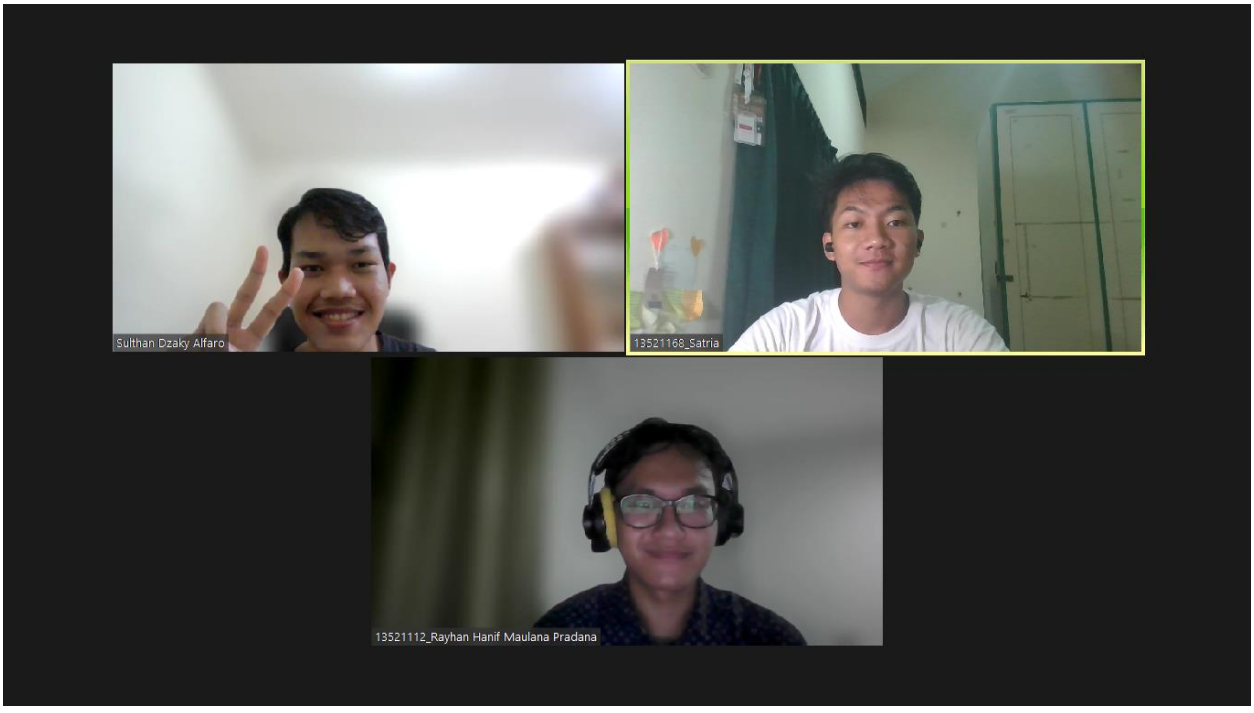
TUGAS BESAR 2 IF2123
ALJABAR LINIER DAN GEOMETRI
2022/2023

Oleh

Rayhan Hanif Maulana Pradana	13521112
Sulthan Dzaky Alfaro	13521159
Satria Octavianus Nababan	13521168



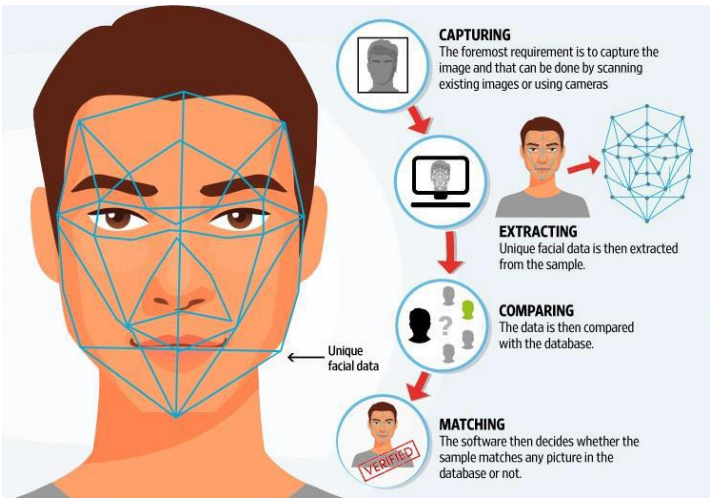
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

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



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. Berikut merupakan langkah rinci dalam pembentukan eigenface.

Algoritma Eigenface:

1. Langkah pertama adalah menyiapkan data dengan membuat suatu himpunan S yang terdiri dari seluruh training image, (Γ1, Γ2, …, ΓM)

$$S = (\Gamma 1, \Gamma 2, \ldots, \Gamma M) \quad (1)$$

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 (Γi) dengan nilai tengah (Ψ)

$$\phi i = \Gamma i - \Psi \quad (3)$$

4. Langkah keempat adalah menghitung nilai matriks kovarian (C)

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

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

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

$$C \text{ x } vi = \lambda i \text{ x } vi \quad (5)$$

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

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

$l = 1, 2, \ldots, M$

Tahapan Pengenalan wajah :

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

$$\mu new = \nu \text{ x } \Gamma new - \Psi \quad (7)$$

2. Gunakan metode euclidean distance untuk mencari jarak (distance) terpendek antara nilai eigen dari training image dalam database dengan nilai eigen dari image testface.

$$\varepsilon k = \Omega - \Omega k \quad (8)$$

$\Omega = \mu 1 , \mu 2 , \dots , \mu M$

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
LANDASAN TEORI

2.1 Perkalian Matriks

Matriks merupakan suatu susunan bilangan yang memiliki bentuk persegi atau persegi panjang yang terdiri dari baris dan kolom, Salah satu operasi dasar dari matriks yaitu perkalian matriks. Dalam perkalian antara 2 matriks,ada syarat tertentu untuk menggunakan operasi ini. Syaratnya adalah jumlah kolom pada matriks yang pertama harus sama dengan jumlah baris matriks yang kedua sehingga menghasilkan matriks yang memiliki jumlah baris matriks yang pertama dan jumlah kolom matriks yang kedua.

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

Untuk prosedur perkalian matriks,berikut cara perkalian 2 matriks:

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

$c_{ij} = a_{i1}b_{1i} + a_{i2}b_{2i} + \cdots + a_{in}b_{nj}$

Contoh :

A = $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

B = $\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$

$$A \times B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 1\times 1 + 2\times 3 + 3\times 5 & 1\times 2 + 2\times 4 + 3\times 6 \\ 4\times 1 + 5\times 3 + 6\times 5 & 4\times 2 + 5\times 4 + 6\times 6 \end{bmatrix} = \begin{bmatrix} 22 & 28 \\ 49 & 64 \end{bmatrix}$$

Adapun terdapat perkalian matriks dengan skalar. Contoh:

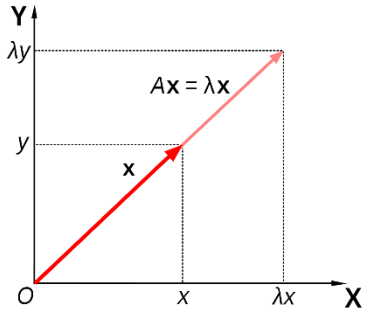
$$2A = 2 \times \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \end{bmatrix}$$

2.2 Nilai Eigen dan Vektor Eigen

Kata Eigen berasal dari bahasa Jerman yang memiliki arti asli atau karakteristik. Dengan demikian, nilai eigen dapat diartikan nilai karakteristik dari sebuah matriks persegi.

Misalkan terdapat matriks A yang berukuran a x a maka vektor tidak nol x di R^n disebut vektor eigen dari A jika terdapat perkalian Ax yang sama dengan perkalian suatu skalar λ dengan x, yaitu Ax = λx. Skalar λ disebut nilai eigen dari A, dan x disebut dengan vektor eigen yang berhubungan dengan λ.

Vektor eigen x merepresentasikan vektor kolom yang apabila dikalikan dengan sebuah matriks persegi akan menghasilkan vektor lain yang merupakan kelipatan vektor itu sendiri.



Untuk menentukan nilai eigen dan vektor eigen dari suatu matriks A yang berukuran a x a, dapat dihitung melalui persamaan karakter dari matriks tersebut yaitu det(λI-A)=0 dengan λ nilai eigen dan I adalah matriks identitas. Melalui persamaan tersebut, didapat akar-akar persamaan λ yang disebut dengan nilai-nilai eigen. Setelah mendapat nilai eigen, kita dapat menentukan eigen vektor dengan mendistribusikan nilai nilai tadi kedalam persamaan λI-A=0.

Contoh:

Misal A = $\begin{bmatrix} 1 & -2 \\ 1 & 4 \end{bmatrix}$

a.) Nilai eigen

det(λI-A)=0 → $\begin{vmatrix} \lambda - 1 & 2 \\ -1 & \lambda - 4 \end{vmatrix} = 0 \rightarrow (\lambda - 1)(\lambda - 4) + 2 = 0 \rightarrow (\lambda - 2)(\lambda - 3) = 0$

didapat nilai eigen λ=2 atau λ=3

b.) Vektor Eigen

$\lambda = 2 \rightarrow \begin{bmatrix} 1 & 2 \\ -1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow solusi\ x_1 = -2x_2, x_2 = t, t \in R$

Vektor eigen : x: $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -2t \\ t \end{bmatrix} = t \begin{bmatrix} -2 \\ 1 \end{bmatrix}$

$\lambda = 3 \rightarrow \begin{bmatrix} 2 & 2 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow solusi\ x_1 = -x_2, x_2 = t, t \in R$

Vektor eigen : x: $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -t \\ t \end{bmatrix} = t \begin{bmatrix} -1 \\ 1 \end{bmatrix}$

2.3 Eigenface

Eigenface merupakan suatu algoritma yang menggunakan Principal Component Analysis (PCA) untuk mengurangi dimensionalitas serta untuk mencari vektor terbaik guna mendistribusikan citra wajah ke dalam ruang wajah yang ada. Tujuan utama dari PCA adalah untuk mencari vektor yang paling cocok yang dapat menggambarkan distribusi citra wajah di dalam ruang citra ke dalam ruang wajah. Untuk membentuk principal component sejumlah m eigenvector digunakan sesuai dengan distribusi eigenvalue. Eigenvector dan eigenvalue didapatkan dari matriks kovarian yang di-generate dari citra wajah yang dilatihkan. Eigenvector diurutkan berdasarkan eigenvalue-nya (tinggi ke rendah) dan dipilih sebanyak m eigenvector pertama untuk membentuk principal component. Eigenface pada dasarnya adalah himpunan eigenvector. Eigenvector diturunkan dari sebuah matriks kovarian dari ruang vektor yang mungkin dari wajah-wajah manusia. Untuk mendapatkan Eigenface, sejumlah himpunan gambar citra wajah diambil di bawah kondisi pencahayaan yang sama kemudian dilakukan normalisasi ke ukuran resolusi pixel yang sama. Eigenface kemudian diekstrak dari data citra dengan menggunakan PCA. PCA dikenal sebagai teknik untuk mereduksi dimensi dengan mentransformasi sejumlah variabel yang berkorelasi ke dalam sejumlah kecil variabel yang tidak berkorelasi. Eigenface akan tampak seperti area gelap dan terang yang disusun ke dalam pola tertentu.

Eigenface dihitung dengan mengalikan nilai selisih antar citra A dengan nilai vektor eigen V. Matriks E
dapat dinyatakan sebagai :

E = A V

Keterangan :
E : Matriks Eigenface
A : Matriks Selisih Antarcitra
V : Matriks vektor eigen

Algoritma Eigenface dapat merepresentasikan wajah dengan baik ke dimensi yang lebih rendah dari dimensi citra. Penggunaan algoritma Eigenface dan Euclidean Distance pada proses klasifikasi memberikan tingkat akurasi sebesar 91 % pada pengenalan sejumlah data. Algoritma Eigenface dapat meringankan proses komputasi pada pengenalan sejumlah besar data wajah karena menggunakan metode yang dapat mengurangi dimensi citra.

BAB III
IMPLEMENTASI PROGRAM DALAM PYTHON

Dalam pembuatan aplikasi mengenai face recognition, kami menggunakan bahasa Python dalam pembuatan algoritma Eigen Vector dan Eigen Value dan face recognition. Matriks pixel direpresentasikan sebagai array dua dimensi dan pengolahan citra dilakukan dengan library OpenCV Python. Selain itu kami menggunakan library TKinter, Pillow untuk tampilan dalam pembuatan aplikasi desktop serta library numpy untuk mempermudah pemrosesan matriks. Struktur program utama kami terdiri dari 5, yaitu GUI.py, eigen2.py, extracdata_func.py dan extractor_func.py, kofarian.py, dan eigendistance.py

3.1 GUI.py

File ini berisi program untuk menampilkan aplikasi ke layar desktop. Pada file ini kami menggunakan library Pillow dan Tkinter untuk menampilkan aplikasi sekaligus menampilkan gambar pada aplikasi tersebut.

File ini juga berisi fungsi-fungsi dalam pemorsesan gambar menjadi sebuah matriks dengan menggunakan file extractor_func.py dan extracdata_func.py. Untuk pengolaham matriks kita menggunakan fungsi seperti eigenVector,eigenface,kofarian,pencarian kemiripan wajah(dalam eigendistance.py fungsinya new_image) untuk mencari kemiripan wajah.

Untuk menggunakan aplikasi ini,pertama tama kita masukkan dataset training untuk sebagai acuan dalam mencari kemiripan. Setelah itu, input gambar yang akan dicari kemiripannya. Lalu program akan mencarinya dengan pengolahan matriks dengan menggunakan fungsi fungsi seperti eigenVector,eigenface,kofarian,vektorberat,dan lain lain. Setelah mendapatkan vektor berat, lalu dicari jarak euclidian terkecil untuk mencari kemiripan wajah.

3.2 extracdata_func.py dan extractor_func.py

File ini berisi fungsi fungsi untuk memproses dataset menjadi array matriks foto dan memproses gambar menjadi sebuah matriks. Untuk pemrosesan dataset, pertama tama gambar yang ada di dataset akan diubah menjadi gambar abu-abu untuk mempermudah perhitungan nanti. Lalu program akan mengubah gambar NxN menjadi sebuah matriks. Lalu matriks ini akan dijadikan 1 kolom dengan banyak row N^2/vektor. Lalu matriks akan dimasukkan ke dalam array dan seterusnya untuk gambar selanjutnya. Begitu pula untuk pemrosesan gambar.

3.3 kofarian.py

File ini berisi fungsi untuk mengubah array matriks foto menjadi kofarian. Pertama tama mencari rata rata dengan cara menambah semua matriks foto lalu dibagi dengan jumlah foto. Lalu matriks matriks foto yang ada di array tadi masing masing matriks akan dikurangi dengan rata rata yang didapat. Setelah itu, matriks matriks ini digabung menjadi 1 matriks,kita misalkan matriks A dengan dimensi $N^2 \times M$ dengan N^2 panjang vektor gambar dan M banyaknya foto. Setelah itu menggunakan perkalian matriks,kita kalikan matriks A^T dengan A lalu menghasilkan matriks $M \times M$ yang selanjutnya digunakan untuk mencari eigenVector.

3.4 eigen2.py

File ini berisi fungsi untuk mencari nilai eigen serta vektor eigen dari matiks kovarian yang didapat dari hasil extract foto yang hendak diuji. Pencarian nilai eigen dan vektor eigen diimplementasikan dengan metode dekomposisi QR yang menggunakan prosedur Gram-Schmidt. Program menerima input berupa matriks kovarian $M \times M$ yang kemudian akan didekomposisi dengan metode dekomposisi QR. Untuk mendapatkan nilai-nilai eigen, dekomposisi QR dilakukan berulang-ulang sebanyak sepuluh kali, dimana elemen-elemen pada diagonal utama matriks R hasil dekomposisi terakhir merupakan nilai-nilai eigen dari matriks kovarian. Eigen vektor didapat dari kolom-kolom matriks Q hasil dekomposisi terakhir.

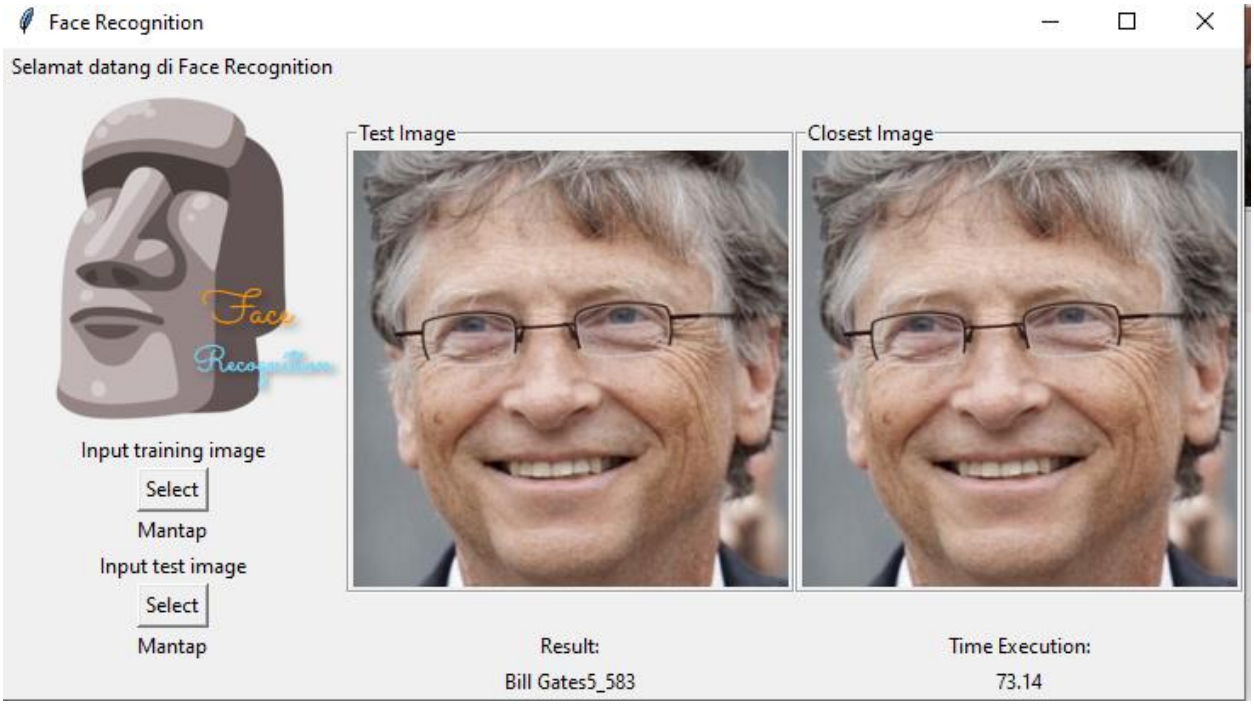
3.5 eigendistance.py

File ini berisi fungsi untuk mencari panjang euclidian dari matrix foto yang kita uji dengan training dataset. Pertama tama mencari eigenface dengan mengalikan hasil matriks A yang ada di kofarian.py dengan eigenvector dari hasil perkalian A^T dengan A. Setelah itu,kita proyeksikan masing masing matriks foto dengan eigenface untuk mencari berat vektor dari masing masing foto. Setelah itu inputkan gambar yang akan ditest lalu diubah menjadi matriks lalu kita cari vektor berat dengan memproyeksikannya dengan eigen face yang tadi sudah didapatkan lalu setelah itu kita badingkan dengan setiap vektor berat masing masing foto training dengan cara mengurangi vektor berat gambar test dengan vektor berat tiap matriks foto lalu cari nilai yang paling terkecil untuk mendapatkan gambar dengan kemiripan terdekat.

BAB IV

EXPERIMENT

1.Tampilan apabila ada kemiripan

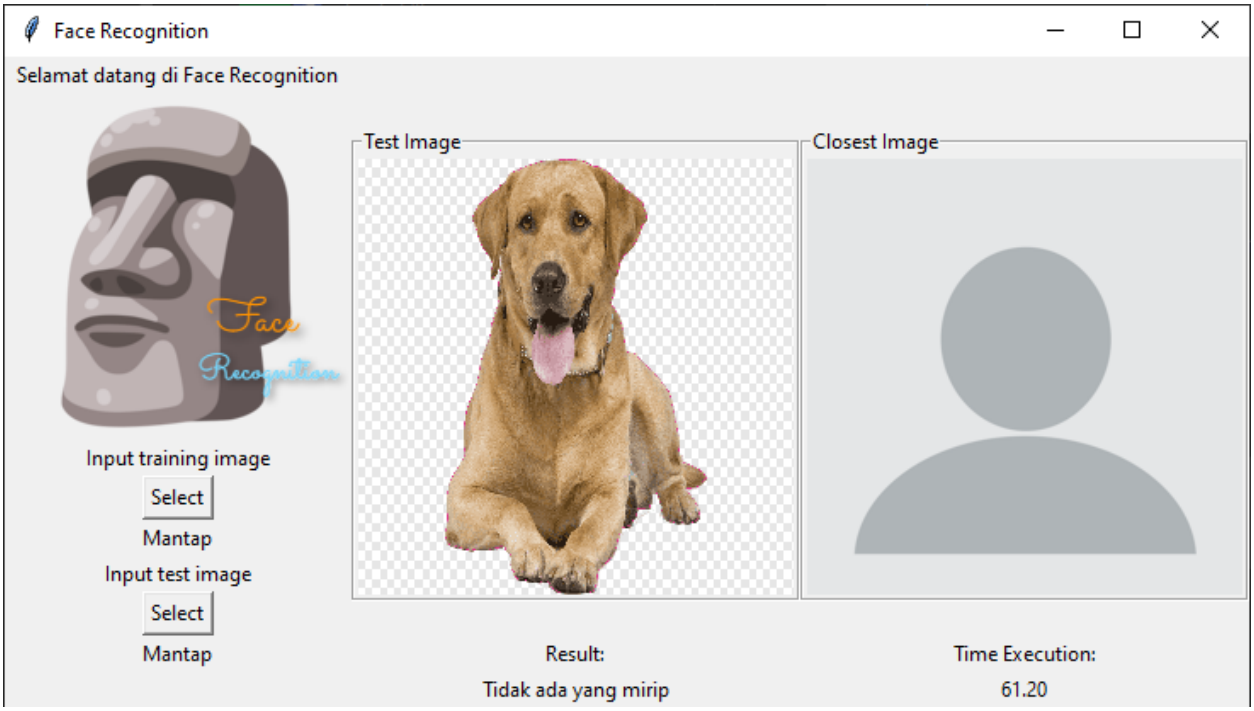


Pada eksperimen pertama menggunakan training *image* sebanyak 27 buah, yang mana menghasilkan hasil yang persis karena *input test image* yang kita berikan diambil dari folder *dataset training image* kita, dapat dilihat juga pada *time executionnya* selama 73.14 detik.

2.Tampilan apabila tidak ada yang mirip

Pada eksperimen kedua menggunakan *training image* sebanyak 25 buah, dan *input test image* yang ukurannya berbeda, program akan mengubah ukuran *input test image* menjadi persegi yaitu 100 x 100 pixel, dan dapat kita lihat bahwa hasil eksperimennya tidak ada foto yang mirip dari *training image* kita, dengan *time excetion-nya* selama 44.88 detik.

3.Tampilan apabila gambar yang diinputkan bukan wajah seseorang



Pada eksperimen ketiga dengan input training image yang sama sebanyak 25 buah, dan input test image nya berupa foto hewan yang bukan wajah manusia sudah pasti memberikan hasil yang tidak ada kemiripan karena pada input training image kita tidak terdapat foto hewan.

BAB IV

Kesimpulan, Saran, dan Refleksi

5.1 Kesimpulan

Pada tugas kali ini,kami berhasil membuat program aplikasi desktop yang dapat melakukan face recognition pada gambar yang diinputkan. Kekurangan dari aplikasi ini yaitu, ada beberapa syarat foto yang digunakan, seperti dimensi foto harus persegi,foto orang yang ada di gambar harus terlihat muka. Hal-hal ini yang membuat face recognition tidak begitu akurat.

5.2 Saran

Karena untuk mencari eigenvector sangatlah rumit, untuk cara yang lebih sederhana, disarankan menggunakan metode QR untuk mendapatkan eigenvector dan eigenvalue. Walaupun proses eksekusinya cukup lama, tetapi hasilnya cukup akurat.

Untuk hasil yang lebih akurat, maka disarankan untuk memperbanyak dataset training image, dan juga dapat memperbesar batasan dari eigendistance nya.

5.3 Refleksi

Dalam tugas besar kedua Aljabar Linear dan Geometri ini, kami telah mempelajari dan memahami salah satu metode face recognition pada gambar dengan menerapkan materi yang telah dipelajari di dalam perkuliahan yaitu Nilai Eigen dan Vektor Eigen. Kami juga menambah pengetahuan tentang pembuatan aplikasi desktop yang dapat menjadi skill tambahan dan pengantar perkuliahan selanjutnya.

DAFTAR REFERENSI

https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2022-2023/algeo22-23.htm

https://en.m.wikipedia.org/wiki/Eigenface

https://www.neliti.com/publications/135138/pengenalan-wajah-menggunakan-algoritma-eigenface-dan-euclidean-distance

https://medium.com/machine-learning-world/feature-extraction-and-similar-image-search-with-opencv-for-newbies-3c59796bf774

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

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

https://www.andreinc.net/2021/01/25/computing-eigenvalues-and-eigenvectors-using-qr-decomposition

https://www.math.ucla.edu/~yanovsky/Teaching/Math151B/handouts/GramSchmidt.pdf

https://numpy.org/doc/stable/reference/generated/numpy.swapaxes.html

Lampiran

Link Repository : https://github.com/rayhanp1402/Algeo02-21112

Link video demo : https://youtu.be/v9gYrnXBM_w