aplikasi khusus dari konsep vektor eigen dalam pengenalan wajah. Eigenfaces dihasilkan melalui analisis komponen utama (PCA), yang digunakan untuk mengurangi dimensi gambar wajah. Setiap eigenface mewakili pola utama atau fitur wajah yang paling signifikan dalam dataset gambar wajah.

Proses pembuatan Eigenfaces dimulai dengan mengonversi setiap gambar wajah menjadi vektor satu dimensi (dengan meratakan gambar). Kemudian, PCA digunakan untuk menghitung eigenfaces, yang merupakan vektor eigen dari matriks kovarians yang dihasilkan dari dataset gambar wajah. Setiap eigenface ini menggambarkan perbedaan terbesar antar wajah, dan kombinasi dari beberapa eigenfaces dapat digunakan untuk mengenali wajah.

Dalam prakteknya, gambar wajah dapat diproyeksikan ke dalam ruang eigenfaces, di mana proyeksi ini memungkinkan identifikasi wajah yang lebih efisien dan akurat.

Hubungan Antara Konsep-Konsep Ini dalam Pengenalan Wajah

- Perkalian matriks digunakan dalam proses PCA untuk mengonversi data gambar wajah menjadi ruang dimensi yang lebih rendah (menggunakan eigenfaces).
- Nilai eigen mengukur kontribusi setiap komponen utama (eigenface) dalam representasi wajah.
- Vektor eigen mewakili fitur utama yang diidentifikasi dalam wajah yang dapat digunakan untuk membedakan individu dalam dataset.
- Eigenface adalah representasi geometris dari vektor eigen, yang menunjukkan pola wajah utama yang digunakan untuk pencocokan wajah dalam aplikasi pengenalan wajah.

Dengan menggunakan konsep-konsep ini, aplikasi pengenalan wajah dapat mengidentifikasi dan mencocokkan wajah berdasarkan pola yang ditemukan melalui analisis komponen utama.

BAB III
IMPLEMENTASI PROGRAM

A. Implementasi Program Pengenalan Wajah Menggunakan Eigenfaces

Program pengenalan wajah ini menggunakan metode Eigenfaces untuk mengidentifikasi wajah seseorang. Program ini melibatkan dua hal penting: teknologi yang digunakan dan cara mengompresi data wajah agar lebih efisien dan mudah diproses. Berikut penjelasan mengenai teknologi yang digunakan dan cara kompresi yang diterapkan.

1. Teknologi yang Digunakan (Tech Stack)
    a. Streamlit:

       Streamlit adalah framework Python yang memungkinkan pengembangan aplikasi web interaktif dengan cepat dan mudah. Aplikasi ini menggunakan Streamlit untuk membuat antarmuka pengguna (UI) yang memungkinkan pengguna mengunggah gambar wajah dan memilih folder dataset gambar wajah.
    b. Fungsi dalam Program: Streamlit digunakan untuk membuat tampilan antarmuka aplikasi, di mana pengguna dapat mengunggah gambar dan memilih dataset gambar wajah untuk diproses.
    c. OpenCV (cv2):

       OpenCV adalah pustaka open-source yang digunakan untuk pengolahan gambar dan visi komputer. OpenCV menyediakan berbagai fungsi untuk membaca gambar, mengubah ukuran gambar, mengonversinya ke dalam format grayscale, dan melakukan operasi pengolahan citra lainnya.
    d. Fungsi dalam Program: OpenCV digunakan untuk membaca gambar, mengubah gambar menjadi hitam-putih (grayscale), mengubah ukuran gambar, dan meratakan gambar agar bisa digunakan dalam proses perhitungan.
    e. Numpy:

Numpy adalah pustaka Python yang digunakan untuk manipulasi array dan komputasi numerik. Pustaka ini penting dalam operasi matematis yang digunakan untuk pengolahan data matriks dan perhitungan PCA. Fungsi dalam Program: Numpy digunakan untuk melakukan perhitungan matematis seperti operasi pada matriks dan vektor, yang diperlukan dalam proses pengolahan data gambar.

f. Pillow (PIL):

Pillow adalah pustaka Python untuk pemrosesan gambar, yang memungkinkan membuka, memodifikasi, dan menyimpan gambar dalam berbagai format.
Fungsi dalam Program: Pillow digunakan untuk membuka dan menampilkan gambar yang diunggah serta hasil pencocokan wajah.

g. Time:

Pustaka standar Python untuk menangani waktu dan pengukuran durasi. Fungsi dalam Program: Pustaka ini digunakan untuk mencatat berapa lama waktu yang dibutuhkan untuk memproses dan mencocokkan wajah

2. Proses Algoritma Kompresi yang Digunakan

Pada program ini, proses kompresi data wajah dilakukan menggunakan Principal Component Analysis (PCA) untuk mengurangi ukuran data wajah dan menghasilkan Eigenfaces, yang memungkinkan program mengenali wajah dengan lebih efisien. Berikut adalah langkah-langkahnya:

a. Persiapan Dataset Gambar Wajah:
    - Dataset gambar wajah disusun dalam folder, di mana setiap folder berisi gambar wajah seseorang.
    - Setiap gambar diubah menjadi ukuran standar 100x100 piksel agar semua gambar memiliki ukuran yang sama.
    - Gambar kemudian diubah menjadi grayscale (hitam-putih) untuk menyederhanakan data dan hanya mempertahankan informasi yang penting.

- Gambar wajah kemudian diubah menjadi vektor satu dimensi (rata-rata pikselnya), yang memudahkan perhitungan selanjutnya.

b. Menghitung Wajah Rata-rata:
    - Wajah rata-rata dihitung dengan cara mengambil rata-rata dari setiap pixel di seluruh gambar dalam dataset.
    - Wajah rata-rata ini digunakan untuk mengurangi variasi data akibat faktor seperti pencahayaan yang berbeda. Semua gambar akan dikurangi dengan wajah rata-rata ini untuk menghasilkan data yang lebih konsisten.
c. Menghitung Matriks Kovarians:
    - Matriks kovarians mengukur hubungan antar elemen (pixel) dalam gambar. Matriks ini penting untuk menemukan pola utama dalam gambar wajah menggunakan PCA.
d. Menghitung Eigenfaces:
    - Eigenfaces adalah pola-pola utama yang ditemukan melalui analisis matriks kovarians. Setiap eigenface adalah sebuah vektor yang mewakili fitur utama dalam wajah, seperti bentuk wajah atau posisi mata.
    - Dalam proses ini, kita menggunakan power iteration untuk menghitung nilai eigen dan vektor eigen dari matriks kovarians.
e. Proyeksi Gambar Wajah ke dalam Ruang Eigenfaces:
    - Setelah menghitung Eigenfaces, gambar wajah yang ingin diuji diproyeksikan ke ruang eigenfaces untuk mendapatkan representasi yang lebih sederhana.
    - Proyeksi ini mengurangi dimensi data wajah yang awalnya besar (100x100 piksel) menjadi lebih kecil, hanya menggunakan beberapa Eigenfaces yang paling penting.
f. Mencocokkan Wajah:
    - Wajah yang telah diproyeksikan dibandingkan dengan wajah dalam dataset menggunakan jarak Euclidean untuk mengetahui seberapa mirip keduanya.

- Wajah yang memiliki jarak Euclidean terkecil dianggap sebagai wajah yang paling mirip dengan gambar uji.

g. Menentukan Pencocokan Berdasarkan Threshold:
   - Setelah menghitung jarak Euclidean, kita menetapkan threshold (batas) untuk menentukan apakah wajah yang diuji cocok dengan wajah dalam dataset.
   - Jika jarak Euclidean lebih kecil dari threshold, maka wajah tersebut dianggap cocok. Jika tidak, aplikasi akan memberi tahu bahwa wajah tidak ditemukan dalam dataset.

B. Bagaimana Kompresi Digunakan dalam Proses Pengolahan Gambar

a. **Pengurangan Dimensi**: Menggunakan PCA, kita dapat mengurangi ukuran data gambar wajah yang besar (misalnya, gambar 100x100 piksel yang memiliki 10.000 data) menjadi jumlah yang jauh lebih kecil, hanya beberapa Eigenfaces yang paling penting.

b. **Pengurangan Ukuran Data**: Dengan menggunakan Eigenfaces, kita hanya menyimpan informasi yang paling relevan, seperti fitur wajah yang paling menonjol, sehingga kita bisa bekerja dengan data yang lebih kecil dan lebih cepat, tanpa kehilangan informasi penting.

Program ini menggunakan teknologi seperti Streamlit untuk antarmuka pengguna dan OpenCV untuk pemrosesan gambar. Algoritma kompresi yang digunakan adalah PCA, yang membantu mengurangi ukuran data gambar wajah dan menemukan Eigenfaces. Dengan mengurangi dimensi data wajah, aplikasi dapat mengenali dan mencocokkan wajah lebih cepat dan lebih akurat.

BAB IV
EKSPERIMEN DAN ANALISIS HASIL

A. Deskripsi Umum Aplikasi

Aplikasi ini adalah implementasi pengenalan wajah menggunakan Metode Eigenfaces yang berbasis pada Principal Component Analysis (PCA). PCA digunakan untuk mengurangi dimensi gambar wajah dan menemukan "komponen utama" (Eigenfaces) yang dapat digunakan untuk membedakan wajah yang satu dengan yang lainnya. Aplikasi ini dirancang dengan menggunakan Streamlit untuk membangun antarmuka pengguna yang interaktif dan mudah digunakan.

B. Langkah-langkah Utama dalam Aplikasi

a. Persiapan Gambar:

   Aplikasi pertama-tama akan meminta pengguna untuk mengunggah gambar uji, yang akan dibandingkan dengan gambar dalam dataset wajah yang telah dilatih sebelumnya

b. Pengolahan Gambar:
   - Semua gambar dalam dataset akan diubah menjadi format grayscale untuk mengurangi kompleksitas komputasi.
     Setiap gambar akan diubah ukurannya menjadi ukuran standar (misalnya, 100x100 pixel) dan diratakan menjadi vektor satu dimensi.

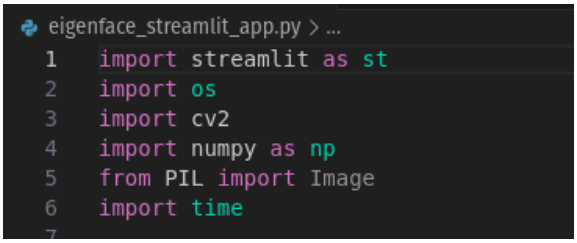
c. Pencocokan Wajah:
   - Setelah gambar uji diunggah, aplikasi akan menghitung jarak Euclidean antara vektor fitur wajah uji dan dataset wajah yang sudah ada.
   - Jika jarak tersebut lebih kecil dari ambang batas yang ditentukan, maka aplikasi akan mengonfirmasi bahwa wajah uji dikenali.

d. Output:
   - Gambar uji akan ditampilkan bersama dengan gambar yang paling cocok dari dataset.

- Aplikasi juga akan menampilkan jarak Euclidean antara gambar uji dan gambar yang dicocokkan serta waktu yang diperlukan untuk proses pencocokan.

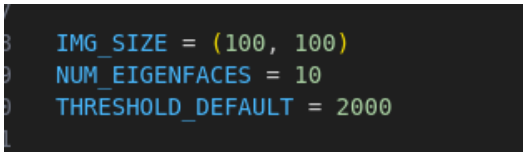
C. Penjelasan Kode Secara Rinci

1. Import Library



- streamlit: Digunakan untuk membuat antarmuka pengguna (UI) aplikasi web interaktif.
- os: Digunakan untuk berinteraksi dengan file system, terutama untuk membaca folder dan file gambar.
- cv2: Merupakan pustaka OpenCV yang digunakan untuk pemrosesan gambar, seperti membaca gambar, mengonversinya menjadi grayscale, dan meresize gambar.
- numpy: Digunakan untuk manipulasi array dan matriks yang diperlukan dalam komputasi pengolahan citra dan vektor.
- PIL.Image: Digunakan untuk manipulasi gambar seperti konversi format gambar.
- time: Untuk menghitung waktu eksekusi dari seluruh proses.

2. Konstanta dan Parameter



- IMG_SIZE: Ukuran standar untuk meresize gambar menjadi 100x100 piksel. Hal ini diperlukan untuk menyamakan ukuran gambar sebelum diproses lebih lanjut.
- NUM_EIGENFACES: Jumlah Eigenfaces yang akan digunakan untuk analisis pengenalan wajah. Eigenfaces adalah komponen utama yang ditemukan melalui PCA.
- THRESHOLD_DEFAULT: Ambang batas kemiripan wajah yang digunakan untuk mencocokkan wajah uji dengan dataset. Semakin rendah nilai threshold, semakin ketat kriteria pencocokan wajah.

3. Fungsi load_training_images(folder_path)

Fungsi ini digunakan untuk memuat dataset gambar wajah yang ada dalam folder tertentu. Dataset ini terdiri dari sub-folder yang masing-masing berisi gambar wajah yang terkait dengan satu individu.

```
def load_training_images(folder_path):
    image_vectors, labels, raw_images = [], [], []
    for person_name in os.listdir(folder_path):
        person_folder = os.path.join(folder_path, person_name)
        if not os.path.isdir(person_folder):
            continue
        for filename in os.listdir(person_folder):
            img_path = os.path.join(person_folder, filename)
            img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
            if img is not None:
                img_resized = cv2.resize(img, IMG_SIZE)
                img_vector = img_resized.flatten()
                image_vectors.append(img_vector)
                labels.append(person_name)
                raw_images.append(img_resized)
    return np.array(image_vectors).T, np.array(labels), raw_images
```

- image_vectors: Menyimpan vektor fitur dari setiap gambar (gambaran wajah yang sudah diproses).
- labels: Menyimpan label yang terkait dengan setiap gambar (nama orang).
- raw_images: Menyimpan gambar asli (sebelum diproses) untuk ditampilkan setelah pencocokan wajah.

Fungsi ini akan mengakses setiap gambar dalam folder dataset, mengubahnya menjadi gambar grayscale, meresize, dan meratakannya menjadi vektor satu dimensi.

4. **Fungsi compute_mean_face(data_matrix)**

Fungsi ini menghitung wajah rata-rata (mean face) dari semua gambar dalam dataset.

```
def compute_mean_face(data_matrix):
    mean_face = np.mean(data_matrix, axis=1, keepdims=True)
    centered_matrix = data_matrix - mean_face
    return mean_face, centered_matrix
```

- mean_face: Wajah rata-rata dihitung dengan cara mengambil rata-rata setiap piksel dari seluruh gambar.
- centered_matrix: Matriks yang dihasilkan dengan mengurangkan mean_face dari setiap gambar. Ini penting untuk tahap selanjutnya dalam PCA.

5. **Fungsi compute_covariance_trick(centered_matrix)**

Fungsi ini menghitung matriks kovarians yang diperlukan dalam langkah PCA untuk mendapatkan komponen utama.

```
def compute_covariance_trick(centered_matrix):
    return centered_matrix.T @ centered_matrix
```

- centered_matrix.T @ centered_matrix: Matriks kovarians dihitung dengan mengalikan matriks terpusat dengan transpose-nya sendiri.

6. **Fungsi power_iteration(A, num_iter=1000, tol=1e-6)**

Fungsi ini digunakan untuk mencari nilai eigen dan vektor eigen utama menggunakan metode power iteration.

```
def power_iteration(A, num_iter=1000, tol=1e-6):
    n = A.shape[0]
    b = np.random.rand(n)
    b = b / np.linalg.norm(b)
    for _ in range(num_iter):
        Ab = A @ b
        b_new = Ab / np.linalg.norm(Ab)
        if np.linalg.norm(b - b_new) < tol:
            break
        b = b_new
    eigenvalue = b.T @ A @ b
    return eigenvalue, b
```

- A @ b: Perkalian matriks kovarians dengan vektor acak b.
- b_new: Vektor eigen baru setelah iterasi. Proses ini diulang hingga konvergen.

7. **Fungsi compute_top_eigenfaces(centered_matrix, k=10)**

Fungsi ini digunakan untuk menghitung k eigenfaces teratas berdasarkan matriks kovarians yang telah dihitung sebelumnya.

```
def compute_top_eigenfaces(centered_matrix, k=10):
    L = compute_covariance_trick(centered_matrix)
    eigenvectors = []
    for _ in range(k):
        _, eigenvector = power_iteration(L)
        eigenface = centered_matrix @ eigenvector
        eigenface = eigenface / np.linalg.norm(eigenface)
        eigenvectors.append(eigenface)
        L = L - (eigenvector @ eigenvector.T) * (eigenvector.T @ L @ eigenvector)
    return np.array(eigenvectors).T

def project_face(face_vector, mean_face, eigenfaces):
    centered = face_vector - mean_face.flatten()
    weights = eigenfaces.T @ centered
    return weights
```

- L: Matriks kovarians yang digunakan untuk menghitung eigenfaces.
- eigenface: Vektor yang menunjukkan fitur utama dari gambar wajah.

8. **Fungsi project_face(face_vector, mean_face, eigenfaces)**

Fungsi ini digunakan untuk memproyeksikan gambar wajah uji ke ruang eigenfaces.

```python
def project_face(face_vector, mean_face, eigenfaces):
    centered = face_vector - mean_face.flatten()
    weights = eigenfaces.T @ centered
    return weights
```

- weights: Bobot dari gambar wajah yang diproyeksikan ke ruang eigenfaces.

9. Fungsi main()

Fungsi utama yang menginisiasi aplikasi menggunakan Streamlit.

```python
def main():
    st.title("🧠 Face Recognition App (EigenFace)")

    dataset_path = st.text_input("📁 Dataset Folder Path", "dataset")
    uploaded_file = st.file_uploader("📤 Upload Gambar Test", type=["jpg", "jpeg", "png"])
    threshold = st.slider("🎚️ Threshold Kemiripan", 0, 10000, value=THRESHOLD_DEFAULT)

    if st.button("🚀 Eksekusi") and uploaded_file is not None:
        start_time = time.time()

        # Load gambar test (tidak dari dataset)
        file_bytes = np.asarray(bytearray(uploaded_file.read()), dtype=np.uint8)
        test_img = cv2.imdecode(file_bytes, cv2.IMREAD_GRAYSCALE)
        test_img_resized = cv2.resize(test_img, IMG_SIZE)
        test_vector = test_img_resized.flatten()

        # Load data latih dari folder dataset
        data_matrix, labels, raw_images = load_training_images(dataset_path)
        if data_matrix.size == 0:
            st.error("❌ Dataset kosong atau tidak ditemukan.")
            return

        mean_face, centered_matrix = compute_mean_face(data_matrix)
        eigenfaces = compute_top_eigenfaces(centered_matrix, NUM_EIGENFACES)

        projected_weights = []
        for i in range(data_matrix.shape[1]):
            weights = project_face(data_matrix[:, i], mean_face, eigenfaces)
            projected_weights.append(weights)
        projected_weights = np.array(projected_weights)

        test_weights = project_face(test_vector, mean_face, eigenfaces)
        distances = np.linalg.norm(projected_weights - test_weights, axis=1)
        closest_idx = np.argmin(distances)
        closest_distance = distances[closest_idx]

        col1, col2 = st.columns(2)
        col1.image(test_img_resized, caption="Test Image", width=200)
        col2.image(raw_images[closest_idx], caption=f"Matched Image: {labels[closest_idx]}", width=200)

        st.markdown("### 🔍 Hasil")
        if closest_distance < threshold:
            st.success(f"Wajah dikenali sebagai: **{labels[closest_idx]}**")
        else:
            st.error("Tidak ditemukan wajah yang cocok.")
        st.write(f"📉 Jarak Euclidean: {closest_distance:.2f}")
        st.write(f"⏱️ Waktu Eksekusi: {time.time() - start_time:.2f} detik")

if __name__ == "__main__":
    main()
```

Fungsi ini mengatur antarmuka pengguna, memungkinkan pengguna untuk mengunggah gambar uji dan memilih folder dataset, serta menentukan ambang batas kemiripan wajah. Setelah gambar diunggah, aplikasi melakukan langkah-langkah berikut:

a. Membaca dan memproses gambar uji.
b. Memuat gambar-gambar dari folder dataset dan menghitung eigenfaces.

c. Menghitung bobot gambar uji dan dataset, serta menghitung jarak Euclidean untuk mencocokkan wajah.
d. Menampilkan hasil pencocokan bersama gambar uji dan gambar yang paling cocok dari dataset.

10. Output Aplikasi

Output aplikasi mencakup:

- Test Image: Gambar uji yang diunggah oleh pengguna.
- Matched Image: Gambar yang paling mirip berdasarkan dataset.
- Euclidean Distance: Ukuran kesamaan antara gambar uji dan dataset.
- Execution Time: Waktu yang diperlukan untuk menjalankan proses pencocokan wajah.

BAB V
KESIMPULAN

Proyek ini berhasil mengimplementasikan sistem pengenalan wajah berbasis metode Eigenface dengan menggunakan konsep Principal Component Analysis (PCA). Program ini mampu mengidentifikasi wajah dengan mengubah citra wajah menjadi bentuk vektor, menghitung wajah rata-rata, membentuk eigenfaces, dan melakukan proyeksi gambar uji ke ruang fitur eigen. Proses pencocokan dilakukan dengan menghitung jarak Euclidean antara wajah uji dan data latih, dan hasilnya mampu menunjukkan wajah yang paling mirip dalam dataset beserta nilai jarak dan waktu eksekusinya.

Melalui pemanfaatan pustaka seperti OpenCV, Numpy, dan antarmuka interaktif Streamlit, sistem ini menunjukkan kinerja yang cukup baik dalam mengenali wajah-wajah yang telah ada dalam dataset. Keberhasilan ini menunjukkan bahwa penerapan konsep aljabar linier seperti eigenvalue, eigenvector, dan transformasi ruang vektor dapat diimplementasikan secara efektif dalam pengolahan citra digital, khususnya untuk aplikasi biometrik seperti face recognition.

Berdasarkan projek yang telah kami kerjakan, kami mendapati beberapa hal yang bisa dikembangkan, antara lain:

1. Menambah dataset yang lebih besar dan beragam, termasuk ekspresi wajah, pencahayaan, dan sudut wajah yang berbeda, untuk meningkatkan akurasi.
2. Menggunakan deteksi wajah otomatis (misalnya dengan Haar Cascade dari OpenCV) sebelum proses PCA, agar sistem bisa langsung mengambil bagian wajah tanpa perlu cropping manual.
3. Mengintegrasikan metode klasifikasi lain, seperti SVM atau k-NN setelah proyeksi PCA untuk membandingkan akurasi pengenalan.
4. Menambahkan sistem login atau autentikasi, jika sistem akan digunakan dalam konteks keamanan nyata.
5. Peningkatan performa dan efisiensi, misalnya dengan mempercepat perhitungan eigenvector menggunakan pustaka yang lebih optimal seperti scikit-learn.

Adapun pemahaman yang kami dapatkan dari pengerjaan projek ini ialah:

1. Penerapan konsep aljabar linier dalam dunia nyata, khususnya melalui eigenvector, eigenvalue, dan transformasi ruang vektor,
2. Bagaimana kompresi data dengan PCA mampu menyederhanakan data besar tanpa kehilangan informasi penting,
3. Pentingnya pra-pemrosesan gambar dalam meningkatkan kualitas hasil pengenalan wajah,
4. Dan bagaimana membangun antarmuka interaktif yang mempermudah pengguna akhir melalui Streamlit.

Tugas ini tidak hanya memperkuat pemahaman teknis kami, tetapi juga mengasah kemampuan kerja tim dan manajemen proyek. Dengan tantangan nyata yang dihadapi dalam pemrosesan citra dan implementasi algoritma, kami merasa lebih siap untuk menghadapi proyek-proyek teknologi yang lebih kompleks ke depannya.

DAFTAR PUSTAKA

Muhammad Syarif, Andrian Rakhmatsyah, Adiwijaya. 2007. Pengenalan Wajah Manusia Menggunakan Metode Eigenface dan City Blok. Diakses pada situs https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://openlibrary.telkomuniversity.ac.id/pustaka/files/94778/resume/pengenalan-wajah-manusia-menggunakan-metode-eigenface-dan-city-block.pdf&ved=2ahUKEwi4n8GgjemNAxXg3TgGHVvGGJIQFnoECGMQAQ&usg=AOvVaw1doVwjGxQWkBi4W4urc2qz

Tri Mulyono, Kusworo Adi dan Rahmat Gernowo. 2014. Sistem Pengenalan Wajah Dengan Metode Eigenface dan Jaringan Syaraf Tiruan. Diakses pada http://download.garuda.kemdikbud.go.id/article.php?article=1406763&val=1288&title=SISTEM%20PENGENALAN%20WAJAH%20DENGAN%20METODE%20EIGENFACE%20DAN%20JARINGAN%20SYARAF%20TIRUAN%20JST

Widodo Muda Saputra, Helmie Arif Wibawa, Nurdin Bahtiar. 2013. *Pengenalan Wajah Menggunakan Algoritma Eigenface Dan Euclidean Distance.* Diakses pada situs https://www.neliti.com/id/publications/135138/pengenalan-wajah-menggunakan-algoritma-eigenface-dan-euclidean-distance

LAMPIRAN

https://github.com/agischa/eigenface-project