LAPORAN TUGAS BESAR 2

IF2123 ALJABAR LINIER DAN GEOMETRI

APLIKASI NILAI EIGEN DAN EIGENFACE PADA PENGENALAN WAJAH

(FACE RECOGNITION)

Disusun oleh:

Shelma Salsabila	13521115
Asyifa Nurul Shafira	13521125
Akhmad Setiawan	13521164

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

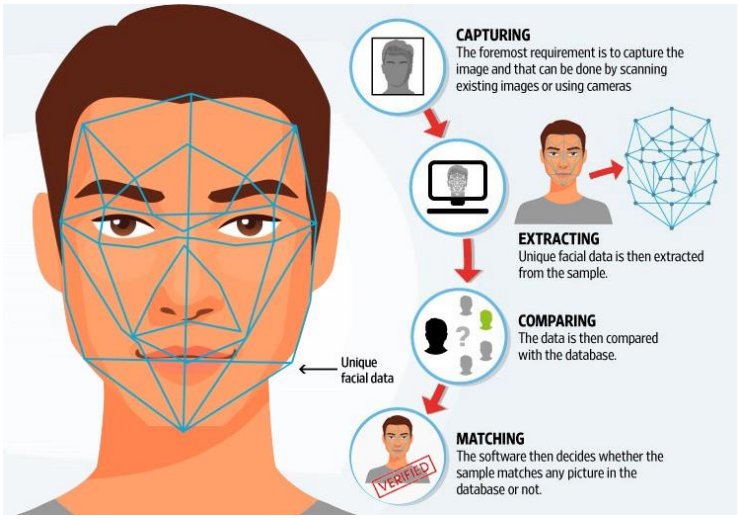
Institut Teknologi Bandung

2022

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. Alur proses sebuah sistem pengenalan wajah diperlihatkan pada Gambar 1.1



Gambar 1.1 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 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 langkah rinci dalam pembentukan eigenface:

1. Langkah pertama adalah menyiapkan data dengan membuat suatu himpunan S yang terdiri dari seluruh training image, (Γ1, Γ2, …, ΓM)

$$S = (\Gamma 1, \Gamma 2, \ldots, \Gamma M) \quad (1)$$

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) antara nilai training image (Γi) dengan nilai tengah (Ψ)

$$\phi i = \Gamma i - \Psi \quad (3)$$

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

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

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

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

$$C \text{ x } vi = \lambda i \text{ x } vi \quad (5)$$

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

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

$$l = 1, \ldots, M$$

Berikut langkah dalam tahapan pengenalan wajah:

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

$$\mu new = v \text{ x } \Gamma new - \Psi \quad (7)$$

$$\Omega = \mu 1 , \mu 2 , \ldots , \mu M$$

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

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

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) atau library pemrosesan gambar lainnya (contoh PIL). Fungsi untuk mengekstraksi fitur menggunakan fungsi ekstraksi yang sudah tersedia di dalam library. Fungsi Eigen harus diimplementasikan dan untuk operasi matriks lainnya dapat menggunakan library.

Input yang akan dimasukkan pengguna untuk eksekusi program:

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.

Kami membuat 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 dalam *library* atau Bahasa Python.
7. Input gambar akan berukuran minimal 256 x 256.
8. Hasil training dapat disimpan, namun tetap wajib dapat dilakukan training kembali jika diperlukan user.

BAB 2

Teori Singkat

2.1 Perkalian Matriks

Ada dua jenis perkalian pada matriks yaitu :

1) Perkalian Matriks dengan Skalar

Bila terdapat suatu skalar k dan matriks $\boldsymbol{Am \times n}$ dengan elemen aij maka $\boldsymbol{kA}$ adalah matriks yang berukuran $m \times n$ dengan elemen $kaij$ (Sembiring, 2003: 21). Berdasarkan definisi di atas, perkalian $\boldsymbol{kA}$ adalah sebuah matriks baru yang setiap elemennya merupakan perkalian antara suatu bilangan k dengan setiap elemen di $\boldsymbol{A}$. dan perkalian matriks dengan skalar ini bersifat komutatif yaitu $\boldsymbol{kA} = \boldsymbol{Ak}$

$$3\begin{bmatrix}2 & 1\\3 & 0\\5 & 5\end{bmatrix} = \begin{bmatrix}2.3 & 1.3\\3.3 & 0.3\\5.3 & 5.3\end{bmatrix} = \begin{bmatrix}6 & 3\\9 & 0\\15 & 15\end{bmatrix}$$

2) Perkalian Matriks dengan Matriks

Definisi (Howard Anton, 1987: 25):

Jika $\boldsymbol{A}$ adalah matriks $m \times r$ dan $\boldsymbol{B}$ adalah matriks $r \times n$, maka hasil kali $\boldsymbol{AB}$ adalah matriks $m \times n$ yang entri-entrinya ditentukan sebagai berikut: untuk mencari entri dalam baris i dan kolom j dari $\boldsymbol{AB}$ pilihlah baris i dari matriks A dan kolom j pada matriks $\boldsymbol{B}$. Kalikanlah entri-entri yang bersesuaian dari baris dan kolom tersebut bersama-sama dan kemudian tambahkanlah hasil kali yang dihasilkan.

Perkalian matriks dengan matriks hanya dapat dioperasikan jika banyaknya kolom dari matriks pertama sama dengan banyaknya baris pada matriks kedua, jika syarat tersebut tidak terpenuhi, maka hasil kali tidak dapat didefinisikan. Perkalian matriks dengan matriks ini tidak bersifat komutatif atau $\boldsymbol{AB} \neq \boldsymbol{BA}$.

$$\boldsymbol{A} = \begin{bmatrix} 2 & 1 \\ 3 & 0 \\ 5 & 5 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 2 & 2 \end{bmatrix}$$

$$\boldsymbol{AB} = \begin{bmatrix} (2.1)+(1.0) & (2.0)+(1.2) & (2.3)+(1.2) \\ (3.1)+(0.0) & (3.0)+(0.2) & (3.3)+(0.2) \\ (5.1)+(5.0) & (5.0)+(5.2) & (5.3)+(5.2) \end{bmatrix}$$

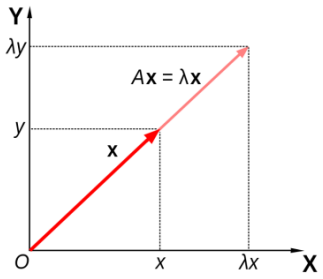
$$= \begin{bmatrix} 2 & 2 & 10 \\ 3 & 0 & 9 \\ 5 & 10 & 25 \end{bmatrix}$$

2.2 Nilai dan Vektor Eigen

Jika A adalah matriks n x n maka vektor tidak-nol x di Rn disebut **vektor eigen** dari A jika Ax sama dengan perkalian suatu skalar l dengan x, yaitu

Ax = lx

Skalar l disebut **nilai eigen** dari A, dan x dinamakan vektor eigen yang berkoresponden dengan l. Nilai eigen menyatakan nilai karakteristik dari sebuah matriks yang berukuran n x n. Vektor eigen x menyatakan matriks kolom yang apabila dikalikan dengan sebuah matriks n x n menghasilkan vektor lain yang merupakan kelipatan vektor itu sendiri.



Sumber gambar: Wikipedia

Dengan kata lain, operasi Ax = lx menyebabkan vektor x menyusut atau memanjang dengan faktor l dengan arah yang sama jika l positif dan arah berkebalikan jika l negatif.

Jika diberikan sebuah matriks A berukuran n x n. Vektor eigen dan nilai eigen dari matriks A dihitung sebagai berikut:

Ax = lx

IAx = lIx (kalikan kedua ruas dengan I = matriks identitas)

$$\mathbf{Ax = lIx}$$

$$\mathbf{(lI – A)x = 0}$$

x = 0 adalah solusi trivial dari (lI – A)x = 0

Agar (lI – A)x = 0 memiliki solusi tidak-nol, maka haruslah

$$\mathbf{det(lI – A) = 0}$$

Persamaan det(lI – A) = 0 disebut persamaan karakteristik dari matriks A, dan akar-akar persamaan tersebut, yaitu l, dinamakan akar-akar karateristik atau nilai-nilai eigen.

2.3 EigenFace

Eigenface adalah algoritma pegenalan wajah yang berdasarkan pada Principle Component Analysis (PCA). Dengan kata lain, eigenfaces menggunakan perhitungan PCA sebagai pendekatannya. Eigenfaces dapat ditentukan dengan prinsip-prinsip analisis komponen terhadap sekumpulan citra sampel dengan wajah terpusat dan terutama dengan ukuran yang setara. Pola wajah di dalam suatu citra dapat ditentukan dengan memindahkan frame pada subcitra yang menutupi atau membatasi keseluruhan pola pada citra ke lokasi lain yang berukuran sama.

Menurut layman (Al Fatta, 2009) eigenfaces adalah sekumpulan unsur wajah yang dibuat standart yang diambil dari analisis statistik dari banyak gambar wajah. Algoritma eigenface secara keseluruhan cukup sederhana.Training Image dipresentasikan dalam sebuah vector flat (gabungan vector) dan digabung bersama-sama menjadi matriks tunggal. Eigenfaces dari masing-masing citra kemudian diekstraksi dan kemudian disimpan di dalam dataset. Test Image yang masuk didefinisikan juga nilai eigenfaces-nya dan dibandingkan dengan eigenfaces dari image yang telah tersimpan di dalam dataset (Prasetyo & Rachmatun, 2008). Eigenfaces ini dianggap sebagai sebuah sederetan ciri yang bersama-sama memberi karakter variasi diantara citra-citra wajah. Setiap titik citra wajah bisa dinyatakan dalam satu atau lebih eigenvector sehingga sekumpulan eigenvector dapat ditampilkan sebagai sekumpulan wajah. Sekumpulan eigenvector yang digunakan inilah yang disebut sebagai eigenfaces.

BAB 3

Implementasi Program

3.1 Mengekstrasi Fitur-Fitur dari Foto Training

Setiap input gambar dan dataset yang dimiliki akan diekstrak menjadi matriks untuk kemudian dilakukan pengolahan. Foto-foto yang mulanya berwarna, akan diubah menjadi *grayscale* terlebih dahulu. Untuk proses ekstraksi yang dilakukan, kami menggunakan library OpenCV (Open Source Computer Vision). Library ini dipilih karena sangat cocok digunakan untuk program yang memerlukan proses pengolahan citra gambar. Secara singkat, proses ekstraksi yang dilakukan sebagai berikut.

INPUT FOTO → OPENCV → GRAYSCALE → MATRIKS

3.2 Mencocokkan Fitur-Fitur Sebuah Gambar dengan Dataset

Setelah melakukan proses ekstraksi, kami melakukan pencocokkan fitur-fitur terhadap gambar yang dimasukkan ke dalam program melalui aplikasi GUI sederhana. Pencocokkan dilakukan dengan menggunakan fungsi-fungsi untuk mencocokkan fitur-fitur sebuah gambar dengan dataset yang keduanya telah disimpan dalam tipe gambar (jpg, png, dll).

3.3 *Graphical User Interface (GUI)*

Program kami memanfaatkan library python yakni Tkinter untuk membuat aplikasi sederhana berbasis GUI dalam pencocokkan wajah dengan dataset yang ada. Penggunaan library ini didasarkan dengan pertimbangan kemudahan karena Tkinter sudah umum menjadi library standar GUI dari python. Tkinter menyediakan beberapa fitur yang kami manfaatkan, seperti fitur *entry input* untuk masukan dari pengguna, *button* sebagai media interaktif yang menandakan program dimulai, *canvas* untuk menampilkan gambar, serta *text* dan *message* untuk media komunikasi dengan pengguna terhadap respons input yang dimasukkan.

Entry input digunakan untuk kedua masukan, baik untuk *test image* maupun folder dataset. Pengguna dapat memasukkan *directory* tempat file/folder berada ataupun langsung meng-klik ikon folder yang ada di sebelahnya. Dengan meng-*import class* filedialog pada Tkinter, *button* ikon folder ini akan otomatis menampilkan directory dari file/folder yang dipilih pada *entry input*. Khusus untuk masukan *test image*, pengguna dapat memilih menggunakan *webcam* untuk meng-input *test image*. *Auto capture* ini menggunakan library OpenCV, dan akan meng-*capture* secara otomatis dalam waktu 5 detik. Foto hasil tangkapan akan disimpan dalam folder

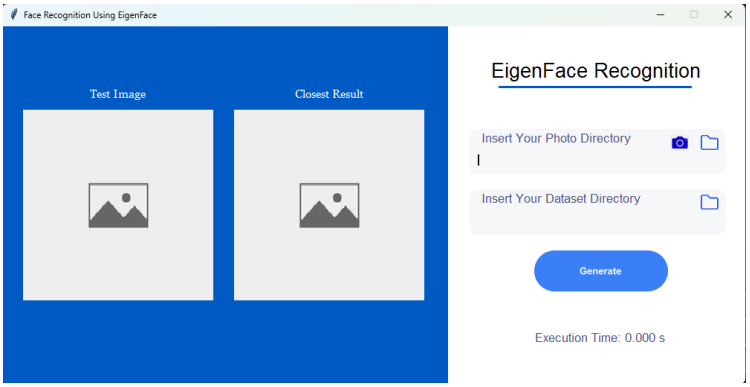
dataset_test dalam file bernama test.jpg. Setelah selesai meng-*capture* secara otomatis kamera akan tertutup dan *entry input* akan langsung terisi mengarah ke *directory* dataset_test\test.jpg, untuk dapat langsung digunakan sebagai *test image*.

Ketika pengguna telah memasukkan input *test image* dan folder dataset, pengguna dapat meng-klik *button generate* untuk kemudian langsung mengeksekusi proses pencocokan gambar pada dataset. Jika berhasil dikenali, foto *test image* dan foto hasil pengenalan akan dimunculkan keduanya, dan muncul pemberitahuan ”*Image Found*” dan *execution time* akan ikut ditampilkan. Namun jika tidak ditemukan, yang dimunculkan hanya foto *test image* saja dan muncul pemberitahuan ”*Not Found Any*” tanpa ada *execution time* yang tercatat.

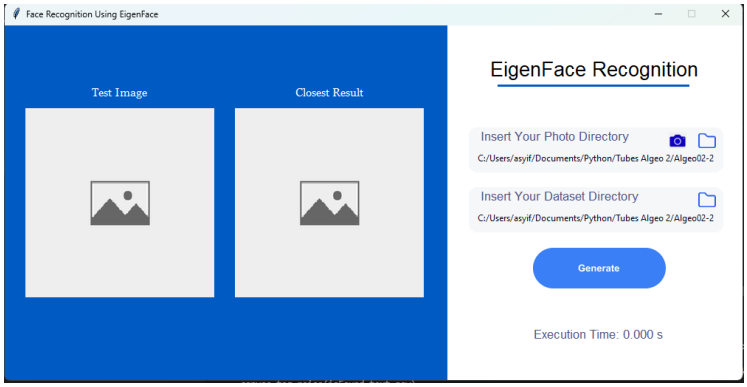
BAB 4

Eksperimen

4.1 Tampilan Awal



Gambar 4.1.1 Tangkapan Layar GUI Sebelum Memasukkan Input Foto dan Dataset



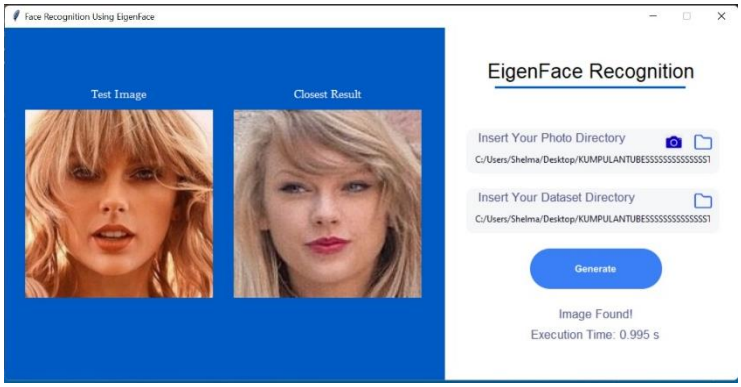
Gambar 4.1.2 Tangkapan Layar GUI Setelah Memasukkan Input Foto dan Dataset

Tampilan awal memperlihatkan panel sebelum menerima input yang terdiri atas judul program, *entry input* untuk test image dan dataset, tombol kamera, tombol "Generate", *execution time*, dan dua slot gambar. Untuk memasukkan file test image dapat dilakukan dengan menekan icon folder ataupun kamera yang akan mengambil gambar secara real time. Ketika input file test image dan dataset sudah dipilih maka *entry input* akan berisi alamat tempat file/folder gambar berada.

4.2 Tampilan Hasil Pencocokan Wajah

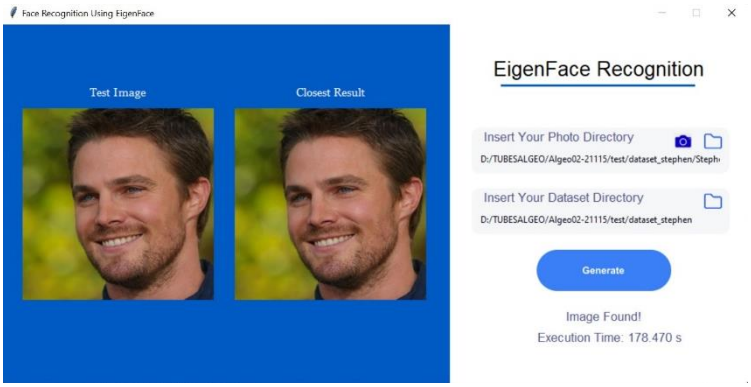
Setelah *user* melakukan input, selanjutnya program akan memulai proses pengolahan gambar. Durasi yang dibutuhkan program dalam mengolah gambar bergantung pada banyak gambar yang terdapat dalam dataset. Semakin sedikit gambar dataset maka durasi yang

dibutuhkan akan semakin sedikit karena proses pencocokkan yang dilakukan akan lebih sedikit dibandingkan dengan dataset yang memiliki lebih banyak gambar. Setelah proses pencocokkan selesai hasil program akan muncul sebagai berikut.



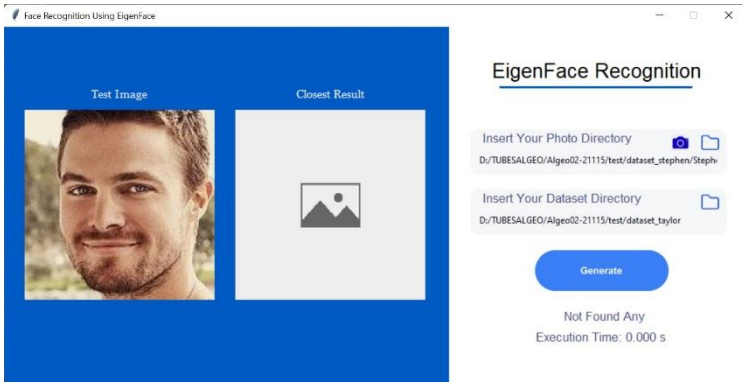
Gambar 4.2.1 Tangkapan Layar GUI Hasil Pencocokkan dengan Metode EigenFace dan 14 gambar dataset

Pada **Gambar 4.2.1** ditampilkan hasil kasus pencocokkan *test image* yang pertama dengan dataset yang berisi 14 gambar training. Terdapat gambar yang memiliki jarak Euclidean paling dekat dengan *test image* sehingga program menampilkan keterangan “Image Found!” dan execution time yang berlangsung selama 0.995 detik.



Gambar 4.2.2 Tangkapan Layar GUI Hasil Pencocokkan dengan Metode EigenFace dan 159 gambar dataset

Pada **Gambar 4.2.2** ditampilkan hasil kasus pencocokkan *test image* dengan dataset yang jauh lebih banyak dibandingkan dengan kasus pertama, yaitu berisi 159 gambar training. Pada kasus ini juga terdapat gambar yang memiliki jarak Euclidean paling dekat dengan *test image* sehingga program menampilkan keterangan “Image Found!”, tapi dengan execution time yang lebih lama, yaitu selama 178.470 detik.



Gambar 4.2.3 Tangkapan Layar GUI Hasil Pencocokkan Jika Tidak Ditemukan Gambar yang Mendekati

Pada **Gambar 4.2.3** ditampilkan hasil kasus pencocokkan *test image* dengan dataset dimana tidak ditemukan gambar yang memiliki jarak Euclidean paling dekat dengan *test image* sehingga program menampilkan keterangan “Not Found Any” dan tidak memiliki execution time, yaitu 0.000 detik.

BAB 5

Kesimpulan, Saran, dan Refleksi

5.1 Kesimpulan

Pada tugas besar kali ini, kami telah membuat sebuah program pengenalan wajah dengan dataset yang telah ditentukan sebelumnya. Fitur-fitur dalam program kami antara lain, insert test image dan dataset, execution time, serta tampilan gambar test image dan hasil pencocokan. Hasil implementasi lengkap program dapat diakses melalui pranala yang tertera pada halaman terakhir laporan.

5.2 Saran

Masalah utama dalam program pengenalan wajah ini adalah efisiensi waktu karena algoritma yang dibuat dapat menjadikan program berjalan sangat lambat. Oleh karena itu, diharapkan untuk pengembangan selanjutnya dapat dilakukan optimalisasi algoritma tersebut sehingga efisiensi waktunya lebih cepat.

5.3 Refleksi

Setelah menyelesaikan tugas besar ini, kami sadar bahwa tugas ini sangat penting dan bermanfaat untuk dipelajari karena saat ini penggunaan face recognition sebagai fitur keamanan dan keperluan lainnya sudah banyak digunakan di dunia, khususnya Indonesia. Semoga tugas besar ini dapat membantu kami dan juga pembaca agar dapat berkembang menjadi lebih baik lagi. Dari eksperimen diatas juga dapat disimpulkan bahwa program yang kami kerjakan krang lebihnya dapat berjalan dengan baik. Kami juga dapat merasa bahwa dengan menggunakan bahasa Python dan OpenCV tugas besar ini menjadi lebih mudah dikerjakan.

Daftar Referensi

Howard Anton & Chris Rores, Elementary Linear Algebra, 10th Edition

https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-Nilai-Eigen-dan-Vektor-Eigen-Bagian1.pdf di akses pada 21 November 2022

http://eprints.uny.ac.id/35053/2/Bab%20II.pdf# di akses pada 21 November 2022

https://eprints.umm.ac.id/35652/3/jiptummpp-gdl-mochalimas-47096-3-bab2.pdf di akses pada 21 November 2022

Lampiran

Link Repository:

https://github.com/blixa-rd/Algeo02-21115

Link Youtube:

https://youtu.be/Gdkro_oxwEs