

Tugas Besar 2 IF2123

Aljabar Linear dan Geometri



**Kelompok
“Cape Face”**

Semester I Tahun 2022/2023

13521123 William Nixon

13521148 Johanes Lee

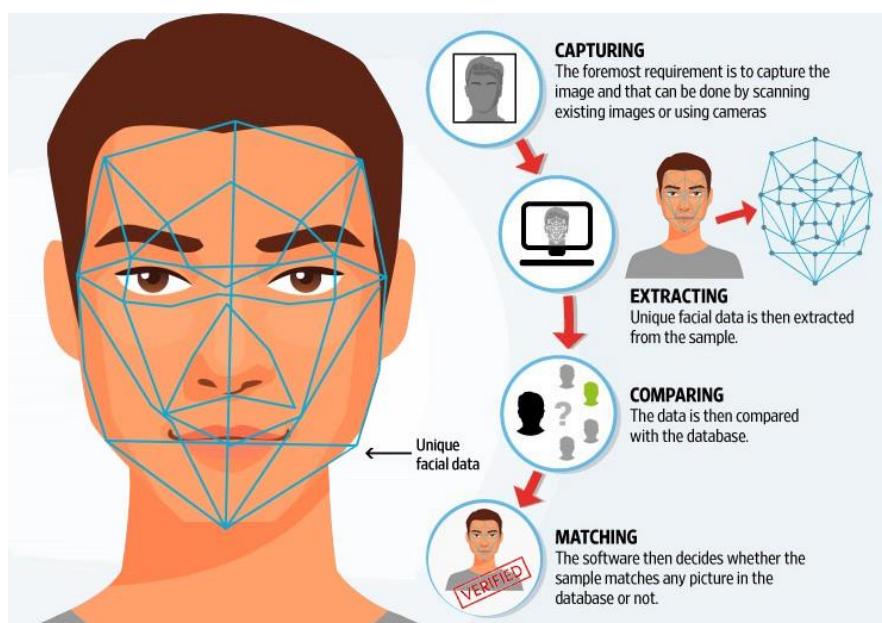
13521170 Haziq Abiyyu Mahdy

Bab I

Deskripsi Masalah

1.1 Abstraksi

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. Seperti namanya, matriks eigenface menggunakan eigenvector dalam pembentukannya. Berikut merupakan langkah rinci dalam pembentukan eigenface.

1.2 Spesifikasi

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/herveisburak/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.

Bonus:

Bagian A (Kamera)

1. Terdapat fitur kamera yang dapat mengenali wajah secara *realtime* menggunakan *webcam* ketika program dijalankan.
2. Teknis pengenalan wajah melalui kamera dibebaskan oleh pembuat program. (contoh: program dieksekusi setiap 10 detik sekali).
3. Fitur kamera merupakan fitur **tambahan**, fitur utama upload gambar melalui GUI tetap harus ada.

Bagian B (Video)

1. Video penjelasan algoritma dan aplikasi program yang diunggah ke youtube.
2. Video dibuat sekreatif mungkin dengan target untuk mensosialisasikan ilmu yang kalian sudah pelajari dan terapkan pada program ini. Bukan hanya video mengenai penggunaan aplikasi.

Bab II

Teori Singkat

2.1 Perkalian Matriks

Jika A adalah matriks $m \times r$ dan B adalah matriks matriks $r \times n$, maka hasil perkalian AB adalah matriks $m \times n$ yang elemen-elemennya ditentukan dengan aturan sebagai berikut: elemen baris- i dan kolom- j diperoleh dengan mengambil elemen-elemen baris- i matriks A serta elemen-elemen kolom- j matriks B kemudian mengalikan masing-masing pasangan elemen pada baris dan kolom tersebut dengan indeks yang bersesuaian dan menjumlahkan hasil-hasil perkalian tersebut. Perkalian dua buah matriks tidak selalu bersifat komutatif. Berikut beberapa properti aritmatika yang melibatkan perkalian dua buah matriks.

- (a) $A(BC) = (AB)C$
- (b) $A(B + C) = AB + AC$
- (c) $(B + C)A = BA + CA$
- (d) $a(BC) = (aB)C = B(aC)$

2.2 Nilai Eigen

Jika A sebuah matriks $n \times n$, vektor tidak nol x dalam R^n disebut eigen vektor matriks A jika Ax merupakan vektor x dikali sebuah skalar sehingga diperoleh

$$Ax = \lambda x$$

untuk beberapa nilai λ . Skalar λ ini disebut nilai eigen matriks A dan vektor x disebut vektor eigen yang berkorespondensi dengan λ tersebut. Dengan demikian, jika x adalah vektor eigen matriks A , maka pengalian vektor tersebut dengan matriks A tidak mengubah arah vektor. Sebagai tambahan, Jika A adalah matriks segitiga ataupun matriks diagonal, nilai-nilai eigen matriks tersebut adalah elemen-elemen diagonalnya. Teorema ini dapat digunakan dalam mencari nilai-nilai eigen suatu matriks dengan mencari terlebih dahulu matriks diagonal melalui transformasi kesamaan sebagai berikut.

$$A \rightarrow P^{-1}AP$$

Matriks P merupakan matriks nonsingular serta hasil transformasi ini akan menghasilkan matriks baru yang memiliki nilai-nilai eigen yang sama dengan matriks A . Jika transformasi tersebut menghasilkan matriks diagonal, proses ini disebut diagonalisasi matriks A .

2.3 Vektor Eigen

Vektor eigen dari suatu matriks A ($n \times n$) adalah suatu matriks kolom yang berukuran $n \times 1$ yang apabila dikalikan dengan matriks A akan menghasilkan vektor lain yang merupakan kelipatan vektor itu sendiri. Untuk setiap nilai eigen (λ) dari matriks A , dapat ditemukan vektor eigen dengan menyulihkan λ pada sistem persamaan linier berikut.

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

Solusi dari sistem persamaan linier tersebut akan berupa solusi parametrik, dan vektor-vektor yang menjadi basis dari ruang solusi sistem persamaan linier tersebut disebut sebagai vektor eigen.

$$A = \begin{bmatrix} 3 & 0 \\ 8 & -1 \end{bmatrix}$$

Sebagai contoh, matriks A memiliki nilai eigen $\lambda = 3$ dan $\lambda = -1$. Untuk $\lambda = 3$, diperoleh sistem persamaan linier berikut.

$$\begin{bmatrix} 0 & 0 \\ -8 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

SPL tersebut memiliki solusi $x_1 = \frac{1}{2}t$, $x_2 = t$, $t \in R$, sehingga vektor berikut

$$\begin{bmatrix} \frac{1}{2} \\ 2 \\ 1 \end{bmatrix}$$

merupakan vektor eigen yang berkoresponden dengan nilai eigen $\lambda = 3$. Untuk $\lambda = -1$, diperoleh sistem persamaan linier berikut.

$$\begin{bmatrix} -4 & 0 \\ -8 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

SPL tersebut memiliki solusi $x_1 = 0$, $x_2 = t$, $t \in R$, sehingga vektor berikut

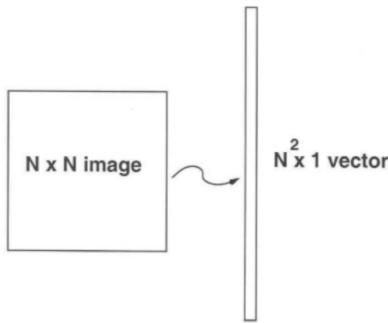
$$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

merupakan vektor eigen yang berkoresponden dengan nilai eigen $\lambda = -1$.

2.4 Eigenface

Eigenface merupakan salah satu dari berbagai algoritma yang dapat digunakan untuk pengenalan wajah (*face recognition*). Berikut adalah algoritma *training* eigenface.

1. Lakukan ekstraksi setiap gambar pada dataset menjadi matriks berukuran $N \times N$. Ekstraksi gambar dilakukan dengan kakas OpenCV. Pada tugas ini, ukuran gambar minimal 256×256 .
2. Setiap matriks gambar dimampatkan (*di-flatten*) dengan cara menyusun tiap kolom matriks ke dalam vektor berukuran $N^2 \times 1$, sehingga diperoleh vektor x_1, x_2, \dots, x_m dengan m adalah jumlah gambar yang ada pada dataset.



3. Hitung rata-rata (Ψ) dari setiap vektor, kemudian kurangi setiap vektor dengan vektor rata-rata.

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

$$a_i = x_i - \Psi$$

4. Susun setiap vektor a_i ke dalam matriks A berukuran $N^2 \times M$, sehingga

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

5. Hitung matriks kovarian dengan mengalikan A dengan A^T . Matriks A memiliki ukuran $N^2 \times M$, dan A^T memiliki ukuran $M \times N^2$, sehingga menghasilkan matriks $N^2 \times N^2$ yang tidak efisien untuk dihitung. Sehingga, kita mengalikan matriks A^T dengan matriks A, yang menghasilkan matriks berukuran $M \times M$, kemudian kita dapat mencari nilai eigen dan vektor eigen matriks tersebut.

$$Cov = A^T A$$

6. Hitung nilai eigen dan vektor eigen matriks kovarian awal dengan persamaan berikut

$$A^T A v_i = \lambda_i v_i$$

$$A A^T A v_i = \lambda_i A v_i$$

$$C' u_i = \lambda_i u_i$$

di mana

$$C' = A A^T$$

$$u_i = A v_i$$

7. Dari persamaan di atas, dapat disimpulkan nilai eigen matriks C' ($A A^T$) sama dengan nilai eigen matriks Cov ($A^T A$), dan vektor eigen matriks C' (u_i) dapat dicari dengan mengalikan vektor eigen matriks Cov (v_i) dengan matriks A.

8. Ambil K vektor eigen dari matriks C' yang berkoresponden dengan K nilai eigen terbesar ($K < M$). Vektor eigen ini memiliki ukuran $N^2 \times 1$.
9. Ambil vektor-vektor a_i yang telah dihitung sebelumnya dan nyatakan vektor tersebut sebagai kombinasi linier dari K vektor eigen u_j . Vektor u_j disebut *EigenFaces*.

$$a_i = \sum_{j=1}^K w_j u_j$$

10. Ambil koefisien *EigenFaces* (w) dari persamaan di atas dan nyatakan dalam bentuk vektor

$$x_i = [w_{1i} \ w_{2i} \ : \ w_{ji}]$$

Berikut adalah algoritma deteksi wajah (testing)

1. Lakukan ekstraksi dan *flatten* pada gambar testing, sehingga diperoleh vektor y .
2. Kurangi vektor y dengan vektor rata-rata (Ψ)

$$\emptyset = y - \Psi$$

3. Nyatakan vektor \emptyset sebagai kombinasi linier dari vektor-vektor u

$$\emptyset = \sum_{j=1}^K w_j u_j$$

4. Dari persamaan di atas, dapat dinyatakan vektor Ω yang berisi koefisien-koefisien vektor u

$$\Omega = [w_1 \ w_2 \ : \ w_j]$$

5. Ambil vektor Ω dan kurangi dengan *training image*, kemudian ambil vektor dengan jarak terkecil antara vektor *training* dan vektor *testing*

$$e_r = \min_l \|\Omega - \Omega_l\|$$

6. Jika $e_r < T_n$, dengan T_n adalah batas toleransi, maka wajah dapat dikenali. Jika tidak, maka wajah tidak cocok dengan wajah manapun di *training set*.

Bab III

Implementasi Program

3.1 Kakas Program

3.1.1 *OpenCV library*

OpenCV adalah *open-source library* yang ditujukan untuk aplikasi *computer-vision* dan *machine learning*. Di dalam aplikasi pengenalan wajah ini, OpenCV secara khusus digunakan untuk pembacaan dan pengolahan *file* gambar wajah menjadi sebuah matriks. *Library* ini juga digunakan untuk mengubah ukuran gambar menjadi 256 x 256 *pixels*, mengubah suatu gambar menjadi gambar *grayscale*, hingga memungkinkan fitur kamera pada aplikasi.

Library ini juga dipakai untuk melakukan auto-crop dari foto wajah frontal dengan metode Haar Cascade object detection. Data hasil training diperoleh dari openCV, di dalam file XML haarcascade_frontalface_alt.xml. (Sumber: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html)

3.1.2 *NumPy library*

NumPy adalah *library* Python yang menyediakan dukungan untuk tipe data *array* dan matriks multidimensi yang besar, bersamaan dengan berbagai koleksi fungsi matematika yang beroperasi pada array ini. Di dalam aplikasi ini, NumPy digunakan secara luas dalam pemanfaatan tipe data matriks dan array dan operasi-operasinya, seperti perkalian ataupun penjumlahan dan pengurangan matriks, *norm* suatu vektor (dalam representasi matriks), *transpose* matriks, pembuatan matriks identitas, dan banyak operasi matriks lainnya.

3.1.3 *SciPy library*

SciPy adalah *open-source library* yang digunakan dalam *scientific computing* serta *technical computing*. Salah satu modul yang disediakan *library* ini adalah modul aljabar linear. Di dalam aplikasi ini, modul aljabar linear yang disediakan khusus digunakan untuk mencari basis *null space* suatu matriks dalam pencarian *eigen vectors*.

3.1.4 *OS module*

Modul OS menyediakan fungsi-fungsi untuk pembuatan atau penghapusan suatu direktori; mengambil konten suatu file; mengubah atau melihat suatu *path* direktori; dan fungsi-fungsi lain yang berhubungan dengan *operating system*. Di dalam aplikasi ini, modul OS digunakan untuk melihat daftar *file* ataupun subdirektori dalam suatu direktori (terutama folder dataset), menggabungkan suatu *path*, serta memperoleh *absolute path* dari *relative path* suatu *file/direktori* untuk pembacaan *file*.

3.1.5 *Math module*

Math merupakan modul *built-in* yang disediakan Python untuk menyelesaikan persoalan matematika. Di dalam aplikasi ini, satu-satunya fungsi yang disediakan modul ini dan digunakan adalah fungsi akar kuadrat. Hal ini dilakukan untuk

memperoleh kecepatan operasi yang lebih tinggi dibandingkan menggunakan operator pangkat (dengan nilai pangkat 0,5) pada Python.

3.1.6 *Time module*

Modul Time pada Python menyediakan berbagai cara dalam merepresentasikan waktu di dalam sebuah program. Di dalam aplikasi ini, modul Time digunakan untuk melakukan penundaan terhadap eksekusi suatu blok program (yaitu pada saat inisiasi konfigurasi kamera). Selain itu, modul ini juga digunakan dalam menghitung perbedaan waktu yang telah berlalu dan khususnya diaplikasikan dalam fitur kamera serta tampilan waktu pemrosesan dataset pada aplikasi ini.

3.1.7 *PySimpleGUI library*

PySimpleGUI adalah salah satu *open-source library* yang dapat digunakan dalam *GUI programming*. Penggunaan *library* ini memungkinkan aplikasi *face recognition* dibuat dengan *GUI* yang sederhana dan mudah diimplementasikan. *Library* ini juga sekaligus digunakan dalam *styling* tampilan aplikasi.

3.1.8 *Python Imaging library*

Python Imaging library (PIL) adalah *library* yang dapat menambahkan kapabilitas pemrosesan gambar ke *interpreter* Python. Dua dari banyak modul yang disediakan PIL adalah modul Image dan modul ImageTk. Modul Image digunakan dalam aplikasi ini untuk pembacaan gambar dan juga mengonversi sebuah *array* menjadi data gambar. Selain itu, modul ImageTk digunakan untuk mengubah data gambar menjadi objek PhotoImage yang kemudian akan diproses dengan fungsi-fungsi lain pada *library* ini. Salah satu penggunaannya yaitu dalam menampilkan foto pada aplikasi.

3.2 Modul Program Aplikasi Pengenalan Wajah

3.2.1 Modul Eigen

Modul eigen menyediakan fungsi GetEigenInfo yang mengembalikan sebuah tuple berisi *array of eigen values* dan *matrix of eigen vectors*. Pada *matrix of eigen vectors*, kolom-kolom matriks tersebut merupakan vektor-vektor eigen yang berasosiasi dengan nilai eigen vektor dengan indeks kolom yang sesuai dengan indeks baris pada *array of eigen values*.

GetEigenInfo memanfaatkan dua fungsi, yaitu GetJacobi yang mengembalikan nilai-nilai eigen matriks simetris dengan memanfaatkan *Jacobi eigenvalue algorithm* serta GetEigenVectors yang mengembalikan matriks dengan kolom-kolomnya merupakan vektor-vektor eigen matriks tersebut.

Fungsi GetEigenVectors menerima nilai-nilai eigen suatu matriks dan juga matriks itu sendiri serta mencari basis-basis *null space* untuk memperoleh vektor-vektor eigen untuk tiap nilai eigen.

3.2.2 Modul Util

Modul Util menyediakan fungsi-fungsi untuk pembacaan dan pemrosesan *file* gambar-gambar wajah serta algoritma dalam mendapatkan *eigenface* masing-masing gambar dataset. Selain itu, modul ini juga memiliki fungsi-fungsi yang digunakan dalam tahapan pengenalan wajah, seperti fungsi GetEuclideanDistance yang

menghitung jarak euclidean minimum vektor representasi gambar uji dengan masing-masing vektor representasi gambar dataset yang telah diproses.

Dalam pemrosesan dataset, digunakan modul Eigen untuk memperoleh vektor-vektor eigen matriks kovarian dan kemudian dilakukan pengurutan secara menurun berdasarkan nilai-nilai eigen yang berkaitan dengan masing-masing vektor eigen.

3.2.3 Modul Face Detector

Modul Face Detector berguna untuk melakukan cropping wajah secara inteligen dari sebuah image. Modul ini digunakan dengan pertimbangan bahwa algoritma Eigenface merupakan algoritma yang sepenuhnya bergantung dari foto input yang diberikan. Artinya, algoritma sangat sensitif terhadap perbedaan background dari foto dataset, tanpa fitur yang dapat membedakan wajah ataupun objek.

Modul Face Detector dibangun dengan menggunakan library openCV, dengan bantuan model Haar Cascade Frontal Face yang telah di-*pre-train* oleh library tersebut. File hasil training disimpan di dalam folder haarcascade_frontalface_alt.xml. Dengan adanya modul face detector, diharapkan program utama akan lebih resilien terhadap perbedaan kondisi latar dari citra uji dan dataset, karena telah dinormalisasi.

Modul Face Detector harus dijalankan sebelum eksekusi program pada dataset training secara manual. Lalu, pengguna dapat memilih folder dataset sebagai citra-citra uji di dalam folder test/cropped.

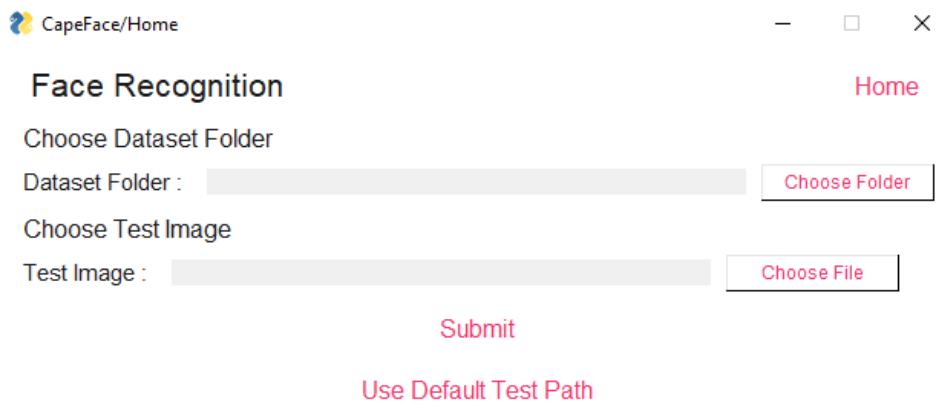
3.2.4 Program Utama

Program utama mengandung driver dari segala fungsi di util, face detector, dan juga eigen.py. Program ini memiliki beberapa fungsi yang bertanggung jawab untuk menampilkan GUI dan juga Camera. Berikut merupakan ringkasan dari flow program utama.

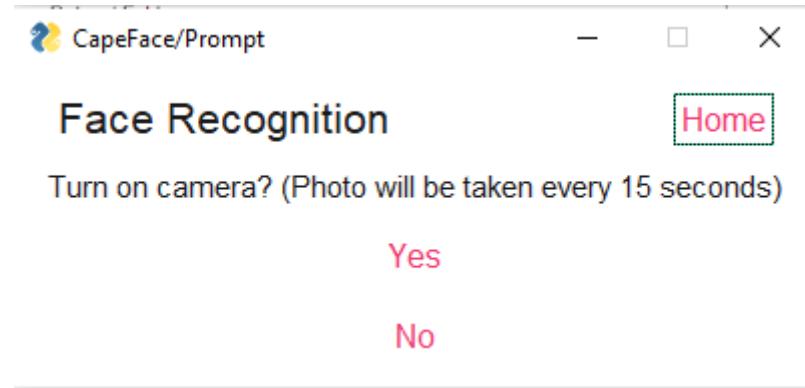
Terdapat boolean INTELLI_CROP di dalam program, yang pada mode default diaktifkan untuk meningkatkan akurasi program (harus dikonfigurasi di main.py). Dataset harus di preprocess terlebih dahulu dengan menjalankan program crop untuk mengcrop dataset. Jika ingin mematikan fitur crop, maka dataset yang digunakan haruslah tidak di crop juga.

Program memiliki beberapa fungsi penting yaitu:

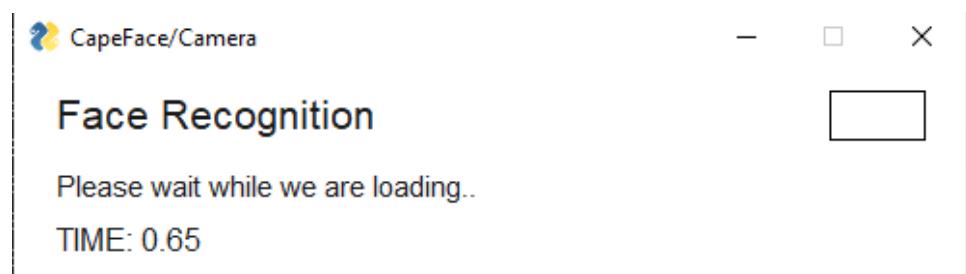
- SetupFile(), berfungsi untuk menampilkan GUI yang memrompt user untuk memasukkan folder dataset uji dan juga citra uji. Terdapat opsi use default yaitu memakai dataset dan citra uji bawaan dari program.



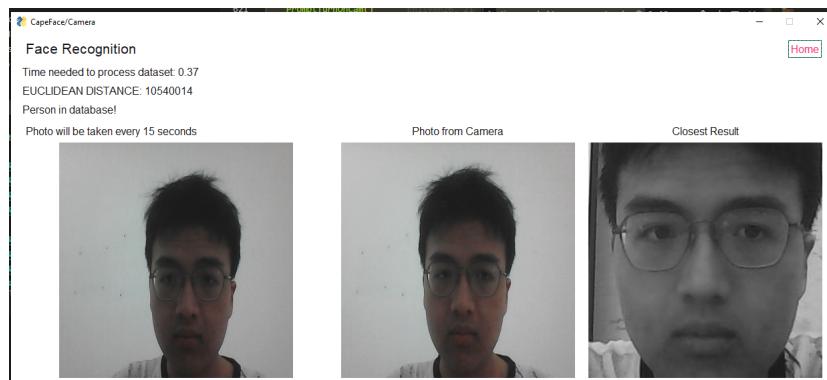
- `PromptTurnOnCam()`, berfungsi untuk menampilkan GUI yang memprompt user untuk memasukkan pilihan apakah user ingin menyalaikan kamera atau tidak. Jika ya maka akan mengset variabel `turnOnCam` menjadi `True` dan jika tidak akan mengset variabel `turnOnCam` menjadi `False`.



- `Loading()`, berfungsi untuk melakukan pemrosesan pada dataset yang dipilih. Proses dilakukan dengan asinkronus dengan membuat thread baru agar bersifat non-blocking. Program akan mengset variabel boolean `datasetLoaded` menjadi `True` ketika selesai memproses dataset.
- `LoadingScreen()`, akan menampilkan layar loading selagi pemrosesan terhadap dataset berlangsung. Berisi timer selagi menunggu proses `Loading()` selesai dijalankan.



- `DisplayResultCam()` dan `DisplayResult()` masing-masing akan menampilkan GUI dengan mode kamera ON dan OFF secara berturut-turut. Akan memanggil fungsi-fungsi di library util sesuai dengan kebutuhan seperti mencari gambar di database yang paling mirip dan lain sebagainya.



```
## Kode Python Main
while True:
    SetupFile()
    if (err):
        continue
```

```
PromptTurnOnCam()
if (empty_test_err):
    continue
if (goToHome):
    Continue

start_time = time.time()
threading.Thread(target=Loading,
                  args=(),
                  daemon=True).start()
LoadingScreen()
if turn_on_cam:
    DisplayResultCam()
else:
    DisplayResult()
```

Bab IV

Eksperimen

4.1 Pengujian Modul Eigen

Modul eigen merupakan bottleneck dari program kami. Hal ini karena untuk mencari eigenvectors dari dataset uji sebanyak N, diperlukan untuk melakukan operasi pada matriks NxN. Waktu dari processing ini disimpan baik dalam GUI ataupun di dalam terminal testing, sedangkan waktu processing individual image tidak disimpan karena terlambat cepat jika dibandingkan dengan waktu processing di modul eigen.

Label	Data	Output
Fungsi GetEigenInfo	<p>Input : Matriks acak 5x5 bertipe float64, <i>convergence threshold</i> = 1e-10</p> <p>Spesifikasi Laptop : AMD Ryzen 5 5500U with Radeon Graphics, 2100 Mhz, 6 Core(s), 12 Logical Processor(s)</p>	<pre>Calculating Eigen Values : ----Convergence Obtained----- time : 0.0039997100838078125 Our Result : [1.556922390.482682, 1.10797149e+09 3.04102602e+08 -3.62199240e+08 [-1.70824376e+09] Library Result : [1556922390.482682, 1107971491.4906113, 304102601.797132, -362199240.11157525, -17082437 Calculating Eigen Vectors : time : 0.0099999275267519531 Our Result : [[0.1874072 -0.4989514 0.27512688 0.65633392 0.45766476] [0.51778286 -0.65974639 0.51531401 0.18933153 -0.08387548] [0.20931374 -0.37994154 -0.54058791 -0.055576 0.72337138] [0.75163532 -0.43537113 -0.03447827 -0.4908934 -0.05771695] [-0.29666742 -0.02076033 0.60442752 -0.53787949 0.50686337]] Library Result : [[0.1874072 -0.4989514 0.27512688 0.65633392 0.45766476] [0.51778286 -0.65974639 0.51531401 0.18933153 -0.08387548] [0.20931374 -0.37994154 -0.54058791 -0.055576 0.72337138] [0.75163532 -0.43537113 -0.03447827 -0.4908934 -0.05771695] [-0.29666742 -0.02076033 0.60442752 -0.53787949 0.50686337]]</pre>
Analisis	Nilai eigen diperoleh dengan waktu eksekusi yang sangat cepat (0.004 detik) karena data yang kecil. Nilai eigen beserta vektor-vektor eigen yang diperoleh bernilai sangat akurat jika dibandingkan dengan hasil perhitungan <i>library</i> .	
Fungsi GetEigenInfo	<p>Input : Matriks acak 20x20 bertipe float64, <i>convergence threshold</i> = 1e-10</p> <p>Spesifikasi Laptop : AMD Ryzen 5 5500U with Radeon Graphics, 2100 Mhz, 6 Core(s), 12 Logical Processor(s)</p>	<pre>----Convergence Obtained----- time : 0.17400646209176797 Our Result : [1.34760234e+09 2.05741133e+09 2.53779783e+09 1.97130045e+09 [1.34760234e+09 1.29453451e+09 1.16192632e+09 5.57104086e+08 [5.33656523e+08 8.32644624e+07 7.97118178e+06 -3.81289063e+08 [-6.95073043e+08 -1.24877479e+09 -1.46985862e+09 -1.56595927e+09 [-1.89651474e+09 -2.0837801e+09 -2.61247353e+09 -2.87282493e+09 [1.34760234e+09 2.05741133e+09 2.53779783e+09 1.97130045e+09 [1.34760234e+09 1.29453451e+09 1.16192632e+09 5.57104086e+08 [5.33656523e+08 8.32644624e+07 7.97118178e+06 -3.81289063e+08 [-6.95073043e+08 -1.24877479e+09 -1.46985862e+09 -1.56595927e+09 [-1.89651474e+09 -2.0837801e+09 -2.61247353e+09 -2.87282493e+09]</pre>
Analisis	Nilai eigen masih sangat akurat jika dibandingkan dengan hasil perhitungan <i>library</i> serta waktu eksekusi masih singkat (0.174 detik). Vektor-vektor eigen juga akurat jika dibandingkan dengan hasil perhitungan <i>library</i> tetapi tidak dicantumkan karena hasil <i>print</i> yang panjang.	
Fungsi GetEigenInfo	<p>Input : Matriks acak 100x100 bertipe float64, <i>convergence threshold</i> = 1e-10</p> <p>Spesifikasi Laptop : AMD Ryzen 5 5500U with Radeon Graphics, 2100 Mhz, 6 Core(s), 12 Logical Processor(s)</p>	<p><i>Eigen value</i> hasil perhitungan algoritma program :</p>

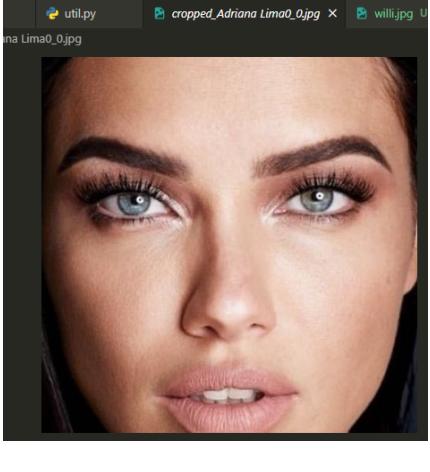
		<pre>Calculating Eigen Values : =====Convergence Obtained===== time : 37.73985695838928 Our Result : [7.73146404e+09 -4.57184200e+09 7.00418011e+09 6.95642544e+09 6.69778683e+09 6.34772259e+09 6.24448496e+09 6.15564920e+09 5.9725696e+09 5.681461108e+09 5.59269886e+09 5.31225528e+09 5.06169584e+09 4.91158442e+09 4.76692872e+09 4.69260928e+09 4.54866283e+09 4.43414448e+09 4.14396161e+09 4.12241884e+09 3.96847713e+09 3.78710663e+09 3.72442629e+09 3.64509566e+09 3.30018035e+09 2.33349634e+09 3.07371626e+09 2.90341158e+09 2.69010933e+09 2.61854160e+09 2.46095454e+09 2.31175244e+09 2.26331107e+09 2.16251725e+09 2.09981852e+09 1.90619920e+09 1.83332193e+09 1.54864869e+09 1.42151912e+09 1.33643448e+09 1.26854796e+09 1.12530503e+09 9.63988249e+08 8.95254166e+08 8.45388100e+08 6.27774071e+08 4.86930157e+08 4.49456262e+08 2.47413636e+08 1.47628169e+08 -1.44076962e+08 -2.23219331e+08 -4.09985131e+08 -5.36791555e+08 -5.90096921e+08 -7.93111134e+08 -8.64681499e+08 -9.612496951e+08 -1.14470429e+08 -1.18799473e+09 -1.25541184e+09 -1.43484892e+09 -1.48313614e+09 -1.60402166e+09 -3.08853960e+09 -3.14823789e+09 -3.35054176e+09 -3.47293349e+09 -4.06409338e+09 -4.38237605e+09 -4.57040151e+09 -4.75983221e+09 -4.81049743e+09 -4.97122503e+09 -5.27543768e+09 -5.43113406e+09 -5.57742616e+09 -5.72907514e+09 -5.75457768e+09 -5.80420652e+09 -6.21578717e+09 -6.36223301e+09 -6.49797588e+09 -6.83162512e+09 -7.07162786e+09 -7.33202460e+09 -7.80100552e+09 -8.80506648e+09]</pre>
		<h3>Eigen value hasil Numpy Library :</h3> <p>Library result :</p> <pre>[7.73146404e+09 -4.57184200e+09 7.00418011e+09 6.95642544e+09 6.69778683e+09 6.34772259e+09 6.24448496e+09 6.15564920e+09 5.9725696e+09 5.681461108e+09 5.59269886e+09 5.31225528e+09 5.06169584e+09 4.91158442e+09 4.76692872e+09 4.69260928e+09 4.54866283e+09 4.43414448e+09 4.14396161e+09 4.12241884e+09 3.96847713e+09 3.78710663e+09 3.72442629e+09 3.64509566e+09 3.30018035e+09 2.33349634e+09 3.07371626e+09 2.90341158e+09 2.69010933e+09 2.61854160e+09 2.46095454e+09 2.31175244e+09 2.26331107e+09 2.16251725e+09 2.09981852e+09 1.90619920e+09 1.83332193e+09 1.54864869e+09 1.42151912e+09 1.33643448e+09 1.26854796e+09 1.12530503e+09 9.63988249e+08 8.95254166e+08 8.45388100e+08 6.27774071e+08 4.86930157e+08 4.49456262e+08 2.47413636e+08 1.47628169e+08 -1.44076962e+08 -2.23219331e+08 -4.09985131e+08 -5.36791555e+08 -5.90096921e+08 -7.93111134e+08 -8.64681499e+08 -9.612496951e+08 -1.14470429e+08 -1.18799473e+09 -1.25541184e+09 -1.43484892e+09 -1.48313614e+09 -1.60402166e+09 -3.08853960e+09 -3.14823789e+09 -3.35054176e+09 -3.47293349e+09 -4.06409338e+09 -4.38237605e+09 -4.57040151e+09 -4.75983221e+09 -4.81049743e+09 -4.97122503e+09 -5.27543768e+09 -5.43113406e+09 -5.57742616e+09 -5.72907514e+09 -5.75457768e+09 -5.80420652e+09 -6.21578717e+09 -6.36223301e+09 -6.49797588e+09 -6.83162512e+09 -7.07162786e+09 -7.33202460e+09 -7.80100552e+09 -8.80506648e+09]</pre>
Analisis		<p>Waktu eksekusi meningkat jauh menjadi 37.740 detik tetapi hasil masih sangat akurat dibandingkan dengan hasil perhitungan <i>library</i>.</p>
Fungsi GetEigenInfo	<p>Input : Matriks acak 100x100 bertipe awal integer, <i>convergence threshold</i> = 1e-10</p> <p>Spesifikasi <i>device</i> : processor Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz, 2304 Mhz, 2 Core(s), 4 Logical Processor(s)</p>	<p><i>Eigen value</i> hasil perhitungan algoritma program :</p> <pre>=====Convergence Obtained===== [-8.160393915e+09 -4.7479605e+09 -7.29965299e+09 -7.10924512e+09 -6.76951069e+09 -6.72139623e+09 -6.36933389e+09 -6.19026760e+09 -5.93835110e+09 -5.74845617e+09 -5.58063583e+09 -5.41418316e+09 -5.25367889e+09 -5.05817411e+09 -4.96931957e+09 -4.78315264e+09 -4.55873120e+09 -4.46866368e+09 -4.27838406e+09 -4.24278154e+09 -3.97913050e+09 -3.82686324e+09 -3.66921914e+09 -3.63591937e+09 -3.37015757e+09 -3.24358294e+09 -3.22152712e+09 -3.09494858e+09 -2.94208276e+09 -2.8645974e+09 -2.76269981e+09 -2.62733677e+09 -2.534005533e+09 -2.34866025e+09 -2.29016004e+09 -2.09208197e+09 -1.86925879e+09 -1.72671783e+09 -1.58339235e+09 -1.47345758e+09 -1.46608602e+09 -1.32263788e+09 -1.00878627e+09 -8.34099823e+08 -7.28902143e+08 -8.00863538e+08 -4.52769070e+08 -3.35693128e+08 -2.29066855e+08 -3.00874884e+07 -1.28483784e+08 -2.50649031e+08 -3.04176276e+08 -4.27784375e+08 -5.62161124e+08 -6.42453890e+08 -8.29088245e+08 -8.87891982e+08 -1.02841362e+09 -1.31346461e+09 -1.38498778e+09 -1.43783365e+09 -1.64681900e+09 -1.83434993e+09 -2.00568343e+09 -2.07988050e+09 -2.18814588e+09 -2.38839154e+09 -2.36884624e+09 -2.55153961e+09 -2.75827766e+09 -2.86955333e+09 -2.93728417e+09 -3.0577344e+09 -3.15378424e+09 -3.37420921e+09 -3.55295735e+09 -3.61852734e+09 -3.78633482e+09 -3.98032030e+09 -3.99111257e+09 -4.13860663e+09 -4.27568317e+09 -4.51069391e+09 -4.92166644e+09 -4.95607789e+09 -5.09519721e+09 -5.21979367e+09 -5.37767147e+09 -5.63326055e+09 -5.79491695e+09 -5.90236560e+09 -6.02392253e+09 -6.24471421e+09 -6.54946967e+09 -6.64100459e+09 -6.79316388e+09 -7.08185016e+09 -7.42771035e+09 -8.11464595e+09] time : 197.31069374084473</pre>

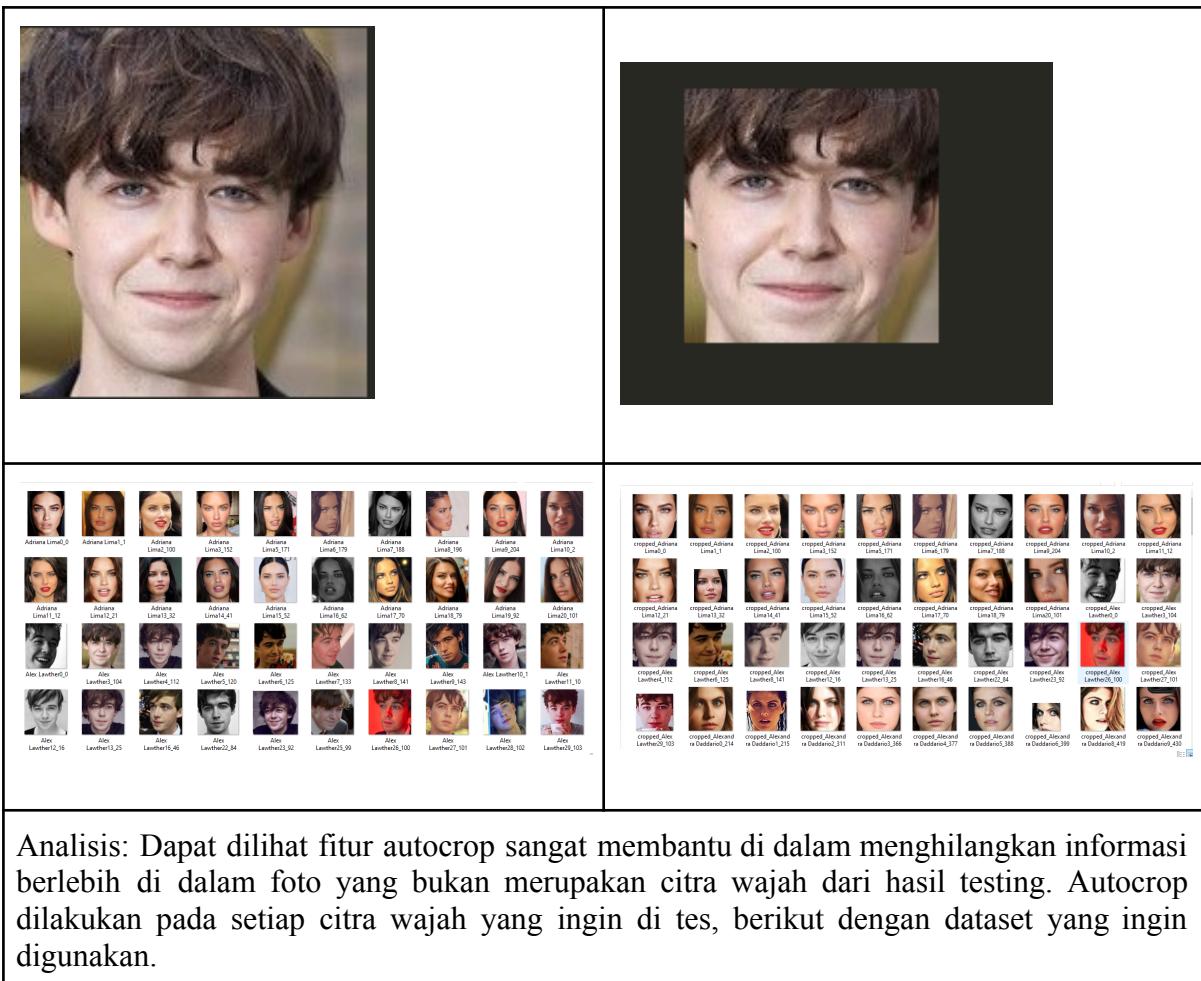
		<pre>\time : 197.31069374084473 [-8.16093915e+09 -7.47496055e+09 -7.29665299e+09 -7.10924512e+09 -6.76951069e+09 -6.72139623e+09 -6.36933309e+09 -6.19026760e+09 -5.93385110e+09 -5.7484617e+09 -5.58063583e+09 -5.41418316e+09 -5.25367089e+09 -5.05871411e+09 -4.96931957e+09 -4.78315264e+09 -4.55867312e+09 -4.46896638e+09 -4.27384006e+09 -4.24278154e+09 -3.97913050e+09 -3.82686324e+09 -3.66921914e+09 -3.63591937e+09 -3.37015757e+09 -3.24358294e+09 -3.22152712e+09 -3.04948458e+09 -2.94208276e+09 -2.86458974e+09 -2.76269981e+09 -2.62733677e+09 -2.53400533e+09 -2.34866025e+09 -2.29016004e+09 -2.09208197e+09 -1.86925879e+09 -1.72071703e+09 -1.58339235e+09 -1.47345758e+09 -1.46608602e+09 -1.32263788e+09 -1.09978627e+09 -8.34099823e+08 -7.28902143e+08 -6.80063538e+08 -4.52769070e+08 -3.35693128e+08 -2.29066855e+08 3.00874804e+08 1.28483784e+08 2.50649031e+08 3.04176276e+08 4.27784375e+08 5.62161124e+08 6.42453890e+08 8.29088245e+08 8.87891982e+08 1.02841362e+09 1.31346461e+09 1.38498778e+09 1.43783365e+09 1.64681900e+09 1.83434993e+09 2.00568343e+09 2.07980500e+09 2.18814508e+09 2.30839154e+09 2.36884624e+09 2.55153961e+09 2.75827760e+09 2.86955333e+09 2.93728417e+09 3.05770344e+09 3.15378424e+09 3.37420921e+09 3.55295735e+09 3.61852734e+09 3.78633482e+09 3.95172030e+09 3.99111257e+09 4.13860063e+09 4.27568317e+09 4.51069391e+09 4.92166644e+09 4.95607789e+09 5.09519721e+09 5.21979367e+09 5.37767147e+09 5.63326055e+09 5.79941695e+09 5.90236560e+09 6.02392253e+09 6.24471421e+09 6.54946967e+09 6.64100459e+09 6.79316388e+09 7.08185016e+09 7.42771035e+09 8.11464595e+09] (venv) PS C:\Users\ASUS\Desktop\tubes_algeo2></pre>
Analisis	Dibutuhkan waktu 197 detik untuk mendapatkan nilai eigen. Terlihat bahwa spesifikasi device yang digunakan juga menentukan waktu eksekusi program. Nilai yang diperoleh masih sangat akurat apabila dibandingkan dengan nilai-nilai eigen dari <i>library</i> .	
Fungsi GetEigenInfo	<p>Matrik 100x100 acak simetris, <i>convergence threshold</i> : 1e-5</p> <p>Spesifikasi <i>device</i> : processor Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz, 2304 Mhz, 2 Core(s), 4 Logical Processor(s)</p>	<p><i>Eigen value</i> hasil perhitungan algoritma program :</p> <pre>=====Convergence Obtained===== [-7.97557985e+09 -7.60335304e+09 -7.15687697e+09 -6.99024459e+09 -6.74001289e+09 -6.48457691e+09 -6.42349395e+09 -6.40494195e+09 -6.11081359e+09 -5.88105855e+09 -5.80793229e+09 -5.60794933e+09 -5.45445333e+09 -5.29906522e+09 -5.09502694e+09 -5.02023702e+09 -4.85876387e+09 -4.58864770e+09 -4.52871342e+09 -4.26661910e+09 -4.13871954e+09 -4.05516543e+09 -3.94633976e+09 -3.79556318e+09 -3.41769007e+09 -3.34920632e+09 -3.23897319e+09 -3.10065239e+09 -3.09930947e+09 -2.91378581e+09 -2.74411128e+09 -2.50398075e+09 -2.39493738e+09 -2.17262895e+09 -2.11982634e+09 -1.99214296e+09 -1.90930947e+09 -1.63228980e+09 -1.57495105e+09 -1.42365737e+09 -1.36042685e+09 -1.29563712e+09 -1.06108605e+09 -1.01285399e+09 -8.11753935e+08 -6.45355290e+08 -5.41851127e+08 -3.62723214e+08 -2.66407339e+08 -1.66953058e+08 -2.92566598e+08 7.75686338e+07 -3.23924874e+08 4.41256849e+08 4.78173294e+08 7.29657917e+08 -8.30989984e+08 9.56671457e+08 1.11012719e+09 1.35108949e+09 -1.44803782e+09 1.51770745e+09 1.67398554e+09 1.74463589e+09 -1.88838354e+09 1.92717882e+09 2.01586879e+09 2.26158674e+09 -2.37487703e+09 2.57806019e+09 2.80243110e+09 2.84709214e+09 -3.07501627e+09 3.10550627e+09 3.20635953e+09 3.38934338e+09 -3.50753784e+09 3.56318026e+09 3.79273758e+09 3.84449358e+09 -4.24205165e+09 4.39888129e+09 4.52320336e+09 4.57992301e+09 -6.42495709e+09 4.77566276e+09 4.87850294e+09 5.07527641e+09 -5.19177888e+09 5.39871011e+09 5.83368169e+09 5.84489919e+09 -6.07357583e+09 6.26128073e+09 6.44759876e+09 6.69781218e+09 -7.08296108e+09 7.13689569e+09 7.39834756e+09 8.15196907e+09] time : 168.0308372642517</pre>

Eigen value hasil *Numpy library* :

	<pre> time : 168.0306372642517 [-7.97557985e+09 -7.60335304e+09 -7.15687697e+09 -6.99024459e+09 -6.74001289e+09 -6.48457691e+09 -6.42349395e+09 -6.40484195e+09 -6.11081355e+09 -5.88165855e+09 -5.80793229e+09 -5.68794933e+09 -5.45445333e+09 -5.29966522e+09 -5.09502694e+09 -5.02023702e+09 -4.85876387e+09 -4.68064770e+09 -4.52871342e+09 -4.26601910e+09 -4.13871954e+09 -4.05516543e+09 -3.94633970e+09 -3.79556318e+09 -3.41769007e+09 -3.34970632e+09 -3.23897319e+09 -3.18065239e+09 -3.00956291e+09 -2.91378581e+09 -2.74411128e+09 -2.50398075e+09 -2.39493738e+09 -2.17262895e+09 -2.11982634e+09 -1.99214296e+09 -1.90930947e+09 -1.63228980e+09 -1.57495105e+09 -1.42306573e+09 -1.36042685e+09 -1.29563712e+09 -1.06108605e+09 -1.01285399e+09 -8.11753935e+08 -6.45355290e+08 -5.41051127e+08 -3.62723214e+08 -2.66407339e+08 -1.66953058e+08 -2.92566508e+07 7.75686380e+07 3.23924874e+08 4.41256849e+08 4.78173294e+08 7.29657917e+08 8.30098994e+08 9.56671457e+08 1.11012719e+09 1.35109493e+09 1.44803782e+09 1.51770745e+09 1.67398554e+09 1.74463589e+09 1.80838354e+09 1.92717882e+09 2.01586879e+09 2.26150674e+09 2.37487703e+09 2.57806019e+09 2.80242110e+09 2.84709214e+09 3.07501627e+09 3.18550627e+09 3.20635053e+09 3.38934338e+09 3.50753784e+09 3.563118026e+09 3.79273758e+09 3.84449358e+09 4.24205165e+09 4.39888129e+09 4.52320336e+09 4.57992301e+09 4.64295709e+09 4.77566276e+09 4.87850294e+09 5.07527641e+09 5.19177808e+09 5.39871011e+09 5.83368169e+09 5.84409919e+09 6.07357583e+09 6.26120073e+09 6.44759876e+09 6.69781218e+09 7.08296108e+09 7.13689569e+09 7.39834756e+09 8.15196907e+09] </pre>
Analisis	Dibutuhkan waktu 168 detik untuk mencari eigenvalue. Hal ini menunjukkan bahwa peningkatan <i>threshold</i> dari 1e-10 menjadi 1e-5 dapat mempercepat waktu eksekusi program. Terlihat juga bahwa <i>threshold</i> bernilai 1e-5 masih menghasilkan nilai eigen yang akurat.

4.2.2. Pengujian autocrop

Data	Output
	



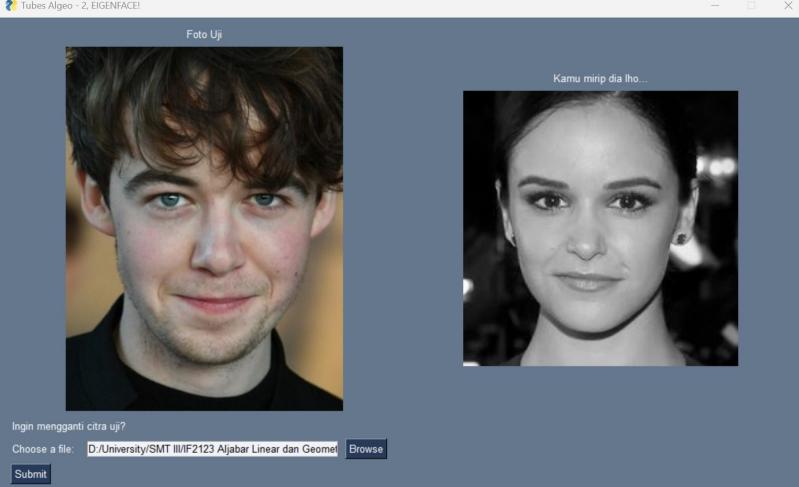
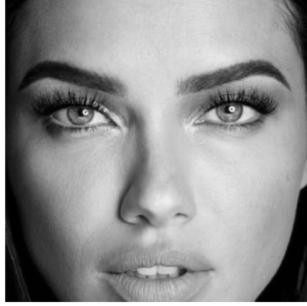
Analisis: Dapat dilihat fitur autocrop sangat membantu di dalam menghilangkan informasi berlebih di dalam foto yang bukan merupakan citra wajah dari hasil testing. Autocrop dilakukan pada setiap citra wajah yang ingin di tes, berikut dengan dataset yang ingin digunakan.

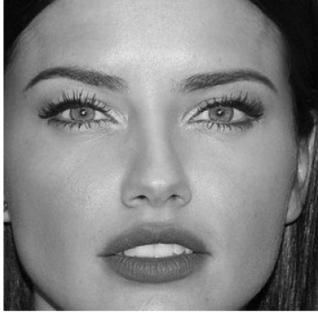
4.2.3 Pengujian *face recognition*

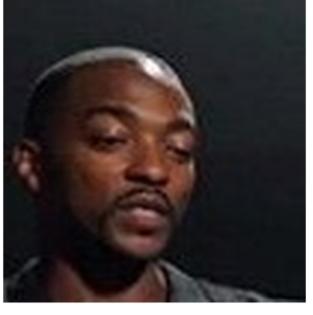
Dilakukan pengujian terhadap aplikasi pengenalan wajah yang telah dibuat. Semua pengujian dilakukan menggunakan dataset sebanyak 168 gambar berisi wajah 10 orang berbeda. Dilakukan *auto-crop* latar belakang pada setiap gambar dataset sebelum *runtime* serta juga dilakukan *auto-crop* pada gambar uji saat *runtime* untuk meningkatkan akurasi program. Auto-crop akan membuat dimensi citra uji menjadi persegi, sehingga dimensi citra uji pun tidak menjadi masalah. (Note: gambar dataset harus di crop terlebih dahulu dengan program *face_detector.py*)

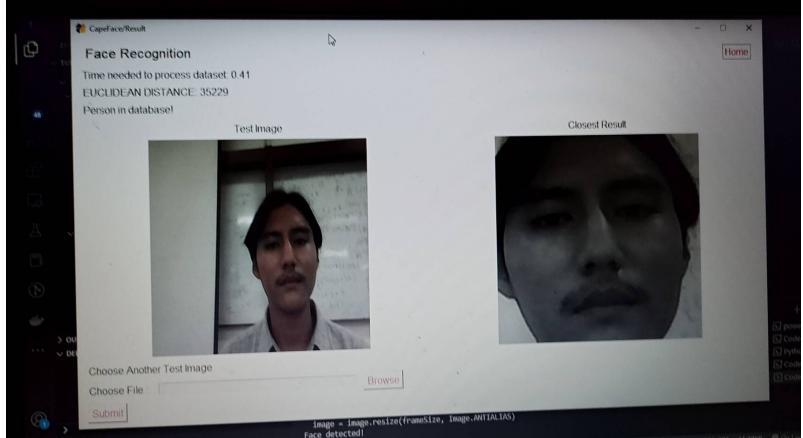
Selain itu, ditentukan dua buah *threshold*, yaitu *threshold* bawah bernilai 150 000 000 dan *threshold* atas bernilai 300 000 000 (nilai *threshold* perlu disesuaikan dengan *dataset* yang dimiliki). Nilai jarak kecil dari *threshold* bawah menandakan gambar uji memiliki wajah yang mirip dengan salah satu gambar pada dataset. Nilai jarak besar dari *threshold* menandakan tidak ada wajah yang mirip pada dataset. Terakhir, nilai di antara kedua *threshold* menandakan program tidak dapat menentukan dengan pasti kemiripan wajah gambar uji.

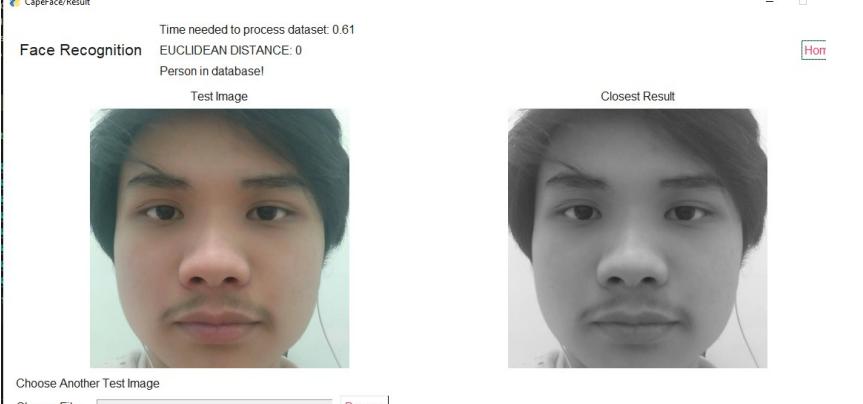
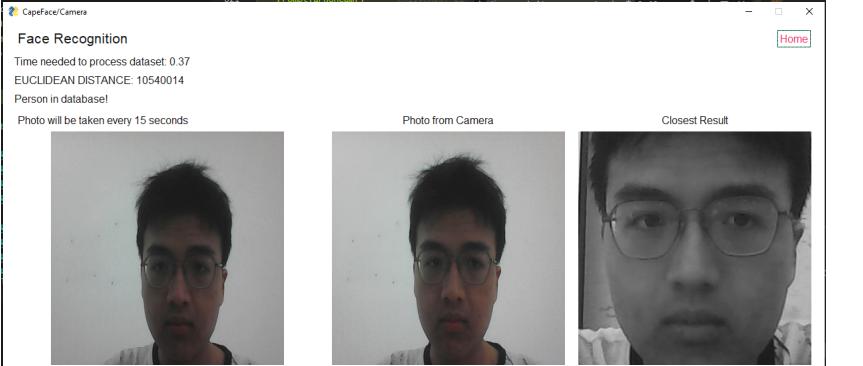
Selain menggunakan dataset yang digunakan di atas, dilakukan juga pengujian dataset kecil yang berisi beberapa gambar muka anggota kelompok beserta dosen mata kuliah IF2123. Pengujian ini ditujukan hanya untuk eksplorasi aplikasi. Hasil pengujian mungkin tidak dapat dipakai sebagai referensi akurasi karena jumlah dataset dan data uji yang sangat kecil.

Label	Hasil Uji
Test case 0	 <p>Ingin mengganti citra uji? Choose a file: D:/University/SMT III/IF2123 Aljabar Linear dan Geometri <input type="button" value="Browse"/> <input type="button" value="Submit"/></p>
Analisis	<p>Variasi dimensi, warna gambar:</p> <p>Gambar diatas menunjukkan hasil citra uji dari program v.1. yang belum menggunakan fitur autocrop. Tanpa autocrop, citra uji akan dipaksa menjadi berukuran persegi sehingga proporsi citra akan berubah dan terdistorsi. Background juga berkontribusi besar pada hasil operasi. Hasil yang didapat tidak memuaskan.</p>
Test case 1	<p>Face Recognition Home</p> <p>Time needed to process dataset: 286.64 EUCLIDEAN DISTANCE: 0 Person in database!</p> <p>Test Image </p> <p>Closest Result </p> <p>Choose Another Test Image Choose File: <input type="file"/> <input type="button" value="Browse"/></p>
Analisis	<p>Gambar di dalam dataset:</p> <p>Dibutuhkan 286.64 detik (168x168) untuk memproses dataset yang digunakan. Program menebak dengan benar gambar uji yang berada di dalam dataset itu sendiri (jarak <i>euclidian</i> bernilai 0). Warna citra uji tidak masalah baik grayscale maupun colored, karena program akan mengkonversi setiap citra menjadi grayscale.</p>

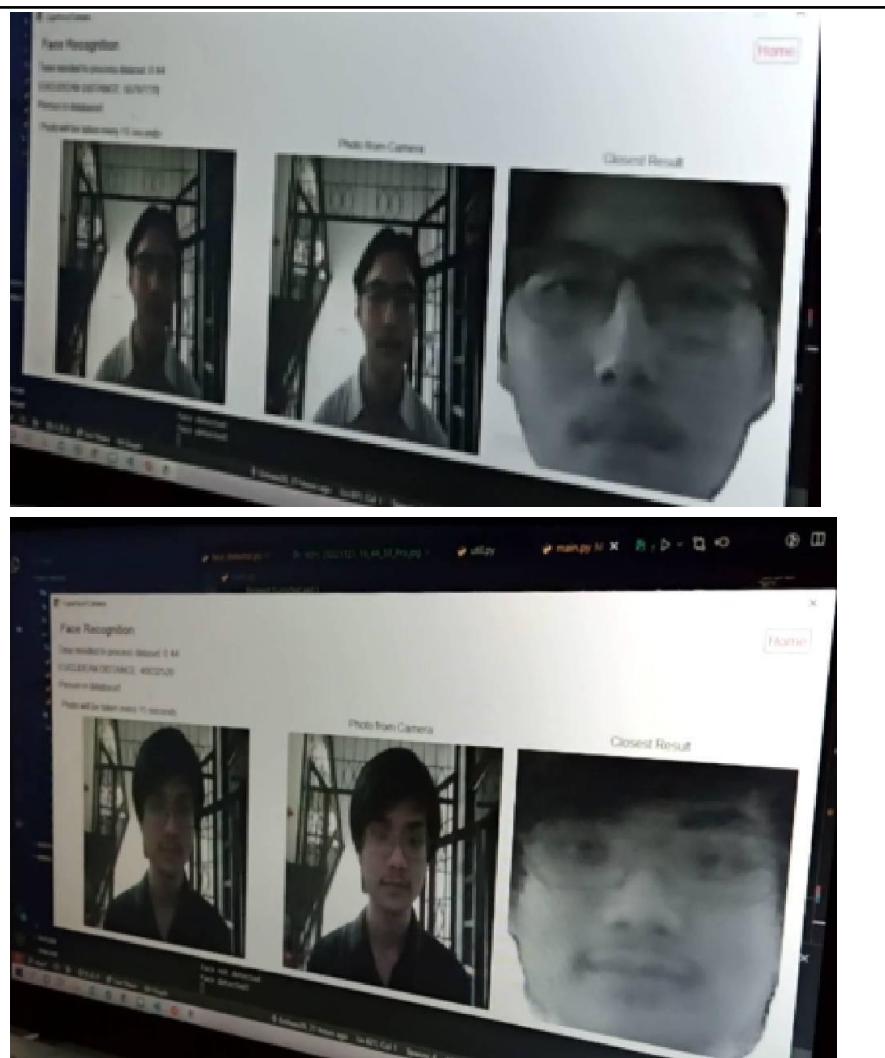
Test case 2	<p>Face Recognition</p> <p>Time needed to process dataset: 281.87</p> <p>EUCLIDEAN DISTANCE: 110551904</p> <p>Person in database!</p> <p style="text-align: right;">Home</p> <div style="display: flex; justify-content: space-around; align-items: center;"> Test Image  Closest Result  </div> <p>Choose Another Test Image</p> <p>Choose File : <input type="text" value="D:/Algeo/Tugas/Tubes_2/tubes_algeo2/test/Datatest/C"/> <input type="button" value="Browse"/></p>
Analisis	<p>Variasi background:</p> <p>Program menebak dengan benar gambar wajah yang berada di luar dataset dengan jarak <i>euclidian</i> bernilai 110 551 904. Background tidak menjadi masalah karena program melakukan autocrop pada citra uji (internal, tidak ditampilkan pada test image).</p>
Test Case 3	<p>Face Recognition</p> <p>Time needed to process dataset: 281.87</p> <p>EUCLIDEAN DISTANCE: 166292131</p> <p>Picture is unrecognized, maybe not a person/in DB</p> <p style="text-align: right;">Home</p> <div style="display: flex; justify-content: space-around; align-items: center;"> Test Image  Closest Result  </div> <p>Choose Another Test Image</p> <p>Choose File : <input type="text"/> <input type="button" value="Browse"/></p>
Analisis	<p>Variasi pencahayaan::</p> <p>Program juga menebak dengan benar gambar yang berada di luar dataset. Namun, nilai jarak berada di antara kedua <i>threshold</i> sehingga program tidak dapat memastikan kemiripan kedua gambar tersebut. Hal ini karena foto uji dilakukan dibawah pencahayaan yang berbeda.</p> <p>Hal ini disebabkan adanya sedikit perbedaan penampilan kedua wajah tersebut walaupun merupakan dua orang yang sama (Adriana Lima) dan ditandakan dengan jarak yang hanya sedikit berada di atas <i>threshold</i> bawah.</p>

Test Case 4	<p>Face Recognition</p> <p>Time needed to process dataset: 28187</p> <p>EUCLIDEAN DISTANCE: 283488683</p> <p>Picture is unrecognized, maybe not a person/in DB</p> <p style="text-align: right;">Home</p> <div style="display: flex; justify-content: space-around; align-items: center;"> Test Image  Closest Result  </div> <p>Choose Another Test Image</p> <p>Choose File : <input type="text" value="D/Algeo/Tugas/Tubes_2/tubes_algeo2/test/Datatest/C"/> <input type="button" value="Browse"/></p>
Analisis	<p>Variasi Angle:</p> <p>Program masih menebak dengan benar gambar yang berada di luar data set tetapi nilai jarak hampir mendekati <i>threshold</i> atas. Hal ini disebabkan posisi wajah yang tidak menghadap ke depan pada kedua gambar serta adanya bagian wajah yang tertutupi pada gambar dataset sehingga menghasilkan jarak yang cukup besar.</p> <p>Fitur autocrop akan mencoba untuk menghilangkan background citra uji dan meninggalkan fitur fasialnya saja. Beruntung, di dalam dataset mengandung foto citra yang juga menampilkan side-profile orang di dalam citra uji.</p>
Test Case 5	<p>Face Recognition</p> <p>Time needed to process dataset: 28187</p> <p>EUCLIDEAN DISTANCE: 147893261</p> <p>Person in database!</p> <p style="text-align: right;">Home</p> <div style="display: flex; justify-content: space-around; align-items: center;"> Test Image  Closest Result  </div> <p>Choose Another Test Image</p> <p>Choose File : <input type="text" value="D/Algeo/Tugas/Tubes_2/tubes_algeo2/test/Datatest/C"/> <input type="button" value="Browse"/></p>
Analisis	<p>Variasi pencahayaan, angle:</p>

	<p>Program menebak dengan benar dengan nilai jarak hampir mendekati <i>threshold</i> bawah. Citra uji diambil dari angle yang berbeda dengan pencahayaan yang juga berbeda.</p> <p>Akan tetapi bukan masalah besar karena program membuang eigenvector pertama hasil eigenvalue terbesar, yang lebih menangkap perbedaan pencahayaan daripada fitur wajah. (Sumber: https://cseweb.ucsd.edu/classes/wi14/cse152-a/fisherface-pami97.pdf)</p>
Test Case 6	<p>Face Recognition</p> <p>Time needed to process dataset: 22755</p> <p>EUCLIDEAN DISTANCE: 342339744</p> <p>Person not in database</p> <p>Test Image</p>  <p>Closest Result</p>  <p>Choose Another Test Image</p> <p>Choose File : <input type="text"/> <input type="button" value="Browse"/></p>
Analisis	<p>Image tidak mengandung orang:</p> <p>Program tidak dapat mendeteksi wajah pada gambar uji serta jarak melebihi <i>threshold</i> atas sehingga dinyatakan tidak ada gambar pada dataset yang mirip. Namun, tetap ditampilkan gambar dengan jarak euclidian terdekat. Citra uji tidak mengandung wajah, sehingga auto-crop akan gagal dan langsung memproses citra uji yang diterima.</p>
Uji Eksploratif 1	

	 <p>Time needed to process dataset: 0.61 Face Recognition EUCLIDEAN DISTANCE: 7483291 Person in database!</p> <p>Test Image Closest Result</p> <p>Choose Another Test Image</p> <p>Choose File : C:\Users\ASUS\Desktop\tubes_a1geo2\test\dataset_itbin\ Browse</p>
	 <p>Time needed to process dataset: 0.61 Face Recognition EUCLIDEAN DISTANCE: 0 Person in database!</p> <p>Test Image Closest Result</p> <p>Choose Another Test Image</p> <p>Choose File : Browse</p>
Analisis	<p>Uji eksploratif:</p> <p>Berikut merupakan hasil uji dari dataset kecil ITB yang terdiri dari anggota tim dan pak Rinaldi Munir. Hasil yang diperoleh cukup memuaskan.</p>
Uji eksploratif 2	 <p>CapeFace/Camera Face Recognition Time needed to process dataset: 0.37 EUCLIDEAN DISTANCE: 10540014 Person in database! Photo will be taken every 15 seconds</p> <p>Photo from Camera Closest Result</p>
Analisis	<p>Uji eksploratif, Bonus:</p> <p>Berikut merupakan hasil pengujian dari fitur bonus, yaitu pengambilan foto dan face recognition secara real-time. Pengambilan foto diambil setiap 15 detik sekali. Hasil yang diperoleh cukup memuaskan.</p>

Uji
eksploratif 3



Analisis Uji eksploratif Bonus:

Hasil pengujian aplikasi dengan fitur kamera dan *background* bangunan.

Bab V

Kesimpulan, Saran, dan Refleksi

Pengenalan wajah atau *face recognition* dapat dibuat penerapannya dengan memanfaatkan prinsip nilai-nilai dan vektor-vektor eigen suatu matriks representatif citra gambar. Terdapat banyak teknik yang dapat digunakan dalam memperoleh nilai maupun vektor eigen suatu matriks. Salah satu algoritma yang dapat digunakan khusus untuk matriks simetris dengan elemen-elemen bilangan real adalah *Jacobi eigenvalue algorithm*. Algoritma ini dapat digunakan pada aplikasi *face recognition* ini karena matriks kovarian yang diperoleh dalam algoritma pencarian *eigenface* dijamin simetris dan semua elemen matriks tersebut bilangan real. Setelah diperoleh nilai-nilai eigen, vektor-vektor eigen dapat diperoleh dengan mencari *null space* persamaan karakteristik untuk masing-masing nilai eigen yang bersangkutan. Setelah itu, bobot *eigen face* masing-masing gambar pada dataset dapat dicari dan kemudian tahapan pengenalan wajah dapat dilakukan.

Dalam membandingkan suatu gambar uji dengan gambar-gambar dataset, dihitung jarak euclidian bobot *eigenface* untuk gambar uji dengan masing-masing bobot *eigenface* untuk dataset tersebut. Selain itu, perlu juga ditentukan suatu *threshold* yang digunakan dalam acuan perbandingan dan menentukan apakah wajah pada gambar uji merupakan wajah yang mirip dengan suatu wajah pada dataset. Pada aplikasi ini, ditentukan dua nilai *threshold* (bawah dan atas). Apabila jarak terdekat lebih kecil dibandingkan nilai *threshold* bawah, maka gambar uji diputuskan mirip dengan gambar yang memiliki jarak terdekat tersebut. Apabila nilai jarak berada di antara dua buah *threshold*, program tidak dapat menentukan dengan pasti kemiripannya sehingga gambar dengan jarak terdapat mungkin memiliki wajah seseorang yang sama maupun berbeda. Terakhir, program akan memastikan bahwa wajah pada gambar uji tidak memiliki kemiripan pada semua gambar dataset jika nilai jarak lebih besar dibandingkan nilai *threshold* atas.

Aplikasi yang berhasil dibuat memiliki waktu eksekusi yang cukup besar ketika dataset memiliki lebih dari 200 gambar. Optimisasi *Jacobi eigenvalue algorithm* dapat dilakukan untuk meningkatkan performa aplikasi. Salah satu optimisasi yang dapat dilakukan adalah dengan *Scheduled Relaxation Jacobi method* (sumber: <https://www.sciencedirect.com/science/article/pii/S002199911630198X>).

Selain masalah efisiensi algoritma, akurasi pengenalan wajah pada aplikasi ini belum dapat dikatakan akurat. Salah satu hal yang dapat meningkatkan akurasi tersebut adalah dengan melakukan *image pre-processing* lebih jauh. Program saat ini hanya melakukan *auto-crop* terhadap wajah pada gambar (baik gambar dataset maupun gambar data uji) sehingga program belum dapat mengeliminasi faktor-faktor lain yang menyebabkan ketidakakuratan hasil perhitungan, seperti ketidakseragaman latar belakang gambar (latar belakang tetap ada walaupun sudah dilakukan *auto-crop*), gambar *mirror*, serta posisi dan arah wajah yang tidak seragam. Selain itu, algoritma program masih belum dapat menentukan fitur-fitur penting

yang perlu diperhatikan sehingga program memperhatikan semua pixel pada gambar dan dapat mengganggu akurasi.

Pembuatan program ini telah banyak menambah wawasan kelompok mengenai algoritma-algoritma pencarian nilai eigen dan vektor eigen serta sedikit wawasan dan pengenalan tentang *machine learning*. Kelompok juga menyelesaikan program dengan kompak dan saling bekerja sama sehingga kesulitan dalam prosesnya dapat cepat teratasi. Setiap anggota kelompok berterima kasih karena telah dapat bekerja sama dengan baik serta juga berterima kasih dengan tugas yang telah menambah ilmu dan pengalaman kelompok.

Referensi

- Howard Anton, Elementary Linear Algebra, 11th edition, John Wiley and Sons, 2010.
- <https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/>
- https://en.wikipedia.org/wiki/Jacobi_rotation
- https://en.wikipedia.org/wiki/Jacobi_eigenvalue_algorithm
- <https://www.andreinc.net/2021/01/25/computing-eigenvalues-and-eigenvectors-using-qr-decomposition>
- <https://www.sciencedirect.com/science/article/pii/S002199911630198X>
- <https://cseweb.ucsd.edu/classes/wi14/cse152-a/fisherface-pami97.pdf>

Lampiran

GitHub : <https://github.com/williamnixon20/Algeo02-21123>

Demo : <https://youtu.be/HjBvDzwtg-0>