

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



**Kelompok 42 – NT (Never Tsurrender)**

Hosea Nathanael Abetnego – 13521057

Ariel Jovananda – 13521086

Rava Maulana – 13521149

## 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.

Terdapat berbagai teknik untuk memeriksa citra wajah dari kumpulan citra yang sudah diketahui seperti jarak *euclidean*, *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 dua 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*.

## BAB 2

### Teori Singkat

#### 2.1 Perkalian Matriks

Matriks dibuat dengan melakukan operasi perkalian matriks pada dua matriks. Jika  $q = r$ , matriks  $p \times q$  dapat dikalikan dengan matriks  $r \times s$ , menghasilkan matriks  $p \times s$   $C$ . Jika, misalnya,  $q = r = n$ , maka

$$C_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

Matriks yang digunakan untuk perkalian tidak bersifat komutatif. Operasi perkalian matriks sangat berperan dalam berbagai penerapan matriks untuk pemecahan masalah, seperti menemukan solusi persamaan linier (menggunakan berbagai metode matriks), melakukan transformasi linier, dan mendapatkan matriks SVD hanyalah beberapa contoh fungsi yang digunakan dalam berbagai aplikasi matriks untuk menyelesaikan masalah.

#### 2.2 Nilai Eigen dan Vektor Eigen

Nilai eigen mewakili nilai definisi dari matriks  $n \times n$ . Mencari akar persamaan pendefinisian yang mempunyai rumus berikut

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

merupakan salah satu cara untuk mencari nilai eigen matriks  $A$ . Sedangkan vektor eigen ( $x$ ) adalah vektor yang menyatakan.

$$Ax = \lambda x.$$

Berikut adalah cara mendapatkan vektor eigen.

1. Tentukan nilai eigen.
2. Tentukan basis untuk ruang eigen.
3. Basis kemudian digunakan untuk membuat vektor eigen.

Ada beberapa disiplin ilmu di mana vektor eigen digunakan. Vektor eigen digunakan dalam grafik komputer, khususnya di bidang informatika.

#### 2.3 Eigenface

Untuk mendapatkan nilai eigenface harus melewati beberapa tahap.

1. Mencari matriks *covarian* (matriks *covarian* digunakan untuk mencari *eigenvalue* dan *eigenvector*)

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

$$L = A^T A \quad L = \phi_m^T \phi_n$$

2. Setelah mendapatkan matriks *covarian*, kita bisa mencari *eigenvalue*, berikut adalah contoh cara mendapatkan *eigenvalue*.

$$Lxv = \lambda x v$$

$$Lxv = \lambda I x v$$

$$(L - \lambda I) = 0 \text{ atau } (\lambda I - L) = 0$$

$$0 = \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

$$0 = \det \begin{bmatrix} \lambda - 2 & -1 & 0 \\ -1 & \lambda - 2 & 0 \\ 0 & 0 & \lambda - 3 \end{bmatrix}$$

Maka *eigenvalue* yang dihasilkan adalah

$$\lambda_1 = 3, \lambda_2 = 1, \lambda_3 = 3$$

$$v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

3. Ketika *eigenvalue* sudah diperoleh, langkah selanjutnya adalah untuk mendapatkan *eigenvector*.

$$\begin{bmatrix} \lambda - 2 & -1 & 0 \\ -1 & \lambda - 2 & 0 \\ 0 & 0 & \lambda - 3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{Dihasilkan eigenvector } v_1 \text{ adalah } \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix},$$

Ulangkan proses, untuk v2 dan v3.

4. Tahap terakhir adalah menghitung *eigenface*-nya, yang bisa diperoleh dengan rumus berikut.

$$\mu_i = \sum_{k=1}^M v_{ik} \phi_k$$

$$l = 1, \dots, M$$

## BAB 3

### Implementasi Program

#### 3.1 Main Program

Program utama yang mengompilasi semua fungsi terdapat pada file `FaceRecognition.py`. fungsi `train_dataset` digunakan untuk membentuk data training face yang akan digunakan untuk menentukan Euclidean distance antara *test face* dan *training face*. Untuk mengeksekusi semua fungsi dan GUI, file `main.py` digunakan dalam mengompilasi integrasi penghitungan dengan GUI.

#### 3.2 Selisih dan Mean

Langkah pertama yang dilakukan program adalah menentukan matriks selisih setiap gambar dan matriks rata-rata dari gambar-gambar tersebut kemudian matriks selisih yang diperoleh, digabung kan sehingga terbentuk matriks A yang setiap kolomnya merupakan matriks selisih setiap training face, fungsi `mean_selisih` digunakan untuk menghasilkan dua hal, yaitu matriks A (matriks hasil konkatenasi matriks selisih) dan matriks mean.

#### 3.3 Matriks Kovarian

Matriks A yang telah diperoleh dari fungsi `mean_selisih` kemudian digunakan untuk menentukan matriks *covarian*. Matriks  $A^T$  dikalikan dengan matriks A untuk menghasilkan matriks *covarian*. Matriks *covarian* tersebut kemudian digunakan untuk menentukan *eigenvector*.

#### 3.4 Eigenvalue, Eigenvector, dan Eigenface

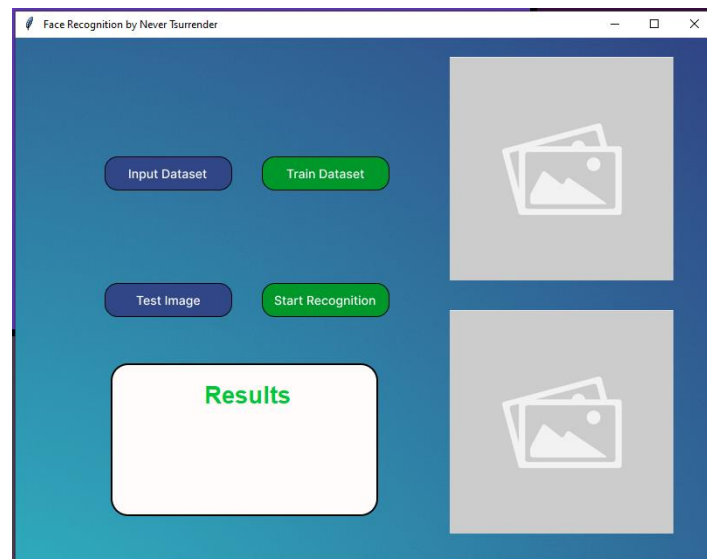
Terdapat beberapa fungsi, antara lain adalah fungsi `eigen`, `qr_decomposition`, dan `householder_transformation`. Fungsi utama dari fungsi-fungsi tersebut dalah fungsi `eigen` yang akan mengembalikan matriks *eigenvector*. Eigen vector kemudian digunakan untuk menentukan *eigenface*. Penghitungan *eigenface* dilakukan pada file `main` (`FaceRecognition.py`) dengan menormalisasikan *eigenvector*.

#### 3.5 Koefisien dan Euclidian Distance

Setelah diperoleh *eigenface* untuk semua gambar, ditentukan matriks *weight* atau koefisien dari setiap *eigenface* untuk setiap gambar. Hal yang sama dilakukan untuk *test face* sehingga diperoleh pula matriks *weight* dari *test face*. Kemudian *weight* dari *test face* akan dibandingkan dengan setiap matriks *weight* dari *training face* dan ditentukan *euclidean distance*-nya sehingga diperoleh gambar yang memiliki jarak terdekat dengan *test face*.

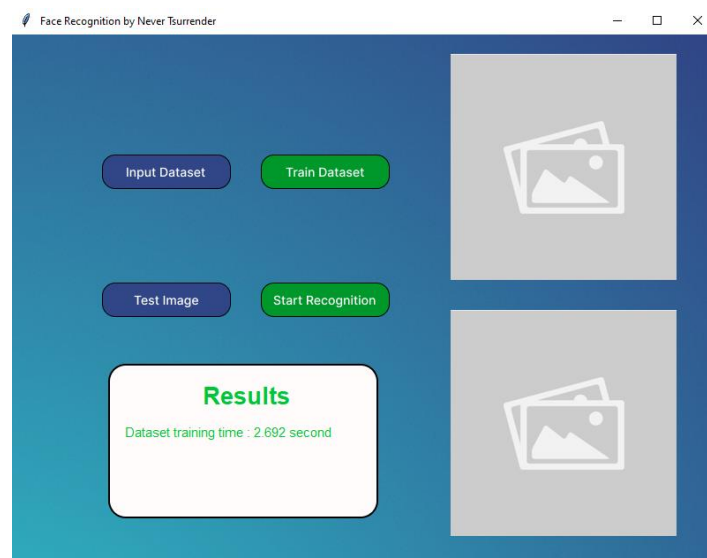
### 3.6 Graphical User Interface

Data-data yang diperoleh tersebut ditampilkan melalui GUI menggunakan modul tkinter. Tampilan awal sebelum dilakukan penghitungan adalah seperti berikut :



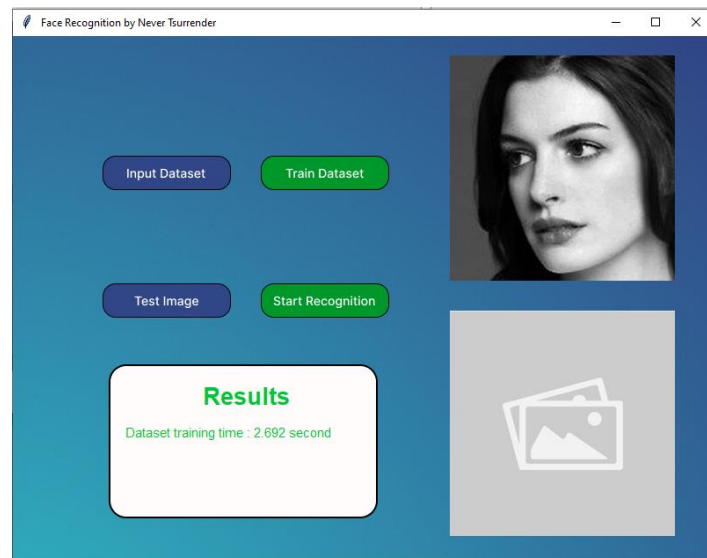
Gambar 3.1. Tampilan awal GUI

Kemudian dengan menekan tombol “Input Dataset”, user diminta untuk memasukkan folder yang berisikan gambar yang akan dijadikan training image. Dengan menekan tombol “Train Dataset” penghitungan training image dimulai dan setelah selesai akan menampilkan pesan bahwa training telah selesai. Dapat dilihat pada gambar berikut:

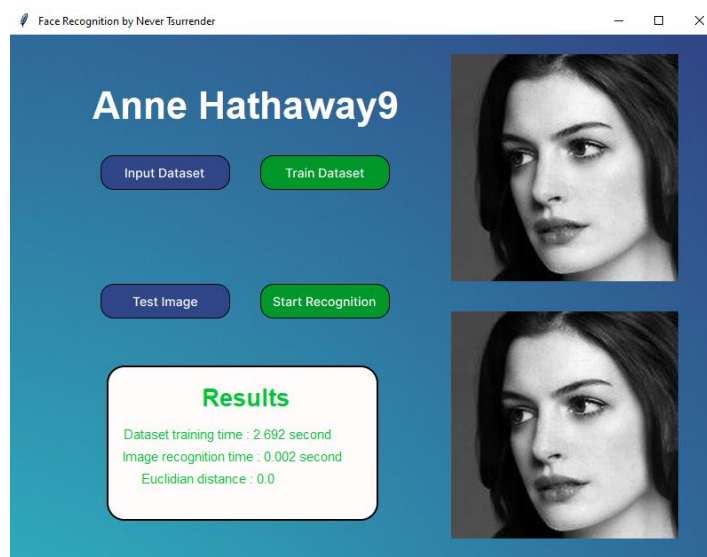


Gambar 3.2. Tampilan setelah *training* dataset

Untuk menguji gambar, pertama user harus menentukan gambar yang akan menjadi *test face* dengan menekan tombol “Test Image”. Setelah ditentukan, dengan menekan tombol “Start Recognition” program akan memulai penghitungan dan menentukan gambar dengan *euclidean distance* terkecil terhadap *test face*. Tampilan pada GUI adalah sebagai berikut :



Gambar 3.3. Tampilan setelah memasukkan *test image*




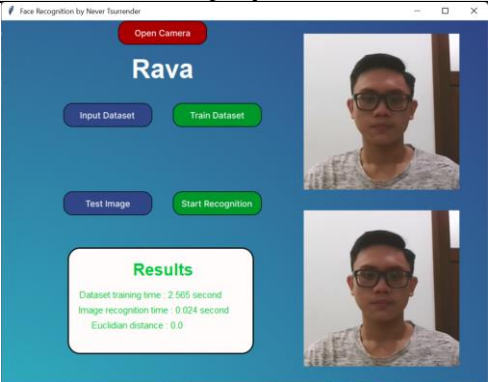

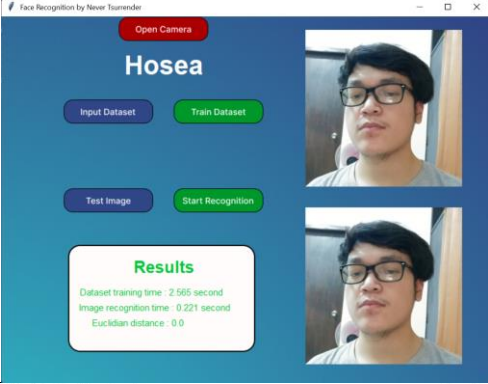


Gambar 3.4. Tampilan setelah melakukan pengenalan wajah

## BAB 4

### Eksperimen

#### 4.1 Pengujian Akurasi Program

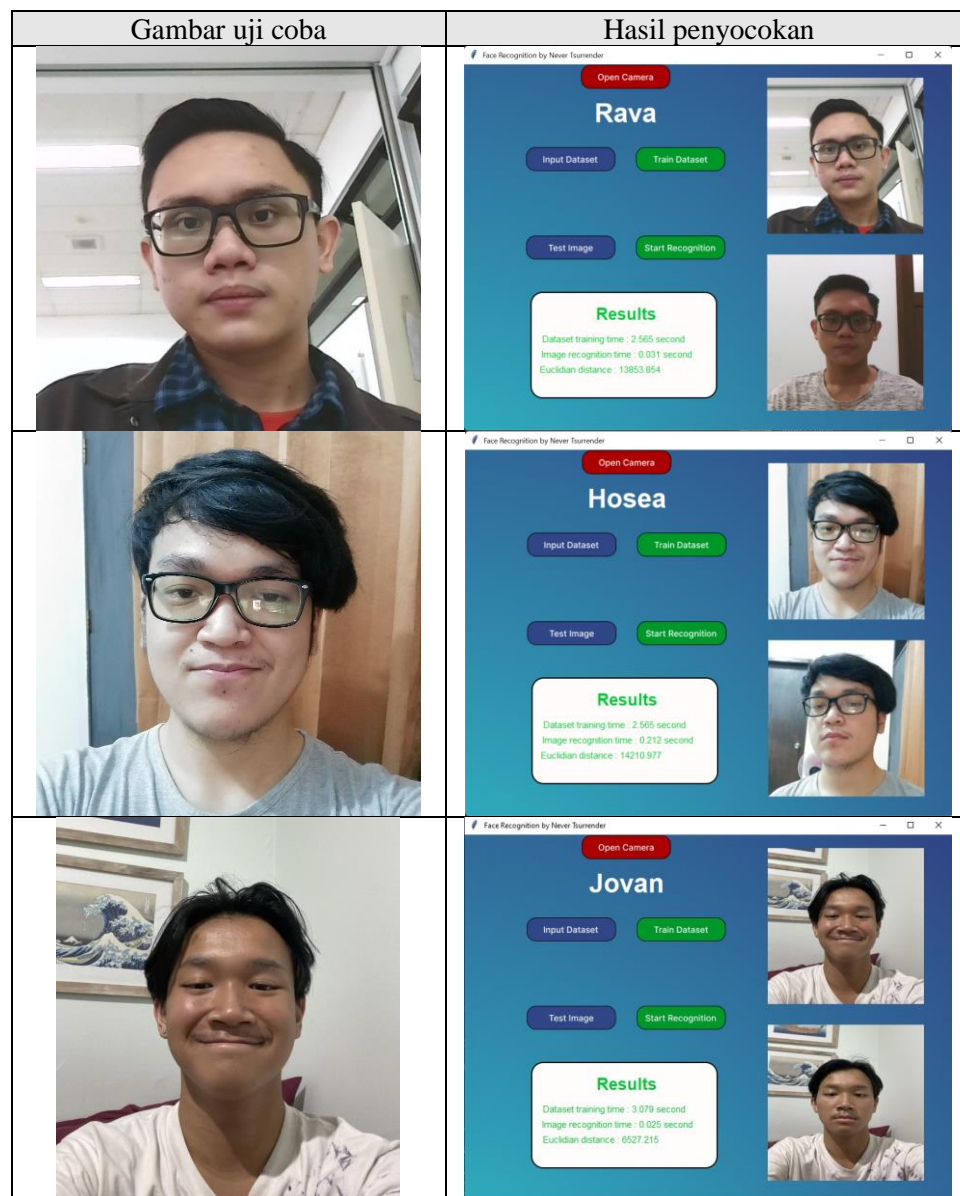
Untuk menguji tingkat akurasi program, kami menggunakan dataset yang berisi 44 gambar dengan dimensi gambar yang berbeda-beda. Pada pengujian pertama, kami mengambil gambar uji coba dari dataset. Tolak ukur keberhasilan program adalah program mampu menyocokkan gambar tersebut dengan gambar yang ada pada dataset dan *euclidian distance* dari gambar uji coba bernilai nol karena gambar diambil dari dataset. Berikut ini hasil uji coba pertama yang kami lakukan.

Gambar uji coba	Hasil penyocokan	<i>Euclidian distance</i>
		0.0
		0.0
		0.0

Tabel 4.1. Pengujian pertama dengan gambar uji coba yang diambil dari dataset



Pada pengujian kedua, kami mengambil gambar uji coba dari luar dataset. Tolak ukur keberhasilan program adalah program mampu mengenali wajah pada gambar uji coba dan menampilkan gambar pada dataset yang memiliki wajah yang sama seperti gambar uji coba. Berikut ini hasil uji coba kedua yang kami lakukan.



Tabel 4.2. Pengujian kedua dengan gambar uji coba diluar dataset

Berdasarkan kedua pengujian di atas, program pengenalan wajah menggunakan metode *eigenface* kelompok kami sudah memiliki tingkat akurasi yang baik dalam mengenali wajah dari gambar uji coba. Pada pengujian pertama, *euclidian distance* yang bernilai nol menunjukkan bahwa algoritma pengolahan gambar yang digunakan pada proses pengolahan dataset dan pengolahan gambar uji coba sudah diimplementasikan dengan benar. Pada pengujian kedua, walaupun gambar uji coba yang digunakan tidak terdapat di dalam dataset, program mampu mengenali wajah orang yang terdapat pada gambar dan menyocokkannya

dengan wajah pada dataset. Hal tersebut menunjukkan algoritma pembentukan wajah rata-rata (*meanface*) dan *eigenface* sudah diimplementasikan dengan benar.

## 4.2 Pengujian Efisiensi Program

Pengujian efisiensi program dilakukan dengan mengukur waktu eksekusi program terhadap variasi jumlah gambar pada dataset. Waktu eksekusi program dibagi menjadi dua, waktu pengolahan dataset (*training time*) dan waktu pengenalan wajah (*image recognition time*). Berikut ini hasil pengukuran waktu eksekusi program yang kami lakukan.

Ukuran dataset (jumlah gambar)	<i>Training time</i> tanpa <i>cache</i> (detik)	<i>Training time</i> dengan <i>cache</i> (detik)	<i>Image recognition</i> <i>time</i> (detik)
20	4.815	1.071	0.004
40	8.105	1.774	0.01
60	11.642	3.124	0.014
80	16.901	3.868	0.022
100	21.242	4.874	0.038
120	24.353	5.063	0.059
140	28.946	6.891	0.056
160	33.179	7.841	0.044
180	36.647	8.218	0.08
200	41.384	9.525	0.112

Tabel 4.3. Pengukuran waktu eksekusi program terhadap variasi jumlah gambar pada dataset

Berdasarkan tabel di atas, waktu eksekusi program bertambah secara linear seiring dengan bertambahnya jumlah gambar pada dataset. Hal ini menunjukkan bahwa algoritma pengenalan wajah yang digunakan memiliki kompleksitas  $\Theta(n)$ . Selain itu, penggunaan file *cache* pada dataset dapat mengurangi waktu eksekusi program hingga empat kali lipat dari waktu eksekusi tanpa menggunakan file *cache*.

## BAB 5

### Kesimpulan, Saran, dan Refleksi

#### 5.1 Kesimpulan

Program ini berhasil mengaplikasikan penggunaan nilai Eigen dalam sistem *face recognition*. Nilai eigen digunakan untuk menentukan *eigenface* yang kemudian menjadi acuan program untuk menentukan gambar dengan jarak terdekat dari wajah yang diuji. Program dapat menerima sekian banyak *training image* dan *test face* yang kemudian ditentukan *eigenface*-nya menggunakan penghitungan nilai eigen dan ditentukan satu gambar dengan *euclidean distance* terdekat dengan *test face*.

Program ini juga berhasil memanfaatkan modul Tkinter pada python untuk membuat GUI yang mengintegrasikan program *face recognition*. Tampilan GUI mempermudah pengguna untuk menggunakan program sehingga dapat mempercepat pemakaian program.

#### 5.2 Saran

Dalam pengerjaan tugas besar ini, terdapat beberapa kendala yang dialami, salah satu kendala terbesar yang dialami adalah perubahan algoritma 4 hari sebelum pengumpulan. Ini terjadi karena referensi yang menjadi acuan program tidak sesuai dan harus diubah menurut referensi lain. Oleh karena itu, disarankan untuk menentukan referensi yang sesuai terlebih dahulu sebelum memulai pembuatan program.

#### 5.3 Refleksi

Pengerjaan tugas besar ini dilakukan seiring dengan tugas besar 3 mata kuliah lainnya sehingga pengerjaan menjadi sulit dikarenakan harus membagi waktu antara tugas-tugas lainnya. Pembagian waktu dan kerja sama yang baik menjadi kunci dalam menyelesaikan semua pekerjaan ini tepat waktu dan maksimal. Kerja sama yang baik dan kontribusi yang baik juga akan meringankan beban setiap anggota kelompok dalam pengerjaan keempat tugas besar tersebut.

## Referensi

1. <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-Nilai-Eigen-dan-Vektor-Eigen-Bagian1.pdf>
2. <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2022-2023/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian2-2022.pdf>
3. <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2022-2023/Tubes2-Algeo-2022.pdf>
4. <https://jurnal.untan.ac.id/index.php/jcskommipa/article/download/9727/9500>
5. <https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/>

## Repository github

<https://github.com/RMA1403/Algeo02-21057>