

FACE RECOGNITION MENGGUNAKAN ALGORITMA EIGENFACE

Diajukan sebagai pemenuhan tugas besar 2.



Oleh:

Kelompok 7 (*kombi-NaToRy*)

1. 13521067 - Yobel Dean Christoper
2. 13521131 - Jeremy Dharmawan Rahardjo
3. 13521162 - Antonio Natthan Krishna

Dosen Pengampu : Dr. Ir. Rinaldi Munir, MT.

IF2123 - Aljabar Linier dan Geometri

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2022**

DAFTAR ISI

DAFTAR ISI	1
BAB I	2
BAB II	3
2.1. Interpretasi Citra sebagai Matriks	3
2.2. Nilai Eigen dan Vektor Eigen	4
2.3. Dekomposisi QR	5
2.4. Dekomposisi QR dengan Algoritma Schwarz-Rutishauser	6
2.5. Tahapan Training (Algoritma Eigenface)	6
BAB III	7
3.1. Pustaka/Kakas	8
3.1.1. NumPy	8
3.1.2. OpenCV	8
3.1.3. PIL	9
3.1.4. Tkinter	10
3.1.5. Custom Tkinter	10
3.2. Algoritma Input File dan Kompresi	10
3.3. Algoritma Pemrosesan Training Image menjadi Eigenface	11
3.3.1 Dekomposisi QR dengan Algoritma Schwarz-Rutishauser	11
3.3.2 Kalkulasi Vektor Eigen dan Eigenface	12
3.4. Tahapan Uji Gambar	13
BAB IV	14
1. Interface	15
2. Testing Error Handling	15
3. Testing DataSet Kaggle	17
4. Testing DataSet Kaggle	20
5. Testing DataSet Kaggle	22
6. Testing DataSet dengan Input Kamera	25
7. Rangkuman Testing	26
BAB V	28
DAFTAR REFERENSI	30
LAMPIRAN	30

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 besar 2 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. Berikut merupakan langkah rinci dalam pembentukan eigenface.

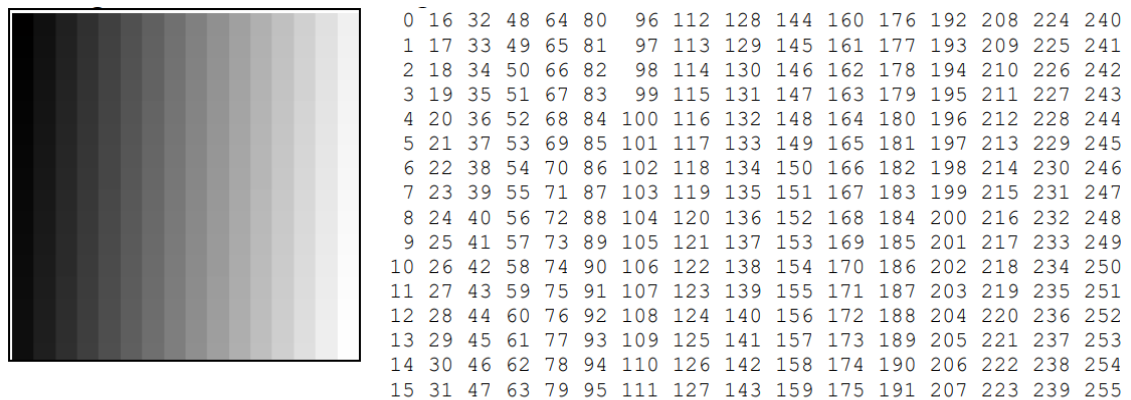
Dalam Tugas Besar 2 ini, kami ditugaskan untuk membuat aplikasi pengenalan wajah sederhana. Aplikasi ini menggunakan bahasa Python dengan menggunakan library utama OpenCV. Sebuah test-image akan diuji dengan data dari dataset input, setelah itu aplikasi akan menampilkan gambar dari dataset yang paling mirip dengan test-image. Aplikasi ini juga menerima input dari kamera secara langsung, sehingga pemrosesan dapat dilakukan secara real-time.

BAB II

DASAR TEORI

2.1. Interpretasi Citra sebagai Matriks

Terdapat relasi antara matriks dan gambar digital. Sebuah gambar digital pada sebuah komputer disusun oleh matriks *pixel*. Sebuah gambar hitam-putih disusun oleh sebuah matriks yang elemennya berupa integer (0, 1, ..., 255). Angka-angka ini merepresentasikan warna yang secara konstan berubah dari 0 (hitam) hingga 255 (putih). Seperti contoh,



Gambar 1. Representasi gradasi warna hitam-putih dalam sebuah matriks

Gradasi warna seperti yang ditunjukkan pada Gambar 1 dapat direpresentasikan sebagai sebuah matriks. Dalam dunia pemrosesan gambar, matriks inilah yang akan diolah sedemikian rupa, sehingga dapat merepresentasikan suatu gambar yang *custom* sesuai dengan keinginan pengguna.

Dalam gambar berwarna, terdapat beberapa versi representasi matriks, salah satunya dalam model RGB. Gambar yang menggunakan model warna RGB direpresentasikan dalam 3 buah matriks gambar hitam putih (yang masing masing mewakili warna Red, Green, Blue). Ketiga matriks ini kemudian disatukan sehingga dapat menghasilkan sebuah representasi gambar berwarna.

Penggunaan matriks sebagai representasi gambar merupakan sebuah hal yang powerful. Dengan menggunakan berbagai properti yang dimiliki oleh matriks, kita dapat

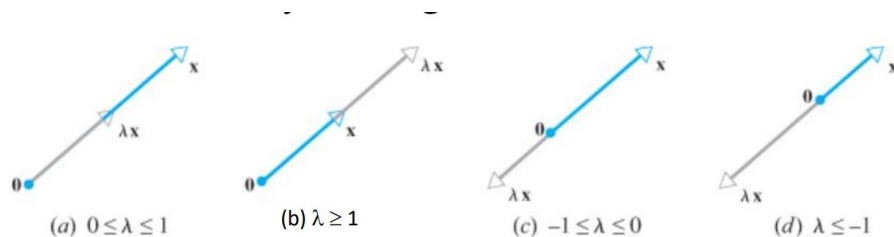
melakukan pemrosesan gambar, seperti image scaling dengan menggunakan *bicubic interpolation* dan pengenalan wajah pada training image dengan menggunakan *eigenface*.

2.2. Nilai Eigen dan Vektor Eigen

Jika A merupakan sebuah matriks berukuran $n \times n$, maka vektor tidak nol \mathbf{x} disebut vektor eigen dari A jika $A\mathbf{x}$ sama dengan dengan perkalian suatu skalar dengan \mathbf{x} sesuai persamaan,

$$A\mathbf{x} = \lambda\mathbf{x}$$

Skalar λ disebut sebagai nilai eigen dari A , dan \mathbf{x} disebut sebagai vektor eigen yang berkoresponden dengan λ . Dikarenakan vektor eigen dapat dinyatakan sebagai perkalian skalar λ , representasi vektor dari $A\mathbf{x}$ merupakan vektor \mathbf{x} yang memanjang dan memendek dalam arah yang sama.



Gambar 2. Representasi vektor eigen dalam koordinat Cartesian

Misalkan A merupakan matriks berukuran $n \times n$ maka, nilai eigen dan vektor eigen dari matriks A dapat dihitung sebagai berikut,

$$A\mathbf{x} = \lambda\mathbf{x}$$

$$IA\mathbf{x} = \lambda I\mathbf{x}$$

$$(\lambda I - A)\mathbf{x} = 0$$

$\mathbf{x} = 0$ merupakan solusi trivial dari $(\lambda I - A)\mathbf{x} = 0$. Agar $(\lambda I - A)\mathbf{x}$ memiliki solusi tidak-nol, haruslah,

$$\det(\lambda I - A) = 0$$

Persamaan $\det(\lambda I - A) = 0$ disebut persamaan karakteristik dari matriks A dan akar-akar persamaan tersebut, yaitu λ , dinamakan akar-akar karakteristik atau nilai-nilai eigen.

2.3. Dekomposisi QR

$$\begin{array}{c} \mathbf{A} \\ \left[\begin{array}{c|c|c} | & | & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \\ | & | & | \end{array} \right] \end{array} = \begin{array}{c} \mathbf{Q} \\ \left[\begin{array}{c|c|c} | & | & | \\ \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ | & | & | \end{array} \right] \end{array} \begin{array}{c} \mathbf{R} \\ \left[\begin{array}{ccc} \mathbf{e}_1^T \cdot \mathbf{a}_1 & \mathbf{e}_1^T \cdot \mathbf{a}_2 & \mathbf{e}_1^T \cdot \mathbf{a}_3 \\ 0 & \mathbf{e}_2^T \cdot \mathbf{a}_2 & \mathbf{e}_2^T \cdot \mathbf{a}_3 \\ 0 & 0 & \mathbf{e}_3^T \cdot \mathbf{a}_3 \end{array} \right] \end{array}$$

orthogonal unit vector
Upper Diagonal matrix

Gambar 3. Ilustrasi dekomposisi QR

Dekomposisi QR merupakan salah satu teknik mendekomposisi suatu matriks menjadi matriks Q sebagai basis ortonormal dan matriks R berupa matriks segitiga atas. Matriks Q sendiri disusun atas basis-basis ortonormal dari matriks A yang diperoleh dengan proses Gram-Schmidt, sementara itu matriks R merupakan hasil kali dalam antara basis ortonormal dengan vektor-vektor kolom A. Adapun proses Gram-Schmidt adalah sebagai berikut:

$$\begin{array}{ll}
 \mathbf{u}_1 = \mathbf{v}_1, & \mathbf{e}_1 = \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \\
 \mathbf{u}_2 = \mathbf{v}_2 - \text{proj}_{\mathbf{u}_1}(\mathbf{v}_2), & \mathbf{e}_2 = \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \\
 \mathbf{u}_3 = \mathbf{v}_3 - \text{proj}_{\mathbf{u}_1}(\mathbf{v}_3) - \text{proj}_{\mathbf{u}_2}(\mathbf{v}_3), & \mathbf{e}_3 = \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \\
 \mathbf{u}_4 = \mathbf{v}_4 - \text{proj}_{\mathbf{u}_1}(\mathbf{v}_4) - \text{proj}_{\mathbf{u}_2}(\mathbf{v}_4) - \text{proj}_{\mathbf{u}_3}(\mathbf{v}_4), & \mathbf{e}_4 = \frac{\mathbf{u}_4}{\|\mathbf{u}_4\|} \\
 \vdots & \vdots \\
 \mathbf{u}_k = \mathbf{v}_k - \sum_{j=1}^{k-1} \text{proj}_{\mathbf{u}_j}(\mathbf{v}_k), & \mathbf{e}_k = \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}.
 \end{array}$$

Gambar 4. Proses Gram-Schmidt

Matriks Q yang disusun dari basis ortonormal memiliki keunikan bahwa determinannya akan sama dengan 1, dengan keunikan ini diperoleh metode mencari nilai eigen yang sama dengan matriks A apabila R merupakan matriks segitiga atas.

$$A = QR$$

$$R = Q^T A$$

$$Q^T Q = Q Q^T = I$$

2.4. Dekomposisi QR dengan Algoritma Schwarz-Rutishauser

Ketidakstabilan numerik pada kalkulasi dekomposisi QR konvensional membuat dikembangkanya metode baru yang lebih efisien dan cepat. Algoritma Schwarz-Rutishauser merupakan modifikasi Algoritma QR konvensional yang melibatkan ortogonalisasi basis dengan proses Gram-Schmidt yang diusulkan oleh H. R. Schwarz, H. Rutishauser, dan E. Stiefel dalam papernya berjudul *Numerik symmetrischer Matrizen* (Stuttgart, 1968). Adapun langkah-langkahnya sebagai berikut:

1. Inisialisasi matriks Q sama dengan A dan matriks R yang merupakan matriks 0
2. Matriks Q yang berukuran $m \times n$ memiliki vektor kolom yang membangun ruang kolom Q, untuk tiap vektor kolom ke-k $\mathbf{q}_k \in \mathbf{Q}$, $k=1 \dots n$, lakukan hal berikut:
 - a. Untuk vektor $\mathbf{q}_i \in \mathbf{Q}(k)$, $i=1..k$, dapatkan elemen matriks $R[i][k]$ dengan persamaan: $r_{i,k} = \langle \vec{q}_i^T, \vec{q}_k \rangle$ (lakukan hasil kali dalam \mathbf{q}_i^T dan \mathbf{q}_k), lalu perbarui kolom ke-k dari q dengan persamaan $\vec{q}_k = \vec{q}_k - r_{i,k} * \vec{q}_i$
 - b. Lakukan iterasi tersebut sebanyak k kali
3. Perbarui elemen $R[k][k]$ dengan mengambil nilai norm dari vektor kolom \mathbf{q}_k ($r_{k,k} = \|\vec{q}_k\|$)
4. Normalisasi vektor kolom ke-k $\mathbf{q}_k \in \mathbf{Q}$, dengan normnya itu sendiri (atau dibagi dengan elemen $R[k][k]$) untuk memperoleh vektor yang membangun basis ortonormal
5. Iterasi langkah 1-4 sebanyak n kali, hingga terbentuk Basis ortonormal vektor Q beserta matriks segitiga atas R
6. Setelah selesai, mengembalikan nilai negatif dari Q dan R(-Q dan -R).

2.5. Tahapan Training (Algoritma Eigenface)

1. Langkah pertama adalah menyiapkan data dengan membuat suatu himpunan S yang terdiri dari seluruh training image, $(\Gamma_1, \Gamma_2, \dots, \Gamma_M)$
$$S = (\Gamma_1, \Gamma_2, \dots, \Gamma_M)$$
2. Langkah kedua adalah ambil nilai rata-rata atau mean (Ψ)

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

3. Langkah ketiga kemudian cari selisih (Φ) antara nilai training image (Γ_i) dengan nilai tengah (Ψ)

$$\phi_i = \Gamma_i - \Psi \quad (3)$$

4. Langkah keempat adalah menghitung nilai matriks kovarian (C)

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

$$L = A^T A \quad L = \phi_m^T \phi_n$$

5. Langkah kelima menghitung eigenvalue (λ) dan eigenvector (v) dari matriks kovarian (C)

$$C \times v_i = \lambda_i \times v_i$$

6. Langkah keenam, setelah eigenvector (v) diperoleh, maka eigenface (μ) dapat dicari dengan:

$$\mu_l = \sum_{k=1}^M v_{lk} \phi_k$$

$$l = 1, \dots, M$$

7. Langkah ketujuh, setelah mendapatkan eigenface(μ), tiap citra yang sudah dinormalisasi merupakan kombinasi linear dari eigenface yang ada.

$$\Omega = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_k \end{bmatrix}$$

Dimana bobot diperoleh dengan mengalikan matriks citra dengan eigenface.

BAB III

IMPLEMENTASI PROGRAM

3.1.Pustaka/Kakas

3.1.1. NumPy



Gambar 5. NumPy

NumPy merupakan sebuah pustaka untuk bahasa pemrograman Python, dimana NumPy memberikan dukungan untuk himpunan dan matriks multidimensi yang besar, dan dilengkapi koleksi sejumlah besar fungsi matematika tingkat tinggi untuk beroperasi pada himpunan ini.

Numpy mengusung konsep paralisme, dimana komputasi array pada NumPy sendiri jauh lebih cepat daripada fitur konvensional pada Python. Selain itu, Numpy sendiri terintegrasi dengan bahasa-bahasa pemrograman lain seperti C, C++, dan Fortran.

3.1.2. OpenCV



Gambar 6. NumPy

OpenCV merupakan pustaka pengolahan citra digital yang tersedia dalam bahasa C++ maupun Python. Di dalam OpenCV sudah mempunyai banyak fitur, antara lain : pengenalan wajah, pelacakan wajah, deteksi wajah, Kalman filtering, dan berbagai jenis metode kecerdasan buatan. Selain itu, OpenCV juga menyediakan berbagai algoritma sederhana terkait *Computer Vision* untuk *low level API*.

3.1.3. PIL



Gambar 7. PIL

PIL adalah library open-source tambahan untuk Python yang fungsi utamanya adalah memanipulasi file gambar. PIL diciptakan oleh Fredrik Lundh pada tahun 1995, dan pengembangannya dihentikan pada tahun 2011. PIL di-fork dan diteruskan oleh library Pillow.

Pillow mendukung banyak format file populer, misalnya PNG, JPG/JPEG, TIFF, dan BMP. Jika perlu, Pillow dan Python mendukung library decoder tambahan. Tipe manipulasi antara lain masking, filtering, enhancement, menambahkan teks, manipulasi per pixel, dan lain-lain.

3.1.4. Tkinter



Gambar 8. TKINTER

Tkinter adalah pustaka standar GUI untuk Python. Dengan Python dijalankan bersamaan dengan Tkinter, kita dapat membuat sebuah aplikasi dengan GUI dengan cepat dan mudah. Tkinter merupakan sebuah alat yang powerful untuk membuat interface berorientasi objek. Membuat sebuah aplikasi dengan GUI menggunakan Tkinter hanya memerlukan beberapa langkah,

1. Melakukan import modul Tkinter pada main program
2. Membuat GUI aplikasi dengan menambahkan widget-widget tkinter
3. Melakukan event-handling pada aplikasi

3.1.5. Custom Tkinter

CustomTkinter adalah sebuah pustaka UI-Python yang dibangun dari Tkinter. CustomTkinter menawarkan antarmuka yang baru, modern, dan widget yang fully customizable. Widget di dalam CustomTkinter dibuat dengan cara yang sama dengan normal Tkinter, namun juga dapat digunakan sebagai kombinasi dari beberapa widget Tkinter. Dengan CustomTkinter, kita dapat mendapatkan antarmuka yang konsisten dengan rupa yang modern di seluruh platform sistem operasi (Windows, MacOS, Linux).

3.2. Algoritma Input File dan Kompresi

Kami menggunakan pustaka Zipfile untuk melakukan ekstraksi dataset uji yang dikemas pada format zip dan menggunakan pustaka OpenCV dan PIL untuk mengolah kumpulan citra uji(JPG/PNG) yang dikemas dalam file zip tersebut. Citra yang kami uji

sebagai model pengenalan citra dengan eigenface akan diubah ukurannya menjadi 256x256 piksel hitam-putih. Sertelah itu, kami akan ubah menjadi larik $256^2 \times 1$, kemudian akan diijajarkan sebanyak M buah(sesuai banyak data citra uji) yang dikemas dalam bentuk matriks. Berikut adalah implementasi kode kami:

```
def get_img(img):
    dim = (256, 256)
    image = cv2.imread(img)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    resized = cv2.resize(gray, dim, interpolation =
cv2.INTER_AREA)
    return resized.reshape(256*256)

def get_img_PIL(img):
    theImg = norm_img(np.array(Image.open(img)))
    return theImg
```

```
def norm_img(image):
    dim = (256,256)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    resized = cv2.resize(gray, dim, interpolation =
cv2.INTER_AREA)
    return resized.reshape(256*256)
```

3.3. Algoritma Pemrosesan *Training Image* menjadi Eigenface

3.3.1 Dekomposisi QR dengan Algoritma Schwarz-Rutishauser

Ketidakstabilan algoritma QR konvensional membuat kami mengimplementasikan Algoritma Schwarz-Rutishauser. Adapun implementasi kode kami adalah sebagai berikut (disertai fungsi lain untuk menghitung norm_vector, proj_vector, cosine_similarity)

```
def QR_decomposition(M, type='float64'):

    M = np.array(M, dtype=type)
    (m,n) = np.shape(M)

    Q = np.array(M, dtype=type)
    R = np.zeros((n, n), dtype=type) #langkah 1

    for k in range(n):
```

```

for i in range(k):
    R[i,k] = np.transpose(Q[:,i]).dot(Q[:,k])
    Q[:,k] = Q[:,k] - R[i,k] * Q[:,i]

R[k,k] = norm_vector(Q[:,k])
Q[:,k] = Q[:,k] / R[k,k]

return -Q, -R

```

```

def proj(u, v):
    v_norm_squared = sum(v**2)

    proj_of_u_on_v = (np.dot(u, v)/v_norm_squared)*v
    return proj_of_u_on_v

def norm_vector(v):
    return np.sqrt(np.sum(np.power(v,2)))
    #JIKA PAKAI FOR LOOP, TIDAK EFISIEN WAKTU

def cosine_sim(a,b):
    return abs(np.dot(a,b)/(norm_vector(a)*norm_vector(b)))

```

3.3.2 Kalkulasi Vektor Eigen dan Eigenface

Alih-alih menghitung kovarians dari matriks berukuran $256^2 \times 256^2$, kita dapat menghitung kovarians matriks dari matriks berukuran $M \times M$, dengan M adalah jumlah citra latih(seperti pada bagian 2.5 Tahapan Training) . Kami memanfaatkan algoritma QR untuk mendekomposisi matriks kofaktor sehingga akan diperoleh matriks segitiga atas yang memiliki determinan sama dengan matriks kofaktor sehingga memiliki nilai eigen yang sama(dan juga vektor eigennya). Vektor eigen dapat diperoleh dari matriks Q yang diproses melalui algoritma QR.

Berikut adalah kode utama kami(mengambil 75% dari himpunan vektor eigen):

```

def training_parameters(training):
    #Menghitung kovarians dari data training
    avg = avg_image(training)
    A = training - avg
    kov = covariance(A)

    #menghitung eigenface dan weight
    eigval, eigvec = theEigen(kov,100)

```

```

        eigval, eigenvect = eigenSort(eigval,eigenvect)
        eigenvect = np.array([k/norm_vector(k) for k in eigenvect])
#normalisasi vektor eigen
        x = int(training.shape[0]*0.75)
        eigfaces = np.array(get_eigenfaces(eigenvect,x,training))
#75% eigenface pertama
        weight_training = get_weight(eigfaces,A)

        return eigfaces, weight_training, avg

```

Berikut kalkulasi nilai eigen dan vektor eigen dengan Algoritma QR:

```

def theEigen(M,iterasi):
    vec = np.identity(M.shape[0])
    Y = np.array(M)
    for i in range(iterasi):
        Q, R = QR_decomposition(Y)
        Y = R @ Q
        vec = vec @ Q

    return Y.diagonal(),vec

```

Dan ini tahapan mencari M pertama eigenface

```

def get_eigenfaces(eigenvectors,n,training_set):
    reduced_data = np.array(eigenvectors[:n]).T
    eigenface = np.dot(training_set.T,reduced_data)
    return eigenface.T

```

3.4. Tahapan Uji Gambar

Tahapan testing File dimulai dengan mengambil gambar. Implementasi kode pada main.py. Gambar diolah oleh opencv dan numpy agar menjadi citra berukuran 256x256(black and white) untuk diolah kemudian dan yang akan ditampilkan pada aplikasi dalam ukuran 512*512. Berikut implementasi kodenya:

```

def openTestImage(self):
    self.state = False
    self.label_info_2.configure(image=self.imageUNDEF)
    self.label_4.configure(text = "PRESS START")
    self.label_5.configure(text = "")
    self.label_infot3.configure(text = "Execution time: ")
    filetypes = (

```

```

        ('Image Files', ['*.jpg', '*.png']),
        ('JPG Files', '*.jpg'),
        ('PNG Files', '*.png')
    )
    filename = fd.askopenfilename(
        title='Open Test Image',
        initialdir='/',
        filetypes=filetypes
    )

    if filename :
        self.imageTest =
ImageTk.PhotoImage((Image.open(filename)).resize((512, 512),
Image.ANTIALIAS))
        self.label_info_1.configure(image=self.imageTest)
        self.imageTestGrayscale = get_img_PIL(filename)
        filenameshow = ""
        lenname = len(filename)-1
        while (filename[lenname] != '/'):
            filenameshow = filename[lenname] + filenameshow
            lenname = lenname-1
        self.label_2i.configure(text=filenameshow)

```

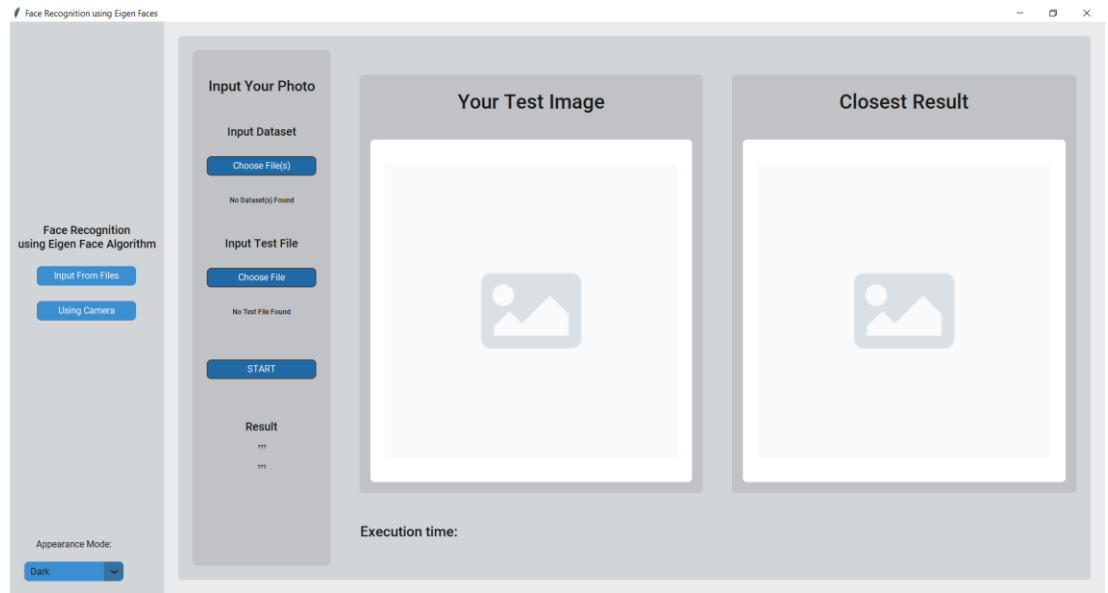
Kemudian, citra kami yang berukuran 256x256.

BAB IV

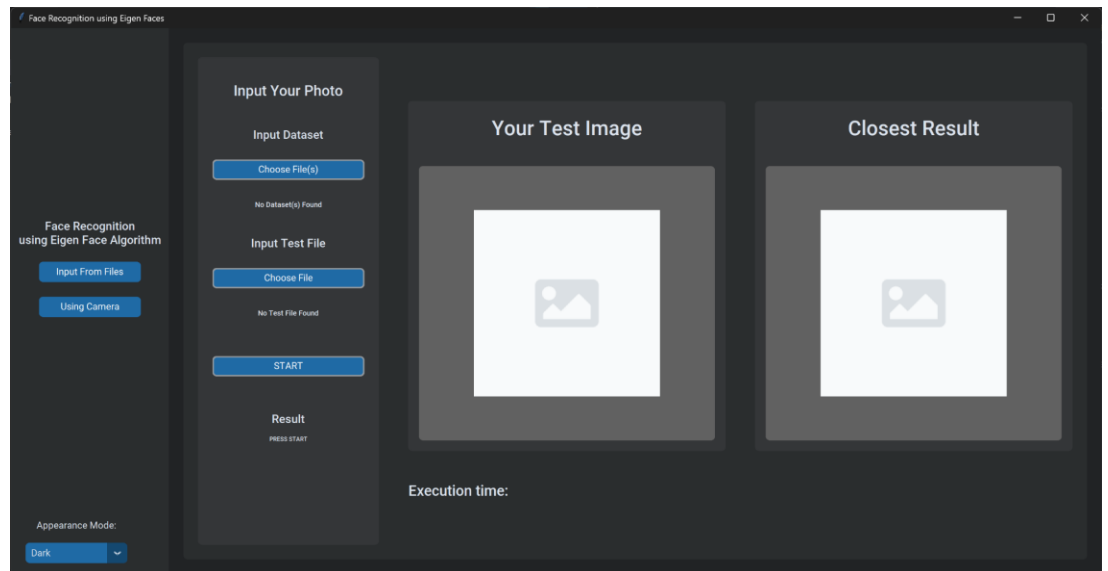
EKSPERIMEN

1. Interface

a. Normal mode

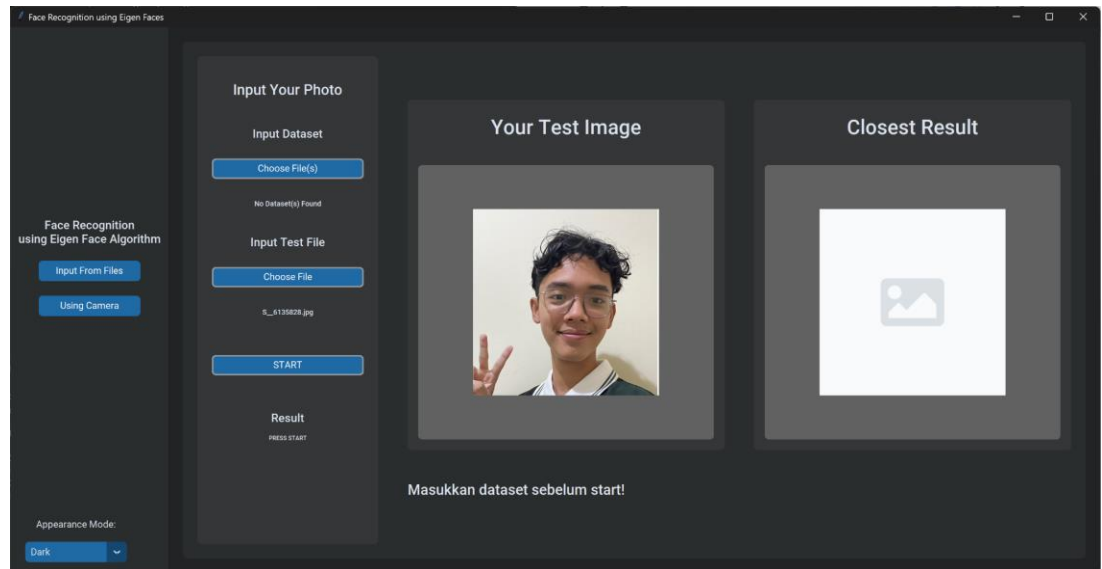


b. Dark mode

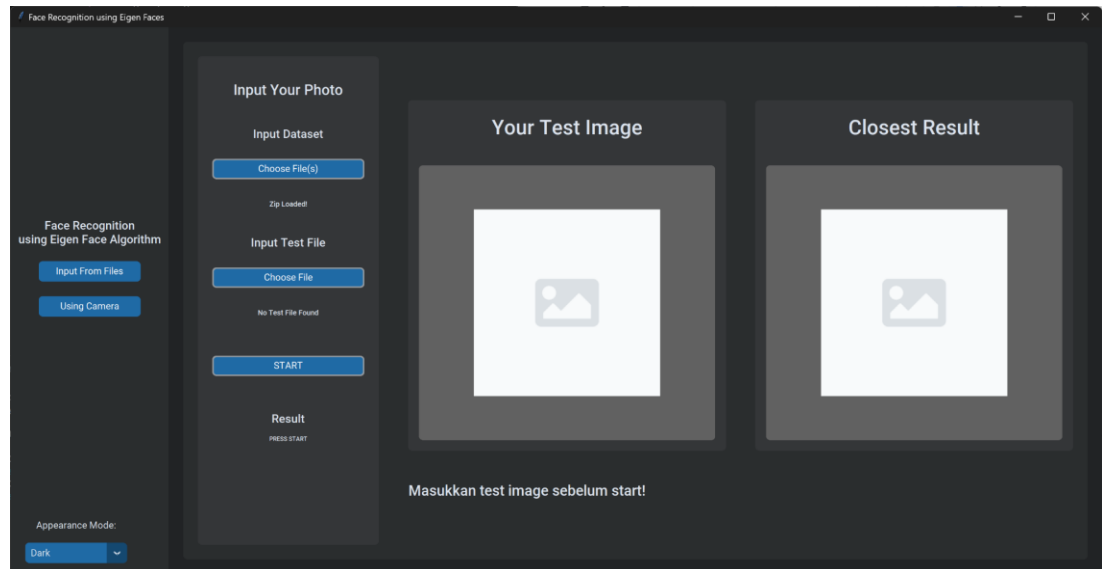


2. Testing Error Handling

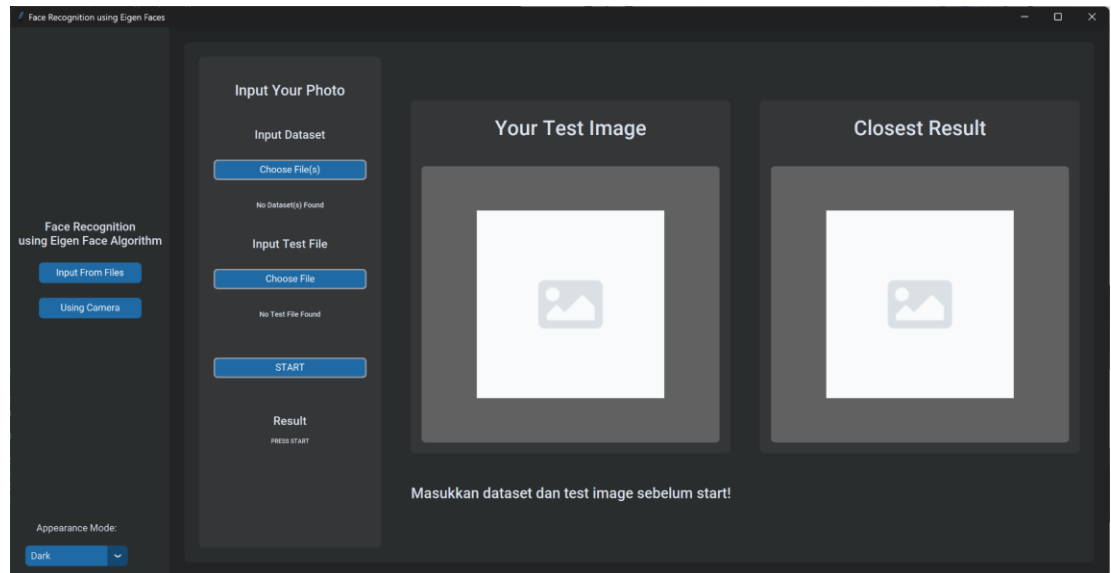
a. START tanpa *dataset*



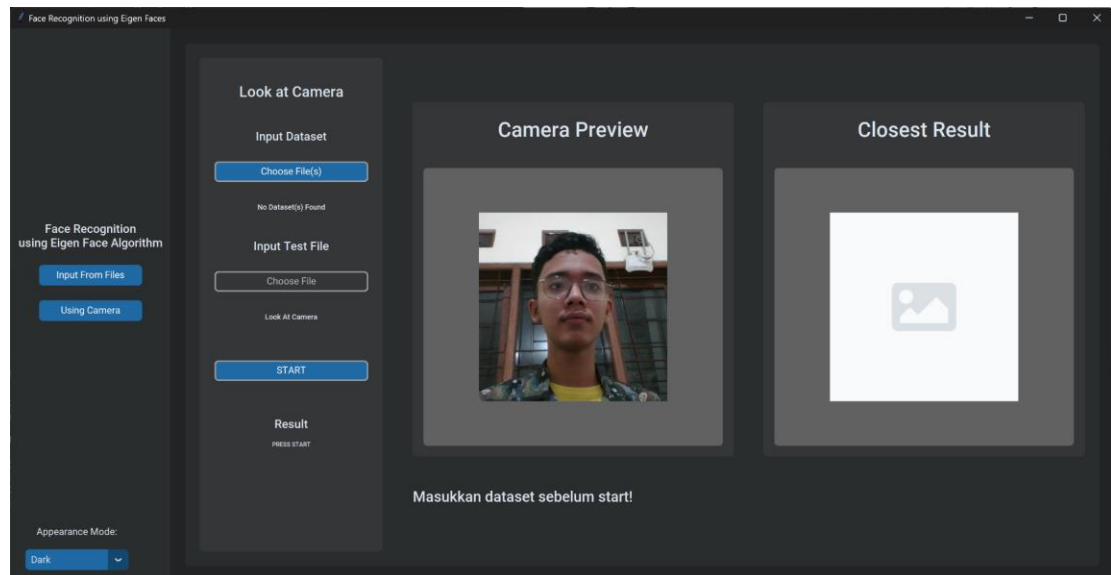
b. START tanpa *test-image*



c. START tanpa *dataset* dan *test-image*



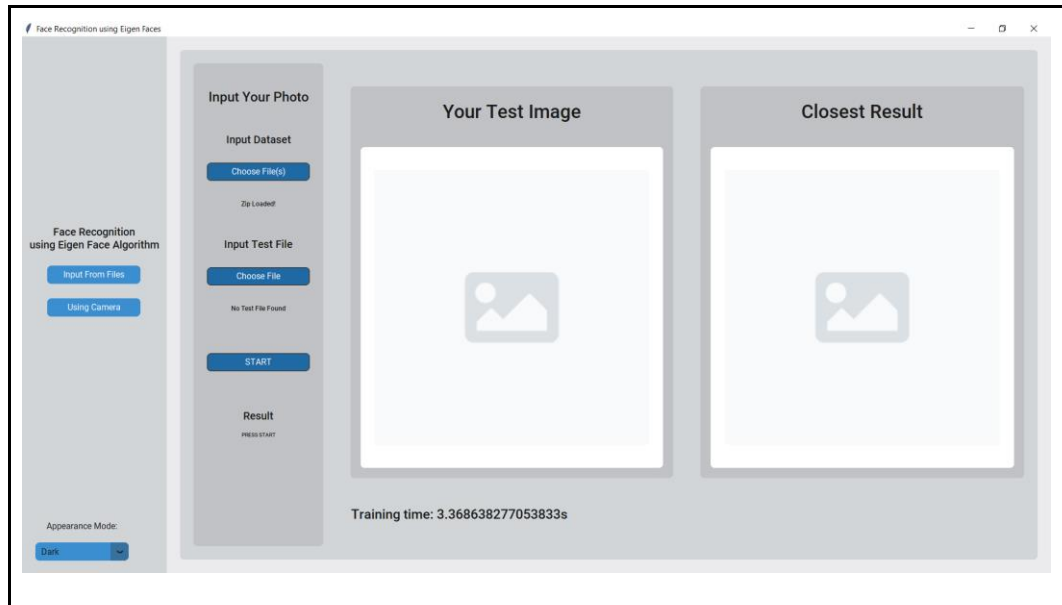
d. Testing tanpa *dataset* (pada input camera)



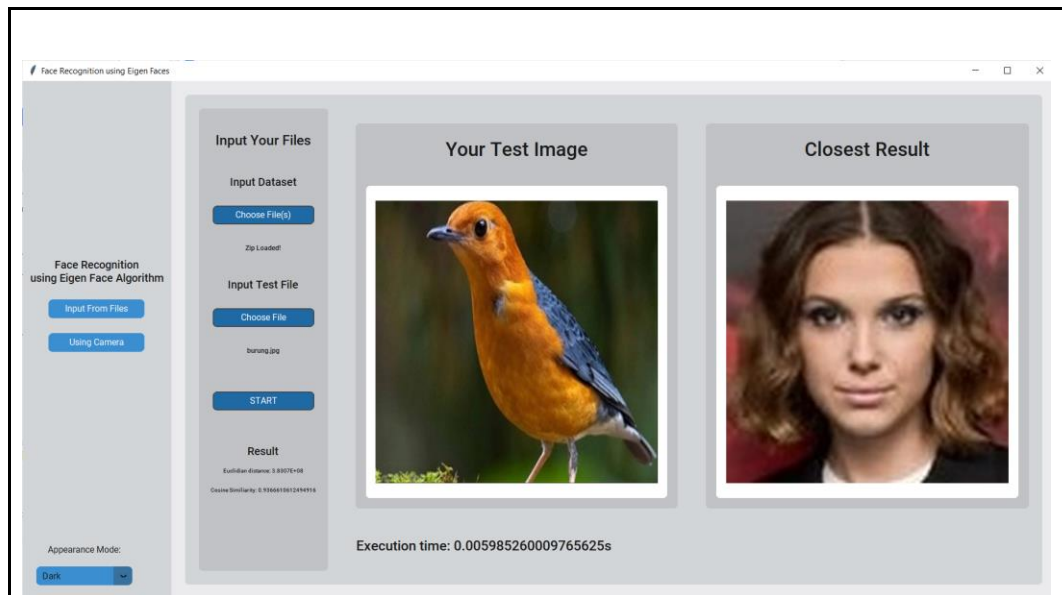
3. Testing DataSet Kaggle

a. Identitas Dataset

Banyaknya Dataset	104 gambar
Banyak Dekomposisi QR	100x
Training Time	3.37s

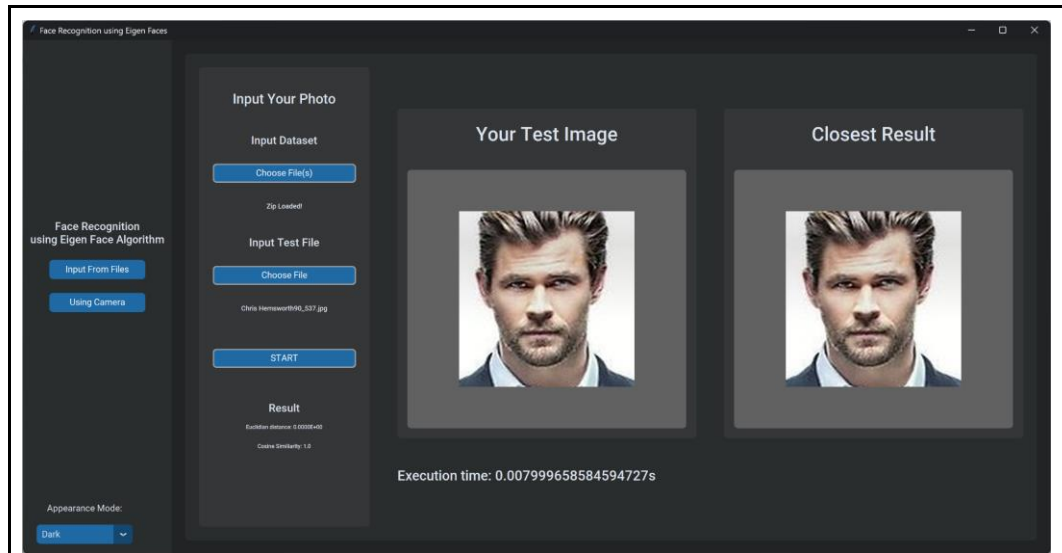


b. Testing 1 (Gambar sembarang)



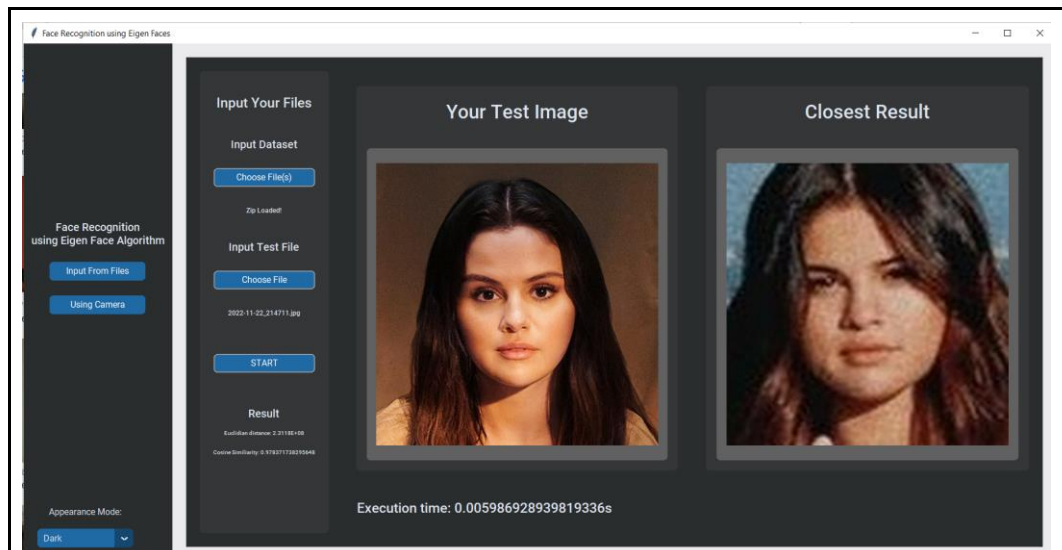
Test Gambar	Burung
Hasil Deteksi	Millie Bobby Brown
Euclidian Distance	3.08E8
Waktu Eksekusi	0.00699 s
<i>Weight Cosine Similarity</i>	0.97055

c. Testing 2 (Gambar yang terdapat di dalam dataset)



Test Gambar	Chris Hemsworth
Hasil Deteksi	Chris Hemsworth
Euclidian Distance	0
Waktu Eksekusi	0.00799 s
<i>Weight Cosine Similarity</i>	1

d. Testing 3



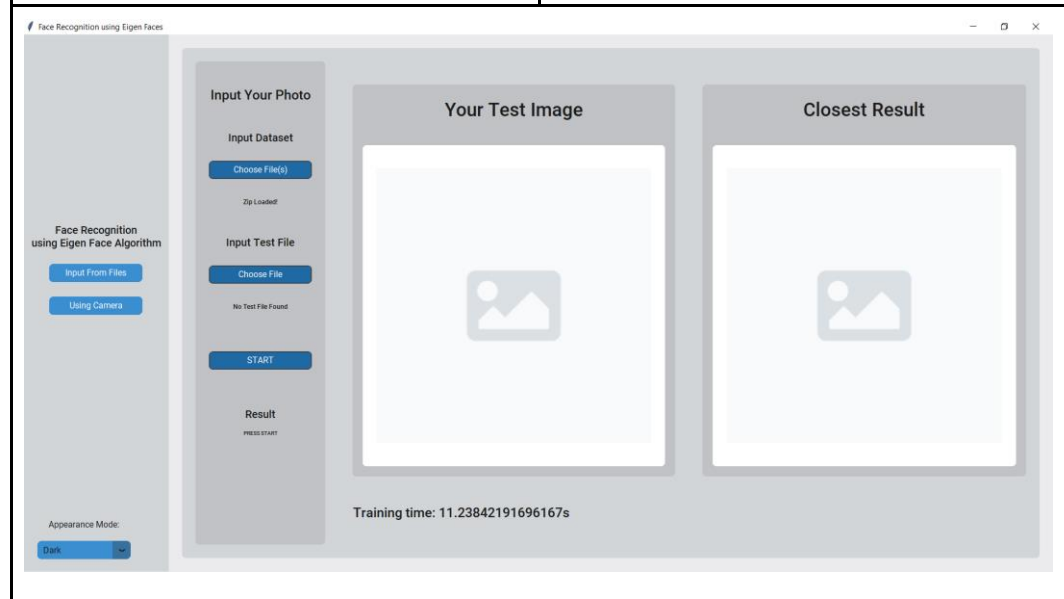
Nama Gambar	Selena Gomez
Waktu Eksekusi	0.00599

Euclidian distance	2.3E8
<i>Weight Cosine Similarity</i>	0.97

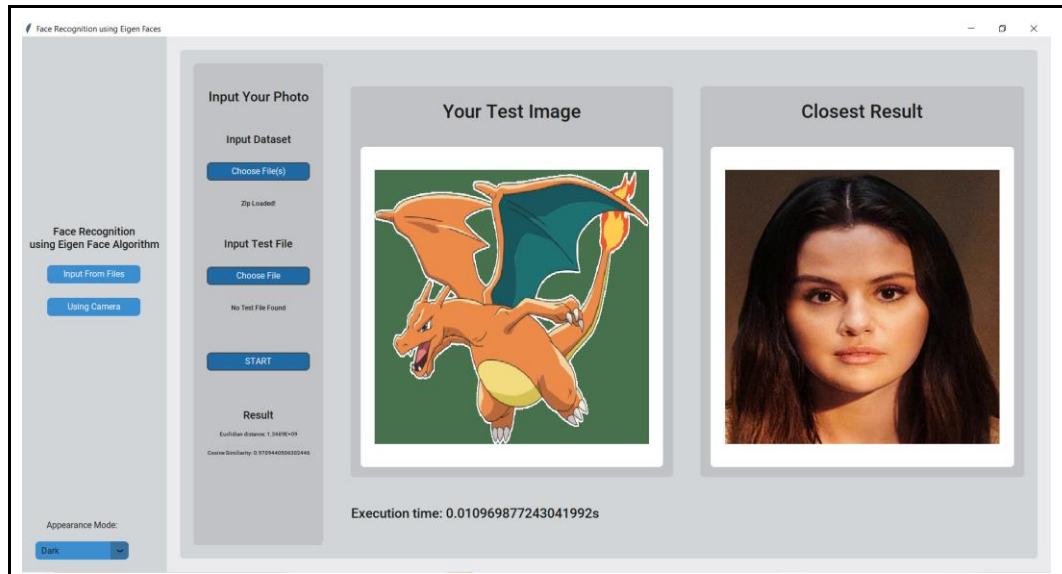
4. Testing DataSet Kaggle

a. Identitas Dataset

Banyaknya Dataset	221 gambar
Banyak Dekomposisi QR	60x
Training Time	11.2s

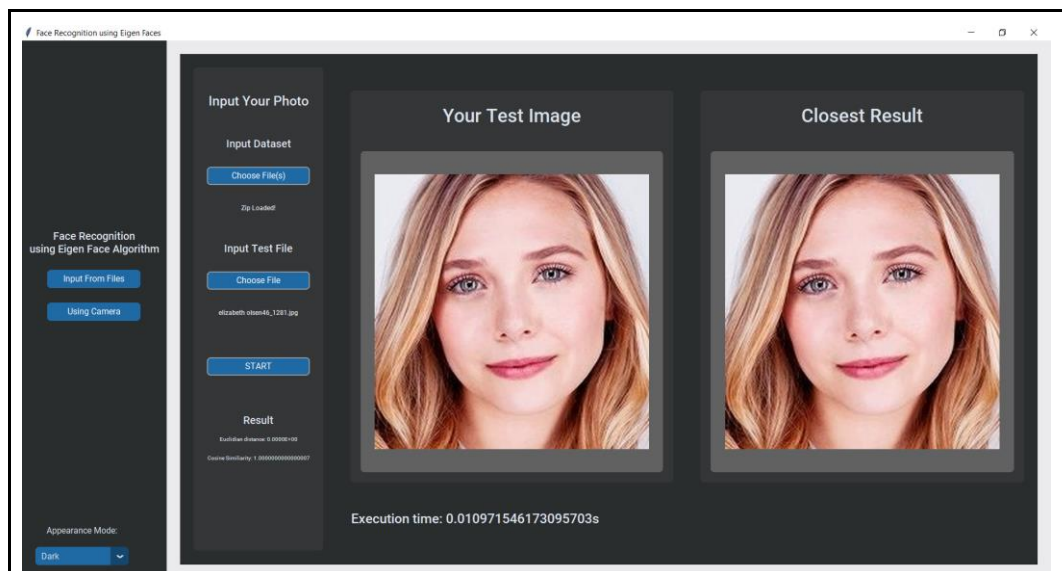


b. Testing 1 (Gambar sembarang)



Test Gambar	Pokemon
Hasil Deteksi	Selena Gomez
Euclidian Distance	1.34E9
Waktu Eksekusi	0.01s
<i>Weight Cosine Similarity</i>	0.97

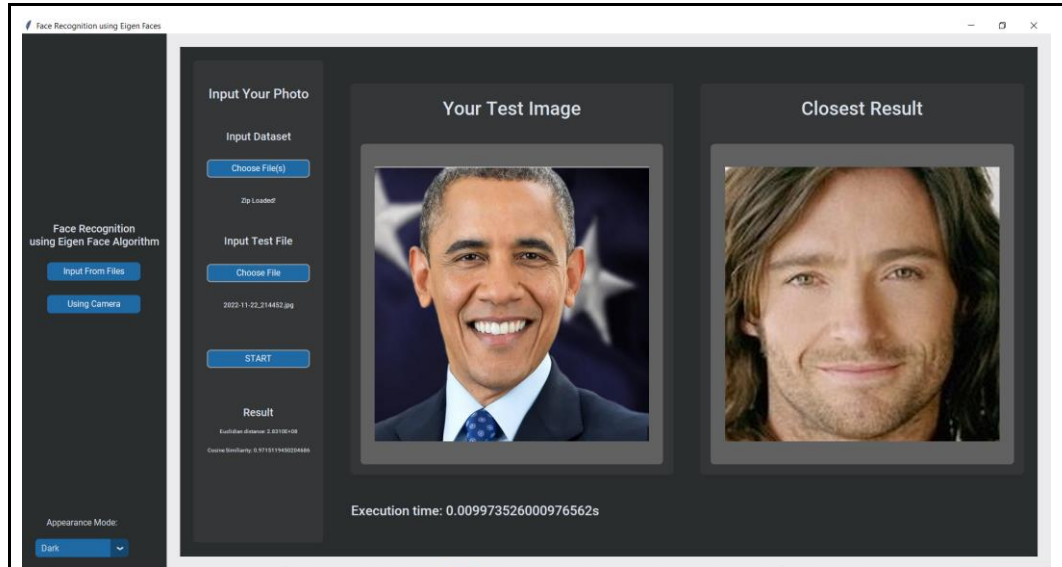
c. Testing 2 (Gambar yang terdapat di dalam dataset)



Test Gambar	Elizabeth Olsen
Hasil Deteksi	Elizabet Olsen

Euclidian Distance	0
Waktu Eksekusi	0.01 s
<i>Weight Cosine Similarity</i>	1

d. Testing 3

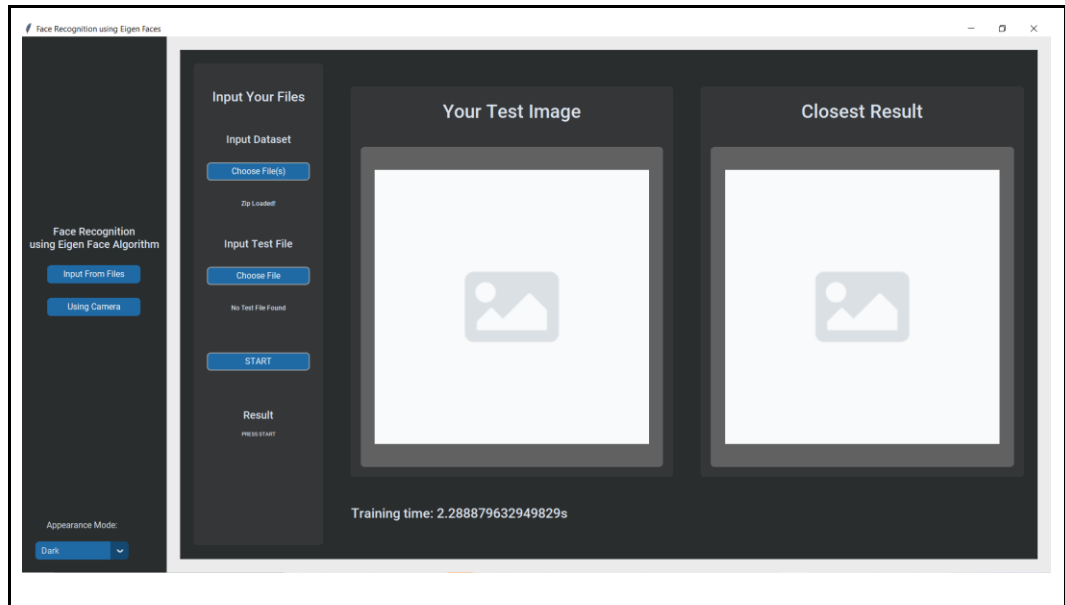


Nama Gambar	Obama
Ukuran Gambar	190,764 bytes
Waktu Eksekusi	0.00599
<i>Weight Cosine Similarity</i>	2.83E8

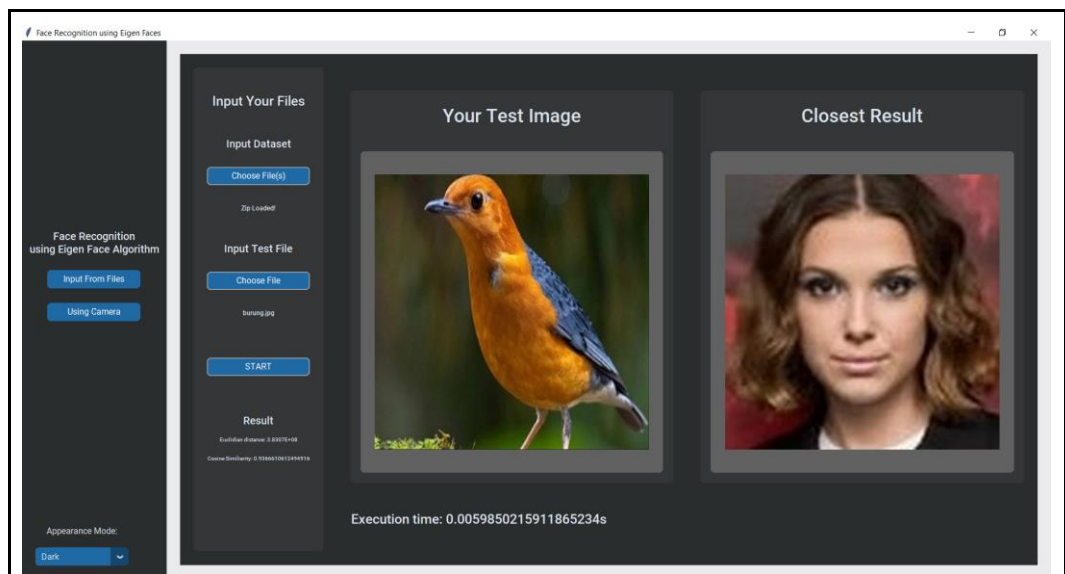
5. Testing DataSet Kaggle

a. Identitas Dataset

Banyaknya Dataset	104 gambar
Banyak Dekomposisi QR	60x
Training Time	2.28s

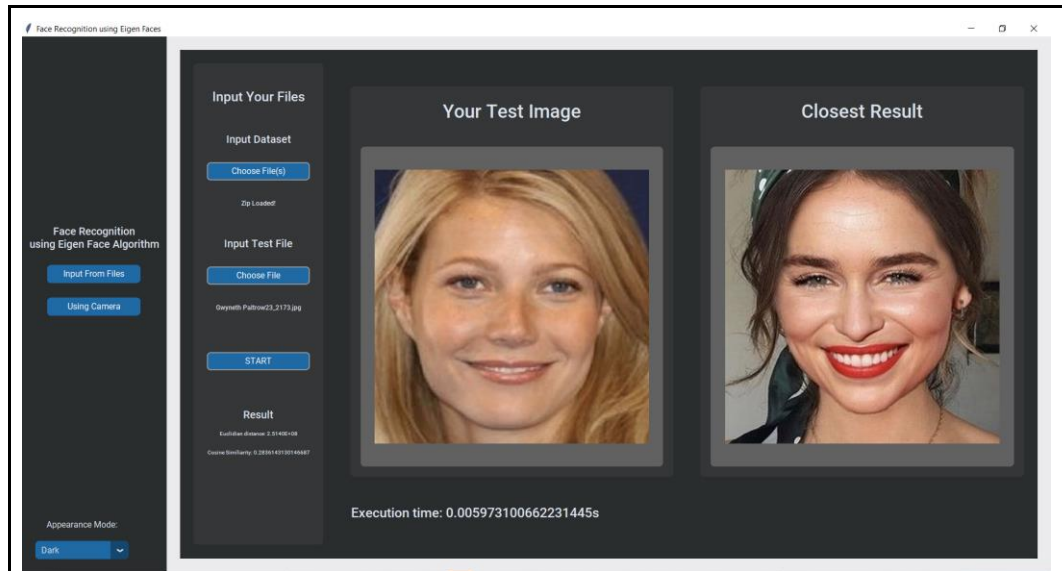


b. Testing 1 (Bukan Wajah)



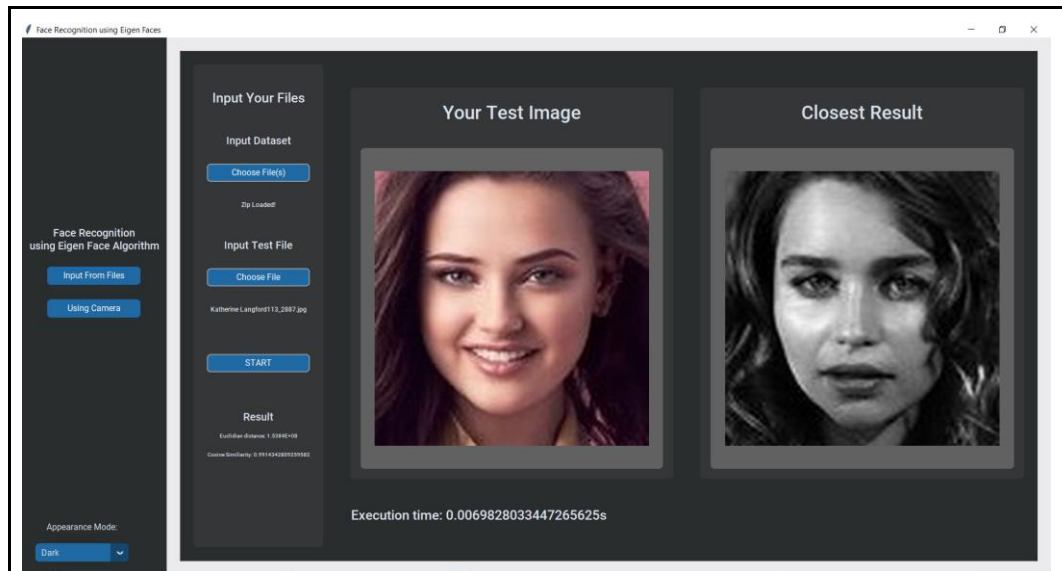
Test Gambar	Burung
Hasil Deteksi	Millie Bobby Brown
Euclidian Distance	3.8E8
Waktu Eksekusi	0.005s
<i>Weight Cosine Similarity</i>	0.97

c. Testing 2



Test Gambar	Gwyneth Pathrow
Hasil Deteksi	Emilia Clarke
Euclidian Distance	2.51E8
Waktu Eksekusi	0.006 s
<i>Weight Cosine Similarity</i>	0.28

d. Testing 3



Nama Gambar	Katherine Langford
Hasil Deteksi	Emilia Clarke

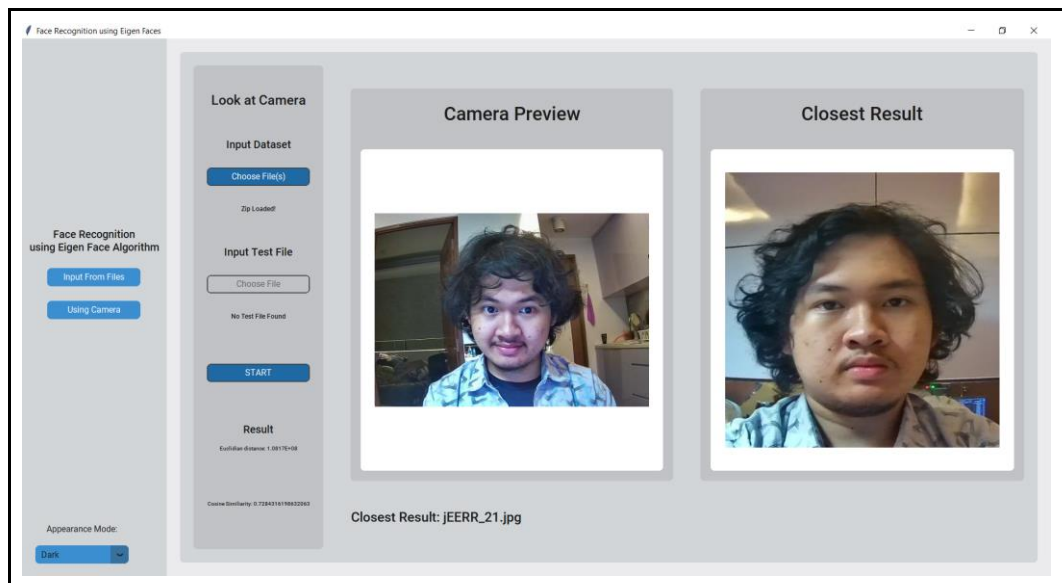
Waktu Eksekusi	0.006
<i>Weight Cosine Similarity</i>	1.53E8

6. Testing DataSet dengan Input Kamera

a. Identitas Dataset

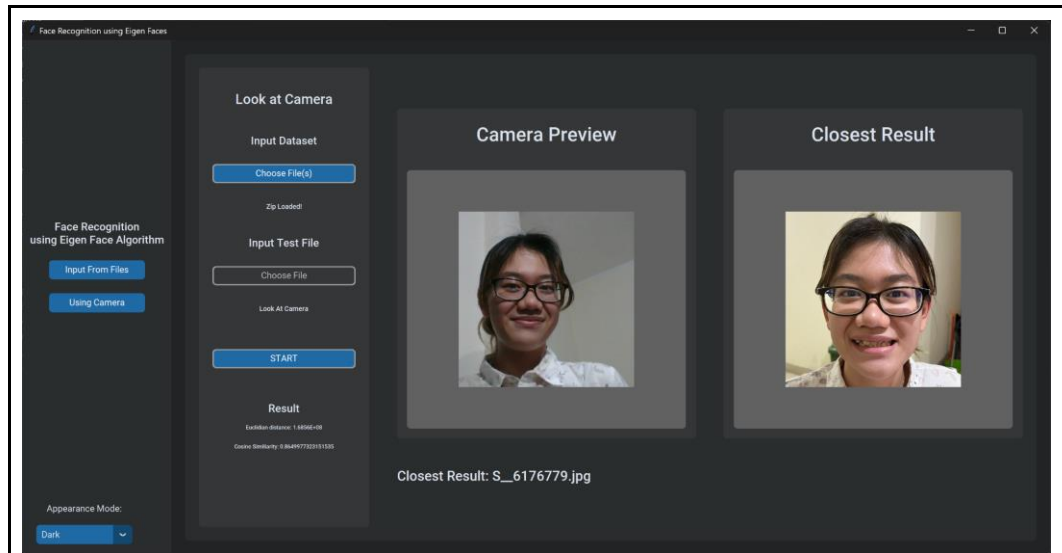
Banyaknya Dataset	50 (terdiri atas 8 orang)
-------------------	---------------------------

b. Testing 1



Gambar Testing	Jerry
Gambar yang sesuai	jEERR_21.jpg
Euclidian Distance	1.081E7
<i>Weight Cosine Similarity</i>	0.73

c. Testing 2



Gambar Testing	Cathleen
Ukuran Gambar	S_6176779.jpg
Euclidian Distance	1.6856E8
<i>Weight Cosine Similarity</i>	0.86

7. Rangkuman *Testing*

a. Pengaruh Banyaknya iterasi QR dengan waktu *training*

100x	3.37 s
60x	2.25 s

Banyaknya dekomposisi QR sangat berdampak pada waktu training. Namun, perbandingan kedua waktu training berbanding lurus. Waktu yang dibutuhkan program untuk melakukan satu kali dekomposisi QR berada di angka 0,37 s.

b. Pengaruh waktu test Image dengan banyaknya database

104x	2.25 s
221x	11.2 s

Dengan jumlah iterasi QR yang sama(60 kali), terdapat hubungan yang tidak linear antara jumlah dataset training dan waktu training.

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

Tugas besar ini sangat membantu kami dalam mengeksplorasi bidang *image processing*. Tugas besar *Face Recognition* dengan *Eigenface* ini ditulis dengan menggunakan bahasa Python dengan berbagai library. Tugas ini sangat membantu kami mengeksplor library Python yang selama ini belum pernah kami gunakan. Oleh karena itu, kami tentunya sangat bangga terhadap diri kami karena telah menyelesaikan seluruh spesifikasi tugas yang diberikan dengan sangat baik, termasuk spesifikasi bonus.

Seluruh fitur dalam program ini sudah berjalan dengan baik. Tampilan antarmuka aplikasi kami memiliki rupa yang modern berkat library CustomTkinter. Kami senang dapat melakukan eksplorasi pembuatan GUI hingga memiliki tampilan seperti tertera pada bagian sebelumnya. Namun, dengan tugas besar ini, kami menyadari bahwa algoritma *eigenface* bukanlah merupakan sebuah algoritma yang memiliki tingkat keakuratan yang tinggi. Dibandingkan dengan sistem pengenalan wajah yang ada saat ini, seperti yang paling populer Face ID dari Apple dan pembelajaran mesin untuk rekognisi citra berbasis jaringan saraf tiruan, algoritma eigenface dapat dikatakan kalah saing.

Adapun secara kuantitatif, euclidian distance berbanding lurus dengan jumlah dataset yang ditraining. Namun, apabila suatu citra bukan wajah, terdapat nilai euclidian distance yang lebih tinggi daripada yang bukan wajah, sehingga citra perlu diberikan batasan tertentu sesuai dengan dataset trainingnya. Kami menyadari bahwa metode pengenalan wajah dengan eigenface sangat sensitif terhadap postur muka, kondisi lingkungan, dan pencahayaan.

Bagaimanapun, tugas ini sangat membantu kami dalam memahami lebih jauh terkait materi kuliah IF2123 - Aljabar Linier dan Geometri, terutama terkait nilai eigen dan vektor eigen, dekomposisi matriks, dan pengenalan terhadap bidang *image processing*. Tugas ini dapat memberikan pengalaman eksplorasi yang menyenangkan kepada kami karena kami sangat menikmati proses pembuatan aplikasi ini dari awal hingga selesai.

Akhir kata, sekali lagi kami bangga mempersembahkan tugas besar ini yang telah kami kerjakan selama 3 minggu terakhir ini. Pengerjaan tugas besar ini merupakan salah satu yang paling menyenangkan diantara semua tugas besar lainnya. Terima kasih kepada Yobel, Jerry dan

Nate beserta Bapak Rinaldi dan para asisten Aljabar Linier dan Geometri Tahun 2022 yang telah memberikan kami pengalaman tubes yang lumayan menyenangkan.

DAFTAR REFERENSI

Anton, Howard, *Elementary Linear Algebra, 10th edition*, John Wiley and Sons, 2010.

Munir, Rinaldi. *Homepage Rinaldi Munir*. Diakses dari

<https://informatika.stei.itb.ac.id/~rinaldi.munir/>

Ratz, Arthur. *Can QR Decomposition Be Actually Faster? Schwarz-Rutishauser Algorithm*. Diakses dari <https://towardsdatascience.com/can-qr-decomposition-be-actually-faster-schwarz-rutishauser-algorithm-a32c0cde8b9b>

<https://id.wikipedia.org/wiki/NumPy>

<http://elib.mi.sanu.ac.rs/files/journals/ncd/12/ncd12017.pdf>

Shah, Deval. *Face Recognition using eigenfaces technique*. Diakses dari

<https://medium.com/@devalshah1619/face-recognition-using-eigenfaces-technique-f221d505d4f7>

LAMPIRAN

Tautan repository tugas besar: <https://github.com/yobeldc/Algeo02-21067>