LAPORAN TUGAS BESAR
IF2123 ALJABAR LINIER DAN GEOMETRI
SEMESTER I 2022-2023

Disusun oleh:

Kelompok 08 (BPJS)

Tobias Natalio Sianipar	13521090
Bintang Dwi Marthen	13521144
Zidane Firzatullah	13521163

PROGRAM STUDI

TEKNIK INFORMATIKA

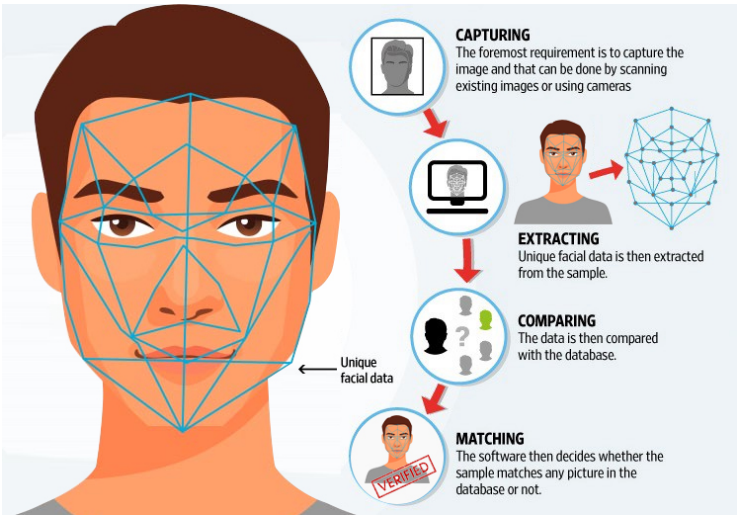
SEKOLAH TEKNIK ELEKTRO

DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG 2022

1. DESKRIPSI PERSOALAN

Pengenalan wajah merupakan sebuah teknologi biometrik yang biasa digunakan untuk mengindentifikasi wajah seseorang. Pengenalan wajah sendiri seringkali digunakan untuk berbagai kepentingan, terutama keamanan seperti akses kunci layar ponsel. Teknologi pengenalan wajah merupakan salah satu implementasi dari pembelajaran mesin, diberikan suatu basis data kumpulan citra wajah sebagai model pembelajaran bagi wajah-wajah yang akan diidentifikasi kelak (hanya akan mengenali wajah yang telah ada di basis data). Alur dari teknologi pengenalan wajah dapat dilihat pada Gambar 1.



Gambar 1 Alur Proses Teknologi Pengenalan Wajah
(Sumber: https://www.shadowsystem.com/page/20)

Terdapat berbagai metode untuk mengimplementasikan teknologi pengenalan wajah: *euclidean distance and cosine similarity, principal component analysis (PCA),* dan *eigenface*. Pada tugas besar ini, akan digunakan metode *eigenface* yang merupakan salah satu implementasi matriks dalam pengenalan wajah.

Seperti pada pembelajaran mesin, metode pengenalan wajah dengan *eigenface* akan terbagi menjadi dua tahap: pembelajaran dan pencocokkan. Pada tahapan pembelajaran atau pelatihan, akan diberikan sekumpulan citra wajah yang kemudian akan dinormalisasi dari berwarna (RGB) menjadi *grayscale* (matriks). Hasil dari normalisasi tersebut akan digunakan dalam perhitungan *eigenface*. Sesuai dengan namanya, metode *eigenface* akan menggunakan *eigenvector* dalam pembentukannya.

2. TEORI SINGKAT

2.1. Perkalian Matriks

Suatu matriks dapat dikalikan dengan suatu nilai skalar dan matriks lainnya. Hasil perkalian suatu matriks A dengan nilai skalar

merupakan suatu matriks baru dengan setiap elemen-elemennya dikalikan dengan skalar *k*. Secara umum, perkalian matriks *A* dengan nilai skalar *k* adalah sebagai berikut:

$$k\begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{bmatrix} = \begin{bmatrix} k \times a_{1,1} & \cdots & k \times a_{1,n} \\ \vdots & \ddots & \vdots \\ k \times a_{m,1} & \cdots & k \times a_{m,n} \end{bmatrix} \text{--- (1)}$$

Sementara itu, hasil perkalian matriks dengan matriks tidak dapat dilaksanakan secara langsung. Perkalian matriks *A* dengan matriks *B* hanya dapat dilaksanakan apabila jumlah kolom matriks *A* sama dengan jumlah baris matriks *B* dan hasil dari perkaliannya akan menghasilkan matriks dengan jumlah barisnya sebanyak baris matriks *A* dan jumlah kolomnya sebanyak kolom matriks *B*. Elemen-elemen dari matriks hasil perkalian dapat ditentukan dengan peraturan baris-kolom:

$$(AB)_{i,j} = a_{i,1}b_{1,j} + a_{i,2}b_{2,j} + a_{i,3}b_{3,j} + \cdots + a_{i,r}b_{r,j} \text{ --- (2)}$$

2.2. Nilai Eigen dan Vektor Eigen

Bila terdapat suatu matriks persegi *A* dan vektor tidak nol *x* (vektor eigen dari *A*), jika *Ax* merupakan hasil perkalian skalar λ dengan *x* maka λ disebut sebagai nilai eigen dari A.

$$Ax = \lambda x \text{ --- (3)}$$

Secara umum, operasi perkalian matriks *A* dengan vektor *x* merupakan operasi penyusutan atau pemanjangan vektor *x* dengan faktor *λ*. Berikut merupakan contoh untuk suatu matriks persegi *A* dengan dimensi 2x2.

$$Vektor\ x = \begin{bmatrix} 1 \\ 2 \end{bmatrix} merupakan\ vektor\ eigen\ dari\ A = \begin{bmatrix} 3 & 0 \\ 8 & -1 \end{bmatrix}$$

$$Ax = \begin{bmatrix} 3 & 0 \\ 8 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \end{bmatrix} = 3x$$

$$\therefore \lambda = 3$$

Terdapat suatu metode untuk menentukan nilai eigen dan vektor eigen dari suatu matriks persegi *A*. Karena *Ax = λx* maka *λx – Ax = 0* sehingga dapat ditulis ulang menjadi *(λ – A)x = 0*. Akan tetapi, nilai dari *λ – A* merupakan hasil pengurangan skalar dengan matriks sehingga perlu dilakukan normalisasi dengan cara mengalikan λ dengan matriks identitas berukuran sama dengan matriks *A*.

$$(\lambda I - A)x = 0 \text{ --- (4)}$$

Persamaan 4 dapat dipandang sebagai suatu persoalan sistem persamaan linier yang direpresentasikan dalam bentuk matriks. Untuk menyebabkan nilai *λ* menjadi nilai tidak nol maka determinan dari *(λI – A)* haruslah nol.

$$det(\lambda I - A) = 0 \text{ --- (5)}$$

Berikut merupakan contoh mencari nilai eigen dan vektor eigen dari suatu matriks persegi *A* berukuran 2x2

$$A = \begin{bmatrix} 3 & 0 \\ 8 & -1 \end{bmatrix}$$

$$\lambda I = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$$

$$\lambda I - A = \begin{bmatrix} \lambda - 3 & 0 \\ 8 & \lambda + 1 \end{bmatrix}$$

$$\det(\lambda I - A) = (\lambda - 3) \times (\lambda + 1)$$

$$\therefore \lambda = 3 \; atau \; \lambda = -1$$

$Mencari\ vektor\ eigen\ dari\ A$

$$(\lambda I - A)x = \begin{bmatrix} \lambda - 3 & 0 \\ -8 & \lambda + 1 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$Untuk\ \lambda = 3$

$$\begin{bmatrix} 0 & 0 \\ -8 & 4 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$-8x_1 + 4x_2 = 0 \rightarrow 8x_1 = 4x_2 \rightarrow x_1 = ½x_2$$

$$x_1 = ½t, x_2 = t, t \in R$$

$$\therefore x = t \begin{bmatrix} ½ \\ 1 \end{bmatrix}, t \in R$$

$Untuk\ \lambda = -1$

$$\begin{bmatrix} -4 & 0 \\ -8 & 0 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$-4x_1 = 0 \; dan -8x_1 = 0$$

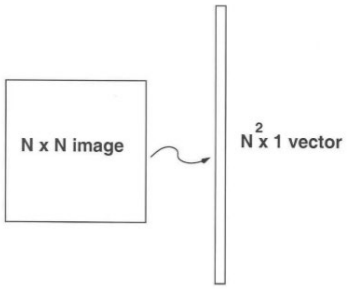
$$x_1 = 0, x_2 = t, t \in R$$

$$\therefore x = t \begin{bmatrix} 0 \\ 1 \end{bmatrix}, t \in R$$

2.3. Eigenface

2.3.1. Algoritma Pelatihan

1. Kumpulkan citra wajah dengan dimensi N x N sebagai kumpulan citra wajah untuk pelatihan model



Gambar 2 Ilustrasi Konversi Gambar menjadi Vektor
(Sumber: https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/)

2. Konversikan setiap citra menjadi vektor berukuran N^2 x 1 (disimbolkan dengan *x*)

3. Kalkulasikan nilai rata-rata dari vektor muka (disimbolkan dengan ψ) dan kemudian kurangi nilai tersebut dengan nilai citra pelatihan untuk normalisasi (disimbokan dengan a_i)
$$\psi = \frac{1}{m}\sum_{i=1}^{m} x_i \text{ --- (6)}$$
$$a_i = x_i - \psi \text{ --- (7)}$$
4. Satukan nilai setiap nilai a_i dalam suatu matriks berukuran N^2 x M dengan M adalah jumlah citra wajah pelatihan
$$\mathrm{A} = [a_1\ a_2\ a_3 \ldots\ a_m]$$
5. Kalikan matriks A^T dengan matriks A untuk menentukan matriks kovarian dari A (C)
$$\mathrm{C} = A^T A \text{ --- (8)}$$
6. Kalkulasikan nilai eigen (λ) dan vektor eigen (v) dari matriks kovarian dengan tahapan berikut
$$A^T A v_i = \lambda_i v_i$$
$$A A^T A v_i = \lambda_i A v_i$$
$$C' u_i = \lambda_i u_i \text{ --- (9)}$$
Dari persamaan tersebut, nilai eigen dari C' akan sama dengan nilai eigen dari C dan vektor eigennya akan berelasi dengan persamaan
$$u_i = A v_i \text{ --- (10)}$$
7. Tentukan nilai eigen dan vektor eigen terbesar dari matriks kovarian (disimbolkan dengan M)
8. Petakan nilai eigen dan vektor eigen dari matriks kovarian tereduksi ke C' dengan menggunakan persamaan 10
9. Tentukan jumlah vektor eigen dari C' yang berkorespondensi dengan K nilai eigen terbesar (K < M), vektor eigen ini akan memiliki ukuran N^2
10. Gunakan vektor eigen yang didapatkan pada tahap 9 dan representasikan setiap vektor wajah dalam kombinasi linier dari K vektor eigen terbaik
$$x_i - \psi = \sum_{j=1}^{K} w_j u_j \text{ --- (11)}$$
u_j merupakan nilai yang disebut sebagai *eigenface*
11. Gunakan koefisien dari *eigenface* dan representasikan wajah hasil pelatihan dalam bentuk vektor kolom dari koefisien - koefisien tersebut

$$x_i = \begin{bmatrix} w_1^i \\ w_2^i \\ w_3^i \\ . \\ . \\ w_k^i \end{bmatrix} \text{--- (12)}$$

2.3.2. Algoritma Pencocokan

1. Diberikan suatu wajah *y* untuk diidentifikasi, wajah *y* ini haruslah terdapat pada tengah citra dan memiliki dimensi yang sama dengan citra wajah yang digunakan pada tahapan pelatihan model *eigenface*
2. Normalisasikan nilai dari citra wajah tersebut dengan nilai rata-rata yang didapatkan pada tahapan pelatihan

   $$\phi = y - \psi \text{ --- (13)}$$
3. Konversikan vektor yang telah dinormalisasi tersebut menjadi vektor *eigenspace* untuk mendapatkan kombinasi linier *eigenface*

   $$\phi = \sum_{i=1}^{k} w_i u_i \text{ --- (14)}$$
4. Generasikan suatu vektor koefisien dari koefisien yang didapatkan pada tahap 3

   $$\Omega = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ . \\ . \\ w_k \end{bmatrix} \text{--- (15)}$$
5. Tentukan nilai minimum perbedaan antara nilai Ω yang didapatkan dengan nilai-nilai yang disimpan oleh citra wajah pada model pelatihan

   $$e_r = min_l \, \|\Omega - \Omega_l\| \text{ --- (16)}$$
6. Apabila nilai e_r yang didapatkan kurang dari nilai batas toleransi, maka wajah *y* dikenali sebagai suatu wajah dari kumpulan wajah pelatihan model, bila nilainya lebih dari nilai batas toleransi maka tidak ada wajah yang cocok dengan wajah *y* pada kumpulan wajah pelatihan model

3. IMPLEMENTASI PROGRAM

Implementasi pengenalan wajah menggunakan *eigenface* ini dilakukan dalam bahasa *python.* Kelompok ini menggunakan library *OpenCV* dan *numpy* untuk mengubah data pixel pada kumpulan gambar yang dipilih sebagai dataset *training image* menjadi matriks X yang merepresentasikan satu gambar tiap kolomnya menggunakan fungsi **get_pict_array** pada *module*

otf, kemudian *average face* yang dicari dengan membagi hasil penjumlahan tiap elemen pada vektor kolom matriks X dengan panjang vektor kolom atau jumlah pixel tiap gambar digunakan untuk mendapatkan matrix A yang merupakan hasil pengurangan matriks X dengan matrix *average face* menggunakan fungsi **get_mean_vspace** dan **get_mean_diff_array** pada *module* **otf**. Selanjutnya, untuk mengurangi jumlah komputasi yang dilakukan oleh program, matrix *Covariance* dicari dengan melakukan perkalian matrix $A^T \times A$ yang menghasilkan matrix dengan ukuran $M \times M$, (M adalah jumlah gambar yang digunakan sebagai *training image*) menggunakan fungsi **get_compressed_cov_array** pada *module* **otf**. Kemudian, program mencari *eigen vector* dan *eigen value* menggunakan metode *QR Iteration*, pada algoritma ini, *eigen value* didapatkan dari nilai elemen diagonal dari hasil iterasi perkalian matriks $R \times Q$ yang didapatkan dari hasil *QR Decomposition* yang dilakukan pada matrix *Covariance* menggunakan fungsi **qr** pada *module* **eigen**, Sedangkan *eigen vector* didapatkan dari hasil iterasi perkalian matrix Q. Hal ini dilakukan berulang kali hingga hasil perkalian matrix $R \times Q$ membentuk matrix *Upper Triangle* menggunakan fungsi **eig** pada *module* **eigen**. Kemudian *eigen vector* yang didapatkan diurutkan berdasarkan besar nilai *eigen value*-nya untuk mendapatkan *eigen vector* dengan pengaruh yang signifikan terhadap gambar. matriks berisi *eigen vector* dengan ukuran $M \times M$ tersebut kemudian dikalikan dengan matrix A untuk mendapatkan matriks *eiegenfaces* dengan tiap vektor kolom merepresentasikan suatu gambar *eigenface* dengan panjang vektor sama dengan banyak pixel pada input gambar menggunakan fungsi **eigenfaces** pada *module* **eigenface**. Kemudian, array *weight* dicari dengan perkalian tiap vektor kolom *eigenface* dengan tiap vektor kolom matrix A. Setiap array *weight* yang merepresentasikan suatu gambar pada dataset kemudian di simpan sebagai *tuple* (<nama orang pada gambar>, array *weight*, <*path* ke gambar>) melalui fungsi **build_recog_face** pada *module* **ymldb**. terakhir, untuk melakukan pengenalan wajah, program mencari array *weight* gambar yang akan di-*test* dan mencari *euclidean distance* array *weight* gambar tes dengan tiap array *weight* gambar pada dataset, gambar dengan *euclidean distance* terkecil diasumsikan sebagai gambar dari orang yang sama pada gambar tes, hal ini dilakukan menggunakan fungsi **find_match** dan **find_min_euclid** pada module **recog**.

4. EKSPERIMEN

Data Set Training	Data Set Validasi	Lama Pelatihan	Akurasi Pencocokan
new_train (26 citra)	new_val	00:30.468	32,258%
new_val (31 citra)	new_train	00:15.419	38,462%
new_train2 (22 citra)	new_val2	01:17.392	94,286%

new_val2 (35 citra)	new_train2	02:17.168	86,364%
new_train3 (38 citra)	new_val3	13:14.578	80,952%
new_val3 (21 citra)	new_train3	07:42.907	84,211%

Tabel 1 Data Hasil Eksperimen

Terdapat tiga jenis dataset yang digunakan pada eksperimen: dataset yang relatif tidak cocok dengan *eigenface*, dataset yang relatif optimal untuk *eigenface* dan telah dipotong, dan dataset yang relatif optimal untuk *eigenface* tetapi belum dipotong. Pertama-tama terdapat perbedaan yang drastis antara akurasi untuk dataset yang optimal dan kurang optimal untuk *eigenface*. Selain itu, terdapat juga perbedaan drastis untuk dataset yang telah dipotong dan belum dipotong dalam segi lama pelatihan.

Untuk membahas mengenai perbedaan akurasi yang drastis, pertama-tama perlu diketahui terlebih dahulu batasan-batasan dari *eigenface*: wajah yang ingin dikenali harus terdapat di tengah citra, wajah yang ingin dikenali harus menghadap ke arah depan dengan baik, dan sensitif terhadap cahaya, bayangan, serta ukuran wajah pada citra. Dengan batasan-batasan tersebut dan melihat citra-citra wajah yang digunakan pada dataset pertama maka dapat disimpulkan bahwa rendahnya akurasi karena batasan-batasan tersebut tidak dipenuhi oleh dataset tersebut.

Gambar *3* Salah Satu Citra Wajah pada Dataset Pertama (Tidak Menghadap Depan dengan Baik)

Mengenai perbedaan waktu pelatihan yang berbeda drastis, dapat dijelaskan dengan resolusi citra yang diolah. Dataset kedua memiliki resolusi-resolusi yang lebih rendah dibandingkan dengan dataset ketiga karena telah dipotong menjadi beraspek rasio 1:1. Karena resolusinya yang lebih rendah, maka program hanya perlu memroses lebih sedikit informasi dari citra pada dataset tersebut.

Meskipun dataset kedua dan ketiga mayoritas datanya telah optimal, terdapat beberapa citra wajah yang dapat menyebabkan salahnya hasil pengenalan karena pada dataset tersebut juga disisipkan beberapa citra wajah yang tidak optimal untuk menguji kakas *haarcascades* untuk mengatur posisi wajah menjadi tengah dan diputar sehingga lurus.

Gambar *4* Salah Satu Citra Wajah pada Dataset Kedua (Tidak Menghadap Depan dengan Baik)

Selain hal-hal tersebut, selama eksperimen didapati satu penemuan yang menarik. Penemuan tersebut adalah bagaimana mesin yang berbeda dapat menghasilkan hasil basis data yang berbeda meskipun menggunakan data dan algoritma yang sama. Salah satu penyebabnya adalah perbedaan bagaimana setiap mesin memroses nilai-nilai bilangan *float*. Sebagai contoh: terdapat perbedaan hasil penghitungan *weight* pada mesin dengan *processor* Intel dan mesin dengan *processor* AMD karena *library* IntelMKL (*Intel Math Kernel Library*) yang tentu saja tidak terdapat pada mesin dengan *processor* AMD.

Gambar *5* Hasil Penciptaan Basis Data pada Mesin dengan *Processor* Intel

Gambar *5* Hasil Penciptaan Basis Data pada Mesin dengan *Processor* AMD

5. KESIMPULAN, SARAN, DAN REFLEKSI

5.1. Kesimpulan

5.1.1. Teknologi *eigenface* merupakan salah satu algoritma pengenalan wajah yang menerapkan konsep *eigen* vektor dan nilai *eigen*

5.1.2. Meskipun *eigenface* mudah untuk diimplementasi, terdapat berbagai batasan pada citra wajah yang digunakan

5.2. Saran

5.2.1. Diberikan basis data yang sesuai dengan batasan teknologi yang digunakan (*eigenface* memiliki banyak keterbatasan tetapi basis data yang diberikan banyak yang melanggar batasan-batasan tersebut)

5.2.2. Pemberian tugas besar mempertimbangkan spesifikasi perangkat mahasiswa yang terbatas

5.3. Refleksi

5.3.1. Sering-sering bertanya kepada teman yang lebih mengerti supaya tidak tersesat dalam proses pengerjaan

5.3.2. Kesalahan terkadang bukan dari segi *source code* tetapi bisa juga dari basis data yang dimiliki mengingat kualitas data sangat penting dalam pembelajaran mesin

5.3.3. Perbedaan hasil yang didapati dapat terjadi karena perbedaan mesin yang digunakan

5.3.4. Apabila didapati basis data yang digunakan sekarang tidak memenuhi batasan teknologi yang digunakan, segera mencari basis data baru atau membuatnya sendiri

6. REFERENSI

Anton, H., & Rorres, C. (2014). Elementary Linear Algebra Applications Version. In H. Anton, & C. Rorres, *Elementary Linear Algebra Applications Version* (pp. 28-30;291-298). Canada: Wiley.

Howse, J., & Minichino, J. (2020). *Learning OpenCV 4 Computer Vision with Python 3.*

pawangfg. (2021, September 24). *ML | Face Recognition Using Eigenfaces (PCA Algorithm)*. Diakses dari Geeks for Geeks: https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/ pada 4 November 2022

Robinson, M. B., Matthew, E., Jon, K., & Doug, K. (2004, Desember 21). *Face Recognition using Eigenfaces*. Diakses dari openstax CNX: https://cnx.org/contents/oic3Nj4q@2.1:0j_nJCAQ@2/Thresholds-for-Eigenface-Recognition pada 14 November 2022

UoM, I. (2021, April 12). *Face Recognition Algorithm using Eigenfaces*. Diakses dari YouTube: https://youtu.be/61NuFlK5VdU pada 10 November 2022

7. LAMPIRAN

7.1. Link Repository GitHub
https://github.com/Marthenn/Algeo02-21090

7.2. Link Video Bonus
https://youtu.be/qDNEF38dl48