

APLIKASI NILAI EIGEN DAN EIGENFACE PADA APLIKASI PENGENALAN WAJAH (*FACE RECOGNITION*)

LAPORAN TUGAS BESAR

**Diajukan sebagai salah satu tugas mata kuliah Aljabar Linear dan Geometri pada
Semester II Tahun Akademik 2022/2023**

oleh

Henry Anand Septian Radityo	13521004
Matthew Mahendra	13521007
Ahmad Nadil	13521024



**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2022**

DAFTAR ISI

DAFTAR ISI	2
BAB I	1
I.I Spesifikasi Tugas	1
BAB II	2
II.I Perkalian Matriks	2
II.II Nilai dan Vektor Eigen	2
II.III Eigenface	3
BAB III	5
III.I Bahasa Program	5
III.II Input Gambar	5
III.III Operasi Matriks	5
III.IV Nilai Eigen dan Vektor Eigen	5
III.V Eigenface dan Euclidean Distance	6
III.VI Implementasi Kakas	6
BAB IV	7
IV.I Dataset	7
IV.II Pengenalan Wajah dari Dataset	7
IV.III Pengenalan Wajah dari Luar Dataset	8
IV.IV Handling Dataset yang Belum Dipilih	12
IV.V Penggunaan Kamera sebagai Masukan	13
BAB V	14
V.I Kesimpulan	14
V.II Saran	14
V.III Refleksi	14
LAMPIRAN	17

BAB I

DESKRIPSI MASALAH

I.I Spesifikasi Tugas

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

1. Program menerima input folder dataset dan sebuah gambar citra wajah.
2. Basis data wajah dapat diunduh secara mandiri melalui <https://www.kaggle.com/datasets/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.

BAB II

TEORI SINGKAT

II.I Perkalian Matriks

Operasi perkalian dapat dilakukan pada matriks. Misalkan ada matriks A_{mn} dan matriks B_{ij} . Syarat dari perkalian matriks adalah $n = i$. Hasil perkalian akan menghasilkan sebuah matriks C_{mj} . Proses perkalian adalah mengalikan setiap baris ke- x pada matriks A dengan kolom ke- y B lalu menjumlahkannya untuk membuat elemen C_{xy} atau secara notasi algoritmik dapat dibuat sebagai berikut,

```
if (n = i) then
    x traversal 0..m
        y traversal 0..j
            Cxy ← 0
            z traversal 0..i
                Cxy ← Cxy + (Axz × Bzy)
```

Kasus di atas adalah untuk perkalian matriks dengan matriks. Untuk perkalian matriks skalar, maka skalar k dikalikan ke setiap elemen pada matriks.

II.II Nilai dan Vektor Eigen

Untuk setiap matriks A_{nn} maka vektor x tidak nol di R^n adalah vektor eigen A jika memenuhi persamaan

$$Ax = \lambda x$$

dengan λ adalah skalar yang disebut nilai eigen dari A dan berpasangan dengan vektor x .

Penyelesaian x dan λ dapat diselesaikan melalui metode karakteristik polinomial yang memanfaatkan persamaan,

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

dengan nilai eigen ditentukan dari penyelesaian $\det(A - \lambda I)$ dan vektor x setelah mendapatkan nilai eigen disubstitusikan ke persamaan di atas. Akan tetapi untuk ukuran matriks yang lebih besar, menurut teori Abel-Ruffini, tidak ada solusi dalam radikal untuk polinomial berderajat lima ke atas. Untuk itu dimanfaatkan metode lain untuk mencari nilai eigen dan vektor eigen.

Dalam tugas ini, kelompok memanfaatkan algoritma QR dimana Q adalah matriks singular dan R adalah matriks segitiga atas. Q dan R didapatkan dengan dekomposisi dari matriks A atau umumnya disebut dekomposisi QR. Nilai eigen didapatkan dengan mengalikan R dengan Q dan diulang beberapa kali agar hasilnya ter-konvergen. Diagonal utama dari perkalian matriks tersebut adalah nilai eigen. Untuk vektor eigen, dibuat matriks identitas yang ukurannya sama dengan A, lalu matriks tersebut dikalikan dengan Q. Matriks yang dihasilkan adalah vektor-vektor eigen yang urutannya bersesuaian dengan nilai-nilai eigen (urutan pada diagonal utama) dan nilainya mengurut mengecil.

II.III Eigenface

Algoritma eigenface memanfaatkan nilai eigen dan vektor eigen untuk melakukan face recognition. Garis besar algoritma ini adalah sebagai berikut,

1. Dari kumpulan dataset gambar pelatihan (*training images*), dinormalisasi menjadi grayscale dan digabung menjadi himpunan matriks S
2. Dari S, dihitung rata-rata ψ dari keseluruhan matriks S. Hasil dari rata-rata tersebut dapat menghasilkan citra seperti gambar di bawah ini,



3. Dari S, dihitung selisihnya dengan ψ yang menghasilkan himpunan selisih ϕ
4. Dari ϕ yang didapatkan, dihitung matriks kovarian C dengan rumus

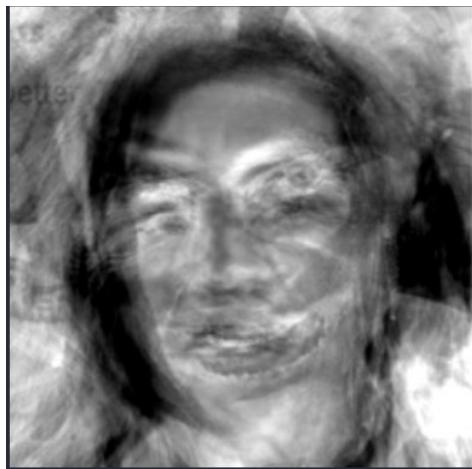
$$C = \frac{1}{M} \sum_{n=1}^M \Phi_n (\Phi_n)^T = AA^T$$

dengan M adalah jumlah matriks pada himpunan Φ , A adalah himpunan selisih matriks

5. Kemudian dihitung nilai eigen dan vektor eigen dari C
6. Setelahnya dihitung eigenface μ dengan rumus

$$\mu_i = \sum_{k=1}^M \Phi_k x_{ik}$$

Contoh eigenface adalah sebagai berikut,



7. Dari sejumlah *eigenface* diambil beberapa saja (misalnya sebanyak K). Alasannya karena *eigenface* yang di bawah mengalami *noise*. Dari K *eigenface* ditentukan *weight* w dengan mengalikan μ_k dengan Φ_k
8. Selanjutnya dilakukan proses pengenalan wajah dengan menerapkan langkah 1-3 dilanjutkan dengan 7 pada gambar yang akan dikenali. Hasil pada langkah 7 menghasilkan weight new face.
9. Setelahnya dilakukan jarak terkecil dari *euclidean distance* dengan persamaan

$$\epsilon_k = ||\Omega - \Omega_k||$$

nilai ϵ terkecil adalah wajah yang sama. Ω adalah himpunan weight.

10. Dari *euclidean distance* yang didapatkan, dibandingkan dengan nilai *euclidean distance* maksimumnya dan diberikan batas error. Jika tidak melebihi batas error, maka wajah dapat dikenali, selain itu wajah tidak dapat dikenali. Batas adalah threshold T yaitu,

$$T = 0.15\epsilon_k$$

BAB III

IMPLEMENTASI PROGRAM

III.I Bahasa Program

Program dibuat menggunakan bahasa Python 3 dengan menggunakan kakas OpenCV, Numpy untuk membantu proses manipulasi gambar dan perhitungan pada matriks. GUI diimplementasikan dengan menggunakan kakas Tkinter dari Python.

III.II Input Gambar

Input gambar direalisasikan pada file InputImage.py. Pada kakas ini, ada operasi untuk mengubah sebuah folder dataset menjadi himpunan gambar Γ , dan mengubah gambar menjadi ukuran 256x256.

Fungsi yang digunakan adalah ImgToMatrix(filename : string) dan DataSetToMatrix(dir : string).

III.III Operasi Matriks

Operasi Matriks direalisasikan pada file OperasiMatriks.py. Pada kakas ini, ada operasi untuk melakukan perhitungan rata-rata matriks dalam Γ , penjumlahan dua matriks, operasi selisih matriks, dan operasi membuat kovarian matriks.

Fungsi yang digunakan adalah SumOfMatrix(M1,M : array of array of integer), RataRataMatrix(S : array of array of integer), Selisih(S: array of array of array of integer, data : integer), dan Kovarian(S: array of array of array of integer, data : integer)

III.IV Nilai Eigen dan Vektor Eigen

Fungsi untuk mencari nilai dan vektor eigen direalisasikan dalam Eigen.py. Pada kakas ini terdapat operasi untuk mencari vektor satuan dari suatu vektor, melakukan dot product vektor, mencari vektor ke- k pada suatu matriks, dan melakukan dekomposisi QR dengan mengubah matriks menjadi Q, R.

III.V Eigenface dan Euclidean Distance

Fungsi untuk mencari eigenface serta euclidean distance direalisasikan dalam Eigenface.py. Kakas ini digunakan untuk proses pengenalan wajah. Terdapat fungsi untuk membuat eigenface dan mencari weightnya, fungsi untuk mencari weight dari wajah baru, dan fungsi untuk mencari euclidean distance. Pada euclidean distance, juga dikeluarkan boolean apakah suatu wajah dikenali atau tidak melalui tes pada threshold.

III.VI Implementasi Kakas

Dari kelas yang telah dibuat, implementasinya adalah sebagai berikut,

```
# Siapkan himpunan S
S = II.DataSetToMatrix(dir)

# Hitung rata-rata dari himpunan S
mean = OM.RataRataMatrix(S)

# Hitung selisih dari himpunan S
s2 = OM.Selisih(S, len(S))

# Buat Kovarian
cov = OM.kovarian(s2, len(s2))

# Hitung EigenVector dari Kovarian
eigenval, eigenvec = Eig.getEigen(cov)

# Hitung Weight Dataset Images
eigface,weightf = EigenFace(eigenvec, s2, S)

# Hitung Weight Test Image
weightnf = EigF.EigenNewFace(path,mean,eigface)

# Menghitung Euclidean Distance dan mengembalikan id gambarnya
idx,th = EigF.EuclideanDistance(weightf,weightnf)
```

BAB IV

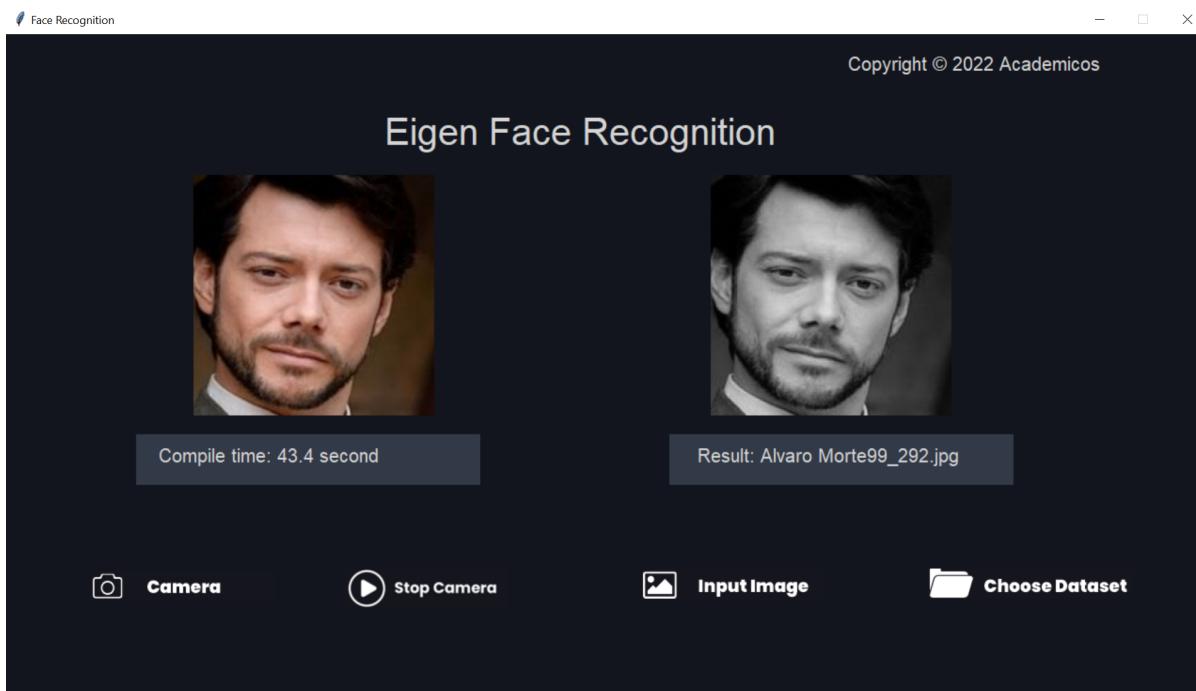
EKSPERIMENT

IV.I Dataset

Disediakan tiga dataset bernama dataset1, dataset2, dan dataset3. Dataset1 terdiri dari 6 wajah orang yang berbeda, masing-masing terdiri dari 5 gambar wajah. Dataset2 terdiri dari 4 wajah orang yang berbeda, masing-masing terdiri dari 20 gambar wajah. Dataset3 terdiri dari 5 wajah orang yang berbeda, masing-masing terdiri dari 10 gambar wajah.

Dataset1 terdiri dari wajah Adriana Lima, Amber Heard, Ben Affleck, Bill Gates, Elon Musk, dan Tom Hiddleston. Dataset2 terdiri dari wajah Dwayne Johnson, Alvaro Morte, Gwyneth Paltrow, dan Andy Samberg. Dataset3 terdiri dari wajah Elizabeth Olsen, Emma Watson, Jason Momoa, Tom Cruise, dan Zac Efron.

IV.II Pengenalan Wajah dari Dataset



Pengenalan wajah dari dalam dataset akan menghasilkan gambar itu langsung karena euclidean distance yang dihasilkan adalah 0. Foto diambil dari dataset2.

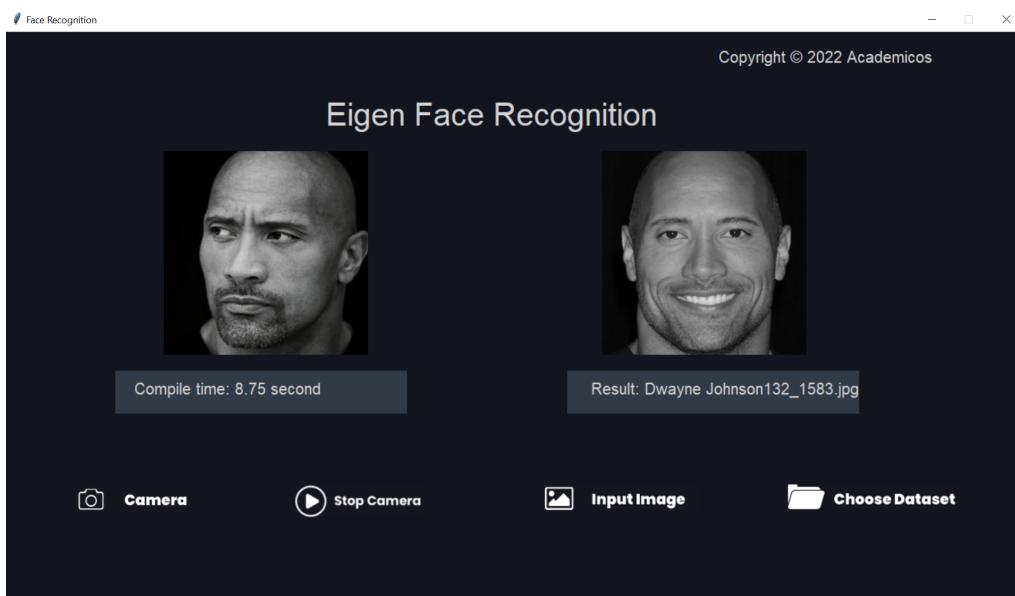
IV.III Pengenalan Wajah dari Luar Dataset

Foto yang digunakan tidak ada dalam dataset. Program mengeluarkan foto Dwayne Johnson.

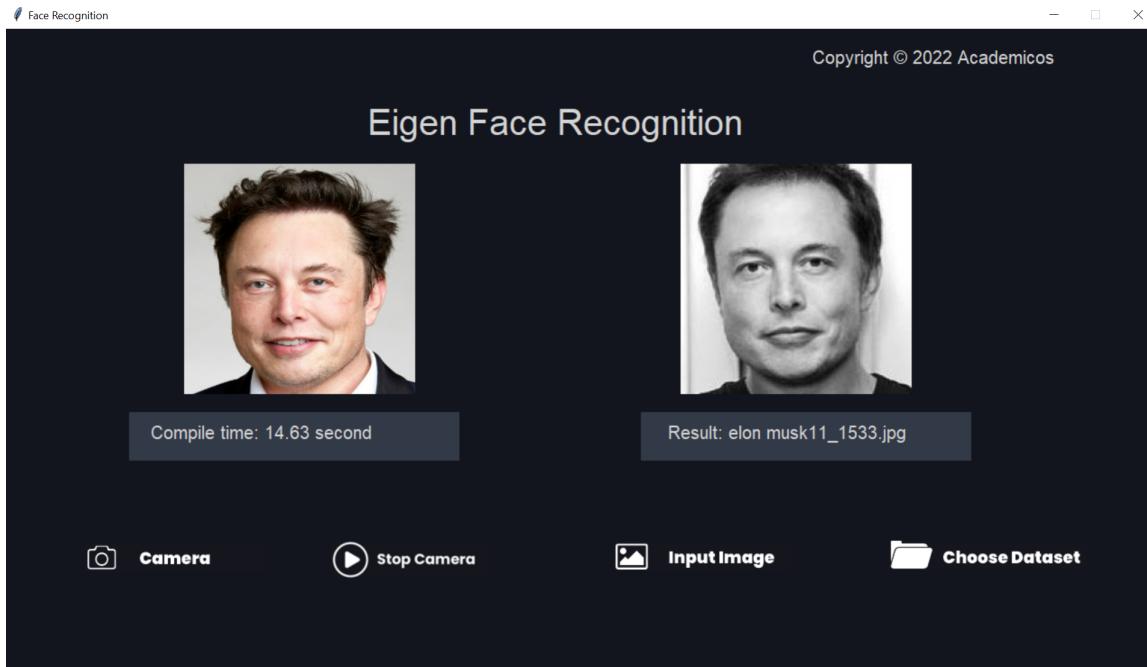
Dataset yang digunakan adalah dataset2



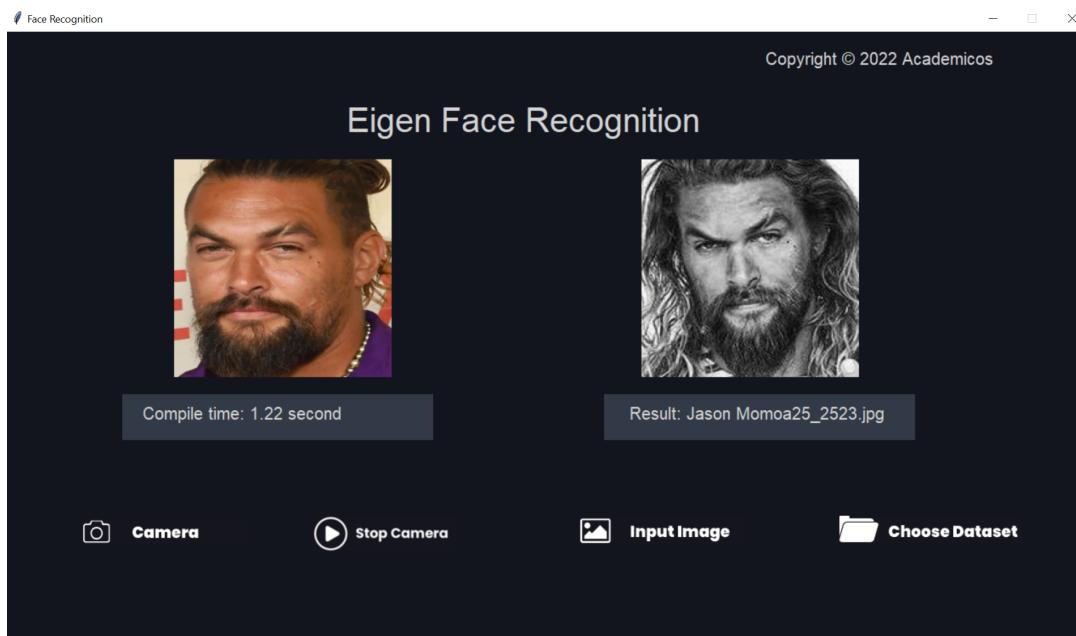
Kasus yang sama untuk Dwayne Johnson namun dengan wajah yang dilihat dari samping



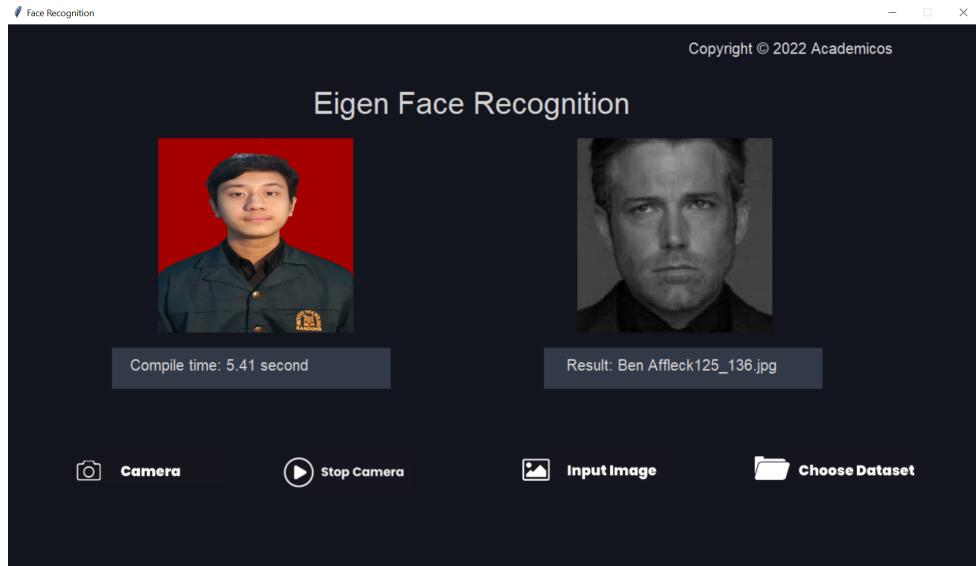
Pada dataset2, digunakan wajah dari luar dataset yang ukurannya tidak menggunakan rasio 1:1. Hasilnya sebagai berikut,



Kasus berikutnya menggunakan dataset3 dengan citra yang tidak memiliki rasio 1:1. Hasilnya adalah sebagai berikut,

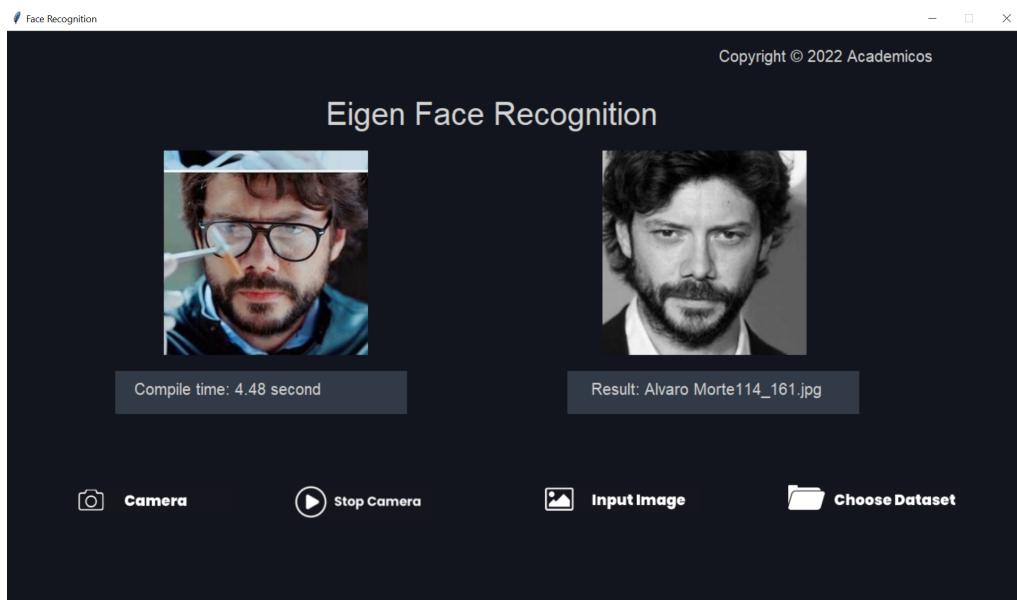


Wajah input akan dimampatkan menjadi 256×256 . Karena itu untuk kasus untuk wajah yang berukuran tidak kotak (rasio 1:1) dan muka tidak di-center dapat memberikan error terhadap hasil dari pengenalan wajah seperti gambar di bawah ini,

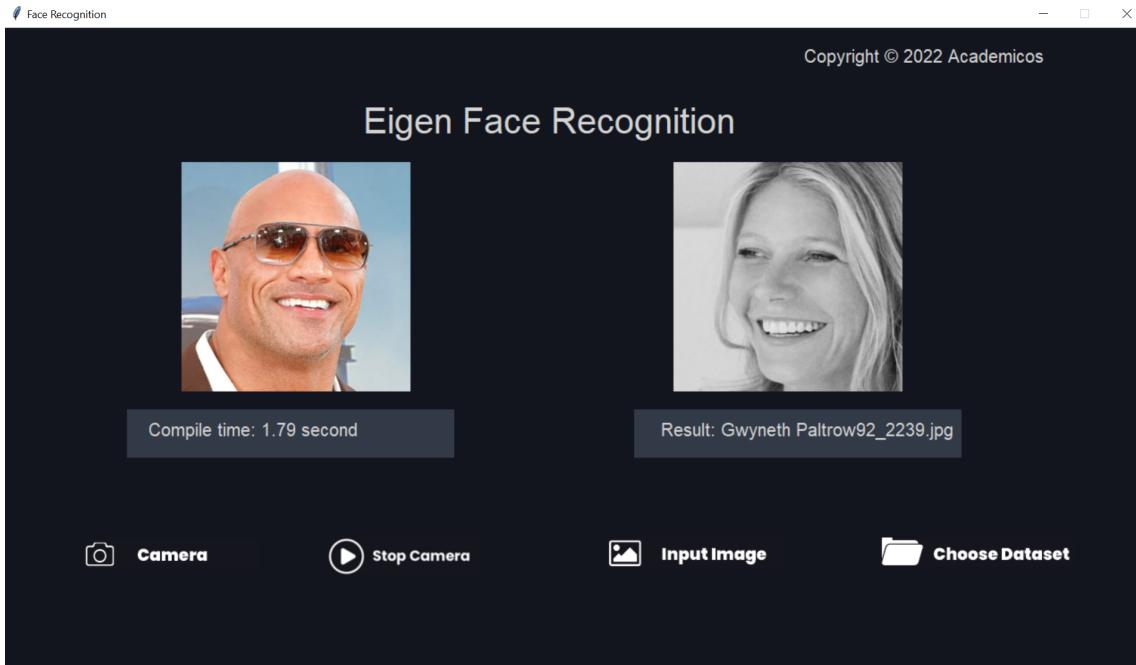


Program seharusnya memberikan gambar yang tidak dikenali, akan tetapi masih menghasilkan wajah yang dianggap paling serupa. Dataset yang digunakan adalah dataset1

Penggunaan atribut pada wajah yang akan dikenali, selama tidak terlalu memengaruhi wajah dapat tetap dikenali, misalnya untuk kacamata yang terkena pantulan masih dapat dikenali seperti kasus di bawah ini pada dataset2,

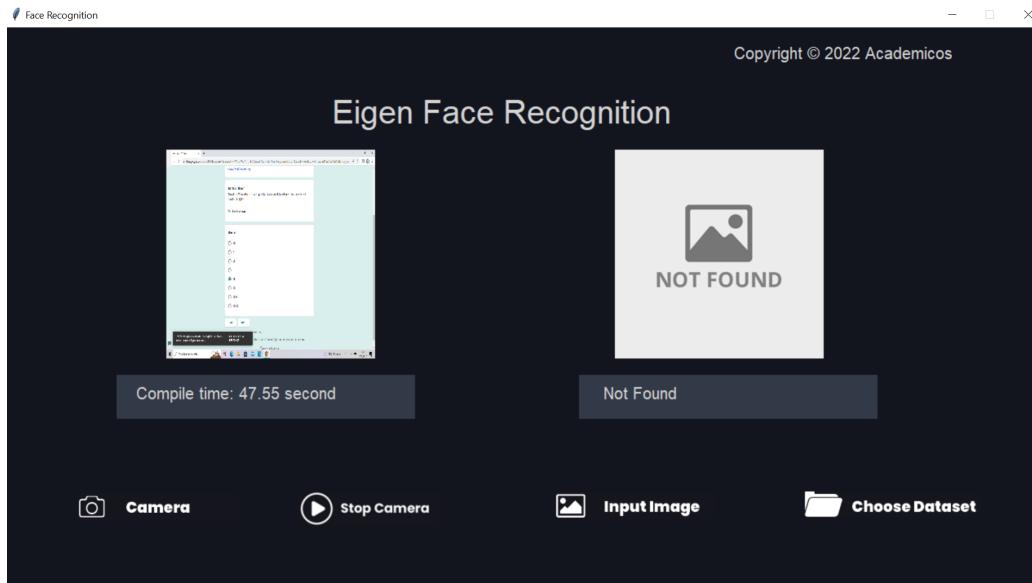


Akan tetapi, untuk kasus penggunaan atribut yang memengaruhi wajah seperti kacamata hitam, dapat memengaruhi hasil. Sebagai contoh,

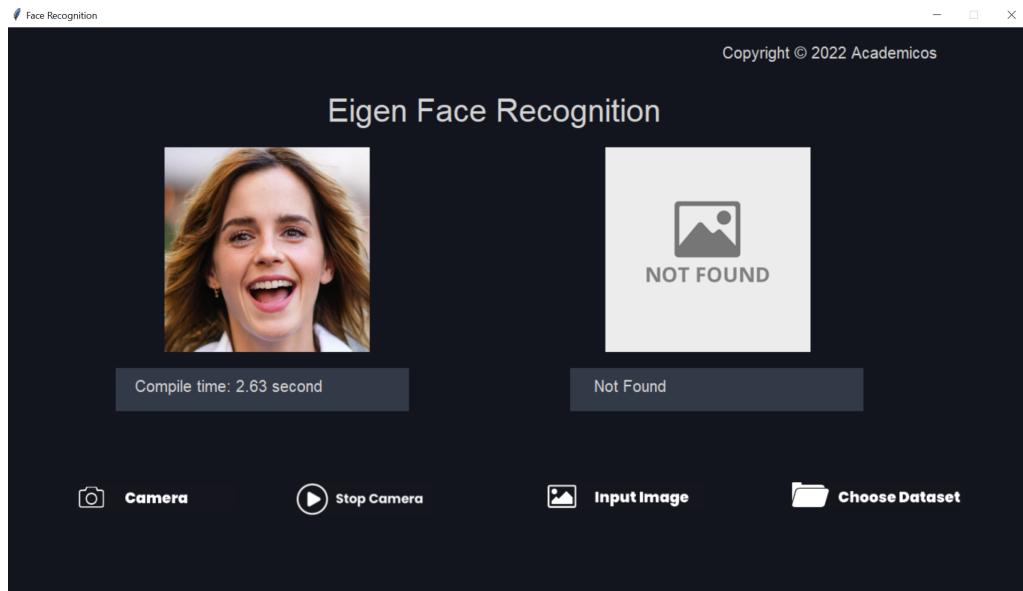


IV.IV Pengenalan Wajah yang Tidak Dikenal dalam Dataset

Digunakan dataset1. Digunakan gambar yang tidak memiliki wajah seperti di bawah ini.



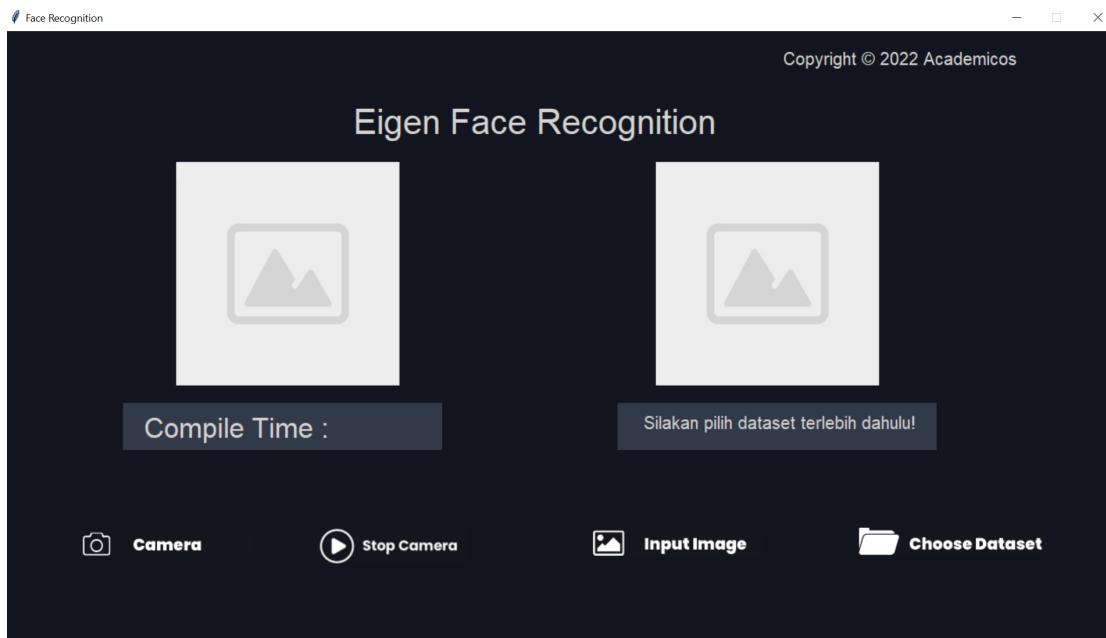
Karena gambar bukan merupakan wajah, maka tidak dikenali dalam dataset



Wajah yang digunakan ada pada dataset3. Oleh karena itu,wajah tidak ada dalam dataset sehingga tidak dikenali.

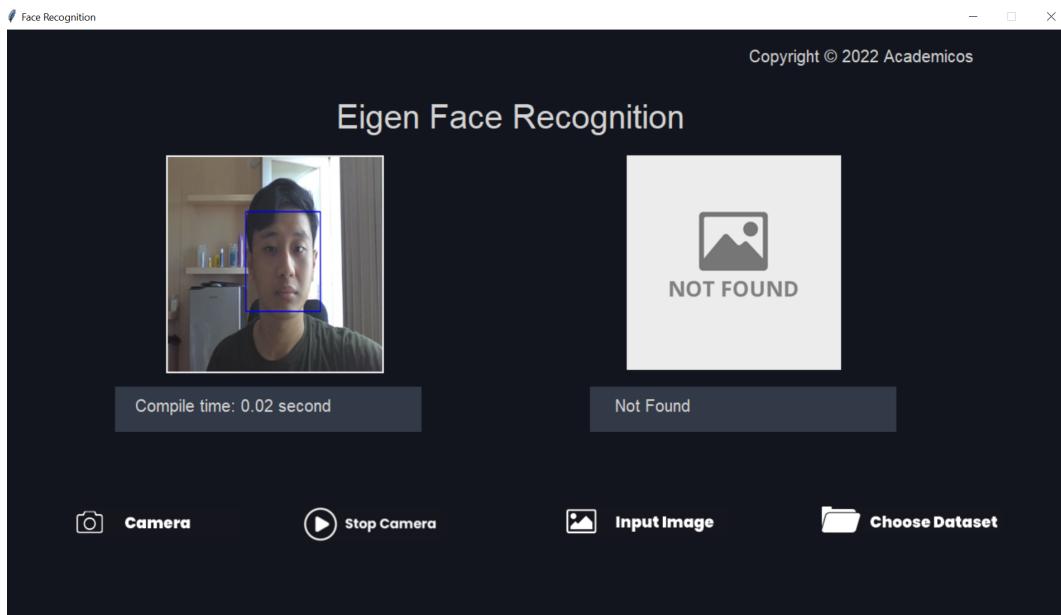
IV.IV Handling Dataset yang Belum Dipilih

Pengguna tidak memasukkan dataset, maka menampilkan error.

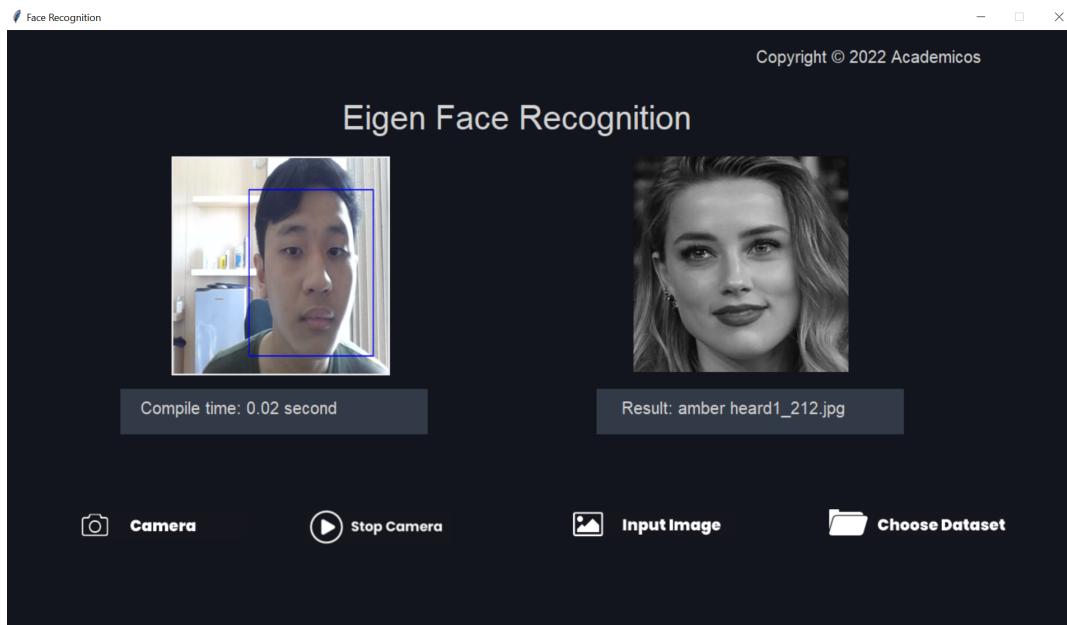


IV.V Penggunaan Kamera sebagai Masukan

Sama seperti kasus sebelumnya, wajah yang dimasukkan melalui kamera tidak ada dalam dataset1. Oleh sebab itu tidak dikenali oleh program. Kamera akan mengambil citra setiap 10 detik sekali. Hasilnya seperti di bawah ini,



Akan tetapi, kesalahan masih dapat terjadi apabila muka karena kasus penerangan pada wajah seperti contoh di bawah ini.



BAB V

KESIMPULAN DAN SARAN

V.I Kesimpulan

Dari tugas ini disimpulkan hal-hal sebagai berikut,

1. pengenalan wajah dapat menggunakan algoritma eigenface yang menghitung euclidean distance dari wajah yang akan dikenali dengan wajah yang pada dataset
2. perhitungan eigenface menggunakan eigenvector dan nilai eigen
3. nilai eigen dan vektor eigen dapat ditentukan menggunakan cara lain selain mencari persamaan karakteristik yaitu menggunakan QR Algorithm atau yang lainnya

V.II Saran

Untuk tugas dan aplikasi dari algoritma eigenface, diberikan saran sebagai berikut,

1. melakukan proses algoritma QR dengan lebih efektif untuk mengurangi waktu jalan program
2. menggunakan gambar test dan dataset yang berukuran relatif sama untuk akurasi serta pencahayaan yang kurang lebih sama
3. dalam proses pengenalan, wajah harus ditengahkan dan harus pada posisi yang lurus untuk keakuratan terbaik. Solusi lainnya adalah menggunakan operasi transformasi untuk memetakan wajah sehingga lurus dan lebih mudah untuk dikenali
4. untuk penggunaan kamera, harus dipastikan kamera yang digunakan harus memiliki resolusi yang baik agar dapat menangkap citra wajah dengan baik. Begitu pula untuk mengambil citra untuk digunakan pada dataset

V.III Refleksi

Dari penggeraan tugas ini, kelompok belajar untuk berpikir kreatif dan tidak terpaku pada satu materi saja, namun mencoba untuk mencari materi tambahan untuk lebih memahami tugas yang diberikan ini. Kelompok juga belajar untuk bekerja sama dengan efektif serta belajar untuk membagi waktu tugas-tugasnya.

Kelompok belajar bahwa aplikasi nilai eigen dan vektor eigen dapat digunakan pada berbagai permasalahan komputasi, salah satunya adalah pengenalan citra wajah. Selain itu, ada berbagai algoritma yang dapat digunakan untuk mendekomposisi dan mencari nilai eigen dan vektor eigen, salah satunya QR Algorithm. Kelompok juga belajar terkait kompleksitas program dan cara membuat suatu program efektif melalui analisa O notation dan perulangan yang ada.

REFERENSI

- Elvis, “Finding matrix eigenvectors using QR decomposition,” *Stack Exchange*, 01-Mar-1959. [Online]. Available: <https://stats.stackexchange.com/questions/20643/finding-matrix-eigenvectors-using-qr-decomposition>. [Accessed: 16-Nov-2022].
- G. For Geeks, “ML: Face recognition using eigenfaces (PCA algorithm),” *GeeksforGeeks*, 24-Sep-2021. [Online]. Available: <https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/>. [Accessed: 16-Nov-2022].
- I. Yanovsky, “QR Decomposition with Gram-Schmidt.” University of California, Los Angeles.
- J. M. Valverde, “[explanation] face recognition using eigenfaces,” *Lipman's Artificial Intelligence Directory*, 05-Oct-2015. [Online]. Available: <http://laid.delanover.com/explanation-face-recognition-using-eigenfaces/>. [Accessed: 16-Nov-2022].

LAMPIRAN

Pranala Github: <https://github.com/IceTeaXXD/Algeo02-21004>

Pranala Bonus B: <https://youtu.be/wheuIME0sqI>