

IF2123-03 Aljabar Linier dan Geometri

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

LAPORAN TUGAS BESAR



Oleh

Eunice Sarah Siregar	13521013
Syarifa Dwi Purnamasari	13521018
Kenny Benaya Nathan	13521023

**Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
INSTITUT TEKNOLOGI BANDUNG
2022**

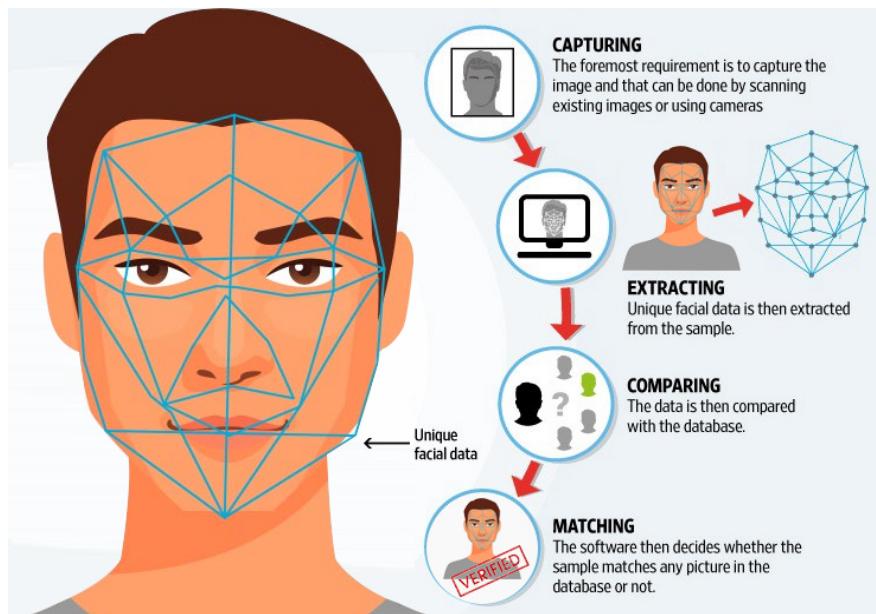
DAFTAR ISI

DAFTAR ISI	2
BAB I	3
DESKRIPSI MASALAH	3
BAB II	4
TEORI SINGKAT	4
2.1 Perkalian Matriks	4
2.2 Nilai Eigen dan Vektor Eigen	5
2.3 Eigenface Algorithm	5
BAB III	7
IMPLEMENTASI PROGRAM	7
3.1 Penjelasan Tech Stack	7
3.2 Penjelasan Garis Besar Algoritma Face Recognition	7
BAB IV	9
BAB V	10
DAFTAR REFERENSI	11

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



Gambar 1.1 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. Seperti namanya, matriks eigenface menggunakan eigenvector dalam pembentukannya.

BAB II

TEORI SINGKAT

2.1 Perkalian Matriks

Perkalian matriks hanya dapat dilakukan ketika jumlah kolom matriks pertama dan jumlah baris matriks kedua bernilai sama sehingga akan menghasilkan matriks yang baris dan kolomnya merupakan gabungan dari matriks pertama dan kedua. Secara formal, perkalian matriks dirumuskan sebagai,

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

Mengalikan matriks dapat dilakukan dengan mengalikan baris pertama dari matriks pertama dengan kolom pertama dari matriks kedua, lalu dijumlahkan seperti gambar berikut.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} p & q \\ r & s \end{pmatrix} = \begin{pmatrix} ap + br & aq + bs \\ cp + dr & cq + ds \end{pmatrix}$$

Gambar 2.1 Alur proses perkalian matriks (Sumber:
<https://suntimulanjari.blogspot.com/2019/09/matriks-operasi-matriks-penjumlahan.html>)

$$\begin{aligned} A \times B &= \begin{pmatrix} 8 & 9 & 6 \\ 7 & 4 & 3 \\ 5 & 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 3 & 2 \\ 4 & 6 & 5 \\ 7 & 9 & 8 \end{pmatrix} \\ &= \begin{pmatrix} 8.1 + 9.4 + 6.7 & 8.3 + 9.6 + 6.9 & 8.2 + 9.5 + 6.8 \\ 7.1 + 4.4 + 3.7 & 7.3 + 4.6 + 3.9 & 7.2 + 4.5 + 3.8 \\ 5.1 + 2.4 + 1.7 & 5.3 + 2.6 + 1.9 & 5.2 + 2.5 + 1.8 \end{pmatrix} \\ &= \begin{pmatrix} 8 + 36 + 42 & 24 + 54 + 54 & 16 + 45 + 48 \\ 7 + 16 + 21 & 21 + 24 + 27 & 14 + 20 + 24 \\ 5 + 8 + 7 & 15 + 12 + 9 & 10 + 10 + 8 \end{pmatrix} \\ &= \begin{pmatrix} 86 & 132 & 109 \\ 44 & 72 & 58 \\ 20 & 36 & 28 \end{pmatrix} \end{aligned}$$

Gambar 2.2 Contoh perkalian matriks (Sumber:
<https://www.antotunggal.com/2021/09/rumus-perkalian-matriks-dan-perkalian-skalar-matriks-lengkap.html>)

2.2 Nilai Eigen dan Vektor Eigen

Sebuah matriks persegi A memiliki ukuran $n \times n$. Vektor tidak nol v di R^n disebut sebagai vektor eigen dari A jika Av sama dengan perkalian vektor v dengan sebuah nilai skalar λ yang selanjutnya dapat disebut sebagai nilai eigen dari matriks A . Nilai eigen menyatakan nilai karakteristik dari sebuah matriks. Secara formal, nilai eigen direpresentasikan sebagai persamaan,

$$A \cdot v = \lambda \cdot v$$

A : matriks persegi ukuran $n \times n$

v : vektor eigen dari matriks A

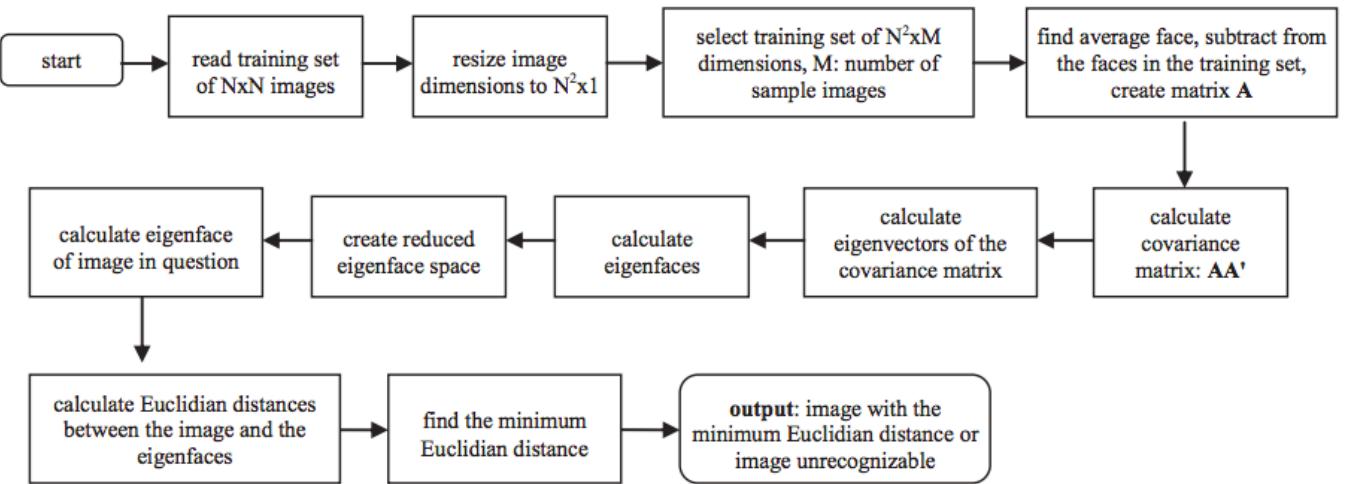
λ : nilai eigen dari matriks A

Vektor Eigen v memberikan matriks kolom yang apabila dikalikan dengan matriks persegi A akan menghasilkan vektor lain yang merupakan kelipatan dari vektor tersebut. Besar dari vektor tersebut tergantung dengan nilai eigen λ . Bisa memanjang ataupun menyusut. Bisa juga arahnya menjadi kebalikan dari vektor awal apabila nilai λ negatif, ataupun searah apabila nilai λ positif.

2.3 Eigenface Algorithm

Eigenface adalah salah satu metode *face recognition* berdasarkan *Principal Component Analysis* (PCA) yang datang dari sebuah jurnal yang ditulis oleh Turk dan Pentland pada tahun 1991. Pada dasarnya, *Eigenface* merupakan himpunan dari *eigenvector*.

Eigenface akan mengurangi dimensi dari foto wajah yang di-*input* agar lebih mudah untuk ditangani dengan menemukan pola wajah dari sekumpulan foto wajah (*training image*) yang ada dalam sebuah *dataset*. Pola tersebut dapat ditemukan dari PCA untuk menentukan *eigenfaces* pada subruang dataset tersebut. Semua foto dalam *dataset* tersebut akan direpresentasikan dalam sebuah kombinasi linier dari *eigenfaces*.



Gambar 2.3 Alur proses algoritma *Eigenface* (Sumber: *A Face Recognition System Based on Eigenfaces Method*, Sciverse Science Direct, Müge Çarıkçı and Figen Özén, 2011)

BAB III

IMPLEMENTASI PROGRAM

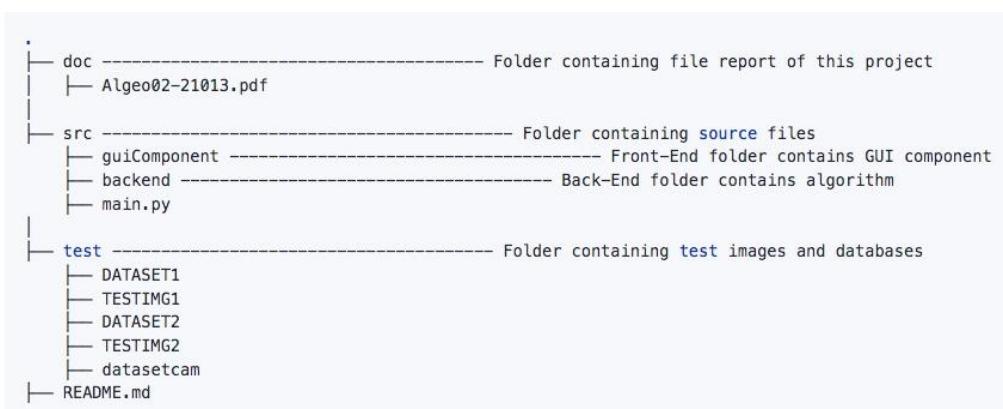
3.1 Penjelasan Tech Stack

Pada tugas besar ini, penulis menggunakan library *OpenCV*. *OpenCV* merupakan salah satu library dalam Python yang dimanfaatkan dalam pengolahan dan analisis isi suatu gambar. *OpenCV* pada tugas besar ini digunakan untuk membaca file gambar, mengubahnya menjadi matriks *grayscale*, dan menyimpan gambar hasil dari *face recognition*. Selain *OpenCV*, penulis juga menggunakan library *NumPy* yang menyediakan fungsi siap pakai untuk memudahkan dalam melakukan perhitungan saintifik matriks.

Di samping itu, penulis juga menggunakan library PIL atau *Pillow*. PIL adalah library *open-source* dalam bahasa Python yang digunakan untuk memanipulasi gambar. PIL diciptakan oleh Fredrik Lundh pada tahun 1995 dan pengembangannya dihentikan pada tahun 2011 kemudian di-fork dan diteruskan oleh library *Pillow*. Penulis menggunakan library *Pillow* untuk menampilkan gambar pada GUI hasil *face recognition* yang dibuat penulis.

3.2 Penjelasan Garis Besar Algoritma *Face Recognition*

Algoritma *face recognition* pada program penulis terletak pada folder ‘src’. Struktur *tree* dari program yang kami buat adalah sebagai berikut.



Gambar 3.1 *Tree Structure* Folder ‘src’ Program

Berdasarkan implementasi dan fungsionalitasnya, penulis membagi program menjadi 3 bagian yaitu bagian *guiComponent* yang berisi *frontend* program, bagian *backend* yang berisi *algorithm.py*, *qrDecomposition.py*, dan *euclideanDistance.py*, serta bagian *main.py* yang digunakan sebagai driver utama program.

a. Bagian *guiComponent*

Pada bagian GUI (Graphical User Interface) yang berinteraksi langsung dengan *user* ditulis dengan menggunakan bahasa Python serta menggunakan beberapa library. Program ini menggunakan library seperti Tkinter, Pillow, dan time. Elemen yang berinteraksi langsung dengan *user*, yaitu *button*, *text*, *file*. Pada program ini juga menerima input berisi *folder* dan *file* dengan jenis *file* *.jpg. Untuk menampilkan *execution time*, menggunakan *library* time dengan menampilkan durasi pemrosesan foto. Ukuran elemen, posisi elemen, dan *font text* juga dapat langsung diimplementasikan dengan Tkinter.

b. Bagian *backend*

Pada bagian *backend*, terdapat 3 program yaitu *algorithm.py*, *qrDecomposition.py*, dan *euclideanDistance.py*. Pada *algorithm.py* terdapat satu *function* yang dibuat yaitu *training_image* yang digunakan untuk mengolah folder *training image* yang digunakan sebagai dataset pencarian dalam *face recognition* serta program untuk mengolah *test image* yang diberikan sebagai input. Pada *qrDecomposition.py* terdapat 3 *function* yang dibuat yaitu fungsi untuk *QR Decomposition*, *householder*, dan *QR to Eigen*. Dekomposisi QR adalah salah satu alat dasar dalam komputasi statistika. Misalkan matriks X berukuran $n \times p$ mempunyai rang $d \leq p$. Komponen dari dekomposisi QR adalah matriks (q_{ik}) berukuran $n \times d$ dan matriks (r_{kj}) berukuran $d \times p$. Matriks X dapat ditulis sebagai berikut.

$$X = QR \text{ atau}$$

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} =$$

$$\begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1d} \\ q_{21} & q_{22} & \cdots & q_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ q_{n1} & q_{n2} & \cdots & q_{nd} \end{pmatrix} x$$

$$\begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1d} & \cdots & r_{1p} \\ 0 & r_{22} & \cdots & r_{2d} & \cdots & r_{2p} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{dd} & \cdots & r_{dp} \end{pmatrix}$$

atau

$$(x_1 \ x_2 \ \cdots \ x_p) = (q_1 \ q_2 \ \cdots \ q_d)(r_1 \ r_2 \ \cdots \ r_p)$$

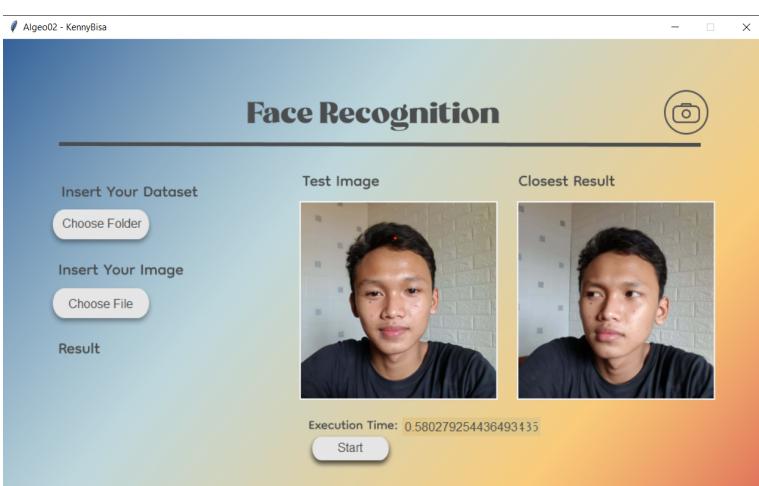
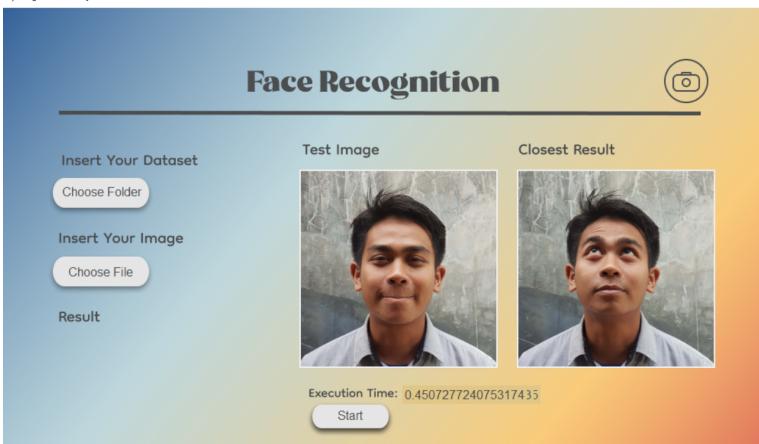
Gambar 3.2 Rumus *QR Decomposition*

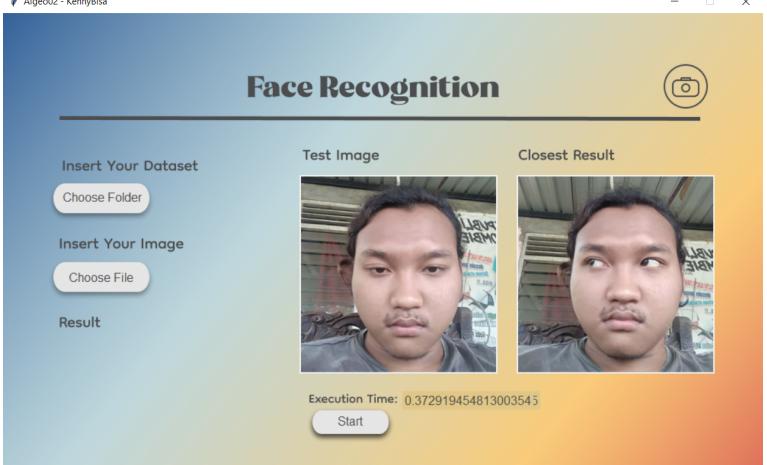
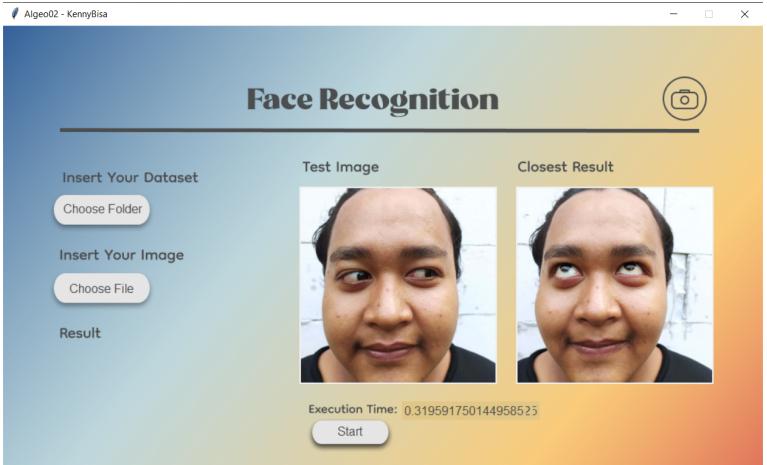
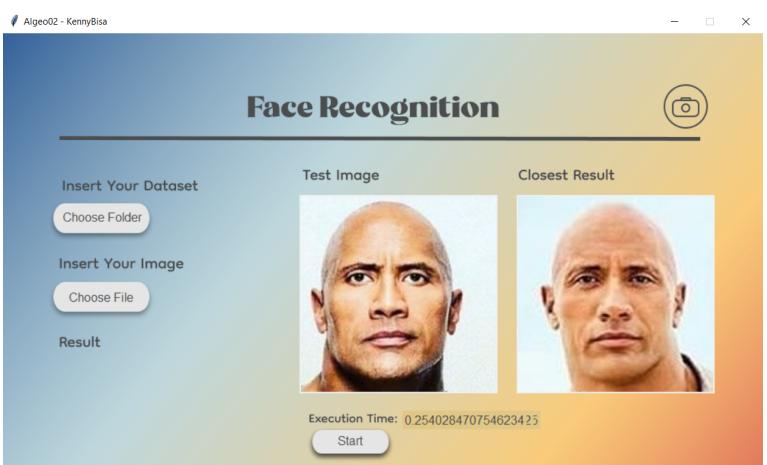
Pada *euclideanDistance.py* hanya terdapat satu *function* yaitu fungsi untuk menghitung jarak euclidean yang biasa diterapkan untuk membantu proses klasifikasi pada data mining.

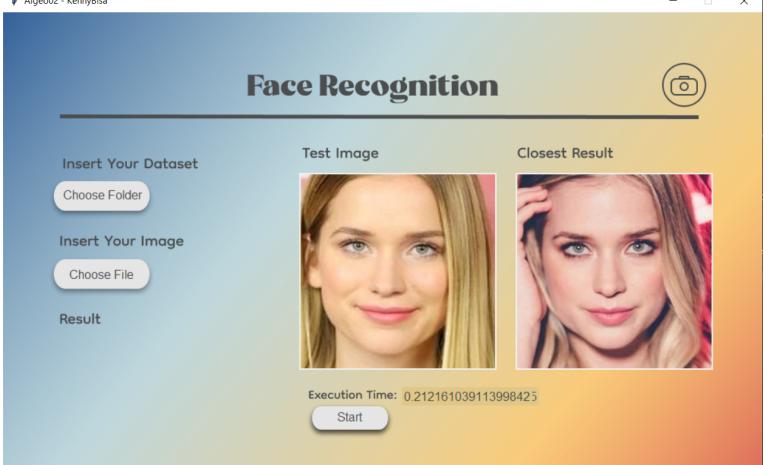
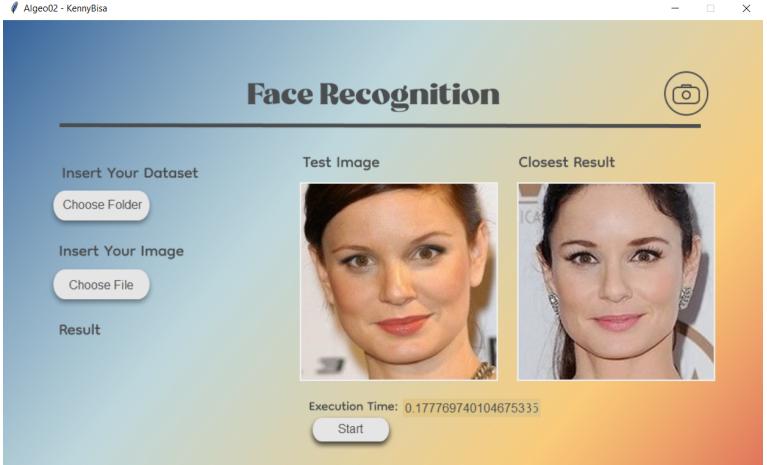
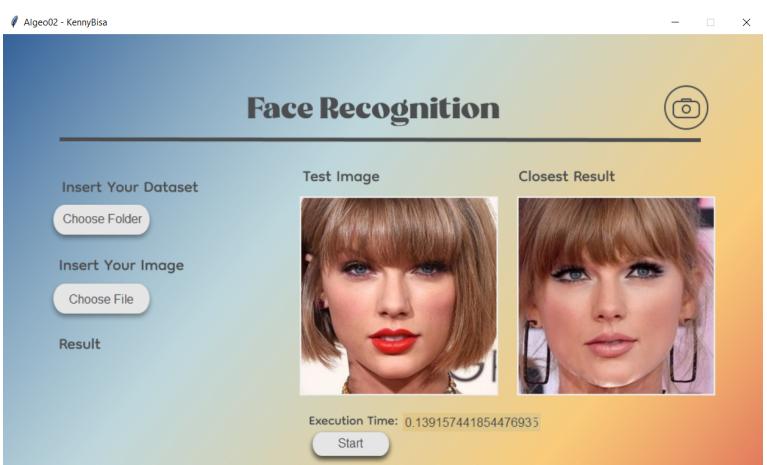
$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

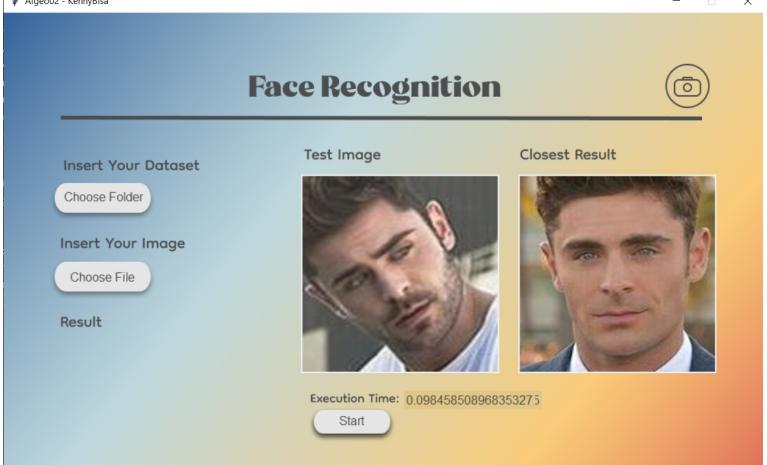
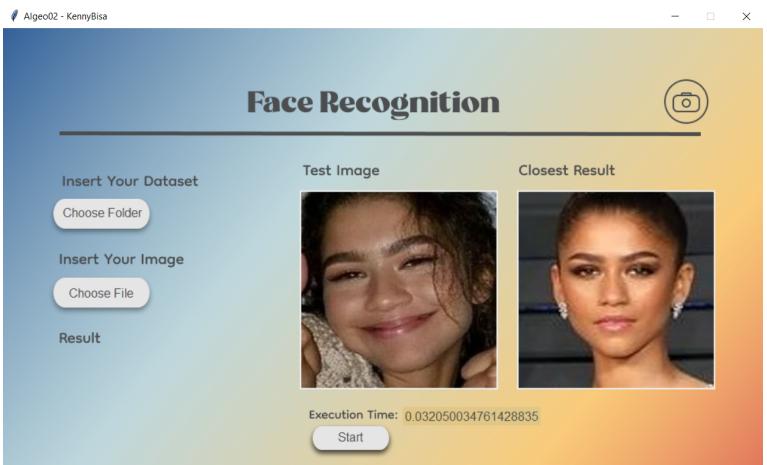
Gambar 3.3 Rumus *Euclidean Distance*

BAB IV EKSPERIMEN

No.	Gambar	Deskripsi
1.	 A screenshot of a Face Recognition application window titled "Face Recognition". The window has three main sections: "Insert Your Dataset" on the left with "Choose Folder" and "Result" buttons; "Test Image" in the center displaying a portrait of a man; and "Closest Result" on the right displaying another portrait of the same man. Below the images is the text "Execution Time: 0.580279254436493435" and a "Start" button.	<p>Dataset: 84 Foto 10 orang yang berbeda Distance: 6006456.10937843</p>
2.	 A screenshot of the same Face Recognition application window. The "Test Image" section shows a portrait of a man looking slightly to the side. The "Closest Result" section shows a portrait of the same man looking upwards. Below the images is the text "Execution Time: 0.450727724075317435" and a "Start" button.	<p>Dataset: 84 Foto 10 orang yang berbeda Distance: 4477444.753075063</p>

3.		<p>Dataset: 84 Foto 10 orang yang berbeda Distance: 4375313.478570745</p>
4.		<p>Dataset: 84 Foto 10 orang yang berbeda Distance: 8311313.273351287</p>
5.		<p>Dataset: 105 Foto 105 orang yang berbeda Distance: 15484935.623858808</p>

6.		Dataset: 105 Foto 105 orang yang berbeda Distance: 11155175.282749655
7.		Dataset: 105 Foto 105 orang yang berbeda Distance: 12949911.120884879
8.		Dataset: 105 Foto 105 orang yang berbeda Distance: 11633329.564857062

9.		Dataset: 105 Foto 105 orang yang berbeda Distance: 12082679.672876323
10.		Dataset: 105 Foto 105 orang yang berbeda Distance: 15352132.764228461

BAB V

KESIMPULAN

Setiap wajah dapat direpresentasikan dengan matriks. Dari matriks tersebut didapatkan sebuah matriks Eigenface yang berisikan kumpulan eigenvector. Menggunakan perkalian dasar matriks, akan didapatkan eigen value yang nantinya dipakai untuk mendapatkan eigenvector. Dengan menggunakan bahasa pemrograman Python didapatkan eigenface dengan selisih jarak terdekat.

Penulis menyarankan untuk membuat kode lebih rapi dan tertata lagi. Selain itu, disarankan untuk lebih panjang lagi deadline yang diberikan agar tugas dapat diselesaikan lebih baik lagi.

DAFTAR REFERENSI

<https://jurnal.untan.ac.id/index.php/jcskommipa/article/download/9727/9500>

<http://laid.delanover.com/explanation-face-recognition-using-eigenfaces/>

<https://github.com/vutsalsinghal/EigenFace/blob/master/Face%20Recognition.ipynb>

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

<https://fmipa.unmul.ac.id/files/docs/4.%20Rito.pdf>

Link Github: <https://github.com/kennypanjaitan/Algeo02-21013.git>