

LAPORAN TUGAS BESAR 2
IF2123 ALJABAR LINIER DAN GEOMETRI
APLIKASI NILAI EIGEN DAN EIGENFACE PADA
PENGENALAN WAJAH



Kelompok Yuk Bisa Yuk:

Husnia Munzayana (13521077)
Austin Gabriel Pardosi (13521084)
Puti Nabilla Aidira (13521088)

Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2022

BAB I

DESKRIPSI MASALAH

Pengenalan wajah (*Face Recognition*) adalah teknologi biometrik yang bisa dipakai untuk mengidentifikasi wajah seseorang untuk berbagai kepentingan khususnya keamanan. Program pengenalan wajah melibatkan kumpulan citra wajah yang sudah disimpan pada database lalu berdasarkan kumpulan citra wajah tersebut, program dapat mempelajari bentuk wajah lalu mencocokkan antara kumpulan citra wajah yang sudah dipelajari dengan citra yang akan diidentifikasi. Terdapat berbagai teknik untuk memeriksa citra wajah dari kumpulan citra yang sudah diketahui seperti jarak Euclidean dan *cosine similarity*, principal component analysis (PCA), serta Eigenface.

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

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.

BAB II

TEORI SINGKAT

2.1. Matriks

Matriks adalah susunan bilangan berbentuk persegi panjang. Ordo matriks menyatakan banyaknya elemen baris dan kolom dari suatu matriks, sebagai contoh matriks A_{ij} menyatakan bahwa sebuah matriks memiliki i baris dan j kolom. Dalam pemrograman, matriks direpresentasikan sebagai *array* dua dimensi, $A[x][y]$ dengan x menyatakan baris dan y menyatakan kolom. Terdapat beberapa jenis matriks, antara lain :

- a. Matriks baris, yaitu matriks yang memiliki 1 baris
- b. Matriks kolom, yaitu matriks yang memiliki 1 kolom
- c. Matriks persegi, yaitu matriks yang memiliki jumlah baris dan kolom yang sama
- d. Matriks identitas, yaitu matriks persegi yang elemen pada diagonal utama bernilai 1 sedangkan elemen lain bernilai 0.
- e. Matriks segitiga atas, yaitu matriks persegi dengan elemen di bawah diagonal utama bernilai 0.
- f. Matriks segitiga bawah, yaitu matriks persegi dengan elemen di atas diagonal utama bernilai 0.
- g. Matriks ortogonal, yaitu matriks dengan matriks kolomnya saling ortogonal satu sama lain (hasil kali titik sama dengan 0).

Berbeda dengan operasi pada bilangan, operasi matriks memiliki aturannya tersendiri, antara lain :

- a. Penjumlahan dan Pengurangan matriks
Syarat : Kedua matriks memiliki ordo yang sama
Proses : Menjumlahkan atau mengurangi elemen pada matriks yang bersesuaian

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix}$$

- b. Perkalian matriks dengan skalar
Proses: Mengalikan setiap elemen matriks dengan bilangan skalar.

$$k \times \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} k \times a_{11} & k \times a_{12} \\ k \times a_{21} & k \times a_{22} \end{bmatrix}$$

- c. Perkalian dua matriks
Syarat : Ordo matriks pertama $i \times j$, dan ordo matriks kedua $j \times k$ sehingga dihasilkan

matriks perkalian $i \times k$

Proses : Mengalikan elemen pada baris matriks A dengan kolom matriks B, kemudian menjumlahkan hasil perkalian tersebut.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \\ = \begin{bmatrix} (a_{11} \times b_{11}) + (a_{12} \times b_{21}) + (a_{13} \times b_{31}) & (a_{11} \times b_{12}) + (a_{12} \times b_{22}) + (a_{13} \times b_{32}) \\ (a_{21} \times b_{11}) + (a_{22} \times b_{21}) + (a_{23} \times b_{31}) & (a_{21} \times b_{12}) + (a_{22} \times b_{22}) + (a_{23} \times b_{32}) \end{bmatrix}$$

d. Transpose matriks

Proses : Menukarkan baris matriks menjadi kolom matriks dan sebaliknya

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}^T = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \end{bmatrix}$$

e. Determinan, yaitu nilai yang dapat dihitung dari unsur-unsur suatu matriks persegi. Terdapat berbagai metode pencarian determinan, seperti dengan metode minor-kofaktor, aturan *sarrus*, atau menggunakan metode Operasi Baris Elementer (OBE) sehingga dihasilkan matriks segitiga.

2.2. Nilai Eigen dan Vektor Eigen

Nilai eigen menyatakan nilai karakteristik dari sebuah matriks persegi, sedangkan vektor eigen adalah vektor kolom tidak nol yang jika dikalikan dengan matriks persegi A akan menghasilkan vektor lain yang merupakan kelipatan dari vektor eigen itu sendiri, atau secara definisi dapat dinyatakan sebagai berikut :

Jika A adalah matriks persegi, maka vektor tidak nol x di R^n disebut vektor eigen dari matriks A jika memenuhi :

$$Ax = \lambda x$$

Dengan λ adalah nilai eigen dari A dan x adalah vektor eigen yang berkoresponden dengan λ .

Persamaan di atas dapat ditulis ulang sehingga menghasilkan persamaan :

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

Nilai eigen dan vektor eigen tersebut dapat dihitung dengan memanfaatkan persamaan karakteristik matriks yang dapat dicari dengan teorema persamaan karakteristik:

Jika A adalah matriks persegi persegi, λ adalah nilai eigen dari matriks A jika dan hanya jika memenuhi persamaan :

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

2.3. Eigenface

Eigenface adalah salah satu teknik untuk memeriksa citra wajah dari kumpulan citra yang telah diketahui. Proses pengenalan wajah dengan metode *eigenface* terbagi menjadi 2 tahap, yaitu tahap ekstraksi data set atau kumpulan citra dan tahap pengenalan wajah dari *testface*.

Proses ekstraksi dengan algoritma *eigenface* dilakukan melalui beberapa tahapan:

- Melakukan ekstraksi setiap kumpulan citra atau training face menjadi matriks kolom.
- Mencari nilai tengah sehingga membentuk averageface dari m gambar

$$\psi = \frac{1}{m} \sum_{n=1}^m \Gamma_n$$

- Menghitung selisih setiap training face dengan averageface.

$$\phi_i = \Gamma_i - \psi$$

- Menghitung nilai matriks *covarian*, mengukur sejauh mana dua variabel terkait secara linear.

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

$$L = A^T A$$

$$C = \phi_m^T \phi_n$$

- Menghitung nilai eigen dan vektor eigen dengan melakukan dekomposisi

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

- f. Mencari eigenface

$$\mu_i = \sum_{k=1}^m v_{ik} \phi_k$$

Tahap pengenalan wajah dilakukan melalui tahapan:

- a. Mencari nilai eigen dari test face

$$\mu_{new} = v \times \Gamma_{new} - \psi$$

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

- b. Mencari jarak terdekat antara nilai eigen training face dengan testface dengan metode euclidean distance

$$\varepsilon_k = \Omega - \Omega_k$$

- c. Menentukan gambar dari training image yang memiliki jarak euclidean terkecil dengan test face.

2.4. QR Decomposition

Dekomposisi matriks berarti memfaktorkan sebuah matriks sehingga menjadi hasil kali dari sejumlah matriks lain. Dekomposisi matriks dapat dilakukan dengan beberapa metode, antara lain metode dekomposisi LU, metode dekomposisi QR, dan metode dekomposisi nilai *singular* (*Singular Value Decomposition* – SVD). Metode dekomposisi QR berdasarkan proses Gram-Schmidt dapat menghasilkan faktorisasi matriks menjadi ortogonal matriks dan matriks segitiga. Hal ini sesuai dengan teorema *QR Decomposition Gram-Schmidt*:

Jika A adalah matriks m x n dengan setiap matriks kolom bersifat linier independen, maka A dapat difaktorkan menjadi

$$A = QR$$

dengan

Q = matriks ortogonal m x n, sehingga

R = matriks segitiga atas n x n

Metode *QR Decomposition Gram-Schmidt* dapat dimanfaatkan untuk menentukan nilai eigen dari sebuah matriks persegi A.

2.5. *Euclidean Distance*

Euclidean distance adalah perhitungan jarak dari dua buah titik dalam ruang *euclidean* untuk mempelajari hubungan antara sudut dan jarak. Jarak ini dapat direpresentasikan sebagai panjang segmen garis yang menghubungkan kedua titik tersebut. Euclidean ini masih ada kaitannya dengan teorema *pythagoras*. *Euclidean distance* dapat digunakan sebagai salah satu parameter yang merepresentasikan kemiripan antara dua buah citra atau gambar. Semakin kecil jarak *Euclidean* kedua gambar, maka semakin mirip kedua gambar tersebut, sehingga *Euclidean distance* ini dapat dimanfaatkan dalam proses pengenalan wajah atau *face recognition*. Persamaan untuk menghitung jarak *Euclidean* sebagai berikut:

$$d = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

BAB III

IMPLEMENTASI PROGRAM

A. File imageExt.py

Nama fungsi dan Argumennya	Tech/Kakas yang Digunakan	Gambaran Umum Algoritma
picExtract(dir)	PIL : .open, .ANTIALIAS	Melakukan ekstraksi gambar menjadi matriks
listOfPicExtract(dirpath)	glob os : .path.join Method dan fungsi dari standard library python: .append	Melakukan ekstraksi seluruh gambar di data set menjadi array of matriks.

B. File eigenFace.py

Nama fungsi dan Argumennya	Tech/Kakas yang Digunakan	Gambaran Umum Algoritma
convertOneRow(arr)	Method dan fungsi dari standard library python: .append, len	Mengubah array images m x n x n menjadi $(n^2) \times m$
nilaiTengah(arr)	Method dan fungsi dari standard library python: .append, len	Mencari averageface dari trainingface
selisih(arrmean, arr)	Method dan fungsi dari standard library python: .append, len	Mencari selisih antara training image dengan nilai tengah
covarian(A)	Numpy: .matmul, .transpose.	Mencari matriks covarian $C = \text{transpose}(A) \times A$
Q_i(Q_min, i, j, k)	-	(Fungsi pembantu QRDec) Menyusun matriks Q_t dengan mengganti elemen-elemen pada kiri atas Q_{\min} (matriks minor Q) dengan elemen-elemen matriks identitas.

QRDec(A)	<p>Numpy: .zeros, .identity, .sign, .sqrt, .matmul, .transpose.</p> <p>Struktur data tambahan: list, map.</p> <p>Lambda (fungsi anonim).</p> <p>Fungsi dari standard library python: len, sum.</p>	<p>Melakukan QR Decomposition dengan melakukan iterasi Householder Reflection pada minor matriks Q_t ke k sebanyak $(n-1)$ kali.</p> <p>Kemudian, Q didapat dari perkalian transpose Q_t pada setiap iterasi. Sedangkan R didapat dari perkalian Q_t pada setiap iterasi dikalikan A (nilai awal R).</p>
eigenvector(C)	<p>Numpy: .copy, .identity, .matmul, .diag, .empty, .shape, .hsplit, .concatenate, .vsplit</p> <p>Scipy: .linalg.null_space.</p> <p>Fungsi dari standard library python: len.</p>	<p>Mencari eigen values dari matriks C dengan melakukan iterasi QR Decomposition sebanyak 1000 kali. Eigen values didapat dari elemen-elemen diagonal matriks hasil perkalian $R @ Q$ pada iterasi QR Decomposition terakhir.</p> <p>Selanjutnya, eigen vector didapat dari solusi persamaan $((I * \text{eigen_value}) - C) * x = 0$. Solusi persamaan dihitung dengan bantuan <code>scipy.linalg.null_space</code>.</p> <p>Selanjutnya dilakukan split & concatenate untuk menyaring vektor-vektor solusi yang valid (ukurannya > 0) dan merapihkan format matriks.</p>
calceigface(A, eigvecs)	<p>Numpy: .dot</p> <p>Fungsi dari standard library python: len.</p>	<p>Menghitung eigen face dengan mengalikan setiap eigen vector dari matriks covarian' dengan matriks A. (Hal ini karena kelompok kami menggunakan matriks covarian' $A^t * A$ bukan $A * A^t$ untuk keperluan efisiensi memori)</p>
reconstruct(eigface,	Numpy: .dot, .reshape,	Melakukan rekonstruksi

diff)	.transpose Method dan fungsi dari standard library python: .append, len.	untuk masing-masing <i>training image</i> dengan mengalikan (<i>dot product</i>) eigen face dengan masing-masing <i>normalized image</i> (diff[:, i]) kemudian mengalikannya lagi dengan eigen face. Selanjutnya, matriks ditambahkan dengan transpose rata-rata <i>training image</i> untuk mendenormalisasikannya.
projectquery(eigface, normquery)	Numpy: .dot, .reshape, .transpose Fungsi dari standard library python: len.	Melakukan rekonstruksi pada query image dari eigen face hasil training dengan algoritma yang sama dengan reconstruct.
eucdist(arr1, arr2)	Numpy: .dot, .abs, .sqrt, .min, transpose.	Mencari jarak euclidean dari 2 buah matriks arr1 dan arr2 dengan menghitung akar dari sum square elemen: transpose (abs(arr1 - arr2)) * abs(arr1 - arr2). Kemudian mengembalikan nilai elemen minimum dari matriks jarak euclidean tersebut.
findface(prq, rec_face, th)	Fungsi dari standard library python: len.	Mencari indeks training image yang memiliki jarak euclidean terdekat dengan hasil matriks hasil rekonstruksi query dengan treshold yang digunakan = $0.47 * (\text{Normalized Training Face Range})$. Jika jarak > treshold, image tidak ditemukan, mengembalikan indeks -1.

Program Utama

Tahapan Proses	Tech/Kakas yang Digunakan	Gambaran Umum Algoritma
----------------	---------------------------	-------------------------

Training Image	<i>(Menggunakan fungsi-fungsi di atas)</i>	<ul style="list-style-type: none"> - Mengekstrak gambar untuk membentuk matriks: <code>imageExt.listOfPicExtract(dir), np.array(convertOneRow(imageExt.arrPic))</code>. - Menghitung matriks rata-rata: <code>nilaiTengah(arr)</code>. - Menghitung matriks normalized image: <code>np.array(selisih(meann, arr))</code> - Menghitung matriks covarian: <code>covarian(diff)</code> - Menghitung eigen vector & eigen face: <code>eigenvector(cov), calceigface(diff, eigv)</code>. - Merekonstruksi training image: <code>reconstruct(eigface, diff)</code>. - Melakukan 'normalisasi' pixel value menjadi dalam rentang unsigned integer 0 -255 dengan rumus: $(arr - np.min(arr)) / (np.max(arr) - np.min(arr)) * 255$.
Testing (Query Image)	<i>(Menggunakan fungsi-fungsi di atas)</i>	<ul style="list-style-type: none"> - Mengekstrak gambar untuk membentuk matriks: <code>imageExt.picExtract("../test/queryface.jpg"), np.reshape(query, ((imageExt.size*imageExt.size), 1))</code>. - Menormalisasi query dengan rata-rata training image: <code>np.array(selisih(meann, query))</code>. - Merekonstruksi query image: <code>projectquery(eigface, normquery)</code>. - Melakukan 'normalisasi' pixel value menjadi dalam

		<p>rentang unsigned integer 0 -255 dengan rumus:</p> $\frac{(\text{arr}-\text{np.min}(\text{arr}))}{(\text{np.max}(\text{arr})-\text{np.min}(\text{arr}))} * 255.$ <ul style="list-style-type: none"> - Mencari indeks gambar dengan jarak euclidean terdekat: <code>idxclosestface = findface(prq, rec_face)</code>. - Jika gambar ditemukan (indeks $\neq -1$), gambar berwarna dengan indeks tersebut diakses kemudian dinormalisasi pixel valuenya (dengan rumus yang sama). Lalu, disimpan sebagai 'res.png' dalam folder 'test'. - Jika gambar tidak ditemukan (indeks = -1) men-set boolean <code>facenotfound</code> menjadi true.
Execution Time Counting	Time: .time	Mencatat waktu awal dan waktu akhir kemudian mengurangkannya untuk menghitung <i>execution time</i> yang disimpan dalam variabel <code>timetaken</code> .

C. File GUI.py

Nama fitur	Tech/Kakas yang Digunakan	Gambaran Umum Algoritma
<code>_init_(self)</code>	Tkinter, customtkinter, PIL,	<ul style="list-style-type: none"> - Membuat semua attribut yang akan ditampilkan di GUI - Mengatur letak posisi dari masing-masing attribut - Mengakses lokasi dari gambar yang akan ditampilkan pada attribut
<code>button_event(self)</code>	Tkinter, time, PIL	<ul style="list-style-type: none"> - Memulai GUI untuk

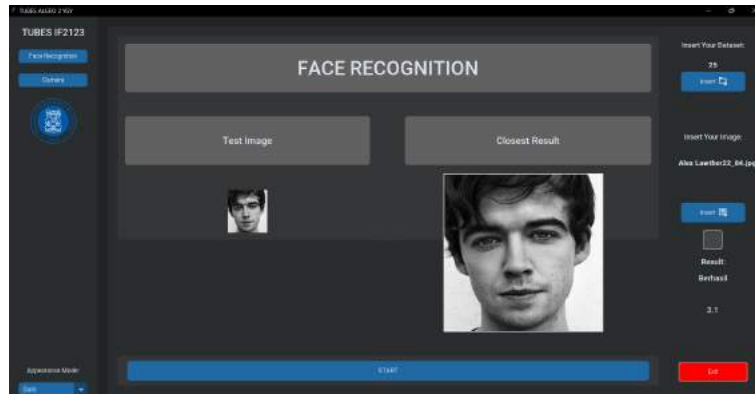
		<p>mencari citra wajah terdekat yang ada di dataset</p> <ul style="list-style-type: none"> - Menampilkan waktu execution time - Menampilkan citra wajah terdekat yang ada do dataset ke tampilan GUI
change_appereance_mode(self, new_apperance_mode)	Tkinter, customtkinter	<ul style="list-style-type: none"> - Mengganti tema dari tampilan GUI (bisa dark theme, bisa light theme).
load_image(self, path, image_size)	Tkinter, PIL	<ul style="list-style-type: none"> - Mengambil image yang akan digunakan untuk tampilan logo pada tampilan GUI (logo itb, dan logo pada button insert dataset, dan logo pada button insert image)
select_folder(self)	Tkinter	<ul style="list-style-type: none"> - Membuka file dialog untuk memilih folder dataset - Menampilkan nama folder dataset di tampilan GUI
select_picture(self)	Tkinter, PIL, customtkinter	<ul style="list-style-type: none"> - Membuka file dialog untuk memilih citra uji - Menampilkan citra uji hasil di tampilan GUI - Mengatur ukuran dari citra uji hasil untuk ditampilkan
result(self)	Tkinter, customtkinter	<ul style="list-style-type: none"> - Mengeluarkan text “Berhasil” atau “Tidak Berhasil” dari kegiatan test uji citra di tampilan GUI

BAB IV

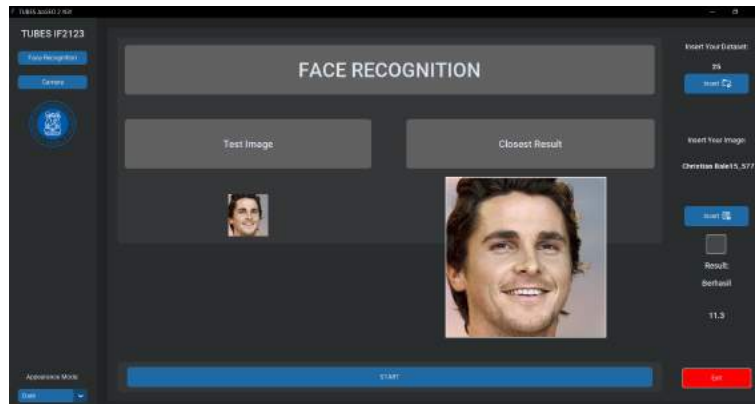
EKSPERIMEN STUDI KASUS

A. Variasi Ukuran Gambar (*Dataset: 25 training images*) Terhadap Waktu Eksekusi

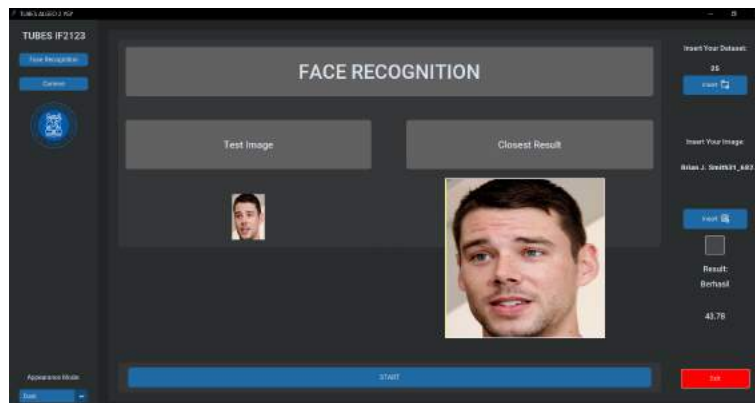
a. $256 * 256$



b. $512 * 512$

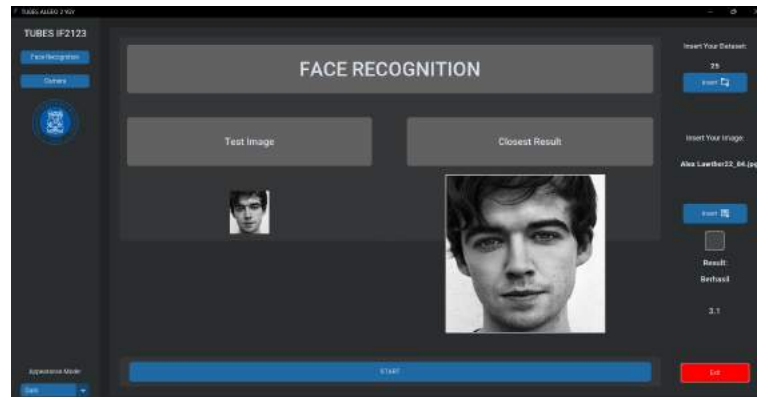


c. $1024 * 1024$

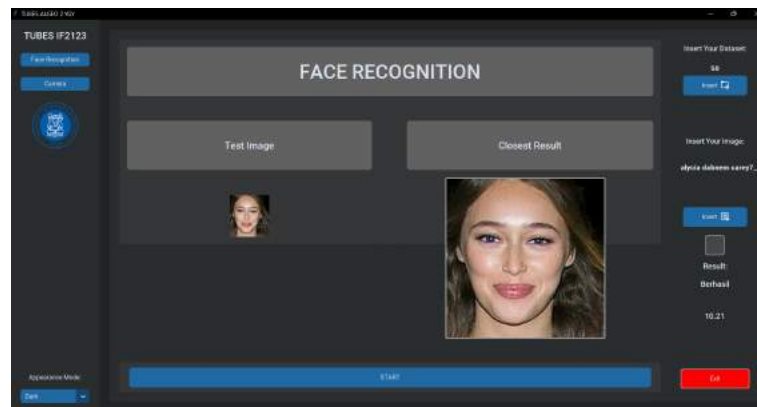


B. Variasi Banyak *Dataset* Terhadap Waktu Eksekusi

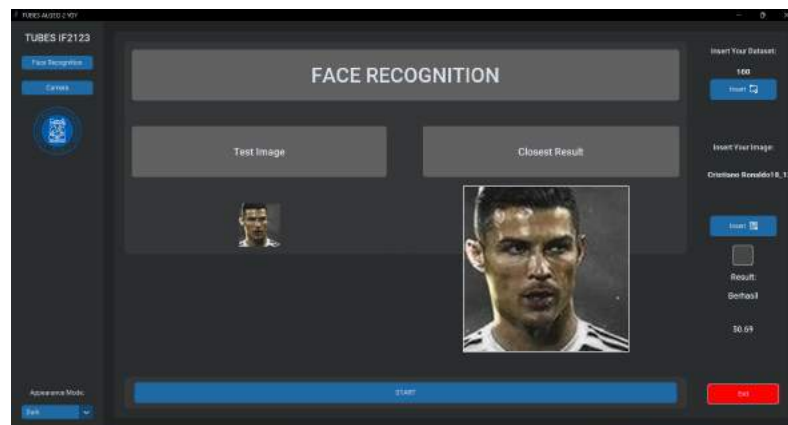
a. 25 training images



b. 50 training images



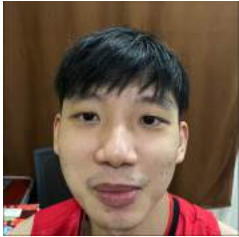













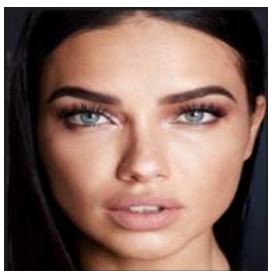
c. 100 training images


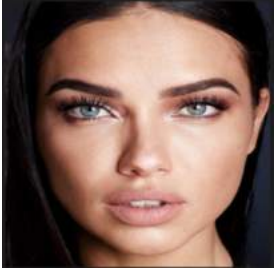
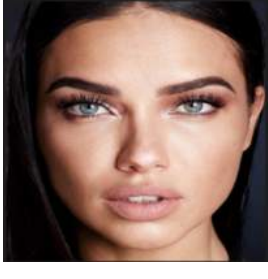





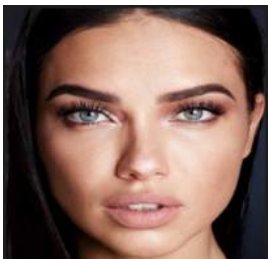


C. Akurasi

Data set I: dilakukan 5 pengetesan dengan query orang yang sama yang ada dalam *training images*.

Data set II: dilakukan 2 pengetesan dengan query orang yang sama & 1 pengetesan dengan query orang yang sama yang ada dalam *training images*.



<i>Query</i>	<i>Result</i>	<i>Desired Result</i>	Keterangan
			Correct
			Correct
			Correct
			False
			False





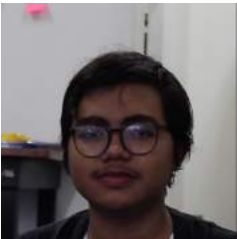





			Correct
			False
			False

D. Variasi *Threshold* terhadap *False Acceptance* dan *False Rejection*.

Dilakukan 6 pengetesan pada masing-masing : 3 orang yang ada dalam *training image* dan 3 orang yang tidak ada dalam *training image*.







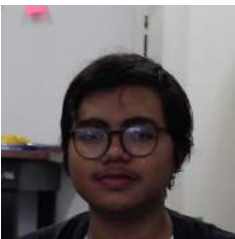



- a. $\text{Threshold} = 0.8 * (\text{Normalized Training Face Range})$



<i>Query</i>	<i>Result</i>	Keterangan
		True Accept

		True Accept
		True Accept
		False Accept
		False Accept
		False Accept

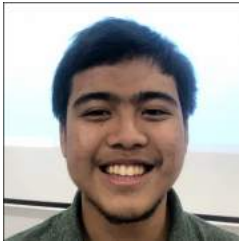





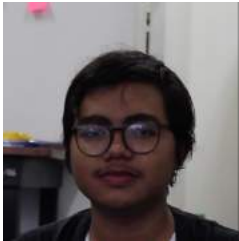
b. $\text{Threshold} = 0.5 * (\text{Normalized Training Face Range})$




<i>Query</i>	<i>Result</i>	Keterangan
--------------	---------------	------------

		True Accept
		True Accept
		True Accept
		False Accept
		False Accept

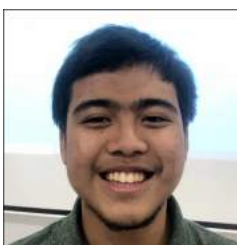

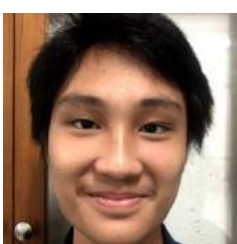


		False Accept
---	--	--------------

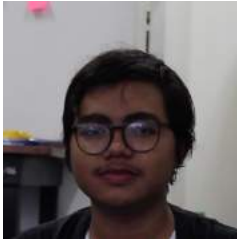



c. Threshold = $0.47 * (\text{Normalized Training Face Range})$

<i>Query</i>	<i>Result</i>	Keterangan
		True Accept
		True Accept
		True Accept
		True Reject

		True Reject
		False Accept

d. $\text{Threshold} = 0.3 * (\text{Normalized Training Face Range})$

<i>Query</i>	<i>Result</i>	Keterangan
		True Accept
		False Reject
		True Accept

		True Reject
		True Reject
		False Accept

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

A. Kesimpulan

Program Face recognition dengan metode eigenface yang di atas dibuat dengan mengimplementasikan materi perkuliahan IF2123 Aljabar Linier dan Geometri, seperti nilai eigen, vektor eigen, dekomposisi matriks, dan sebagainya. GUI diimplementasikan untuk mempercantik tampilan dan mempermudah penggunaan program. Pengguna dapat memasukkan folder yang berisi kumpulan citra training face dan file citra uji atau test face. Program kemudian akan melakukan pengolahan terhadap training face dan testface sehingga dihasilkan *average face*, *normalized image*, *reconstruct face*, hingga akhirnya dicari *euclidean distance* terkecil antara testface dengan training face pada dataset untuk menentukan hasil face recognition. Hasil tersebut kemudian ditampilkan ke layar.

Program dapat dijalankan pada variasi ukuran image dan variasi jumlah dataset. Semakin besar ukuran image dan semakin banyak jumlah image dalam dataset akan memakan waktu yang lebih lama. Dari hasil percobaan, jika digunakan testface yang terdapat pula pada folder dataset, didapatkan hasil yang tepat pada 3 dari 3 percobaan. Untuk percobaan dengan testface orang yang sama tetapi gambar yang berbeda dengan dataset, didapatkan hasil yang tepat pada 1 dari 5 percobaan. Untuk percobaan variasi *Threshold* terhadap *False Accept* dan *False Rejection*, didapatkan *Threshold* yang paling efektif digunakan sebesar $47\% \times \text{range normalized image}$.

B. Saran

Banyaknya referensi algoritma eigenface berbeda yang terdapat pada internet terkadang membuat bingung referensi mana yang harus diambil sehingga perlu dilakukan eksplorasi yang cukup lama untuk dapat menentukan algoritma mana yang akhirnya akan dipakai. Maka, lebih baik apabila diadakan proses asistensi untuk memastikan ketepatan referensi.

C. Refleksi

Melalui Tugas Besar IF2123 Aljabar Linier dan Geometri, kami telah banyak melakukan eksplorasi berbagai hal terkait GUI, ekstraksi gambar dalam bahasa Python, implementasi perhitungan algoritma eigenface dalam bahasa Python, dan sebagainya. Melalui tugas besar ini pula kami belajar untuk dapat bekerja sama dengan rekan kerja serta tidak menunda pekerjaan.

BAB VI

DAFTAR REFERENSI

1. <https://www.quantstart.com/articles/QR-Decomposition-with-Python-and-NumPy/>
2. https://github.com/ranriy/Face-Recognition-using-Eigenfaces/blob/master/Wasnik_04_01.ipynb
3. <https://numpy.org/doc/>
4. <https://docs.scipy.org/doc/scipy/>
5. <https://pillow.readthedocs.io/en/stable/reference/Image.html>
6. <https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/>
7. <https://akupintar.id/info-pintar/-/blogs/matriks-pengertian-operasi-determinan-invers-dan-contoh-soal>
8. <https://jagostat.com/aljabar-linear/nilai-eigen-dan-vektor-eigen>
9. <https://www.math.ucla.edu/~yanovsky/Teaching/Math151B/handouts/GramSchmidt.pdf>
10. <https://guzintamath.com/textsavvy/2019/02/02/eigenvalue-decomposition/>
11. <http://www.kitainformatika.com/2019/10/mengukur-jarak-euclidean-teori-dan.html>
12. <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2022-2023/algeo22-23.htm#SlideKuliah>

Link Repository :

<https://github.com/munzayanahusn/Algeo02-21077>

Link Video (Bonus Bagian B):

<https://youtu.be/1ctLeIGSEwc>