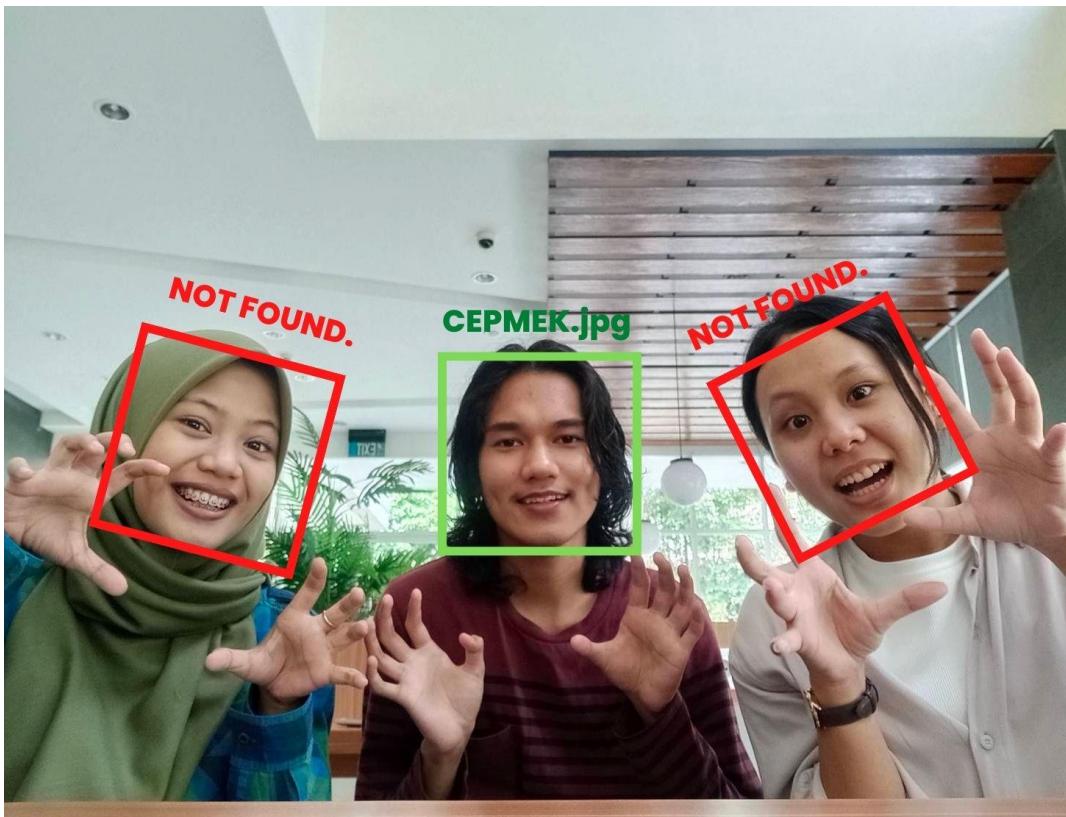


# Tugas Besar 2 IF 2123 Aljabar Linear dan Geometri

## Aplikasi Nilai Eigen dan EigenFace pada Pengenalan Wajah (Face Recognition)



**Kelompok “KamuNennyNamaKelompoknyaApa”**

Kartini Copa 13521026

M. Malik I. Baharsyah 13521029

Jauza Lathifah Annassalafi 13521030

## DAFTAR ISI

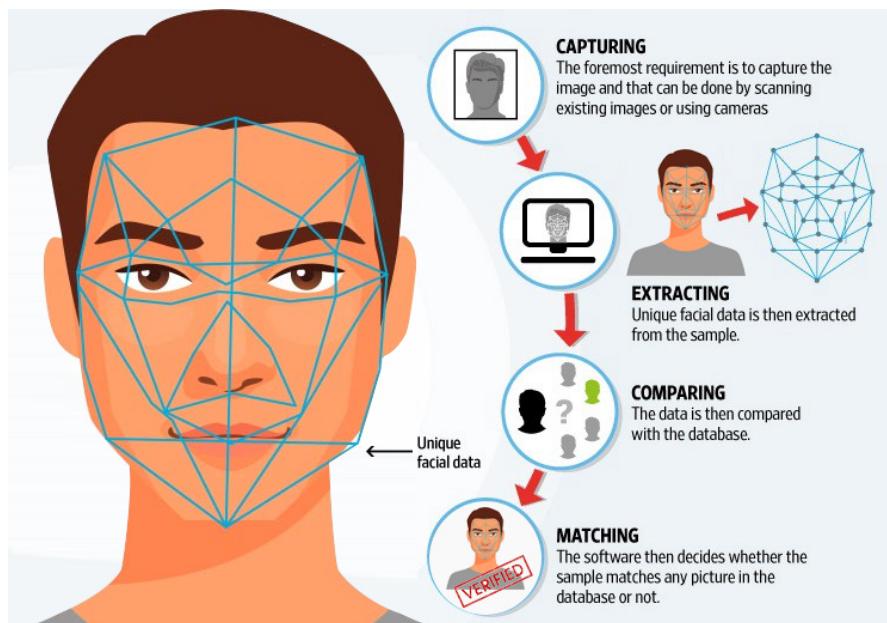
<b>BAB I</b>	<b>3</b>
<b>DESKRIPSI MASALAH</b>	<b>3</b>
1.1 Deskripsi Masalah	3
<b>BAB II</b>	<b>5</b>
<b>TEORI SINGKAT</b>	<b>5</b>
2.1 Perkalian Matriks	5
2.2 Nilai Eigen	6
2.3 Vektor Eigen	6
2.4 Eigenface	7
2.4 QR Decomposition	8
<b>BAB III</b>	<b>9</b>
<b>IMPLEMENTASI</b>	<b>9</b>
3.1 Tech Stack	9
3.2 Garis Besar Algoritma	10
<b>BAB IV</b>	<b>14</b>
<b>EKSPERIMEN</b>	<b>14</b>
4.1 Tampilan GUI	14
4.1. Hasil Face Recognition Berdasarkan File Gambar	14
4.2. Hasil Face Recognition dengan Webcam	15
<b>BAB V KESIMPULAN</b>	<b>19</b>
5.1 Kesimpulan	19
5.2 Saran	19
5.3 Refleksi	20
<b>DAFTAR REFERENSI</b>	<b>20</b>
<b>LAMPIRAN</b>	<b>20</b>

# BAB I

## DESKRIPSI MASALAH

### 1.1 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. Alur proses sebuah sistem pengenalan wajah diperlihatkan pada Gambar 1.



**Gambar 1.** Alur proses di dalam sistem pengenalan wajah (Sumber: <https://www.shadowsystem.com/page/20>)

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 pencocokan. 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 menggunakan eigenvector.

Pada tugas besar Algeo 2 ini, akan dibuat program pengenalan wajah dalam Bahasa Python berbasis GUI dengan spesifikasi sebagai berikut:

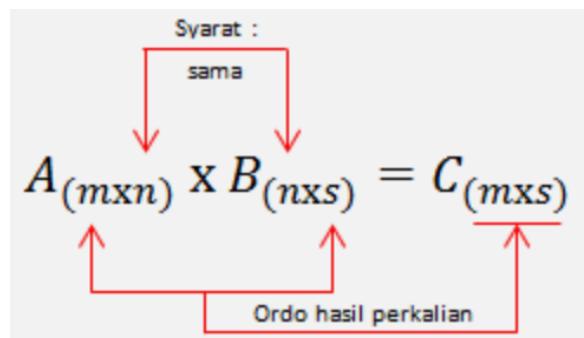
1. Program menerima input folder dataset dan sebuah gambar citra wajah.
2. Program menampilkan gambar citra wajah yang dipilih oleh pengguna.
3. Program melakukan pencocokan wajah dengan koleksi wajah yang ada di folder yang telah dipilih. Metrik untuk pengukuran kemiripan menggunakan eigenface + jarak euclidean.
4. Program menampilkan 1 hasil pencocokan pada dataset yang paling dekat dengan gambar input atau memberikan pesan jika tidak didapatkan hasil yang sesuai.
5. Program menghitung jarak euclidean dan nilai eigen & vektor eigen yang ditulis sendiri. Tidak boleh menggunakan fungsi yang sudah tersedia di dalam library atau Bahasa Python.

## BAB II

### TEORI SINGKAT

#### 2.1 Perkalian Matriks

Perkalian dua buah matriks misalnya matriks  $A$  dan  $B$  bisa dilakukan jika jumlah kolom  $A$  sama dengan jumlah baris  $B$ . Perkalian kedua matriks akan menghasilkan matriks dengan jumlah baris sama dengan baris matriks  $A$  dan jumlah kolom sama dengan kolom matriks  $B$ .



Gambar 2.1.1 Syarat perkalian matriks

$$C_{m \times n} = A_{m \times r} \times B_{r \times n}$$

Misalkan  $A = [a_{ij}]$  dan  $B = [b_{ij}]$  maka  $C = A \times B = [c_{ij}]$

$$[c_{ij}] = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{in}b_{nj}$$

The diagram shows the element-wise multiplication of two 3x3 matrices,  $A$  and  $B$ , to produce matrix  $C$ . Matrix  $A$  is given as  $\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$  and matrix  $B$  as  $\begin{pmatrix} p & q \\ r & s \\ t & u \end{pmatrix}$ . An arrow labeled "Kalikan sesuai urutannya" (Multiply according to its order) points to the element  $a$  in matrix  $A$ . The resulting matrix  $C$  is shown as  $\begin{pmatrix} ap+br+ct & aq+bs+cu \\ dp+er+ft & dq+es+fu \\ gp+hr+it & gq+hs+iu \end{pmatrix}$ .

Gambar 2.1.2 Perkalian matriks

## 2.2 Nilai Eigen

Nilai eigen menyatakan nilai karakteristik dari sebuah matriks yang berukuran  $n \times n$ . Jika  $A$  adalah matriks  $n \times n$  maka vektor tidak nol  $x$  di  $\mathbb{R}_n$  disebut vektor eigen dari  $A$  jika  $Ax$  sama dengan perkalian suatu skalar dengan  $x$ , sebagai berikut.

$$Ax = \lambda x$$

Skalar  $\lambda$  disebut nilai eigen dari  $A$ , dan  $x$  dinamakan vektor eigen yang berkoresponden dengan  $\lambda$ . Dengan kata lain, nilai eigen menyatakan nilai karakteristik dari sebuah matriks yang berukuran  $n \times n$ .

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

$$Ax = \lambda x$$

$$IAx = \lambda Ix \text{ (kalikan kedua ruas dengan } I = \text{matriks identitas)}$$

$$Ax = \lambda Ix$$

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

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

Agar  $(\lambda I - A)x = 0$  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.

## 2.3 Vektor Eigen

Vektor eigen  $x$  menyatakan matriks kolom yang apabila dikalikan dengan sebuah matriks  $n \times n$  menghasilkan vektor lain yang merupakan kelipatan vektor itu sendiri. Operasi  $Ax = \lambda x$  menyebabkan vektor  $x$  menyusut atau memanjang dengan faktor dengan arah yang sama jika positif dan arah berkebalikan jika negatif.

Vektor eigen didapatkan dengan memasukan semua nilai eigen kedalam persamaan  $(\lambda I - A)x = 0$  dengan  $\lambda$  adalah nilai eigen sehingga didapat basis ruangan eigen untuk tiap  $\lambda$ .

Untuk  $\lambda = 5 \rightarrow$

$$\begin{bmatrix} \lambda - 3 & 2 & 0 \\ 2 & \lambda - 3 & 0 \\ 0 & 0 & \lambda - 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 2 & 2 & 0 \\ 2 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Selesaikan dengan eliminasi Gauss:

$$\begin{array}{l} \text{matriks augmented} \quad \begin{bmatrix} 2 & 2 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} R1/2 \begin{bmatrix} 1 & 1 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} R2 -2R1 \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ \text{diperoleh persamaan: } x_1 + x_2 = 0 \rightarrow x_1 = -x_2 \\ \text{misal } x_2 = s, x_3 = t, \text{ maka } x_1 = -s \end{array}$$

Ruang eigen:  $E(5) = \{ \mathbf{x} = \begin{bmatrix} -s \\ s \\ t \end{bmatrix} = s \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} + t \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, s \text{ dan } t \in \mathbb{R} \}$

Basis ruang eigen:  $\left\{ \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$  karena  $\begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$  dan  $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$  bebas liner

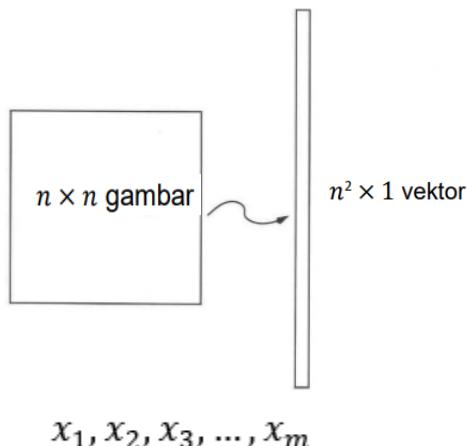
Gambar 2.3.1 Contoh Vektor Eigen

Vektor Eigen juga dapat dicari menggunakan beberapa algoritma lain salah satunya, yaitu menggunakan QR Decomposition

## 2.4 Eigenface

*Eigenface* merupakan salah satu algoritma pengenalan pola wajah yang berdasarkan pada Principle Component Analysis (PCA). Prinsip dasar dari pengenalan wajah adalah dengan mengutip informasi unik wajah tersebut kemudian di-*encode* dan dibandingkan dengan hasil *decode* yang sebelumnya dilakukan. Dalam metode *eigenface*, *decoding* dilakukan dengan menghitung vektor eigen kemudian direpresentasikan dalam sebuah matriks yang berukuran besar.

Melakukan konversi gambar menjadi vektor berukuran  $n \times n$



$$\Psi = \frac{1}{m} \sum_{i=1}^m x_i$$

$$a_i = x_i - \Psi$$

Kemudian diperoleh matriks berukuran  $n^2 \times m$

$$A = [a_1 \ a_2 \ a_3 \dots \ a_m]$$

Mencari kovarian dari matriks  $A$  dengan cara sebagai berikut.

$$Kov = A^T A$$

Setelah melewati tahap-tahap di atas, maka akan dicari nilai eigen dan vektor eigen.

## 2.4 QR Decomposition

*QR decomposition* atau biasa juga dikenal dengan faktorisasi QR merupakan salah satu metode pemfaktoran matriks selain Singular Value Decomposition (SVD). *QR decomposition* dapat digunakan untuk mencari determinan suatu matriks persegi. Pada *QR decomposition*, sebuah matriks  $A$  akan difaktorkan menjadi dua buah matriks yaitu matriks ortogonal  $Q$  dan  $R$  merepresentasikan matriks segitiga atas dengan persamaan sebagai berikut.

$$A = QR$$

$$Q^T \times Q = Q \times Q^T = I$$

$$Q^T = Q^{-1}$$

Invers dari matriks ortogonal sama dengan transpose dari matriks itu sendiri.  $A$  dan  $B$  merupakan matriks yang *similar* jika ada matriks non-*singular*  $X$ .

$$B = X^{-1} \times A \times X$$

Karena  $X$  merupakan matriks non-*singular*, maka matriks  $A$  dan  $B$  memiliki nilai eigen. Dengan demikian, dengan menggunakan *QR decomposition* dapat diperoleh nilai eigen suatu matriks.

## BAB III

### IMPLEMENTASI

#### 3.1 Tech Stack

Program pengenalan wajah ini diimplementasikan dalam aplikasi *desktop GUI* python, program dibagi menjadi dua bagian yaitu *frontend* dan *backend*. Bagian *frontend* merupakan tampilan aplikasi yang dapat dilihat pengguna, sedangkan bagian *backend* proses *training* dan deteksi gambar. Program menerima input folder dataset dan sebuah gambar citra wajah. Program melakukan pencocokan wajah dengan koleksi wajah yang ada di folder yang telah dipilih. Pengukuran kemiripan menggunakan *eigenface* dan jarak *euclidean* yang diimplementasikan pada *code*. Program akan menampilkan hasil pencocokan pada dataset yang paling dekat dengan gambar *input* atau memberikan pesan jika tidak didapatkan hasil yang sesuai.

*Library* yang kami gunakan adalah sebagai berikut.

- Numpy : mempercepat operasi matriks dan kalkulasi, dan proses QR decomposition
- cv2 : *library* opencv-python untuk membaca sebuah gambar menjadi matriks.
- os : mendapatkan *filepath* dari file yang dieksekusi program
- time : mendapatkan waktu eksekusi program
- tkinter : *framework* GUI

Bentuk pengaplikasiannya adalah sebagai berikut.

```
from tkinter import*
from PIL import Image, ImageTk
from tkinter import filedialog
import os
import getEigenFace
import cv2
import numpy as np
```

Gambar 3.1.1 Aplikasi *library*

## 3.2 Garis Besar Algoritma

Algoritma pengenalan wajah pada program kami terletak di dalam folder 'src'. Pada program terdapat pengimplementasian algoritma *eigenface*. Pengimplementasiannya dengan melakukan pengumpulan dataset menjadi satu matriks, mencari nilai mean matriks, mencari selisih antara nilai mean dengan nilai setiap *dataset image*, menghitung nilai matriks kovarian tereduksi, dan mencari nilai eigen dan vektor eigen. Setelah didapat nilai eigen dan vektor eigen, dicari nilai bobot dengan mengalikan transpose vektor eigen dengan matriks selisih. Pada tahapan akhir, akan ditemui gambar dengan jarak *euclidean* paling kecil yang dikenali oleh program paling menyerupai *test face* atau tidak terdapat citra wajah yang mirip dengan test face jika tidak memenuhi.

Implementasi algoritma QR *decomposition* ditunjukkan pada gambar 3.1.2 di bawah. Fungsi QR *decomposition* akan mengembalikan matriks Q dan R.

```
def QRDecomposition(matrix):
    n, m = matrix.shape
    matrixQ = np.empty((n, n))
    matrixU = np.empty((n, n))
    matrixU[:, 0] = matrix[:, 0]
    matrixQ[:, 0] = matrixU[:, 0] / EuclideanDistance(matrixU[:, 0])
    for i in range(1, n):
        matrixU[:, i] = matrix[:, i]
        for j in range(i):
            matrixU[:, i] -= (matrix[:, i] @ matrixQ[:, j]) * matrixQ[:, j]
        matrixQ[:, i] = matrixU[:, i] / EuclideanDistance(matrixU[:, i])
    matrixR = np.zeros((n, m))
    for i in range(n):
        for j in range(i, m):
            matrixR[i, j] = matrix[:, j] @ matrixQ[:, i]
    return matrixQ, matrixR
```

Gambar 3.1.2 Implementasi algoritma QR *decomposition*

Implementasi algoritma vektor eigen ditunjukkan pada gambar 3.1.3 di bawah.

```
def eigValVec(matrix):
    eigVector = np.eye(matrix.shape[0])
    X = np.copy(matrix)
    for i in range(1):
        Q,R = QRDecomposition(X)
        eigVector = np.matmul(eigVector, Q)
        X = np.matmul(R, Q)
    return np.diag(X), eigVector

def EuclideanDistance(matrix):
    return np.sqrt(np.sum(np.square(matrix)))
```

Gambar 3.1.3 Implementasi algoritma vektor eigen

Implementasi algoritma *eigenface* ditunjukkan pada gambar 3.1.4 di bawah.

```
def getEigenFace(path):
    global mean_face
    global subtracted_face
    global start_time_training
    global end_time_training
    global evects
    start_time_training = time.time()
    nData = len(os.listdir(path))
    facematrix = getFolder.folderToMatriks(path)
    facematrix_t = np.transpose(facematrix)
    mean_face = function.mean(facematrix_t)
    subtracted_face = function.selisih(facematrix_t, mean_face)
    cov = function.kovarian(subtracted_face)
    cov = cov / nData
    evals = function.eigValVec(cov)[0]
    egvecs = function.eigValVec(cov)[1]
    idx = evals.argsort()[:-1]
    evals = evals[idx]
    evects = egvecs[:,idx]
    evects = evects[:,0:nData-1]
    evects = np.matmul(subtracted_face, evects)
    norms = np.array([])
    for i in range(0, nData-1):
        norms = np.append(norms, function.EuclideanDistance(evects[:,i]))
    evects = evects / norms
    bobot = np.matmul(np.transpose(evects), subtracted_face)
    end_time_training = time.time()
    return bobot
```

Gambar 3.1.4 Implementasi algoritma *eigenface*

Setelah mendapat nilai bobot berdasarkan nilai eigen dan vektor eigen, algoritma sudah mampu menemukan wajah dengan jarak euclidean terpendek. Dengan ini,

algoritma dapat diimplementasikan pada pendekripsi wajah secara realtime menggunakan webcam. Dengan *library* opencv, webcam dipanggil lalu mencari wajah dengan klasifikasi haarcascade. Setelah kamera berhasil *track* suatu wajah, fungsi rekognisi menggunakan eigenface akan dipanggil lalu nama gambar yang mirip ditampilkan di layar jika jarak euclidean berada di bawah ambang batas.

```
def detectCam(bobot, datapath, img):
    testface = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    testface = faceAlignment.faceAlignment(testface)
    testface = testface.ravel()
    testface = np.matrix(testface)
    testface = np.transpose(testface)
    testface = function.selisih(testface, mean_face)
    s = np.transpose(evects) * testface
    diff = bobot - s
    norms = np.array([])
    for i in range(0, len(os.listdir(datapath))):
        norms = np.append(norms, function.EuclideanDistance(diff[:,i]))
    min = np.amin(norms)
    max = np.amax(norms)
    idx = np.argmin(norms)
    nama = os.listdir(datapath)[idx]
    return nama, min/max
```

Gambar 3.1.5 Implementasi algoritma pendekripsi wajah

```

def Webcam():
    if (not eigenfaces.any()):
        popupmsg("Mohon pilih dataset terlebih dahulu")
    else:
        cam = cv2.VideoCapture(0)
        face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

    while True:
        _, img_cam = cam.read()

        faces = face_cascade.detectMultiScale(img_cam, 1.1, 4)

        nama_cam = "NOT FOUND"

        for (x, y, w, h) in faces:

            if len(faces) > 0:
                hasil_cam = getEigenFace.detectCam(eigenfaces, filepath, img_cam)
                if hasil_cam[1] < 0.5:
                    nama_cam = hasil_cam[0]

                cv2.rectangle(img_cam, (x, y), (x+w, y+h), (255, 0, 0), 2)

                font = cv2.FONT_HERSHEY_SIMPLEX
                if len(faces) > 0:
                    cv2.putText(img_cam, nama_cam, (x, y-10), font, 0.9, (36,255,12), 2)

        cv2.imshow('Deteksi Webcam', img_cam)
        k = cv2.waitKey(30) & 0xFF
        if k==27:
            break
    cam.release()

```

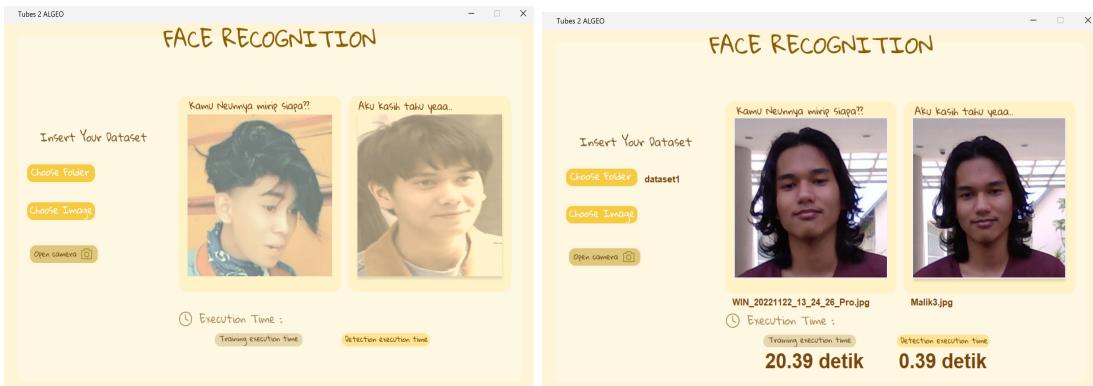
Gambar 3.1.6 Implementasi algoritma pendekripsi wajah dengan webcam

## BAB IV

# EKSPERIMENT

### 4.1 Tampilan GUI

Berikut tampilan GUI kami pada saat sebelum mengeluarkan hasil dan setelah terdapat hasil gambar yang mirip.



Gambar 4.1.1 Tampilan GUI

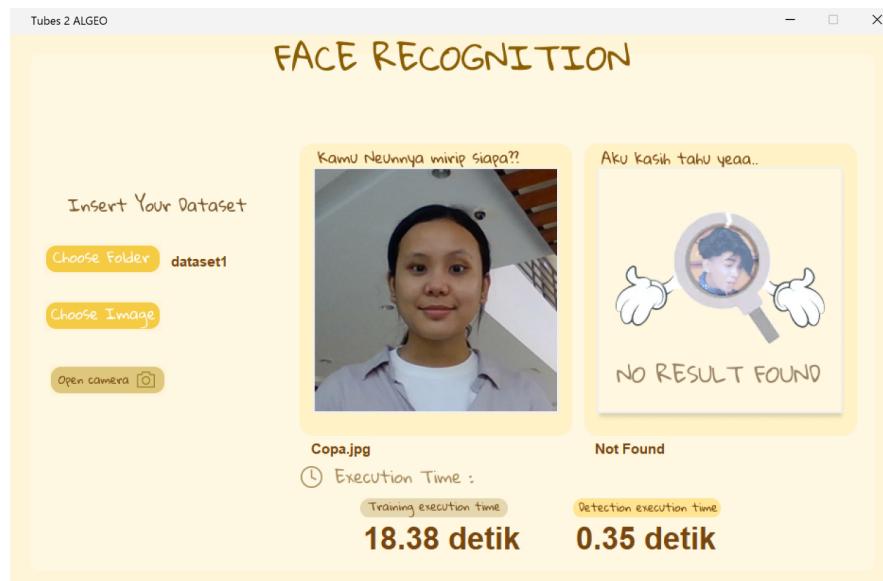
#### 4.1. Hasil Face Recognition Berdasarkan File Gambar

Berikut merupakan hasil eksekusi program tampilan dari program pengenalan wajah kami. Proses yang pertama adalah memilih dataset dengan menekan tombol 'Choose Folder'. Proses selanjutnya dengan memilih gambar yang ingin diuji dengan menekan tombol 'Choose Image'. Setelah memilih dataset dan gambar, program akan menghasilkan *output* gambar yang memiliki kemiripan atau jika tidak ada gambar yang mirip maka akan ditampilkan 'No result Found'.

Pada eksperimen yang kami lakukan, kami menggunakan folder dataset yang berisikan 230 gambar dengan lama waktu eksekusi yang diperlukan sebesar 20 detik untuk mengubah folder yang berisikan gambar-gambar menjadi sebuah matriks. Lalu, dipilih sebuah gambar yang ingin kita lakukan pencocokan wajah dan terhitung sekitar 0.4 detik lama waktu eksekusi pencocokan wajah. Dibawah ini hasil pencocokan wajah jika gambar ditemukan dan gambar tidak ditemukan.



Gambar 4.1.1 Gambar ditemukan



Gambar 4.1.2 Gambar tidak ditemukan

## 4.2. Hasil Face Recognition dengan Webcam

Fitur kedua adalah pengenalan wajah dengan webcam. Pilih folder dataset yang berisikan gambar-gambar. Lalu, buka kamera dan akan terdeteksi jika gambar wajah

kita terdapat di dalam dataset dan akan mengeluarkan “not found” jika tidak terdapat gambar wajah kita di dalam folder dataset.

Berikut merupakan tampilan wajah berhasil dikenali karena wajah terdapat dalam dataset.



Gambar 4.1.3 Wajah terdeteksi

Tampilan apabila wajah tidak dikenali karena tidak terdapat dalam dataset ditunjukkan pada gambar 4.1.4 di bawah.



Gambar 4.1.3 “NOT FOUND” karena tidak ada gambar di dataset yang cocok

## BAB V

### KESIMPULAN

#### 5.1 Kesimpulan

- Program pengenalan wajah merupakan aplikasi dari materi pembelajaran mata kuliah Aljabar Linier dan Geometri IF2123. Melalui penerapan materi-materi tersebut, program ini berhasil diselesaikan sesuai dengan ketentuan yang tertera pada spesifikasi Tugas Besar 2 IF2123 Aljabar Linier dan Geometri dengan mencari nilai eigen dan vektor eigen dari suatu matriks, mencari QR *decomposition*, kemudian menggunakannya untuk mencari *eigenface* sehingga dapat dilakukan pengenalan wajah
- Vektor eigen merupakan vektor yang berada pada suatu ruang eigen (*eigenspace*) sehingga terdapat beragam kemungkinan nilai dari suatu vektor eigen. Nilai yang inkonsisten ini dapat berpengaruh pada hasil perkalian matriks-matriks dekomposisi
- Metode *eigenface* sensitif terhadap variasi pencahayaan, pose, skala, ekspresi wajah, and occlusion. Sehingga terdapat banyak kesalahan pengenalan wajah pada program. Program juga tidak sesempurna *machine learning* sehingga banyak kesalahan pengenalan wajah yang terjadi
- Program dapat menampilkan hasil kemiripan gambar pada tampilan GUI

#### 5.2 Saran

- Para penulis menyarankan untuk mengalokasi waktu demi mendalami dan memahami masalah yang akan diselesaikan serta penyelesaian yang dirasa sesuai sebelum memulai tahap penggerjaan
- Pemahaman materi aljabar linier dan geometri yang lebih dalam dibutuhkan untuk penggerjaan program pengenalan wajah karena sangat berkorelasi
- Pengetahuan kompleksitas algoritma lebih dalam agar eksekusi program lebih efisien dan cepat
- Tingkat akurasi pada program perlu dimaksimalkan karena program kami masih banyak kurangnya sehingga banyak kesalahan pengenalan wajah yang terjadi

### **5.3 Refleksi**

- Melalui tugas besar ini para penulis menyadari secara nyata implementasi dari materi Aljabar Linier dan Geometri IF2123
- Para penulis sadar akan pentingnya program pengenalan wajah di era digital dan dapat mengimplementasikan program pengenalan wajah dalam bahasa python.
- Para penulis menyadari perlunya pemahaman lebih dalam untuk menyempurnakan program pengenalan wajah yang telah dibuat
- Kelancaran proses penggerjaan tugas besar ini jauh dari ideal. Beragam kendala dan juga hambatan ditemui oleh para penulis selama proses penyelesaian tugas besar. Para penulis merasa perlu belajar dan meningkatkan kemampuan untuk menyelesaikan tugas besar.

## **DAFTAR REFERENSI**

“Nilai Eigen dan Vektor Eigen Bagian 1” by Rinaldi Munir

<https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-Nilai-Eigen-dan-Vektor-Eigen-Bagian1.pdf>

Munir, Rinaldi. 2021. Singular Value Decomposition. Diakses pada 7 November 2022

<https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-19bSingular-value-decomposition.pdf>

pythontutorial. Diakses pada 12 November 2022

<https://www.pythontutorial.net/tkinter>

“Face Recognition Using Eigenfaces (PCA Algorithm)

<https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/>

## **LAMPIRAN**

Berikut kami cantumkan *link repository* dan *link youtube*.

*Link repository* : <https://github.com/Darkfirm/Algeo02-21026.git>