

# **LAPORAN TUGAS BESAR**

**IF2123 Aljabar Linier dan Geometri  
Kelas Mahasiswa (K-1) / Kelompok teskenal**

**Dosen : Dr. Judhi Santoso, M.Sc.**



**Anggota Kelompok :**

**(13521059) Arleen Chrysanthia Gunardi**

**(13521097) Shidqi Indy Izhari**

**(13521107) Jericho Russel Sebastian**

**Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung**

**2022**

# Daftar Isi

<b>Daftar Isi</b>	<b>2</b>
<b>Bab 1:</b>	
<b>Deskripsi Masalah</b>	<b>3</b>
1.1 Deskripsi Masalah	3
1.2 Solusi Permasalahan	3
<b>Bab 2:</b>	
<b>Teori Singkat</b>	<b>5</b>
2.1 Perkalian Matriks	5
2.2 Nilai Eigen	5
2.3 Vektor Eigen	5
<b>Bab 3:</b>	
<b>Implementasi program</b>	<b>7</b>
3.1 Penjelasan kaskas yang digunakan	7
3.2 Garis besar algoritma face recognition yang diimplementasikan	9
<b>Bab 4:</b>	
<b>Eksperimen</b>	<b>12</b>
4.1 Hasil pengujian gambar	12
<b>Bab 5:</b>	
<b>Penutup</b>	<b>15</b>
5.1 Kesimpulan	15
5.2 Saran	15
5.3 Refleksi	16
<b>Daftar Referensi</b>	<b>17</b>
<b>Lampiran</b>	<b>18</b>

# Bab 1:

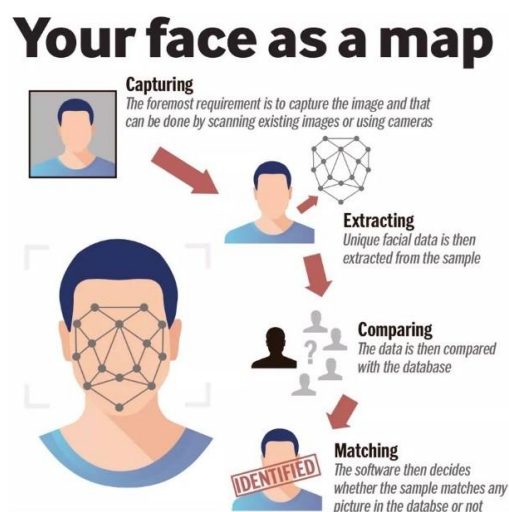
## Deskripsi Masalah

### 1.1 Deskripsi Masalah

Pengenalan wajah atau *face recognition* merupakan teknologi biometrik yang dapat digunakan untuk mengidentifikasi wajah seseorang untuk berbagai kepentingan khususnya di bidang keamanan. Program pembuatan pengenalan wajah melibatkan kumpulan citra wajah tersebut yang sudah disimpan pada database yang kemudian berdasarkan kumpulan citra wajah tersebut, program dapat mempelajari bentuk wajah dan mencocokkan antara kumpulan citra wajah dengan citra wajah baru yang akan diidentifikasi. Berbagai teknik dapat digunakan untuk membuat program *face recognition* seperti *cosine similarity* dan jarak Euclidean, *principal component analysis* (PCA), dan Eigenface.

### 1.2 Solusi Permasalahan

Pada tugas besar mata kuliah IF2123 Aljabar Linier dan Geometri yang ke-2 ini, kami akan mengimplementasikan penggunaan Eigenface untuk pembuatan program *face recognition*. Tahap alur program pengenalan wajah bekerja adalah sebagai berikut:



Program ini terdiri dari dua tahapan utama yaitu *training* dan pencocokan. Pada tahap *training*, diperlukan *database* yang terdiri dari sejumlah citra wajah yang telah dinormalisasi dari RGB ke Grayscale yang kemudian akan dihitung matriks Eigenface-nya. Selanjutnya, pencocokan akan dilakukan dengan cara mencari selisih terdekat antara nilai eigen dari wajah yang berada pada *database* dengan nilai eigen dari wajah yang ingin dikenali. Pembuatan program akan dikerjakan dengan menggunakan bahasa pemrograman Python dan memanfaatkan beberapa *library* seperti OpenCV, PIL, dan numpy.

## Bab 2:

# Teori Singkat

### 2.1 Perkalian Matriks

Matriks, pada umumnya, memiliki dua unsur utama yaitu kolom dan baris. Perkalian matriks melibatkan dua komponen ini. Setiap nilai di baris matriks yang pertama akan dikalikan dengan setiap nilai di kolom matriks yang kedua kemudian dijumlahkan nilainya untuk mendapatkan hasil dari perkalian matriks tersebut. Berikut adalah salah satu contoh perkalian dari dua buah matriks berukuran 2x2.

$$A \times B$$
$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} j & k \\ l & m \end{pmatrix}$$
$$A \times B = \begin{pmatrix} aj + bl & ak + bm \\ cj + dl & ck + dm \end{pmatrix}$$

### 2.2 Nilai Eigen

Nilai eigen, atau *eigen value* sering dilambangkan dengan  $\lambda$  yang bermakna sebagai karakteristik dari matriks yang berukuran  $N \times N$ . Apabila suatu nilai eigen dari matriks A dikalikan dengan matriks A, hasilnya akan berupa kelipatan matriks A itu sendiri. Nilai eigen dapat digunakan untuk membentuk vektor eigen.

### 2.3 Vektor Eigen

Jika matriks A berukuran  $N \times N$ , maka vektor tidak nol  $x$  di dalam  $R_n$  dinamakan vektor eigen (*eigenvector*) dari A jika  $Ax$  adalah kelipatan skalar dari  $x$ . Vektor eigen merupakan vektor karakteristik.

## 2.4 Eigenface

Eigenface merupakan salah satu pengimplementasian dari eigen value dan eigen vektor yang didasari oleh PCA (*Principal Component Analysis*) dan ditemukan oleh Sirovich bersama Kirby. Pembuatan algoritma Eigenface didasari dengan kumpulan vektor eigen. Eigenface dapat digunakan untuk membuat program *face recognition*.

## Bab 3:

# Implementasi program

### 3.1 Penjelasan kakas yang digunakan

Dalam mengerjakan tugas besar ini, kami menggunakan beberapa library Python yang terdiri dari Tkinter, numpy, os, time, dan Pillow. Tkinter merupakan salah satu library standar python untuk membuat GUI (*Graphical User Interface*) pada program. Kami memilih Tkinter sebagai library pada pembuatan GUI karena lebih *beginner-friendly* dibanding library GUI lainnya. Numpy digunakan untuk memproses array dan matriks dalam skala besar, dipakai untuk segala kebutuhan pada perhitungan matriks dalam program. Sebagai contoh, numpy digunakan dalam pengubahan foto menjadi array dan penggunaan fungsi *matmul* dalam perkalian matriks. Library os digunakan untuk mengakses nama file dan folder tanpa directory penuhnya untuk ditampilkan pada GUI. Pillow, library untuk memproses banyak jenis gambar dan dipakai untuk membuka file gambar dan *me-resize* ukuran gambar sehingga gambar dapat diseragamkan. Library time berguna untuk menghitung *processing time* program mulai dari kedua file dan folder di-input hingga hasil akhirnya didapatkan.

Berikut adalah rincian lengkap tentang kegunaan library yang kami gunakan.

Library	Pemakaian
tkinter	<ol style="list-style-type: none"><li>1. Pembacaan file dan folder menggunakan <code>filedialog</code></li><li>2. Peletakkan tulisan menggunakan <code>canvas</code></li><li>3. Pembuatan root window menggunakan <code>Tk</code></li><li>4. Penulisan tulisan dalam window menggunakan <code>canvas</code></li><li>5. Membuat display frame penampilan gambar menggunakan <code>frame</code></li></ol>

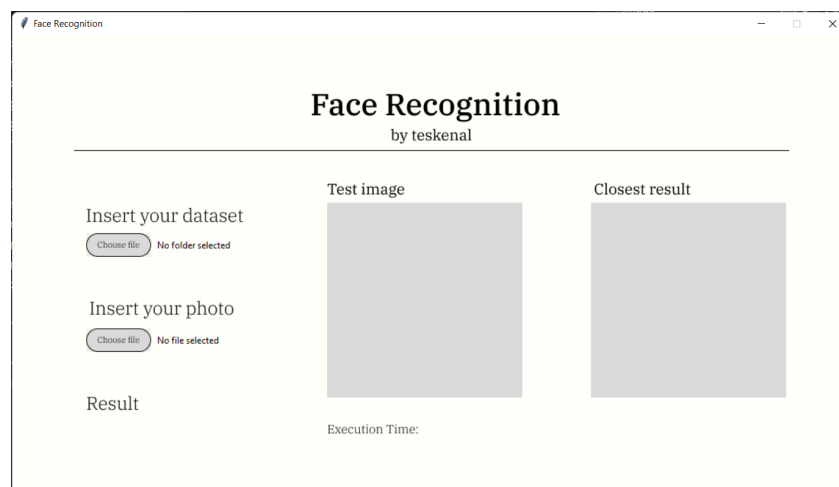
numpy	<ol style="list-style-type: none"> <li>1. Menggunakan tipe data array numpy untuk list dan matriks</li> <li>2. Menghitung operasi dasar matriks (perkalian, pengurangan, dan transpos)</li> </ol>
os	<ol style="list-style-type: none"> <li>1. Mendapatkan nama file saja (tanpa path secara keseluruhan) dengan memakai path.split</li> <li>2. Mengambil nama folder saja (tanpa path secara lengkap) dengan memakai path.basename</li> <li>3. Menggabungkan nama file saja dengan path-nya agar mendapatkan path lengkap dengan memakai path.join</li> <li>4. Memeriksa apakah path tersedia menggunakan path.isfile</li> <li>5. Mengambil seluruh file dan folder dari path yang dipilih memakai listdir</li> </ol>
time	<ol style="list-style-type: none"> <li>1. Perhitungan waktu start program hingga waktu finish untuk menghitung lamanya execution time</li> </ol>
PIL	<ol style="list-style-type: none"> <li>1. Membuka file foto dengan menggunakan Image</li> <li>2. Mengubah ukuran foto menjadi 256 x 256 menggunakan resize</li> </ol>
cv2	<ol style="list-style-type: none"> <li>1. Mengubah foto menjadi matriks memakai imread</li> <li>2. Mengubah warna dari RGB menjadi grayscale menggunakan cvtColor</li> </ol>



### 3.2 Garis besar algoritma *face recognition* yang diimplementasikan

Seluruh hasil pekerjaan kami letakkan pada folder “src” pada *repository* yang dilampirkan pada laporan ini. Dalam folder tersebut, terdapat dua subfolder lagi yaitu “GUI” dan “util”.

Sesuai namanya, folder “GUI” berisi segala keperluan yang berhubungan dengan *user interface*, yaitu source code dan foto-foto yang akan dimasukkan ke dalamnya. GUI bertujuan untuk memudahkan pengguna untuk menggunakan aplikasi *face recognition*, yaitu dengan menekan tombol saja, tidak perlu melakukan input perintah melalui keyboard. Seperti yang telah disebutkan pada subbab 3.1, kami menggunakan library tkinter untuk mengimplementasikan GUI pada program ini yang dipakai dalam source code window.py.



Pada UI, terdapat dua buah tombol yang berfungsi untuk memilih *dataset* berupa folder dan foto yang ingin diproses dengan *dataset* tersebut. Setelah kedua input dimasukkan, program akan mulai berjalan dengan cara memproses folder dataset yang telah dipilih menjadi *training image* yang akan dicari nilai minimumnya dengan foto yang dipilih menggunakan nilai eigen dan vektor eigen. Semakin kecil jarak *training image* dengan foto yang dipilih, berarti semakin mirip kedua wajah tersebut. Selain itu, GUI juga akan menampilkan seberapa cepat program dijalankan dalam hitungan detik, dan juga hasil nama foto yang mirip dengan foto yang dites.

Pada folder “util” terdapat algoritma-algoritma utama dari program yang kami buat. Algoritma tersebut meliputi *eigenface*, *eigenpair*, dan *processing image*. Secara singkat,

eigenface berisi tentang perhitungan nilai eigenface dapat dihasilkan dari input dataset hingga proses identifikasi wajah baru dengan dataset yang dimasukkan. Selain itu, eigenpair akan mengembalikan array yang berisi taksiran semua nilai eigenvalue dari matriks menggunakan metode dekomposisi QR pada matriks. Dan yang terakhir, *processing image* berguna untuk mengubah ukuran matriks menjadi 256 x 256, membaca file foto sebagai matriks, dan mengubah warna matriks gambar dari RGB menjadi grayscale.

Misalkan terdapat dataset sebanyak  $M$  gambar dan setiap gambar diubah ke dalam bentuk matriks grayscale 256 x 256. Untuk menghitung eigenface, program memanggil prosedur eigenface dengan parameter dataset, yaitu array of matriks gambar data uji ( $N \times N$ ). Kemudian, seluruh matriks dataset tersebut diubah menjadi vektor ( $N^2 \times 1$ ). Berdasarkan vektor dataset tersebut, dihitung rata-ratanya. Setelah mendapatkan vektor rata-rata, selisih antara setiap vektor gambar dengan rata-rata dihitung, kemudian digabungkan menjadi suatu matriks  $A$  dengan dimensi  $N^2 \times M$ . Dari matriks  $A$ , dapat dihitung matriks kovarian dengan mengalikan matriks  $A$  dengan matriks transpose dari  $A$ . Namun, karena matriks kovarian tersebut akan berukuran  $N^2 \times N^2$ , maka matriks kovarian dapat dihitung dengan mengalikan matriks transpose  $A$  dengan matriks  $A$  sehingga berukuran  $M \times M$ . Ini dapat dilakukan karena nilai eigen value dan eigen vector dari kedua matriks kovarian tersebut bernilai sama. Dengan matriks kovarian berukuran  $M \times M$  tersebut, diperoleh nilai eigen value dan eigen vector. Eigenface dapat diperoleh dengan mengalikan matriks  $A$  dengan seluruh eigen vector. Kemudian, hitung ruang wajah (*weight*) dari setiap gambar dataset dengan mengalikan vektor transpose dari eigenface dengan masing-masing vektor selisihnya.

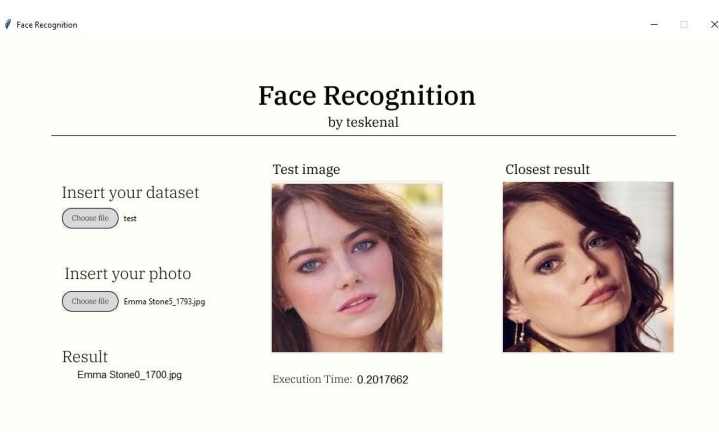
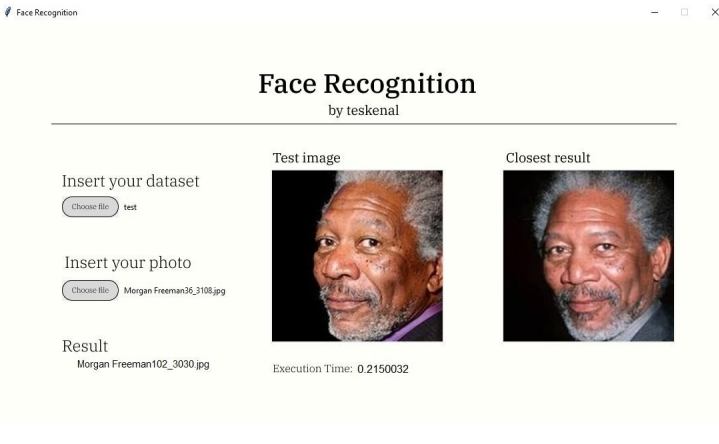
Untuk mengidentifikasi wajah mana yang terdekat dengan gambar uji, digunakan fungsi `testImage` dengan parameter `newFace`, yaitu matriks gambar wajah baru yang ingin diujikan. Pada fungsi ini, matriks wajah baru diubah ke dalam bentuk vektor. Kemudian, dari vektor wajah baru, dicari selisih antara vektor wajah baru dengan vektor rata-rata dataset. Untuk menghitung ruang wajah (*weight*) baru, seluruh eigenface dataset dikalikan dengan selisih wajah baru. Dari situ, dapat dihitung jarak Euclidean antara wajah baru dengan dataset. Jarak Euclidean dihitung dengan menghitung *magnitude* (panjang) vektor selisih antara ruang wajah baru dengan ruang wajah masing-masing gambar di dataset. Gambar wajah dengan jarak Euclidean terkecil

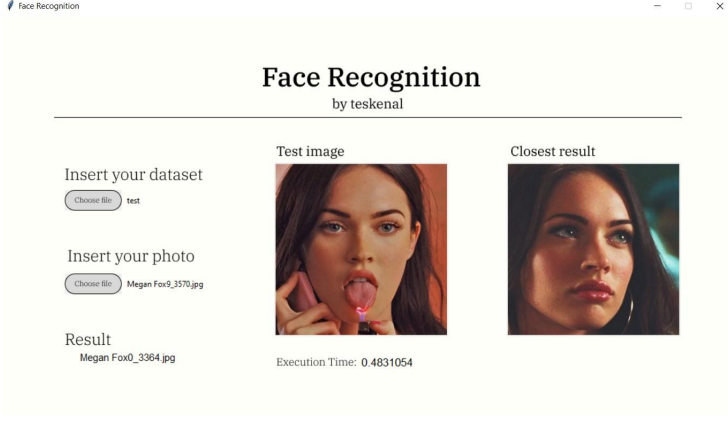
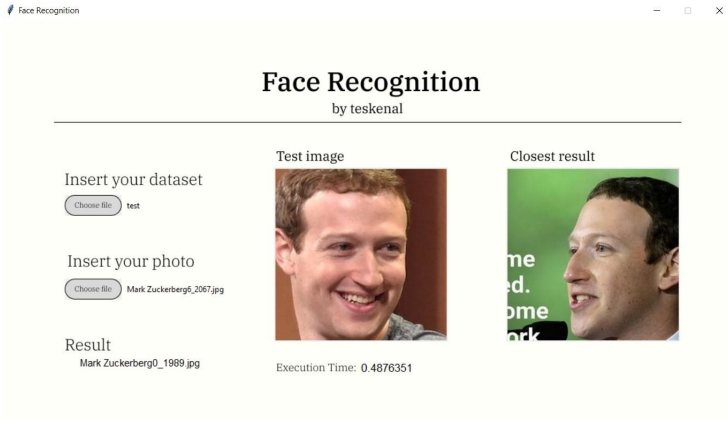
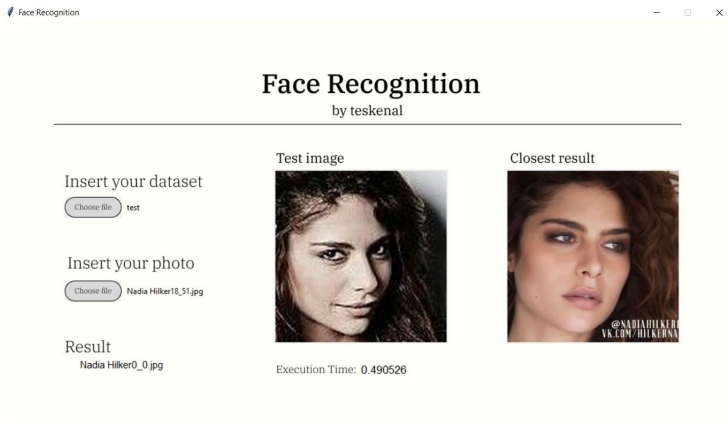
merupakan gambar wajah terdekat dengan wajah baru yang diujikan. Dengan demikian, gambar wajah terdekat dapat diperoleh dengan algoritma tersebut.

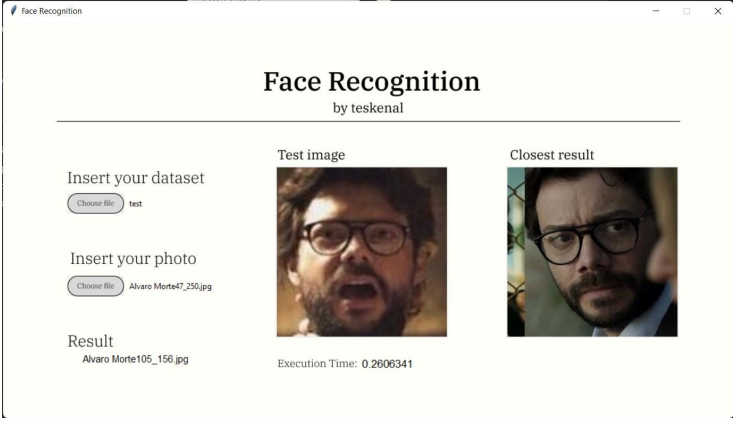
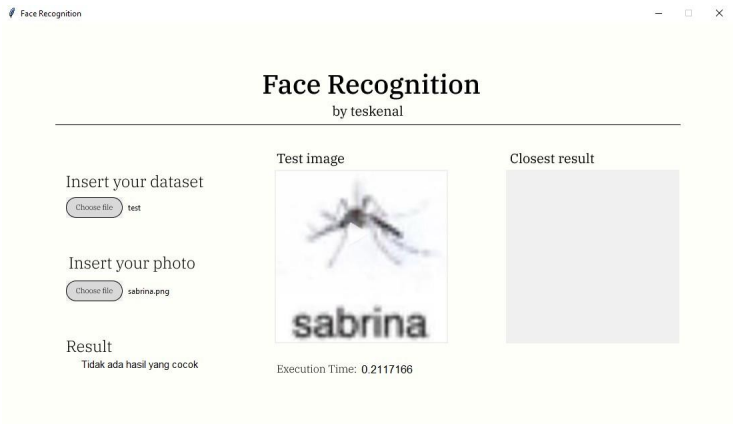
# Bab 4:

## Eksperimen

### 4.1 Hasil pengujian gambar

No	Gambar + hasil	Detail
1		<p>File uji : Emma Stone5_1793</p> <p>File hasil: Emma Stone0_1700</p> <p>Execution time: 0.2017662 detik</p> <p>Euclidean distance: 696052215.116415</p>
2		<p>File uji: Morgan Freeman36_3108</p> <p>File hasil: Morgan Freeman102_3030</p> <p>Execution time: 0.2150032 detik</p> <p>Euclidean distance: 450587319.7374177</p>

3		<p>File uji: Megan Fox9_3570</p> <p>File hasil: Megan Fox0_3364</p> <p>Execution time: 0.4831054 detik</p> <p>Euclidean distance: 310781916.7790244</p>
4		<p>File uji: Mark Zuckerberg6_2067</p> <p>File hasil: Mark Zuckerberg0_1989</p> <p>Execution time: 0.4876351</p> <p>Euclidean distance: 507394603.0892922</p>
5		<p>File uji: Nadia Hilker18_51</p> <p>File hasil: Nadia Hilker0_0</p> <p>Execution time: 0.490526 detik</p> <p>Euclidean distance: 795515570.9883033</p>

6		<p>File uji: Alvaro Morte47_250</p> <p>File hasil: Alvaro Morte105_158</p> <p>Execution time: 0.2606341 detik</p> <p>Euclidean distance: 587723856.5367136</p>
7		<p>File uji: sabrina</p> <p>File hasil: tidak ada hasil yang cocok</p> <p>Execution time: 0.2117166 detik</p> <p>Euclidean distance: 2565285366.6797857</p>

# Bab 5:

## Penutup

### 5.1 Kesimpulan

Pembuatan program pengenalan wajah atau *face recognition* sukses dapat dibuat dengan mengimplementasikan algoritma eigenface. Algoritma eigenface terdiri dari pengaplikasian beberapa materi yang diajarkan pada mata kuliah IF2123 Aljabar Linier dan Geometri, yaitu

1. Operasi pada matriks
2. Operasi pada vektor
3. Nilai eigen
4. Vektor eigen
5. Dekomposisi matriks.

Pada dasarnya, suatu foto akan diubah menjadi bentuk matriks dan vektor agar dapat diproses dan dihitung nilai-nilai eigennya menggunakan seluruh aspek di atas kemudian wajah baru diidentifikasi dengan kumpulan foto dataset. Selanjutnya, seluruh spesifikasi pada tugas besar yang kedua ini juga telah kami penuhi sesuai dengan ketentuan yang ada. Seperti pembuatan GUI, penampilan *execution time*, dan penampilan foto hasil terdekat berdasarkan *dataset* yang dimasukkan.

### 5.2 Saran

Kami sadar bahwasannya masih terdapat beberapa kekurangan pada proses pengerjaan tugas besar ini. Berikut adalah beberapa saran yang telah kami diskusikan dan kumpulkan untuk pembuatan program serupa kedepannya:

1. Membuat GUI yang lebih menarik secara tampilan agar terlihat tidak sepi.
2. Mengatur kode looping sebagaimana mungkin agar menjadi lebih efektif sehingga proses jalannya algoritma tidak memakan waktu yang cukup lama.
3. Tambahkan fitur lainnya seperti pengimplementasian perhitungan pada kode.

Besar harapan kami agar saran-saran tersebut dapat diimplementasikan agar proses serta hasil pembuatan program dapat berjalan dan selesai dengan hasil yang lebih memuaskan.

### 5.3 Refleksi

Pembuatan tugas besar kedua pada mata kuliah IF2123 Aljabar Linier dan Geometri ini membuat kami tersadar akan betapa hebatnya implementasi matriks dan vektor dalam perkembangan teknologi. Meskipun terlihat sederhana, proses pembuatan algoritma ini ternyata memerlukan proses yang cukup rumit dan kompleks sehingga hal ini menjadi tantangan tersendiri bagi kami. Selain itu, kami juga menjadi banyak belajar tentang beragam library yang tersedia pada bahasa pemrograman python yang memudahkan dan membantu proses pembuatan program dari awal hingga akhir mulai dari numpy, tkinter, pillow, opencv, hingga os. Meskipun jauh dari kata sempurna, proses pengerjaan program pengenalan wajah ini merupakan pembelajaran yang amat berharga bagi kami.



# Daftar Referensi

1. <https://www.kaggle.com/datasets/hereisburak/pins-face-recognition>
2. <https://saintif.com/perkalian-matriks/>
3. <https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/>
4. <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-Nilai-Eigen-dan-Vektor-Eigen-Bagian1.pdf>
5. <https://stackoverflow.com/questions/12201577/how-can-i-convert-an-rgb-image-into-grayscale-in-python>
6. <https://www.plus2net.com/python/tkinter/filedialog-upload-display.php>
7. <https://www.neliti.com/publications/135138/pengenalan-wajah-menggunakan-algoritma-eigenface-dan-euclidean-distance>

# Lampiran

Link Repository GitHub: <https://github.com/JerichoFletcher/Algeo02-21059>

Link Video YouTube: <https://youtu.be/PmrFQroUkIQ>