

# **PROGRAM FACE RECOGNITION DALAM BAHASA PYTHON**

## **LAPORAN TUGAS BESAR**

**Disusun untuk memenuhi salah satu tugas besar  
mata kuliah Aljabar Linier dan Geometri**

**IF2123-03**

**Oleh**

**Bernardus Willson**

**13521021**

**Raditya Naufal A.**

**13521022**

**Raynard Tanadi**

**13521143**



**Program Studi Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
INSTITUT TEKNOLOGI BANDUNG**

**2022**

## DAFTAR ISI

<b>DAFTAR ISI</b>	<b>1</b>
<b>BAB I</b>	<b>3</b>
<b>BAB II</b>	<b>4</b>
A. Perkalian Matriks	4
B. QR Decomposition	5
C. Nilai Eigen dan Vektor Eigen	6
D. Eigen Face	7
<b>BAB III</b>	<b>8</b>
A. ImgToMtrx	9
B. meanMtrx	9
C. normalized	9
D. covariance	10
E. norm	10
F. QR_Decomposition	10
G. EigenV	11
H. EigenFace	11
I. EFSum	11
J. WData	12
K. WTest	12
<b>BAB IV</b>	<b>13</b>
A. Gambar Tes Terdapat di Dalam Dataset	13
1. Perubahan Ukuran Gambar	13
2. Perubahan Jumlah Dataset	13
B. Gambar Tes Mirip dengan yang Ada di Dalam Dataset	14
1. Perubahan Ukuran Gambar	14
2. Perubahan Jumlah Dataset	15
C. Gambar Tes Tidak Mirip dengan yang Ada di Dalam Dataset	15
1. Perubahan Ukuran Gambar	15
2. Perubahan Jumlah Dataset	16
<b>BAB V</b>	<b>17</b>
A. Kesimpulan	17
B. Saran	17

C. Refleksi	17
<b>REFERENSI</b>	<b>18</b>
<b>LAMPIRAN</b>	<b>19</b>

## **BAB I**

### **DESKRIPSI MASALAH**

Pengenalan wajah (*face recognition*) merupakan sebuah teknologi biometrik yang mampu mengidentifikasi fitur-fitur wajah seseorang untuk melakukan berbagai kepentingan salah satunya keamanan. Sebuah program pengenalan wajah membutuhkan berbagai citra wajah di sebuah database dan dengan menggunakan citra-citra tersebut program pengenalan wajah akan mempelajari bentuk wajah dan akan membandingkan citra wajah baru dengan citra wajah yang dipelajari.

Pada tugas besar 2 IF2123, kami diminta untuk membuat sebuah program pengenalan wajah menggunakan bahasa python. Program pengenalan wajah ini menggunakan algoritma eigenfaces dalam proses pelatihannya dan menggunakan TKinter sebagai tatap muka pengguna. Program ini kemudian akan digunakan untuk mendeteksi dan menemukan wajah masukan user dan kemiripannya dengan citra wajah yang sudah ada di database.

## BAB II

### TEORI SINGKAT

#### A. Perkalian Matriks

Dalam matematika, perkalian matriks adalah suatu operasi biner dari dua matriks yang menghasilkan sebuah matriks. Agar dua matriks dapat dikalikan, banyaknya kolom pada matriks pertama harus sama dengan banyaknya baris pada matriks kedua. Matriks hasil perkalian keduanya, akan memiliki baris sebanyak baris matriks pertama, dan kolom sebanyak kolom matriks kedua. Perkalian matriks A dan B dinyatakan sebagai AB. (Nykamp DQ, "Multiplying matrices and vectors." From *Math Insight*. [http://mathinsight.org/matrix\\_vector\\_multiplication](http://mathinsight.org/matrix_vector_multiplication)).

Jika A adalah matriks berukuran  $m \times n$  dan B adalah matriks berukuran  $n \times p$ , dengan elemen-elemen sebagai berikut,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix}$$

Hasil perkalian kedua matriks tersebut,  $\mathbf{C} = \mathbf{AB}$  adalah sebuah matriks berukuran  $m \times p$ . (Lipschutz, S.; Lipson, M., 2009).

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{pmatrix}$$

Dengan setiap entri pada matriks C didefinisikan sebagai

$$c_{ij} = a_{i1}b_{1j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^n a_{ik}b_{kj},$$

Untuk nilai  $i = 1, \dots, m$  dan nilai  $j = 1, \dots, p$ . Dengan kata lain, entri  $c_{ij}$  adalah hasil yang didapatkan dengan mengalikan secara berpasang-pasangan entri di baris ke-i matriks A dengan entri di kolom ke-j matriks B, lalu menjumlahkan semua hasil perkalian ini. Interpretasi lain dari

proses ini, entri cij adalah hasil perkalian titik baris ke-i matriks A dengan kolom ke-j matriks B. Dengan demikian, AB juga dapat ditulis sebagai

$$C = \begin{pmatrix} a_{11}b_{11} + \cdots + a_{1n}b_{n1} & a_{11}b_{12} + \cdots + a_{1n}b_{n2} & \cdots & a_{11}b_{1p} + \cdots + a_{1n}b_{np} \\ a_{21}b_{11} + \cdots + a_{2n}b_{n1} & a_{21}b_{12} + \cdots + a_{2n}b_{n2} & \cdots & a_{21}b_{1p} + \cdots + a_{2n}b_{np} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{11} + \cdots + a_{mn}b_{n1} & a_{m1}b_{12} + \cdots + a_{mn}b_{n2} & \cdots & a_{m1}b_{1p} + \cdots + a_{mn}b_{np} \end{pmatrix}$$

Hal ini menyebabkan hasil perkalian AB hanya terdefinisi jika dan hanya jika banyaknya kolom di A sama dengan banyaknya baris di B, yang dalam kasus ini sebanyak n. (Nykamp DQ, "Multiplying matrices and vectors." From *Math Insight*. [http://mathinsight.org/matrix\\_vector\\_multiplication](http://mathinsight.org/matrix_vector_multiplication)).

## B. QR Decomposition

Dalam aljabar linier, dekomposisi QR, juga dikenal sebagai faktorisasi QR atau faktorisasi QU, adalah dekomposisi matriks  $A$  menjadi perkalian  $A = QR$  dari matriks ortogonal  $Q$  dan matriks segitiga atas  $R$ . Dekomposisi QR sering digunakan untuk menyelesaikan masalah linear least squares dan merupakan dasar dari algoritma nilai eigen tertentu, yaitu algoritma QR.

(Trefethen, Lloyd N.; Bau, David III, 1997)

Ada beberapa metode untuk benar-benar menghitung dekomposisi QR, seperti salah satunya adalah menggunakan proses Gram–Schmidt. Pertimbangkan proses Gram–Schmidt yang diterapkan pada kolom matriks peringkat kolom penuh  $A = [a_1 \dots a_n]$ , dengan produk dalam  $\langle v, w \rangle = v^T w$ .

$$\text{proj}_{\mathbf{u}} \mathbf{a} = \frac{\langle \mathbf{u}, \mathbf{a} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle} \mathbf{u}$$

$$\begin{aligned} \mathbf{u}_1 &= \mathbf{a}_1, & \mathbf{e}_1 &= \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \\ \mathbf{u}_2 &= \mathbf{a}_2 - \text{proj}_{\mathbf{u}_1} \mathbf{a}_2, & \mathbf{e}_2 &= \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \\ \mathbf{u}_3 &= \mathbf{a}_3 - \text{proj}_{\mathbf{u}_1} \mathbf{a}_3 - \text{proj}_{\mathbf{u}_2} \mathbf{a}_3, & \mathbf{e}_3 &= \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \\ \vdots & & \vdots & \\ \mathbf{u}_k &= \mathbf{a}_k - \sum_{j=1}^{k-1} \text{proj}_{\mathbf{u}_j} \mathbf{a}_k, & \mathbf{e}_k &= \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|} \end{aligned}$$

$$\begin{aligned}
\mathbf{a}_1 &= \langle \mathbf{e}_1, \mathbf{a}_1 \rangle \mathbf{e}_1 \\
\mathbf{a}_2 &= \langle \mathbf{e}_1, \mathbf{a}_2 \rangle \mathbf{e}_1 + \langle \mathbf{e}_2, \mathbf{a}_2 \rangle \mathbf{e}_2 \\
\mathbf{a}_3 &= \langle \mathbf{e}_1, \mathbf{a}_3 \rangle \mathbf{e}_1 + \langle \mathbf{e}_2, \mathbf{a}_3 \rangle \mathbf{e}_2 + \langle \mathbf{e}_3, \mathbf{a}_3 \rangle \mathbf{e}_3 \\
&\vdots \\
\mathbf{a}_k &= \sum_{j=1}^k \langle \mathbf{e}_j, \mathbf{a}_k \rangle \mathbf{e}_j
\end{aligned}$$

di mana  $\langle \mathbf{e}_i, \mathbf{a}_i \rangle = \|\mathbf{u}_i\|$ . Dapat ditulis dalam bentuk matriks:

$$A = QR$$

di mana:

$$Q = [\mathbf{e}_1 \quad \cdots \quad \mathbf{e}_n]$$

dan

$$R = \begin{bmatrix} \langle \mathbf{e}_1, \mathbf{a}_1 \rangle & \langle \mathbf{e}_1, \mathbf{a}_2 \rangle & \langle \mathbf{e}_1, \mathbf{a}_3 \rangle & \cdots & \langle \mathbf{e}_1, \mathbf{a}_n \rangle \\ 0 & \langle \mathbf{e}_2, \mathbf{a}_2 \rangle & \langle \mathbf{e}_2, \mathbf{a}_3 \rangle & \cdots & \langle \mathbf{e}_2, \mathbf{a}_n \rangle \\ 0 & 0 & \langle \mathbf{e}_3, \mathbf{a}_3 \rangle & \cdots & \langle \mathbf{e}_3, \mathbf{a}_n \rangle \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \langle \mathbf{e}_n, \mathbf{a}_n \rangle \end{bmatrix}.$$

### C. Nilai Eigen dan Vektor Eigen

Dalam aljabar linear, vektor eigen (*eigenvector*) atau vektor karakteristik dari suatu matriks berukuran  $n \times n$  adalah vektor tak nol yang hanya mengalami perubahan panjang ketika dikali dengan matriks tersebut. Nilai eigen (*eigenvalue*) yang berasosiasi dengan vektor tersebut, umumnya dilambangkan dengan  $\lambda$ , menyatakan besar perubahan panjang vektor yang terjadi. Secara umum dalam ruang vektor multidimensi, vektor eigen tidak mengalami rotasi ketika ditransformasikan oleh matriks. Hal ini berlaku untuk matriks dengan elemen bilangan real, dan akan mengalami rotasi ketika elemen berupa bilangan kompleks. (Kuttler, K., 2012). Nilai eigen dan vektor eigen berguna dalam proses kalkulasi matriks, yang keduanya diterapkan dalam bidang matematika murni dan matematika terapan, contohnya pada transformasi linear. (Leon, Steven J., 2001).

Diberikan sebuah matriks  $A$  berukuran  $n \times n$ . Vektor eigen dan nilai eigen dari matriks  $A$  dihitung sebagai berikut:

$$\begin{aligned} A\mathbf{x} &= \lambda\mathbf{x} \\ I A\mathbf{x} &= \lambda I\mathbf{x} \quad (\text{kalikan kedua ruas dengan } I = \text{matriks identitas}) \\ A\mathbf{x} &= \lambda I\mathbf{x} \\ (\lambda I - A)\mathbf{x} &= 0 \end{aligned}$$

$\mathbf{x} = 0$  adalah solusi trivial dari  $(\lambda I - A)\mathbf{x} = 0$

Agar  $(\lambda I - A)\mathbf{x} = 0$  memiliki solusi tidak-nol, maka haruslah  $\det(\lambda I - A) = 0$

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

#### D. Eigen Face

Dalam bahasa teori informasi metode ini adalah mengekstrak informasi dari citra yang relevan, mengencodennya seefisien mungkin dan membandingkannya dengan citra wajah yang lain yang telah di encode dengan metode yang sama. Eigenface memecah citra wajah menjadi bagian kecil yang menjadi karakter fitur dari citra yang disebut face space, yaitu bagian penting dari citra wajah yang bisa mewakili ciri-cirinya.

Algoritma eigenface secara keseluruhan cukup sederhana. Training image direpresentasikan dalam sebuah vector flat dan digabungkan bersama-sama menjadi sebuah matriks tunggal, Eigenface dari masing-masing citra kemudian diekstraksi dan disimpan dalam file temporary atau database. Test image yang masuk didefinisikan juga nilai eigenfaces-nya dan dibandingkan dengan eigenfaces dari image dalam database atau file temporary.



### BAB III

#### IMPLEMENTASI PUSTAKA

<b>Nama</b>	<b>Kembalian</b>	<b>Parameter</b>	<b>Deskripsi</b>
ImgToMtrx	Vektor dari foto	Path file image	Mengubah foto RGB menjadi vektor
meanMtrx	Vektor mean	Path folder dataset	Merata-rata semua vektor foto dataset
normalized	Vektor normalized	Path folder dataset	Mencari normal dari foto dataset
covariance	Matrix covariance	Path folder dataset	Mencari covariance dataset
norm	Skalar matrix	Matrix matrix	Mencari normal dari suatu matrix
QR_Decomposition	Matrix Q dan Matrix R	Matrix covariance	Melakukan QR decomposition
EigenV	List eigenvalue dan Matrix eigenvector	Matrix covariance	Mencari nilai eigen dan vektor eigen
EigenFace	List eigenface sejumlah K	Path folder dataset	Menghasilkan K jumlah EigenFace
EFSum	Matrix total EigenFace	Matrix EigenFace	Menjumlahkan matrix EigenFace terpilih
WData	W gambar dataset sebanyak jumlah dataset	Matrix EigenFace dan Path folder dataset	Mendapatkan W dari setiap gambar dataset dengan mengalikan normal setiap foto dengan K jumlah eigenface yang sudah dijumlahkan
WTest	W gambar test	Matrix EigenFace, Path image test, Path folder dataset	Mendapatkan W dari gambar tes dengan mengalikan normal foto tes dengan K

			jumlah eigenface yang sudah dijumlahkan
MinEuclidianDistance	Indeks dan nilai euclidian distance terkecil	W gambar dataset sebanyak jumlah dataset dan W gambar test	Mencari jarak euclidian dari w setiap foto di dataset jika dibandingkan dengan w foto tes. Lalu cari yang terkecil.

#### A. ImgToMtrx

```
def ImgToMtrx(img):
    image = cv2.imread(r"" + img)
    image = cv2.resize(image, (256,256), interpolation = cv2.INTER_AREA)
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    result = gray_image.flatten()
    return result
```

#### B. meanMtrx

```
def meanMtrx(pth):
    mean = [0 for i in range(256*256)]
    c = 0
    dirs = os.listdir(r"" + pth)
    for file in dirs:
        a = ImgToMtrx(pth+"/"+file)
        c+=1
        for i in range(256*256):
            mean[i] += a[i]
    for i in range(256*256):
        mean[i] /= c
    return mean
```

#### C. normalized

```
def normalized(pth):
```

```

mean = meanMtrx(pth)
dirs = os.listdir(r"" + pth)
c = 0
norm = [[0 for i in range(256*256)] for j in range(len(dirs))]
for file in dirs:
    a = ImgToMtrx(pth+"/"+file)
    for i in range(256*256):
        norm[c][i] += (a[i] - mean[i])
    c += 1
return norm

```

#### D. covariance

```

def covariance(pth):
    norm = normalized(pth)
    normT = np.transpose(norm)
    cov = np.matmul(norm, normT)
    return cov

```

#### E. norm

```

def norm(matrix) :
    matrix= np.square(matrix)
    matrix= np.sum(matrix)
    matrix= np.sqrt(matrix)
    return matrix

```

#### F. QR\_Decomposition

```

def QR_Decomposition(cov):
    N, M = cov.shape
    Q = np.empty((N, N))
    u = np.empty((N, N))
    u[:, 0] = cov[:, 0]
    Q[:, 0] = u[:, 0] / norm(u[:, 0])
    for i in range(1, N):
        u[:, i] = cov[:, i]
        for j in range(i):
            u[:, i] -= (cov[:, i] @ Q[:, j]) * Q[:, j]
        Q[:, i] = u[:, i] / norm(u[:, i])
    R = np.zeros((N, M))

```

```

for i in range(N):
    for j in range(i, M):
        R[i, j] = cov[:, j] @ Q[:, i]
return Q, R

```

## G. EigenV

```

def EigenV(cov):
    pQ = np.eye(cov.shape[0])
    for i in range(100):
        Q, R = QR_Decomposition(cov)
        pQ = pQ @ Q
        cov = R @ Q
    return np.diag(cov), pQ

```

## H. EigenFace

```

def EigenFace(pth):
    cov = DataCov.covariance(pth)
    norm = DataCov.normalized(pth)
    EigenVal, EigenVec = Eigen.EigenV(cov)
    EigenFace = np.matmul(EigenVec, norm)
    k = 0
    resultarr = []
    leneigen = round(len(norm)*0.06)
    for i in range(leneigen):
        EigenFaces = EigenFace[i]
        result = [[0 for i in range(256)] for j in range(256)]
        for i in range(256):
            for j in range(256):
                result[i][j] += EigenFaces[k]
                k += 1
        k = 0
        resultarr.append(result)
    return resultarr

```

## I. EFSum

```

def EFSum(EFMatrix):
    EFMatSum = [[0 for i in range(256)] for j in range(256)]
    for i in range(len(EFMatrix)):

```

```

        EFMatSum = np.add(EFMatSum, EFMatrix[i])
    return EFMatSum

```

## J. WData

```

def WData(EFMatrix, pth):
    WTotal = []
    W = [[0 for i in range(256)] for j in range(256)]
    path = r"" + pth
    norm = normalizedD(pth)
    dirs = os.listdir(path)
    for i in range(len(dirs)):
        W[i] = np.matmul(norm[i], EFSum(EFMatrix))
        WTotal.append(W[i])
    return WTotal

```

## K. WTest

```

def WTest(EFMatrix, pthdata, pth):
    W = [[0 for i in range(256)] for j in range(256)]
    norm = normalizedT(pthdata, pth)
    W = np.matmul(norm, EFSum(EFMatrix))
    return W

```

## L. MinEuclideanDistance

```


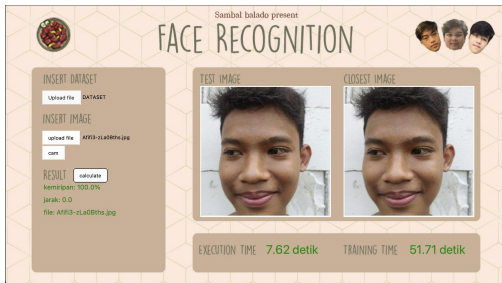
def MinEuclideanDistance(WTest, WData):
    min = 999999999999999999
    selisih = [0 for i in range(len(WTest))]
    for i in range(len(WTest)):
        selisih[i] = np.subtract(WData, WTest[i])
        selisih[i] = np.square(selisih[i])
        selisih[i] = np.sum(selisih[i])
        selisih[i] = np.sqrt(selisih[i])
        if (min > selisih[i]):
            min = selisih[i]
            index = i
    return min, index

```

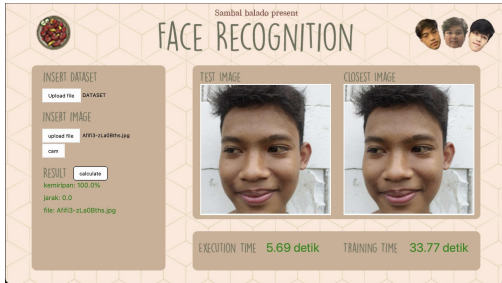
## BAB IV EKSPERIMEN

### A. Gambar Tes Terdapat di Dalam Dataset

#### 1. Perubahan Ukuran Gambar

Hasil Tes	Deskripsi
	<p>Gambar 1 menggunakan folder DATASET dengan setiap gambarnya berukuran 216x216, sedangkan Gambar 2 menggunakan folder DATASET dengan setiap gambarnya berukuran 512x512. Terdapat perubahan signifikan dalam waktu training image dan waktu menghitung euclidean distance yang dibutuhkan, untuk Gambar 1 membutuhkan waktu training 33.77 detik dan execution time 5.69 detik, sedangkan untuk Gambar 2 membutuhkan waktu training 51.71 detik dan execution time 7.62 detik. (dalam kasus ini penambahan ukuran gambar sebanyak 2x menambah waktu eksekusi sebanyak hampir 1.5x). Untuk euclidean distance-nya adalah 0 karena test image ada di dalam dataset sehingga didapat kemiripan 100%.</p>
	

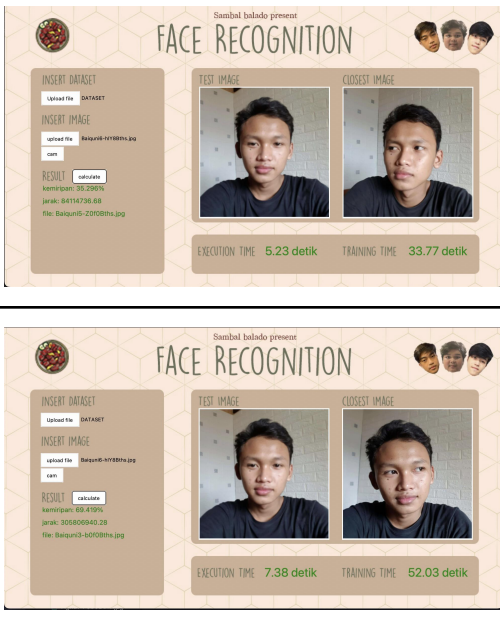
#### 2. Perubahan Jumlah Dataset

Hasil Tes	Pengaruh Perubahan
	<p>Gambar 1 menggunakan folder DATASET berisi 84 file gambar .jpg, sedangkan Gambar 2 menggunakan folder DATASET REDUCED berisi 40 file gambar .jpg. Terdapat perubahan signifikan dalam waktu training image dan waktu menghitung euclidean distance yang dibutuhkan, untuk Gambar 1 membutuhkan waktu training 33.77 detik dan execution time 5.69 detik, sedangkan untuk Gambar 2 membutuhkan waktu training 15.29 detik dan execution time 2.53 detik. (dalam kasus ini pengurangan dataset sebanyak 2x mengurangi waktu eksekusi sebanyak 2x juga). Untuk euclidean distance-nya adalah 0 karena test image ada di dalam dataset sehingga didapat</p>



	kemiripan 100%.
---	-----------------

## B. Gambar Tes Mirip dengan yang Ada di Dalam Dataset

### 1. Perubahan Ukuran Gambar

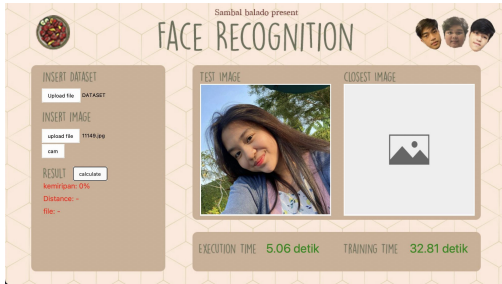
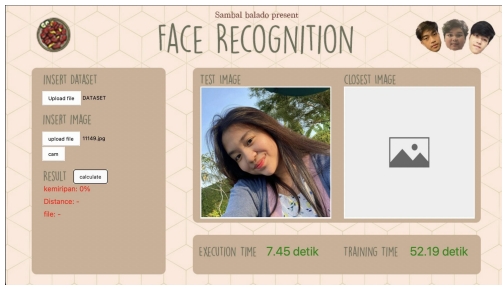
Hasil Tes	Pengaruh Perubahan
	<p>Gambar 1 menggunakan folder DATASET dengan setiap gambarnya berukuran 216x216, sedangkan Gambar 2 menggunakan folder DATASET dengan setiap gambarnya berukuran 512x512. Terdapat perubahan signifikan dalam waktu training image dan waktu menghitung euclidean distance yang dibutuhkan, untuk Gambar 1 membutuhkan waktu training 33.77 detik dan execution time 5.23 detik, sedangkan untuk Gambar 2 membutuhkan waktu training 52.03 detik dan execution time 7.38 detik. (dalam kasus ini penambahan ukuran gambar sebanyak 2x menambah waktu eksekusi sebanyak hampir 1.5x). Untuk gambar yang mirip, euclidean distance-nya berkisar antara 30 juta sampai dengan 80 juta. Kami mengambil kesimpulan bahwa semakin besar ukuran gambarnya, semakin akurat pula gambar yang didapat, selain itu euclidean yang didapat juga lebih kecil.</p>

## 2. Perubahan Jumlah Dataset

Hasil Tes	Pengaruh Perubahan
	<p>Gambar 1 menggunakan folder DATASET berisi 84 file gambar .jpg, sedangkan Gambar 2 menggunakan folder DATASET REDUCED berisi 40 file gambar .jpg. Terdapat perubahan signifikan dalam waktu training image dan waktu menghitung euclidean distance yang dibutuhkan, untuk Gambar 1 membutuhkan waktu training 33.77 detik dan execution time 5.23 detik, sedangkan untuk Gambar 2 membutuhkan waktu training 15.29 detik dan execution time 2.43 detik. (dalam kasus ini pengurangan dataset sebanyak 2x mengurangi waktu eksekusi sebanyak 2x juga). Untuk gambar yang mirip, euclidean distance-nya berkisar antara 60 juta sampai dengan 80 juta. Kami mengambil kesimpulan bahwa semakin banyak dataset yang dipakai, semakin akurat pula gambar yang didapat, selain itu euclidean yang didapat juga lebih kecil.</p>
	

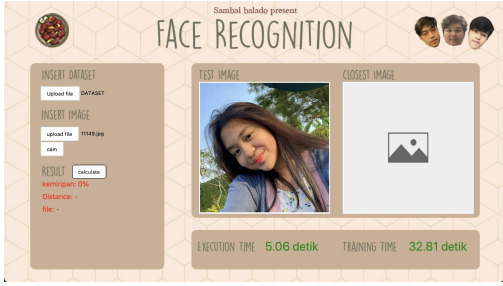
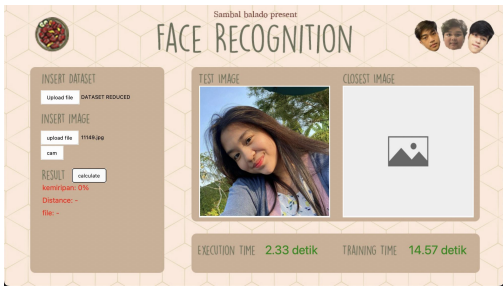
## C. Gambar Tes Tidak Mirip dengan yang Ada di Dalam Dataset

### 1. Perubahan Ukuran Gambar

Hasil Tes	Pengaruh Perubahan
	<p>Gambar 1 menggunakan folder DATASET dengan setiap gambarnya berukuran 216x216, sedangkan Gambar 2 menggunakan folder DATASET dengan setiap gambarnya berukuran 512x512. Terdapat perubahan signifikan dalam waktu training image dan waktu menghitung euclidean distance yang dibutuhkan, untuk Gambar 1 membutuhkan waktu training 32.81 detik dan execution time 5.06 detik, sedangkan untuk Gambar 2 membutuhkan waktu training 52.19 detik dan execution time 7.45 detik. (dalam kasus ini penambahan ukuran gambar sebanyak 2x menambah waktu eksekusi sebanyak hampir 1.5x). Untuk euclidean distance-nya sangat besar karena tidak ada gambar yang mirip, sehingga melewati threshold yang telah kami tetapkan.</p>
	



## 2. Perubahan Jumlah Dataset

Hasil Tes	Pengaruh Perubahan
	<p>Gambar 1 menggunakan folder DATASET berisi 84 file gambar .jpg, sedangkan Gambar 2 menggunakan folder DATASET REDUCED berisi 40 file gambar .jpg. Terdapat perubahan signifikan dalam waktu training image dan waktu menghitung euclidean distance yang dibutuhkan, untuk Gambar 1 membutuhkan waktu training 32.81 detik dan execution time 5.06 detik, sedangkan untuk Gambar 2 membutuhkan waktu training 14.57 detik dan execution time 2.33 detik. (dalam kasus ini pengurangan dataset sebanyak 2x mengurangi waktu eksekusi sebanyak 2x juga). Untuk euclidean distance-nya sangat besar karena tidak ada gambar yang mirip, sehingga melewati threshold yang telah kami tetapkan.</p>
	

## **BAB V**

### **KESIMPULAN**

#### **A. Kesimpulan**

Pengenalan wajah atau face recognition adalah sebuah teknologi biometrik yang mampu mengidentifikasi fitur-fitur wajah seseorang untuk melakukan berbagai kepentingan. Pengenalan wajah ini dapat dilakukan salah satunya dengan menggunakan EigenFaces (PCA algorithm).

Pada tugas besar 2 ini, kami telah membuat program yang dapat mengenali wajah dan mencari wajah yang paling mirip dari dataset yang ada. Program ini sendiri dibuat pada bahasa python dengan algoritma EigenFaces. Untuk proses mendapatkan nilai eigen dan eigen vektor yang nantinya akan dibutuhkan pada algoritma EigenFaces, kami menggunakan algoritma QR. Program ini ditampilkan dengan melalui TKinter . Dengan begitu, kami dapat menyimpulkan bahwa algoritma EigenFaces dapat digunakan untuk membuat program pengenalan wajah

#### **B. Saran**

Setelah menyelesaikan tugas besar kedua ini, kami memiliki beberapa saran untuk kelompok kami, yaitu pemberian komentar pada kode yang dibuat untuk membuat kami dapat lebih mengerti dalam memahami kode tersebut sehingga semua anggota kelompok bisa melakukan debugging jika terdapat bug.

#### **C. Refleksi**

Untuk refleksi, menurut kami, kami dapat melakukan pembagian kerja dan timeline dengan lebih baik supaya pengerjaan dapat lebih teratur dan terstruktur.

## REFERENSI

1. <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2022-2023/Tubes2-Algeo-2022.pdf>
2. <https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/>
3. <http://laid.delanover.com/explanation-face-recognition-using-eigenfaces/>
4. <https://www.neliti.com/publications/135138/pengenalan-wajah-menggunakan-algoritma-eigenface-dan-euclidean-distance>

## LAMPIRAN

Link Repository Github : <https://github.com/Raditss/Algeo02-21021>

Link Video Bonus : <https://youtu.be/r6tBeh2SuAI>