Laporan Tugas Besar 2
IF 2123 Aljabar Linier dan Geometri
Aplikasi Nilai Eigen dan EigenFace pada Pengenalan Wajah (*Face Recognition*)

Dosen Pengampu:

Ir. Rila Mandala, M.Eng., Ph.D.

Disusun Oleh:

Muhammad Salman Hakim Alfarisi	13521010
Haikal Ardzi Shofiyyurrohman	13521012
Agsha Athalla Nurkareem	13521027

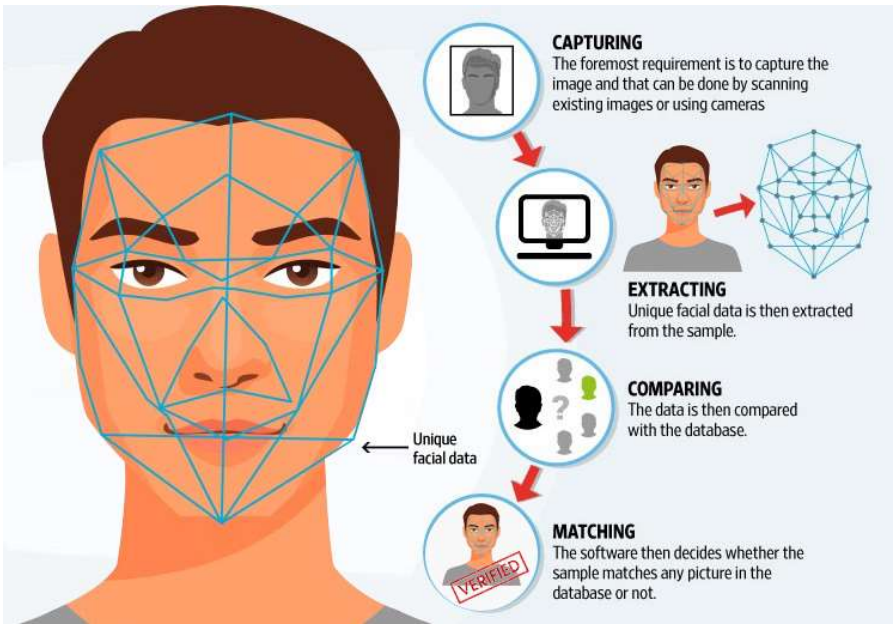
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha No. 10

Semester Ganjil Tahun 2022/2023

DAFTAR ISI

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 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.

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

   S = (Γ1, Γ2, …, ΓM) (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 x $vi = \lambda i$ x vi (5)

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

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

   $l = 1, \dots, M$

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.

   $\mu new = v$ x $\Gamma new - \Psi$ (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. $\varepsilon k = \Omega - \Omega k$ (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.

BAB II
TEORI SINGKAT

2.1. Perkalian Matriks

Perkalian matriks adalah nilai pada matriks yang bisa dihasilkan dengan cara dikalikan-nya tiap baris dengan setiap kolom yang memiliki jumlah baris yang sama. Setiap anggota matriks ini nantinya akan dikalikan dengan anggota elemen matriks lainnya. Operasi perkalian matriks ini digunakan pada berbagai pencarian nilai pada program *face recognition* seperti nilai eigen, vektor eigen, dan eigenface.

2.2. Nilai Eigen

Nilai eigen adalah nilai karakteristik dari sebuah matriks yang berukuran n x n. Nilai eigen didapatkan dari operasi $\boldsymbol{Ax = \lambda x}$ dengan $\boldsymbol{A}$ adalah matriks n x n, $\boldsymbol{x}$ adalah vektor eigen, $\boldsymbol{\lambda}$ adalah skalar yang disebut nilai eigen.

2.3. Vektor Eigen

Vektor eigen adalah vektor tidak-nol di R^n yang didapat dari operasi yang sama seperti nilai eigen. Vektor eigen menyatakan matriks kolom yang apabila dikalikan dengan sebuah matriks n x n menghasilkan vektor lain yang merupakan kelipatan vektor itu sendiri.

2.4. Eigenface

Eigenface adalah metode yang berguna untuk pengenalan dan deteksi wajah dengan menentukan variasi dari koleksi foto wajah-wajah dan menggunakan variasi tersebut untuk *encode* dan *decode* sebuah wajah dengan cara *machine learning* tanpa informasi utuh yang mampu mengurangi kerumitan dari ruang dan komputasi pada proses pengenalan wajah.

BAB III
IMPLEMENTASI PROGRAM

3.1. Struktur Program

3.1.1. Front-End

Pada *front-end* kami menggunakan library *tkinter, customtkinter, timeit, dan PIL*. *Library tkinter* digunakan untuk membuat *GUI*, sedangkan *customtkinter* digunakan untuk memperindah tampilan. *Library timeit* digunakan untuk mendapatkan *execution time*. *Library PIL* digunakan untuk mendapatkan foto yang ingin ditampilkan. Program *front-end* diimplementasikan pada file GUI.py. Pada GUI.py kami membuat 3 fungsi, yakni fungsi *open_folder*, *open_image*, dan *main*.

Fungsi *open_folder* digunakan untuk memperoleh folder dataset yang dipilih. Fungsi *open_image* digunakan untuk memperoleh file foto yang dipilih, sedangkan fungsi main digunakan untuk memanggil algoritma perhitungan *face recognition*.

3.1.2. Back-End

Pada *back-end* kami menggunakan bahasa python dan menggunakan beberapa *library* pendukung seperti *Numpy, OpenCV,* dan *OS* untuk pengolahan citra menjadi matriks, dan operasi-operasi untuk pengolahan matriks. Pada *numpy* kami menggunakan *numpy mean* untuk mencari nilai rata-rata *dataset*, *numpy concatenate* untuk mengkonkat matriks dari hasil pengurangan tiap matriks *training image* dengan nilai rata-rata, dan dilanjutkan dengan *numpy covarian* untuk mendapatkan matriks kovarian dari dataset tersebut. Pada openCV kami menggunakan berbagai kakas untuk mengolah citra mulai dari mengkonversi citra dari berwarna menjadi hitam-putih, mengubah ukuran dari citra dataset dan *imagetest* menjadi ukuran yang sama (contohnya 256x256 untuk semua citra) agar bisa melakukan perbandingan melalui perhitungan selisih jarak antara *imagetest* dengan setiap nilai eigenface.

Pada *back-end* kami membuat 6 file program dengan *main.py* sebagai program utama yang akan dijalankan pada *front-end*, *AlgortimaEigen.py* untuk mendapatkan nilai eigenvector dan eigenvalue dari nilai kovarian pada pengolahan matriks dataset, *covarian.py* untuk mendapatkan matriks covarian dari konkatenasi seluruh matirks selisih dari dataset, *img_processing.py* digunakan untuk mengolah citra menjadi matriks yang akan digunakan untuk mencari eigenface, *mean.py* digunakan untuk mendapatkan nilai rata-rata dari dataset, dan *operation.py* yang digunakan untuk kebutuhan pengolahan dan operasi-operasi aritmatik dari pengolhan dataset. Secara rinci fungsi-fungsi yang kami dugnakan adalah sebagai berikut:

- DekomposisiQR(matrix): mendekomposisi suatu matriks untuk mendapatkan nilai dari matrix Q dan matriks R,
- EigenDariQR(matrix): mendapatkan nilai dari eigenvalue dan eigenvector dengan menggunakan fungsi DekomposisiQR yang diiterasi berulang-ulang.
- matrix_covarian(matrix): mendapatkan nilai dari matrix kovarian yang berasal dari transpos konkatenasi matriks dari seluruh selisih dataset yang dikali dengan konkatenasi matriks dari seluruh selisih dataset.

- list_files(directory): mendapatkan daftar dari file dataset yang dibutuhkan.
- get_image(path): mendapatkan daftar matrix warna hitam-putih dari citra yang akan digunakan untuk data *face recognition*.
- resize_image(images, size): merubah ukuran dari matriks citra.
- Mean(listMatriksGambar): mendapatkan nilai rata-rata dari seluruh dataset.
- JumlahMatriks(matriksGambar1, matriksGambar2): menjumlahkan 2 buah matriks dengan menggunakan *numpy add.*
- KurangMatriks(matriksGambar1, matriksGambar2): mengurangkan 2 buah matriks dengan menggunakan *numpy substract*.
- KaliMatriks(matriksGambar1, matriksGambar2): melakukan operasi perkalian matriks dengan menggunakan *numpy dot*.
- TransposeMatriks(matriksGambar): melakukan transpos matriks dengan menggunakan *numpy trnaspose*.
- printMatriks(matriksGambar): menampilkan matriks.
- concat(matriksGambar1, matriksGambar2): mengkonkatenasi kesamping dua buah matriks.
- concatBawah(matriksGambar1, matriksgambar2): mengkonkatenasi kebawah dua buah matriks.
- concatAllimage(himpunanGambar): mengkonkatenasi seluruh elemen himpunan matriks kesamping.
- concatAllImageBawah(himpunanGambar): mengkonkatenasi seluruh elemen himpunan matriks kebawah.
- flattenImage(himpunanGambar): merubah dimensi ukuran menjadi pipih (NxN menjadi 1xN*N) dari himpunan serta mengkonkatenasi seluruh elemen himpunan gambar.
- getMinIndex(list): mendapatkan indeks dengan nilai elemen minimum.
- normalized(matrix): mendapatkan nilai normalisasi dari suatu matriks.
- normalizedArray(matriks): mendapatkan nilai normalisasi dari suatu array.

3.3. Garis Besar Program

Program diawali dengan user menjalankan program. User diminta untuk memasukan dataset sebagai *training image* dan *test image* sebagai citra yang akan dicari kemiripannya. Setelah user menginput kedua input tersebut, lalu user akan menekan tombol start dan program pun mulai menjalankan proses pengenalan wajah.

Proses pengenalan wajah menggunakan algoritma eigenface, dengan langkah-langkah algoritma yang dapat dilihat pada BAB I. Lalu selanjutnya tahap pengenalan wajah dimulai dengan menerapkan cara pada tahapan pertama perhitungan eigenface untuk mendapatkan nilai eigen dari image tersebut dengan rumus sebagai berikut:

$$\mu new = v \times \Gamma new - \Psi \quad (7)$$

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

Γnew = test image.

Selanjutnya gunakan metode *euclidean distance* untuk mencari jarak terpendek antara nilai eigen dari *training image* dalam database dengan nilai eigen dari *image test*.

$$\varepsilon k = \Omega - \Omega k$$

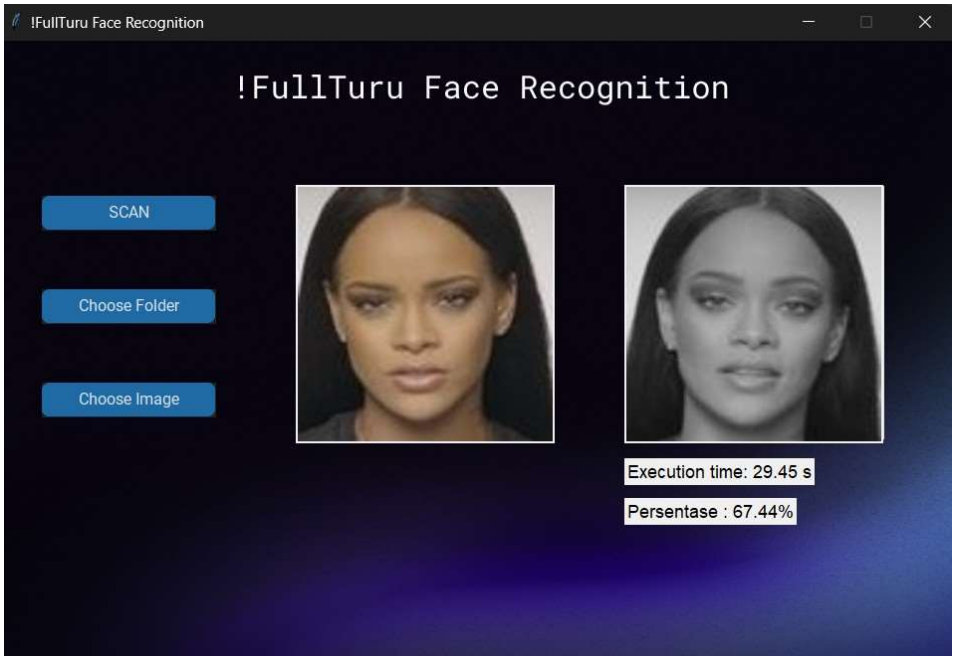
lalu akan didapat citra termirip, tetapi jarak minimal perlu dibatasi karena jarak minimum dapat didapatkan dengan foto yang mana saja sehingga dibatasi nilai jaraknya dengan rumus:

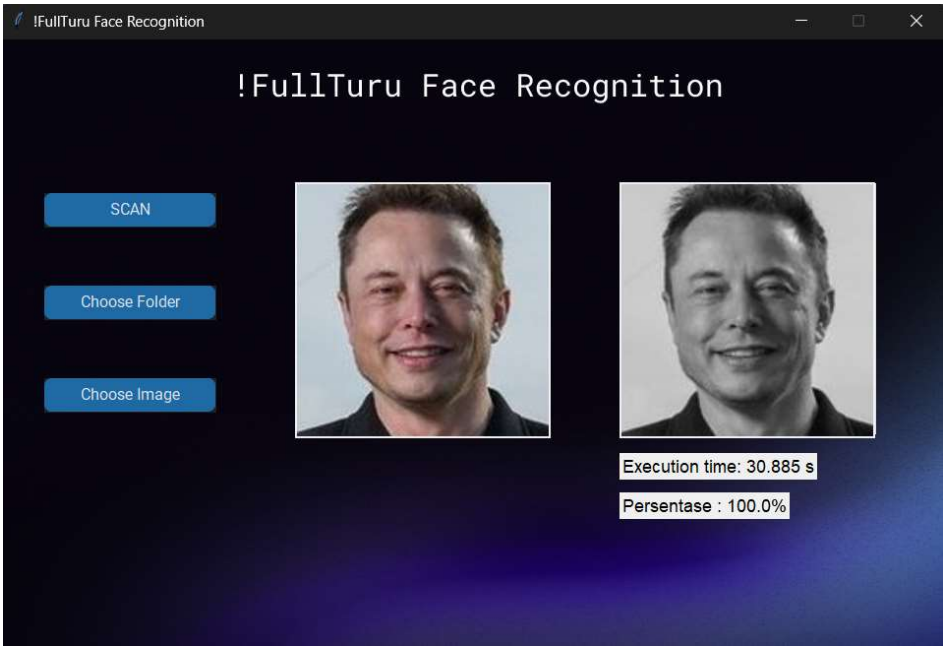
Limit = 0.8*max(euclideanDistance)

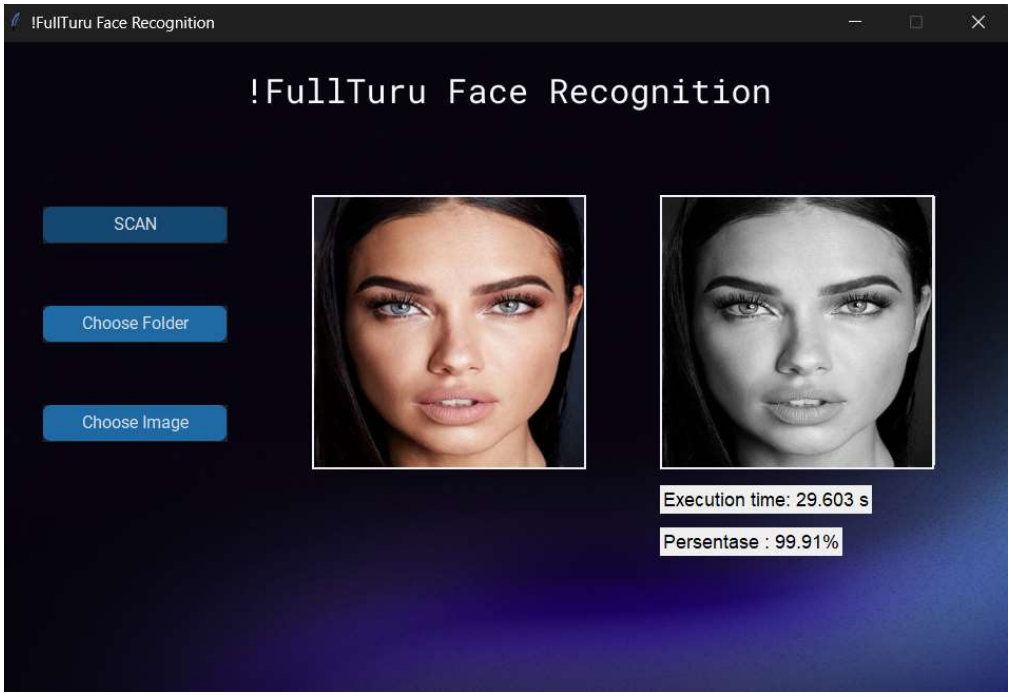
Pada akhir proses user akan mendapatkan citra terdekat dari *test image*, perkiraan waktu proses, dan persentase kemiripan dengan rumus:

percentage=((max-min)/max)*100%,
percentage<Limit.

BAB IV
EKSPERIMEN






!FullTuru Face Recognition
!FullTuru Face Recognition
SCAN
Choose Folder
Choose Image
Execution time: 29.603 s
Persentase : 99.91%

BAB V
KESIMPULAN, SARAN, DAN REFLEKSI

5.1. Kesimpulan

Dengan algoritma eigenface kita dapat mengenali wajah seseorang yang didapati dari nilai terkecil *euclidean distance* dari citra seseorang tersebut dengan seluruh database, dengan batasan nilai minimum jaraknya 80% dari maksimum jarak euclidean.

5.2. Saran

- Sebelum melakukan algoritma eigenface, citra harus dipastikan memiliki struktur wajah atau tidak.

5.3. Refleksi

Dari progam yang kami buat kami memahami bahwa proses pengenalan wajah secara otomatis dengan komputasi dapat dilaksanakan dengan menggunakan algoritma eigenface.

DAFTAR REFERENSI

https://www.gramedia.com/literasi/perkalian-matriks/#Pengertian_Perkalian_Matriks

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

https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184

LAMPIRAN

Link github: *https://github.com/archmans/Algeo02-21010*