

TUGAS BESAR 2
IF2123 ALJABAR LINIER DAN GEOMETRI
APLIKASI NILAI EIGEN DAN EIGENFACE PADA PENGENALAN WAJAH
(Face Recognition)
SEMESTER 1 TAHUN 2022/2023



Disusun Oleh

Kelompok 22 NNN

13521110 Yanuar Sano Nur Rasyid

13521121 Saddam Annais Shaquille

13521150 I Putu Bakta Hari Sudewa

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

DAFTAR ISI

DAFTAR ISI	1
BAB I DESKRIPSI MASALAH	2
BAB II TEORI SINGKAT	4
2.1 Perkalian Matriks	4
2.2 Nilai Eigen dan Vektor Eigen	4
2.3 <i>Eigenface</i>	5
BAB III IMPLEMENTASI	7
3.1 GUI	8
3.2 Nilai Eigen dan Vektor Eigen	9
3.3 <i>Eigenface</i> dan Prediksi	10
BAB IV EKSPERIMEN	12
4.1 Tampilan GUI	12
4.2 Hasil Pengenalan Wajah	14
BAB V KESIMPULAN SARAN DAN REFLEKSI	17
5.1 Kesimpulan	17
5.2 Saran	17
5.3 Refleksi	18
DAFTAR PUSTAKA	19
LAMPIRAN	20

BAB I

DESKRIPSI MASALAH

Tugas Besar 2 ini merupakan implementasi konsep Mata Kuliah Aljabar Linier dan Geometri untuk membuat sebuah program pengenalan wajah (*Face Recognition*). 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 beberapa teknik untuk memeriksa citra wajah dengan wajah yang sudah ada di database, seperti jarak Euclidean dan *cosine similarity*, principal component analysis (PCA), serta Eigenface. Program *face recognition* ini menggunakan teknik Eigenface dengan algoritma Eigenface seperti berikut.

Algoritma Eigenface

1. Langkah pertama adalah menyiapkan data dengan membuat suatu himpunan S yang terdiri dari seluruh training image, $(\Gamma_1, \Gamma_2, \dots, \Gamma_M)$

$$S = (\Gamma_1, \Gamma_2, \dots, \Gamma_M) \quad (1)$$

2. Langkah kedua adalah ambil nilai rata-rata atau mean (Ψ)

$$\Psi = \frac{1}{M} \sum_{n=1}^M \Gamma_n \quad (2)$$

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)

$$C = \frac{1}{M} \sum_{n=1}^M \phi_n \phi_n^T = AA^T \quad (4)$$

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

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

$$C \times v_i = \lambda_i \times v_i \quad (5)$$

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

$$\mu_i = \sum_{k=1}^M v_{ik} \phi_k \quad (6)$$

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

Tahapan Pengenalan wajah :

1. Sebuah image wajah baru atau test face (Γ_{new}) akan dicoba untuk dikenali, pertama terapkan cara pada tahapan pertama perhitungan eigenface untuk mendapatkan nilai eigen dari image tersebut.

$$\mu_{new} = v \times \Gamma_{new} - \Psi \quad (7)$$

$$\Omega = \mu_1, \mu_2, \dots, \mu_M$$

2. Gunakan metode euclidean distance untuk mencari jarak (distance) terpendek antara nilai eigen dari training image dalam database dengan nilai eigen dari image testface. $\varepsilon_k = \Omega - \Omega_k \quad (8)$

Pada tahapan akhir, akan ditemui gambar dengan euclidean distance paling kecil maka gambar tersebut yang dikenali oleh program paling menyerupai test face selama nilai kemiripan di bawah suatu nilai batas. Jika nilai minimum di atas nilai batas maka dapat dikatakan tidak terdapat citra wajah yang mirip dengan test face.

Program dibuat dengan Bahasa Python dengan memanfaatkan sejumlah library di OpenCV (Computer Vision) atau library pemrosesan gambar lainnya (contoh PIL). Fungsi untuk mengekstraksi fitur dari sebuah citra wajah tidak perlu anda buat lagi, tetapi menggunakan fungsi ekstraksi yang sudah tersedia di dalam library. Fungsi Eigen dilarang import dari library dan harus diimplementasikan, sedangkan untuk operasi matriks lainnya silahkan menggunakan library.

Spesifikasi Tugas

Buatlah program pengenalan wajah dalam Bahasa Python berbasis GUI dengan spesifikasi sebagai berikut:

1. Program menerima input *folder dataset* dan sebuah gambar citra wajah.
2. Basis data wajah dapat diunduh secara mandiri melalui <https://www.kaggle.com/datasets/hereisburak/pins-face-recognition>.
3. Program menampilkan gambar citra wajah yang dipilih oleh pengguna.
4. Program melakukan pencocokan wajah dengan koleksi wajah yang ada di folder yang telah dipilih. Metrik untuk pengukuran kemiripan menggunakan eigenface + jarak *euclidean*.
5. Program menampilkan 1 hasil pencocokan pada dataset yang paling dekat dengan gambar input atau memberikan pesan jika tidak didapatkan hasil yang sesuai.
6. 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.
7. Input gambar akan berukuran MINIMAL 256 x 256.

BAB II

TEORI SINGKAT

2.1 Perkalian Matriks

Perkalian matriks adalah operasi biner yang menghasilkan matriks baru dari dua matriks yang dikalikan. Jika terdapat dua buah matriks, yaitu matriks A yang memiliki ukuran $m \times n$ dan matriks B yang memiliki ukuran $n \times p$ dan memiliki elemen sebagai berikut sebagai berikut

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

Hasil perkalian matriks dari matriks A dan B adalah matriks $C = A \times B$ yang memiliki ukuran $m \times p$

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

Dengan nilai tiap elemen C_{ij} adalah sebagai berikut

$$C_{ij} = \sum_{k=1}^m a_{ik}b_{kj}$$

2.2 Nilai Eigen dan Vektor Eigen

Jika A adalah matriks $n \times n$, maka vektor tidak nol di R^n disebut vektor eigen dari A . Jika Av sama dengan produk skalar λ dengan v , yaitu $Av = \lambda v$. Skalar λ disebut nilai eigen dari A , dan v disebut vektor eigen yang berkoresponden dengan λ . Kata “eigen” berasal dari Bahasa Jerman yang berarti “asli” atau “karakteristik”. Oleh karena itu, nilai eigen mewakili nilai karakteristik dari matriks berukuran $n \times n$.

Misalkan terdapat suatu vektor eigen v yang merupakan matriks kolom dan terdapat matriks A berukuran $n \times n$. Perkalian antara matriks A dan vektor eigen v akan menghasilkan vektor lain yang merupakan kelipatan dari v . Kelipatan dari vektor v adalah sebesar λ .

Bila terdapat suatu matriks A yang memiliki ukuran $n \times n$. Nilai eigen dan eigen vektor dari matriks A dapat dicari dengan cara sebagai berikut

$$\begin{aligned} Av &= \lambda v \\ A Iv &= \lambda Iv \\ Ax &= \lambda Iv \\ (\lambda I - A)v &= 0 \end{aligned}$$

v merupakan solusi tidak nol sehingga $\det(\lambda I - A) = 0$. Persamaan $\det(\lambda I - A) = 0$ disebut persamaan karakteristik dari matriks A dan akar-akar persamaannya, yaitu λ disebut akar-akar karakteristik atau nilai-nilai eigen. Untuk mendapatkan vektor eigen, λ yang didapat dari persamaan sebelumnya dapat disubstitusikan ke persamaan $(\lambda I - A)v = 0$.

Selain cara di atas, terdapat cara lain yang lebih efisien dan lebih mudah untuk diimplementasikan dalam program. Matriks A dapat didekomposisi menjadi matriks lain yang masih memiliki nilai eigen dan vektor eigen yang sama. Matriks A akan diubah menjadi matriks diagonal atas sehingga semua nilai eigennya terdapat di diagonal matriks A . Setelah mendapatkan nilai eigen, dapat dicari pula vektor eigen dari matriks A tersebut. Berbagai metode mencari eigen value dan eigen vektor adalah sebagai berikut

1. *Lanczos Algorithm*
2. *Power Iteration*
3. *Inverse Iteration*
4. *Rayleigh Quotient Iteration*
5. *QR Algorithm*
6. *Jacobi eigenvalue algorithm*
7. *Divide-and-conquer*

Pada tugas besar ini, *QR Algorithm* dipilih untuk mencari nilai eigen dan vektor eigen.

2.3 Eigenface

Eigenface adalah sebutan untuk suatu set vektor eigen dalam konteks pengenalan citra wajah menggunakan nilai eigen dan vektor eigen. Pendekatan pengenalan gambar menggunakan nilai eigen dan vektor eigen dikembangkan pertama kali oleh Sirovich dan Kirby. Setelah itu, Matthew Turk dan Alex Pentland menggunakannya dalam klasifikasi wajah.

Terdapat beberapa Langkah yang harus diikuti untuk mendapatkan eigenface, yaitu sebagai berikut:

1. Siapkan *training set* gambar wajah yang akan digunakan untuk melatih model prediksi. Gambar dalam dataset tersebut harus memenuhi kriteria yaitu diambil dalam kondisi pencahayaan yang sama, disesuaikan agar mata dan mulut sejajar di semua gambar, dan memiliki ukuran gambar yang seragam, yaitu $(m \times n)$. Kemudian, setiap gambar diubah menjadi vektor berukuran $(n \times m, 1)$ dengan cara menggabungkan baris piksel pada gambar asli. Kemudian, semua gambar disimpan dalam satu matriks T sehingga tiap kolom matriks adalah sebuah gambar.
2. Hitung rata-rata dari gambar yang telah disimpan dalam T kemudian kurangi tiap gambar dengan rata-rata yang telah dihitung dan simpan di matriks A .
3. Hitung nilai eigen dan vektor eigen dari matriks kovarians A . Setiap vektor eigen memiliki dimensi yang sama dengan gambar aslinya sehingga dapat dilihat sebagai gambar. Oleh karena itu, vektor eigen dari matriks kovarians ini disebut eigenface atau wajah eigen.

4. Urutkan vektor eigen berdasarkan korespondensi dari nilai eigen. Ambil sebanyak K vektor eigen yang berkorespondensi dengan K nilai eigen terbesar. K adalah jumlah eigen vektor yang akan diambil. Nilai dari K merupakan bilangan bulat asumsi yang dibebaskan kepada pembuat. Biasanya, nilai K merupakan 10% dari jumlah gambar dalam dataset.

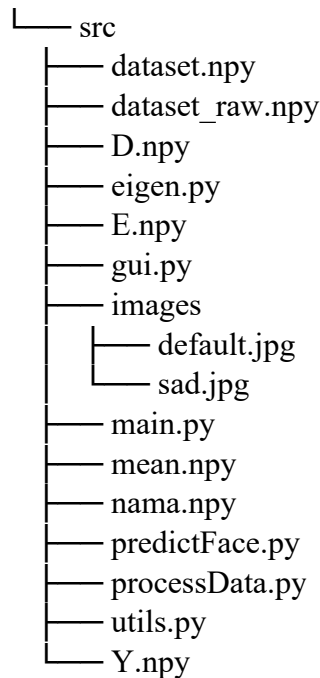
BAB III

IMPLEMENTASI

Tech stack yang digunakan untuk membuat program *face recognition* ini terdiri atas python dengan library NumPy, OpenCV, mediapipe sebagai framework untuk memproses dan mengolah data dan PIL (Python Imaging Library), tkinter, dan customtkinter sebagai framework untuk menampilkan data hasil olahan sebelumnya. Berikut adalah penjelasan singkat dari techstack yang digunakan.

- a) NumPy
NumPy (Numerical Python) merupakan salah satu library python yang berfokus pada komputasi saintifik. Dalam pengerjaan program ini, NumPy merupakan salah satu basis komputasi matematika yang digunakan untuk mengolah matriks gambar, menghitung nilai eigen, dan sebagainya.
- b) OpenCV
Open Source Computer Vision (OpenCV) adalah library yang sering digunakan dalam pengolahan citra. Citra dari dataset dan citra tes diekstraksi menjadi nilai-nilai pixelnya dengan menggunakan library ini.
- c) Mediapipe
Secara umum mediapipe merupakan library yang memiliki fitur dalam *machine learning* suatu citra, seperti deteksi objek dan wajah. Pada program ini, mediapipe digunakan untuk menghilangkan latar belakang citra sehingga citra menjadi lebih jelas dan komputasi yang dilakukan lebih akurat.
- d) PIL
Serupa dengan OpenCV, PIL (Python Imaging Library) adalah salah satu library pengolahan citra. PIL digunakan dalam program ini dikarenakan integrasinya yang lebih mudah dengan library GUI yang digunakan.
- e) Tkinter
Tkinter adalah library bawaan python yang digunakan dalam pembuatan GUI program ini. Mulai dari tampilan input hingga hasil yang diperoleh diimplementasikan dengan menggunakan library ini.
- f) Customtkinter
Berbasiskan tkinter, customtkinter adalah GUI library yang menyediakan *widget-widget* siap pakai untuk memperindah tampilan pada program ini. Selain itu, karena berbasiskan tkinter, library ini dapat dikombinasikan dengan tkinter sehingga mempercepat proses pembuatan GUI dengan hasil yang maksimal.

Berikutnya terkait struktur program dari aplikasi ini, secara garis besar terdiri dari 6 file utama yang terletak pada folder *src*, yaitu *main.py*, *eigen.py*, *predictFace.py*, *processData.py*, *utils.py*, dan *gui.py*, serta beberapa file dengan ekstensi *.npy* yang digunakan dalam *caching* data hasil training dataset. Lalu, dari *flow* program sendiri, kami bagi menjadi tiga poin besar, yaitu GUI, nilai dan vektor eigen, serta *eigenface* dan prediksi.



Struktur tree dari directory src.

3.1 GUI

Sesuai yang sudah dijelaskan pada bagian techstack, graphical user interface (GUI) pada program ini menggunakan *framework* tkinter dan customtkinter dengan *framework* pembantu OpenCV dan PIL sebagai alat untuk mengubah data menjadi gambar yang bisa dilihat oleh pengguna. Pemilihan penggunaan customtkinter dibandingkan hanya tkinter saja karena dengan menggunakan library customtkinter tampilan program lebih terlihat modern dan lebih user friendly jika dibandingkan dengan hanya menggunakan tkinter.

Desain GUI terdiri atas *frame* yang dapat menyimpan elemen UI seperti teks, tombol, gambar, dan switch. Secara garis besar terdapat dua *frame* di dalam program yaitu *frame* kiri yang menyimpan tombol yang bisa diakses oleh user dan berfungsi untuk menjalankan program dan *frame* kanan yang menampilkan hasil akhir setelah program selesai dijalankan. Komponen-komponen tersebut disusun dengan method grid agar tersusun dengan teratur. Penjelasan detail dari komponen GUI adalah untuk *frame* kiri terdapat tombol choose file yang berfungsi untuk menerima input file atau folder dari user, tombol compare yang berfungsi untuk membandingkan citra input dengan citra yang terdapat dalam dataset, dan tombol webcam yang berfungsi untuk menggunakan webcam user sebagai input foto yang akan dibandingkan. Sementara itu, untuk *frame* kanan terdiri dari judul program, gambar yang akan dites dan gambar yang paling mirip dari dataset, hasil perbandingan, dan lama waktu eksekusi program.

Cara kerja tombol tersebut adalah memanggil fungsi yang sudah dibuat ketika tombol ditekan oleh user. Contohnya tombol choose file yang akan memanggil module file dialog dari tkinter untuk menampilkan windows pemilihan file oleh user. Jika user menekan tombol saat

masih terdapat data yang kurang maka akan memunculkan windows error dengan komponen yang dibutuhkan untuk menekan tombol itu.

Untuk menampilkan gambar GUI berbasis tkinter ini, gambar yang akan ditampilkan perlu dikonversi terlebih dahulu dengan menggunakan module Image kemudian ImageTK dari PIL agar dapat ditampilkan. Contohnya adalah gambar default yang terdapat pada src/images/default.jpg ditampilkan pada GUI dengan cara sebelumnya. Sementara itu, gambar yang berasal dari hasil olah data atau array dapat ditampilkan dengan mengkonversi dengan cara yang sama, tetapi menggunakan method from array.

Semua komponen dan fungsi yang sudah dijelaskan di atas diimplementasikan pada file gui.py dengan membuat sebuah Class App berbasis customtkinter.

3.2 Nilai Eigen dan Vektor Eigen

QR algorithm dipilih untuk mencari nilai eigen dan vektor eigen yang akan digunakan pada eigenface. *QR algorithm* menggunakan dekomposisi QR untuk mendapatkan matriks lain yang berbentuk diagonal atas sekaligus memiliki eigen value dan eigen vektor yang sama. Misalkan A adalah suatu matriks yang memiliki ukuran $n \times n$ maka matriks tersebut dapat didekomposisi menjadi $A = QR$. Dengan Q sebagai matriks orthogonal dan R sebagai matriks diagonal atas. Nilai dari Q dapat dicari menggunakan *householder transformation*, kemudian nilai dari R dapat dicari $Q^{-1}A = R$.

QR algorithm pada dasarnya menggunakan iterasi pada dekomposisi QR. Iterasi dilakukan sampai terbentuk matriks yang diagonal atas. Persamaan dari *QR algorithm* adalah sebagai berikut

$$\begin{aligned} A_0 &= Q_0 R_0 \\ A_1 &= R_0 Q_0 = Q_1 R_1 \\ A_2 &= R_1 Q_1 = Q_2 R_2 \\ A_3 &= R_2 Q_2 = Q_3 R_3 \\ &\vdots \quad \quad \quad \vdots \\ A_k &= R_{k-1} Q_{k-1} = Q_k R_k \end{aligned}$$

Matriks $A_0, A_1, A_2 \dots A_k$ memiliki nilai eigen dan vektor eigen yang sama. Perbedaannya adalah matriks A_k memiliki bentuk diagonal atas. Nilai eigen dari matriks tersebut berada di diagonalnya dan matriks dari vektor eigennya adalah $E = Q_0 Q_1 Q_2 \dots Q_n$. *QR Algorithm* dapat ditulis dalam *pseudocode* seperti berikut:

```
Set  $A_0 = A = Q_0 R_0$ 
Set  $E = Q_0$ 
for  $k = 0, 1, 2, \dots$  (until convergence)
    Compute  $A_k = Q_k R_k$ 
    Set  $A_{k+1} = R_k Q_k$ 
    Set  $E = E Q_k$ 
End
```

Setelah dijalankan, nilai eigennya adalah diagonal dari A dan matriks vektornya adalah E

3.3 Eigenface dan Prediksi

Seperti yang telah dijelaskan pada bab sebelumnya terkait *eigenface*, *eigenface* adalah sebutan untuk suatu set vektor eigen yang berasal dari pengolahan suatu citra wajah dengan menggunakan nilai-nilai eigennya. Langkah-langkah yang telah dijelaskan pada bab sebelumnya adalah salah satu acuan yang digunakan dalam implementasi prediksi wajah dalam program ini. Karena *eigenface* sendiri merupakan suatu set dari vektor eigen, maka proses mendapatkan nilai vektor-vektor eigen tersebut telah dijelaskan di bagian 3.2, sedangkan untuk bagian ini akan lebih berfokus kepada pengolahan citra awal hingga menghasilkan suatu matriks yang siap dicari nilai dan vektor eigennya dan proses menghasilkan prediksi citra.

Implementasi dari proses pengolahan gambar ini terdapat pada file *processData.py*. Proses pengolahan citra dimulai dengan mengolah citra dari dataset citra wajah (*training dataset*) yang dimasukkan oleh user. Berikutnya, masing-masing citra wajah dari dataset tersebut dihapus latar belakangnya, dicrop sehingga hanya terlihat bagian wajahnya saja, ukurannya menjadi 256 x 256 pixel, dan memiliki pewarnaan *grayscale*. Modifikasi gambar-gambar tersebut menggunakan library OpenCV, mediapipe, serta NumPy. Setelah itu, gambar sudah siap untuk masuk ke dalam tahap selanjutnya yaitu menggunakan metode *eigenface*. Pertama, matriks hasil ekstraksi citra wajah yang sebut saja berukuran $N \times N$ dikonkatenasi sehingga ukurannya berubah menjadi $N^2 \times 1$. Selanjutnya, ditentukan nilai rata-rata dari semua matriks hasil ekstraksi tersebut dengan menggunakan formula $\Psi = \frac{1}{M} \sum_{i=1}^M x_i$, dengan x_i adalah matriks ke- i hasil ekstraksi. Setelah diperoleh nilai rata-rata, tiap matriks awal x_i dikurangi dengan Ψ dan diperoleh matriks baru ϕ_i . Matriks ϕ_i inilah yang kemudian disusun menjadi sebuah *training* matriks berukuran $N^2 \times M$, sebut saja sebagai matriks $A = (\phi_1, \phi_2, \dots, \phi_M)$. Langkah selanjutnya, adalah menghitung nilai matriks kovarian C dan dari matriks kovarian inilah akan diperoleh nilai dan vektor eigen dari masing-masing citra awal. Setelah diperoleh nilai dan vektor eigen dari matriks kovarian C , berikutnya vektor eigen tersebut diurutkan berdasarkan nilai eigennya dari yang terbesar, lalu vektor-vektor tersebut dinormalisasi sehingga membentuk sebuah matriks baru E yang berukuran $N^2 \times D$ dengan D adalah jumlah vektor eigen yang diinginkan, biasanya nilai D adalah 10% dari jumlah citra dalam dataset. Berikutnya dibentuk sebuah matriks $Y = E^T A$ yang merupakan *weight* dari citra-citra pada dataset. Pada tahap ini proses training citra pada dataset telah selesai. Untuk proses pengenalan wajah pada program, program akan mengurangi gambar tes dengan Ψ . Kemudian, *weight* dari gambar test dihitung dengan cara mengalikan hasil perhitungan sebelumnya dengan *eigenface*. Kemudian menghitung selisih *Euclidian distance* antara berat baru dengan *weight* yang ada pada *training images*. Jika nilai terdapat pada ambang tertentu maka program akan menyatakan gambar mirip dan begitu pula sebaliknya.

Euclidian Distance

$$d(A, B) = \sqrt{\sum_{i=1}^D (a_i - b_i)^2}$$

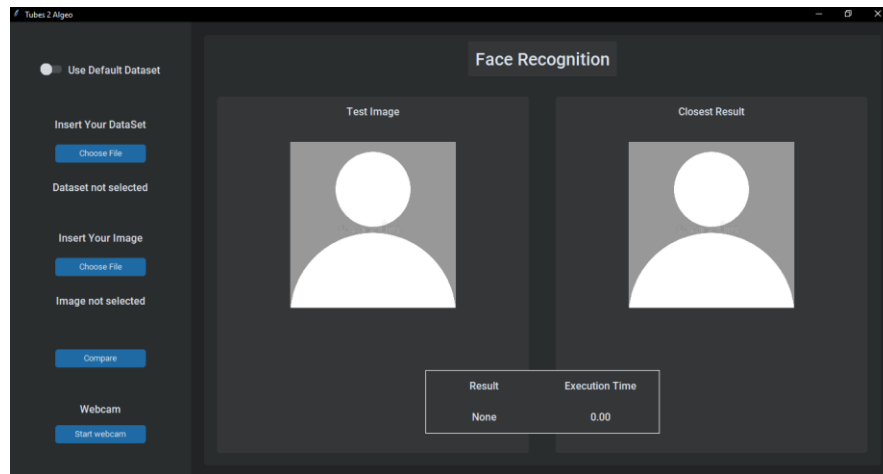
Jika jarak minimum $d(A, B)$ antara ω dengan y_i lebih besar daripada suatu *threshold* θ , maka citra tersebut tidak diketahui, jika sebaliknya, maka citra tersebut diketahui.

BAB IV

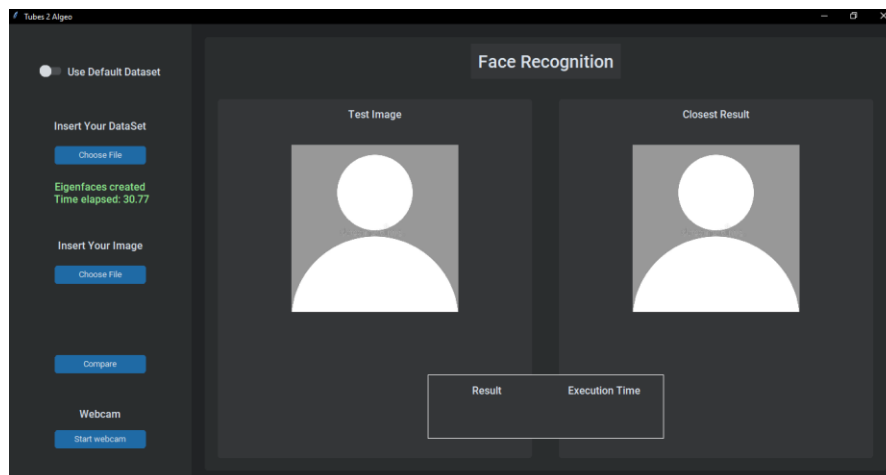
EKSPERIMEN

4.1 Tampilan GUI

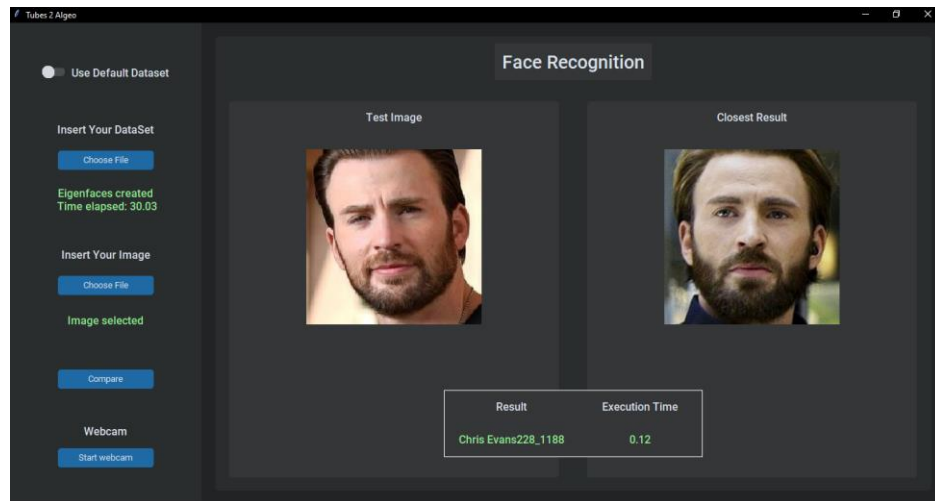
Berikut tampilan GUI kami pada saat sebelum training, setelah training, ketika mengenali wajah, dan ketika kamera dinyalakan.



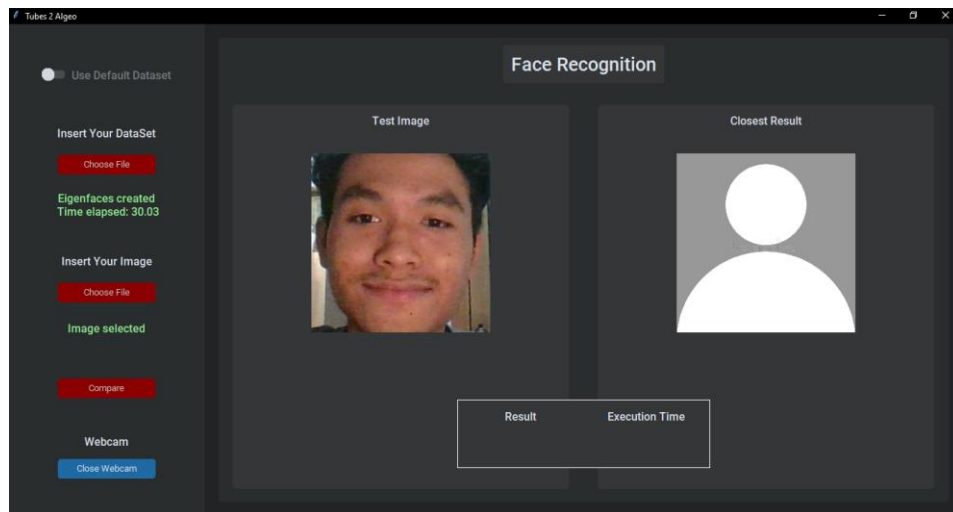
Gambar 4.1.1 GUI Sebelum training



Gambar 4.1.2 GUI Setelah training

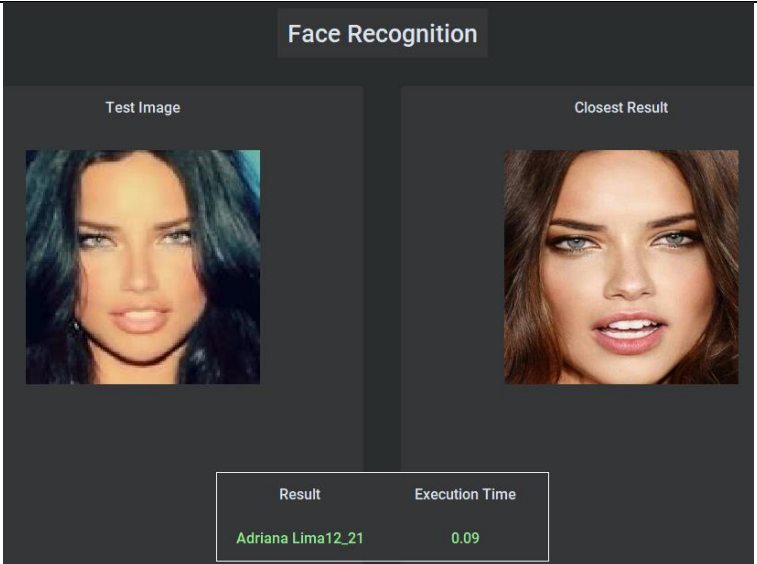
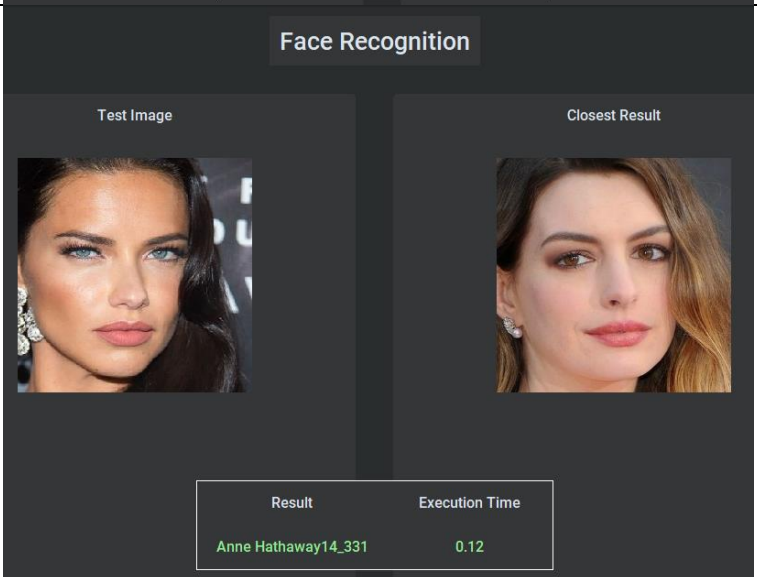


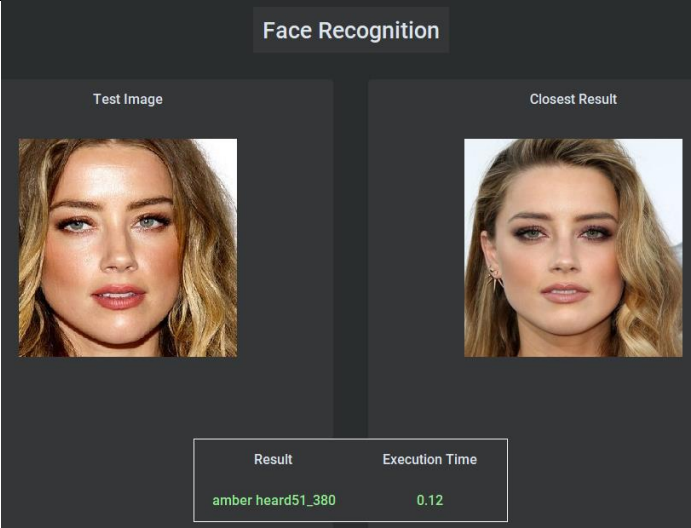
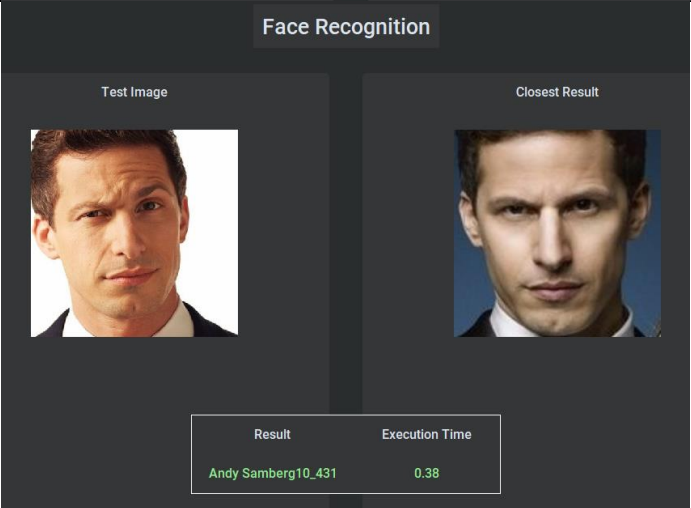
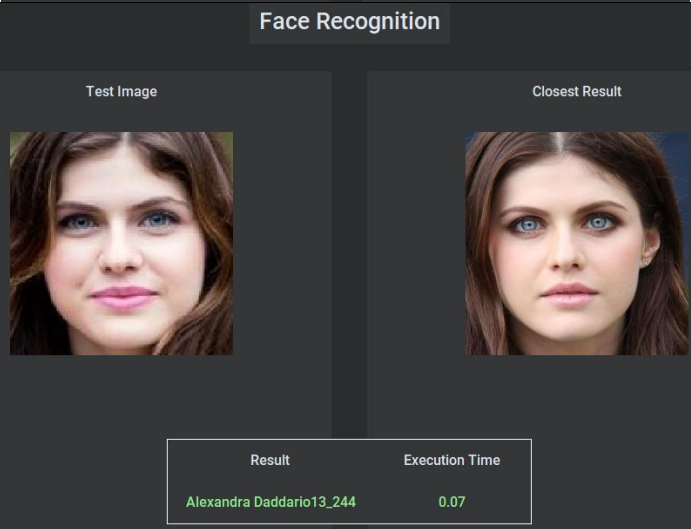
Gambar 4.1.3 GUI Ketika Mengenali Wajah

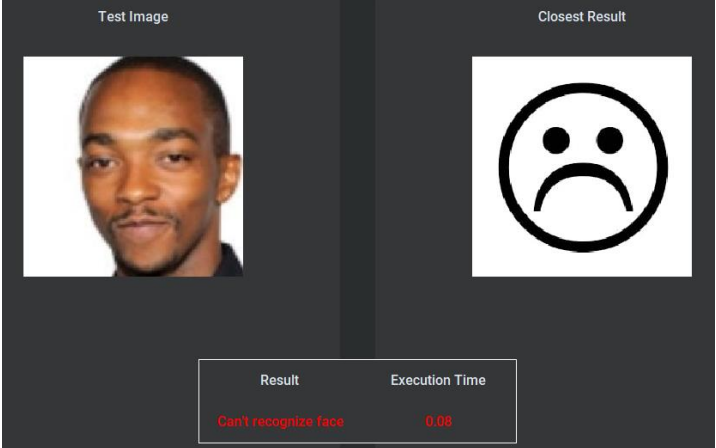


Gambar 4.1.4 GUI Ketika kamera dinyalakan

4.2 Hasil Pengenalan Wajah

No	Kondisi gambar kedua gambar	Gambar
1	<ul style="list-style-type: none"> • Hasil benar • Pencahayaan sama • Letak mata, hidung, dan mulut hampir sama • Ekspresi mirip 	
2	<ul style="list-style-type: none"> • Hasil salah • Pencahayaan mirip • Letak mata, hidung, dan mulut hampir sama • Ekspresi mirip • Pose mirip 	

3	<ul style="list-style-type: none"> • Hasil benar • Pencahayaan sama • Letak mata, hidung, dan mulut hampir sama • Ekspresi mirip • Pose mirip 	
4	<ul style="list-style-type: none"> • Hasil benar • Pencahayaan berbeda • Letak mata, hidung, dan mulut hampir sama • Ekspresi mirip 	
5	<ul style="list-style-type: none"> • Hasil benar • Pencahayaan sama • Letak mata, hidung, dan mulut hampir sama • Ekspresi mirip 	

6	<ul style="list-style-type: none"> • Hasil salah • Tidak dapat mengenali wajah 	
---	--	--

Analisis :

Dari eksperimen di atas, terlihat bahwa beberapa hal yang sangat mempengaruhi pengenalan wajah pada metode eigenface. Pengenalan wajah akan menampilkan gambar yang memiliki pencahayaan, ekspresi, dan letak fitur wajah yang paling dekat dengan gambar tes. Akibat hal ini, model pengenalan wajah dapat mengenali wajah seseorang jika dan hanya jika terdapat gambar di dataset train yang mirip dengan gambar tes. Jika tidak terdapat gambar yang mirip dengan dataset tes, model pengenalan wajah cenderung menampilkan gambar yang salah. Untuk mengatasi hal tersebut, model pengenalan wajah menggunakan metode eigenface memerlukan dataset train yang sangat besar dan tiap orang di dataset tersebut memiliki pencahayaan, ekspresi, dan letak fitur wajah yang bermacam-macam supaya model pengenalan wajah ini dapat mengenali gambar tes dengan akurat.

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

5.1 Kesimpulan

Setelah melakukan pengerjaan dan pengujian, program, berbahasa python, *face recognition* dengan algoritma eigenfaces berhasil dibuat sesuai dengan materi yang sudah diajarkan pada mata kuliah IF2123 Aljabar Linier dan Geometri. Materi yang digunakan tersebut terdiri atas

1. Operasi Matriks dan Sifat-Sifatnya
2. Nilai Eigen dan Vektor Eigen
3. Dekomposisi QR Matriks
4. Algoritma Eigenface

Program yang dibuat berdasarkan materi tersebut berhasil diimplementasikan sesuai dengan spesifikasi yang tertera pada Tugas Besar 2 IF2123 Aljabar Linier dan Geometri dengan menggunakan algoritma Eigenface yaitu menyiapkan seluruh *training images* dalam sebuah himpunan matriks, kemudian cari nilai rata-ratanya dan hitung selisih nilai *training images* dan nilai *mean*. Langkah selanjutnya adalah menghitung matriks kovarian dari nilai matriks yang sudah dihitung sebelumnya dan hitung nilai nilai eigen dan vektor eigen dari matriks kovarian tersebut. Vektor eigen yang diperoleh disebut eigenface. Langkah terakhir adalah menghitung berat dari tiap gambar dengan cara mengalikan antara selisih nilai *training images* dan *mean* dengan eigenface hasil perhitungan sebelumnya. Untuk proses pengenalan wajah pada program, program akan mengurangi gambar tes dengan *mean*. Kemudian, weight dari gambar test dihitung dengan cara mengalikan hasil perhitungan sebelumnya dengan eigenface. Kemudian menghitung selisih Euclidian distance antara berat baru dengan weight yang ada pada *training images*. Jika nilai terdapat pada ambang tertentu maka program akan menyatakan gambar mirip dan begitu pula sebaliknya.

5.2 Saran

Melihat program yang dibuat, tim penulis memiliki beberapa saran mengenai pengerjaan Tugas Besar 2 Aljabar Linier yaitu:

1. Melihat masih ada beberapa hasil dari program yang menebak salah mengenai kemiripan wajah, penulis menyarankan untuk mengoptimalkan kembali algoritma yang digunakan untuk membandingkan apakah suatu wajah mirip atau tidak dan memperhitungkan faktor-faktor seperti pencahayaan, posisi, dan ekspresi wajah yang dapat membuat program membuat kesalahan dalam perhitungan.
2. Aplikasi dapat dikembangkan dengan memanfaatkan *machine learning* dengan menambahkan fitur feedback dari user yang menunjukkan apakah hasil dari program sesuai dengan gambar yang dicoba atau tidak. Hasil feedback tersebut dapat menjadi bahan *machine learning* agar program dapat bekerja lebih baik.

3. Pemrosesan gambar masih diubah menjadi *greyscale* dan tidak mengolah data RGB. Oleh karena itu, program dapat dikembangkan dengan mengolah data yang masih berbentuk RGB karena jika gambar diubah menjadi *greyscale* terlebih dahulu warna tidak akan menjadi faktor penentu wajah mirip atau tidak.

5.3 Refleksi

Dalam pembuatan tugas besar 2 Aljabar Linier dan Geometri ini, kami telah mempelajari dan mengeksplor lebih dalam terkait materi nilai dan vektor eigen, baik itu dari segi teoritis (melakukan dekomposisi QR) maupun implementasi, yaitu dalam pembuatan program pengenalan wajah ini. Selain hal-hal yang berkaitan dengan aljabar linear, kami juga menjadi lebih paham tentang bahasa pemrograman python, baik dari *library-library* atau module yang digunakan, seperti Tkinter, OpenCV, Mediapipe, dsb serta cara melakukan implementasi dengan menggunakan *library* atau module tersebut.

DAFTAR PUSTAKA

Anton, Howard, and Chris Rorres. Elementary Linear Algebra: Applications Version. John Wiley & Sons, 2010

Munir, Rinaldi. 2022. IF2123 Aljabar Geometri <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2022-2023/algeo22-23.htm>, diakses 13 November 2022 hingga 19 November 2022

<https://stackoverflow.com/>, diakses 13 November 2022 hingga 19 November 2022

<https://www.geeksforgeeks.org/>, diakses 13 November 2022 hingga 19 November 2022

https://mathinsight.org/matrix_vector_multiplication, diakses 17 November 2022 hingga 20 November 2022

<http://www.scholarpedia.org/article/Eigenfaces>, diakses 17 November 2022 hingga 20 November 2022

Jevtic, Dubravka R. Face Recognition Using Eigenface Approach. Serbian Journal of Electrical Engineering. Februari, 2022.

Lampiran

Link repository: <https://github.com/SaddamAnnais/Algeo02-21110>