

LAPORAN TUGAS BESAR

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

Mata kuliah: Aljabar Linier dan Geometri (IF2123)



Disusun oleh: Kelompok Tatata

Anggota :

Althaaf Khasyi Atisomya – 13521130

Cetta Reswara Parahita – 13521133

Brigita Tri Carolina– 13521156

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

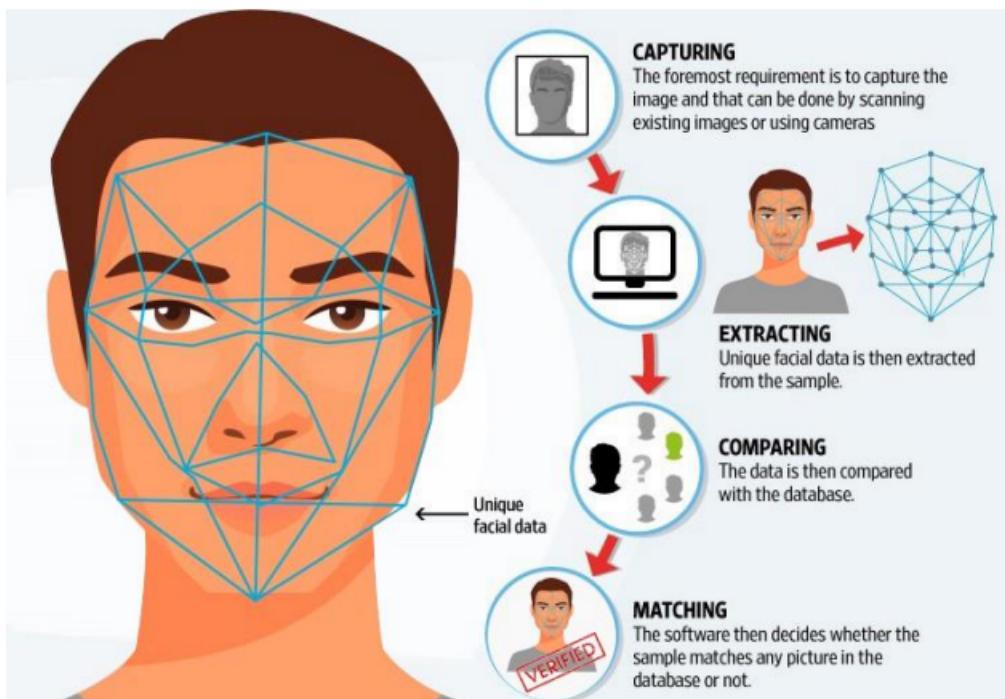
DAFTAR ISI

DAFTAR ISI.....	2
BAB I Deskripsi Masalah	3
1.1. Algoritma Eigenface	4
1.2. Tahapan Pengenalan wajah.....	5
BAB II Teori Singkat.....	6
2.1 Perkalian Matriks.....	6
2.2 Nilai Eigen dan Vektor Eigen	6
2.2.1. Menentukan Nilai Eigen dan Vektor Eigen	7
2.3 Eigenface.....	7
2.3.1. Algoritma Eigenface	7
BAB 3 Implementasi.....	9
BAB 4 Eksperimen	14
BAB 5 Kesimpulan, Saran, dan Refleksi.....	22
5.1 Kesimpulan.....	22
5.2 Saran.....	22
5.3 Refleksi.....	23
Tautan Terkait.....	24
Referensi	25

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. Alur proses sebuah sistem pengenalan wajah diperlihatkan pada Gambar 1.



Gambar 1 Alur proses di dalam sistem pengenalan wajah (Sumber:

<https://www.shadowsystem.com/page/20>)

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 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.

1.1. Algoritma Eigenface

1. Langkah pertama adalah menyiapkan data dengan membuat suatu himpunan S yang terdiri dari seluruh training image, ($\Gamma_1, \Gamma_2, \dots, \Gamma_M$)

$$S = (\Gamma_1, \Gamma_2, \dots, \Gamma_M) \quad (1)$$

2. Langkah kedua adalah ambil nilai rata-rata atau mean (Ψ)

$$\Psi = \frac{1}{M} \sum_{n=1}^M \Gamma_n \quad (2)$$

3. Langkah ketiga kemudian cari selisih (ϕ_i) antara nilai training image (Γ_i) dengan nilai tengah (Ψ)

$$\phi_i = \Gamma_i - \Psi \quad (3)$$

4. Langkah keempat adalah menghitung nilai matriks kovarian (C)

$$C = \frac{1}{M} \sum_{n=1}^M \phi_n \phi_n^T = A A^T \quad (4)$$

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

5. Langkah kelima adalah menghitung eigenvalue (λ) dan eigenvector (v) dari matriks kovarian (C)

$$C \times v_i = \lambda_i \times v_i \quad (5)$$

6. Langkah keenam, setelah eigenvector (v) diperoleh, maka eigenface (μ) dapat dicari dengan: $l = 1, \dots, M$

$$\mu_i = \sum_{k=1}^M v_{ik} \phi_k \quad (6)$$

1.2.Tahapan Pengenalan wajah

1. Sebuah image wajah baru atau test face (Γ_{new}) akan dicoba untuk dikenali, pertama terapkan cara pada tahapan pertama perhitungan eigenface untuk mendapatkan nilai eigen dari image tersebut.

$$\mu_{new} = v \times \Gamma_{new} - \Psi \quad (7)$$

$$\Omega = \mu_1, \mu_2, \dots, \mu_M$$

2. Gunakan metode euclidean distance untuk mencari jarak (distance) terpendek antara nilai eigen dari training image dalam database dengan nilai eigen dari image testface.

$$\varepsilon k = \Omega - \Omega k \quad (8)$$

3. Pada tahapan akhir, akan ditemui 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. Program dibuat dengan Bahasa Python dengan memanfaatkan sejumlah library di OpenCV (Computer Vision).

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/herveisburak/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 dalam library atau Bahasa Python.

BAB II

Teori Singkat

2.1 Perkalian Matriks

Perkalian matriks adalah nilai pada matriks yang dihasilkan dengan cara dikalikannya tiap baris dengan setiap kolom yang memiliki jumlah baris yang sama. Setiap anggota matriks ini nantinya akan dikalikan dengan anggota elemen matriks lainnya. Jika A adalah matriks berukuran $m \times n$ dan B adalah matriks berukuran , dengan elemen-elemen sebagai berikut.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix} \quad \mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{pmatrix}$$

Maka hasil perkalian Matriks A dan B dinyatakan sebagai $\mathbf{C} = \mathbf{A}\mathbf{B}$ dengan elemen matriks c sebagai berikut.

$$c_{ij} = a_{i1}b_{1j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^n a_{ik}b_{kj},$$

Hal ini menyebabkan hasil perkalian matriks A dan B hanya terdefinisi apabila jumlah kolom matriks A sama dengan jumlah baris matriks B.

2.2 Nilai Eigen dan Vektor Eigen

Jika A adalah suatu matriks berukuran $n \times n$, maka vektor eigen adalah matriks tidak nol x di R^n dari A, dengan Ax merupakan perkalian suatu skalar λ dengan x , yaitu:

$$Ax = \lambda x$$

Vektor eigen x menyatakan matriks kolom yang apabila dikalikan dengan sebuah matriks kolom yang apabila dikalikan dengan matriks $n \times n$ akan menghasilkan kelipatan dari matriks itu sendiri. Operasi $Ax = \lambda$ akan menyebabkan vektor x menyusut atau memanjang dengan faktor λ .

2.2.1. Menentukan Nilai Eigen dan Vektor Eigen

Misal suatu matriks A dengan ukuran $n \times n$. Vektor eigen dan nilai eigen dari matriks A dihitung sebagai berikut.

$$\begin{aligned} Ax &= \lambda x \\ IAx &= \lambda Ix \\ Ax &= \lambda Ix \\ (\lambda I - A)x &= 0 \end{aligned} \quad \begin{array}{l} \text{Kalikan kedua matriks dengan} \\ \text{suatu matriks identitas} \end{array}$$

Agar $(\lambda I - A)x = 0$ memiliki solusi tidak nol maka nilai determinannya harus sama dengan nol. $\det(\lambda I - A)x = 0$ adalah persamaan karakteristik dari matriks A dan akar-akar dari persamaan tersebut, yaitu λ adalah akar-akar karakteristik atau nilai-nilai eigen. Nilai dari vektor eigen dapat dihitung dengan memasukkan nilai λ yang sudah ditemukan pada persamaan $(\lambda I - A)x = 0$.

2.3 Eigenface

Matriks Eigenface dihitung dari olahan sejumlah citra wajah dalam representasi matriks. Pada program pengenalan wajah terdapat tahap *training* dan tahap pencocokan. Pada tahap *training* sekumpulan data set berupa citra wajah akan dinormalisasi dari RGB ke *Grayscale* (matriks). Hasil normalisasi kemudian digunakan pada perhitungan Eigenface. Matriks Eigenface menggunakan *eigenvector* dan *eigenvalue* pada pembentukannya. Berikut adalah algoritma pembentukan *eigenface*.

2.3.1. Algoritma Eigenface

1. Langkah pertama adalah menyiapkan data dengan membuat suatu himpunan S yang terdiri dari seluruh training image, $(\Gamma_1, \Gamma_2, \dots, \Gamma_M)$

$$S = (\Gamma_1, \Gamma_2, \dots, \Gamma_M) \quad (1)$$

2. Langkah kedua adalah ambil nilai rata-rata atau mean (Ψ)

$$\Psi = \frac{1}{M} \sum_{n=1}^M \Gamma_n \quad . \quad (2)$$

3. Langkah ketiga kemudian cari selisih (Φ) antara nilai training image (Γ_i) dengan nilai tengah (Ψ)

$$\phi_i = \Gamma_i - \Psi \quad (3)$$

4. Langkah keempat adalah menghitung nilai matriks kovarian (C)

$$C = \frac{1}{M} \sum_{n=1}^M \phi_n \phi_n^T = AA^T \quad (4)$$
$$L = A^T A \quad L = \phi_m^T \phi_n$$

5. Langkah kelima menghitung eigenvalue (λ) dan eigenvector (v) dari matriks kovarian (C)

$$C \times vi = \lambda i \times vi \quad (5)$$

6. Langkah keenam, setelah eigenvector (v) diperoleh, maka eigenface (μ) dapat dicari dengan:

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

BAB 3

Implementasi

3.1 Bagian Extractor Image (extractor.py)

Pengolahan face image menjadi matrix dilakukan dengan memanfaatkan library OpenCV. Matrix yang dihasilkan berupa matrix grayscale dari face image yang telah di resize menjadi ukuran 256 x 256. Fungsi extractImage digunakan untuk menghasilkan himpunan matrix image dari semua face image yang ada pada folder. Terdapat juga fungsi saveImage yang akan digunakan untuk menyimpan image dataset yang cocok dengan test face pada folder test/output

3.2 Bagian Eigen Value dan Eigen Face (eigen.py)

3.2.1. Dot, Magnitude, dan Identity Maker

Pada proses mengolah suatu matriks untuk menghasilkan nilai eigen dan vector eigen, operasi yang akan banyak dipakai adalah perkalian dot, perhitungan normal atau magnitude, serta pembentukan matriks identitas. Fungsi dot akan mengembalikan bilangan bulat hasil perkalian silang antar dua matriks, fungsi magnitude akan mengembalikan nilai normal dari sebuah vektor, dan fungsi identity maker akan membuat matriks identitas (I) dari input ukuran yang dimasukkan.

3.2.2. QR Decomposition

Pencarian nilai eigen dan vektor eigen akan dilakukan dengan melakukan manipulasi hasil dekomposisi matriks menjadi bentuk QR. QR *decomposition* adalah bentuk faktorisasi matriks menjadi hasil kali Q dan R, di mana Q adalah matriks orthogonal dan R adalah matriks segitiga atas. Fungsi dekomposisi QR akan menerima masukan matriks persegi dan menghasilkan 2 buah matriks Q dan R.

3.2.3. Matrix Hessenberg dan Given Rotation

Matriks Heisenberg adalah matriks yang berbentuk persegi dan memiliki bentuk hampir triangular. Terdapat dua tipe matriks Hessenberg, yakni upper-Hessenberg dan lower-Hessenberg. Pada pengurangan kali ini, digunakan matriks upper-Hessenberg untuk mendapatkan aproksimasi nilai eigen dari sebuah matriks. Upper-Hessenberg memiliki nilai 0 pada seluruh elemen di bawah sub-diagonal pertama.

Untuk menemukan nilai eigen, dilakukan operasi perkalian given rotation pada matriks hessenberg awal sehingga didapatkan sebuah matriks segitiga atas di mana semua elemen diagonal utamanya adalah nilai-nilai eigen yang dicari.

3.2.4. Segitiga Atas Checker dan Different of E Magnitude

Tujuan akhir dari perlakuan perkalian given rotation adalah didapatkan sebuah matriks segitiga atas, sehingga akan dilakukan iterasi hingga seluruh elemen dari matriks hasil berbentuk matriks segitiga atas. Pemberhentian iterasi ini dapat dilakukan dengan dua cara pengecekan, yakni apakah hasil akhir dari matriks telah berbentuk matriks segitiga atas ataukah selisih dari magnitude nilai-nilai diagonal sementaranya sudah mendekati 0 (karena hal ini mengindikasikan bahwa operasi rotasi telah menemukan nilai E yang sesuai).

Untuk itu dibuat fungsi segitiga atas checker dan perhitungan different of E magnitude pada iterasi. Dari dua metode ini, didapatkan bahwa perhitungan selisih norma E dapat menghasilkan nilai eigen dan nilai vektor yang sesuai dengan waktu kerja algoritma yang lebih efektif (didapat dua algoritma pengecekan pada kasus yang sama dapat menghemat hingga 80% waktu penggerjaan dengan menggunakan algoritma Different of E Magnitude).

3.2.5. Algoritma Eigen Values dan Eigen Vectors

Dengan melakukan kombinasi gabungan berbagai fungsi di atas, didapatkan algoritma perhitungan nilai eigen dan vektor eigen sebagai berikut.

```
% Set up initial estimate
H = hess(A);
E = diag(H);
change = 1;
steps = 0;
% loop while estimate changes
while change > 0
    Eold = E;
    % apply QR method
    [Q R] = qr(H);
    H = R*Q;
    E = diag(H);
    % test change
    change = norm(E - Eold);
    steps = steps +1;
end
```

3.3 Bagian Eigen Face (eigenface.py)

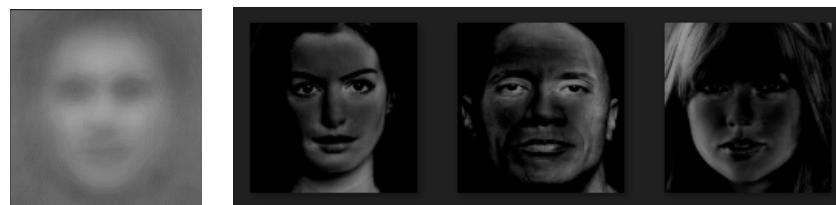
Bagian ini berisi pengolahan image dataset dan testface untuk mendapatkan image dataset dengan jarak eclidean terendah dari image test face.

3.3.1. Flatten Image

Fungsi Flatten digunakan untuk mengubah himpunan matrix image berukuran $M \times 256 \times 256$ menjadi $M \times 256^2$. Terdapat pula fungsi Unflatten yang merupakan fungsi invers dari flatten yang nantinya digunakan untuk mengembalikan image ke ukuran semula.

3.3.2. Average, Selisih, dan Covarian

Fungsi Average digunakan untuk mencari rata-rata dari image dataset (average). Fungsi selisih digunakan untuk mencari selisih(A) dataset dan average, nantinya juga digunakan untuk mencari selisih test face dan average pada tahap pengenalan wajah. Matrix covarian didefinisikan sebagai perkalian $A \times A^T$ ($256^2 \times 256^2$), matrix covarian inilah yang nantinya akan dicari eigenvalue dan eigenvectornya. Untuk simplicitas, covarian yang digunakan adalah $A^T \times A$ yang berdimensi $M \times M$ dan dibagi M (diimplementasikan pada fungsi Covarian). $A \times A^T$ dan $A^T \times A$ memiliki eigenvalue yang sama, sedangkan eigenvectornya saling berhubungan dengan persamaan $Av = \mu$ (Fungsi EigenvectorAAt). Setelah diperoleh eigen vector dari matrix covarian, eigenvector ini akan dinormalisasi dengan fungsi NormalizeEigenVector).



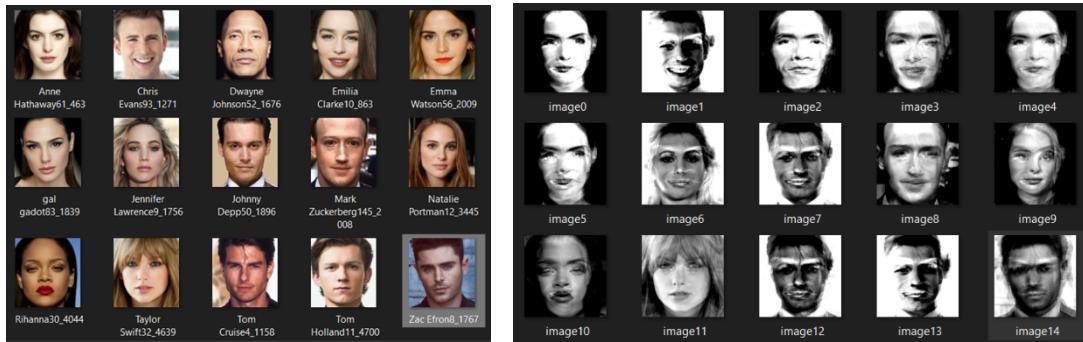
Gambar 2 Contoh average image dan selisih image



Gambar 3 Contoh eigenface

3.3.3. Tahap Pengenalan Wajah

Pada tahap ini akan dilakukan pencocokan test face dengan dataset. Pertama-tama kita perlu mengetahui bobot eigenvector setiap dataset dan test face dengan cara mengalikan tiap eigenvector(μ) dan tiap selisih (A) yang diimplementasikan pada fungsi Weight. Untuk mengecek apakah bobot yang dihasilkan sudah benar terdapat fungsi Reconstruct yang akan menghasilkan image yang telah di rekonstruksi yaitu penjumlahan average dengan perkalian eigenvector dan bobot.



Gambar 4 Dataset sebelum dan sesudah di rekontruksi

Setelah mendapatkan bobot dari dataset dan test face, akan dihitung jarak euclidean antara bobot dari dataset dan bobot dari testface dengan fungsi euclideanDistance. Pada tahapan akhir, akan ditemui 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.

3.4 Bagian Bonus Masukan dari Webcam (webcam.py)

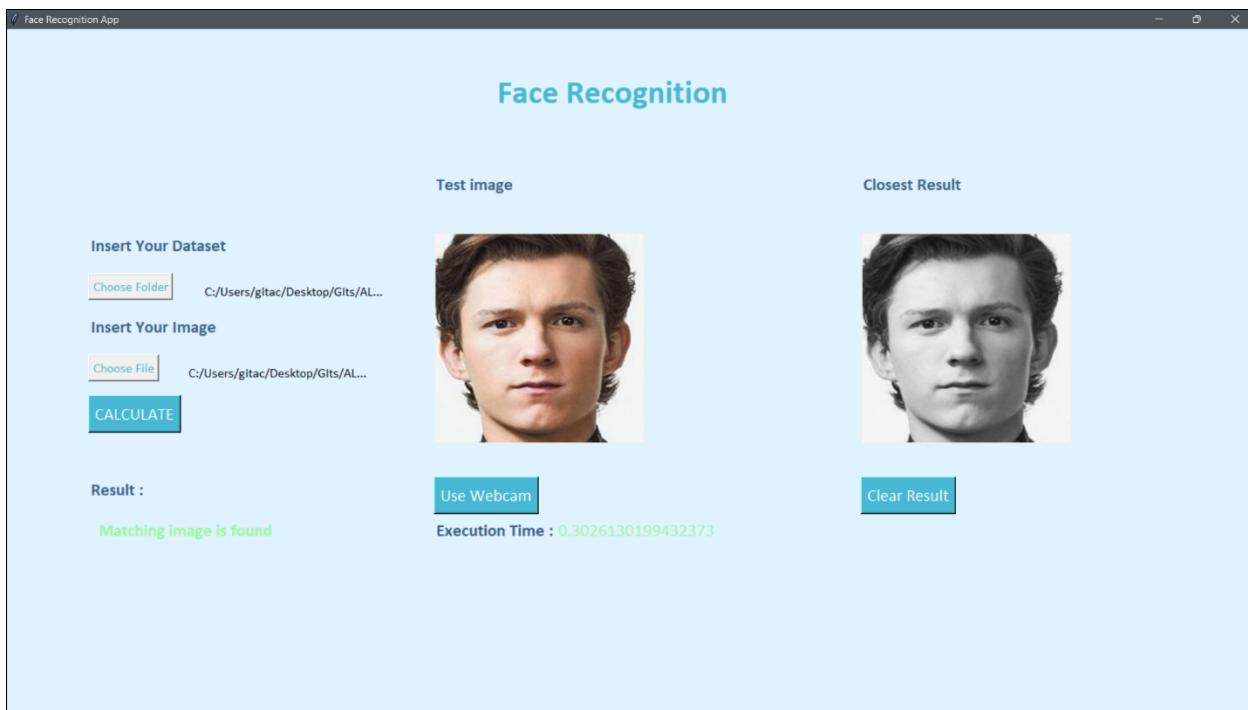
Untuk membuat fitur masukan dari webcam, diperlukan penggunaan beberapa library, di antara lain OpenCV, Operating System Interface, Matplotlib, dan Time. Masukan video dari webcam akan dilakukan dengan sebelumnya menghapus data-data input yang sudah ada di file, kemudian melakukan penyimpanan foto tiap 10 detik sekaligus pengecekan apakah ada foto di database yang memiliki kemiripan. Penyimpanan foto akan berhenti saat ditemukan foto yang jarak euclideananya di bawah threshold, telah

mencapai batas waktu pencarian (5 menit), atau user melakukan input space untuk menghentikan proses input dari webcam.

BAB 4

Eksperimen

- Dataset sejumlah 20 image, dan test image ada dan sama persis di dataset, ukuran foto 209 x 221



Analisis

Dataset image yang hanya berjumlah 20 image membuat *execution time* dari program sangat singkat yaitu hanya 0,3 detik saja. Selain itu test case yang sama persis dengan suatu image yang terdapat di dataset membuat pencarian menjadi semakin cepat juga dan nilai *euclidian distance* paling minimumnya bernilai 0.

- Dataset sejumlah 20 image, dan test image ada (orangnya ada di dataset) namun tidak sama persis di dataset, ukuran foto 1182x1600

Face Recognition

Test image



Closest Result



Insert Your Dataset

Choose Folder C:/Users/gitac/Desktop/Gits/AL...

Insert Your Image

Choose File C:/Users/gitac/Desktop/Gits/AL...

CALCULATE

Result :

Matching image is found

Use Webcam

Clear Result

Execution Time : 0.3722655773162842

Analisis

Dataset yang hanya berjumlah 20 image membuat membuat *execution time* dari program sangat singkat yaitu hanya 0,37 detik saja. Program dapat tetap merecognize *test image* yang sedikit berbeda dengan yang terdapat di dataset dengan baik. Test image yang sedikit berbeda dengan yang terdapat di dataset membuat *execution time* sedikit lebih lama dari kasus pertama.

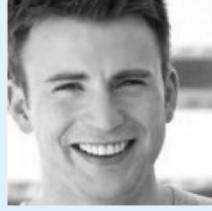
3. Dataset sejumlah 20 image, test image tidak ada pada dataset, ukuran 1200x1200

Face Recognition

Test image



Closest Result



Insert Your Dataset

Choose Folder C:/Users/gitac/Desktop/Gits/AL...

Insert Your Image

Choose File C:/Users/gitac/Desktop/Gits/AL...

CALCULATE

Result :

Matching image is found

Use Webcam

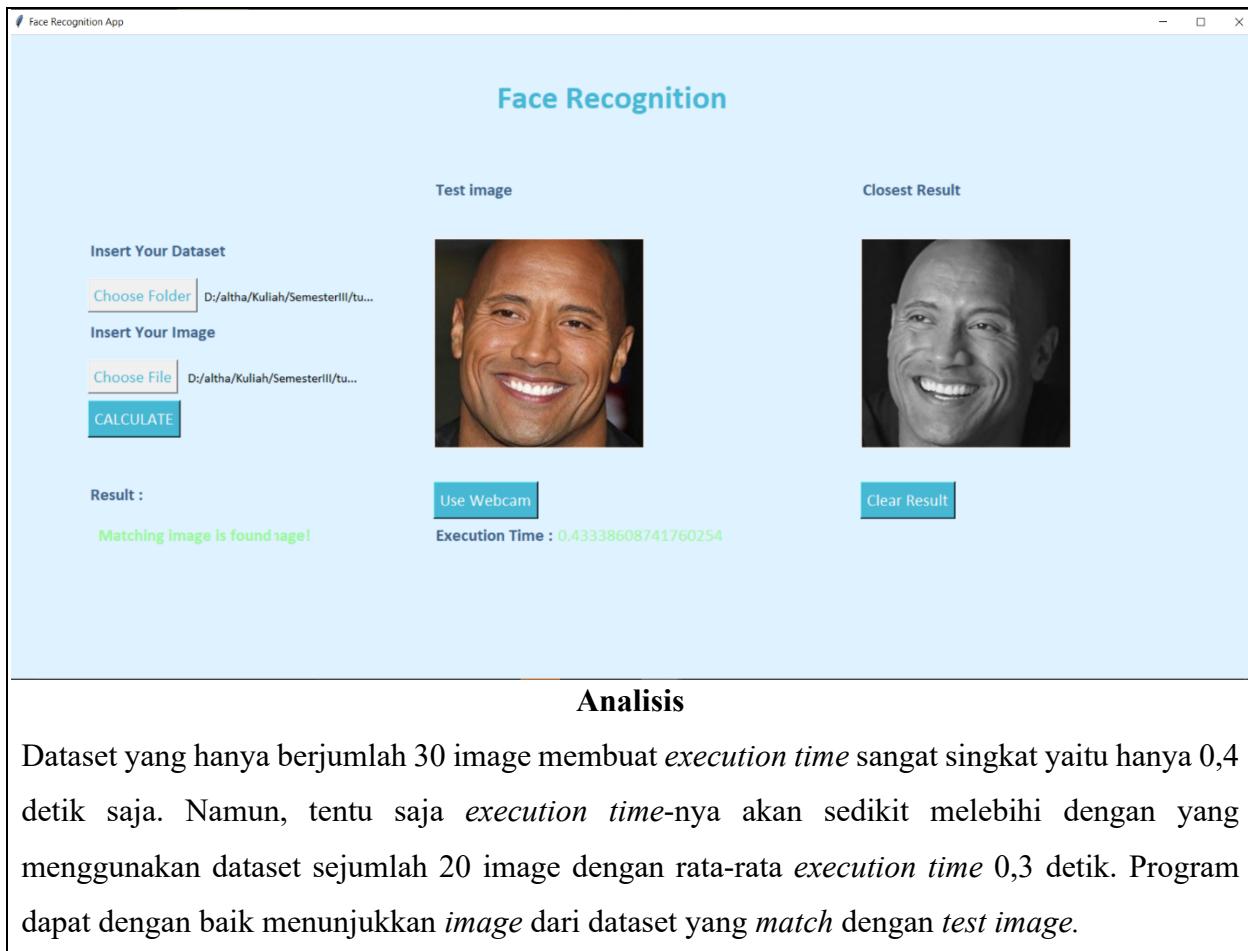
Execution Time : 0.3204984664916992

Clear Result

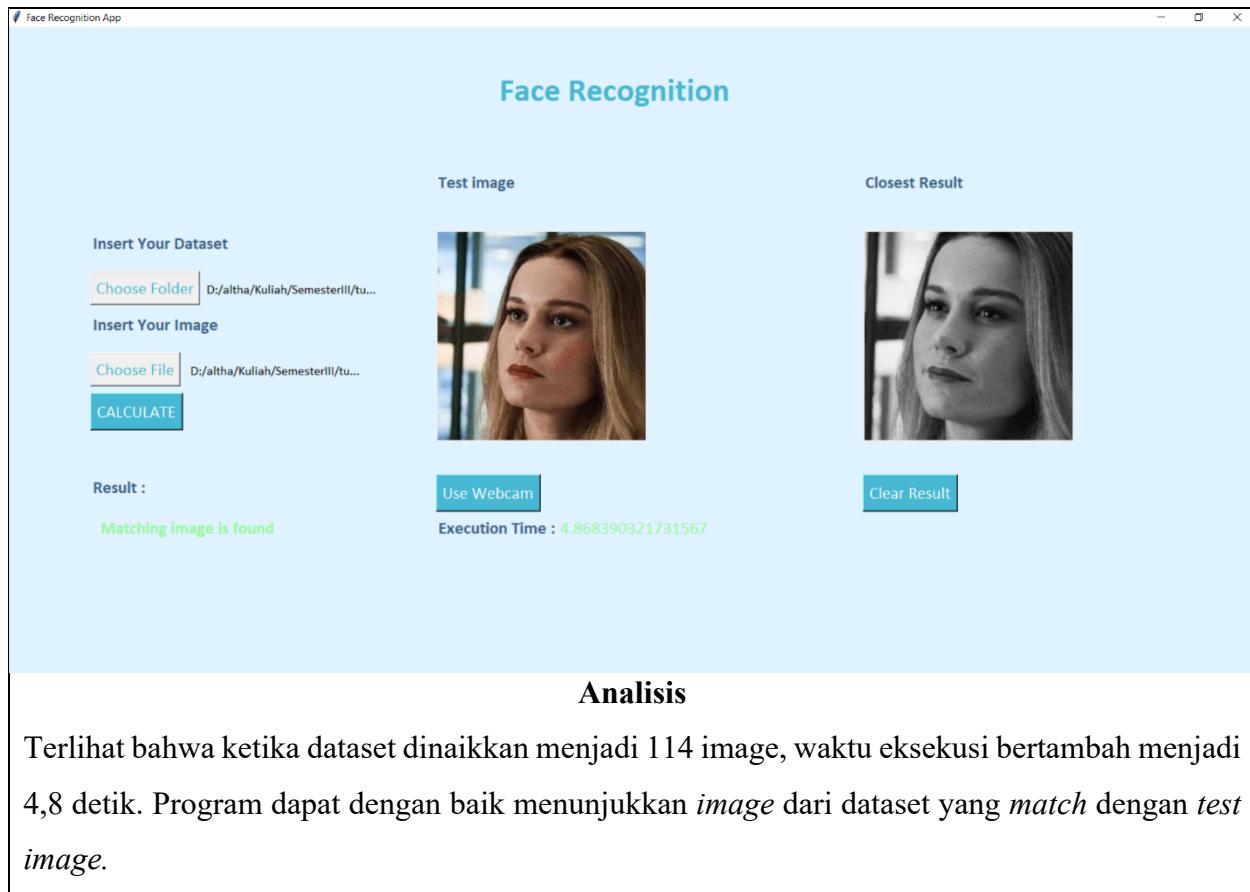
Analisis

Test image tidak ada terdapat dataset, namun program tetap dapat menunjukkan foto yang paling menyerupai test image karena euclidian distance dari kedua foto berada di bawah threshold yang diberikan yaitu di bawah 20.000. *Execution time* sangat singkat yaitu 0,3 detik karena dataset yang digunakan hanya berjumlah 20 image.

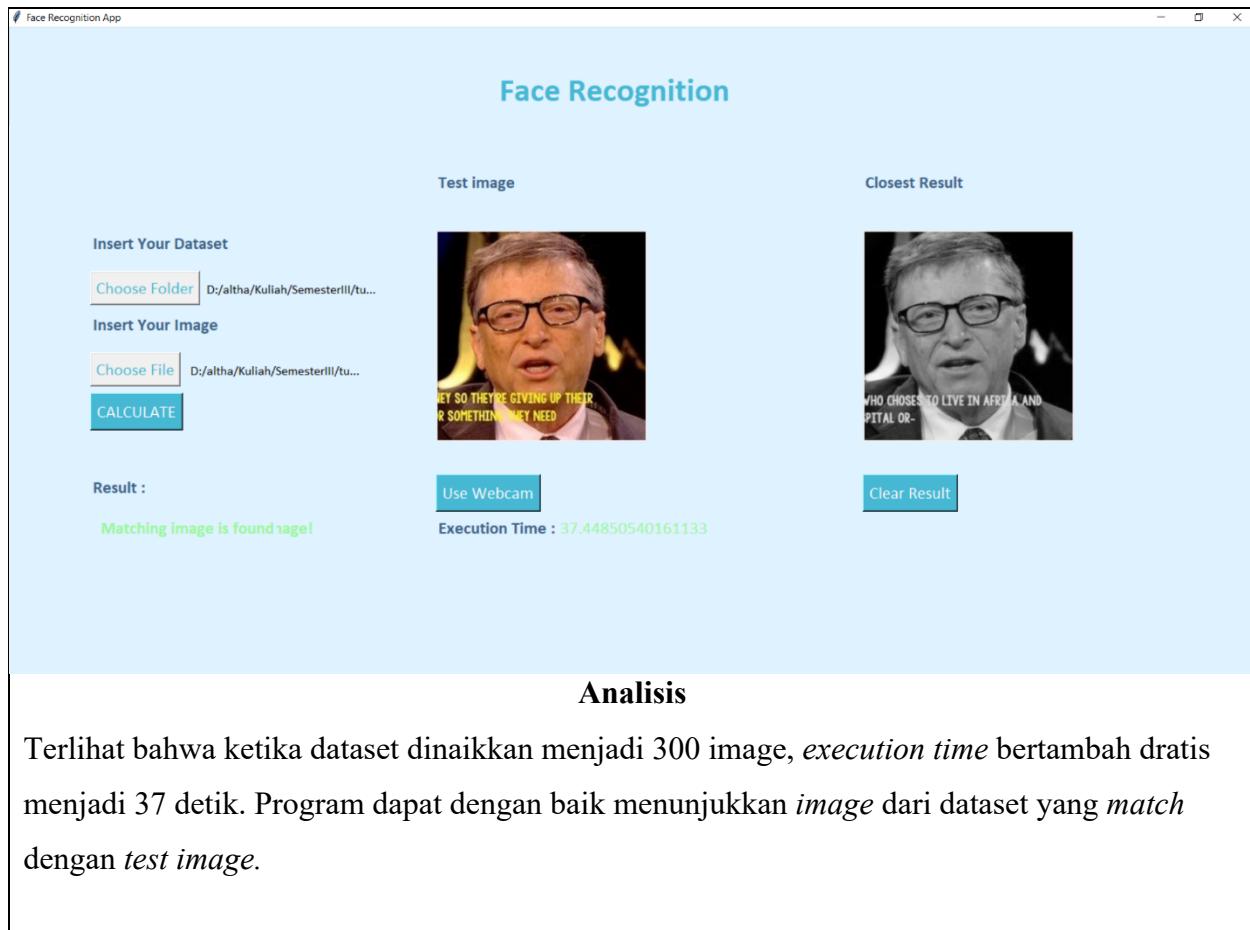
4. Dataset sejumlah 30 image, test image ada pada dataset namun tidak sama persis, ukuran foto 251x266



5. Dataset sejumlah 114 image, test image ada di dataset tetapi tidak sama persis



6. Dataset sejumlah 300 image, test image ada di dataset tetapi tidak sama persis



7. Dataset 300, image tidak sama persis dengan yang di dataset

Face Recognition

Test image



Closest Result



Insert Your Dataset

Choose Folder C:/Users/gitac/Desktop/Gits/AL...

Insert Your Image

Choose File C:/Users/gitac/Desktop/Gits/AL...

CALCULATE

Result :

Couldn't find matching image!

Use Webcam

Execution Time : 25.097874641418457

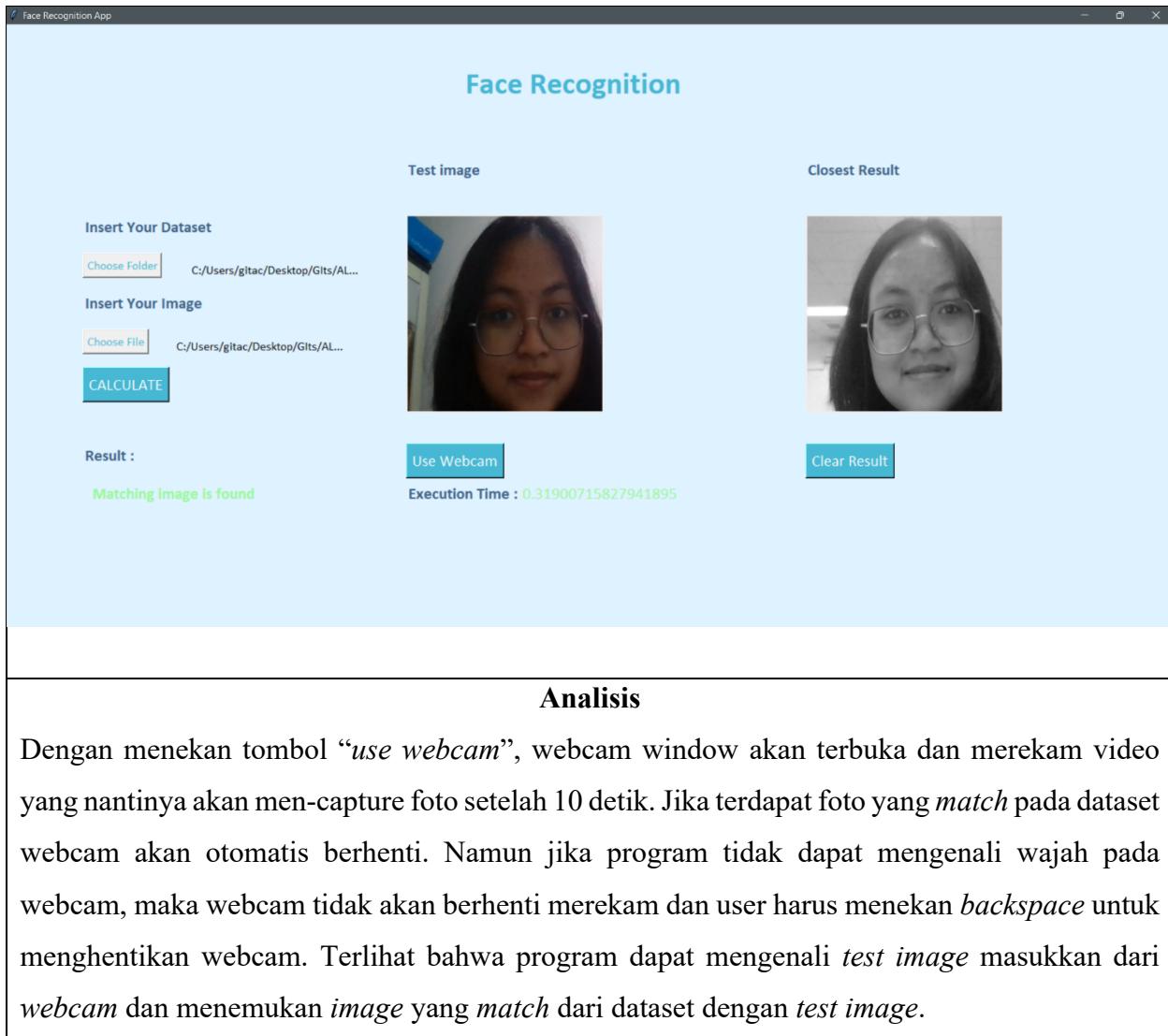
Clear Result

Analisis

Terlihat bahwa ketika image yang tidak pakai sedikit berbeda atau tidak sama persis dengan yang ada di dataset dan *euclidian distance* minimumnya melebihi threshold yang telah ditentukan, maka program akan mengembalikan pesan *couldn't find matching image*. Hal ini dapat terjadi karena pengaruh indikator lighting dari gambar yang menyebabkan *euclidian distance* semakin membesar.

Bonus

Ukuran Foto 352x288



Analisis

Dengan menekan tombol “use webcam”, webcam window akan terbuka dan merekam video yang nantinya akan men-capture foto setelah 10 detik. Jika terdapat foto yang *match* pada dataset webcam akan otomatis berhenti. Namun jika program tidak dapat mengenali wajah pada webcam, maka webcam tidak akan berhenti merekam dan user harus menekan *backspace* untuk menghentikan webcam. Terlihat bahwa program dapat mengenali *test image* masukkan dari *webcam* dan menemukan *image* yang *match* dari dataset dengan *test image*.

BAB 5

Kesimpulan, Saran, dan Refleksi

5.1 Kesimpulan

Terdapat berbagai teknik dan metode untuk pengenalan wajah, di antaranya beberapa teknik untuk memeriksa citra wajah dari kumpulan citra yang sudah diketahui yaitu jarak Euclidian dan *cosine similarity*, *principal component analysis* (PCA), serta eigenface. Pada tugas besar kali ini, digunakan bahasa pemrograman python dengan metode *eigenface* yaitu sekumpulan citra wajah yang ada (*dataset*) direpresentasikan dalam matriks *eigenface* dalam proses *training* atau proses *extract* dengan memanfaatkan library *openCV* pada python. Kemudian dilakukan hal yang sama pada *test image*, lalu dengan menggunakan perhitungan *eigenface* yang sudah didapatkan, dapat dihitung *euclidian distance* dari *test image* dan data citra wajah pada *dataset* untuk kemudian didapatkan gambar pada dataset yang memiliki *euclidian distance* paling kecil dengan *test image*, yaitu gambar yang dikenali program paling menyerupai *test image*. Hasil dari program ini kemudian dituangkan pada app dengan *graphical user interface* dari tkinter untuk penggunaan dan pengaksesan program yang lebih mudah.

5.2 Saran

Kami menyadari Tugas Besar 2 IF 2123 Aljabar Linier dan Geometri Semester 1 Tahun 2022/2023 yang kami kerjakan masih bisa dikembangkan lagi. Adapun catatan yang bisa dikembangkan tersebut adalah:

1. Sistem input webcam untuk program masih dapat dikembangkan dan divariasikan lagi. Seperti webcam tidak hanya berukuran 256x256 dan dapat men-*detect* secara *flexible* berbagai variasi ukuran wajah secara *live*.
2. Algoritma program dapat dibuat lebih efektif lagi agar dapat menerima lebih banyak masukan dataset dan melakukan *training* lebih cepat dan kedepannya program akan dapat digunakan untuk lebih banyak hal.
3. Sebaiknya perlu dilakukan penyempurnaan pada intensitas cahaya gambar, sehingga gambar dengan pencahayaan yang berbeda tetap dapat dikenali dengan baik.

5.3 Refleksi

Kami belajar dari Tugas Besar 2 IF 2123 Aljabar Linier dan Geometri Semester 1 Tahun 2022/2023 untuk bisa dalam manajemen waktu dan manajemen kemampuan masing-masing anggota kelompok agar masing-masing *task* dapat terselesaikan dengan baik. Kami juga mendapatkan pengetahuan-pengetahuan baru yang kemungkinan besar tidak akan kami dapatkan jika tidak mengerjakan tugas besar 2 ini, seperti menggunakan algoritma sendiri dalam mencari *eigenvalue* dan *eigenvector* juga mengerjakan *gui* menggunakan *library python*. Selain itu, kami belajar untuk tidak terlalu mendekati *deadline* dalam pengerjaannya karena kemungkinan-kemungkinan tertentu seperti jatuh sakit. Kami menyadari akan pentingnya mempertahankan kesehatan dan mengatur waktu dengan baik agar tugas besar yang kami kerjakan dapat selesai dengan maksimal.

Tautan Terkait

Link Repository: <https://bit.ly/Algeo02-21130>

Link Video Bonus: <https://youtu.be/HkCise15GCw>

Referensi

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

<https://realpython.com/python-gui-tkinter/>

<https://www.math.ucla.edu/~yanovsky/Teaching/Math151B/handouts/GramSchmidt.pdf>

<https://stats.stackexchange.com/questions/20643/finding-matrix-eigenvectors-using-qr-decomposition>

<http://www.ohiouniversityfaculty.com/youngt/IntNumMeth/lecture17.pdf>

https://en.wikipedia.org/wiki/Givens_rotation

<https://youtu.be/61NuFIK5VdU>