

**LAPORAN
TUGAS BESAR 2 IF2123
ALJABAR LINIER DAN GEOMETRI**

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



oleh

Fatih Nararya R. I.	13521060
Go Dillon Audris	13521062
Eugene Yap Jin Quan	13521074

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2022/2023**

DAFTAR ISI

DAFTAR ISI	ii
BAB I	1
DESKRIPSI MASALAH	1
1.1 Spesifikasi Tugas	1
BAB II	2
TEORI SINGKAT	2
2.1 Nilai Eigen dan Vektor Eigen	2
2.2 Algoritma Komputasi Eigenpair	3
2.2.1 Dekomposisi QR	3
2.2.2 Prosedur Gram-Schmidt	3
2.2.3 QR Algorithm	4
2.3 Principal Component Analysis with Eigenface	5
BAB III	10
IMPLEMENTASI PROGRAM	10
3.1 Algoritma	10
3.1.1 Program Sebagai Kumpulan Fungsi	10
3.1.1.1 path_generator	10
3.1.1.2 image_f_matrix_generator	10
3.1.1.2 image_f_matrix_generator	11
3.1.1.3 average_flatten_generator	11
3.1.1.4 training_matrix_generator	11
3.1.1.4 eigen_generator	11
3.1.1.5 y_generator	11
3.1.1.6 omega_of_target	12
3.1.1.6 euclidean_distance	12
3.1.1.6 bestface	12
3.1.2 Menggabungkan fungsi-fungsi	12
3.2 GUI	12
3.2.1 Tech Stack / Kakas yang digunakan	12
3.2.2 Gambaran Besar GUI (Program Utama) dan Algoritmanya	13
3.2.2.1 getTestImage	14
3.2.2.2 getFolderName	14
3.2.2.3 getCurrentTime	15

3.2.2.4 subtractAndShowTime	15
3.2.2.5 startTrainingSet	15
3.2.2.6 processTrainingSet	15
3.2.2.7 startBestImage	15
3.2.2.8 searchBestImage	15
3.2.3 Gambaran Besar GUI (Kamera)	16
3.2.3.1 startCapturing	16
3.2.3.2 processCaptureImage	17
3.2.3.3 stopCapturing	17
BAB IV	17
EXPERIMEN	18
4.1 Eksperimentasi terhadap Perbedaan Ukuran Gambar Uji	18
4.2 Eksperimentasi terhadap Perbedaan Warna Gambar Uji	22
4.3 Eksperimentasi terhadap Perbedaan Pose Wajah Gambar Uji	33
4.4 Eksperimentasi terhadap Perbedaan Pencahayaan Gambar Uji	37
4.5 Pengujian Fitur Kamera	46
BAB V	50
KESIMPULAN, SARAN, DAN REFLEKSI	50
5.1 Kesimpulan	50
5.2 Saran	50
5.3 Refleksi	51
DAFTAR PUSTAKA	52
LAMPIRAN	54
Link Repository Program	55
Link Video Demo Youtube	55

BAB I

DESKRIPSI MASALAH

1.1 Spesifikasi Tugas

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/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 II

TEORI SINGKAT

2.1 Nilai Eigen dan Vektor Eigen

“Eigen” adalah kata yang berasal dari bahasa Jerman yang bermakna “karakteristik” atau “asli”. Dalam konteks aljabar linier, nilai eigen (*eigenvalue*) dan vektor eigen (*eigenvector*) adalah nilai dan vektor karakteristik dari sebuah matriks persegi. Salah satu aplikasi dari nilai eigen dan vektor eigen adalah dalam pengenalan wajah.

Menurut definisi, skalar λ dan vektor tidak nol \mathbf{x} di ruang \mathbf{R}^n masing-masing merupakan nilai eigen dan vektor eigen dari matriks persegi A jika memenuhi hubungan berikut.

$$A\mathbf{x} = \lambda\mathbf{x}$$

Vektor eigen \mathbf{x} adalah vektor eigen yang berkorespondensi dengan nilai eigen λ . Pasangan nilai eigen dan vektor eigen yang berkorespondensi umumnya disebut sebagai pasangan eigen (*eigenpair*).

Misal A adalah matriks persegi dengan orde n . Prosedur sederhana perhitungan pasangan eigen umumnya melibatkan persamaan karakteristik $\det(\lambda I - A) = 0$. Akar-akar dari persamaan karakteristik ini adalah nilai-nilai eigen dari matriks A . Untuk mendapatkan vektor eigen dari sebuah nilai eigen, langkah yang dilakukan adalah substitusi nilai eigen ke dalam persamaan

karakteristik. Setelah itu, solusi vektor eigen adalah vektor x yang memenuhi persamaan hasil langkah substitusi.

2.2 Algoritma Komputasi *Eigenpair*

Pada program yang telah dibuat, perhitungan *eigenpair* menggunakan algoritma QR, yakni salah satu algoritma iteratif dalam perhitungan *eigenpair*. Untuk perhitungan dekomposisi QR, program ini menggunakan prosedur Gram-Schmidt.

2.2.1 Dekomposisi QR

Algoritma QR memanfaatkan dekomposisi QR, yaitu dekomposisi/faktorisasi sebuah matriks A menjadi sebuah matriks ortogonal Q , yaitu matriks yang memenuhi $Q^T Q = I$, dan sebuah matriks segitiga atas R . Persamaan umum dari dekomposisi QR adalah $A = QR$.

2.2.2 Prosedur Gram-Schmidt

Dalam prosedur Gram-Schmidt, sebuah matriks A terlebih dahulu dipecah menjadi sekumpulan vektor kolom, yaitu $A = [a_1 | a_2 | \dots | a_n]$. Setelah memecah A menjadi vektor-vektor kolom, langkah selanjutnya adalah membentuk matriks ortogonal Q . Matriks Q tersusun atas vektor e_1 hingga e_n , sehingga $Q = [e_1 | e_2 | \dots | e_n]$. Vektor-vektor e_k merupakan normalisasi terhadap vektor u_k , yang diperoleh dari pola berikut.

$$u_1 = a_1, e_1 = \frac{u_1}{||u_1||}$$

$$u_2 = a_2 - (a_2 \cdot e_1)e_1, e_2 = \frac{u_2}{||u_2||}$$

$$u_{k+1} = a_{k+1} - (a_{k+1} \cdot e_1)e_1 - \dots - (a_{k+1} \cdot e_k)e_k, e_{k+1} = \frac{u_{k+1}}{||u_{k+1}||}$$

Tahap terakhir dari prosedur Gram-Schmidt adalah membentuk matriks R . Elemen-elemen dari matriks segitiga R diperoleh dari perkalian titik antara a_k dan e_k , dan memenuhi persamaan berikut.

$$R = \begin{bmatrix} a_1 \cdot e_1 & a_2 \cdot e_1 & \dots & a_n \cdot e_1 \\ 0 & a_2 \cdot e_2 & \dots & a_n \cdot e_2 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_n \cdot e_n \end{bmatrix}$$

Dengan demikian, perhitungan dekomposisi QR dari sebuah matriks persegi A dengan prosedur Gram-Schmidt dapat dinyatakan dalam persamaan di bawah.

$$A = [a_1|a_2| \dots |a_n] = [e_1|e_2| \dots |e_n] \begin{bmatrix} a_1 \cdot e_1 & a_2 \cdot e_1 & \dots & a_n \cdot e_1 \\ 0 & a_2 \cdot e_2 & \dots & a_n \cdot e_2 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_n \cdot e_n \end{bmatrix} = QR$$

2.2.3 QR Algorithm

Algoritma QR adalah salah satu algoritma yang digunakan dalam perhitungan *eigenpair*. Algoritma QR melakukan dekomposisi QR terhadap suatu matriks persegi A dan melakukan perkalian R dan Q hasil dekomposisi secara iteratif. Perulangan ini dilakukan hingga tercapai konvergensi menuju bentuk matriks

segitiga atas, dengan nilai-nilai diagonal utama sebagai nilai-nilai eigen dari A . Secara lebih jelas, pencarian aproksimasi nilai-nilai eigen dari sebuah matriks A mengikuti langkah berikut.

1. Masukkan matriks persegi A_k
2. Dekomposisi A_k menjadi Q_k dan R_k
3. Susun A_{k+1} sebagai perkalian matriks R_k dan Q_k , $A_{k+1} = R_k Q_k$
4. Ulangi langkah 2-3 sebanyak jumlah iterasi yang ditentukan, atau hingga tercapai konvergensi nilai elemen diagonal matriks A_{k+1}

Apabila matriks masukan A bersifat simetris, algoritma QR mampu aproksimasi vektor eigen dari A secara iteratif. Iterasi untuk pencarian vektor-vektor eigen dalam kasus matriks simetris mengikuti langkah-langkah berikut.

1. Deklarasi V sebagai I , yaitu matriks identitas yang seukuran dengan A .
2. Pada setiap iterasi algoritma QR, deklarasi ulang V sebagai perkalian antara Q_k dengan V .
3. Pada akhir iterasi, vektor-vektor kolom dari V akan memuat aproksimasi vektor-vektor eigen dari A .

2.3 Principal Component Analysis with Eigenface

Sebuah gambar wajah memiliki banyak ciri khas atau *unique feature* yang mendefinisikan dirinya (*principal components*). Ketika kita ingin mencocokkan jika sebuah gambar wajah A sama dengan gambar wajah B, maka kita tinggal mengukur saja seberapa jauh *principal components* antara kedua wajah tersebut. Secara garis besar, ini adalah bagaimana PCA (*Principal Components Analysis*)

digunakan. Penjelasan tingkat tinggi ini tidak menjawab bagaimana implementasi nyata sampai ke *nitty gritty* dari pengenalan wajah menggunakan PCA. Di sinilah *eigenvalues* dan *eigenvector* muncul.

Sebuah wajah dapat direpresentasikan sebagai sebuah matriks dengan tiap elemennya adalah nilai RGBA dari tiap *pixel* wajah. Matriks inilah yang kemudian akan kita manipulasi untuk mendapatkan *principal components* dari wajah menggunakan *eigenvector* dan *eigenvalues*.

Pada penjelasan di bawah ini, kita akan mencoba mencari dari sebuah himpunan gambar yang ada, mana gambar yang paling mirip dengan gambar yang kita berikan.

Misalkan matriks dari sebuah gambar adalah I dengan ukuran $m \times n$. Kita “ratakan” matriks tersebut menjadi matriks x dengan menjajarkan kolom-kolomnya menjadi 1 sehingga ia hanya memiliki 1 kolom. Dimensi x adalah $N^2 \times 1$ dengan $N = m \times n$.

Misalkan kita memiliki M buah gambar sehingga terdapat M buah matriks x_i ($i = 1, 2, 3, \dots M$). Maka kita perlu mencari rata-rata dari keseluruhan matriks tersebut (Ψ).

$$\Psi = \frac{1}{M} \sum_{i=1}^M x_i$$

Kemudian kita cari selisih tiap-tiap matriksnya dengan rata-rata.

$$\phi_i = x_i - \Psi$$

Tiap-tiap selisih tersebut akan kita jajarkan menjadi kolom dari *training matrix* (A).

$$A = (\phi_1, \phi_2, \phi_3, \dots, \phi_M)$$

Tahap selanjutnya barulah kita akan menggunakan *eigenvector* dan *eigenvalue*. Kita akan menghitung matriks kovarian (C) dan mencari nilai eigen λ_i dan vektor eigen e_i dari matriks tersebut.

$$C = AA^T$$

$$Ce_i = \lambda_i e_i$$

Karena matriks C memiliki dimensi $N \times N$, maka akan terdapat N buah vektor eigen dan nilai eigen. Andaikan gambar yang kita gunakan memiliki ukuran 256×256 , maka akan terdapat 65536 buah nilai eigen dan vektor eigen. Untuk mencari nilai eigen, kami menggunakan *QR decomposition* berulang-ulang sampai sekitar 40 kali. Metode ini melibatkan perkalian matriks di tiap iterasinya dan mengingat perkalian matriks saja memiliki kompleksitas waktu sebanyak $O(n^3)$, komputasi ini bukan hanya tidak efisien tetapi juga akan memakan waktu yang lama.

Akan tetapi, terdapat sebuah teorema aljabar linier yang dapat kami eksploitasi, yaitu bahwa nilai eigen dari C dapat dicari dengan mencari nilai eigen dari $C_1 = A^T A$. Karena dimensi C_1 adalah $M \times M$, maka jauh lebih mudah untuk melakukan komputasi nilai eigen dan vektor eigennya. Jika μ_i adalah nilai eigen dari C_1 dan v_i adalah vektor eigen dari C_1 , maka kita bisa dapatkan persamaan berikut.

$$A^T A v_i = \mu_i v_i$$

Dengan mengalikan A pada kedua ruas dari kiri, dihasilkan :

$$A^T A A v_i = A \mu_i v_i$$

$$A^T A (A v_i) = \mu_i (A v_i)$$

$$C(A v_i) = \mu_i (A v_i)$$

Sehingga dapat disimpulkan bahwa $M - 1$ pertama nilai eigen λ_i dan vektor eigen e_i dari C adalah μ_i dan $A v_i$. Kita cukup mendapatkan $M - 1$ pertama dari nilai eigen dan vektor eigennya karena sisa nilai eigen dari C memiliki nilai yang kecil (variannya kecil) sehingga *principal components* yang digambarkan oleh mereka juga tidak benar-benar menggambarkan secara akurat atau *tangible* gambar wajah asal mereka. Bahkan $M - 1$ nilai eigen yang kita dapatkan pun 90 persennya tidak krusial dalam pengenalan wajah karena nilainya yang mendekati 0 sehingga juga tidak signifikan.

Setelah mendapatkan vektor eigen e_i kita perlu menormalisasinya lalu membentuknya menjadi matriks baru E dengan e_i sebagai vektor kolomnya. Misalkan bahwa dari $M - 1$ vektor eigen tadi kita hanya mengambil B buah pertama yang paling besar karena tidak signifikannya vektor eigen sisanya, maka dimensi dari matriks E ini adalah $N \times B$.

Matriks E ini kemudian digunakan untuk mendapatkan matriks lain yaitu matriks Y . Bagian dari matriks Y yang kita pedulikan ini adalah y_i yang merupakan vektor kolomnya ($Y = (y_1, y_2, y_3, \dots, y_M)$).

$$Y = E^T A$$

Sekarang kita telah selesai membuat semua nilai yang diperlukan dari *training image*, dan akan melanjutkan ke langkah terakhir yang berurusan dengan *image* yang ingin kita uji.

Gambar yang kita uji diratakan juga seperti langkah pertama dari pengolahan *training image* yang telah disebutkan menjadi matriks P . Kita lakukan operasi berikut untuk mendapatkan ω .

$$\omega = E^T(P - \Psi)$$

Klasifikasi dilakukan dengan menghitung ϵ_i , yaitu jarak antara ω dengan tiap vektor kolom dari Y yaitu y_i . Penghitungan jarak ini dilakukan dengan jarak euclidean. Gambar yang dianggap dikenali kemudian adalah gambar dalam *training image* yang memiliki jarak paling kecil dengan gambar yang diuji.

BAB III

IMPLEMENTASI PROGRAM

3.1 Algoritma

Dalam pengerjaan program ini, kami telah mencari referensi yang cukup lengkap sehingga mendapatkan langkah pengerjaan yang komprehensif seperti yang telah dijelaskan di bab sebelumnya. Maka kami akan banyak merujuk pada variabel-variabel yang dipakai di sana pula.

3.1.1 Program Sebagai Kumpulan Fungsi

Program ini terstruktur sedemikian rupa sehingga tiap-tiap bagian perhitungan dipisah menjadi fungsi tersendiri. Hal ini merupakan refleksi dari alur pengerjaan program yang runut sesuai langkah yang ada dan juga *best practice* untuk mempermudah *testing* dan *debugging*.

3.1.1.1 path_generator

Menerima masukan berupa *absolute path* dari *directory* tempat *training images* disimpan, lalu mengembalikan *list* berisi *absolute path* dari tiap *training images* yang ada di dalam *directory* tersebut.

3.1.1.2 image_f_matrix_generator

Fungsi ini menerima masukan berupa *absolute path* dari *directory* tempat *training images* disimpan, lalu mengembalikan *list* berisi matriks representasi tiap gambar di dalam *directory* tersebut yang sudah diratakan yaitu x_i .

Gambar pertama-tama dibaca oleh *library* opencv yang mengubah gambar menjadi matriks dengan tiap elemennya adalah nilai RGB dari *pixel* dengan jangkauan 0 sampai 255. Kemudian pemerataan dilakukan menggunakan *library* numpy dengan metode *flatten*.

3.1.1.2 image_f_matrix_generator

Fungsi ini menerima masukan berupa *absolute path* dari *directory* tempat *training images* disimpan, lalu mengembalikan *list* berisi seluruh x_i , sebut saja R .

3.1.1.3 average_flatten_generator

Menerima R dari *image_f_matrix_generator* dan menghasilkan rata-rata dari semua matriks dalam *list* tersebut (ψ).

3.1.1.4 training_matrix_generator

Menerima R dan W , kemudian menggunakan selisih x_i dan ψ yang disusun sebagai kolom untuk menghasilkan *training matrix* (A).

3.1.1.4 eigen_generator

Mencari nilai eigen dari matriks C_1 . Mengembalikan *tuple* daftar eigenvalue dan daftar vektor eigen yang sudah dinormalisasi dan dijadikan matriks baru (E).

3.1.1.5 y_generator

Menerima C_1 dan A , lalu menggunakan *eigen_generator* untuk menghasilkan E yang kemudian digunakan menghitung nilai Y .

3.1.1.6 omega_of_target

Menerima nama *file* gambar yang ingin diuji, ψ , dan E , untuk menghasilkan ω .

3.1.1.6 euclidean_distance

Mendapatkan jarak euclidean antara dua buah matriks dengan menggunakan fungsi *square* dan *sum* yang dimiliki oleh *library* numpy.

3.1.1.6 bestface

Menerima nama *file* gambar yang ingin diuji, ψ , Y , *list path training image* dan E untuk menghasilkan gambar dari *training set* yang memiliki jarak euclidean paling dekat dengan ω dibanding gambar lainnya dari *training set*. Jika jarak euclidean distance lebih besar dari 10^6 , maka fungsi akan mengembalikan string kosong.

3.1.2 Menggabungkan fungsi-fungsi

Tiap-tiap fungsi yang telah disebutkan di subbab sebelumnya melakukan tiap bagian terpisah dari perhitungan yang perlu dilakukan untuk melakukan pengenalan wajah. Penggabungan ini dilakukan pada fungsi, di mana hasil dari satu fungsi kemudian diberikan sebagai parameter kepada fungsi lain sampai akhirnya seluruh penggabungan fungsi tersebut dalam fungsi *process_images* dapat menerima *path* gambar tes dan menghasilkan semua nilai yang dibutuhkan oleh fungsi *bestface*.

3.2 GUI

3.2.1 Tech Stack / Kakas yang digunakan

Dalam membuat Graphical User Interface program, digunakan beberapa package dan modul Python seperti:

1. Kivy:

Digunakan sebagai kerangka membuat GUI program dan isinya seperti Button, Image, Camera, Label, dan lain-lain.

2. PIL (Python Imaging Library):

Digunakan dalam beberapa proses yang membutuhkan penyesuaian image agar dapat diproses dengan baik dan menghindari error, seperti menyesuaikan gambar uji dengan ukuran training set, serta mengubah format gambar dari PNG menjadi JPG.

3. Datetime:

Digunakan untuk menentukan lama pemrosesan training set dan pencarian gambar dalam training set yang paling mirip dengan gambar uji.

4. Plyer:

Digunakan untuk membuka file explorer untuk memudahkan pengguna memilih folder training set dan file gambar uji.

3.2.2 Gambaran Besar GUI (Program Utama) dan Algoritmanya

GUI dari program utama didesain untuk memproses training set dan mencari gambar dari training set yang paling mirip dengan gambar uji. Dalam GUI Program Utama, terdapat beberapa tombol utama yaitu memilih folder training

set, memilih file gambar uji, melakukan train pada training set, dan mencari gambar dalam training set yang paling mirip dengan gambar uji.

Tombol untuk memilih folder training set dan memilih file gambar uji akan memanggil fungsi yang menggunakan library Plyer. Gambar uji akan ditampilkan ke layar, dan path folder serta file akan disimpan untuk kemudian di pass ke algoritma utama. Jika folder telah dipilih, maka pengguna dapat melakukan train pada training set. Algoritma utama akan melakukan train dan GUI kemudian menampilkan waktu yang dibutuhkan untuk melakukan train. Pengguna kemudian dapat mencari gambar dari training set yang paling mirip dengan gambar uji. Algoritma utama akan dipanggil lagi dan GUI akan menampilkan gambar hasil, nama file gambar tersebut, serta waktu pemrosesan. Dalam mencari waktu pemrosesan, digunakan library Datetime. Perlu diperhatikan bahwa pengguna harus melakukan train terlebih dahulu sebelum mencari gambar hasil atau algoritma tidak akan dijalankan.

Pengguna cukup melakukan train 1 kali terhadap training set, yang kemudian dapat dipakai untuk mencari gambar hasil dari gambar uji yang berbeda-beda. Pengguna juga dapat melakukan train lagi kepada training set yang berbeda untuk mendapatkan gambar hasil yang berbeda pula.

3.2.2.1 getTestImage

Menggunakan library Plyer untuk memunculkan filechooser agar pengguna dapat memilih file gambar dalam format jpg. Gambar dari file dan namanya akan ditampilkan ke GUI

3.2.2.2 getFolderName

Menggunakan library Plyer untuk memunculkan directory chooser agar pengguna dapat memilih folder training set. Path dari folder training set lalu disimpan dalam suatu variabel.

3.2.2.3 getCurrentTime

Menggunakan library Datetime dan mengembalikan suatu tuple yang berisi jam, menit, detik, dan mikrodetik saat ini.

3.2.2.4 subtractAndShowTime

Menerima waktu awal dan akhir dari pemrosesan, dan melakukan proses untuk menampilkan execution time ke GUI.

3.2.2.5 startTrainingSet

Mengecek apakah sudah ada folder training set yang dipilih oleh pengguna. Jika ada, maka fungsi akan memanggil fungsi processTrainingSet.

3.2.2.6 processTrainingSet

Melakukan train terhadap folder training set yang telah dipilih oleh pengguna. Fungsi ini akan memanggil fungsi getCurrentTime di awal dan akhir pemrosesan, dan di tengah-tengah menjalankan algoritma utama. Terakhir, memanggil fungsi subtractAndShowTime.

3.2.2.7 startBestImage

Mengecek apakah pengguna telah melakukan train terhadap training set dan memilih gambar uji. Jika kondisi dipenuhi, akan memanggil fungsi searchBestImage.

3.2.2.8 searchBestImage

Mencari gambar dari training set yang paling mirip dengan gambar uji dan menampilkan gambar tersebut. Namun, jika jarak euclidean gambar terlalu jauh, maka akan memberitahukan bahwa tidak ada gambar yang mirip.. Fungsi akan memanggil fungsi getCurrentTime di awal dan akhir pemrosesan, dan di tengah-tengah akan melakukan 2 hal, yaitu mengkonversi gambar uji agar sesuai dengan gambar pada training set, lalu memanggil algoritma utama untuk mencari gambar paling sesuai. Selanjutnya, fungsi akan memanggil subtractAndShowTime.

3.2.3 Gambaran Besar GUI (Kamera)

Dalam tugas besar ini, kami juga mengerjakan bonus kamera. GUI dari program telah dirancang untuk dapat berpindah dari screen utama (untuk pemrosesan yang dijelaskan pada subbab 3.2.2) ke screen kamera. Ketika berpindah, maka akan terjadi passing hasil train training set dari screen utama ke screen kamera, dan kamera akan dinyalakan.

Program lalu akan mengambil gambar dari kamera setiap 2 detik, dan setiap 2 detik pula akan memberikan gambar hasil yang paling dekat dengan gambar yang ditangkap oleh kamera atau memberitahu bahwa tidak ada gambar yang cocok bila euclidean distance terlalu besar.. Pemrosesan penangkapan gambar (image

capturing) di sini menggunakan library PIL agar bisa menyesuaikan gambar yang ditangkap agar dapat diolah oleh algoritma utama. Perlu diperhatikan bahwa pemrosesan tangkapan gambar hanya akan dilakukan ketika sudah melakukan train pada training set di screen utama.

3.2.3.1 startCapturing

Fungsi akan menyalakan kamera, dan memeriksa apakah pengguna sudah melakukan train terhadap folder training set pada screen manual. Jika sudah, maka fungsi akan menjadwalkan pemanggilan fungsi processCaptureImage setiap 2 detik.

3.2.3.2 processCaptureImage

Fungsi akan melakukan capture gambar kamera dan mengkonversinya agar sesuai dengan training set. Selanjutnya memanggil algoritma utama untuk mencari gambar hasil yang paling mirip dan menampilkannya ke layar.

3.2.3.3 stopCapturing



Fungsi digunakan untuk mematikan kamera dan menghapus fungsi processCaptureImage dari jadwal.



BAB IV

EXPERIMEN

4.1 Eksperimentasi terhadap Perbedaan Ukuran Gambar Uji

Eksperimentasi dilakukan terhadap 3 gambar dengan tiga ukuran berbeda, dan menggunakan trainingSet1. Gambar uji yang digunakan adalah potret dari Mantan Presiden Megawati Soekarnoputri, Presiden Joko Widodo, dan Presiden Joe Biden. Sebelumnya telah dilakukan train terhadap folder trainingSet1 yang memiliki 114 gambar dan memerlukan waktu eksekusi sekitar 11.47 detik.

Input

Output (kasus uji: 880x880)

Output (kasus uji: 440x440)


<p>Output (kasus uji: 660x660)</p>

<p>Keterangan</p>
<p>Gambar uji konsisten menampilkan gambar yang sama</p>

<p>Input</p>

<p>Output (kasus uji: 906x906)</p>



Output (kasus uji: 453x453)






Output (kasus uji: 680x680)



Keterangan

Gambar uji konsisten menampilkan gambar yang sama

Input

Output (kasus uji: 805x805)

Output (kasus uji: 403x403)

Output (kasus uji: 604x604)



Hasil eksperimentasi adalah ketiga kelompok gambar menunjukkan hasil yang sama pada kelompok gambar masing-masing. Akan tetapi, terdapat perbedaan kecil pada waktu pemrosesan gambar antara gambar yang kecil dengan gambar yang besar. Kami menduga bahwa hal ini terjadi akibat pengubahan ukuran gambar uji menjadi 256x256 sebelum pemrosesan pada program.

4.2 Eksperimentasi terhadap Perbedaan Warna Gambar Uji

Eksperimentasi dilakukan terhadap tiga kelompok gambar dengan enam mode pewarnaan yang berbeda, dan menggunakan trainingSet3. Gambar uji yang digunakan adalah potret Prof. Reini Wirahadikusumah, Motoaki Tanigo, dan Vincent (asisten Algeo). Sebelumnya telah dilakukan train pada folder trainingSet3 yang memerlukan waktu sekitar 552.13 detik.

Input



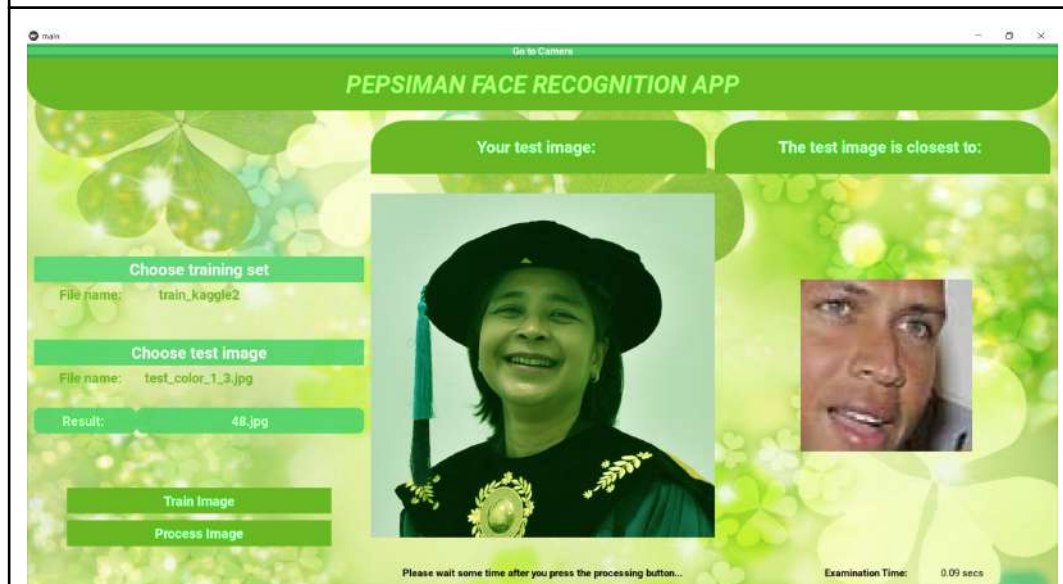
Output (kasus uji: original)



Output (kasus uji: red tint)



Output (kasus uji: green tint)



Output (kasus uji: blue tint)




Output (kasus uji: grayscale)



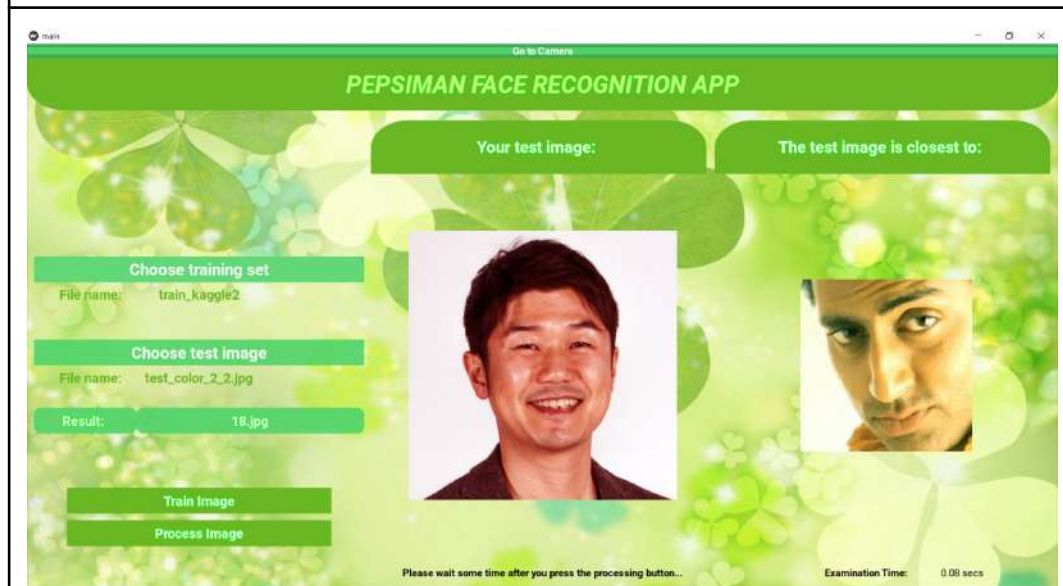
Output (kasus uji: high saturation)


<p>Keterangan</p>
<p>Kelompok gambar uji (selain versi blue tint dan high saturation) konsisten menampilkan gambar yang sama.</p>

<p>Input</p>

<p>Output (kasus uji: original)</p>



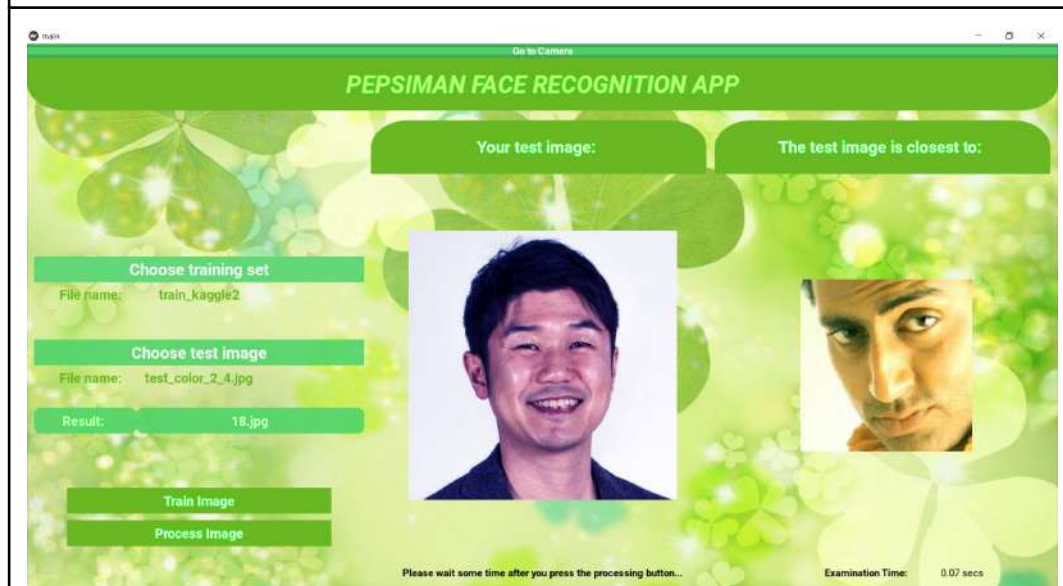
Output (kasus uji: red tint)



Output (kasus uji: green tint)



Output (kasus uji: blue tint)



Output (kasus uji: grayscale)



Output (kasus uji: high saturation)



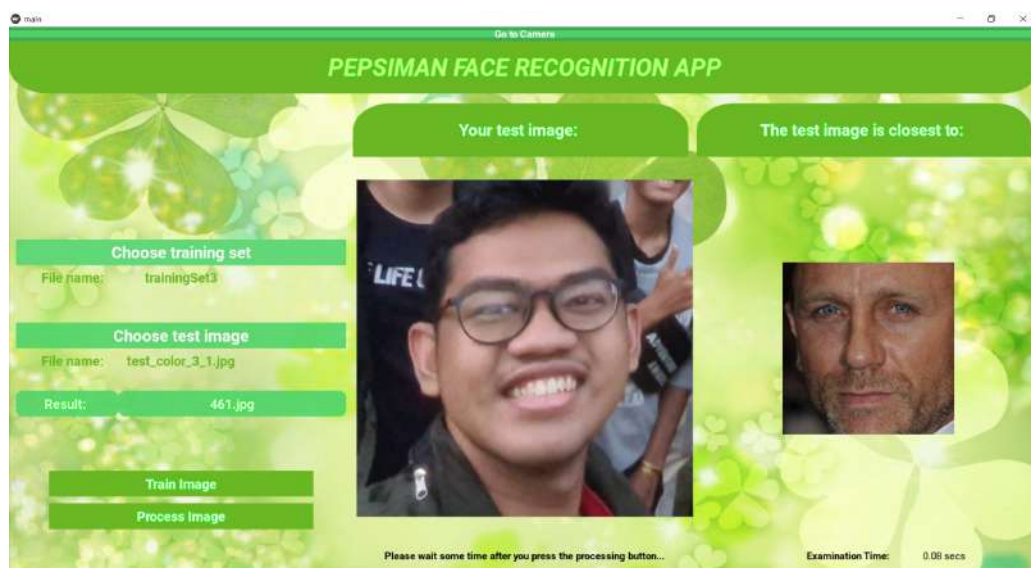
Keterangan

Kelompok gambar uji konsisten menampilkan gambar yang sama

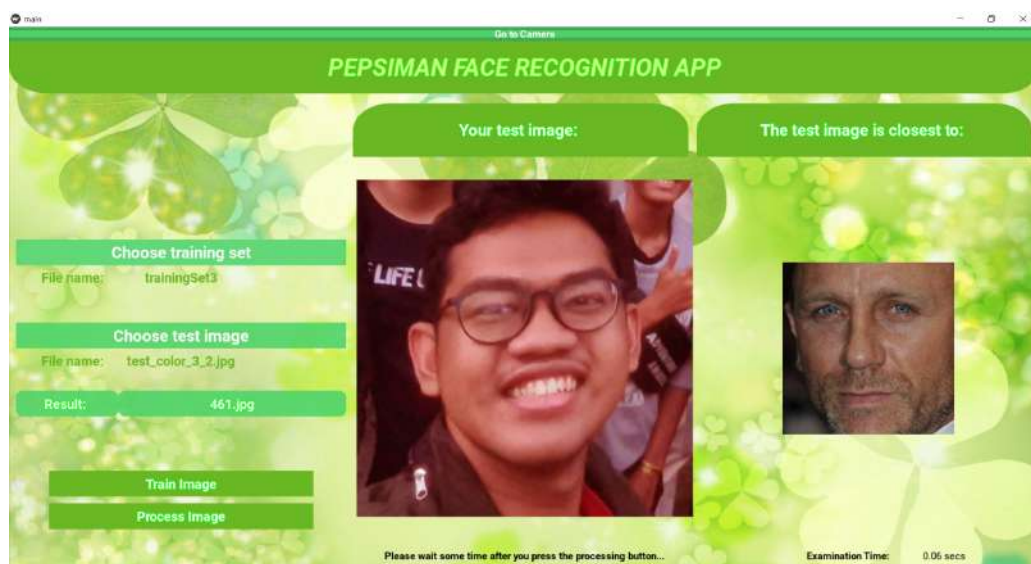
Input



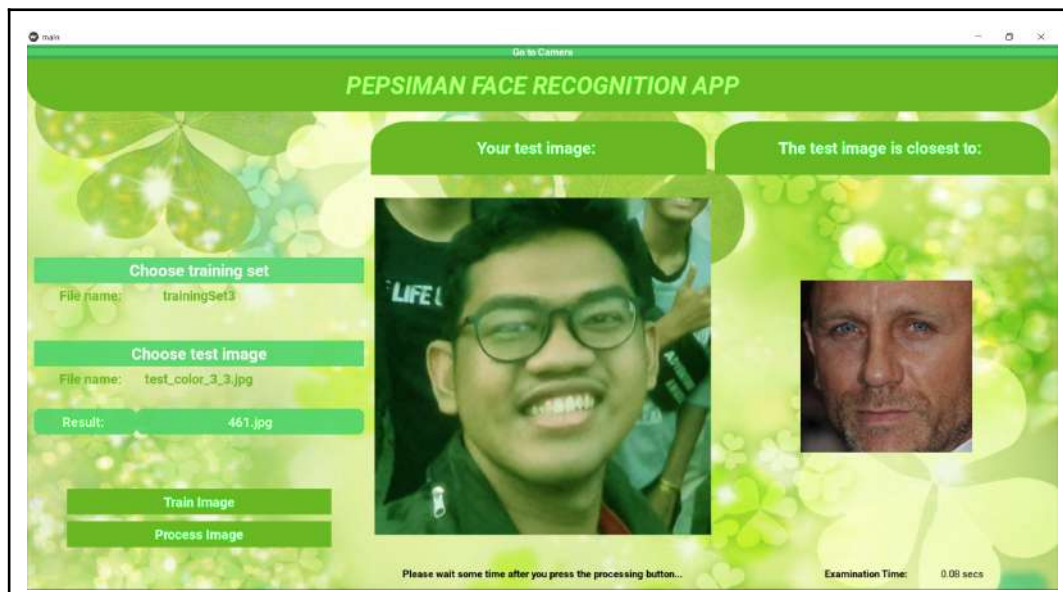
Output (kasus uji: original)



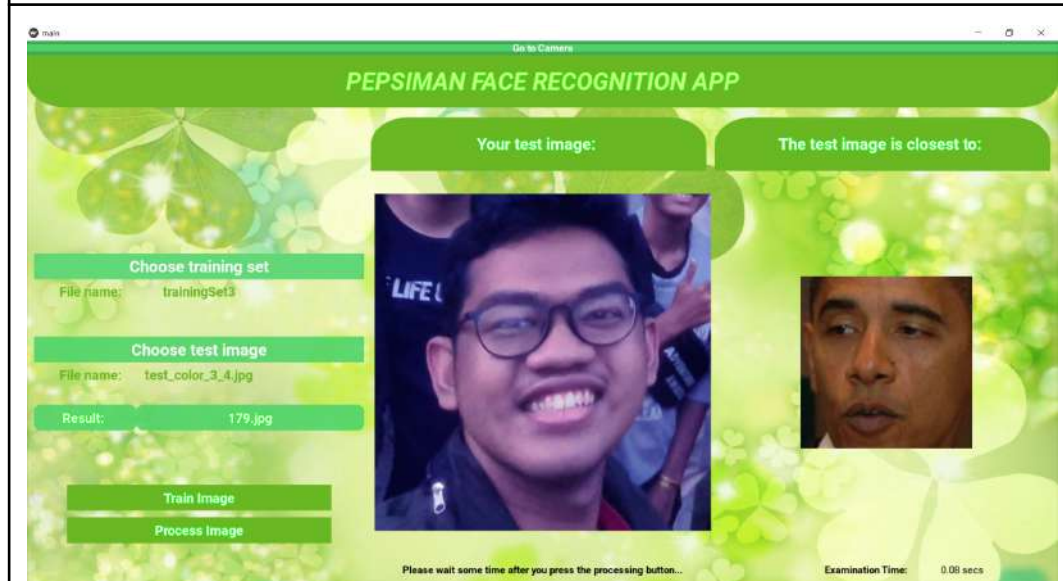
Output (kasus uji: red tint)



Output (kasus uji: green tint)



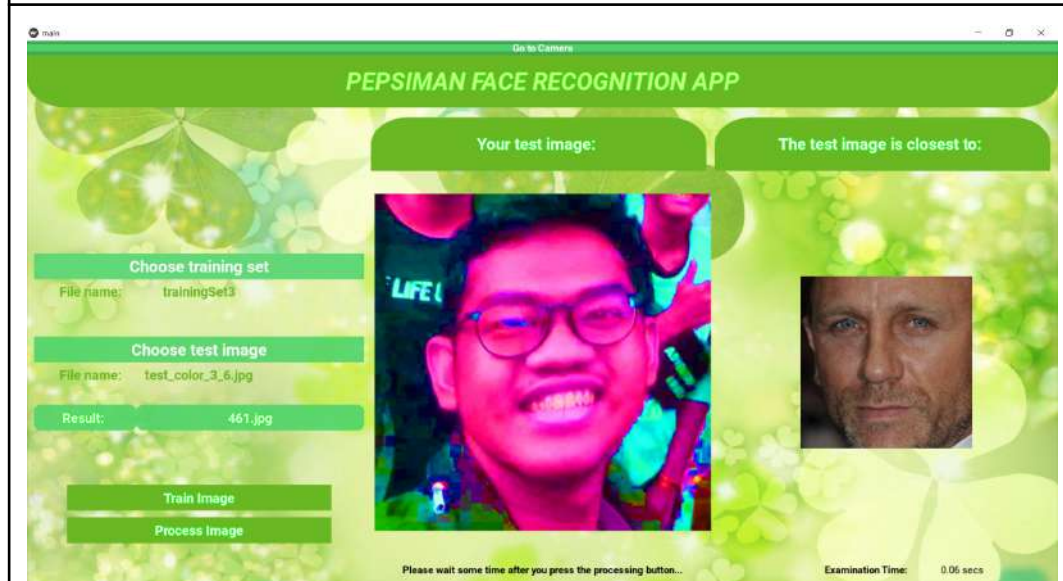
Output (kasus uji: blue tint)



Output (kasus uji: grayscale)



Output (kasus uji: high saturation)



Keterangan

Kelompok gambar uji (selain versi blue tint) konsisten menampilkan gambar yang sama.


Menurut hasil eksperimentasi, pewarnaan dari sebuah gambar uji berpengaruh terhadap hasil yang didapatkan. Kelompok uji pertama menunjukkan perbedaan hasil gambar pada gambar uji “blue tint” dan “high saturation”. Kelompok uji


ketiga juga menunjukkan perbedaan pada gambar uji “blue tint”. Akan tetapi, kelompok gambar uji kedua tidak menunjukkan perbedaan hasil. Hal yang menarik dari pengujian ini adalah pewarnaan biru dan “high saturation” memiliki pengaruh lebih besar jika dibandingkan dengan mode pewarnaan lain.


Berdasarkan hasil pengujian, kami menduga bahwa salah satu faktor yang mempengaruhi pencarian gambar adalah perbedaan warna di sekitar wajah gambar uji. Dapat dilihat bahwa kelompok gambar uji pertama menunjukkan perbedaan signifikan dalam warna latar belakang, sedangkan kelompok kedua tidak menunjukkan perbedaan signifikan.

4.3 Eksperimentasi terhadap Perbedaan Pose Wajah Gambar Uji


Eksperimentasi dilakukan terhadap gambar uji dari orang yang sama dengan pose wajah yang berbeda dan menggunakan trainingSet3. Gambar uji yang digunakan adalah berbagai potret Dr. Rinaldi Munir.

Input

Output

	
Keterangan	
Hasil pencocokkan gambar uji adalah gambar pelatihan ke-290.	

Input

Output

<div> <div>main</div> <div>Go to Camera</div> <div>PEPSIMAN FACE RECOGNITION APP</div> <div> <div>Your test image:</div> <div>The test image is closest to:</div> </div> <div> <div>Choose training set</div> <div>File name: train_kaggle2</div> </div> <div> <div>Choose test image</div> <div>File name: test_face_2.jpg</div> </div> <div> <div>Result: 111.jpg</div> </div> <div> <div>Train Image</div> <div>Process Image</div> </div> <div> <div>Please wait some time after you press the processing button...</div> <div>Examination Time: 0.09 secs</div> </div> </div>	
Keterangan	
Hasil pencocokkan gambar uji adalah gambar pelatihan ke-111.	

Input

Output

main

Go to Camera

PEPSIMAN FACE RECOGNITION APP

Your test image:

The test image is closest to:

Choose training set

File name: train_kaggle2

Choose test image

File name: test_face_3.jpg

Result: 111.jpg

Train Image

Process Image





Please wait some time after you press the processing button...

Examination Time: 0.06 secs

Keterangan

Hasil pencocokkan gambar uji adalah gambar pelatihan ke-111.

Input



Output




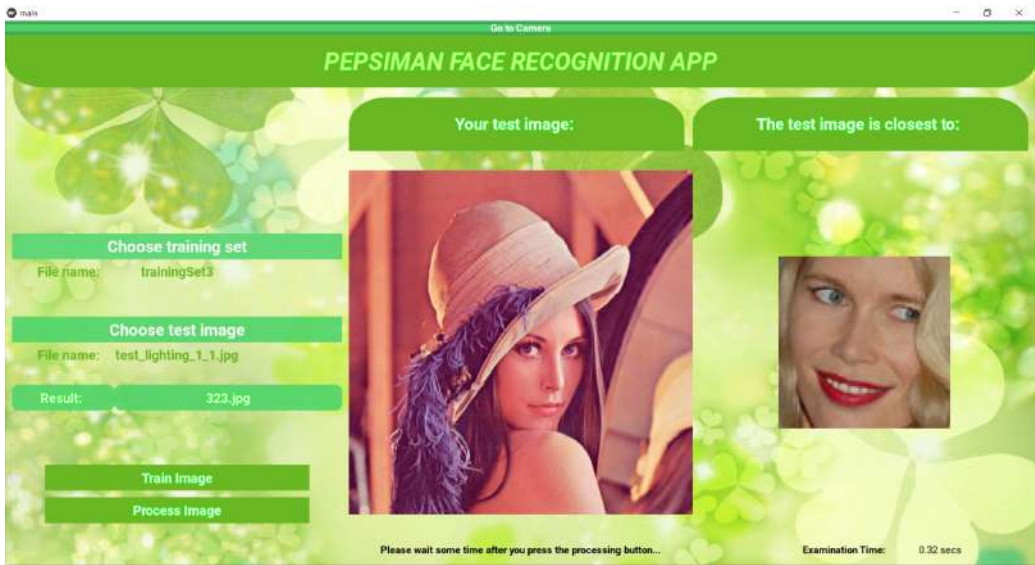
Pada hasil pengujian, kelompok uji menunjukkan hasil yang konsisten pada gambar uji kedua dan gambar uji ketiga, dan menunjukkan hasil yang berbeda pada gambar uji pertama dan keempat. Walaupun perbedaan pose wajah dapat menyebabkan hasil pencocokan yang berbeda, kami tidak dapat menentukan apakah faktor perbedaan pose wajah memiliki pengaruh yang lebih besar jika dibandingkan dengan faktor pewarnaan dan faktor lain.

4.4 Eksperimentasi terhadap Perbedaan Pencahayaan Gambar

Uji

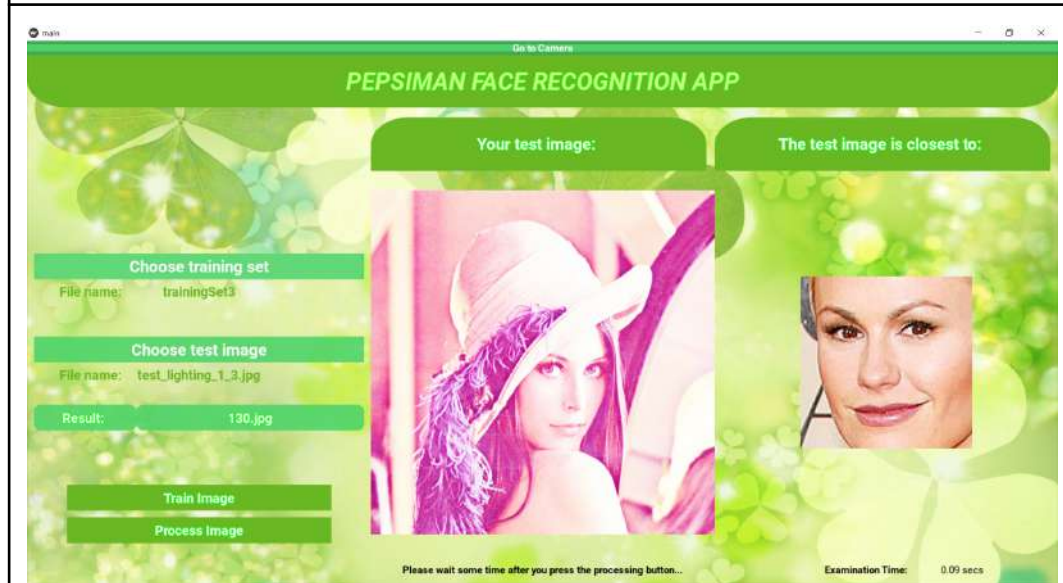
Eksperimentasi dilakukan pada tiga kelompok gambar uji dengan 5 variasi pencahayaan pada masing-masing gambar uji. Pengujian dilakukan terhadap

trainingSet3, dengan gambar uji berupa potret dari model Lenna (Lena), Albert Einstein, dan Vincent (asisten Algeo).

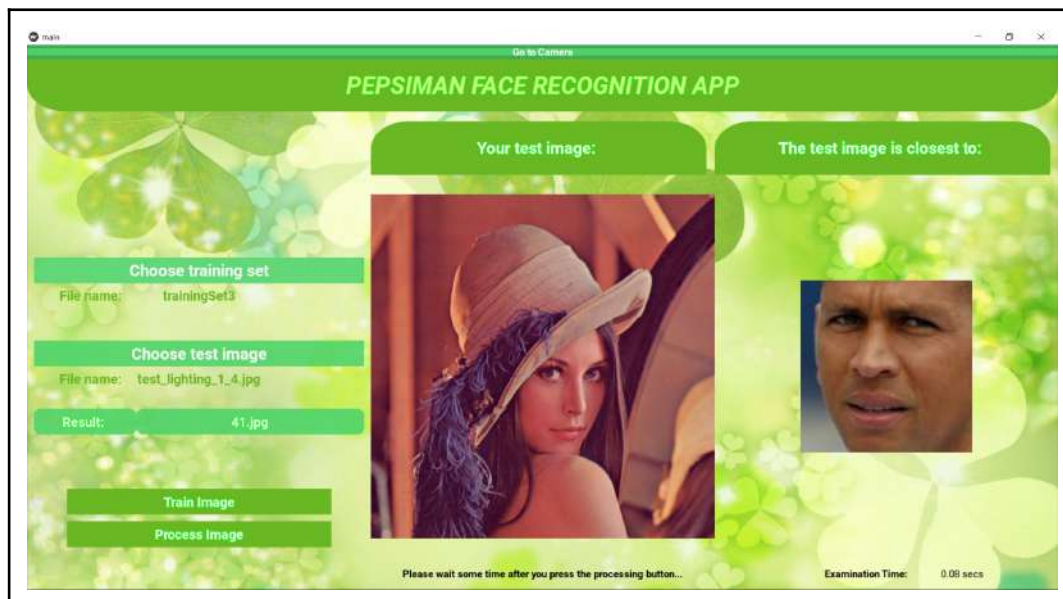
Input

Output (kasus uji: original)

Output (kasus uji: terang)



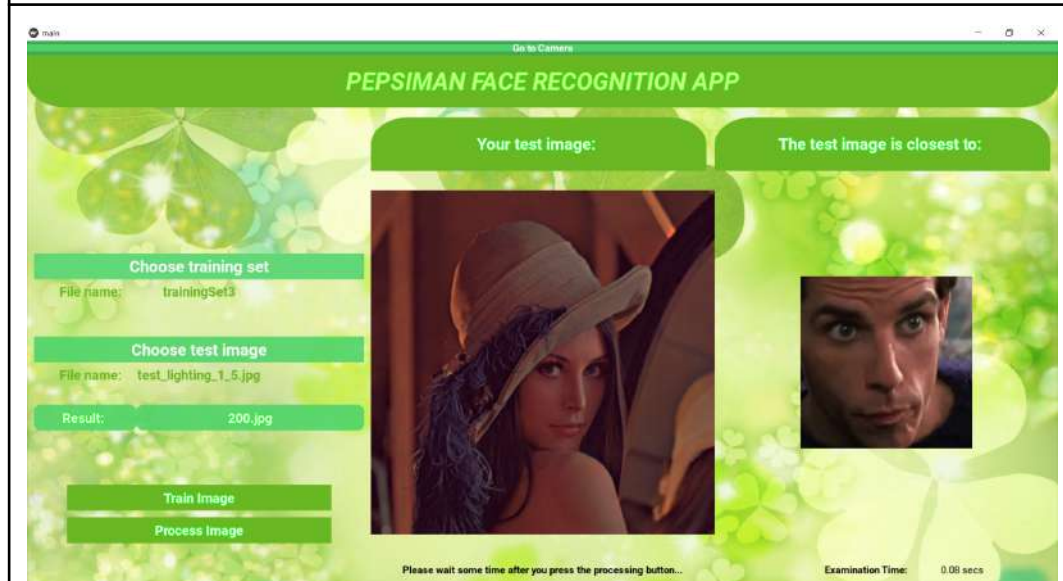
Output (kasus uji: sangat terang)



Output (kasus uji: gelap)



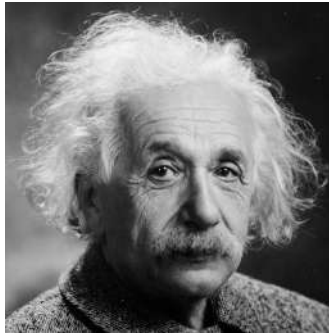
Output (kasus uji: sangat gelap)



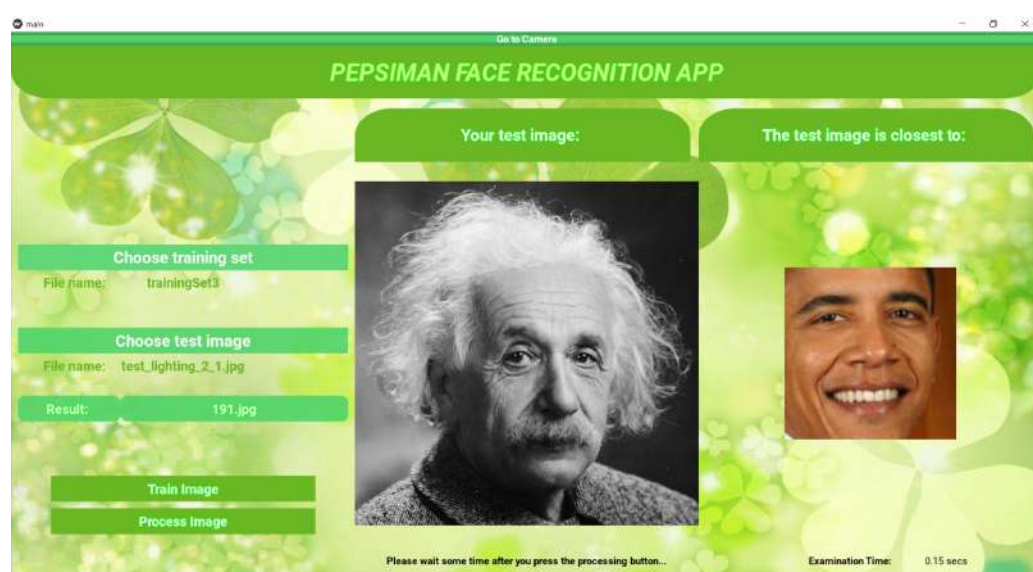
Keterangan

Perbedaan pencahayaan pada gambar uji menghasilkan hasil pencocokan gambar yang berbeda.

Input



Output (kasus uji: original)



Output (kasus uji: terang)



Output (kasus uji: sangat terang)



Output (kasus uji: gelap)



Output (kasus uji: sangat gelap)

main

Go to Camera

PEPSIMAN FACE RECOGNITION APP

Your test image:



The test image is closest to:



Choose training set

File name: trainingSet3

Choose test image

File name: test_lighting_2_5.jpg

Result: 331.jpg

Train Image

Process Image


Please wait some time after you press the processing button...

Examination Time: 0.19 secs

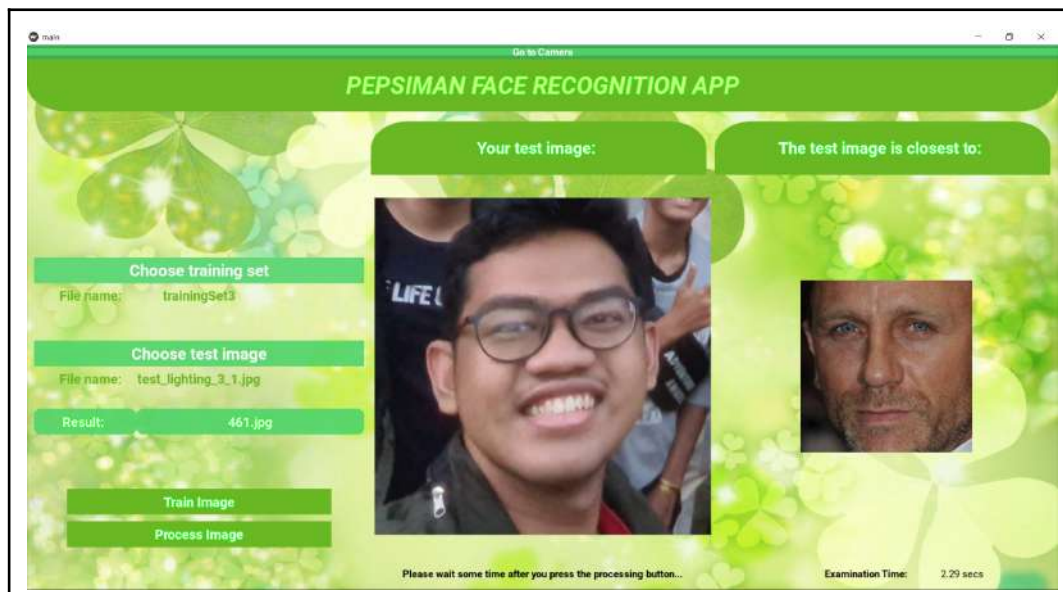
Keterangan

Sama seperti kelompok uji pertama, perbedaan dalam pencahayaan gambar uji mempengaruhi hasil pencocokan oleh gambar.

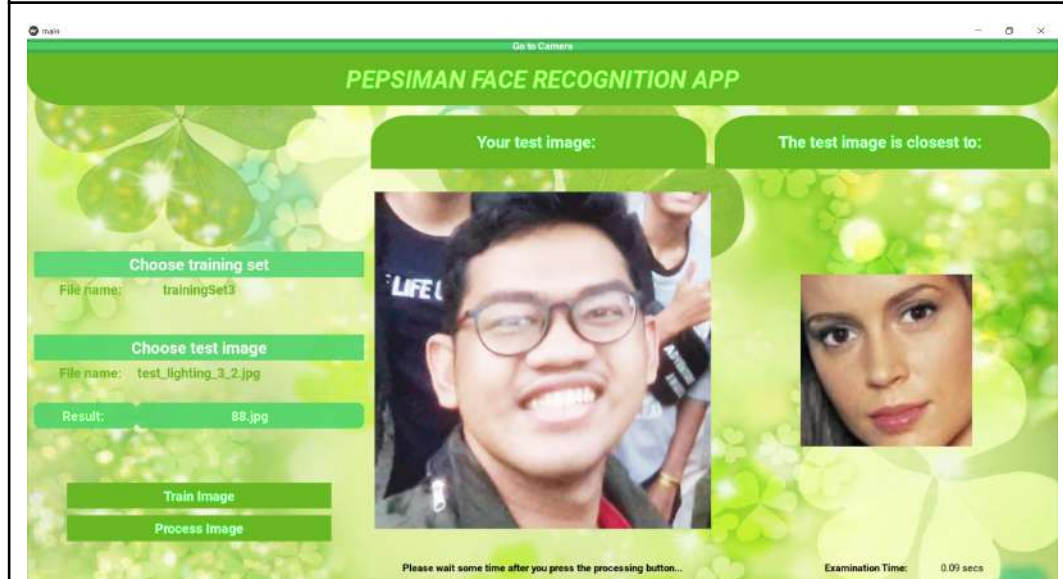
Input



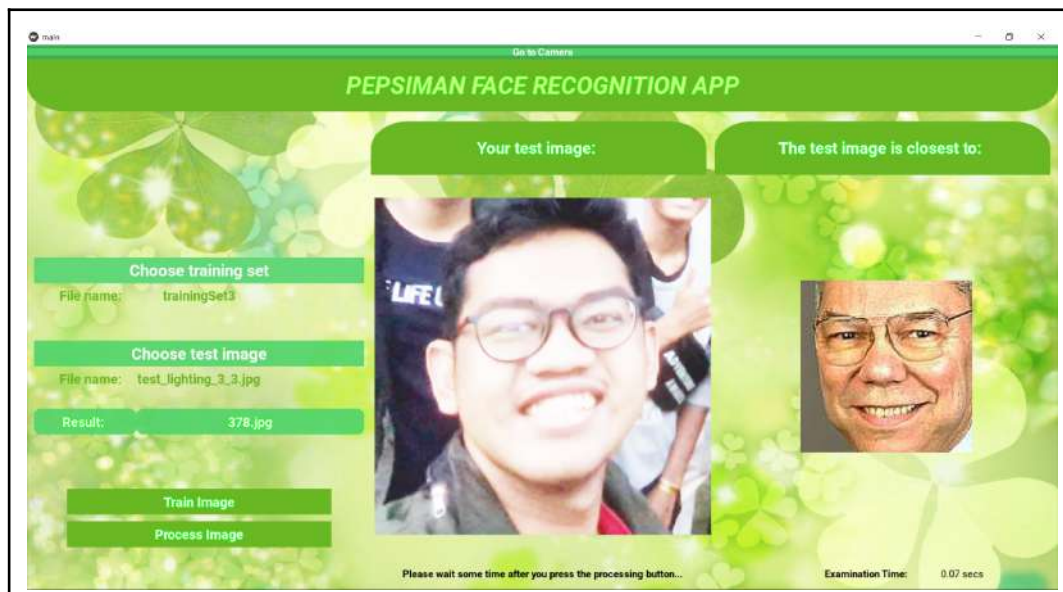
Output (kasus uji: original)



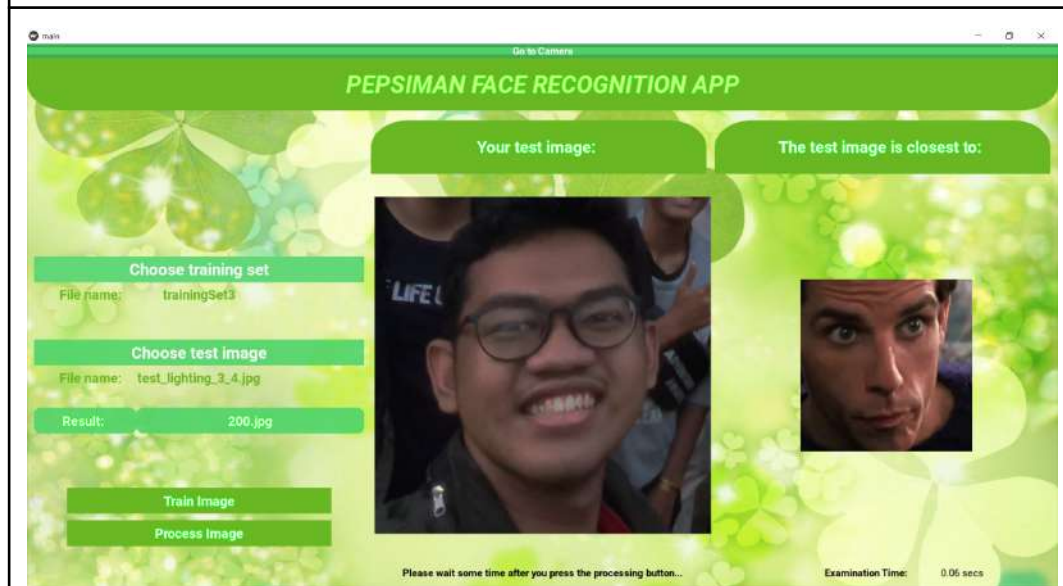
Output (kasus uji: terang)



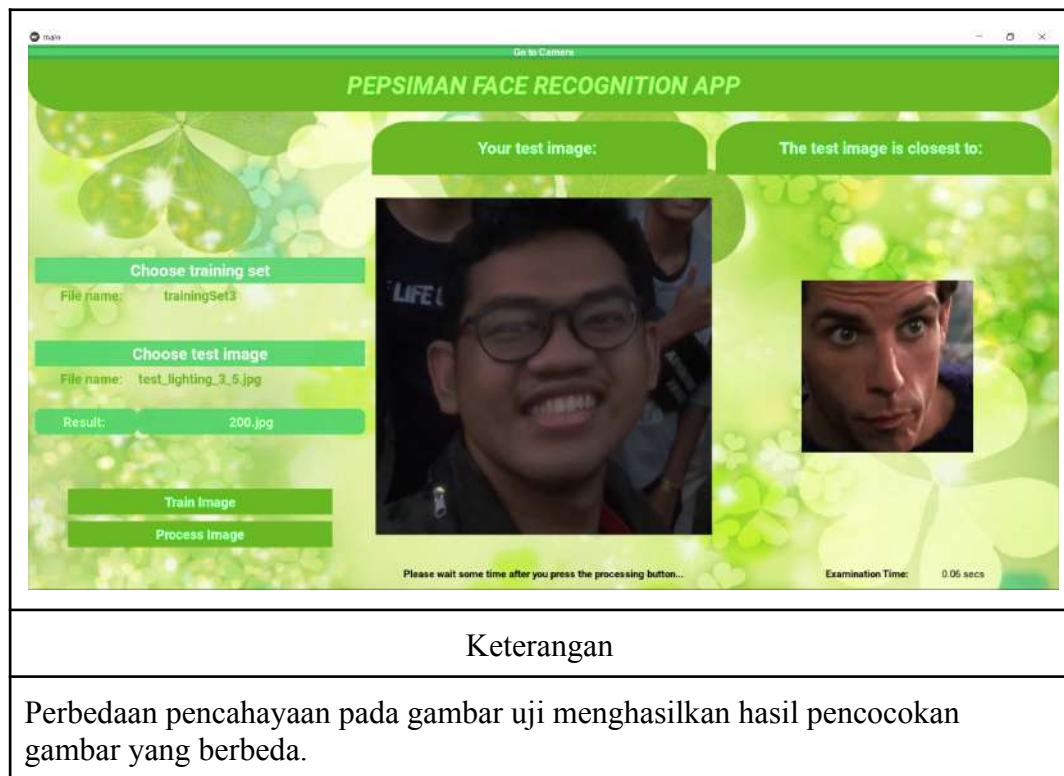
Output (kasus uji: sangat terang)



Output (kasus uji: gelap)



Output (kasus uji: sangat gelap)

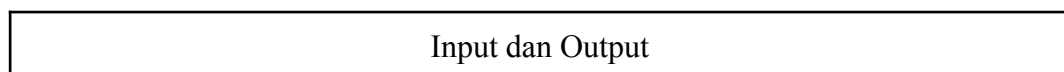
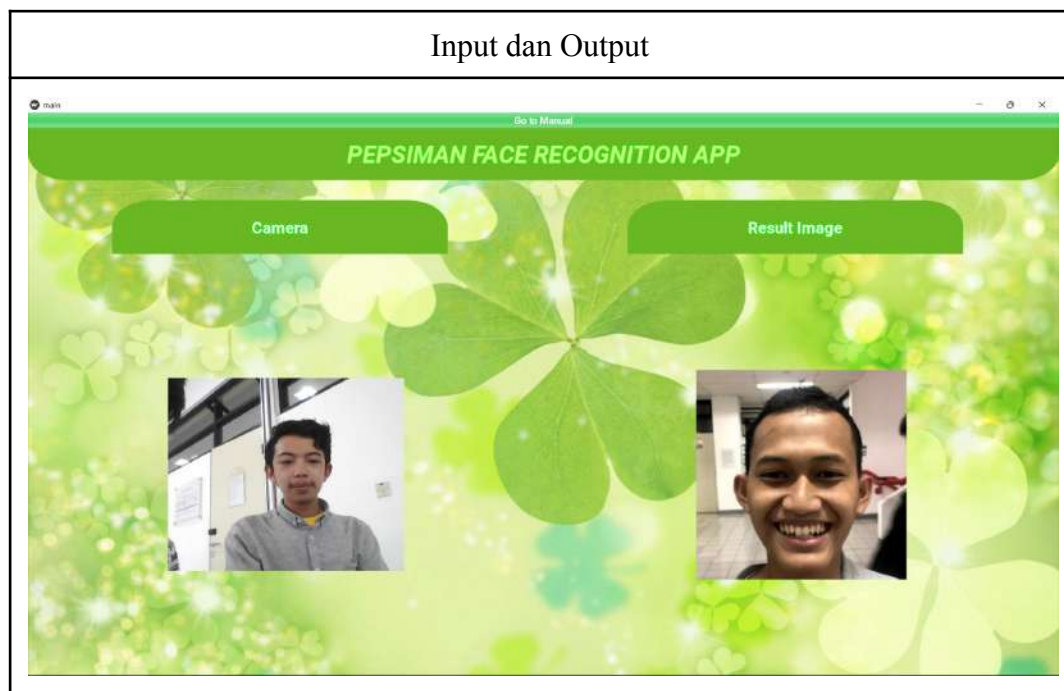


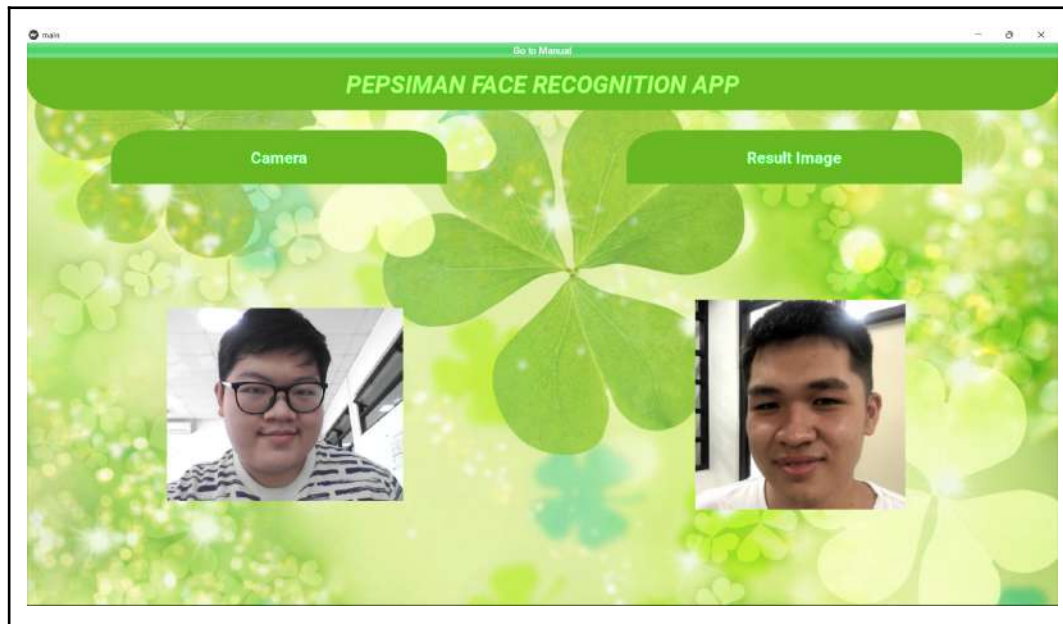
Pada kelompok gambar uji pertama, dapat diperhatikan bahwa setiap variasi pencahayaan menunjukkan hasil yang berbeda. Kelompok gambar uji kedua dan ketiga juga menunjukkan hasil yang serupa, tetapi dengan gambar uji keempat dan kelima menunjukkan hasil yang sama. Berdasarkan hasil ini, kami menyimpulkan bahwa program pengenalan wajah, terutama karya kami, bersifat sensitif terhadap cahaya.

4.5 Pengujian Fitur Kamera

Pengujian fitur kamera dilakukan menggunakan 4 wajah yang diambil langsung menggunakan fitur kamera. Pengujian menggunakan trainingSet2. Hasil pengujian pada tabel-tabel di bawah menunjukkan bahwa fitur kamera berfungsi

normal. Sebelumnya, telah dilakukan train terhadap folder trainingSet2 yang memiliki 47 gambar dan membutuhkan waktu eksekusi sekitar 3.12 detik.

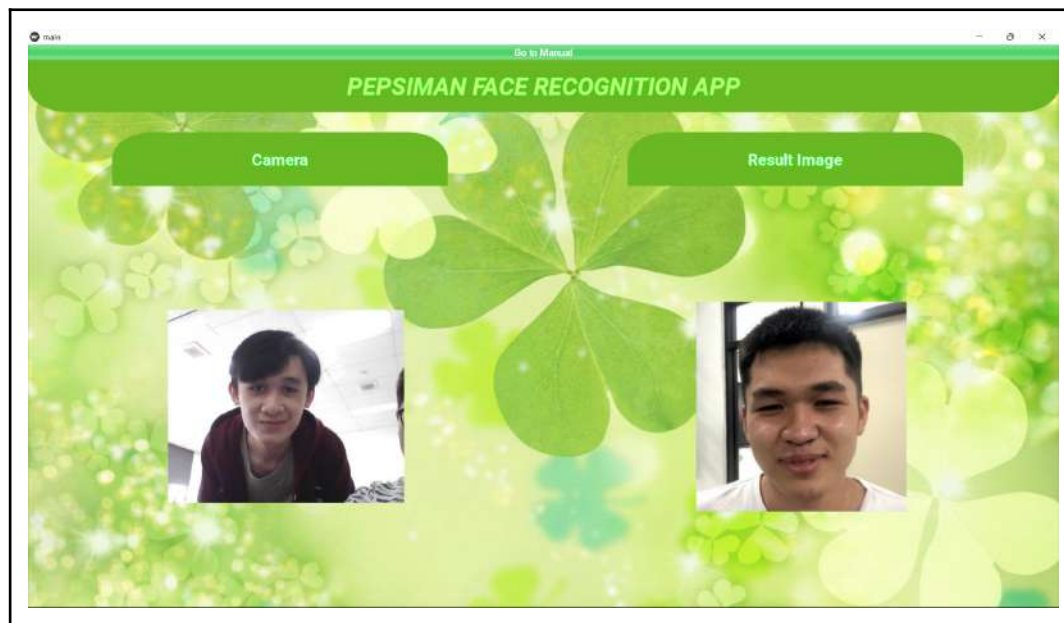




Input dan Output



Input dan Output



BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

5.1 Kesimpulan

1. Referensi yang telah saya baca menyebutkan bahwa ketika membuat pengenalan wajah menggunakan metode ini, sebaiknya dipasang *threshold* kemiripan yang membatasi seberapa dekat suatu gambar minimal untuk bisa dianggap dikenali. Jika nilai paling tinggi dianggap dikenali maka yang terjadi adalah banyaknya *false positives* yang *comical* seperti teman saya yang – menurut *software* miliknya – mirip dengan Brad Pitt.
2. Pengenalan wajah menggunakan metode *eigenface* tidaklah akurat. Sering terjadi *false positive* maupun *false negative* karena hal-hal yang tidak bergantung pada wajah seperti arah menghadapnya wajah, pencahayaan, dan lain-lain mempengaruhi bentuk *eigenface* yang dibentuk. Akibatnya, terkadang gambar dihitung “sama” walaupun sebenarnya yang sama hanyalah *lighting* dan arah wajah.

5.2 Saran

1. Program yang ditulis, meskipun seratus persen fungsional, masih dapat dirapikan baik dari penyederhanaan logika dari fungsi-fungsi di dalamnya maupun penulisan dokumentasi, *comments* yang lebih deskriptif, serta *streamlining* UI/UX.
2. Penggunaan *threading* antara Kivy dan Python dapat digunakan untuk melakukan *passing data* secara lebih efisien dan menghasilkan jalan program yang lebih mulus.

3. Spesifikasi tugas besar bisa lebih dirapikan, terutama terkait penggunaan fitur *equation* di dalam persamaan-persamaan yang dicantumkan

5.3 Refleksi

1. *Testing* pada saat pengembangan perangkat lunak sangatlah krusial untuk memastikan bahwa tidak ada bug, utamanya bug yang bersifat *subtle* sehingga sulit di-*debug*, di dalam program.
2. Penggunaan virtual environment dari Python untuk menjalankan aplikasi yang disarankan oleh beberapa referensi perlu dilakukan hati-hati karena dapat menyebabkan error dan bug seperti tidak ditemukannya modul atau python package.
3. Semakin banyak fungsi yang ditulis dan abstraksi yang dibuat, maka semakin banyak pula dokumentasi yang perlu ditulis.

DAFTAR PUSTAKA

- Anthonissen, M. (2021). *4-6 QR algorithm for computing eigenvalues*. Technische Universiteit Eindhoven. Diakses pada 8 November 2022, dari <https://www.dropbox.com/s/9byoug9odr5eaov/4-6%20qr%20algorithm.pdf?dl=0>.
- Anton, H., Rorres, C. (2014). *Elementary Linear Algebra: Applications Version*. Wiley. ISBN: 9781118434413 1118434412 9781118474228 1118474228
- Elder, J. (2022). *KIVYCODER.COM BY CODEMY.COM*. kivycoder. Diakses pada 6 November 2022, dari <https://kivycoder.com/>.
- Kivy. (2022). Kivy 2.1 Documentation, Getting Started >> Installing Kivy. Kivy Diakses pada 5 November 2022, dari <https://kivy.org/doc/stable/gettingstarted/installation.html#>
- Drury, M. (2017). *How Does A Computer Calculate Eigenvalues?*. Scatterplot Smoothers. Diakses pada 7 November 2022, dari <http://madrury.github.io/jekyll/update/statistics/2017/10/04/qr-algorithm.html>.
- Munir, R. (2020). *Nilai Eigen dan Vektor Eigen (bagian 1)*. IF2123 Aljabar Geometri - Semester I Tahun 2022/2023. Diakses pada 8 November 2022, dari <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-Nilai-Eigen-dan-Vektor-Eigen-Bagian1.pdf>.

- QuantitativeBytes. (2021). *Linear Algebra in C++ - Part 13 - Eigenvectors and eigenvalues [QR decomposition]*. YouTube [video]. Diakses pada 11 November 2022, dari <https://www.youtube.com/watch?v=tYqOrvUOMFc>.
- QuantitativeBytes. (2021). *Linear Algebra in C++ - Part 14 - Eigenvectors and eigenvalues [QR decomposition]*. YouTube [video]. Diakses pada 11 November 2022, dari <https://www.youtube.com/watch?v=tYqOrvUOMFc>.
- Slavković, M., Jevtić, D. (2012). *Face Recognition Using Eigenface Approach*. Serbian Journal of Electrical Engineering. Diakses pada 22 November 2022, dari https://www.researchgate.net/publication/260880521_Face_Recognition_Using_Eigenface_Approach.
- Yanovsky, I. (2007). *QR Decomposition with Gram-Schmidt*. University of California, Los Angeles - Mathematics. Diakses pada 9 November 2022, dari <https://www.math.ucla.edu/~yanovsky/Teaching/Math151B/handouts/GramSchmidt.pdf>.

LAMPIRAN

Link Repository Program

[Fatih20/Algeo02-21060 \(github.com\)](https://github.com/Fatih20/Algeo02-21060)

Link Video Demo Youtube

[Pepsiphile - The Eigen Pairs and Facial Recognition - YouTube](#)