IF2123 - Aljabar Linier dan Geometri
Laporan Tugas Besar 2

Nama	NIM
Hobert Anthony Jonatan	13521079
Tabitha Permalla	13521111
Nicholas Liem	13521135

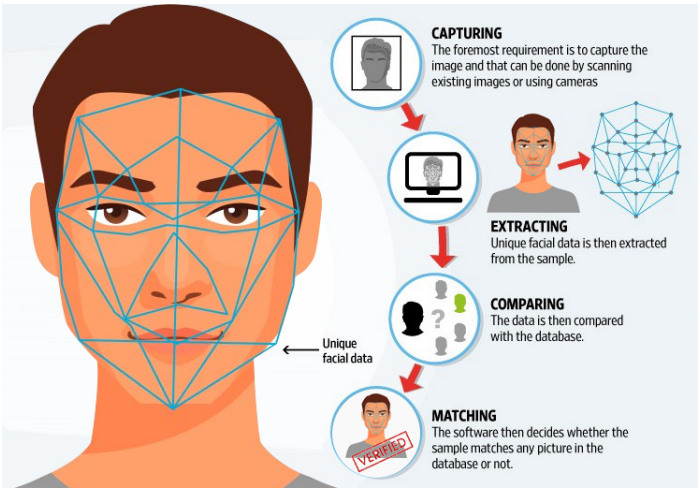
Institut Teknologi Bandung
Sekolah Teknik Elektro dan Informatika

Tahun Ajaran 2022/2023

Daftar Isi

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 basis data 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 Sistem Pengenalan Wajah

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 merupakan langkah rinci dalam pembentukan *eigenface*.

2 Teori Singkat

2.1 Perkalian Matriks

Perkalian dua matriks A dan B (AB) adalah operasi perkalian setiap elemen baris dari dari A dengan setiap elemen kolom dari B. Oleh sebab itu, terdapat suatu restriksi khusus dalam operasi perkalian dua matriks, yakni jika A adalah matriks yang berukuran $M \times N$, maka operasi perkalian matriks bisa dilakukan jika dan hanya jika matriks B memiliki ukuran $N \times R$, dengan R adalah bilangan bulat.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{bmatrix}$$

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in} + b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj}$$

Selain itu, hal yang menarik lagi adalah, jika A adalah matriks berukuran $M \times N$ dan B adalah matriks berukuran $N \times R$, maka jika C adalah hasil perkalian matriks A dan B maka C memiliki ukuran $M \times R$.

2.2 Nilai, Vektor, dan Ruang Eigen

Kata "eigen" berasal dari bahasa Jerman yang berarti asli atau karakteristik. Jika A adalah matriks persegi berukuran $N \times N$, maka vektor tidak nol x di R^n disebut vektor eigen dari A jika persamaan di bawah ini terpenuhi.

$$Ax = \lambda x$$

Arti dari persamaan di atas adalah perkalian suatu matriks A dengan x bernilai sama seperti matriks A dikalikan dengan sebuah skalar. λ pada persamaan di atas disebut sebagai nilai eigen dari A dan x dinamakan vektor eigen yang berkoresponden dengan λ.

Perhitungan nilai eigen dan vektor eigen yang berkorespondensi dengan matriks A didapatkan melalui persamaan di bawah ini.

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

Perhatikan bahwa $x = 0$ adalah solusi trivial dari $(\lambda I - A)x = 0$, agar solusinya tidak-nol maka haruslah $det(A - \lambda I) = 0$. Persamaan ini disebut juga sebagai persamaan karakteristik dan λ disebut sebagai akar-akar karakteristik atau nilai-nilai eigen.

2.3 Proses Pengenalan Wajah

2.3.1 Pembentukan Dataset

Gambar yang diterima memiliki banyak komponen yang memiliki banyak variasi, misalnya sebuah gambar bisa memiliki banyak warna dan bentuk pixel yang berbeda-beda ukurannya. Oleh sebab itu, supaya gambar bisa diproses dengan seragam, gambar akan dibentuk ulang dalam ukuran $N \times N$ pixel dan akan diubah setiap gambar tersebut dari RGB menjadi *Grayscale*.

Gambar-gambar yang sekarang berukuran $N \times N$ pixel dimensinya diubah menjadi $N^2 \times 1$. Setiap gambar ini akan disusun sebagai sebuah list teratur (Γ).

$$\Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_M\}$$

2.3.2 Pencarian Muka Rata-Rata (*Mean Face*)

Muka rata-rata (Ψ) dapat dihitung dengan menggunakan rumus di bawah ini.

$$\Psi = \frac{1}{M} \sum_{n=1}^{M} \Gamma_n$$

2.3.3 Pembentukan Matriks Kovarian

Pertama-tama kita akan mendefinisikan (Φ_i) sebagai gambar pada dataset yang dikurangi dengan muka rata-rata, proses ini dapat disebut sebagai normalisasi gambar.

$$\Phi_i = \Gamma_i - \Psi$$

Kemudian kita akan susun matriks kovarian (C) dari data ini dengan rumus sebagai berikut.

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

Matriks A didefinisikan sebagai hasil konkatenasi semua hasil gambar yang telah dinormalisasi.

$$A = \{\Phi_1 \Phi_2 \ldots \Phi_M\}$$

Terbentuklah matriks kovarian yang diminta.

2.3.4 Pencarian Vektor-Vektor Eigen

Matriks kovarian yang sebelumnya didapat itu berukuran $N^2 \times N^2$ sehingga akan terdapat N^2 buah vektor eigen yang harus dicari. Hal ini secara komputasi akan sangat memakan waktu dan tidak efisien. Oleh sebab itu, kita akan menggunakan cara lain untuk menghitung vektor-vektor eigen yang diminta.

Perhatikan bahwa M adalah jumlah gambar pada dataset dan perhatikan bahwa $M \ll N^2$. Kita dapat membentuk matriks kovarian ulang sehingga hanya sebanyak

$M - 1$ buah vektor eigen yang perlu dicari. Hal ini akan sangat jauh lebih cepat pemrosesannya dibanding harus menghitung sebanyak N^2 buah vektor eigen. Cara kita mendapatkan matriks yang berukuran $M \times M$ dan berkorespondensi dengan dataset adalah dengan menghitung $A^T A$. Matriks ini akan memiliki nilai eigen yang sama dengan matriks yang berukuran $N^2 \times N^2$. Pembuktiannya sebagai berikut.

$$\begin{aligned} A^T A V_i &= \mu_i V_i \\ A A^T A V_i &= A \mu_i V_i \\ C A V_i &= \mu_i A V_i \end{aligned}$$

Dengan demikian kita dapat membentuk matriks $M \times M$ dengan $L = A^T A$ di mana $L_{mn} = \Phi_m^T \Phi_n$ dan bisa dicari sebanyak M jumlah vektor eigen, v_l, dari L.

Perhitungan untuk memperoleh vektor eigen dapat dilakukan menggunakan beberapa metode. Salah satunya adalah dengan memanfaatkan dekomposisi QR. Dari dekomposisi QR ini kita akan memperoleh matriks Q dan matriks R. Apabila kita secara berulang melakukan dekomposisi QR terhadap hasil perkalian matriks dari matriks R dengan matriks Q, pada satu waktu, akan terbentuk suatu matriks yang diagonalnya berisi nilai-nilai eigen dari matriks awal. Misal, matriks diagonal ini diperoleh pada iterasi ke-n, maka hasil perkalian $Q_1 Q_2 ... Q_{n-1} Q_n$ akan menghasilkan matriks vektor eigen.

Dekomposisi QR sendiri dapat dilakukan dengan proses $Gram - Schmidt$. Misal matriks $A = [a_1 ... a_n]$. Maka matriks $Q = [e_1 ... e_n]$, di mana:

$$\begin{aligned} u_1 &= a_1 & e_1 &= \frac{u_1}{||u_1||} \\ u_2 &= a_2 - proj_{u_1} a_2 & e_2 &= \frac{u_2}{||u_2||} \\ u_3 &= a_3 - proj_{u_1} a_3 - proj_{u_2} a_3 & e_3 &= \frac{u_3}{||u_3||} \\ &\vdots & &\vdots \\ u_k &= a_k - \sum_{j=1}^{k-1} proj_{u_j} a_k & e_k &= \frac{u_k}{||u_k||} \end{aligned}$$

Sedangkan matriks R adalah sebagai berikut:

$$\mathrm{R} = \begin{bmatrix} < e_1, a_1 > & < e_1, a_2 > & \cdots & < e_1, a_n > \\ 0 & < e_2, a_2 > & \cdots & < e_2, a_n > \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & < e_n, a_n > \end{bmatrix}$$

2.3.5 Pencarian Muka Eigen

Kita dapat mencari muka eigen (μ_l) dengan rumus di bawah ini.

$$\mu_l = \sum_{k=1}^{M} v_{lk}\Phi_k,\ l = 1, \ldots, M$$

2.3.6 Proyeksi Gambar Dataset ke Ruang Muka

Kemudian, kita perlu memproyeksikan setiap gambar yang telah dinormalisasi ke dataset. Dengan kata lain, kita dapat menyatakan suatu gambar pada dataset sebagai penjumlahan dari muka rata-rata dengan kombinasi linear ruang muka (muka eigen).

$$\Gamma_i = \Psi + \begin{bmatrix}\mu_1, & \mu_2, & \ldots, & \mu_K\end{bmatrix} \begin{bmatrix}\omega_1 \\ \omega_2 \\ \vdots \\ \omega_K\end{bmatrix}$$

Nilai-nilai berat (ω_k) dapat dicari dengan menggunakan rumus $\omega_k = \mu_k^T \Phi_i$. Kita juga dapat menulis ulang persamaan di atas menjadi di bawah ini.

$$\Gamma_i = \Psi + \sum_{k=1}^{M} \mu_k \omega_k$$

2.3.7 Penyimpanan Nilai Berat pada Basis Data

Nilai-nilai berat yang didapatkan tadi (dalam bentuk matriks kolom) dapat dilambangkan sebagai Ω.

$$\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$$

Nilai Ω disimpan pada suatu list atau penampung lain, anggap penampung ini adalah Δ (basis data).

$$\Delta = \{\Omega_1, \Omega_2, \ldots, \Omega_M\}$$

2.4 Proses Pencocokan Wajah dan Jarak Euclidean

Pada saat memproses suatu wajah uji, kita akan lakukan normalisasi terlebih dahulu terhadap data uji.

$$\Gamma_{new} = \Gamma_{uji} - \Psi$$

Kemudian kita akan proyeksikan gambar uji yang telah dinormalisasi (Γ_{new}) ke ruang muka pada basis data. Artinya, sekarang kita akan mencari nilai-nilai berat atau koefisien-koefisien dari kombinasi linear dari Γ_{new} pada ruang muka basis data. Lakukan cara yang sama seperti pada bagian 2.3.6.

Sekarang kita memperoleh nilai Ω_{uji}, sekarang kita akan menggunakan jarak Euclidean untuk menentukan jarak dari gambar uji ke setiap gambar pada basis data.

$$\|\Omega_{uji} - \Omega_K\| = \sum_{i=1}^{M} (\omega_i^n - \omega_i^k)^2$$

Carilah nilai jarak Euclidean paling kecil pada semua gambar pada basis data. Jarak paling kecil memiliki arti bahwa gambar tersebut memiliki kecocokan paling tinggi di antara semua gambar pada dataset sehingga dengan ini diperolehlah hasil akhir gambar yang paling mirip pada basis data dengan gambar uji.

3 Implementasi Program

3.1 Teknologi dan Stack

Pada pengaplikasiannya, program ini dijalankan menggunakan bahasa pemrograman Python versi 3.10.6 dan program ini juga menggunakan beberapa *tech* atau *stack* lain yang mendukung pemrosesan gambar, matriks, dan GUI secara umum. *Tech* tersebut merupakan suatu *library* dalam Python. Secara spesifik, program ini menggunakan OpenCV, Numpy, dan Tkinter.

3.1.1 OpenCV

OpenCV adalah singkatan dari *Open Source Computer Vision Library*. Berdasarkan terminologinya, pustaka ini merupakan suatu pustaka terbuka yang menopang pemrosesan gambar. Dari laman OpenCV, pustaka ini ditujukan untuk memfasilitasi sebuah infrastruktur untuk aplikasi *computer vision* dan melakukan akselerasi dalam persepsi mesin dalam produk komersial.

3.1.2 NumPy

NumPy adalah sebuah pustaka Python yang ditujukan untuk komputasi saintifik. Dari laman NumPy, disebutkan bahwa NumPy adalah suatu pustaka yang memiliki kelebihan unik di mana data yang multidimensional dapat diproses. Selain itu, pustaka ini juga memfasilitasi pemrosesan matriks, manipulasi matiks, operasi statistik dasar, dan sebagainya.

3.1.3 Tkinter

Dari laman Tkinter, Tkinter merupakan pustaka Python yang berfungsi sebagai *interface* program terhadap *Tcl/Tk GUI toolkit*.

3.2 Garis Beras Algoritma Program Pengenalan Wajah

3.2.1 Deskripsi Umum

Pada umumnya, pemrosesan pengenalan wajah bergantung pada komponen-komponen atau fitur unik dalam wajah manusia. Beberapa di antaranya adalah mata, hidung, bibir, rambut, dan sebagainya. Namun, pada pemrosesan wajah menggunakan *eigenface*, kita akan mengektrasi informasi dari berbagai variasi gambar dan menggunakan informasi ini untuk dicocokan dengan gambar lain.

Dari berbagai variasi gambar ini akan dicari sebuah komponen *principal* dari distribusi gambar-gambar pada basis data. Oleh sebab itu, kita akan menggunakan metode *principal component analysis* untuk mencari koleksi vektor eigen yang menunjukkan komponen penting dari sebuah gambar muka. Kumpulan dari vektor-vektor eigen ini akan disebut sebagai *eigenface*.

Kemudian, setiap gambar pada basis data yang telah menjalani proses *principal component analysis* dapat ditulis ulang sebagai kombinasi linear dari *eigenface* tersebut. Nilai-nilai koefisien (*weights*) pada kombinasi linear tersebut akan disimpan sebagai data penting untuk proses pencocokan wajah. Kita akan menggunakan *euclidean distance* atau jarak Euclidean untuk mencari sebarapa mirip suatu gambar uji dengan gambar-gambar yang ada di basis data.

3.2.2 Algoritma Dasar Pengenalan dan Pencocokan Wajah

1. Menyiapkan gambar-gambar wajah pada basis data.

2. Cari *eigenface* dari basis data atau *training set*, kemudian ambil hanya M banyak gambar yang signifikan dari *eigenface* yang berkorespondensi dengan nilai eigen tertinggi. Koleksi *eigenface* ini disebut sebagai ruang muka atau *face space*.

3. Proyeksikan semua gambar pada basis data dengan ruang muka yang didapatkan pada (2) kemudian akan didapatkan nilai-nilai berat atau *weights* kemudian jadikan nilai-nilai ini ke dalam sebuah list dan masukkan list tersebut ke dalam basis data.

4. Proses pencocokan wajah dimulai ketika wajah uji yang dimasukkan akan diproyeksikan terhadap ruang muka yang sama pada basis data dan akan diperoleh nilai-nilai berat dari wajah uji.

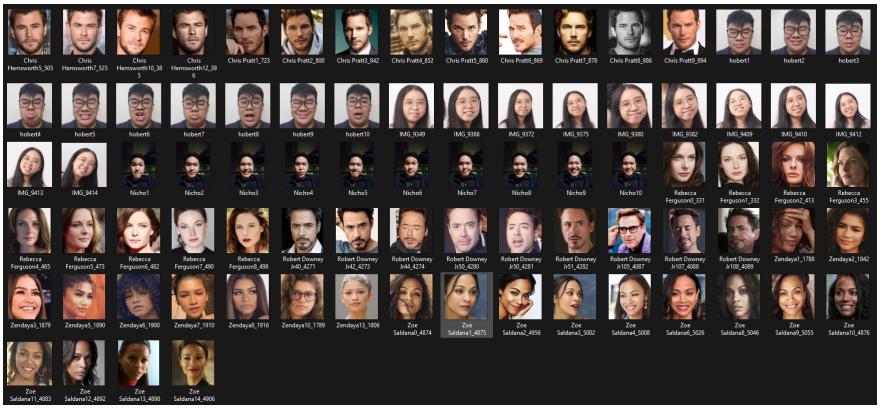
5. Nilai-nilai berat dari wajah uji akan dibandingkan dengan nilai-nilai berat yang ada di basis data dan dicari jarak Euclidean terpendek dari semua set data. Gambar pada basis data dengan jarak Euclidean terpendek dengan gambar wajah uji dapat dikatakan cocok dengan wajah uji.

Algoritma-algoritma ini telah dijelaskan dengan lebih detail pada bab-bab sebelumnya.

4 Eksperimen

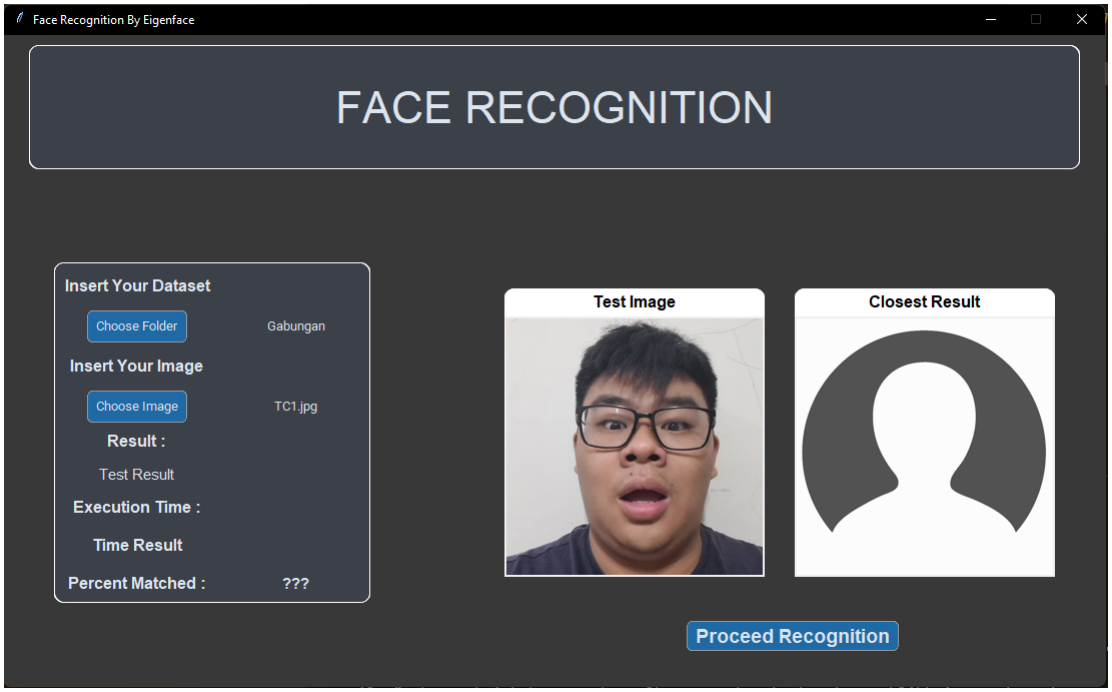
4.1 Dataset

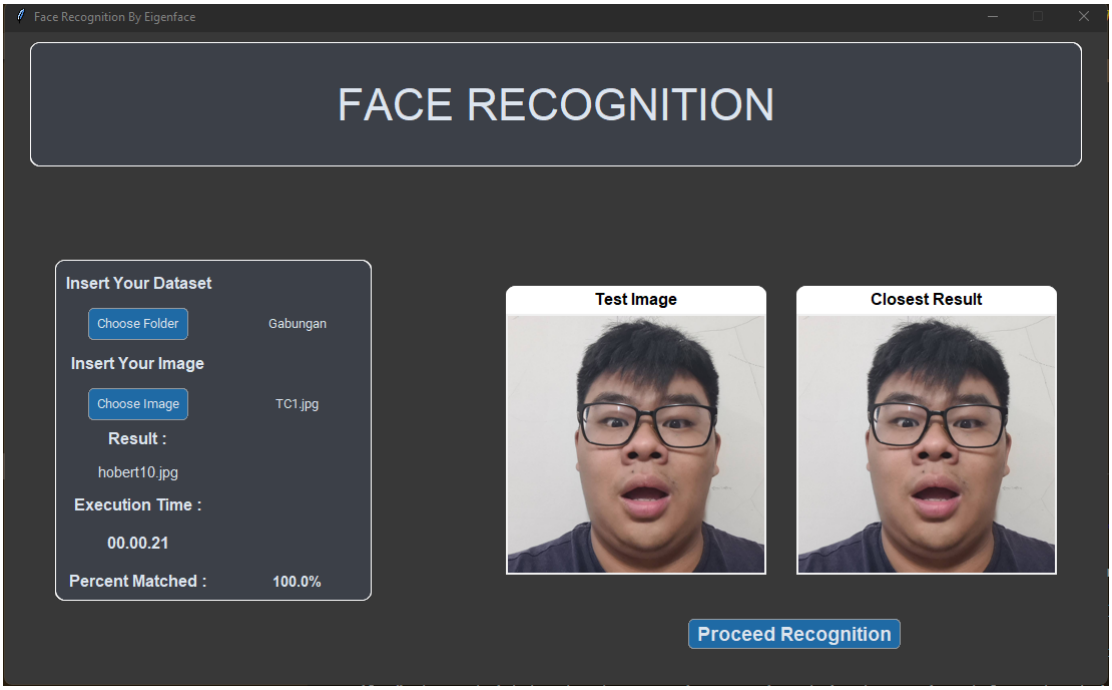
Data set yang disediakan untuk bagian eksperimen ini berjumlah 84 buah. Terdiri dari muka artis (Chris Hemsworth, Chris Pratt, Rebecca Ferguson, Zendaya, Zoe Saldana), dan muka kelompok kami (Hobert, Tabitha, Nicholas).



4.2 Percobaan - Gambar yang Sama

Pada kasus ini, kami mengambil satu gambar dari basis data kemudian menjadikannya gambar uji sehingga gambarnya akan sama persis. Hipotesis awalnya ialah gambar akan menghasilkan gambar yang sama dan mengeluarkan persen match sebesar 100 persen.

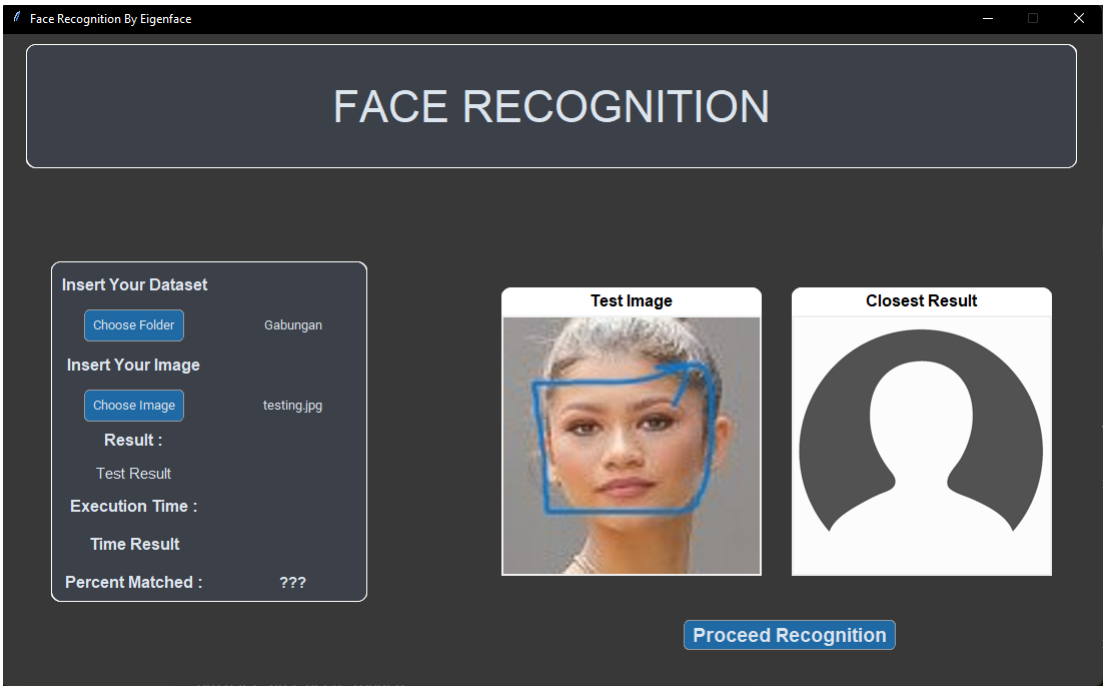


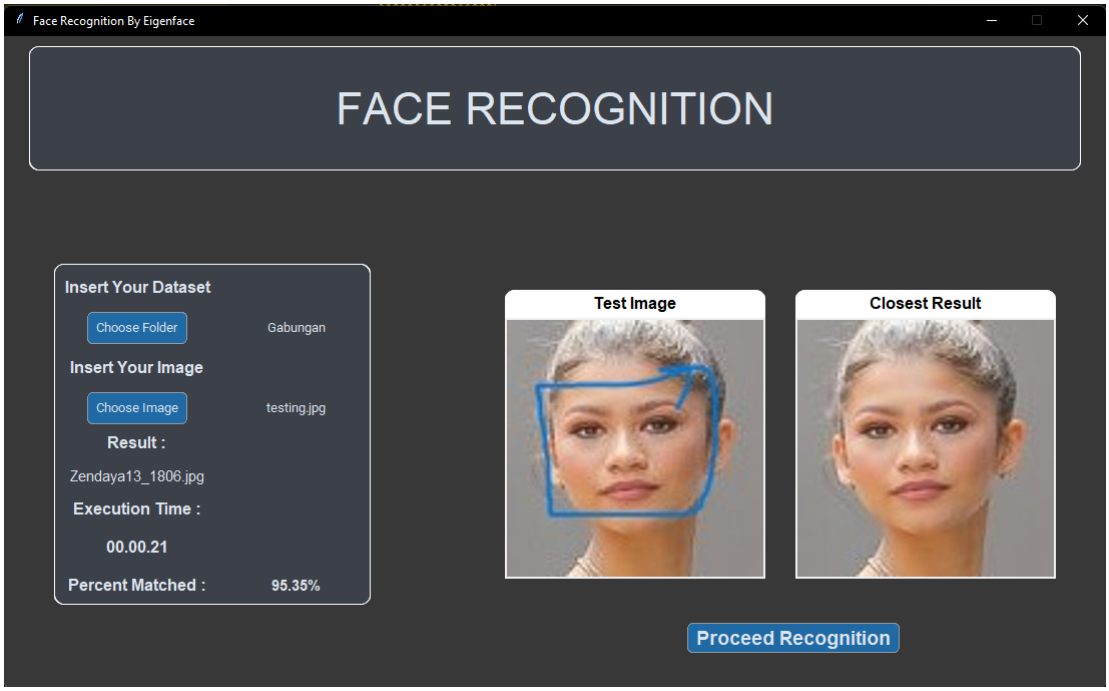


Sesuai ekspektasi diawal, gambar yang dihasilkan sama dan memiliki tingkat kesamaan sebesar 100 persen.

4.3 Percobaan - Gambar yang Sama Tetapi Dicoret

Pada kasus ini, kami mengambil satu gambar dari basis data kemudian mencoret-coret gambar aslinya dan menjadikannya gambar uji. Hipotesis awalnya ialah gambar akan menghasilkan gambar yang sama dengan original dan mengeluarkan persen match yang cukup tinggi.

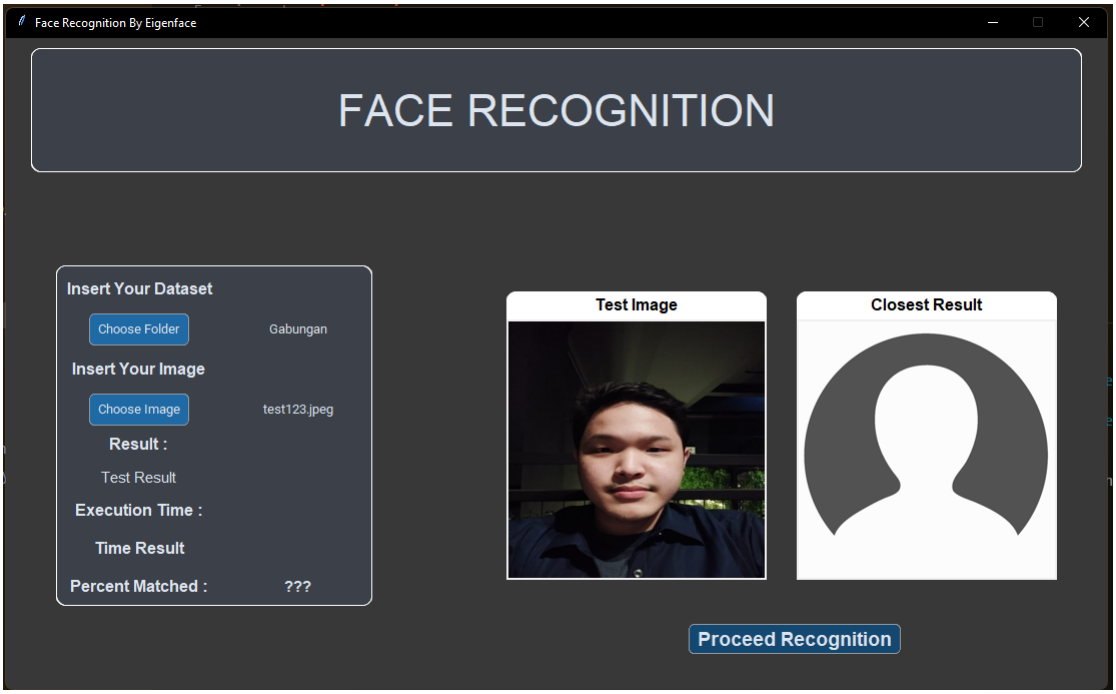


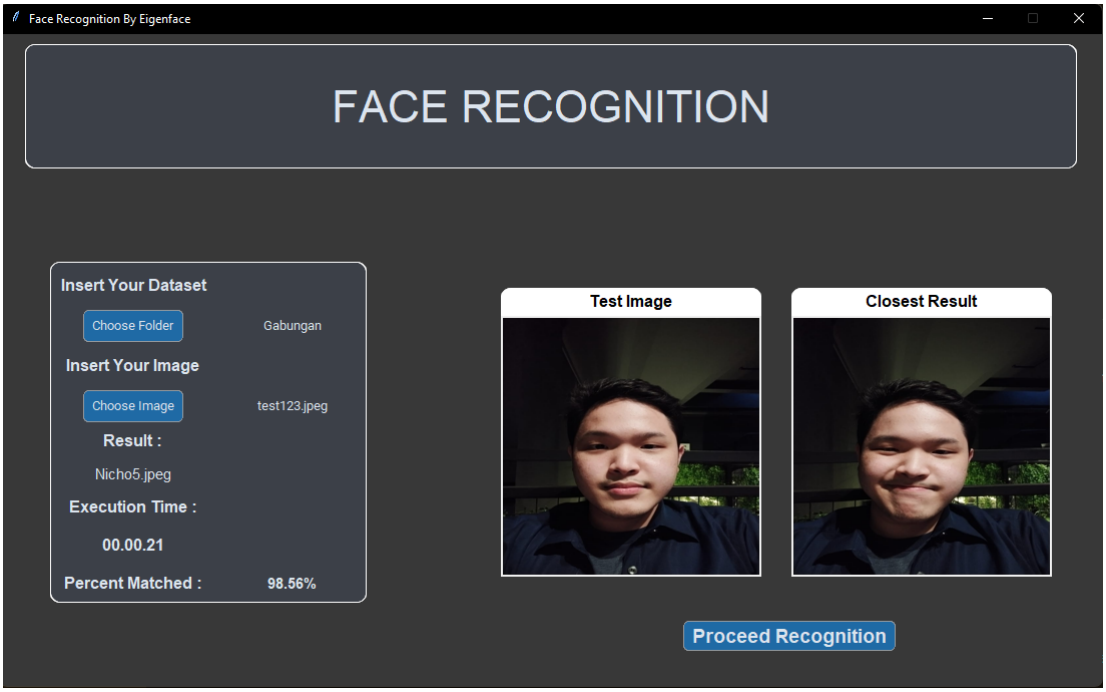


Sesuai espektasi diawal, gambar yang dihasilkan sama dengan yang aslinya dan memiliki tingkat persen match yang tinggi.

4.4 Percobaan - Gambar yang Tidak Ada di Basis Data

Pada kasus ini, kami mengambil satu gambar yang bukan dari basis data tetapi mirip dengan yang ada di basis data. Hipotesis awalnya ialah gambar yang akan dikeluarkan adalah gambar orang yang sama dan memiliki persen match yang cukup tinggi.

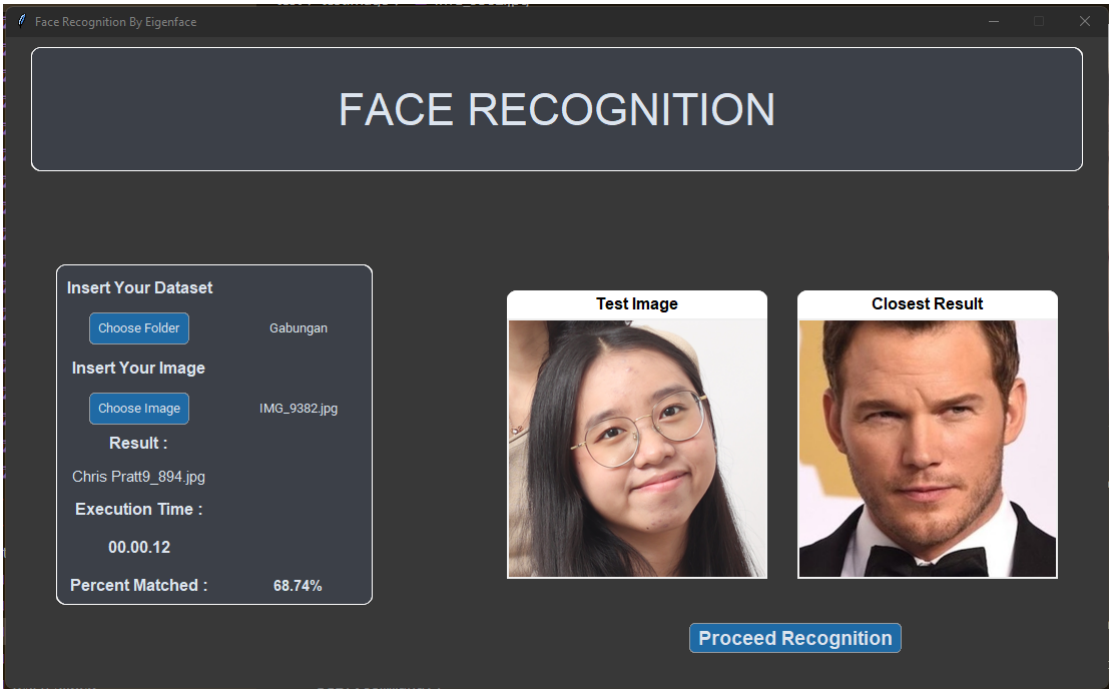


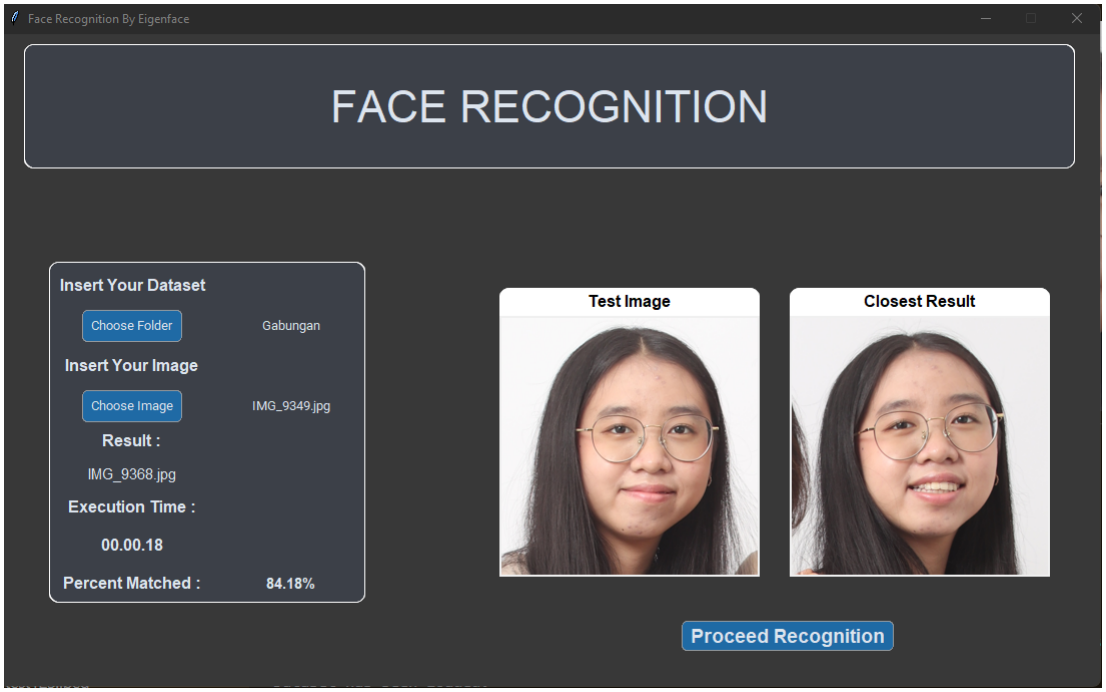


Sesuai espektasi diawal, gambar yang dihasilkan sama dengan orang yang aslinya dan memiliki tingkat persen match yang tinggi.

4.5 Percobaan - Sensitivitas Background Gambar

Pada kasus ini, kami menemukan satu kasus unik di mana background pada suatu image dan posisi muka sangat memengaruhi pengambilan gambar.

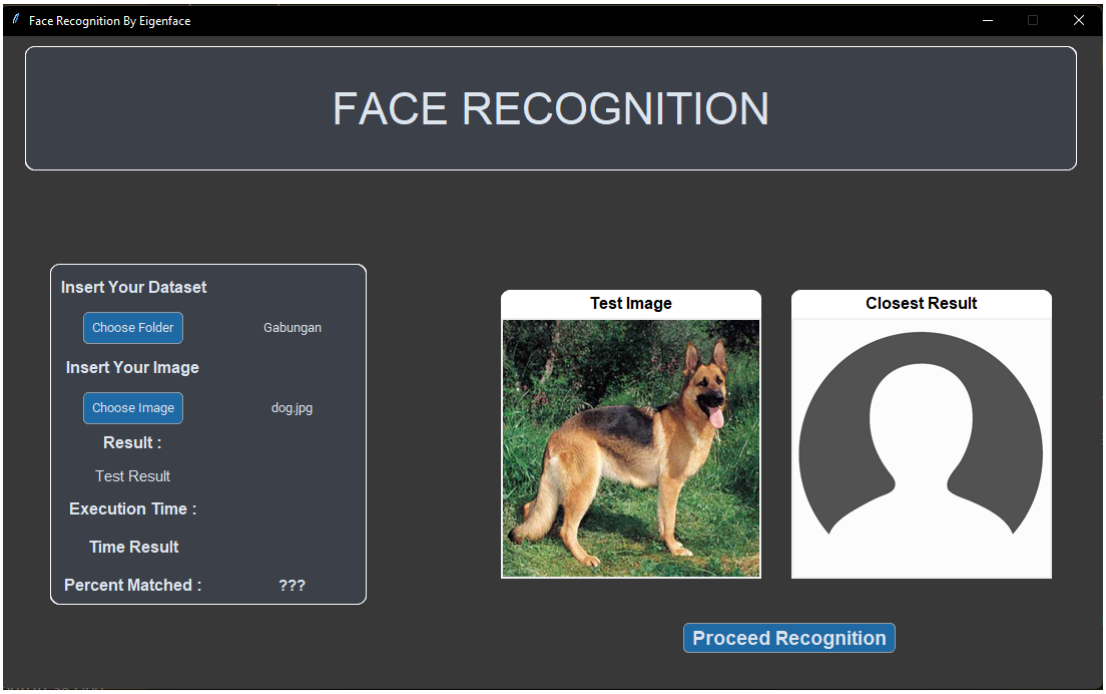


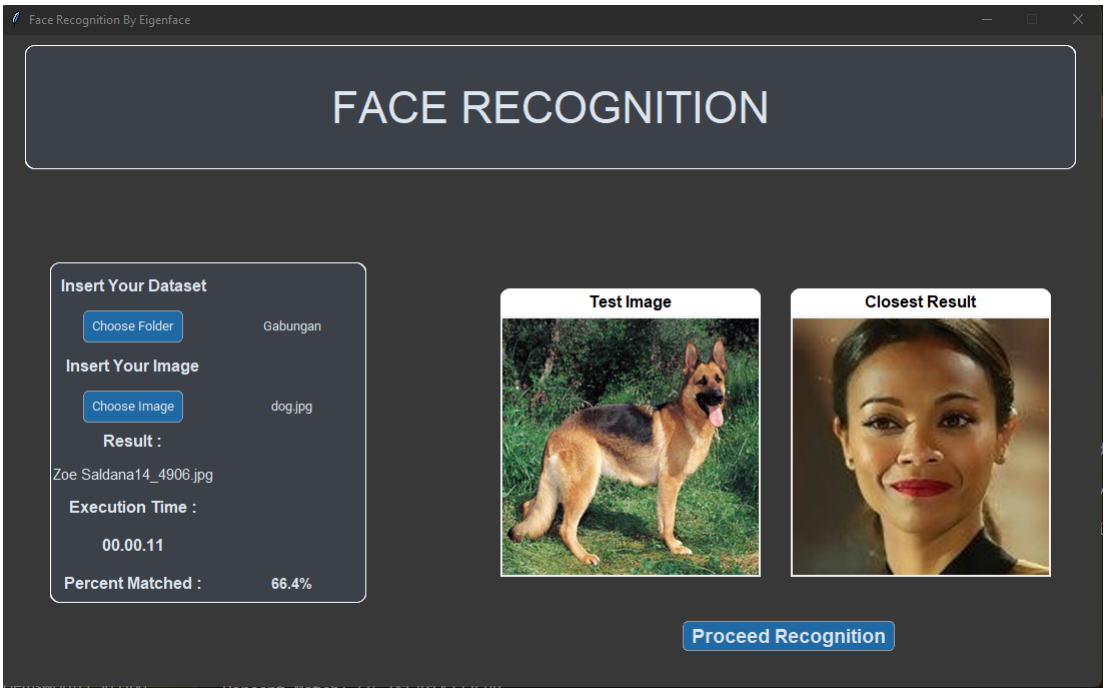


Dapat dilihat pada gambar pertama bahwa hasil yang didapatkan tidak sesuai. Menurut hasil analisis kami, penempatan muka dan background dari gambar sangat memengaruhi hasil euclidean distancenya sehingga menghasilkan gambar yang salah, bahkan nilai persentasenya saja kecil. Ketika dicoba dengan gambar yang dicrop lebih baik dan background yang lebih jernih, hasil yang benar bisa didapatkan.

4.6 Percobaan - Gambar Random

Pada kasus ini, kami mengambil satu gambar random yang sama sekali tidak ada hubungannya dengan basis data. Hipotesis awalnya ialah gambar yang dikeluarkan adalah gambar yang random serta tingkat persen match yang cukup rendah dibandingkan dengan nilai-nilai sebelumnya.





Sesuai espektasi diawal, gambar yang dihasilkan memang random dan tingkat persen match sebesar 71 persen. Alasan kenapa tingkat persen match cukup tinggi mungkin disebabkan karena gambar anjing ini dikurangi dengan muka rata-rata database sehingga mengubah sedikit dari karakteristik gambar anjing sehingga menjadikannya sedikit lebih dekat dengan gambar dibasis data.

5 Kesimpulan, Saran, dan Refleksi

5.1 Kesimpulan

Beberapa hal yang berhasil kami capai dari tugas ini adalah pemahaman konsep yang cukup mendalam tentang pemrosesan gambar dan matriks secara umum. Selain itu, menurut kami, program kami telah mencapai target sesuai spek yang diminta.

5.2 Saran

Saran pengembangan yang dapat dilakukan terhadap pemrosesan muka ini adalah meningkatkan tingkat kualitas dataset gambar serta optimasi algoritma-algoritma dekomposisi untuk mendapatkan vektor-vektor eigen dengan lebih efisien.

5.3 Refleksi

Tugas besar ini diberikan bersamaan dengan tugas-tugas yang tidak kalah besar lain. Oleh sebab itu, manajemen waktu menjadi suatu frasa yang sangat perlu diperhatikan. Selain itu juga kekompakan tim dan kontribusi masing-masing anggota entah dalam bentuk ide, pikiran, dan sebagainya merupakan hal yang sangat berharga dan harus diapresiasi.

6 Daftar Pustaka

Drury, Matthew. *How Does A Computer Calculate Eigenvalues?* URL: https://madrury.github.io/jekyll/update/statistics/2017/10/04/qr-algorithm.html (visited on 11/21/2022).

Geeks, Geeks For. *ML — Face Recognition Using Eigenfaces (PCA Algorithm)*. 2020. URL: https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/ (visited on 11/16/2022).

Kim, Kyungnam. *Face Recognition using Principle Component Analysis*. URL: http://staff.ustc.edu.cn/~zwp/teach/MVA/pcaface.pdf (visited on 11/16/2022).

Matthew Turk, Alex Pentland. "Eigenfaces for recognition". In: *Cognitive Neuroscience* 3.1 (1991), pp. 71–86. DOI: https://doi.org/10.1162/jocn.1991.3.1.71.

Munir, Rinaldi. *Nilai Eigen dan Vektor Eigen (Bagian 1)*. 2022. URL: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-Nilai-Eigen-dan-Vektor-Eigen-Bagian1.pdf (visited on 11/16/2022).

Wikipedia. *QR Decomposition*. URL: https://en.wikipedia.org/wiki/QR_decomposition (visited on 11/21/2022).

7 Lampiran

7.1 Link Repository GitHub

Repo GitHub [Click Me!] or https://github.com/NicholasLiem/Algeo02_21079

7.2 Link Video Demo (YouTube)

Video Demo (YouTube) [Click Me!] or https://youtu.be/61GL9FX2FX8