

**LAPORAN TUGAS BESAR 2 IF 2123**

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



Ditujukan untuk memenuhi salah satu tugas besar mata kuliah IF2123 Aljabar Linier  
dan Geometri pada Semester I Tahun Akademik 2022/2023

Disusun oleh:

<b>Muhammad Equilibrie Fajria</b>	<b>13521047</b>
<b>Haidar Hamda</b>	<b>13521105</b>
<b>Edia Zaki Naufal Ilman</b>	<b>13521141</b>

**PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG**

**2022**

# **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 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.

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.

## BAB II

### TEORI SINGKAT

#### 2.1. Perkalian Matriks

Jika A merupakan sebuah matriks dengan ukuran  $m \times r$  dan B merupakan matriks dengan ukuran  $r \times n$ , maka hasil perkalian antar matriks A dengan B adalah matriks C dengan ukuran  $m \times n$ , seperti persamaan berikut:

$$A_{m \times r} \times B_{r \times n} = C_{m \times n}$$

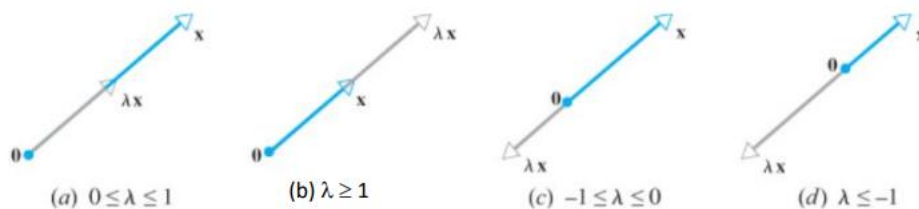
dengan setiap elemen pada matriks C didapatkan dengan persamaan berikut:

$$C = A \times B = [c_{ij}] = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{in}b_{nj}$$

Untuk melakukan perkalian antar matriks terdapat syarat yang perlu dipenuhi terlebih dahulu, yaitu jumlah kolom matriks A harus sama dengan jumlah baris pada matriks B ( $r$ ).

#### 2.2. Nilai Eigen dan Vektor Eigen

Jika A adalah matriks  $n \times n$  maka vektor tidak-nol  $x$  di  $R^n$  disebut vektor eigen dari A jika  $Ax$  sama dengan perkalian suatu skalar  $\lambda$  dengan  $x$ , yaitu  $Ax = \lambda x$ . Skalar  $\lambda$  disebut nilai eigen dari A, dan  $x$  dinamakan vektor eigen yang berkoresponden dengan  $\lambda$ . Kata “eigen” berasal dari Bahasa Jerman yang artinya “asli” atau “karakteristik”. Dengan kata lain, nilai eigen menyatakan nilai karakteristik dari sebuah matriks yang berukuran  $n \times n$ . Vektor eigen  $x$  menyatakan matriks kolom yang apabila dikalikan dengan sebuah matriks  $n \times n$  menghasilkan vektor lain yang merupakan kelipatan vektor itu sendiri. Dengan kata lain, operasi  $Ax = \lambda x$  menyebabkan vektor  $x$  menyusut atau memanjang dengan faktor  $\lambda$  dengan arah yang sama jika  $\lambda$  positif dan arah berkebalikan jika  $\lambda$  negatif.



Gambar 2.1 Ilustrasi Vektor Eigen

Untuk menghitung nilai eigen dan vektor eigen, misalkan ada matriks A berukuran  $n \times n$ .

Vektor eigen dan nilai eigen dari matriks A dihitung sebagai berikut

$$\begin{aligned}A \cdot x &= \lambda x \\I \cdot A \cdot x &= \lambda \cdot I \cdot x \\A \cdot x &= \lambda \cdot I \cdot x \\(I - A)x &= 0\end{aligned}$$

$x = 0$  adalah solusi trivial dari  $(\lambda I - A)x = 0$ . Agar  $(\lambda I - A)x = 0$  memiliki solusi tidak-nol, maka haruslah  $\det(\lambda I - A) = 0$ . Persamaan  $\det(\lambda I - A) = 0$  disebut persamaan karakteristik dari matriks A, dan akar-akar persamaan tersebut, yaitu  $\lambda$ , dinamakan akar-akar karakteristik atau nilai-nilai eigen.

Setelah nilai eigen didapat, substitusi nilai eigen tersebut ke persamaan  $(\lambda I - A)x = 0$  dan basis dari  $x_1$  dan  $x_2$  merupakan vektoreigen yang berkorespondensi dengan nilai eigen yang disubstitusi tadi, lalu ulangi dengan nilai eigen yang lain nya untuk mendapatkan semua vektor eigen.

### 2.3. Eigenface

Eigenface merupakan sebuah teknik komputasi untuk mengekstrak fitur wajah seseorang menggunakan metode *Principal Component Analysis* (PCA). Metode ini digunakan untuk mereduksi dimensi gambar wajah sehingga menghasilkan variabel yang lebih sedikit yang lebih mudah untuk diobservasi dan ditangani. Dari *eigenface* ini, akan dicari jarak *euclidean* terpendek mendapatkan gambar wajah dengan tingkat kemiripan terbesar. Pada metode ini diimplementasikan materi nilai eigen dan vektor eigen.

## BAB III

### IMPLEMENTASI PROGRAM

#### 3.1 Tech Stack

Untuk pembuatan GUI digunakan library Tkinter, yaitu GUI toolkit standar untuk Python interface. Selain Tkinter, digunakan juga library OpenCV dan PIL untuk pemrosesan gambar atau citra. OpenCV ini merupakan library standar yang digunakan untuk pemrosesan citra, computer vision, dan machine learning (ML).

Untuk proses kalkulasi matematis, digunakan NumPy yang menyediakan fungsi-fungsi operasi aljabar dan matriks yang memudahkan dalam pembuatan algoritma program ini. Selain itu, library SciPy juga digunakan untuk menghitung interpolasi data.

#### 3.2 Garis Besar Algoritma

##### a. GUI

Bagian ini menangani tampilan antarmuka pada program. Pada bagian ini program menunggu masukan user berupa folder berisi *training image* dan citra yang akan diproses untuk *face recognition*. Setelah user memasukkan citra dan folder, program memanggil fungsi untuk memproses citra dan menampilkan hasil proses.

##### b. Pencarian Eigenface

Untuk mencari *eigenface*, diperlukan nilai eigen dan eigen vector pada kumpulan dataset. Metode yang digunakan untuk mencari nilai eigen dan eigen vector tersebut adalah metode aljabar linier matriks decomposition. Namun, sebelum dilakukan komposisi, harus dilakukan beberapa proses awal kepada training image, yaitu:

- Dilakukan homogenisasi kumpulan citra pada folder training image dengan menyamakan ukuran setiap citra serta mengubahnya menjadi citra grayscale.

$$S = (\Gamma_1, \Gamma_2, \dots, \Gamma_M)$$

- Data pixel pada setiap citra yang berbentuk sudah matriks  $n \times n$  kemudian di-*flatten* atau dijadikan suatu vektor matriks dengan ukuran  $N^2 \times 1$  dan digabung dengan  $m$  citra lainnya sehingga terbentuk matriks dengan ukuran  $N^2 \times M$ .
- Dicari nilai tengah atau mean ( $\Psi$ ) dari setiap elemen vektor pada matriks ( $\Gamma_i$ )

kemudian didapatkan nilai matriks selisih ( $\Phi$ ) dengan mengurangi nilai matriks vektor dengan nilai mean.

$$\Psi = \frac{1}{M} \sum_{n=1}^M \Gamma_n \quad \Phi_i = \Gamma_i - \Psi$$

- Hitung nilai kovarian (C) dengan persamaan

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

$$L = A^T A \quad L = \Phi_m^T \Phi_n$$

Setelah didapatkan matriks kovarian C, akan dilakukan dekomposisi dengan menggunakan metode *QR Decomposition* yang memiliki persamaan:

$$A = QR$$

Dengan Q merupakan matriks orthogonal dan R merupakan matriks segitiga atas (*upper triangular matriks*).

Algoritma *QR Decomposition* diimplementasikan pada program menggunakan library numpy. Untuk mengurangi ketidakakuratan, dilakukan iterasi sebanyak 1000 kali sehingga nilai Q dan R yang kemudian akan dijadikan eigen vectors dan eigen values menghasilkan nilai yang mendekati nilai sebenarnya atau akurat. Himpunan nilai eigen didapatkan dari diagonal pada matriks R dan eigen vectors didapatkan dari matriks Q.

Dengan eigen vectors sudah didapatkan, maka dicari kumpulan eigen faces dengan cara melakukan dot product antara eigen vectors dengan nilai vektor selisih. Eigen faces yang didapatkan kemudian disimpan pada sebuah matriks.

### a. Face Recognition

Pada bagian face recognition yang diimplementasikan pada file *faceRecog.py*, akan diambil sebuah file citra yang kemudian ditemukan citra lain dari folder dataset sebelumnya yang memiliki tingkat kemiripan paling besar. Untuk melakukan hal tersebut, akan dihitung jarak euclidean terpendek antar citra dan semua citra pada dataset yang sudah dinormalisasi.

Pada awal proses, input citra yang ukurannya sudah homogenisasi dengan citra-citra pada dataset yang lain diinput akan di-*flatten* menjadi sebuah vektor. Vektor dari input citra tersebut kemudian dinormalisasi dengan cara mengurangi nilai vektor dengan nilai mean atau rata-rata yang didapatkan dari dataset.

Sebelum melakukan *face recognition*, diperlukan dua buah weight atau bobot untuk proses perbandingan. Weight pertama adalah nilai bobot dari input citra yang didapatkan dengan cara melakukan dot product antara vektor input citra yang telah dinormalisasi dengan matriks *eigenface* yang ditranspose. Weight kedua adalah nilai bobot dari kumpulan training image pada dataset yang didapatkan dengan cara melakukan dot product antara list kumpulan citra dataset yang telah dinormalisasi dengan matriks *eigenface* yang ditranspose. vektor yang telah dinormalisasi kemudian dilakukan dot product dengan matriks *eigenface* yang ditranspose untuk mendapatkan weight atau bobot dari citra yang dicari.

$$\mu_{new} = v \times \Gamma_{new} - \Psi$$

Dari kedua bobot yang didapatkan, dicari jarak euclidean terpendek antara input citra dengan kumpulan citra pada dataset dengan menggunakan persamaan:

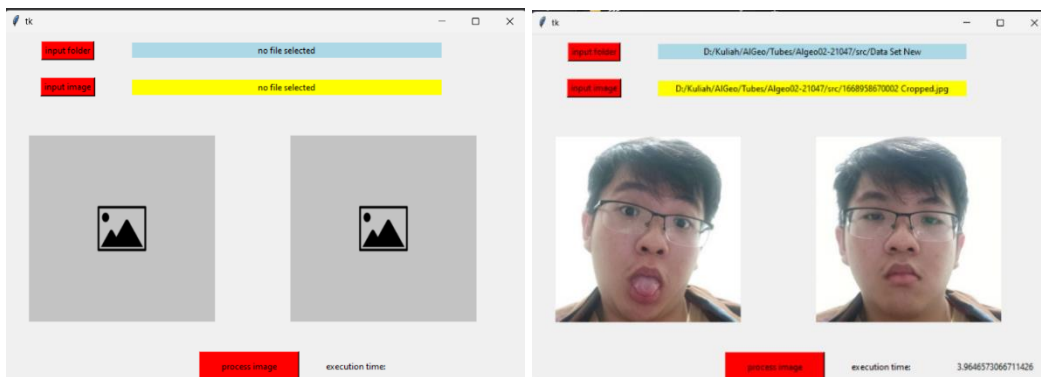
$$\varepsilon_k = \Omega - \Omega_k$$

Citra yang memiliki jarak terpendek euclidean tersebut merupakan citra dengan tingkat kemiripan tertinggi atau dalam kata lain, hasil dari proses *face recognition*. Hasil citra tersebut kemudian akan ditampilkan pada GUI untuk ditunjukkan kepada user.

## BAB IV EKSPERIMEN

### 4.1 Tampilan Aplikasi


Berikut tampilan aplikasi kami pada saat awal dibuka dan setelah mendapat hasil.




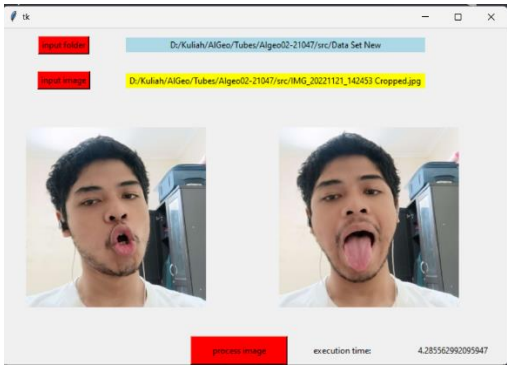


Gambar 4.1. (a) Tampilan aplikasi saat dibuka (b) Tampilan aplikasi saat menunjukkan hasil

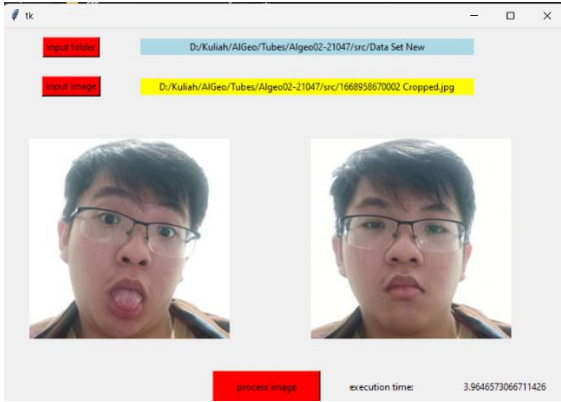


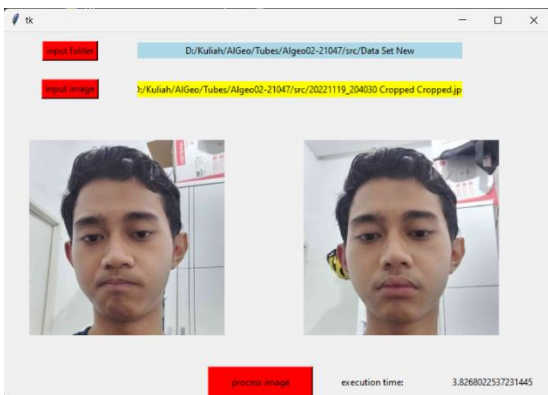
### 4.2. Hasil Face Recognition





Berikut merupakan hasil eksperimen dalam penggunaan aplikasi face recognition kami:


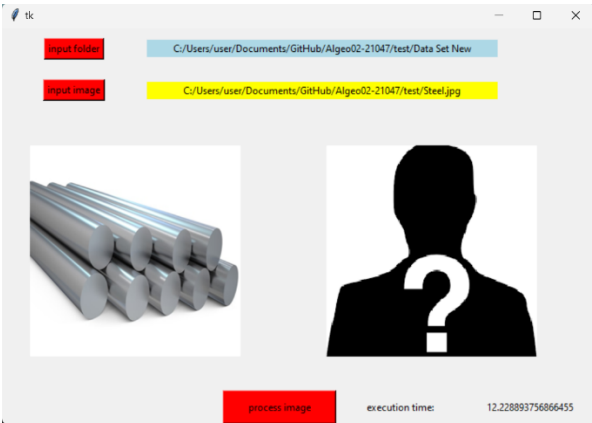
No	Deskripsi	Dimensi Citra	Citra	Waktu Eksekusi
1.	Test Face	256 x 256		4.29 sekon



	Result Face			
	Tampilan Aplikasi			
2.	Test Face	256 x 256		3.96 sekon
	Result Face			

	Tampilan Aplikasi			
3.	Test Face	256 x 256		3.83 sekon
	Result Face			
	Tampilan Aplikasi			

4.	Test Face	1085x814		12.39 sekon
	Result Face			
	Tampilan Aplikasi			
5.	Test Face	783x452		12.23 sekon

	Result Face			
	Tampilan Aplikasi			

Tabel 4.2. Hasil Eksperimen

## BAB V

### KESIMPULAN, SARAN, DAN REFLEKSI

#### 5.1. Kesimpulan

Program *face recognition* yang mengaplikasikan materi yang berhubungan dengan matriks yang dipelajari dalam mata kuliah IF2123 Aljabar Linier dan Geometri telah berhasil dirancang, dibuat, dan dijalankan. Adapun materi-materi yang diimplementasikan pada program ini berupa:

1. Nilai Eigen dan Vektor Eigen
2. Matriks Decomposition
3. Operasi Matriks dan Sifat-sifatnya

Melalui penerapan materi-materi tersebut, program ini berhasil diselesaikan sesuai dengan ketentuan yang tertera pada spesifikasi Tugas Besar 2 IF 2123 Aljabar Linier dan Geometri dengan mencari *eigenface* melalui perhitungan nilai *eigen* dan vektor *eigen* untuk melakukan *face recognition*.

Proses mendapatkan nilai *eigen* dan vektor *eigen* itu sendiri dicapai dengan mengimplementasikan dekomposisi matriks dengan metode *QR Decomposition*, yaitu suatu metode dekomposisi yang memfaktorkan suatu matriks sembarang menjadi dua buah matriks Q dan R, dimana Q merupakan matriks orthogonal dan R merupakan matriks segitiga atas (*upper triangular matrix*). Namun, nilai yang dihasilkan memiliki nilai yang tidak konsisten sehingga perlu dilakukan iterasi sebanyak mungkin agar didapatkan nilai yang mendekati nilai sebenarnya atau akurat. Maka, agar mendapatkan nilai yang cukup akurat, pada algoritma yang diimplementasikan ini, dilakukan iterasi sebanyak 1000 kali. Setelah mendapatkan nilai Q dan R, maka nilai eigen dapat dicari pada diagonal matriks dan vektor eigen didapat dari matriks R.

#### 5.2. Saran

Pengerjaan Tugas Besar 2 Aljabar Linier dan Geometri tentunya tidak luput dari hambatan dan juga kendala. Demi pelaksanaan yang lancar, penulis telah mencatat beberapa saran serta poin

yang dapat diperhatikan bagi pihak yang berminat untuk membuat program *face recognition* menggunakan *eigenface*:

- a. Alokasikan waktu yang cukup banyak untuk bereksplorasi dan mempelajari materi dan ilmu yang akan diimplementasikan sebelum pengerjaan agar pada saat proses berlangsungnya pengerjaan lebih lancar dan lebih memahami teknik algoritma yang perlu diimplementasikan.
- b. Organisasi environment pengerjaan project dengan rapi dan terstruktur, juga mengimplementasikan program secara modular, sehingga pengerjaan lebih nyaman dan mudah dalam menelusuri hasil pekerjaan.
- c. Jika pengerjaan dilakukan secara berkelompok, maka distribusi tugas kepada para anggota diharapkan bobot yang diberikan sesuai dengan kapasitas kemampuan masing-masing. Hal tersebut agar pada kelompok tidak ada anggota yang merasa sangat terbebaskan dibandingkan yang lain. Diharapkan juga agar setiap anggota untuk saling membantu satu sama lain agar pengerjaan lebih lancar dan komunikasi antar kelompok terjaga.

### **5.3. Refleksi**

Tugas Besar 2 Aljabar Linier dan Geometri ini menjadi wadah pembelajaran yang sangat berharga bagi kami. Pada zaman modern ini sudah banyak teknologi yang mengimplementasikan face recognition, namun para penulis tidak pernah mengetahui secara detail bagaimana caranya bekerja. Setelah menyelesaikan Tugas Besar 2 Aljabar Linier dan Geometri ini, para penulis menjadi lebih paham dan bisa lebih mengapresiasi teknologi ini secara lebih.

Dalam proses pengerjaan Tugas Besar ini, banyak terjadi kendala, seperti kurangnya pemahaman konsep seperti pada ekstraksi fitur pada wajah, atau seperti banyaknya keambiguan dalam spesifikasi sehingga ada beberapa momen yang membuat para penulis cukup bingung. Terdapat juga kendala kurangnya referensi yang cukup detail atau informatif untuk level pemahaman kami sehingga cukup tersesat di jalan proses pengerjaan.

Komunikasi dalam kelompok juga salah satu yang bisa dijadikan bahan refleksi. Seperti pembagian tugas yang harus sesuai kapasitas kemampuan masing-masing, komunikasi antar anggota mengenai progress pekerjaan, dan masalah-masalah lain terkait komunikasi kelompok. Hal-hal tersebut perlu ditinjau ulang dan dijadikan bahan evaluasi untuk para penulis untuk

kedepannya agar kerja menjadi lebih efektif dan efisien.



## REFERENSI

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

<https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-01-Review-Matriks.pdf>

<https://people.inf.ethz.ch/gander/papers/qrneu.pdf>

<http://webhome.auburn.edu/~tamtiny/lecture%2011.pdf>

<https://www.neliti.com/id/publications/135138/pengenalan-wajah-menggunakan-algoritma-eigenface-dan-euclidean-distance>

<https://machinelearningmastery.com/face-recognition-using-principal-component-analysis/>

## **LAMPIRAN**

Link youtube:

[https://youtu.be/6Z2riwssc\\_M](https://youtu.be/6Z2riwssc_M)

Link github:

<https://github.com/haidarhamda/Algeo02-21047>