LAPORAN TUGAS BESAR 02

IF2123 ALJABAR LINEAR GEOMETRI

Kelompok FMA

Disusun oleh:

Alexander Jason	13521100
Farhan Nabil Suryono	13521114
Michael Utama	13521137

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

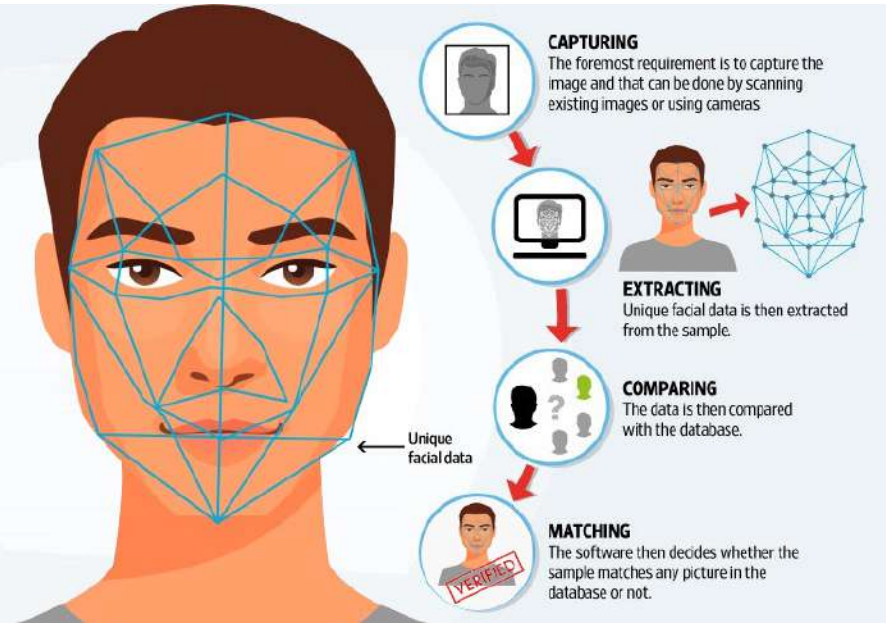
2022

DAFTAR ISI

BAB 1

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. 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 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:

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

$l = 1, \ldots, M$

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.

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

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

2. Gunakan metode euclidean distance untuk mencari jarak (distance) terpendek antara nilai eigen dari training image dalam database dengan nilai eigen dari image testface. $\varepsilon k = \Omega - \Omega k$ (8)

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 2

TEORI SINGKAT

2.1 Perkalian Matriks

Perkalian matriks adalah nilai pada matriks yang bisa dihasilkan dengan cara dikalikan-nya tiap baris dengan setiap kolom yang memiliki jumlah baris yang sama. Setiap anggota matriks ini nantinya akan dikalikan dengan anggota elemen matriks lainnya.

Perkalian matriks ini dilakukan sesuai urutan dan aturan yang berlaku pada perkalian bilangan matriks. Saat sedang menghitung nilai suatu matriks, berarti akan melihat adanya kolom dan juga baris. Kolom dan baris digunakan untuk menentukan maupun menghitung nilai matriks. Pada dasarnya kolom dan baris sangat diperlukan dalam penghitungan matriks.

Dalam matematika, perkalian matriks adalah suatu operasi biner dari dua matriks yang menghasilkan sebuah matriks. Agar dua matriks dapat dikalikan, banyaknya kolom pada matriks pertama harus sama dengan banyaknya baris pada matriks kedua. Matriks hasil perkalian keduanya, akan memiliki baris sebanyak baris matriks pertama, dan kolom sebanyak kolom matriks kedua. Perkalian matriks A dan B dinyatakan sebagai AB.

Jika **A** adalah matriks berukuran m x n dan **B** adalah matriks berukuran n x p, dengan elemen sebagai berikut:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix}$$

Hasil perkalian kedua matriks tersebut adalah matriks **C** yang berukuran m x p

2.2 Nilai Eigen

Jika A adalah sebuah matriks n x n, maka sebuah vektor tak nol x pada R^n disebut **vektor eigen** dari A jika A **x** sama dengan perkalian suatu skalar λ dengan **x**, yaitu

$$A\mathbf{x} = \lambda\mathbf{x}$$

Skalar λ disebut **nilai eigen** dari A, dan **x** dinamakan vektor eigen yang berkoresponden dengan λ. Untuk memperoleh nilai eigen dari sebuah matriks A berukuran n x n, maka persamaan di atas dapat ditulis kembali sebagai:

$$A\mathbf{x} = \lambda\mathbf{I}\mathbf{x}$$

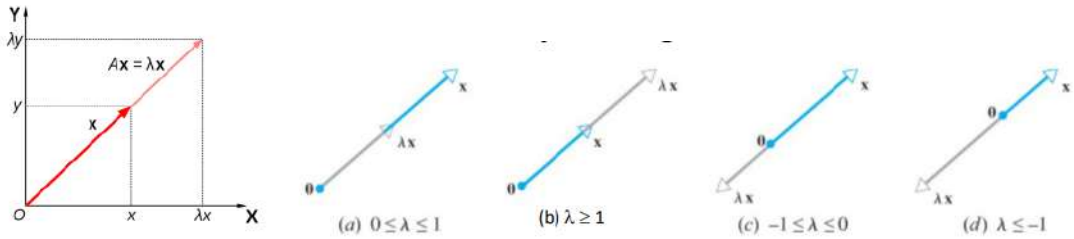
Berdasarkan persamaan tersebut, λ merupakan nilai eigen jika terdapat solusi tak nol, kemudian persamaan tersebut memiliki solusi tak nol jika dan hanya jika

$$\det(\lambda\mathbf{I} - A) = 0$$

Persamaan tersebut akan membentuk polinomial dari matriks A, dan skalar-skalar yang memenuhi adalah nilai-nilai eigen dari matriks A.

2.3 Vektor Eigen

Vektor eigen **x** menyatakan matriks kolom yang apabila dikalikan dengan sebuah matriks n x n menghasilkan vektor lain yang merupakan kelipatan vektor itu sendiri. Dengan kata lain, operasi A**x** = λ**x** menyebabkan vektor x menyusut atau memanjang dengan faktor λ dengan arah yang sama jika λ positif, dan arah berkebalikan jika λ negatif.



2.4 Eigen Face

Eigenface merupakan metode yang digunakan untuk melakukan ekstraksi citra wajah. Dasar dari metode ini ialah Principal Component Analysis (PCA). Metode ini melakukan proyeksi dari ruang citra dengan dimensi yang lebih rendah untuk meningkatkan efisiensi dalam proses komputasi dan mengurangi storage yang diperlukan serta dapat memaksimalkan jarak antara semua kelas wajah.

Analisis algoritma eigenface pada pengenalan wajah merupakan dasar bagi seseorang yang ingin mendalami ilmu biometrik karena identifikasi dilakukan dengan pattern matching sederhana tanpa menggunakan metode pembelajaran khusus. Metode eigenface ini banyak digunakan karena memiliki tingkat akurasi yang cukup baik dan sistem komputasi yang tidak terlalu rumit. Berikut merupakan tahapan perhitungan eigenface:

1. Membuat suatu himpunan matriks yang terdiri dari seluruh training image yang ada di database $(\Gamma_1, \Gamma_2, \ldots \Gamma_M)$

$$\mathbf{Im} = (\boldsymbol{\Gamma_1}, \boldsymbol{\Gamma_2}, \ldots \boldsymbol{\Gamma_M})$$

Keterangan :

Im = matriks yang berisi nilai keseluruhan data

Γi = data ke-i

2. Setelah data diubah ke dalam bentuk matriks satu dimensi, maka akan terbentuk pula suatu matriks besar yang berisi seluruh data referensi. II-3 Kemudian tentukan nilai rata-rata (mean) data referensi tersebut dengan menggunakan persamaan :

$$\psi = \frac{1}{M} \sum_{n=1}^{M} \Gamma \mathbf{n}$$

Keterangan :

M = jumlah data referensi

Ψ = nilai rata-rata

Γn = data ke-n

3. Kemudian cari feature PCA atau ciri data (Φ) dengan mencari selisih antara Training image (Γi) dengan nilai tengah (Ψ), apabila ditemukan nilainya di bawah nol ganti nilainya dengan nol. Gunakan persamaan sebagai berikut:

$$\Phi_i = \Gamma_i - \Psi$$

Keterangan :

Φ_i = pola hasil ekstraksi data ke-*i*

Γ_i = data ke-*i*

Ψ = nilai rata-rata

4. Setelah rata-rata (mean) ditemukan, langkah selanjutnya ialah menghitung nilai matriks kovarian (C) dengan persamaan berikut :

$$\mathbf{C = AA^T}$$

Matriks A adalah matriks yang berisi informasi pola hasil hasil ekstraksi pada seluruh data yang ada

5. Setelah matriks kovarian ditemukan, hitung eigenvalue (λ) dan eigenvector (v) dengan persamaan berikut :

$$\mathbf{AX = \lambda X}$$

Berdasarkan alur proses di atas terlihat bagaimana proses awal sampai ditemukannya bobot dari citra referensi (citra dengan berbagai usia) dan citra uji. Bobot tersebut memberikan informasi ciri utama dari setiap citra.

2.5 Dekomposisi QR

Dekomposisi QR dapat dilakukan dengan beberapa cara, seperti *Gram-Schmidt* dan *Householder Triangularization*. Cara *Gram-Schmidt* cenderung kurang stabil karena banyak sumber-sumber error kecil yang mungkin dikali dengan sebuah pengali besar, sehingga mengakibatkan hasil yang tidak akurat. Cara *Householder* lebih stabil karena tiap iterasinya menggunakan upamatriks yang semakin mengecil, sehingga tidak ada nilai yang mungkin menghasilkan error.

Berikut adalah langkah-langkah dekomposisi QR dengan *Householder Triangularization* pada matriks M.

1. Membentuk *Householder reflector* F, yaitu

$$F = I - 2\frac{v \times v^T}{v^T \times v}$$

Dengan $v = sign(x_1)||x||e_1 + x$

Dan x adalah kolom pertama dari matriks M

2. Membentuk matriks ortogonal Q_k, yaitu

$$Q_k = \begin{bmatrix} I & 0 \\ 0 & F \end{bmatrix}$$

Dengan I adalah matriks identitas berukuran $(k - 1) * (k - 1)$ dan F adalah *Householder reflector*(transformasi cerminan dengan subspace $||x||e_1$ - x).

3. Kalikan Q_k dengan M
4. Hilangkan baris pertama dan kolom pertama dari matriks M.
5. Ulangi langkah 1-4 sampai matriks M kosong
6. Nilai matriks Q adalah

$$Q = \prod_{k=1}^{n} Q_k$$

Sedangkan matriks R dapat dibentuk dari nilai-nilai yang dihilangkan pada langkah 4.

Pada penerapannya, matriks Q_k memiliki banyak nilai yang tidak berpengaruh (identitas dan nol), sehingga hanya nilai F yang perlu dicatat.

BAB 3

IMPLEMENTASI PROGRAM

3.1 FOLDER SOURCE

a. addImage.py

Fungsi/Prosedur

Fungsi / Prosedur	Deskripsi
writeNewTrainingImage(filepath, img)	Menulis img menjadi sebuah foto baru pada folder training

b. eigenface.py

Fungsi / Prosedur	Deskripsi
takeFirst(elem)	Mengambil elemen pertama dari array elem. Kegunaannya untuk proses sorting
load(dbPath)	Melakukan load pck
computeMean()	Mengembalikan matriks rata-rata
diffMatrix(mean)	Mengembalikan matriks selisih
normaVektor(vec)	Mengembalikan besar norma vektor dari vektor vec
normalizeVec(vec)	Melakukan normalisasi vektor vec. Mengembalikan vec bila normanya 0
eigenFace(matCov, mat, useBuiltIn = False)	Mengembalikan eigenVector dan eigenFaceList dari data

	training image
main()	Fungsi utama dari eigenface. Memanggil fungsi-fungsi lain untuk membuat eigenData.pck

c. extract.py

Fungsi / Prosedur	**Deskripsi**
extract_features(imgPath, vector_size = 32)	Mengembalikan hasil ekstrak gambar pada imgPath
batch_extractor(images_path, pickled_db_path="features.pck")	Melakukan ekstrak semua gambar file

d. gui.py

Untuk GUI, kami menggunakan tkinter sebagai modul, dan Pillow untuk import image

Fungsi / Prosedur	**Deskripsi**
insert_img(self)	Ketika tombol tersebut ditekan dan image dimasukkan, maka proses eigenface dimulai
init_cam(self)	Menginisiasi Kamera
open_cam(self)	Fungsi untuk menampilkan kamera dan mengambil gambar lalu dibandingkan dengan training image setiap 15 detik
insert_folder(self)	Memasukkan nama folder untuk dataset

e. img_recognition.py

Fungsi / Prosedur	Deskripsi
loadImage(namaFileGambar)	Mengembalikan hasil extract image pada directory namaFileGambar
newEigenFace(eigenData, imageData)	Membuat EigenFace dari imageData dengan EigenVector dan Mean dari eigenData
euclideanDistance(newEigenFace, comparedEigenFace)	Menghitung Euclidean Distance antara newEigenFace dan comparedEigenFace
imgRecognition(namaFile)	Fungsi Utama img_recognition. Fungsi ini melakukan loadImage namaFile, membuat eigenFace, membandingkan dengan eigenFace seluruh training image, dan mengembalikan gambar termirip

f. qr_decomposition.py

Fungsi / Prosedur	Deskripsi
normaVektor(vec)	Mengembalikan besar norma vektor dari vektor vec
householder(matSrc)	Mengembalikan matriks Q dan R hasil dari QR factorization

3.2 FOLDER TEST

Folder ini berisi data test dari training image serta image untuk uji coba program.

3.3 Implementasi Algoritma

a. GUI

GUI menggunakan library Custom Tkinter, PIL, dan OS. Library Custom Tkinter digunakan untuk mendesain tampilan interface program. Library PIL digunakan untuk mengimport image dari inputan user dan file directory. Library OS digunakan untuk mencari nama file dari gambar ataupun folder.

Program terdapat pada file *gui.py*

b. OpenCV

Library OpenCV digunakan untuk mengambil gambar dari *webcam* dan menampilkannya pada GUI.

```
def open_cam(self):
    _, frame = self.cap.read()
    cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)
    self.img = Image.fromarray(cv2image)
    self.imgtk = ImageTk.PhotoImage(image=self.img)
    self.label_input.configure(image=self.imgtk)

    if (int(time.time() - self.startCamTime) == 5):
        start_time = time.time()
        self.dir = os.path.dirname(os.path.realpath(__file__))
        cv2.imwrite(os.path.join(self.dir, "../test/example/test.jpg"), frame)
        self.label_text_tes.configure(text = 'test.jpg')
        msg, isRecognized, fileName = img_recognition.imgRecognition(os.path.join(self.dir, "../test/example/test.jpg"))
        self.label_bot.configure(text =f'Execution Time: {round(time.time() - start_time,2)} second')
        self.label_text_res.configure(text = msg)
```

Keterangan:

Fungsi untuk membuka kamera dan menampilkannya

c. QR Decomposition

QR Decomposition diimplementasikan dengan *Householder triangularization*, dengan cara seperti teori di bab 2. Fungsi dibuat dengan bantuan numpy, terutama fungsi terkait numpy.array karena memiliki optimasi yang bagus. Program terkait terletak di file *qr_decomposition.py*.

d. Eigenvalue, eigenvector, dan eigenface

Eigenvalue dan eigenvector dicari dengan menggunakan QR algorithm, yaitu melakukan iterasi terhadap matriks asal berulang kali sehingga mendekati matriks segitiga atas. Eigenvalue adalah nilai tidak nol pada matriks yang sudah mengalami iterasi. Matriks berisi eigenvector didapat dari perkalian seluruh nilai Q pada tiap iterasi. Jumlah iterasi pada program dibuat konstan yaitu 1000 kali.

Nilai eigenface didapat dari proyeksi vektor *difference* terhadap K eigenvector dengan eigenvalue tertinggi. Nilai K dibuat konstan yaitu 60, karena sudah cukup mewakili dan tidak terlalu banyak memakan waktu komputasi.

Implementasi program dapat dilihat pada file *eigenface.py* dan *img_recognition.py*

e. Extract

Ekstraksi foto dilakukan dengan bantuan OpenCV, dengan memanggil fungsi cv2.imread(). Foto akan diproses lagi sehingga menghasilkan foto hasil *crop* yang berisi wajah terdeteksi. Setelah itu, ukuran foto akan diubah sehingga seragam, yaitu 256x256 pixel. Format warna foto diubah menjadi cv2.COLOR_BGR2GRAY. Matrix foto akan diubah dari ukuran 256x256 menjadi 65536x1 dengan fungsi flatten().

Implementasi program dapat dilihat pada file *extract.py*

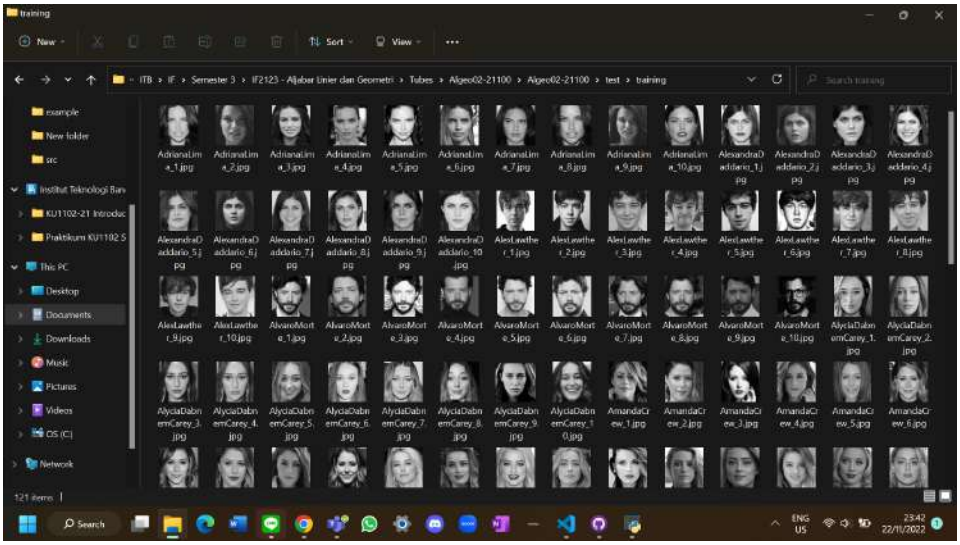
BAB 4

EKSPERIMEN

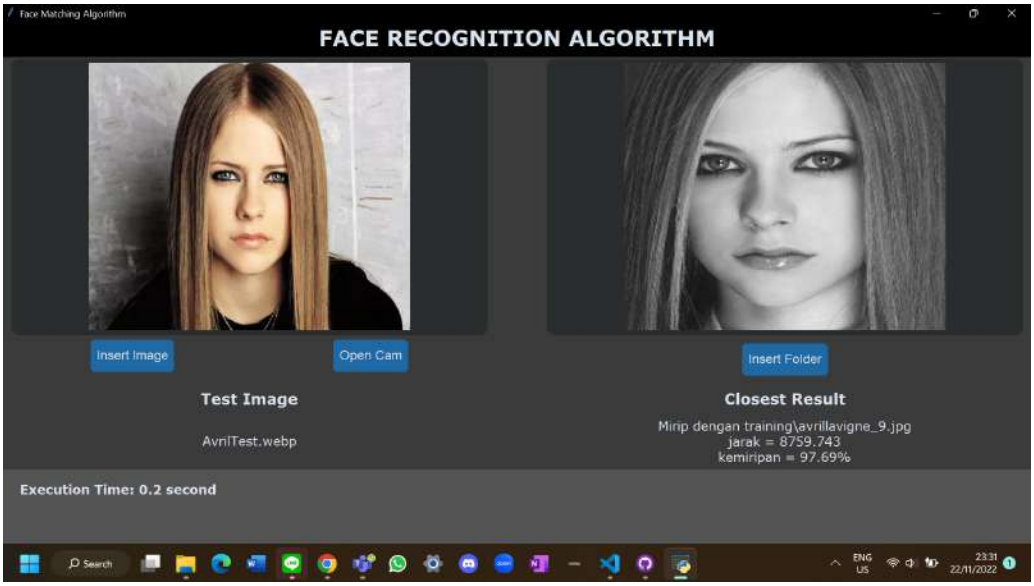
4.1. Hasil Eksperimen 1

a. Folder Dataset yang digunakan

Dataset yang digunakan untuk eksperimen terletak pada folder *test/training* pada github.



b. Hasil tes



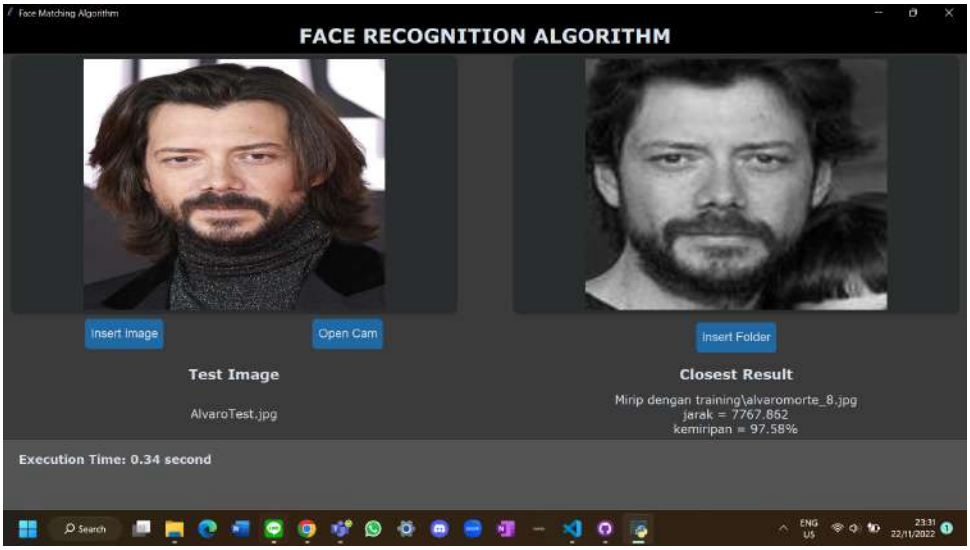
Nama Test Gambar	example/AvrilTest.webp
Hasil Eigenface	training/avrillavigne_9.jpg
Euclidean Distance	8759.743
Waktu Eksekusi	0.2 second
Similarity	97.66%

4.2. Hasil Eksperimen 2

a. Folder dataset yang digunakan

Dataset yang digunakan untuk eksperimen kedua terletak pada folder test/training pada github.

b. Hasil tes



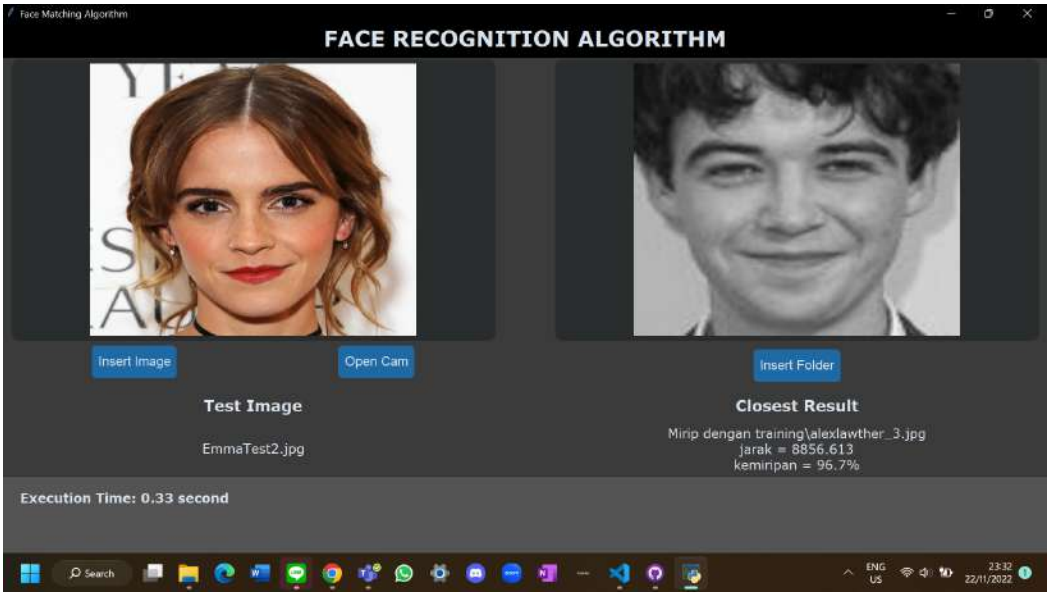
Nama Test Gambar	example/AlvaroTest.jpg
Hasil Eigenface	training/alvaromorte_8.jpg
Euclidean Distance	7767.852
Waktu Eksekusi	0.34 second
Similarity	97.58%

4.3. Hasil Eksperimen 3

a. Folder dataset yang digunakan

Dataset yang digunakan untuk eksperimen terletak pada folder test/training pada github.

b. Hasil tes



Nama Test Gambar	example/EmmaTest2.jpg
Hasil Eigenface	training/alexlawther_3.jpg
Euclidean Distance	8856.613
Waktu Eksekusi	0.33 second
Similarity	96.7%

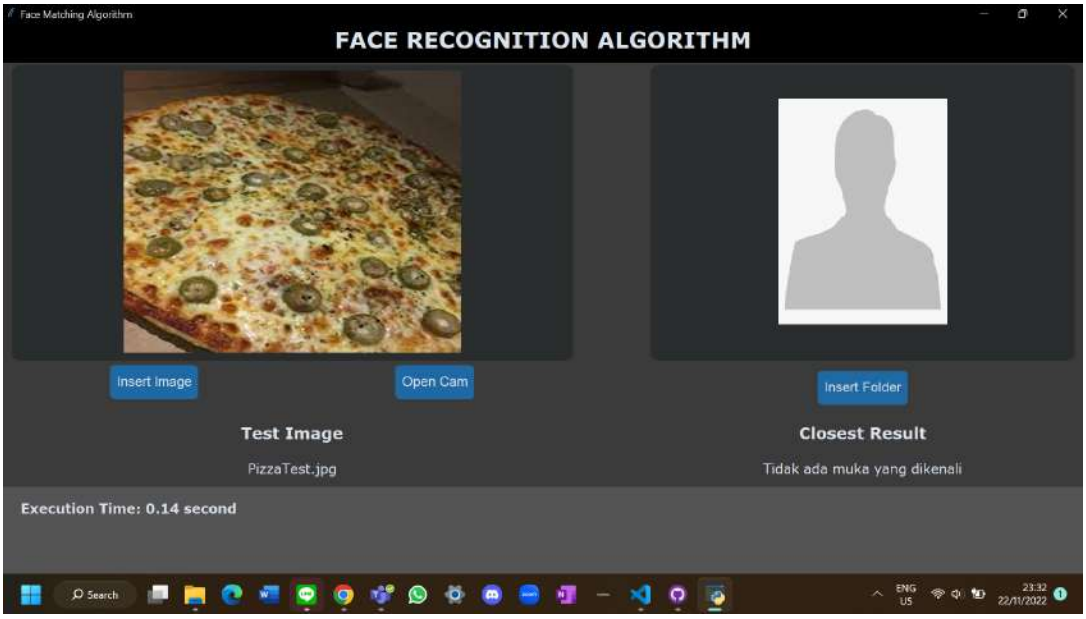
Hasil tersebut ditunjukkan berdasarkan kemiripan ekspresi wajah

4.4. Hasil Eksperimen 4

a. Folder Dataset yang digunakan

Dataset yang digunakan untuk eksperimen terletak pada folder test/training pada github.

b. Hasil tes



Nama Test Gambar	example/PizzaTest.jpg
Hasil Eigenface	-
Euclidean Distance	-
Waktu Eksekusi	0.14 second
Similarity	-

4.5. Hasil Eksperimen Bonus

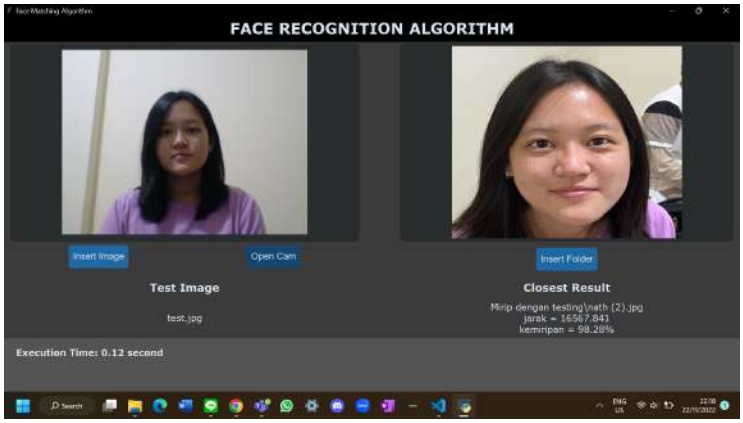
Untuk dataset bonus, kami menggunakan muka teman-teman kami sebagai training image. Kami sudah ijin kepada pihak terkait untuk menggunakan foto mereka sebagai dataset tugas ini.

a. Folder Dataset yang digunakan

b. Hasil tes menggunakan kamera

Nama Test Gambar	test.jpg dari kamera
Hasil Eigenface	jason(2).jpg
Euclidean Distance	15067.383
Waktu Eksekusi	0.13 second
Similarity	97.85%

Nama Test Gambar	test.jpg dari kamera
Hasil Eigenface	cath(3).jpg
Euclidean Distance	12780.83
Waktu Eksekusi	0.13 second
Similarity	96.22%



Nama Test Gambar	test.jpg dari kamera
Hasil Eigenface	nath(2).jpg
Euclidean Distance	16567.841

Waktu Eksekusi	0.12 second
Similarity	98.28%

Nama Test Gambar	test.jpg dari kamera
Hasil Eigenface	juan(3).jpg
Euclidean Distance	16539.054
Waktu Eksekusi	0.12 second
Similarity	88.91%

Nama Test Gambar	test.jpg dari kamera
Hasil Eigenface	hans(2).jpg
Euclidean Distance	13065.498
Waktu Eksekusi	0.12 second
Similarity	96.61%

BAB 5

PENUTUP

5.1 Kesimpulan

Dari materi kuliah Aljabar Linier dan Geometri IF2123 yang kami pelajari di kelas, kami mengimplementasikannya ke dalam sebuah program bahasa python. Program yang kami buat dapat mencari perbandingan gambar menggunakan algoritma eigenface dan menampilkan gambar dengan hasil termirip.

Melalui tugas besar ini, kami mengetahui algoritma dari eigenface dan face recognition. Kami juga jadi mengetahui algoritma agar pemrosesan gambar lebih efisien. Kami juga mengerti dan memahami cara pemakaian tkinter sebagai design GUI program. Selain itu, kami juga memahami algoritma dari opencv untuk mengambil gambar melalui kamera dan mencari hasil kecocokan menggunakan eigenface

5.2 Saran

- Debugging program seharusnya dilakukan lebih awal
- Pengerjaan dapat dikerjakan lebih awal sehingga tidak mengganggu waktu belajar kuis

5.3 Refleksi

Melalui tugas besar ini mendapatkan berbagai macam pengalaman. Kami dapat mengasah kemampuan bekerjasama, berkomunikasi, mendekomposisikan berbagai macam permasalahan yang ada, serta melatih kedisplinan dalam mengerjakan tugas masing-masing. Menurut kami, pembagian tugas kami pun juga sangat rata dan bobotnya sama. Kami juga mengetahui lebih dalam tentang bahasa pemrograman python beserta library baru dan platform Tkinter.

DAFTAR REFERENSI

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

http://repository.uin-suska.ac.id/18226/7/7.%20BAB%20II.pdf

https://jagostat.com/aljabar-linear/nilai-eigen-dan-vektor-eigen

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

https://www.youtube.com/watch?v=yyOXDSlY8d4

https://www.geeksforgeeks.org/cropping-faces-from-images-using-opencv-python/

https://math.stackexchange.com/questions/3913710/intuitive-explanation-of-why-the-modified-gram-schmidt-is-more-stable-than-the-c

https://www.geeksforgeeks.org/cropping-faces-from-images-using-opencv-python/

https://github.com/TomSchimansky/CustomTkinter/wiki

LAMPIRAN

- **Link Repository**

  Link repository kelompok kami untuk tugas besar 2 mata kuliah IF2123 Aljabar Linier dan Geometri adalah sebagai berikut.

  Link : https://github.com/AJason36/Algeo02-21100

- **Link Youtube Bonus**

  Link video youtube kelompok kami untuk tugas besar 2 mata kuliah IF2123 Aljabar Linier dan Geometri adalah sebagai berikut:

  Link: https://youtu.be/jki3-1ObPsw