

LAPORAN
TUGAS BESAR 2 IF2123 ALJABAR LINIER DAN
GEOMETRI
APLIKASI NILAI EIGEN DAN EIGENFACE PADA
PENGENALAN WAJAH (FACE RECOGNITION)



oleh

| | |
|---------------------------------|-----------------|
| Fakhri Muhammad Mahendra | 13521045 |
| Muhamad Aji Wibisono | 13521095 |
| Michael Jonathan Halim | 13521124 |

Kelompok

#TeamOnodera

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2022

DAFTAR ISI

| | |
|---|-----------|
| DAFTAR ISI | i |
| BAB 1 | 1 |
| DESKRIPSI MASALAH | 1 |
| 1.1 Spesifikasi Tugas | 1 |
| 1.1.1 Abstraksi | 1 |
| 1.1.2 Algoritma Eigenface | 2 |
| 1.1.3 Tahapan Pengenalan Wajah | 3 |
| 1.1.4 Penggunaan Program | 4 |
| BAB II | 6 |
| TEORI SINGKAT | 6 |
| 2.1 Perkalian Matriks | 6 |
| 2.2 Vektor Eigen dan Nilai Eigen | 7 |
| 2.3 Rotasi Givens | 7 |
| 2.6 Dekomposisi QR | 9 |
| 2.7 Metode Givens | 9 |
| 2.8 Matriks Hessenberg Atas | 9 |
| 2.9 Algoritma QR | 10 |
| 2.10 Matriks Kovarian | 11 |
| 2.11 Muka Eigen | 11 |
| BAB III | 12 |
| IMPLEMENTASI PUSTAKA DAN PROGRAM DALAM PYTHON | 12 |
| 3.1 Tech Stack | 12 |
| 3.2 Module Dataset To Matrix (imageToMatrix.py) | 12 |
| 3.3 Module EigenFace (eigenface.py, step6eigenface.py, combination.py) | 13 |
| 3.4 Givens Matrix Calculation (givens.py) | 16 |
| 3.5 Hessenberg Form (hessenberg.py) | 17 |
| 3.6 QR Iteration (qr.py) | 18 |
| 3.7 Eigenvalue (eigenvalue.py) | 18 |
| 3.8 Eigenvector (eigenvector.py) | 19 |
| 3.9 Euclidean Distance (euclidean.py) | 19 |
| 3.10 Main Algorithm (main_algo.py) | 20 |
| 3.11 GUI (main.py) | 21 |

| | |
|--|-----------|
| BAB IV | 37 |
| EXPERIMEN | 37 |
| 4.1 Eksperimen Terhadap Gambar yang Sama | 37 |
| 4.2 Eksperimen Terhadap Gambar Orang yang Sama, Namun di Luar Dataset | 38 |
| 4.3 Eksperimen Terhadap Gambar Acak | 39 |
| BAB 5 | 41 |
| KESIMPULAN, SARAN, DAN REFLEKSI | 41 |
| 5.1 Kesimpulan | 41 |
| 5.2 Saran | 41 |
| 5.3 Refleksi | 42 |
| DAFTAR PUSTAKA | 43 |
| LAMPIRAN | 44 |

BAB 1

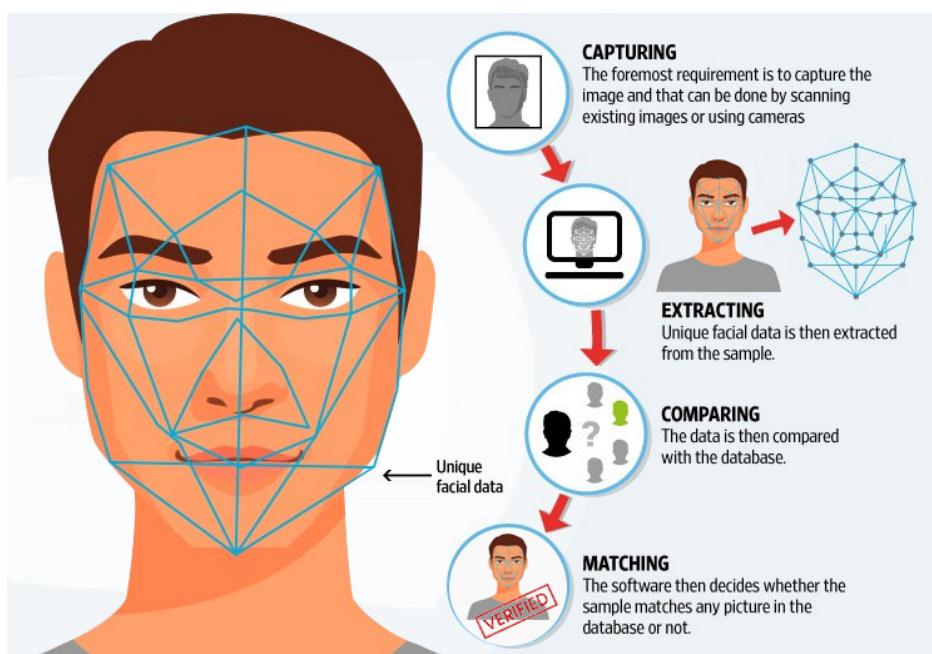
DESKRIPSI MASALAH

1.1 Spesifikasi Tugas

1.1.1 Abstraksi

Pengenalan wajah (Face Recognition) adalah teknologi biometrik yang bisa dipakai untuk mengidentifikasi wajah seseorang untuk berbagai kepentingan khususnya keamanan. Program pengenalan wajah melibatkan kumpulan citra wajah yang sudah disimpan pada database lalu berdasarkan kumpulan citra wajah tersebut, program dapat mempelajari bentuk wajah lalu mencocokkan antara kumpulan citra wajah yang sudah dipelajari dengan citra yang akan diidentifikasi.

Alur proses sebuah sistem pengenalan wajah diperlihatkan pada Gambar 1.



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

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

Terdapat berbagai teknik untuk memeriksa citra wajah dari kumpulan citra yang sudah diketahui seperti jarak Euclidean dan cosine similarity, principal component analysis (PCA), serta Eigenface. Pada Tugas ini, akan dibuat sebuah program pengenalan wajah menggunakan Eigenface.

Sekumpulan citra wajah akan digunakan dengan representasi matriks. Dari representasi matriks tersebut akan dihitung sebuah matriks Eigenface. Program pengenalan wajah dapat dibagi menjadi 2 tahap berbeda yaitu tahap training dan pencocokan. Pada tahap training, akan diberikan kumpulan data set berupa citra wajah. Citra wajah tersebut akan dinormalisasi dari RGB ke Grayscale (matriks), hasil normalisasi akan digunakan dalam perhitungan eigenface. Seperti namanya, matriks eigenface menggunakan eigenvector dalam pembentukannya. Berikut merupakan langkah rinci dalam pembentukan eigenface.

1.1.2 Algoritma Eigenface

1. Langkah pertama adalah menyiapkan data dengan membuat suatu himpunan S yang terdiri dari seluruh training image, $(\Gamma_1, \Gamma_2, \dots, \Gamma_M)$ $S = (\Gamma_1, \Gamma_2, \dots, \Gamma_M)$ (1)
2. Langkah kedua adalah ambil nilai rata-rata atau mean (Ψ) (2)
3. Langkah ketiga kemudian cari selisih (Φ) antara nilai training image (Γ_i) dengan nilai tengah (Ψ) $\phi_i = \Gamma_i - \Psi$ (3)
4. Langkah keempat adalah menghitung nilai matriks kovarian (C) (4)
5. Langkah kelima menghitung eigenvalue (λ) dan eigenvector (v) dari matriks kovarian (C) $C \times v_i = \lambda_i \times v_i$ (5)

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

1.1.3 Tahapan Pengenalan Wajah

1. Sebuah image wajah baru atau test face (Γ_{new}) akan dicoba untuk dikenali, pertama terapkan cara pada tahapan pertama perhitungan eigenface untuk mendapatkan nilai eigen dari image tersebut. $\mu_{new} = v \times \Gamma_{new} - \Psi \quad (7)$ $\Omega = \mu_1, \mu_2, \dots, \mu_M$
2. Gunakan metode euclidean distance untuk mencari jarak (distance) terpendek antara nilai eigen dari training image dalam database dengan nilai eigen dari image testface. $\varepsilon_k = \Omega - \Omega_k \quad (8)$

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 dari sebuah citra wajah tidak perlu anda buat lagi, tetapi menggunakan fungsi ekstraksi yang sudah tersedia di dalam library. Fungsi Eigen dilarang import dari library dan harus diimplementasikan, sedangkan untuk operasi matriks lainnya silahkan menggunakan library.

Kode program untuk ekstraksi fitur dapat dibaca pada artikel ini: Feature extraction and similar image search with OpenCV for newbies, pada laman: <https://medium.com/machine-learning-world/feature-extraction-and-similar-image-searchwith-opencv-for-newbies-3c59796bf774>

Nilai batas kemiripan citra test face dapat ditentukan oleh pembuat program melalui percobaan. Berikut merupakan referensi pengenalan wajah dengan metode eigenface:

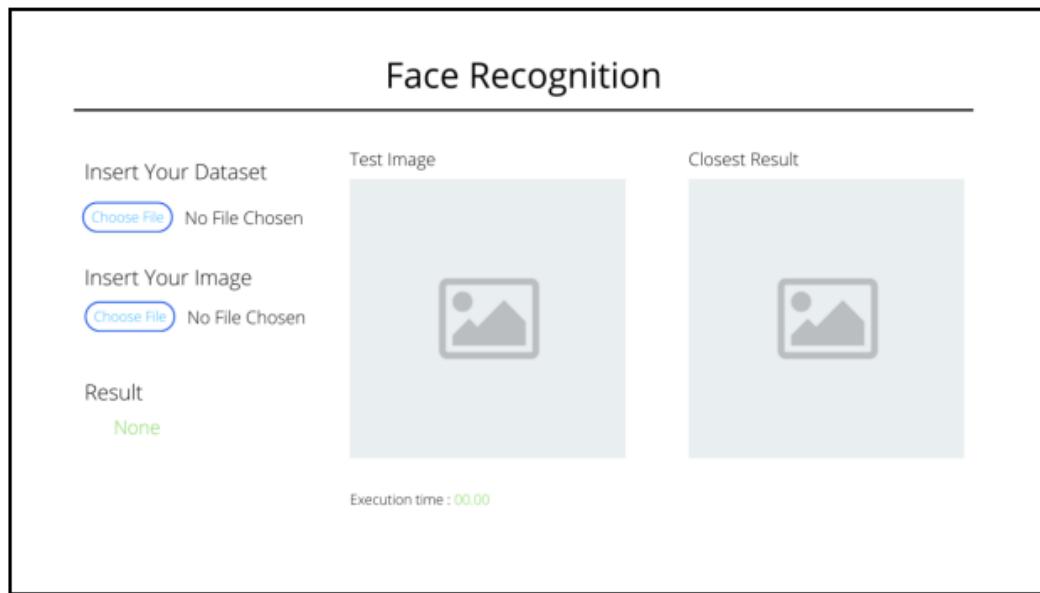
<https://jurnal.untan.ac.id/index.php/jcskommipa/article/download/9727/9500>

<https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/>

1.1.4 Penggunaan Program

Berikut ini adalah 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. Tampilan layout dari aplikasi web yang akan dibangun kurang lebih adalah sebagai berikut. Anda dapat mengubah layout selama layout masih terdiri dari komponen yang sama.



Gambar 3. Contoh tampilan layout dari aplikasi web yang dibangun.

Catatan: Warna biru menunjukkan komponen yang dapat di klik. Warna hijau menunjukkan luaran yang didapat dari hasil eksekusi.

Anda dapat menambahkan menu lainnya, gambar, logo, dan sebagainya. Tampilan GUI dibuat semenarik mungkin selama mencakup seluruh informasi pada layout yang diberikan di atas. Kreativitas menjadi salah satu komponen penilaian.

BAB II

TEORI SINGKAT

2.1 Perkalian Matriks

Dalam matematika, perkalian matriks adalah suatu operasi biner dari dua matriks yang menghasilkan sebuah matriks. Agar dua matriks dapat dikalikan, banyaknya kolom pada matriks pertama harus sama dengan banyaknya baris pada matriks kedua. Matriks hasil perkalian keduanya, akan memiliki baris sebanyak baris matriks pertama, dan kolom sebanyak kolom matriks kedua. Perkalian matriks A dan B dinyatakan sebagai AB.

Jika A adalah matriks berukuran $m \times n$ dan B adalah matriks berukuran $n \times p$, dengan elemen-elemen sebagai berikut,

$$A = [a_{ij}] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

$$B = [b_{ij}] = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix}$$

Hasil perkalian kedua matriks tersebut, $C = AB$ (dinyatakan tanpa menggunakan tanda kali atau titik), adalah sebuah matriks berukuran $m \times p$.

$$C = [c_{ij}] = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{bmatrix}$$

dengan setiap entri pada matriks C didefinisikan sebagai

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

2.2 Vektor Eigen dan Nilai Eigen

Jika A adalah matrix $n \times n$, maka vektor tak nol x di \mathbb{R}^n disebut vektor eigen dari A jika Ax adalah kelipatan skalar dari x ; yaitu

$$Ax = \lambda x$$

untuk suatu skalar λ . Skalar λ disebut sebagai nilai eigen dari A , dan x didefinisikan sebagai vektor eigen yang berkorespondensi dengan λ .

2.3 Rotasi Givens

Rotasi Givens adalah rotasi pada bidang yang bisa di-span oleh dua axis koordinat. Jika suatu vektor dirotasikan sebesar θ radian di bidang (i, j) maka rotasi given bisa direpresentasikan matriks dengan bentuk:

$$G(i, j, \theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

dimana $c = \cos(\theta)$ dan $s = \sin(\theta)$ muncul pada perpotongan dari baris dan kolom ke-i dan ke-j.

2.4 Matriks Ortogonal

Sebuah matriks A disebut ortogonal jika transpos dari matriks tersebut memiliki hasil yang sama dengan inversnya. yaitu jika

$$A^{-1} = A^T$$

atau, secara ekivalen, jika

$$AA^T = A^TA = I$$

2.5 Matriks Segitiga Atas

Sebuah matriks persegi disebut matriks segitiga atas jika semua entri diatas diagonal utamanya adalah nol. Sebagai contoh

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \cdots & u_{1,n} \\ & u_{2,2} & u_{2,3} & \cdots & u_{2,n} \\ & & \ddots & \ddots & \vdots \\ & & & \ddots & u_{n-1,n} \\ 0 & & & & u_{n,n} \end{bmatrix}$$

U merupakan matriks segitiga atas.

2.6 Dekomposisi QR

Dekomposisi QR adalah dekomposisi dari suatu matrix A ke hasil kali nya sehingga

$$A = QR$$

Dimana Q adalah matriks ortogonal dan R adalah matriks segitiga atas. Salah satu cara untuk melakukan dekomposisi QR adalah dengan metode givens.

2.7 Metode Givens

Metode Givens adalah salah satu metode untuk mendekomposisi suatu matriks persegi A menjadi bagian ortogonal (Q) dan bagian segitiga atas (R) sehingga $A = QR$.

Untuk setiap langkah pada metode Givens, dua baris pada matriks yang ingin didekomposisi akan diputar untuk mengeliminasi salah satu entri dari matriks yang sedang di dekomposisi (A). Ketika setiap entri dibawah diagonal utama sudah tereliminasi, maka matriks tersebut merupakan matriks segitiga atas R dari dekomposisi A, dan jika matriks rotasi givens pada tiap langkah diakumulasi, maka hasil kalinya akan menjadi matriks ortogonal Q dari dekomposisi A.

2.8 Matriks Hessenberg Atas

Matrix Hessenberg atas adalah matriks yang memiliki nol sebagai entri dibawah subdiagonal pertama. Sebagai contoh

$$H_n = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & \cdots & h_{1,n} \\ h_{2,1} & h_{2,2} & h_{2,3} & \cdots & h_{2,n} \\ 0 & h_{3,2} & h_{3,3} & \cdots & h_{3,n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{n,n-1} & h_{n,n} \end{bmatrix}$$

merupakan matriks hessenberg atas.

2.9 Algoritma QR

Algoritma QR merupakan sebuah algoritma iterasi untuk mencari hampiran dari nilai eigen dan vektor eigen dari suatu matriks persegi A. Dengan A adalah matriks persegi yang ingin dicari nilai eigen dan vektor eigennya, lakukan

Ambil kondisi awal $A_0 = A$

Pada tahap ke-k lakukan dekomposisi QR sehingga

$$A_k = Q_k R_k$$

Hitung nilai iterasi selanjutnya dengan

$$A_{k+1} = R_k Q_k$$

Lakukan iterasi berulang kali hingga tiap entri di subdiagonal pertama nilai nya mendekati nol. Agar subdiagonal *converge* lebih cepat, ubah kondisi awal dengan A_H , dimana A_H adalah matriks Hessenberg, hasil transformasi dari matriks A.

Diagonal utama dari matriks A_k setelah k iterasi adalah kumpulan hampiran nilai eigen dari matriks awal A. Untuk mendapat kumpulan vektor eigen, dapat dilakukan dengan mengakumulasi nilai Q_k pada tiap iterasi.

2.10 Matriks Kovarian

Matriks kovarian adalah sebuah matriks persegi yang memberikan kovarian dari setiap pasangan elemen dari suatu vector acak.

2.11 Muka Eigen

Muka eigen adalah nama dari himpunan vektor eigen yang digunakan untuk masalah *computer vision* dari pengenalan wajah manusia.

vektor-vektor eigen tersebut diturunkan dari matriks kovarian dari kumpulan wajah-wajah yang dijadikan dataset.

BAB III

IMPLEMENTASI PUSTAKA DAN PROGRAM DALAM PYTHON

3.1 Tech Stack

Untuk program pengenal wajah ini, kami gunakan bahasa pemrograman python dan memanfaatkan beberapa library yang tersedia untuk mendukung program ini menjadi lebih efisien dan baik seperti numpy untuk pemrosesan list, OpenCV untuk pemrosesan gambar, dan tkinter untuk pembuatan GUI aplikasi.

3.2 Module Dataset To Matrix (imageToMatrix.py)

```
def FolderImageToListOfMatrix(folder):
    """
        Function to extract images from dataset to array of
        matrix

    Args:
        folder (string): pathname to folder of dataset

    Returns:
        listOfMatrixFace (array): list of matrix face that
        has been resized and greyed

        listOffFixMatrixFace(array): list of matrix face in
        RGB
    """

def reshapeImage(face):
    """
        Reshape image matrix from 1 x 256^2 to 256x256

    Args:
        face (array): image in shape of flatten

    Returns:
        shape (matrix): image in shape of 256x256
    """
```

3.3 Module EigenFace (eigenface.py, step6eigenface.py, combination.py)

```
def MakeMeanFace(listOfMatrixFace):
    """
        Function to make mean face

    Args:
        listOfMatrixFace (array): List of matrix faces

    Returns:
        mean (array): mean face
    """

def calculateDifference(listOfMatrixFace, meanFace):
    """
        Finding the difference for each face matrix

    Args:
        listOfMatrixFace (array): list of matrix faces
        meanFace (array): mean face

    Returns:
        difference (matrix): the difference of matrix face
        and mean face
    """

def calculateCovariance(difference):
    """
        Function to calculate covariance

    Args:
        difference (matrix): the difference of matrix face
        and mean face

    Returns:
        covariance (matrix): the covariance matrix
    """

def EigenFaces(eigen_vectors, k, difference):
    """
        Compute the eigenfaces from eigen vectors and
        difference
```

```
Args:  
    eigen_vectors (array): list of eigen vectors  
    k (integer): the amount of eigen vector  
    difference (matrix): the difference of matrix face  
and mean face
```

```
Returns:  
    listOfEigenFace (array): list of eigen faces  
    """
```

```
def imageToMatrix(image):  
    """  
        Function to extract images from dataset to array of  
matrix
```

```
Args:  
    image (file): the test image
```

```
Returns:  
    result (array): the image in shape of array 1 x  
256^2  
    """
```

```
def videoToMatrix(image):  
    """  
        Function to extract images from dataset to array of  
matrix
```

```
Args:  
    image (file): the test image
```

```
Returns:  
    result (array): the image in shape of array 1 x  
256^2  
    """
```

```
def processTestImage(test_image):  
    """  
        Function to process test image  
  
Args:  
    test_image (file): the test image
```

```

    Returns:
        image (array): the image in shape of array 1 x
256^2
    """
def differenceTestImage(image, meanFace):
    """
        Function to calculate difference of test image with
mean face

    Args:
        image (array): the image in shape of array 1 x
256^2
        meanFace (array): mean face from dataset

    Returns:
        differenceTestImage (matrix): matrix of difference
between test image and mean face
    """

def solveCombinationLinear(difference,
array_of_eigenfaces):
    """
        Calculate the linear combination of each training face

    Args:
        difference (matrix): the difference of matrix face
and mean face
        array_of_eigenfaces (array): list of eigenfaces

    Returns:
        listOfCombination : list of linear combinations
    """

def solveCombinationLinearTestImage(array_of_eigenfaces,
differenceTestImage):
    """
        Calculate the linear combination of test image

    Args:
        array_of_eigenfaces (array): list of eigenfaces
        differenceTestImage (matrix): the difference of
test image and mean face
    """

```

```

    Returns:
        combination_test (array): the linear combination
    of test image
    """

```

3.4 Givens Matrix Calculation (givens.py)

```

def givens_template(m: int, n: int, k: int, c: float, s:
float):
    """Return givens matrix, given the value of axis m and
n
    and the value of k, c, s.

```

Args:

- m (int): axis rotation of interest
- n (int): axis rotation of interest
- k (int): size of the givens matrix
- c (float): constant
- s (float): constant

Returns:

- matrix: Givens matrix

"""

```

def givens_csr_hessenberg(input_matrix, m: int, n: int):
    """calculate the appropriate c and s constant to make
givens matrix

```

Args:

- input_matrix (matrix): input square matrix
- m (int): axis rotation of interest
- n (int): axis rotation of interest

Returns:

- tuple : tuple of constant c and s

"""

```

def givens_csr_qr(input_matrix, m: int, n: int):
    """Calculate the appropriate c and s constant to make
givens matrix

```

```

Args:
    input_matrix (matrix): input square matrix
    m (int): axis rotation of interest
    n (int): axis rotation of interest

Returns:
    tuple : tuple of constant c and s
"""

```

```

def givens_m_n_hessenberg (matrix, m: int, n: int):
    """Get givens matrix to make hessenberg form

```

```

Args:
    matrix (matrix): matriks
    m (int): axis rotation of interest
    n (int): axis rotation of interest

```

```

Returns:
    matriks: givens matrix to make hessenberg form
"""

```

```

def givens_m_n_qr (matrix, m: int, n: int):
    """Get givens matrix for qr decomposition

```

```

Args:
    matrix (matrix): matriks
    m (int): axis rotation of interest
    n (int): axis rotation of interest

```

```

Returns:
    matriks: givens matrix for qr decomposition
"""

```

3.5 Hessenberg Form (hessenberg.py)

```

def hessenberg_form(matrix):
    """Make hessenberg from of input matrix

```

```

Args:
    matrix (matrix): square matrix

Returns:
    matrix: hessenberg form of input matrix
"""

```

3.6 QR Iteration (qr.py)

```

def qr_decomposition(matrix):
    """Do QR decomposition to input square matrix

Args:
    matrix (matrix): square matrix

Returns:
    Tuple of matrix: Matrix Q and R, the result of
decomposition of A
"""

def qr_iteration_with_accum_q(matrix, n: int):
    """Do QR iteration n times

Args:
    matrix (matrix): square matrix
    n (int): number of iteration

Returns:
    tuple of matrix: Result of iteration n times, and
eigenvector matrix
"""

```

3.7 Eigenvalue (eigenvalue.py)

```

def get_k_eigenvalue_with_accum_q(array):
    """Pick specific amount of eigenvalue and eigenvector

Args:

```

```

        array (array): array of eigenvalue

    Returns:
        tuple: number of eigenvalue selected, array of
selected eigenvalue index
    """

```



```

def compute_eigenvalue_with_accum_q(matrix):
    """Get eigenvalue and eigenvector from matrix

    Args:
        matrix (matrix): square matrix

    Returns:
        tuple : array of eigenvalue, and eigenvector
    """

```

3.8 Eigenvector (eigenvector.py)

```

def get_k_eigenvector(arr_eigenvalue, arr_eigenvector):
    """Get k number of eigenvector

    Args:
        arr_eigenvalue (array): array of eigenvalue
        arr_eigenvector (matrix): array of eigenvector

    Returns:
        tuple : number of k selected, array of eigenvector
selected
    """

```

3.9 Euclidean Distance (euclideandst.py)

```

def euclideandst(vektor1, vektor2):
    """
        Function to calculate euclidean distance between two
vectors
    """

```

```

Args:
    vektor1 (array): vector 1 (same size)
    vektor2 (array): vector 2 (same size)

Returns:
    sum (float): the euclidean distance
"""

def shortestDst(vektor, matriksVektor):
    """
    Function to determine shortest euclidean distance

    Args:
        vektor (array): the vector weight of test image
        matriksVektor (matrix): the list of vector weight
        of training images

    Returns:
        idx (int): the index of the closest image
        cachemin (float) : the shortest euclidean distance
"""

```

3.10 Main Algorithm (main_algo.py)

```

def Load_Dataset(dataset):
    """
    The Main Algorithm to load dataset into training
    images

    Args:
        dataset (string): pathfolder

    Returns:
        meanFace (array) : the mean face
        array_of_eigenfaces (array) : array of eigen faces
        listOfCombination (array) : array of combination
        listOffFixMatrixFace (array) : array of matrix
        faces in RGB Mode
"""

def solveImage(test_image, meanFace, array_of_eigenfaces,
listOfCombination, listOffFixMatrixFace, dataset, type):
    """

```

Function to find the closest image

Args:

- test_image (string) : pathfile
- meanFace (array) : the mean face
- array_of_eigenfaces (array) : array of eigen faces
- listOfCombination (array) : array of combination
- listOfFixMatrixFace (array) : array of matrix

faces in RGB Mode

- dataset (string) : pathfolder
- type (integer) : type of test_image

Returns:

- result (matrix) : the picture of the closest face image
- filename (string) : the name of the closest face image
- dst (float) : the euclidean distance of the test image and the closest face image

"""

3.11 GUI (main.py)

```
def setClick():
    """
        Function to integrate button click into dataset
        loading, if test image is loaded automatically executes
        eigenface algorithm
    """

    Args:
        None

    Returns:
        None

    I.S.:
        Program not currently loading image
    F.S.:
        Folder choosing window will appear for user to
        select a dataset
        Dataset location then will be passed to loadSet
        function
```

```
        if test image is loaded, automatically executes
eigenface algorithm
    """
```

```
def fileClick():
```

```
    """
```

```
        Function to integrate button click into image test
loading, if dataset is loaded automatically executes
eigenface algorithm
```

Args:

None

Returns:

None

I.S.:

```
        Program not currently loading image, loading
dataset, or using camera as viewfinder
```

F.S.:

```
        File choosing window will appear for user to
select a test image
```

Test image is loaded to the program

```
        if dataset is loaded, automatically executes
eigenface algorithm
    """
```

```
def loadSet(folder_path):
```

```
    """
```

```
        Function to fetch necessary values from dataset
```

Args:

folder_path (string): path to dataset folder

Returns:

None

I.S.:

Program not currently loading image

F.S.:

```
        Dataset is loaded and ready for use in eigenface
algorithm
```

```

    While this function does not return a value, it
keeps data in global variables:
    -meanFace (matrix): Average face from a dataset
    -array_of_eigenfaces (matrix): Array of eigenfaces
obtained from dataset
    -listOfCombination (array): Array from linear
combination
    -listOfFixedMatrixFace (matrix): list of faces in
rgb
"""

```

```

def segmenter():
"""
    Function to segment the program executing order in
case of image load then dataset load order
    in which the dataset should be loaded first before
executing eigenface algorithm

    Args:
        None

    Returns:
        None

    I.S.:
        Program loaded image before loading dataset
    F.S.:
        Dataset is loaded first then eigenface algorithm
is executed
"""

```

```

def executeSplit(feed):
"""
    Function to execute eigenface algorithm

    Args:
        feed (matrix): Matrix value from camera feed

    Returns:
        None

    I.S.:

```

```

        Dataset is loaded, test image is chosen or camera
feed is active
    F.S.:
        Closest image from dataset is shown in result
image
        Euclidean distance shown in label
    """
"""

def timeCount(type):
    """
    Function to execute eigenface algorithm

    Args:
        type (int): int value to determine which counter
is being used, 1 for execution, 2 for load

    Returns:
        None

    I.S.:
        Dataset is being loaded or eigenface algorithm is
being executed
    F.S.:
        Elapsed time of operation shown in the
corresponding label in real time
    """

"""

def onRootIconify(event):
    """
    Function to minimize window to tray

    Args:
        event (event): action trigger, in this case it is
always <Unmap>, the unmapping of overlay

    Returns:
        None

    I.S.:
        GUI window is shown
    F.S.:
        GUI window is minimized to tray
    """
"""

```

```

def onRootDeiconify(event):
    """
        Function to minimize display window from tray

    Args:
        event (event): action trigger, in this case it is
        always <Map>, the mapping of overlay

    Returns:
        None

    I.S.:
        GUI window is minimized to tray
    F.S.:
        GUI window is shown
    """

```

```

def start_move(event):
    """
        Function to move window in case of a mouse drag in
        handle

    Args:
        event (event): action trigger, in this case it is
        always <ButtonPress-1>, left mouse button click

    Returns:
        None

    I.S.:
        GUI window is static
    F.S.:
        GUI window is ready to move
    """

```

```

def stop_move(event):
    """
        Function to stop window movement in case of a mouse
        drag release in handle

    Args:
        event (event): action trigger, in this case it is
        always <ButtonRelease-1>, left mouse button release
    """

```

```
Returns:  
    None  
  
I.S.:  
    GUI window is being moved according to mouse  
movement  
F.S.:  
    GUI window is static  
"""
```

```
def do_move(event):  
    """  
        Function to stop window movement in case of a mouse  
drag release in handle  
  
    Args:  
        event (event): action trigger, in this case it is  
always <ButtonRelease-1>, left mouse button release
```

```
    Returns:  
        None  
  
    I.S.:  
        GUI window is ready to move  
F.S.:  
    GUI window is being moved according to mouse  
movement  
"""
```

```
def minimize():  
    """  
        Function to integrate minimize button to iconify  
function  
  
    Args:  
        None  
  
    Returns:  
        None  
  
    I.S.:  
        GUI window is shown
```

```

F.S.:
    GUI window is minimized to tray
"""

def close():
    """
        Function to integrate close button into closing
        functions

    Args:
        None

    Returns:
        None

    I.S.:
        Program is running
    F.S.:
        Closing animation is played and program is stopped
"""

def fullwinswitch(widthval, heightval):
    """
        Function to switch window to a new window size

    Args:
        widthval (int): width value of new window size
        heightval (int): height value of new window size

    Returns:
        None

    I.S.:
        Program is running, opening animation currently
        not playing
    F.S.:
        Window size changes into widthvalxheightval
"""

def fullify():
    """
        Function to switch window to full screen mode

```

```

Args:
    None

Returns:
    None

I.S.:
    Program currently in windowed mode, opening
    animation currently not playing
F.S.:
    Window size changes into size of the screen
    Fullscreen button changes into window button
"""

```

```

def winify():
"""
Function to switch window to windowed mode

Args:
    None

Returns:
    None

I.S.:
    Program currently in fullscreen mode
F.S.:
    Window size changes into 1280x720
    Window button changes into fullscreen button
"""

```

```

def highlight(event, background, Hlimage):
"""
Function to switch button into highlight background
and image in case of mouse entering button area

Args:
    event (event): action trigger, in this case it is
    always <Enter>, the hovering of cursor in the area of the
    button
    background (_Color): new background in case of
    hover
    Hlimage (imageTK): new image in case of hover

```

```

    Returns:
        None

    I.S.:
        Button in idle
        Mouse moving inside button
    F.S.:
        Button image changes into Hlimage
        Button background changes into background
    """

```

```

def unhighlight(event, background, image):
    """
        Function to switch button into normal version in case
        of mouse leaving button area

    Args:
        event (event): action trigger, in this case it is
        always <Leave>, the leaving of cursor from the area of the
        button
        background (_Color): new background in case of
        leaving
        Hlimage (imageTK): new image in case of leaving

```

```

    Returns:
        None

    I.S.:
        Button in idle
        Mouse moving outside button
    F.S.:
        Button image changes into image
        Button background changes into background
    """

```

```

def highlightbtn(event, item, Hlimage, disable):
    """
        Function to switch an image button into highlight
        background and image in case of mouse entering button area

    Args:

```

```

        event (event): action trigger, in this case it is
always <Enter>, the hovering of cursor in the area of the
button
        item (variable): image button name identifier
        Hlimage (imageTK): new image in case of hover
        disable (boolean): passed condition to disable
button in certain conditions

    Returns:
        None

    I.S.:
        Button in idle
        Mouse moving inside button
    F.S.:
        Button image changes into Hlimage if disable
condition is not true
"""

```

```

def unhighlightbtn(event, item, image, disable):
"""
    Function to switch an image button into normal version
in case of mouse leaving button area

    Args:
        event (event): action trigger, in this case it is
always <Leave>, the leaving of cursor from the area of the
button
        item (variable): image button name identifier
        Hlimage (imageTK): new image in case of leaving
        disable (boolean): passed condition to disable
button in certain conditions

    Returns:
        None

    I.S.:
        Button in idle
        Mouse moving outside button
    F.S.:
        Button image changes into image if disable
condition is not true
"""

```

```
def buttonclick(event, item, disable, action):
    """
        Function to execute a command in case of a button
        click

        Args:
            event (event): action trigger, in this case it is
            always <Leave>, the leaving of cursor from the area of the
            button
            item (variable): image button name identifier
            action (function): function to be executed in a
            buttonclick
            disable (boolean): passed condition to disable
            button in certain conditions

        Returns:
            None

        I.S.:
            Button clicked
        F.S.:
            Button clicked animation played and action
            executed if disable condition is not true
    """
```

```
def showHandle(event):
    """
        Function to show title bar in full screen mode

        Args:
            event (event): action trigger, in this case it is
            always <Enter>, the entering of cursor from the area of
            title bar

        Returns:
            None

        I.S.:
            Title bar hidden
        F.S.:
            Title bar shown
    """
```

```

def hideHandle(event):
    """
        Function to hide title bar in full screen mode

    Args:
        event (event): action trigger, in this case it is
        always <Leave>, the leaving of cursor from the area of
        title bar

    Returns:
        None

    I.S.:
        Title bar shown
    F.S.:
        Title bar hidden
    """

```

```

def rollcred(event):
    """
        Function to show credits page if not shown or hide
        credits page if is shown

    Args:
        event (event): action trigger, in this case it is
        always <ButtonPress1>, left mouse button inside a button

    Returns:
        None

    I.S.:
        Hidden button clicked
    F.S.:
        Credits page shown if currently hidden
        Credits page hidden if currently shown
    """

```

```

def startVideo():
    """
        Function to start video feed

    Args:
        None
    """

```

```
Returns:  
    None  
  
I.S.:  
    Camera button clicked when camera feed is off  
    Not currently loading dataset  
F.S.:  
    Videofeed started  
"""
```

```
def stopVideo():  
    """  
        Function to stop video feed  
  
    Args:  
        None  
  
    Returns:  
        None  
  
    I.S.:  
        Camera button clicked when camera feed is on  
        Not currently loading dataset  
    F.S.:  
        Videofeed stopped  
"""
```

```
def highlightCam(event):  
    """  
        Function to switch the camera button into highlight  
        mode  
  
    Args:  
        event (event): action trigger, in this case it is  
        always <Enter>, the hovering of cursor in the area of the  
        button  
  
    Returns:  
        None  
  
    I.S.:  
        Button in idle
```

```

        Mouse moving inside button
F.S.:
    Button image changes into Hlimage
"""

def unhighlightCam(event):
"""
    Function to switch the camera button into normal mode

    Args:
        event (event): action trigger, in this case it is
always <Leave>, the leaving of cursor from the area of the
button

    Returns:
        None

    I.S.:
        Button in idle
        Mouse moving outside button
F.S.:
    Button image changes into normal image
"""

def cameraPress(event):
"""
    Function to integrate camera button into camera
functions

    Args:
        event (event): action trigger, in this case it is
always <ButtonPress1>, left mouse button inside a button

    Returns:
        None

    I.S.:
        Camera button clicked
F.S.:
    Button animation played
    startVideo called if currently not showing feed
    stopVideo called if currently showing feed
"""

```

```

def editalpha(x1, y1, x2, y2, **kwargs):
    """
        Function to add a white rectangle overlay with a
        certain alpha

    Args:
        x1 (int): rectangle start x
        y1 (int): rectangle start y
        x2 (int): rectangle end x
        y2 (int): rectangle end y
        **kwargs (parameter): additional parameters, used
        to determine alpha

    Returns:
        img (_CanvasItemId): canvas item parameter for the
        white rectangle overlay

    I.S.:
        Any
    F.S.:
        White rectangle overlay with a certain alpha added
        on top over main window
    """

```

```

def opening():
    """
        Function to play opening animation when opening the
        program

    Args:
        None

    Returns:
        None

    I.S.:
        Program started
    F.S.:
        Opening animation played
    """

def closing():
    """
    """

```

Function to play closing animation when closing the program

Args:
None

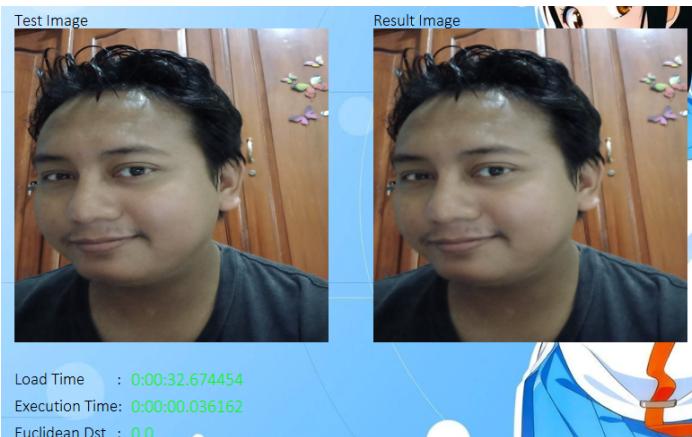
Returns:
None

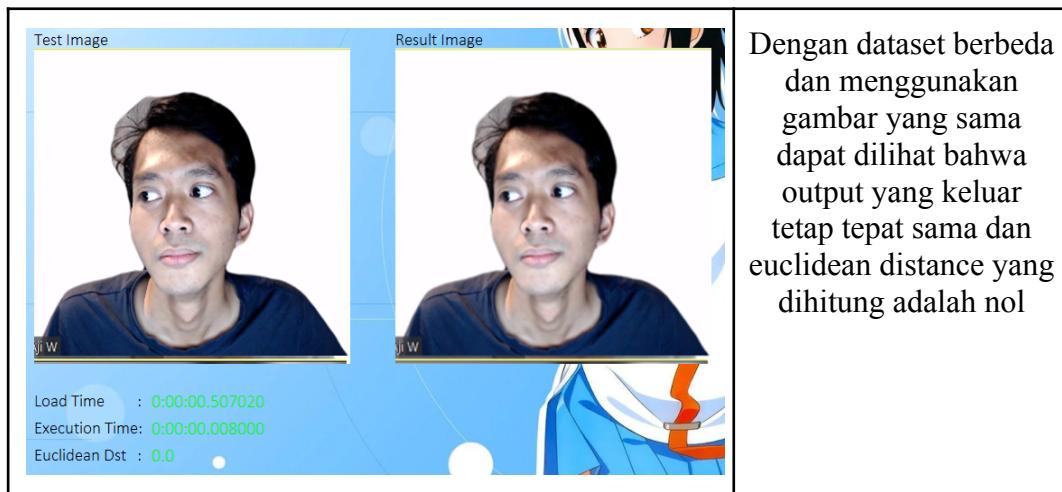
I.S.:
Program running
F.S.:
Closing animation played
"""

BAB IV

EXPERIMEN

4.1 Eksperimen Terhadap Gambar yang Sama

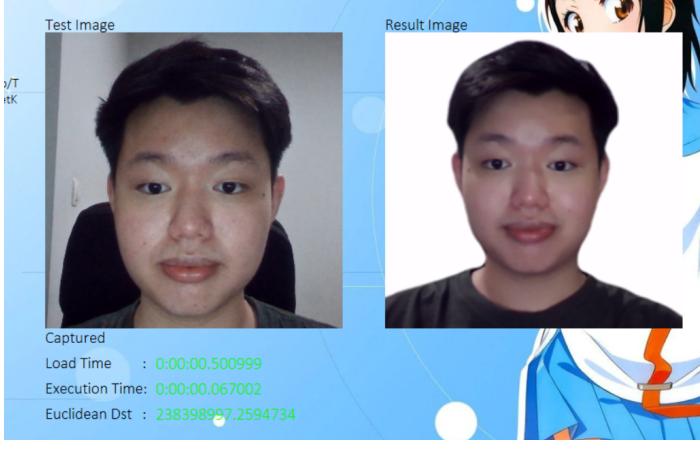
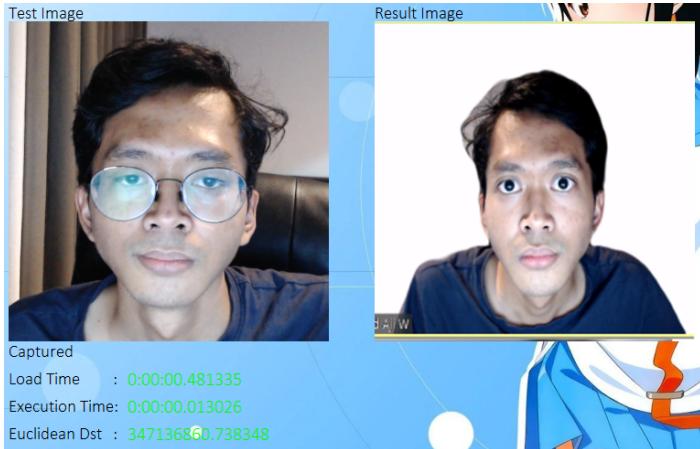
| Gambar | Deskripsi |
|--|--|
|  | Menggunakan gambar yang sama dapat dilihat bahwa output yang keluar tepat sama dan euclidean distance yang dihitung adalah nol |
|  | Menggunakan gambar yang sama dapat dilihat bahwa output yang keluar tepat sama dan euclidean distance yang dihitung adalah nol |



4.2 Eksperimen Terhadap Gambar Orang yang Sama, Namun di Luar

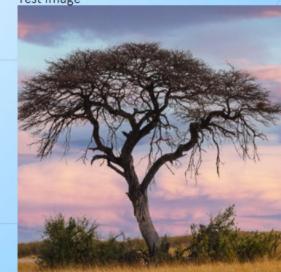
Dataset

| Gambar | Deskripsi |
|---|---|
|  Test Image  Result Image | <p>Dengan orang yang sama namun di luar dataset program dapat mengeluarkan gambar dengan orang yang sama pula. Euclidean distance yang diperoleh berada dalam rentang 100.000.000 yang merupakan jarak yang relatif cukup rendah.</p> |
| Load Time : 0:00:23.805934 Execution Time: 0:00:00.041959 Euclidean Dst : 117172774.93043068 | |

| | |
|---|--|
|  <p>Test Image</p> <p>Captured</p> <p>Load Time : 0:00:00.500999</p> <p>Execution Time: 0:00:00.067002</p> <p>Euclidean Dst : 238398997.2594734</p> <p>Result Image</p> | <p>Dengan orang yang sama namun di luar dataset program dapat mengeluarkan gambar dengan orang yang sama pula yaitu anggota dari kelompok ini bernama Michael. Euclidean distance yang diperoleh berada dalam rentang 200.000.000 yang merupakan jarak yang relatif cukup rendah.</p> |
|  <p>Test Image</p> <p>Captured</p> <p>Load Time : 0:00:00.481335</p> <p>Execution Time: 0:00:00.013026</p> <p>Euclidean Dst : 347136860.738348</p> <p>Result Image</p> | <p>Dengan orang yang sama namun di luar dataset program dapat mengeluarkan gambar dengan orang yang sama pula. Euclidean distance yang diperoleh berada dalam rentang 300.000.000 yang merupakan jarak yang cukup tinggi, namun hal ini dapat disebabkan oleh background yang berbeda.</p> |

4.3 Eksperimen Terhadap Gambar Acak

| Gambar | Deskripsi |
|--------|-----------|
|--------|-----------|

| | |
|---|---|
|  <p>Test Image</p>  <p>Result Image</p> <p>Load Time : 0:00:00.481335 Execution Time: 0:00:00.009986 Euclidean Dst : 325307279.81914216</p> | <p>Dengan dataset berupa monyet program mengeluarkan gambar orang dengan ekspresi relatif mendekati. Euclidean distance yang diperoleh berada dalam rentang 300.000.000 yang merupakan jarak yang cukup tinggi.</p> |
|  <p>Test Image</p>  <p>Result Image</p> <p>Load Time : 0:00:00.556999 Execution Time: 0:00:00.019002 Euclidean Dst : 298256198.8510508</p> | <p>Dengan dataset berupa bola program mengeluarkan gambar orang yang beda jauh. Euclidean distance yang diperoleh hampir mendekati 300.000.000 yang merupakan jarak yang cukup tinggi.</p> |
|  <p>Test Image</p>  <p>Result Image</p> <p>Load Time : 0:00:00.554001 Execution Time: 0:00:00.012003 Euclidean Dst : 290945735.5319033</p> | <p>Dengan dataset berupa pohon program mengeluarkan gambar yang beda jauh. Euclidean distance yang diperoleh hampir mendekati 300.000.000 yang merupakan jarak yang cukup tinggi.</p> |

BAB 5

KESIMPULAN, SARAN, DAN REFLEKSI

5.1 Kesimpulan

Pada Tugas Besar 2 IF2123 Aljabar Linier dan Geometri ini telah diimplementasikan sebuah aplikasi python pengenalan wajah menggunakan metode eigenface yang menggunakan metode-metode yang sudah dipelajari selama akhir semester 3 di mata kuliah Aljabar Linier dan Geometri dan tambahan materi yang kami eksplor secara mandiri, yaitu mengekstrak gambar menjadi matrix, menghitung mean face, menghitung difference, menghitung eigen value dan eigen vector, mencari eigen face, mencari kombinasi linier dari training image, menghitung euclidean distance untuk mencari muka terdekat, dan membuat tampilan GUI untuk mempercantik dan mempermudah interaksi dengan pengguna.

Program ini dibuat dalam bahasa pemrograman Python dan dibuat dalam 3 folder, yaitu src (berisi source code), doc (berisi laporan), dan test (berisi test case yaitu dataset).

Adapun metode-metode yang diterapkan pada aplikasi pengenalan wajah, seperti metode QR Decomposition, ekstraksi image dengan OpenCV, numpy untuk pengolahan matrix, dan tkinter untuk GUI.

5.2 Saran

Dari proses pengerjaan tugas besar ini, kami memiliki banyak kendala selama mengerjakan seperti dataset yang tidak sesuai dengan ketentuan dari algoritma

eigenface. Oleh karena itu, kami memiliki beberapa saran agar tugas ini bisa menjadi lebih baik, yaitu

- a. Optimisasi Metode QR Decomposition menjadi lebih cepat.
- b. Mengumpulkan dataset yang lebih rapih dan sesuai ketentuan eigenface.

5.3 Refleksi

Tugas Besar 2 IF2123 Aljabar Linier dan Geometri Semester I Tahun 2022/2023 merupakan salah satu tugas besar menarik yang penulis dapatkan pada semester ini. Proses penggerjaan tugas ini tentunya melalui berbagai rintangan. Dengan tugas ini, penulis mendapatkan berbagai pengetahuan mengenai penggunaan nilai eigen dan vektor eigen pada salah satu aplikasinya, yaitu pengenalan wajah dengan eigenface. Penulis diberi kesempatan untuk mengimplementasikan pengenalan wajah dengan aplikasi tersebut, keberhasilan akan mengimplementasikan hal tersebut memberi rasa pencapaian dengan keberhasilan membuat hal baru. Dalam tugas besar ini penulis juga mendapatkan kesempatan untuk melakukan eksplorasi lebih dalam terhadap sisi *user interface* dalam bahasa python, khususnya dengan library tkinter.

Tantangan utama yang penulis hadapi adalah optimisasi dan eksplorasi terhadap algoritma yang terbaik untuk digunakan dalam pencarian nilai dan vektor eigen. Rintangan ini dihadapi dengan mempelajari lebih lanjut mengenai nilai dan vektor eigen serta algoritma yang dapat digunakan dan sinergi dari setiap anggota kelompok yang saling membantu dan mengerjakan tugasnya dengan baik.

DAFTAR PUSTAKA

Munir, Rinaldi. (2022). *IF2123 Aljabar Geometri - Semester I Tahun 2022/2023.*

Institut Teknologi Bandung. Diakses pada 3 Oktober 2022, dari

<https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2022-2023/geo22-23.htm>

Anthonissen, Martjin. (2022). *QR algorithm for computing eigenvalues*

[PowerPoint slides]. Technische Universiteit Eindhoven.

<https://www.dropbox.com/s/9byoug9odr5eaov/4-6%20qr%20algorithm.pdf?dl=0>

Navarrete, Pablo; Ruiz-Del-Solar, Javier (November 2002). *Analysis and*

Comparison of Eigenspace-Based Face Recognition Approaches.

International Journal of Pattern Recognition and Artificial Intelligence. 16

(7): 817–830.

Anderson, Edward. (2000). *Discontinuous Plane Rotations and the Symmetric*

Eigenvalue Problem. LAPACK Working Note. University of Tennessee at

Knoxville and Oak Ridge National Laboratory.

LAMPIRAN

[Tautan repositori Github](#)

[Dataset yang digunakan](#)

[Tautan video penjelasan](#)

#TeamOnodera

Fakhri Muhammad Mahendra 13521045

Muhamad Aji Wibisono 13521095

Michael Jonathan Halim 13521124

