

# **Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri**

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

**Semester I Tahun 2022/2023**



### **Kelompok RAN:**

Angela Livia Arumsari	13521094
Rinaldy Adin	13521134
Nathania Callista	13521139

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2022**

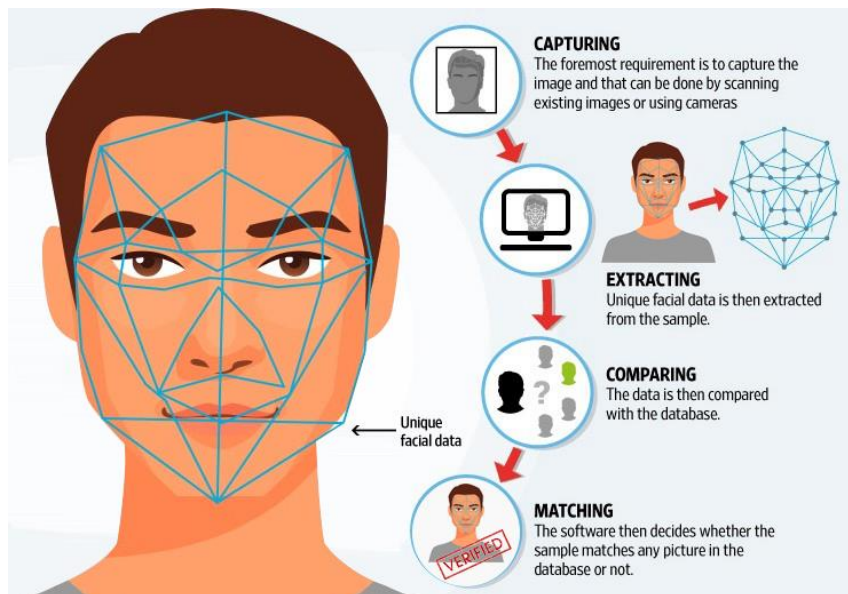
## DAFTAR ISI

BAB I DESKRIPSI MASALAH .....	3
BAB II DASAR TEORI .....	5
2.1    Perkalian Matriks .....	5
2.2    Nilai Eigen.....	5
2.3    Vektor Eigen .....	6
2.4    Eigenface .....	6
2.5    Dekomposisi QR .....	7
2.6    Pengenalan Wajah .....	8
BAB III IMPLEMENTASI PROGRAM.....	10
BAB IV EKSPERIMEN .....	14
4.1    Kasus I.....	14
4.2    Kasus II .....	16
4.3    Kasus III .....	19
4.4    Kasus IV .....	23
BAB V KESIMPULAN, SARAN, DAN REFLEKSI.....	27
5.1    Kesimpulan.....	27
5.2    Saran.....	27
5.3    Refleksi.....	27
DAFTAR REFERENSI .....	28
LAMPIRAN.....	29

## 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 dalam *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 pemrosesan sebuah sistem pengenalan wajah diperlihatkan pada Gambar 1.



Gambar 1 Alur pemrosesan sistem pengenalan wajah

Terdapat berbagai teknik untuk memeriksa citra wajah dari kumpulan citra yang sudah diketahui, seperti jarak Euclidean (*Euclidean Distance*) dan *cosine similarity*, *principal component analysis*, serta *Eigenface*. Pada tugas ini, akan dibuat sebuah program pengenalan wajah yang 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 dinormalisasi dari RGB ke Grayscale (matriks). Hasil normalisasi akan digunakan untuk perhitungan *eigenface*. Seperti namanya, matriks *eigenface* menggunakan *eigenvector* dalam pembentukannya.

Pada tahapan akhir, akan ditemukan sebuah 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

### DASAR TEORI

#### 2.1 Perkalian Matriks

Perkalian matriks adalah perkalian yang melibatkan suatu matriks atau susunan bilangan berupa kolom dan angka, serta memiliki sifat – sifat tertentu. Maatriks bisa dikalikan dengan bilangan bulat, maupun matriks lainnya. Perkalian di dalam matriks memiliki syarat masing – masing.

Hasil perkalian matriks dengan bilangan skalar dapat adalah matriks baru, dimana elemen – elemen dari matriks baru ini adalah hasil perkalian bilangan skalar dengan elemen – elemen pada matriks lama. Contohnya adalah sebagai berikut :

$$k \times \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} ka & kb \\ kc & kd \end{pmatrix}$$

Perkalian matriks dengan matriks lain hanya bisa dilakukan ketika jumlah kolom matriks A sama dengan jumlah baris matriks B, seperti contoh di bawah ini.

$$\begin{aligned} P.Q &= \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\ &= \begin{bmatrix} ax & + & bx \\ cy & + & dy \end{bmatrix} \end{aligned}$$

#### 2.2 Nilai Eigen

Kata eigen berasal dari Bahasa Jerman yang artinya asli atau karakteristik. Misalnya A adalah sebuah matriks  $n \times n$ , maka vektor tidak nol  $x$  di  $\mathbb{R}^n$  disebut vector eigen dari A, jika  $Ax$  sama dengan perkalian suatu scalar  $\lambda$  dengan  $x$ , yaitu

$$Ax = \lambda x$$

Skalar  $\lambda$  disebut nilai Eigen dari A, dan x dinamakan vector eigen yang berkoresponden dengan  $\lambda$ . Nilai eigen dari suatu matriks, dapat dihitung dengan langkah – langkah berikut.

$$Ax = \lambda x$$

$$IAx = \lambda IX$$

$$Ax = \lambda Ix$$

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

$x = 0$  adalah solusi trivial dari persamaan  $(\lambda I - A)x = 0$ . Agar persamaan di atas memiliki solusi tidak nol, maka haruslah

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

Persamaan di atas disebut juga dengan persamaan karakteristik dari matriks A dan akar – akar persamaan dari persamaan itu, yaitu  $\lambda$ , dinamakan akar – akar karakteristik atau nilai – nilai eigen.

### 2.3 Vektor Eigen

Berdasarkan persamaan  $Ax = \lambda x$ , vector eigen  $x$  menyatakan matriks kolom yang apabila dikalikan dengan sebuah matriks  $n \times n$  akan menghasilkan vector lain yang merupakan kelipatan vector itu sendiri.

Dengan kata lain, operasi  $Ax = \lambda x$  menyebabkan vector  $x$  menyusut atau memanjang dengan factor  $\lambda$ , dengan arah yang sama jika  $\lambda$  positif dan arah berkebalikan jika  $\lambda$  negative.

### 2.4 Eigenface

Eigenface merupakan sekumpulan eigen vektor yang mempresentasikan ciri citra wajah. Eigenface pertama kali dikembangkan di MIT dan menggunakan *Principle Component Analysis (PCA)*. Salah satu aplikasi dari Eigenface adalah dalam algoritma pengenalan wajah. Secara umum, langkah pada algoritma Eigenface adalah sebagai berikut:

1. Membuat himpunan citra wajah untuk pelatihan
2. Menghitung matriks rata – rata dan kurangkan nilai tiap wajah dengan rata – rata nilai wajah
3. Menghitung matriks kovarian
4. Menghitung Eigenvecotr dan Eigenvalue dari matriks kovarian. Pada tugas kali ini, digunakan QR Method untuk mendapatkan Eigenvalue dan Eigenvector

5. Pilih *principal componenet*, ambil  $k$  *eigenvector* dengan nilai *eigenvalue* tertinggi dari  $M$  citra.
6. Mengubah citra ke komponen *Eigenface*

## 2.5 Dekomposisi QR

Dekomposisi QR atau faktorisasi QR adalah sebuah metode yang dapat digunakan untuk memecah sebuah matriks (misalnya matriks  $A$ ) menjadi sebuah matriks orthogonal dan sebuah matriks segitiga.

$$A = QR$$

Pada persamaan di atas,  $Q$  adalah sebuah matriks orthogonal dan  $R$  adalah sebuah matriks segitiga. Dalam dekomposisi QR methods, ada beberapa metode yang bisa dilakukan, yaitu Housholder methods dan Gram Schmidt methods. Pada tugas kali ini, dekomposisi QR menggunakan Householder methods dalam memecah matriks.

Householder methods adalah metode pemecahan suatu matriks dengan memanfaatkan matriks transformasi Householder. Langkah – langkah mendapatkan matriks  $Q$  dan  $R$  dengan householders adalah sebagai berikut:

1. Tentukan vector kolom dari matriks  $A$  yang dinamai sebagai vector  $\mathbf{b}$
2. Tentukan vector  $\mathbf{u}$  sehingga  $\mathbf{u} = \mathbf{b} + \text{sign}(b_1)||\mathbf{b}||\mathbf{e}_1$ , dengan  $\mathbf{e}_1$  adalah basis standar
3. Bentuk matriks transformasi Householder  $\mathbf{H}_1$  dengan mensubstitusikan vector  $\mathbf{u}$  pada langkah kedua, dengan persamaan sebagai berikut

$$\mathbf{H}_1 = \mathbf{I} - 2 \frac{\mathbf{u} \mathbf{u}^T}{\mathbf{u}^T \mathbf{u}}$$

dengan  $\mathbf{I}$  adalah matriks identitas.

4. Kalikan matriks transformasi Householder  $\mathbf{H}_1$  dengan matriks  $\mathbf{A}$ . Hasil perkalian tersebut akan membuat semua elemen pada kolom pertama, di bawah diagonal utama  $a'_{11}$  bernilai 0.

$$\mathbf{H}_1 \mathbf{A} = \begin{bmatrix} a'_{11} & a'_{12} & \cdots & a'_{1n} \\ 0 & & & \\ \vdots & & \mathbf{A}' & \\ 0 & & & \end{bmatrix}$$

$\mathbf{A}'$  adalah hasil perkalian dari matriks  $\mathbf{H}_1$  dengan matriks  $\mathbf{A}$

5. Ulangi langkah pertama sampai dengan ketiga dengan menggunakan matriks  $\mathbf{A}'$  pada langkah keempat yang dinamai sebagai  $\mathbf{H}'_2$  untuk menentukan matriks  $\mathbf{H}_2$ .

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \mathbf{H}'_2 & \\ 0 & & & \end{bmatrix}$$

6. Matriks transformasi Householder  $\mathbf{H}_2$  dikalikan dengan matriks  $\mathbf{H}_1\mathbf{A}$ , maka hasil perkalian tersebut akan membuat semua nilai pada kolom kedua, di bawah diagonal utama, bernilai 0

$$\mathbf{H}_2\mathbf{H}_1\mathbf{A} = \begin{bmatrix} a'_{11} & a'_{12} & a'_{13} & \cdots & a'_{1n} \\ 0 & a''_{22} & a''_{23} & \cdots & a''_{2n} \\ 0 & 0 & & & \\ \vdots & \vdots & & \mathbf{A}'' & \\ 0 & 0 & & & \end{bmatrix}$$

7. Tentukan matriks transformasi  $\mathbf{H}_3$  dan seterusnya sampai  $\mathbf{H}_k$ , hingga seluruh entri di bawah diagonal utama bernilai nol.

$$\mathbf{H}_k = \begin{bmatrix} I_{k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}'_k \end{bmatrix}$$

## 2.6 Pengenalan Wajah

Pengenalan wajah adalah suatu cara untuk mengidentifikasi atau mengkonfirmasi identitas dari seseorang menggunakan wajah mereka. Pengenalan wajah mengidentifikasi suatu wajah dengan membandingkan wajah baru tersebut dengan wajah-wajah yang sudah di-*train* dan disimpan terlebih dahulu. Untuk melakukan pengenalan wajah dengan Eigenface, gambar wajah yang baru ( $\Gamma$ ) yang sudah dinormalisasi terhadap *mean face*/muka rata-rata( $\Psi$ ) diproyeksi ke dalam “face space” dengan persamaan,

$$\omega_k = u_k^T(\Gamma - \Psi)$$

Untuk  $k = 1, \dots, M$  dengan  $M$  adalah jumlah eigenface

*Weights* proyeksi tersebut dibuat menjadi sebuah vektor  $\Omega^T = [\omega_1, \omega_2, \dots, \omega_M]$  yang menyimpan “kontribusi” atau koefisien dari setiap eigenface dalam representasi gambar baru tersebut dalam “face space”. Vektor koefisien ( $\Omega$ ) dari proyeksi wajah baru tersebut dibandingkan dengan vektor koefisien masing-masing wajah *training* ( $\Omega_k$ ) dengan menghitung euclidean distance antara dua vektor tersebut.

$$\epsilon_k^2 = \|(\Omega - \Omega_k)\|^2$$

Euclidean distance  $\epsilon_k$  yang paling kecil menunjukkan bahwa wajah ke  $k$  merupakan wajah yang paling mirip dengan wajah masukan. Ketika euclidean distance terkecil dari sebuah wajah masukan lebih dari threshold  $\theta_\epsilon$  yang sudah ditentukan, wajah tersebut tidak dapat



diklasifikasikan sebagai wajah dari orang yang terdapat di dalam kumpulan training image sehingga dapat diartikan bahwa wajah baru tersebut merupakan wajah dari orang yang tidak dikenali.

Karena proyeksi dari suatu wajah baru menggunakan basis gambar yang sudah dipastikan berupa wajah, dapat dievaluasi apakah gambar tersebut merupakan wajah dengan membandingkan gambar wajah yang sudah dinormalisasi terhadap *mean face*  $\Phi = \Gamma - \Psi$  dengan hasil rekonstruksi wajah tersebut dari “face space”  $\Phi_f = \sum_{i=0}^M \omega_i u_i$ . Perbandingan tersebut kembali menggunakan euclidean distance

$$\epsilon^2 = \|\Phi - \Phi_f\|^2$$

Dari jarak tersebut, dapat dievaluasi apakah gambar masukan merupakan wajah dengan mengevaluasi apakah distance tersebut lebih dari threshold  $\theta_\epsilon$  sehingga dapat diartikan bahwa gambar masukan tersebut bukan gambar wajah.

### BAB III

### IMPLEMENTASI PROGRAM

Pada pembuatan aplikasi face recognition, kami menggunakan beberapa library untuk mendukung algoritma face recognition dan pembuatan GUI aplikasi. Untuk keperluan algoritma face recognition, digunakan library numpy untuk melakukan operasi matriks dan cv2 untuk mengekstrak gambar. Untuk keperluan GUI, digunakan library tkinter dalam membuat tampilan dan PIL untuk menampilkan gambar. Selain itu, digunakan library bawaan os untuk membaca directory file.

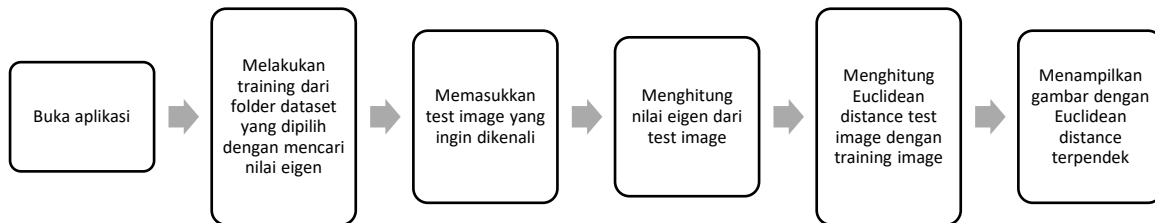
Dalam membuat algoritma face recognition, berikut fungsi-fungsi yang kami implementasikan.

Nama	Hasil	Deskripsi
normalizeImage(path)	Matrix $65536 \times 1$	Membaca <i>image</i> pada <i>path</i> kemudian menormalisasi warna dan ukuran gambar, lalu melakukan <i>reshape matrix</i> hasil <i>extract</i> menjadi $65536 \times 1$
extractMatrices(directory)	Array of Matrix $65536 \times 1$	Melakukan normalisasi warna dan ukuran pada seluruh gambar di <i>directory</i>
meanOfMatrices (trainingMatrices)	Matrix $65536 \times 1$	Menghitung rata-rata dari seluruh <i>matrix</i> hasil <i>training image</i>
differenceOfMatrix (trainingMatrix, mean)	Matrix $65536 \times 1$	Menghitung selisih <i>matrix</i> dari <i>training image</i> dengan rata-rata
differenceList (trainingMatrices, mean)	Array of Matrix $65536 \times 1$	Menghitung selisih <i>matrix</i> dari <i>training image</i> dengan rata-rata dari seluruh gambar yang digunakan untuk <i>training</i>
concatMatrix(result)	Matrix $65536 \times n$ (n adalah banyaknya foto dalam folder)	Menggabungkan semua matrix yang ada di dalam folder dataset, menjadi sebuah matriks berukuran $65536 \times n$ (sesuai dengan banyaknya data)

matrixCovariant (matrixConcat)	Matrikx $n \times n$ (n adalah banyaknya foto dalam folder)	Mengalikan <i>transpose</i> dari <i>matrix</i> yang sudah diconcat dengan <i>matrix</i> yang sudah diconcat (tidak di-transpose)
getEigenVector (matrixCovariant)	Matrix $n \times n$ (n adalah banyaknya foto dalam folder)	Mendapatkan eigen vector dari matrix covariant, dengan menggunakan metode QR decomposition
calculateEigenfaces (A_matrix, eigenVectors)	List berisi Martrix $n \times 1$ (n adalah banyaknya foto dalam folder)	Mendapatkan eigen face dari matrix covariant dan eigen vektor yang sudah orhonormal.
trainFromFolder(path)	Tuple berisi List berisi $n \times 1$ , Matrix $65536 \times 1$ , dan List berisi Matrix $65536 \times 1$	Menjalankan semua fungsi untuk menghasilkan Eigenface, serta mengembalikan List Eigenface, muka rata-rata, dan List Matrix muka
calculateWeighths (eigenfaces, normalizedVector)	Matrix $n \times 1$ (n adalah banyaknya foto dalam folder)	Menghitung nilai <i>weight</i> /koefisien dari muka baru dalam proyeksi eigenspace.
calculateFaceSpaceProj (eigenfaces, weights)	Matrix $65536 \times 1$	Mengembalikan hasil proyeksi/rekonstruksi kembali muka berdasarkan <i>weights</i> /koefisien pada proyeksi eigenspace.
calculateEuclideanDist (A,B)	float	Menghitung euclidean distance antara Matrix/Vektor A dan B
getSimilarImagesPathSorted (unkownImageVector, mean, trainingImages, trainingImagesPaths, eigenfaces, similarityThreshold,)	Tuple berisi List of String, float, float, dan float	Mencari muka paling <i>similar</i> dengan gambar masukan dengan membandingkan euclidean distance muka baru dengan muka training. Fungsi mengembalikan List berisi path gambar, euclidean distance rekonstruksi dengan gambar awal, nilai

		<i>similarity</i> , dan nilai <i>euclidean distance</i> terdekat.
--	--	---

Aplikasi *face recognition* yang kami buat didasarkan pada prinsip Eigenface. Secara umum, alur dari program yang telah dibuat yaitu sebagai berikut.



Algoritma utama yang dimanfaatkan pada proses face recognition yaitu algoritma eigenface. Langkah-langkah perhitungan Eigenface yang diterapkan pada program adalah sebagai berikut.

1. Membuat array of matrix dari citra wajah untuk pelatihan dengan fungsi `extractMatrices`. Langkah ini melakukan ekstraksi matriks dari setiap gambar pelatihan. Tiap gambar pelatihan tersebut akan dinormalisasi warnanya menjadi hitam putih dengan range intensitas yang sama. Ukuran seluruh gambar akan menjadi  $256 \times 256$ . Sedangkan, hasil ekstraksi matriks akan dimampatkan menjadi matriks berukuran  $65536 \times 1$ .
2. Menghitung matriks rata – rata dengan fungsi `meanOfMatrices`. Langkah ini menghitung nilai rata-rata dari seluruh matriks hasil ekstraksi gambar pelatihan.
3. Mengurangkan nilai tiap wajah dengan rata – rata nilai wajah dengan fungsi `differenceList`. Langkah ini menghitung selisih matriks hasil gambar pelatihan dengan matriks rata-rata yang telah dihitung pada langkah sebelumnya. Seluruh matriks kemudian disimpan di *array of matrix*.
4. Menggabungkan semua matrix foto dalam *array of matrix*, menjadi sebuah matrix besar. Langkah ini dilakukan dengan menggunakan fungsi `concat` bawaan *numpy*.
5. Mendapatkan *matrix covariant* dengan cara mengalikan *transpose* dari *matrix concat* dengan *matrix concat*. Langkah ini menghasilkan sebuah matrix yang ukurannya tergantung dengan banyaknya file yang ada.

6. Mendapatkan nilai *eigenvector* dan *eigenvalue* dengan menggunakan metode dekomposisi QR. Dari metode dekomposisi QR, didapatkan matrix Q dan matrix R. Kemudian, lakukan *looping* agar mendapatkan hasil *eigenvector* dan *eigenvalue* yang lebih akurat.
7. Menghitung vektor *eigenface* yang orthonormal dengan mengalikan matrix yang berisi semua vektor wajah dengan *eigenvector* yang berkoresponden dengan *eigenface* yang dicari. Setelah itu *eigenface* tersebut dibagi dengan panjang/*vector norm* vektor itu sendiri agar menghasilkan vektor *eigenface* yang orthonormal

Tahapan pengenalan wajah adalah sebagai berikut.

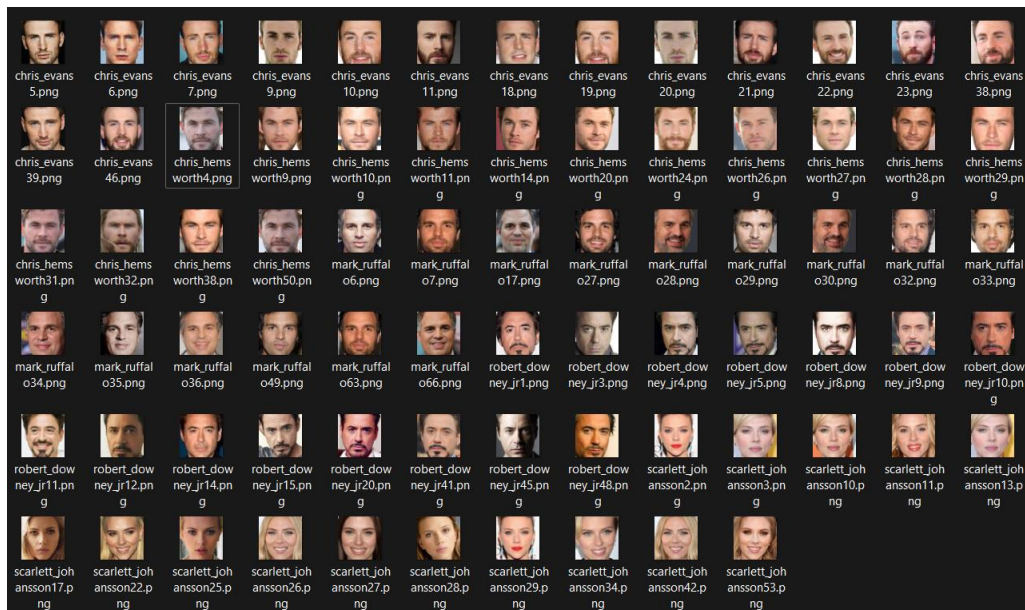
1. Menghitung *weights*/koefisien proyeksi dari wajah masukan menggunakan fungsi `calculateWeights`.
2. Menghitung hasil rekonstruksi wajah berdasarkan *weights* yang didapatkan.
3. Untuk setiap *eigenface*, hitung *weights* dari *eigenface* tersebut lalu cari *euclidean distance* antara *eigenface* dan wajah baru. Nilai *euclidean distance* tersebut disimpan dalam array or tuple yang menyimpan *euclidean distance* serta *filepath* wajah yang berkoresponden dengan *eigenface* tersebut.
4. Sortir list yang berisi *euclidean distance* secara menaik berdasarkan *euclidean distance* yang dihitung. Hitung *similarity* dari wajah paling *similar*.
5. Hitung *euclidean distance* dari vektor wajah baru dengan rekonstruksi wajah tersebut yang sudah dihitung pada langkah 2.
6. Bandingkan *euclidean distance* pada langkah 5 dengan *threshold*, jika lebih dari *threshold*, maka gambar tersebut bukan gambar dari wajah seseorang.
7. Bandingkan *euclidean distance* dari wajah paling similar dengan *threshold*, jika lebih dari *threshold*, maka wajah tersebut bukan wajah dari orang yang terdapat di dalam kumpulan *training image*.
8. Jika gambar masukan dapat diklasifikasikan sebagai wajah dari orang yang terdapat dalam training image, tampilkan wajah tersebut dan *similarity* dan *euclidean distance* wajah tersebut, beserta wajah 3 gambar terdekat lainnya.

## BAB IV EKSPERIMEN

Akan dilakukan eksperimen untuk empat kasus dengan empat dataset yang berbeda. Untuk setiap kasus, akan dilakukan analisis hasil pengenalan wajah yang dilakukan.

### 4.1 Kasus I

Pada kasus pertama, digunakan dataset dengan karakteristik sebagai berikut.



Gambar 4.1.1 Dataset yang digunakan


<b>Nama</b>	Dataset 01
<b>Sumber</b>	Kaggle
<b>Jumlah identitas</b>	5
<b>Jumlah gambar per identitas</b>	15
<b>Ukuran gambar</b>	Bervariasi dari $100 \times 100$ hingga $1400 \times 1400$
<b>Karakteristik gambar</b>	Seluruh gambar telah di-crop sehingga hanya menampilkan wajah, tetapi beberapa foto masih belum di-align dan memiliki ekspresi tersenyum

Hasil pengenalan wajah




### Recognize Face using Eigenface Method

Test Image



Closest Result



Insert Your Dataset

Dataset 01


Insert Your Test Image

chris\_evans29.png  
or

Result

Similarity: 62.76%  
Euclidean Distance: 42.83  
Execution time: 00:21

Other Closest Results

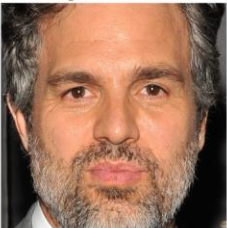


Gambar 4.1.2 Hasil pengenalan wajah 01




### Recognize Face using Eigenface Method

Test Image



Closest Result



Insert Your Dataset

Dataset 01


Insert Your Test Image

mark\_ruffalo60.png  
or

Result

Similarity: 74.80%  
Euclidean Distance: 28.98  
Execution time: 00:21

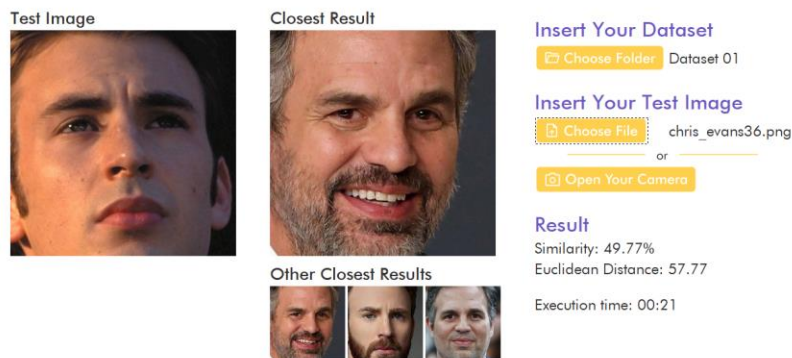
Other Closest Results



Gambar 4.1.3 Hasil pengenalan wajah 02



### Recognize Face using Eigenface Method



Gambar 4.1.4 Hasil pengenalan wajah 03

Pada percobaan kali ini, digunakan jumlah foto sebanyak 75 yang membutuhkan waktu 21 detik untuk menyelesaikan training image. Dari hasil pengenalan wajah yang dilakukan, ada beberapa percobaan dengan hasil yang kurang akurat. Pada pengenalan wajah yang berhasil, test image yang digunakan memiliki wajah yang sepenuhnya menghadap ke kamera. Sedangkan pada pengenalan wajah yang gagal dapat disebabkan oleh posisi wajah yang tidak sepenuhnya menghadap ke kamera, sehingga perhitungan *eigenface* menjadi bias dan menghasilkan hasil yang tidak tepat. Selain itu, ukuran gambar yang sangat bervariasi dapat menyebabkan berkurangnya keakuratan. Hal tersebut dikarenakan adanya proses normalisasi untuk membuat seluruh ukuran gambar menjadi  $256 \times 256$ . Pada pengenalan wajah yang salah, similarity akan berkurang sehingga hasil memang kurang akurat.

## 4.2 Kasus II

Pada kasus kedua, digunakan dataset dengan karakteristik sebagai berikut.





Gambar 4.2.1 Dataset yang digunakan

<b>Nama</b>	Dataset 02
<b>Sumber</b>	Kaggle
<b>Jumlah identitas</b>	10
<b>Jumlah gambar per identitas</b>	5
<b>Ukuran gambar</b>	160 × 160
<b>Karakteristik gambar</b>	Seluruh gambar telah di-crop sehingga hanya menampilkan wajah, tetapi beberapa foto masih belum di-align dan memiliki ekspresi tersenyum

Hasil pengenalan wajah



### Recognize Face using Eigenface Method

Test Image

Closest Result

Insert Your Dataset
  
 Dataset 02

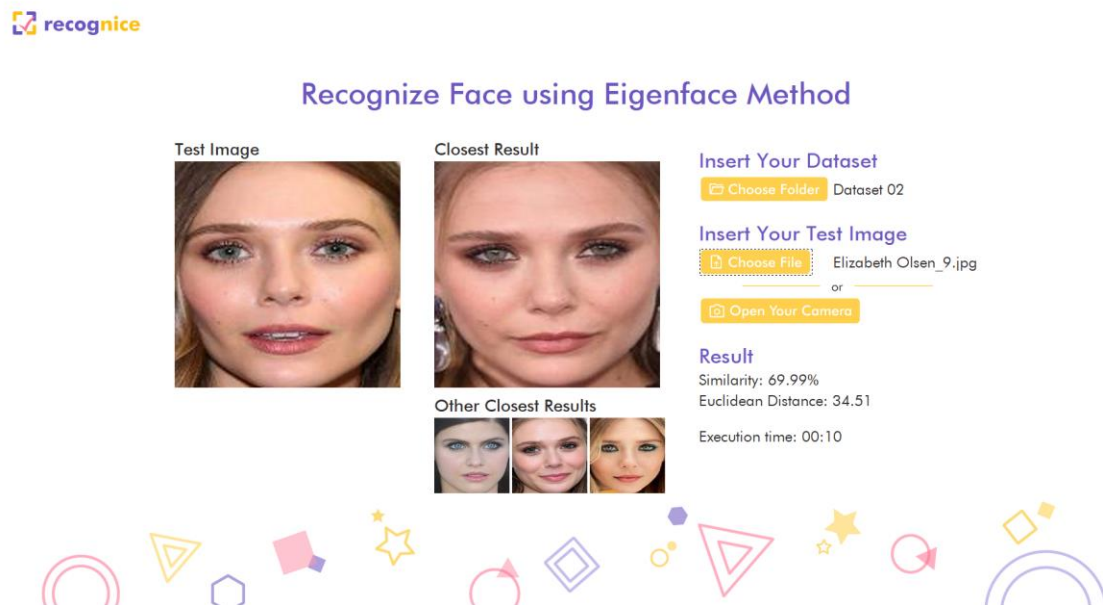
Insert Your Test Image
  
 Alexandra Daddario\_28.jpg
  
or

Result
  
Similarity: 66.26%
  
Euclidean Distance: 38.81
  
Execution time: 00:10

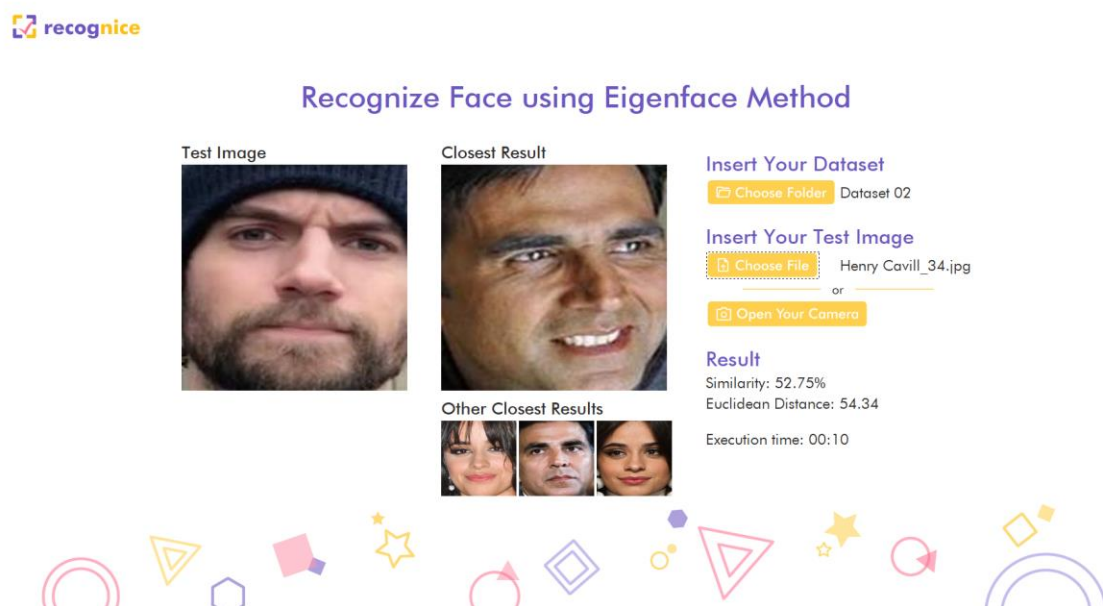
Other Closest Results



Gambar 4.2.2 Hasil pengenalan wajah 01



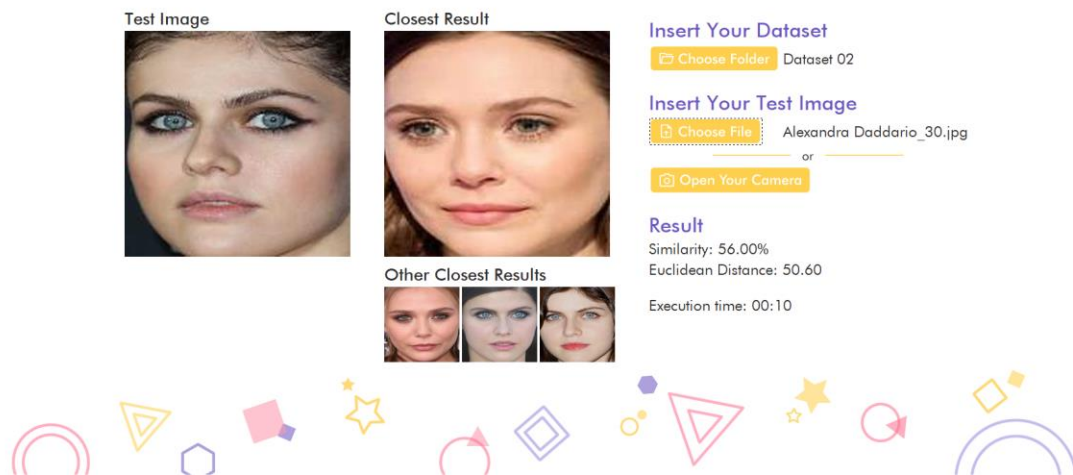
Gambar 4.2.3 Hasil pengenalan wajah 02



Gambar 4.2.4 Hasil pengenalan wajah 03



### Recognize Face using Eigenface Method



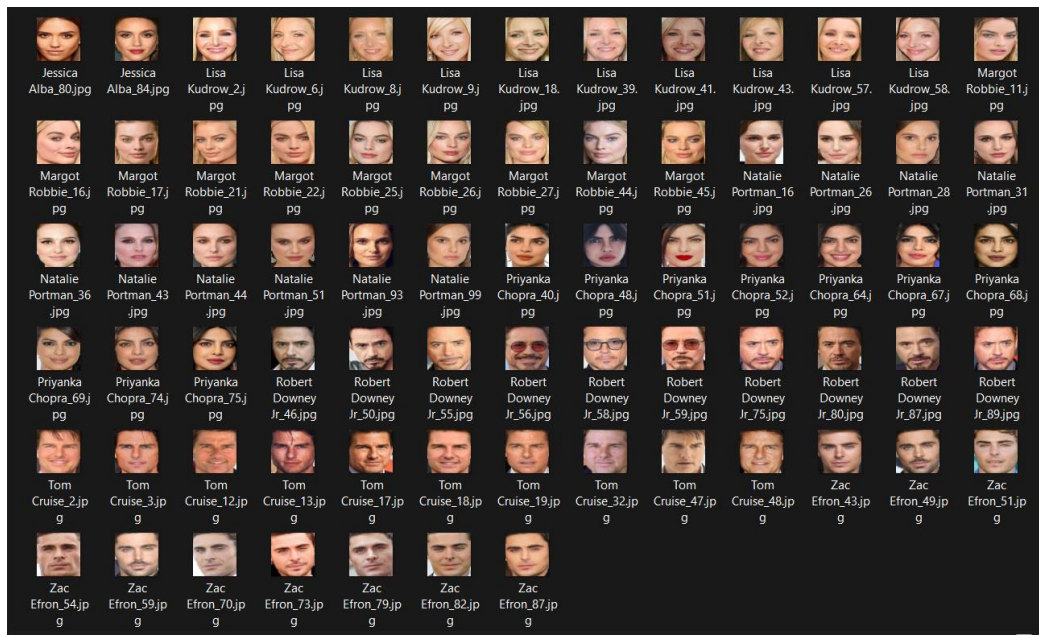
Gambar 4.2.5 Hasil pengenalan wajah 04

Pada percobaan kali ini, digunakan jumlah foto sebanyak 50 yang membutuhkan waktu 10 detik untuk menyelesaikan training image. Dari hasil pengenalan wajah yang dilakukan, ada beberapa percobaan dengan hasil yang kurang akurat. Pada pengenalan wajah yang berhasil, test image yang digunakan memiliki wajah yang sepenuhnya menghadap ke kamera. Sedangkan pada pengenalan wajah yang gagal dapat disebabkan oleh posisi wajah yang tidak sepenuhnya menghadap ke kamera, sehingga perhitungan *eigenface* menjadi bias dan menghasilkan hasil yang tidak tepat. Selain itu, jumlah training image per identitas yang hanya sedikit juga turut memengaruhi keakuratan hasil pengenalan wajah. Pada pengenalan wajah yang salah, similarity akan berkurang sehingga hasil memang kurang akurat.

### 4.3 Kasus III

Pada kasus ketiga, digunakan dataset dengan karakteristik sebagai berikut.

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Nilai Eigen dan EigenFace pada Pengenalan Wajah (Face Recognition)



Gambar 4.3.1 Dataset yang digunakan

<b>Nama</b>	Dataset 03
<b>Sumber</b>	Kaggle
<b>Jumlah identitas</b>	15
<b>Jumlah gambar per identitas</b>	10
<b>Ukuran gambar</b>	160 × 160
<b>Karakteristik gambar</b>	Seluruh gambar telah di-crop sehingga hanya menampilkan wajah, tetapi beberapa foto masih belum di-align, wajah memiliki ekspresi beragam, dan ada yang menggunakan aksesoris


Hasil pengenalan wajah






### Recognize Face using Eigenface Method

Test Image



Closest Result



Insert Your Dataset

Choose Folder Dataset 03

Insert Your Test Image

Choose File Andy Samberg\_57.jpg

or

Open Your Camera


Result

Similarity: 58.91%

Euclidean Distance: 44.58

Execution time: 01:06

Other Closest Results




Gambar 4.3.2 Hasil pengenalan wajah 01




### Recognize Face using Eigenface Method

Test Image



Closest Result



Insert Your Dataset

Choose Folder Dataset 03

Insert Your Test Image

Choose File Brad Pitt\_92.jpg

or

Open Your Camera


Result

Similarity: 46.73%

Euclidean Distance: 50.78

Execution time: 01:06

Other Closest Results



Gambar 4.3.3 Hasil pengenalan wajah 02



### Recognize Face using Eigenface Method

The screenshot displays the 'recognice' web application interface. On the left, under 'Test Image', is a portrait of Jessica Alba. To its right, under 'Closest Result', is a matching portrait of Jessica Alba. Below that, 'Other Closest Results' shows three smaller portraits of different people. On the right side, the 'Insert Your Dataset' section has a 'Choose Folder' button and 'Dataset 03' listed. The 'Insert Your Test Image' section has a 'Choose File' button with 'Jessica Alba\_96.jpg' selected, and an 'Open Your Camera' button. The 'Result' section shows 'Similarity: 49.19%' and 'Euclidean Distance: 59.46'. At the bottom, it says 'Execution time: 01:06'.

Gambar 4.3.4 Hasil pengenalan wajah 03



### Recognize Face using Eigenface Method

The screenshot displays the 'recognice' web application interface. On the left, under 'Test Image', is a portrait of Brad Pitt wearing sunglasses. To its right, under 'Closest Result', is a matching portrait of Brad Pitt wearing glasses. Below that, 'Other Closest Results' shows three smaller portraits of different people. On the right side, the 'Insert Your Dataset' section has a 'Choose Folder' button and 'Dataset 03' listed. The 'Insert Your Test Image' section has a 'Choose File' button with 'Brad Pitt\_21.jpg' selected, and an 'Open Your Camera' button. The 'Result' section shows 'Similarity: 32.06%' and 'Euclidean Distance: 72.42'. At the bottom, it says 'Execution time: 01:06'.

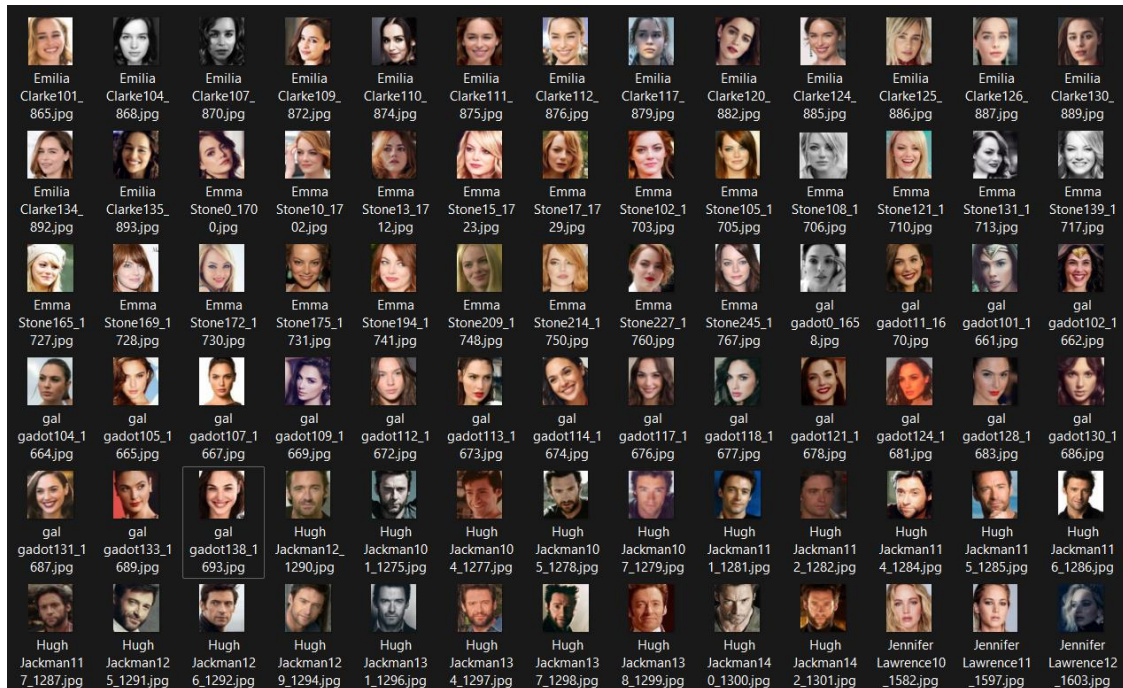
Gambar 4.3.5 Hasil pengenalan wajah 04

Pada percobaan kali ini, digunakan jumlah foto sebanyak 150 yang membutuhkan waktu 1 menit 6 detik untuk menyelesaikan training image. Dari hasil pengenalan wajah yang dilakukan, ada beberapa percobaan dengan hasil yang kurang akurat. Pada pengenalan wajah yang berhasil, test image yang digunakan memiliki wajah yang sepenuhnya menghadap ke kamera. Sedangkan pada pengenalan wajah yang gagal dapat disebabkan oleh posisi wajah

yang tidak sepenuhnya menghadap ke kamera, sehingga perhitungan *eigenface* menjadi bias dan menghasilkan hasil yang tidak tepat. Selain itu, pada wajah yang menggunakan aksesoris, keakuratan pengenalan wajah juga berkurang.

#### 4.4 Kasus IV

Pada kasus keempat, digunakan dataset dengan karakteristik sebagai berikut.



Gambar 4.4.1 Dataset yang digunakan


<b>Nama</b>	Dataset 04
<b>Sumber</b>	Kaggle
<b>Jumlah identitas</b>	10
<b>Jumlah gambar per identitas</b>	20
<b>Ukuran gambar</b>	Tidak selalu persegi dengan ukuran bervariasi dari 200 hingga 500 pixel
<b>Karakteristik gambar</b>	Seluruh gambar telah di-crop sehingga hanya menampilkan wajah, tetapi beberapa foto masih belum di-align, wajah memiliki ekspresi beragam, dan ada yang menggunakan aksesoris

Hasil pengenalan wajah




### Recognize Face using Eigenface Method

Test Image



Closest Result



Insert Your Dataset

Dataset 04

Insert Your Test Image

Adriana Lima26\_149.jpg

or


Result

Similarity: 75.80%

Euclidean Distance: 73.15

Execution time: 02:20

Other Closest Results




Gambar 4.4.2 Hasil pengenalan wajah 01




### Recognize Face using Eigenface Method

Test Image



Closest Result



Insert Your Dataset

Dataset 04

Insert Your Test Image

Emilia Clarke31\_1007.jpg

or


Result

Similarity: 56.71%

Euclidean Distance: 98.69

Execution time: 02:20

Other Closest Results




Gambar 4.4.3 Hasil pengenalan wajah 02






### Recognize Face using Eigenface Method

Test Image



Closest Result



Insert Your Dataset

Dataset 04

Insert Your Test Image

gal gadot217\_1765.jpg

or


Result

Similarity: 67.60%

Euclidean Distance: 92.25

Execution time: 02:20

Other Closest Results




Gambar 4.4.4 Hasil pengenalan wajah 03

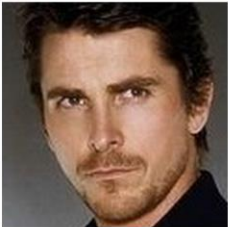


### Recognize Face using Eigenface Method

Test Image



Closest Result



Insert Your Dataset

Dataset 04

Insert Your Test Image

Christian Bale238\_635.jpg

or


Result

Similarity: 67.63%

Euclidean Distance: 84.49

Execution time: 02:20

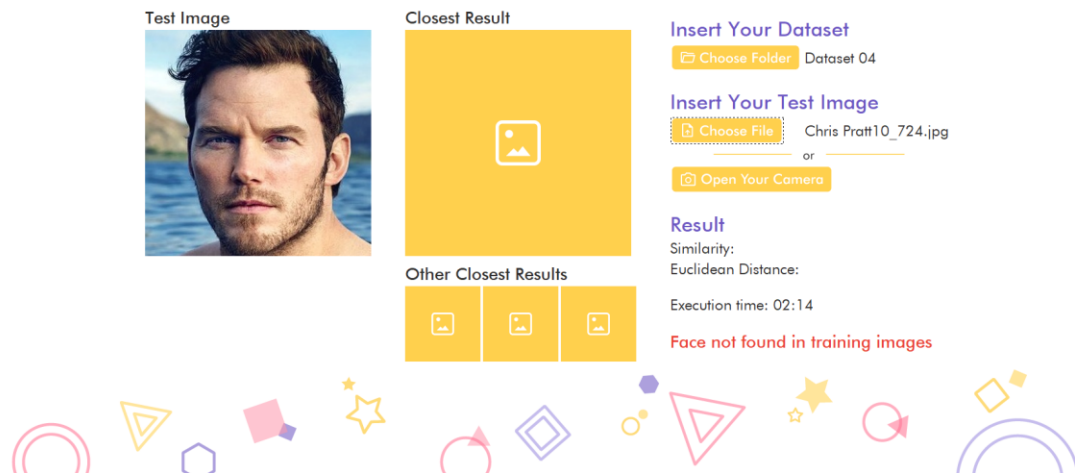
Other Closest Results



Gambar 4.4.5 Hasil pengenalan wajah 04



### Recognize Face using Eigenface Method



Gambar 4.4.6 Hasil pengenalan wajah 05

Pada percobaan kali ini, digunakan jumlah foto sebanyak 200 yang membutuhkan waktu 2 menit 20 detik untuk menyelesaikan training image. Dengan penambahan gambar tiap identitas, dapat diperoleh hasil yang lebih akurat. Meskipun ada beberapa gambar dengan aksesoris maupun wajah yang tidak sepenuhnya menghadap kamera, pengenalan tetap berhasil dilakukan. Hal ini dapat disebabkan peningkatan keakuratan akibat bertambahnya jumlah gambar pada dataset. Selain itu, jika dimasukkan gambar dari wajah yang tidak berada di dataset, aplikasi dapat memberikan pesan error bahwa wajah tersebut tidak ditemukan.

## **BAB V**

### **KESIMPULAN, SARAN, DAN REFLEKSI**

#### **5.1 Kesimpulan**

Pada Tugas Besar 2 ini, kami berhasil membuat aplikasi *recognice*, yaitu sebuah aplikasi untuk *face recognition* berdasarkan Eigenface. Aplikasi ini menerima folder dataset dari pengguna untuk dijadikan training image. Untuk *test image*, pengguna dapat mengunggah *file image* atau menggunakan kamera. Aplikasi *recognice* akan menampilkan hasil gambar yang paling mirip dengan *test image*. Untuk hasil yang optimal, sebaiknya digunakan gambar dengan wajah yang sudah di-crop dan sepenuhnya menghadap ke kamera. Wajah juga diusahakan tidak sedang tersenyum atau menggunakan aksesoris. Selain itu, hasil akan lebih akurat jika jumlah gambar per identitas lebih banyak dan memiliki ukuran yang seragam.

#### **5.2 Saran**

Dalam pengerjaan Tugas Besar 2 ini, diperlukan banyak algoritma yang kurang berkaitan dengan materi pembelajaran yang diberikan di kelas. Hal ini mempersulit proses implementasi program karena perlunya banyak eksplorasi mandiri. Selain itu, terdapat beberapa kesalahan pada spesifikasi dan referensi yang diberikan. Ada baiknya referensi algoritma maupun penjelasan algoritma pada spesifikasi dipersiapkan dengan lebih matang untuk memudahkan implementasi algoritma.

#### **5.3 Refleksi**

Dari seluruh proses pengerjaan Tugas Besar 2 ini, kami belajar pentingnya melakukan eksplorasi mandiri pada algoritma yang kurang kami pahami. Kami juga belajar bereksperimen agar program kami lebih efektif dan optimal. Selain itu, karena tugas ini dikerjakan secara berkelompok, kami juga belajar bagaimana untuk berkomunikasi dan membagi tugas. Secara keseluruhan, kami telah mengerjakan Tugas Besar ini dengan baik. Kami mampu menyelesaikan seluruh fitur dan bonus yang diperlukan.

## DAFTAR REFERENSI

Avengers Faces Dataset. (2022, April 8). Kaggle.

<https://www.kaggle.com/datasets/yasserh/avengers-faces-dataset>

Burak. (2020, March 7). Pins face recognition. Kaggle. Retrieved November 5, 2022, from

<https://www.kaggle.com/datasets/hereisburak/pins-face-recognition>

Face Recognition Dataset. (2020, November 6). Kaggle.

<https://www.kaggle.com/datasets/vasukipatel/face-recognition-dataset>

*Householder Transformation*. (n.d.).

<https://www.youtube.com/watch?v=pOiOH3yESPM&t=1847s>

*Nilai Eigen dan Vektor Eigen*. (n.d.).

<https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-Nilai-Eigen-dan-Vektor-Eigen-Bagian1.pdf>

*QR Algorithm*. (n.d.). [https://madrury.github.io/jekyll/update/statistics/2017/10/04/qr-](https://madrury.github.io/jekyll/update/statistics/2017/10/04/qr-algorithm.html)

[algorithm.html](https://madrury.github.io/jekyll/update/statistics/2017/10/04/qr-algorithm.html)

*QR Decomposition with Householder Reflections*. n.d.). ([https://rpubs.com/aaronsc32/qr-](https://rpubs.com/aaronsc32/qr-decomposition-householder)

[decomposition-householder](https://rpubs.com/aaronsc32/qr-decomposition-householder)

Turk, M., & Pentland, A. (1991). Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1), 71–86. <https://doi.org/10.1162/jocn.1991.3.1.71>

## LAMPIRAN

Repository github dapat diakses pada link berikut:

Repository: <https://github.com/liviaarumsari/Algeo02-21094.git>

Video demo dapat diakses pada link berikut:

Demo Tubes 2 Algeo: <https://youtu.be/JrZH4j8tW3A>