

# **TUGAS BESAR I ALJABAR LINEAR 2025**



**KELOMPOK**

**LUTFIYAH ISTIANA (L0124022)  
SHAFARIF RIFKIKANIR FAUZIAH (L0124031)**

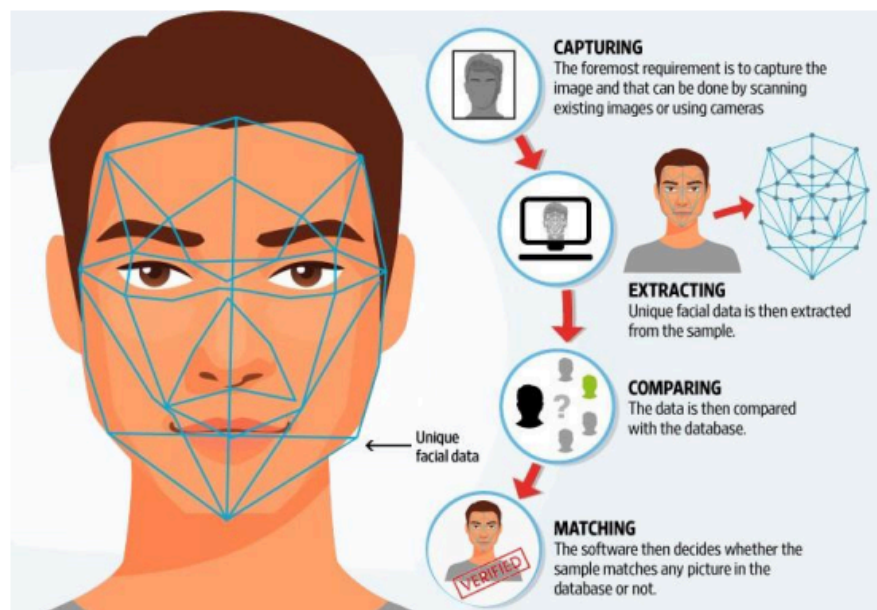
**PROGRAM STUDI INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA  
UNIVERSITAS SEBELAS MARET  
2024/2025**

# BAB 1

## DESKRIPSI MASALAH

### 1.1. Abstrak

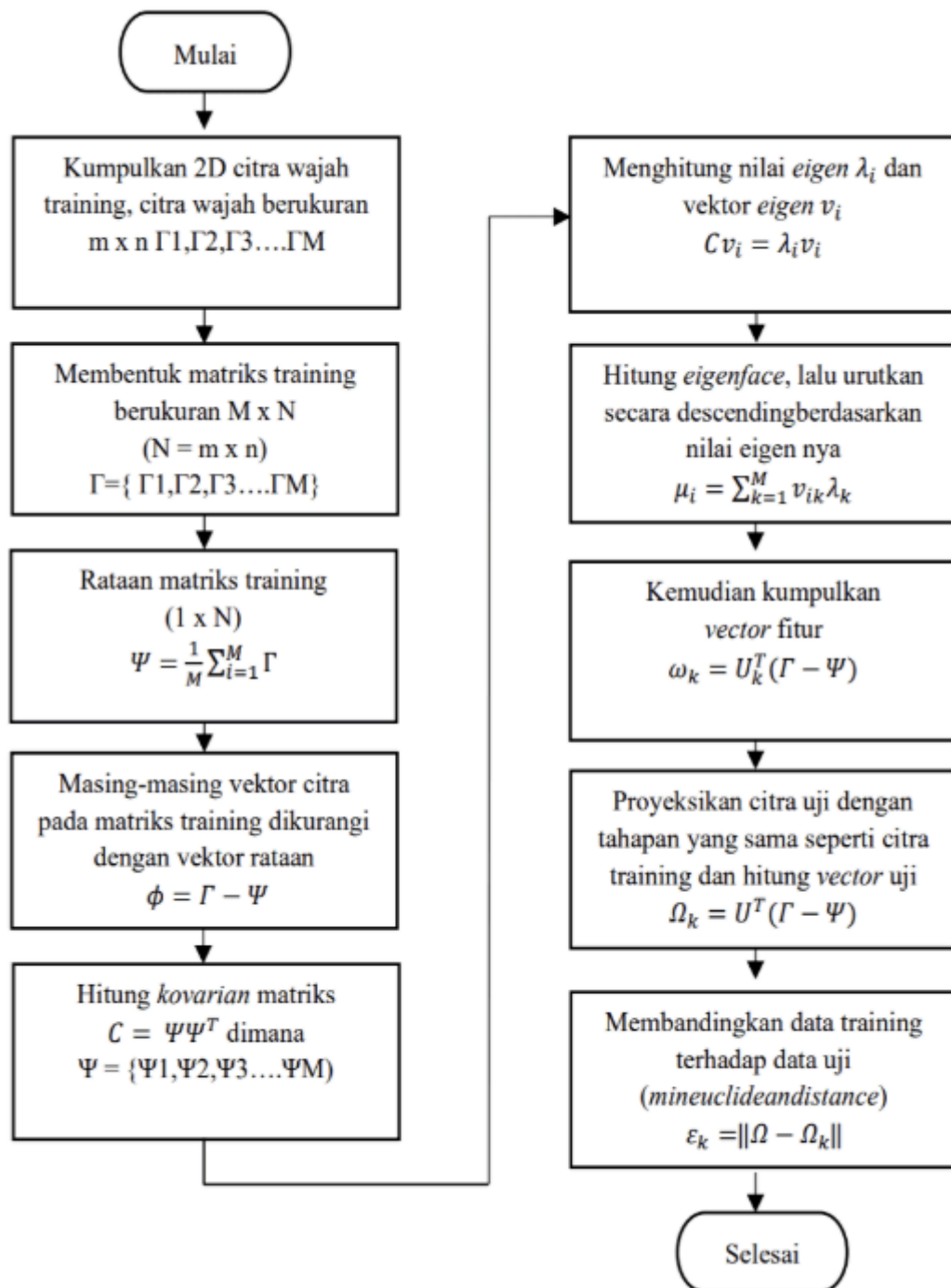
Pengenalan wajah (Face Recognition) adalah teknologi biometrik yang bisa dipakai untuk mengidentifikasi wajah seseorang untuk berbagai kepentingan khususnya keamanan. Program pengenalan wajah melibatkan kumpulan citra wajah yang sudah disimpan pada database lalu berdasarkan kumpulan citra wajah tersebut, program dapat mempelajari dengan citra yang akan diidentifikasi. Alur proses sebuah sistem pengenalan wajah diperlihatkan pada Gambar 1.



Terdapat berbagai teknik untuk memeriksa citra wajah dari kumpulan citra yang sudah diketahui seperti jarak Euclidean dan *cosine similarity*, principal component analysis (PCA), serta Eigenface. Pada Tugas ini, akan dibentuk 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.

## Flowchart Algoritma Eigenface



Pada tahapan akhir, akan ditemui gambar dengan euclidean distance 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.

Program dibuat dengan Bahasa Python dengan memanfaatkan sejumlah library di OpenCV (Computer Vision) atau library pemrosesan gambar lainnya (contoh PIL).

Fungsi untuk mengekstraksi fitur dari sebuah citra wajah tidak perlu anda buat lagi, tetapi menggunakan fungsi ekstraksi yang sudah tersedia di dalam library. Fungsi Eigen dilarang import dari library dan harus diimplementasikan, sedangkan untuk operasi matriks lainnya silahkan menggunakan library.

Kode program untuk ekstraksi fitur dapat dibaca pada artikel ini: Feature extraction and similar image search with OpenCV for newbies, pada laman:

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

Nilai batas kemiripan citra test face dapat ditentukan oleh pembuat program melalui percobaan. Berikut merupakan referensi pengenalan wajah dengan metode eigenface:

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

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

## PENGUNAAN PROGRAM

Berikut ini adalah input yang akan dimasukkan pengguna untuk eksekusi pr

1. **Folder dataset**, berisi folder atau directory yang berisi kumpulan gambar yang digunakan sebagai training image.
2. **File gambar**, berisi file gambar input yang ingin dikenali dengan format file yang bebas selama merupakan format untuk gambar.

Tampilan layout dari aplikasi web yang akan dibangun kurang lebih adalah sebagai berikut. Anda dapat mengubah layout selama layout masih terdiri dari komponen yang sama.



Catatan: Warna biru menunjukkan komponen yang dapat di klik. Warna hijau menunjukkan luaran yang didapat dari hasil eksekusi. Anda dapat menambahkan menu lainnya, gambar, logo, dan sebagainya. Tampilan GUI dibuat semenarik mungkin selama mencakup seluruh informasi pada layout yang diberikan di atas. Kreativitas menjadi salah satu komponen penilaian.

## 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/hereisburak/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 *library* atau Bahasa Python.

## BAB II

### TEORI SINGKAT

#### 2.1. Perkalian Matriks

Jika  $A$  adalah matriks  $m \times r$  dan  $B$  adalah 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  $\lambda$ . Skalar  $\lambda$  ini disebut nilai eigen matriks  $A$  dan vektor  $x$  disebut vektor eigen matriks  $A$ , maka pengalihan faktor tersebut dengan matriks  $A$  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 ( $\lambda$ ) dari matriks A, dapat ditemukan vektor eigen dengan menyulihkan  $\lambda$  pada sistem persamaan linier berikut.

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

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 linear 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  $x_1 = \frac{1}{2}t$ ,  $x_2 = t$ ,  $t \in \mathbb{R}$ , sehingga vektor berikut

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

merupakan vektor eigen yang berkorespondensi dengan nilai eigen  $\lambda = 3$ . Untuk  $\lambda = -1$ , diperoleh sistem persamaan linear 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  $x_1 = 0$ ,  $x_2 = t$ ,  $t \in \mathbb{R}$ , sehingga vektor berikut.

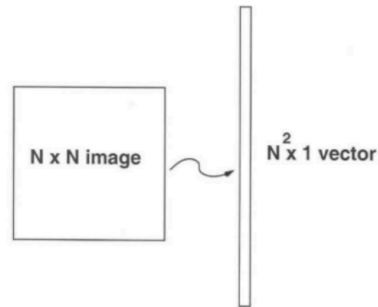
$$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

merupakan vektor eigen yang berkorespondensi 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 kaskas OpenCV. Pada tugas ini, ukuran gambar minimal  $256 \times 256$ .
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 ( $\Psi$ ) 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  $A^T$  dengan matriks  $A$ , yang menghasilkan matriks berukuran  $M \times M$ , kemudian kita dapat mencari nilai eigen dan vektor eigen matriks tersebut.

$$Cov = AA^T$$

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

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

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



$$C'u_i = \lambda_i u_i$$

dimana

$$C' = AA^T$$

$$U_i = Av_i$$

7. Dari persamaan di atas, dapat disimpulkan nilai eigen matriks  $C'$  ( $AA^T$ ) 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  $K$  eigen dari matriks  $C'$  yang berkoresponden dengan  $K$  nilai eigen terbesar ( $K < M$ ). Vektor eigen ini memiliki ukuran  $N^2 \times 1$ .
9. Ambil vector-vektor  $a_i$  yang telah dihitung sebelumnya dan nyatakan vektor tersebut sebagai kombinasi linear 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 ( $\Psi$ )

$$\varnothing = y - \Psi$$

3. Nyatakan vektor  $\varnothing$  sebagai kombinasi linier dari vektor-vektor  $u$ .

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

4. Dari persamaan di atas, dapat dinyatakan vektor  $\Omega$  yang berisi koefisien-koefisien vektor  $u$ .

$$\Omega = [ W_1 \ W_2 : W_j ]$$

5. Ambil vektor  $\Omega$  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. Modul yang digunakan**

##### **3.1.1. Tkinter**

Dalam program, Tkinter berguna sebagai bentuk standar dari program Python yang digunakan untuk membuat antarmuka pengguna grafis (GUI). Tkinter menyediakan elemen seperti tombol (button), label, entry, canvas, dan window utama yang memungkinkan pengguna berinteraksi dengan program secara visual tanpa menggunakan terminal.

##### **3.1.2. PIL (Pillow)**

Pil atau Pillow adalah pustaka untuk memproses gambar, khususnya mengubah format gambar agar dapat ditampilkan di Thinker. Dalam program ini, PIL digunakan untuk mengkonversi citra dari format OpenCv (BGR) ke format yang dikenali Thinker (RGB), serta menampilkan gambar hasil pengenalan.

##### **3.1.3. OpenCV (cv2)**

OpenCv berfungsi untuk pemrosesan citra digital video. Digunakan untuk membaca gambar dari file, menangkap gambar dari *webcam*, mengubah citra ke *grayscale*, melakukan *resize*, serta menangani format warna dari BGR ke RGB.

##### **3.1.4. NumPy**

NumPy digunakan untuk komputasi numerik dan operasi matriks. Dalam konteks ini, NumPy digunakan untuk menangani representasi gambar sebagai *array*, menghitung *eigenfaces*, jarak *Euclidean*, serta melakukan dekomposisi QR untuk mendapatkan nilai dan vektor eigen (*eigenvalues* dan *eigenvectors*).

##### **3.1.5. OS**

OS berfungsi untuk interaksi dengan sistem file, seperti membuka direktori dataset, membaca nama-nama file gambar, membuat *folder cache*, dan mengecek keberadaan file.

##### **3.1.6. Threading**

Digunakan untuk menjalankan proses secara paralel. Hal ini penting agar proses berat seperti pelatihan model dan pengenalan wajah tidak membuat GUI menjadi tidak responsif atau macet.

##### **3.1.7. Time**

Time digunakan untuk mengukur durasi proses pengenalan wajah dan menampilkan waktu eksekusi kepada pengguna. Juga digunakan untuk membuat delay saat countdown sebelum pengambilan gambar dari webcam.

##### **3.1.8. Math**

Math digunakan untuk operasi dasar matematika, seperti pembulatan nilai countdown saat menghitung waktu mundur sebelum pengambilan gambar mode webcam.

### 3.1.9. Sys

Sys digunakan untuk menghentikan program secara aman saat pengguna menutup jendela aplikasi, memastikan semua proses berhenti dengan benar.

### 3.1.10. Base64

Digunakan untuk mengubah nama path dataset menjadi bentuk encoded string, yang kemudian dipakai sebagai nama file change. Ini membantu menciptakan file cache yang unik untuk setiap dataset.

## 3.2. Algoritma program

### 1. *Preprocessing* Gambar

- a. *Resize* ukuran gambar menjadi 256×256
- b. Konversi gambar ke grayscale dan flatten vektor 1D

### 2. Pelatihan Model

- a. Memuat dataset dan melakukan *preprocessing* gambar
- b. Melakukan normalisasi data
- c. Menghitung matriks kovarian
- d. Melakukan dekomposisi QR dengan *Householder Reflection* untuk mendapatkan nilai Eigen dan vektor Eigen
- e. Mengambil k nilai Eigen terbesar
- f. Melakukan perhitungan *Eigenface*
- g. Melakukan perhitungan *weight training*
- h. Menyimpan *weight training* ke dalam file .npz agar tidak diperlukan pelatihan ulang saat menggunakan dataset yang sama di masa depan

### 3. Proses Pengenalan

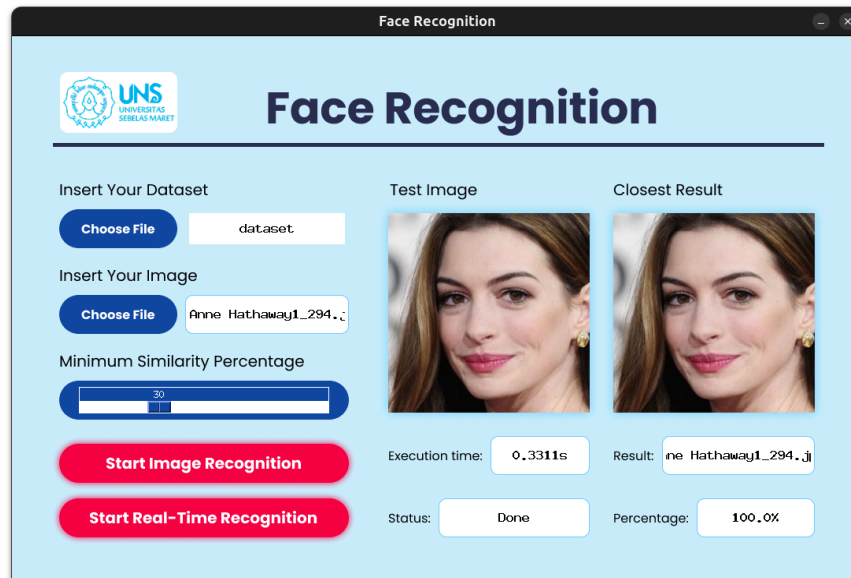
- a. Melakukan *preprocessing* gambar yang akan diuji
- b. Melakukan perhitungan *weight test*
- c. Melakukan perhitungan jarak *euclidean* dan persentase kemiripan
- d. Menentukan hasil pengenalan wajah berdasarkan jarak *Euclidean* terkecil dan persentase kemiripan di atas batas minimum yang telah ditentukan

## BAB IV

### EKSPERIMEN/EKSEKUSI PROGRAM

#### 4.1. Jika data uji ada dalam folder dataset

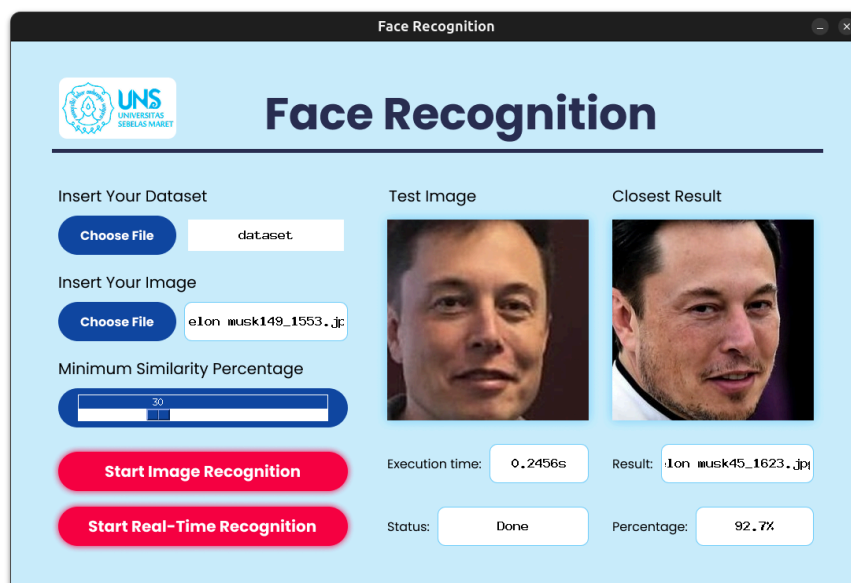
Jika data uji ada dalam folder dataset, maka persentase kemiripan dari hasil pengenalan wajah yang dilakukan adalah 100%.



The screenshot shows the 'Face Recognition' web application interface. The title 'Face Recognition' is prominently displayed. The interface is divided into three main sections: 'Insert Your Dataset', 'Test Image', and 'Closest Result'. In the 'Insert Your Dataset' section, a 'Choose File' button is next to a text box containing 'dataset'. In the 'Insert Your Image' section, a 'Choose File' button is next to a text box containing 'Anne Hathaway1\_294...'. Below these, there is a 'Minimum Similarity Percentage' slider set to 30. Two red buttons are visible: 'Start Image Recognition' and 'Start Real-Time Recognition'. The 'Test Image' section displays a portrait of Anne Hathaway. The 'Closest Result' section shows the same portrait, with 'Execution time: 0.3311s', 'Result: ne Hathaway1\_294...', 'Status: Done', and 'Percentage: 100.0%'.

#### 4.2. Jika data uji namun foto dari subjek masih menjadi bagian dari folder dataset

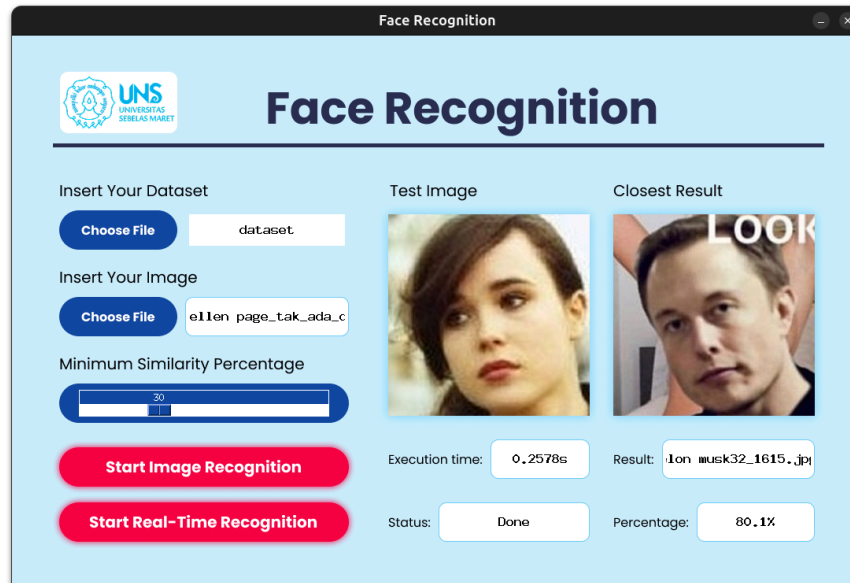
Jika data uji tidak ada dalam folder dataset, namun foto lain dari subjek tersebut masih berada dalam dataset, maka persentase kemiripan dari hasil pengenalan wajah yang dilakukan adalah mendekati 100%, namun tidak mencapai 100%.



The screenshot shows the 'Face Recognition' web application interface. The title 'Face Recognition' is prominently displayed. The interface is divided into three main sections: 'Insert Your Dataset', 'Test Image', and 'Closest Result'. In the 'Insert Your Dataset' section, a 'Choose File' button is next to a text box containing 'dataset'. In the 'Insert Your Image' section, a 'Choose File' button is next to a text box containing 'elon musk149\_1553.jpg'. Below these, there is a 'Minimum Similarity Percentage' slider set to 30. Two red buttons are visible: 'Start Image Recognition' and 'Start Real-Time Recognition'. The 'Test Image' section displays a portrait of Elon Musk. The 'Closest Result' section shows the same portrait, with 'Execution time: 0.2456s', 'Result: lon musk45\_1623.jpg', 'Status: Done', and 'Percentage: 92.7%'.

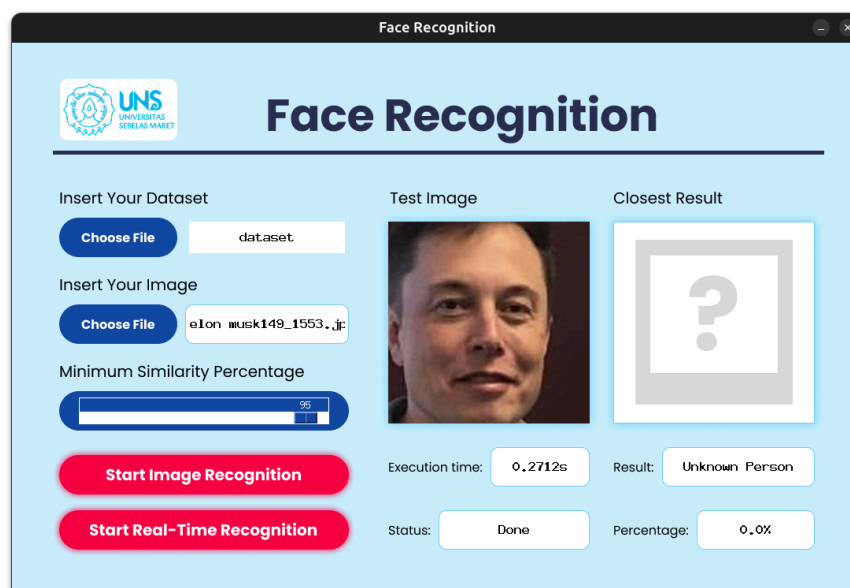
#### 4.3. Jika data foto dari subjek tidak ada dalam dataset

Jika data uji dan foto lain dari subjek tersebut tidak berada dalam dataset, maka persentase kemiripan dari hasil pengenalan wajah yang dilakukan akan lebih rendah. Selain itu, data hasil pengenalan tidak cocok dengan subjek dalam foto.



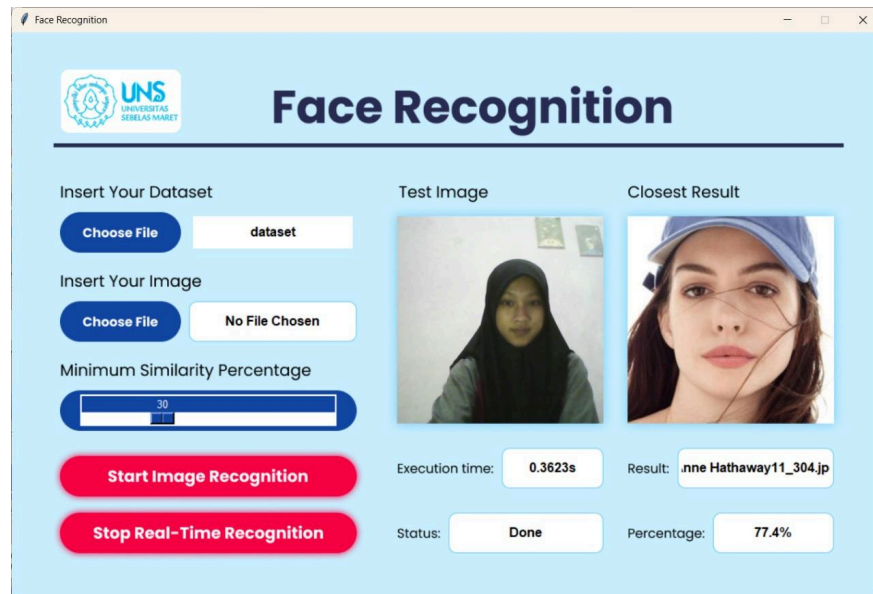
#### 4.3. Jika tidak ada hasil proses pengenalan wajah yang memiliki persentase kemiripan di atas batas minimum yang telah ditentukan

Jika tidak ada hasil proses pengenalan wajah yang memiliki persentase kemiripan di atas batas minimum yang telah ditentukan, maka tidak akan ada hasil pengenalan wajah dan persentase kemiripan otomatis 0%. Selain itu, gambar hasil yang ditampilkan akan berupa gambar tanda tanya.



#### 4.4. Pengenalan wajah secara *real-time* menggunakan kamera

Berikut ini adalah dokumentasi dari demonstrasi program pengenalan wajah secara *real-time* menggunakan kamera.



## BAB V

### KESIMPULAN

#### 5.1. Kesimpulan

Program *face recognition* berbasis *eigenfaces* ini telah berhasil diimplementasikan dengan memanfaatkan dekomposisi nilai dan vektor eigen menggunakan algoritma *Eigenface* dan *eigenvector*. Sistem mampu melakukan *training* dataset gambar wajah dan mengenali wajah baru dengan membandingkan jarak *Euclidean* bobot *eigenface*. Namun, aplikasi ini memiliki beberapa keterbatasan, terutama dalam hal kecepatan komputasi untuk dataset yang besar dan akurasi yang masih bergantung pada kualitas gambar yang di inputkan. Proses *training* yang lambat akibat kompleksitas algoritma *eigenface* dan *eigenvector* serta ketidakadaan *preprocessing* lanjutan deteksi fitur wajah menjadi faktor penghambat utama. Meskipun demikian, antarmuka berbasis Tkinter yang dibangun sudah interaktif dengan adanya fitur pemilihan data set, pemilihan gambar, *real time*, *executing time*, *result*, dan *percentage*.

#### 5.2. Saran

Untuk meningkatkan performa dan akurasi program, beberapa perbaikan dapat dilakukan.

- Pertama, optimasi algoritma *eigenface* dan *eigenvector* dengan metode seperti *Scheduled Relaxation eigenface* dan *eigenvector* atau menggantinya dengan algoritma yang lebih efisien seperti *Singular Value Decomposition (SVD)*.
- Kedua, tambahkan *preprocessing* gambar yang lebih *robust*, termasuk deteksi wajah (*face alignment*), normalisasi pencahayaan, dan penghapusan latar belakang untuk mengurangi noise.
- Ketiga, implementasikan mekanisme pembaruan cache otomatis ketika dataset diubah agar tidak perlu *training* ulang secara manual. Terakhir, pengembangan antarmuka yang lebih informatif dengan visualisasi *eigenfaces* dan grafik kemiripan bisa membantu pengguna memahami proses pengenalan wajah dengan lebih baik.

#### 5.3. Refleksi

Proyek ini memberikan pemahaman mendalam tentang penerapan aljabar linear (nilai/vektor eigen) dalam *machine learning*, khususnya untuk pengenalan wajah. Tantangan utama seperti pemrosesan gambar yang lambat dan akurasi yang belum optimal menjadi pelajaran berharga dalam mengoptimalkan algoritma numerik. Kolaborasi tim selama pengembangan juga memperkuat kemampuan *problem-solving*, terutama dalam mengintegrasikan komputasi intensif dengan antarmuka pengguna. Kedepannya, pengalaman ini dapat menjadi pondasi untuk eksplorasi metode *deep learning* seperti CNN yang lebih akurat, sekaligus mengasah keterampilan dalam mengatasi *bottleneck* komputasi pada sistem *real-time*.



## DAFTAR PUSTAKA

- <https://docs.python.org/3/library/tkinter.ttk.html>
- [https://hendri.lecturer.pens.ac.id/MyPublication/AkhmadHendriawan\\_IES-21-okt-2009.pdf](https://hendri.lecturer.pens.ac.id/MyPublication/AkhmadHendriawan_IES-21-okt-2009.pdf)
- <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2022-2023/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian2-2022.pdf>
- <https://stackoverflow.com/questions/53263842/how-to-calculate-eigenfaces-in-python>
- <https://www.sciencedirect.com/topics/computer-science/eigenface>

## **LAMPIRAN**

Link repositori GitHub:

<https://github.com/lutfiyahistiana/face-recognition-allin2025>

Link video demonstrasi program:

<https://youtu.be/rhHla9gulaI>