Laporan Tugas Besar 2 IF2123

Aljabar Linier dan Geometri

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

Disusun Oleh:

Kelompok 32 - FACEIT

Nigel Sahl	13521043
Muhammad Naufal Nalendra	13521152
Hanif Muhammad Zhafran	13521157

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2022/2023

BAB I

Deskripsi Masalah

Pada tugas besar kedua dari mata kuliah Aljabar Linier dan Geometri kami diberikan tugas untuk membuat sebuah app-based program yang berfungsi untuk face recognition.

Input yang dapat diberikan pada website adalah :

1. File foto yang akan diuji
2. Folder dataset wajah
3. Foto wajah menggunakan real-time camera yang dapat mendeteksi wajah pengguna.

Spesifikasi program adalah 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

Bonus:

- Bagian A (Kamera)
  1. Terdapat fitur kamera yang dapat mengenali wajah secara realtime menggunakan webcam ketika program dijalankan.
  2. Teknis pengenalan wajah melalui kamera dibebaskan oleh pembuat program. (contoh: program dieksekusi setiap 10 detik sekali).
  3. Fitur kamera merupakan fitur tambahan, fitur utama upload gambar melalui GUI tetap harus ada.
- Bagian B (Video)
  1. Video penjelasan algoritma dan aplikasi program yang diunggah ke youtube.
  2. Video dibuat sekreatif mungkin dengan target untuk mensosialisasikan ilmu yang kalian sudah pelajari dan terapkan pada program ini. Bukan hanya video mengenai penggunaan aplikasi.

BAB II

Teori Singkat

1. Perkalian Matriks

Misalkan matriks A dengan ukuran $m \times n$ dan matriks B dengan ukuran $n \times p$.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix}$$

Maka, AB akan menghasilkan matriks C dengan ukuran $m \times p$.

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{pmatrix}$$

Tiap elemen dari C didefinisikan sebagai berikut:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Syarat dua matriks dapat dioperasikan perkalian yaitu banyak kolom matriks pertama harus sama dengan banyak baris matriks kedua.

$$A_{m \times n} B_{n \times t} = C_{m \times t}$$

2. Nilai Eigen dan Vektor Eigen

Jika A adalah matriks $n \times n$ maka vektor tidak-nol $\boldsymbol{x}$ di R^n disebut vektor eigen dari A jika $A\boldsymbol{x}$ sama dengan perkalian suatu skalar λ dengan $\boldsymbol{x}$, yaitu:

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

Skalar λ disebut nilai eigen dari A, dan $\boldsymbol{x}$ dinamakan vektor eigen yang berkorespondensi dengan λ.

Vektor eigen dan nilai eigen dari matriks A dapat dihitung dengan cara sebagai berikut:

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

$$IAx = \lambda Ix$$

$$Ax = \lambda Ix$$

$$(\lambda I - A)x = 0$$

Supaya $(\lambda I - A)x = 0$ memiliki solusi tidak-nol, maka

$$\det(\lambda I - A) = 0 \ldots (1)$$

Dari (1), nilai eigen λ dapat ditentukan. Kemudian seluruh vektor eigen dapat dicari dengan mencari basis ruang eigen dari semua nilai eigen λ.

Cara diatas merupakan cara yang paling dasar untuk mencari nilai eigen dan vektor eigen. Selain cara tersebut, nilai eigen dan vektor eigen juga dapat dicari dengan cara dekomposisi QR. Dekomposisi QR sering digunakan untuk menghitung nilai eigen dan vektor eigen dengan pendekatan komputasional.

3. Dekomposisi QR

Dekomposisi QR merupakan dekomposisi dari suatu matriks A menjadi matriks Q dan R yang hubungannya dapat dinyatakan dengan $A = QR$. Q merupakan matriks orthogonal dan R merupakan matriks segitiga atas. Terdapat beberapa metode untuk melakukan dekomposisi QR, diantaranya adalah menggunakan transformasi Householder.

Matrix transformasi Householder H_u adalah transformasi yang mengambil vektor u kemudian merefleksikannya ke bidang di R^n. H_u dapat dinyatakan dengan persamaan berikut:

$$H_u = I - \frac{2uu^T}{u^T u}, \qquad u \neq 0$$

Dalam dekomposisi QR, vektor u merupakan vektor normal yang didefinisikan sebagai berikut:

$$u = a + sign(a_1)\|a\|_2 e_1$$

Vektor a merupakan vektor kolom dari matriks A, kemudian vektor e_1 merupakan vektor basis standar pertama ($e_1 = [1\ 0\ 0\ \ldots]^T$). $sign(a_1)$ merupakan tanda (+/-) dari elemen pertama pada vektor a.

Matriks transformasi H kemudian dikalikan dengan matriks A untuk membentuk matriks sebagai berikut:

$$H_u A = \begin{bmatrix} X & X & X & X & X & X & X & \cdots & X \\ 0 & X & X & X & X & X & X & \cdots & X \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ldots & \vdots \\ 0 & X & X & X & X & X & X & \cdots & X \\ 0 & X & X & X & X & X & X & \cdots & X \end{bmatrix}.$$

Kemudian, matriks H kedua didapat dengan cara memasukkan submatriks dari matriks H pertama kedalam matriks identitas.

$$H_2 = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & H & & \\ \vdots & & & \\ 0 & & & \end{pmatrix}$$

Proses tersebut terus dilakukan sampai didapat matriks segitiga atas (semua elemen matriks dibawah diagonal utama dibuat menjadi 0).

Matriks Q dan R didapat dengan cara sebagai berikut:

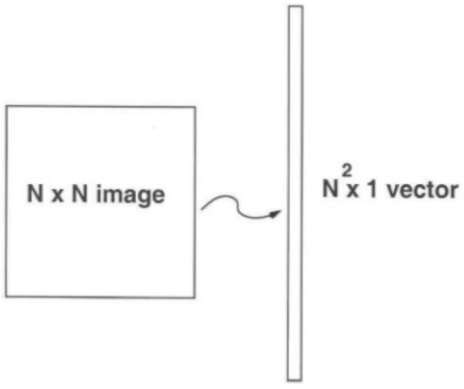
$$R = H_n \ldots H_3 H_2 H_1 A$$

$$Q = H_1 H_2 H_3 \ldots H_n$$

4. Eigenface dan Proses Pengenalan Wajah

Eigenface merupakan komponen-komponen utama dari suatu dataset gambar. Penjumlahan beberapa eigenface dengan *weight* tertentu akan menghasilkan gambar awal pada dataset. Eigenface inilah yang nantinya akan dijadikan komponen utama dalam proses pengenalan wajah.

Perhitungan eigenface dari suatu dataset gambar dengan jumlah gambar M dimulai dengan mengubah seluruh gambar (matriks 2D) dalam dataset menjadi suatu vektor 1D (*flattening*).



Kemudian setiap vektor gambar dikonkatenasi sehingga membentuk matriks $X = [\boldsymbol{x}_1\ \boldsymbol{x}_2 \ldots \boldsymbol{x}_M]$ dengan ukuran $N^2 \times M$, dengan $\boldsymbol{x}_i$ $(i = 1, 2, 3, \ldots, M)$ merupakan vektor gambar ke-i. Setelah

itu, hitung nilai mean dari setiap vektor gambar. Kemudian gunakan nilai mean untuk mengurangkan setiap vektor gambar dengan mean.

$$\boldsymbol{\psi} = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{x}_i$$

$$\boldsymbol{\phi}_i = \boldsymbol{x}_i - \boldsymbol{\psi}$$

Kemudian, nyatakan matriks $A = [\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2\ \ldots\ \boldsymbol{\phi}_M]$ dengan ukuran $N^2 \times M$.

Setelah itu, hitung matriks kovarian C dan hitung nilai eigen $\boldsymbol{\lambda}$ dan vektor eigen $\boldsymbol{z}_i$ dengan cara sebagai berikut:

$$C = AA^T \qquad C\boldsymbol{e}_i = \boldsymbol{\lambda}_i \boldsymbol{e}_i$$

Perhitungan diatas sangat tidak efektif karena melibatkan matriks dengan ukuran yang sangat besar, yaitu matriks C dengan ukuran $N^2 \times N^2$. Untuk mengatasi hal tersebut, dapat dihitung terlebih dahulu matriks $C' = A^T A$ dengan ukuran $M \times M$. Misalkan $\boldsymbol{v}_i$ dan $\boldsymbol{\mu}_i$ adalah vektor eigen dan nilai eigen dari matriks C'. Dengan demikian:

$$A^T A\boldsymbol{v}_i = \boldsymbol{\mu}_i \boldsymbol{v}_i$$

$$AA^T A\boldsymbol{v}_i = A\boldsymbol{\mu}_i \boldsymbol{v}_i$$

$$AA^T (A\boldsymbol{v}_i) = \boldsymbol{\mu}_i (A\boldsymbol{v}_i)$$

$$C(A\boldsymbol{v}_i) = \boldsymbol{\mu}_i (A\boldsymbol{v}_i)$$

Dari persamaan diatas, didapat hubungan sebagai berikut:

$$\boldsymbol{u}_i = A\boldsymbol{v}_i \qquad \boldsymbol{\lambda}_i = \boldsymbol{\mu}_i$$

Vektor eigen $\boldsymbol{u}_i$ yang diambil untuk prediksi hanyalah vektor eigen yang memiliki nilai eigen yang cukup besar, biasanya 5-30% vektor eigen pertama. Kemudian vektor eigen tersebut dinormalisasi dan disatukan menjadi sebuah matriks E dengan $\boldsymbol{e}_i$ merupakan vektor kolom yang merupakan suatu eigenface.

Matriks *weight* Y dari setiap eigenface dalam membentuk gambar pada *training* dataset dapat dihitung dengan cara berikut:

$$Y = E^T A$$

Kemudian untuk proses pengenalan wajah, gambar yang akan dikenali diubah menjadi bentuk vektor 1D P, kemudian dihitung *weight*-nya dengan cara sebagai berikut:

$$\boldsymbol{\omega} = E^T (P - \boldsymbol{\psi})$$

Kemudian, gambar yang dianggap paling mirip dengan gambar yang akan dikenali merupakan gambar dengan jarak *weight* $\boldsymbol{\omega}$ dengan y_i minimum. y_i merupakan vektor ke-I dari matriks Y. Jarak yang digunakan adalah jarak Euclidean. Misalkan s merupakan indeks gambar pada dataset training, maka

$$s = argmin([d(\boldsymbol{\omega}, \boldsymbol{y}_i)]) = argmin([\|\boldsymbol{\omega} - \boldsymbol{y}_i\|])$$

Gambar tidak akan dikenali sebagai salah satu gambar dalam dataset ketika jarak dari *weight* berada diatas suatu *threshold* θ.

BAB III

Implementasi Program

Alur Program:

1. Input gambar wajah

   User dapat memasukkan input ke program berupa folder yang berisi dataset gambar dan sebuah gambar untuk dicari kemiripan dengan dataset yang telah diberikan. Gambar input dapat diberikan melalui fitur kamera yang akan mengambil gambar wajah pengguna secara real-time untuk setelahnya diproses.

2. Pre-processing gambar

   Seluruh gambar pada folder dataset dan gambar input akan diproses menggunakan library OpenCV dan mediapipe untuk memudahkan pendeteksian wajah serta perhitungan eigenface. Pertama background dari gambar akan diubah menjadi warna putih menggunakan library mediapipe. Selanjutnya program akan mendeteksi wajah dan memotong bagian wajah dengan bentuk persegi. Setelah itu gambar akan diubah ukurannya menjadi 256 x 256 dan diubah menjadi citra grayscale. Terakhir, gambar akan didiubah menjadi sebuah vector dan dimasukkan ke sebuah matriks numpy yang menampung semua gambar. Gambar pun siap untuk diproses dan dicari eigenfacenya.

3. Perhitungan eigenface

   Pada langkah pre-processing telah didapat sebuah matriks numpy yang berisi vektor-vektor gambar. Kemudian, dicari mean dari seluruh vektor gambar. Lalu, setiap vektor gambar dikurangi dengan vektor mean. Hasil pengurangan tersebut dipakai untuk menghitung matriks kovarian. Setelah itu, dicari vektor eigen dengan cara mengalikan matriks kovarian dengan hasil pengurangan mean. Terakhir, 30% vektor eigen dengan nilai eigen terbesar diambil untuk menjadi eigenface.

4. Pengenalan wajah menggunakan eigenface

   Kemudian untuk proses pengenalan wajah atau identifikasi, gambar yang akan dikenali diubah menjadi bentuk vektor 1 dimensi, kemudian dihitung *weight*-nya. Kemudian, gambar yang dianggap paling mirip dengan gambar yang akan dikenali merupakan gambar dengan jarak *weight* dengan minimum. merupakan vektor ke-I dari matriks . Jarak yang digunakan adalah jarak Euclidean. Gambar tidak akan dikenali sebagai salah satu gambar dalam dataset ketika jarak dari *weight* berada diatas suatu *threshold*.

5. Program utama dan GUI

Nomor satu sampai empat merupakan Alur program untuk mendapat hasil deteksi wajah. Pada program utama alurnya sedikit ada perubahan. Pertama terdapat beberapa opsi kepada pengguna untuk memulai deteksi wajah test image yaitu :

a. Jika pengguna hanya ingin mendeteksi wajah satu kali dengan dataset yang siberikan maka, pengguna dapat memasukkan dataset dan test image yang ingin dideteksi wajahnya. Kemudian menekan tombol execute untuk memulai pendeteksian. Hasil gambar akan keluar pada bagian result image dan nama hasil wajah akan tampil pada bagian result di pojok kiri bawah.
b. Jika pengguna ingin melakukan tes uji untuk banyak test image dengan dataset yang sama. Memasukkan dataset terlebih dahulu kemudian menekan train image agar hasil pre-processing dari dataset dapat tersimpan selama tidak ada dataset baru yang ingin diuji. Setelah itu pengguna dapat memasukkan test image - tes image yang ingin diuji lalu menekan execute. Program akan dengan cepat melakukan identifikasi dan menampilkan hasilnya.
c. Jika pengguna ingin langsung menggunakan kamera untuk pendeteksian wajah, maka pengguna dapat menggunakan **fitur tambahan** yaitu terdapat tombol camera untuk memasukkan test image secara real time dengan kamera laptop atau komputer pengguna. Untuk selebihnya seperti train dataset terlebih dahulu atau tidak tetap bisa dilakukan seperti pada poin a dan b hanya berbeda di bagian input gambar yang digantikan dengan penangkapan gambar secara langsung dengan kamera. Ketika tombol kamera ditekan, tombol input test image berupa file akan langsung dimatikan (tidak bisa ditekan) selama kamera menyala.

```
dataset_img = preprocess(folderName)
print("Preprocess dataset DONE")
average_face = face_avg(dataset_img)
print("Mean dataset DONE")
normal = normalized_face(dataset_img, average_face)
print("normal dataset DONE")
covariance = covariance_mat(normal)
print("Covariance dataset DONE")
eig_val, eig_vec = eig_val_and_vec(covariance)
eig_vec_img = normal.T @ eig_vec
idx = identification(test_img, average_face, normal, eig_vec_img.T)
print("Identification image DONE")
```

Gambar di atas menunjukkan alur pendeteksi wajah dari pre-process sebuah dataset, file dan foto dari kamera sampai mendapat indeks hasil deteksi dari list gambar.

BAB IV

Eksperimen

Kasus mirip:

Kasus tidak mirip:

Kasus not found:

BAB V

Kesimpulan, saran, dan refleksi

I. Kesimpulan

Program pengenalan wajah telah kami implementasikan dengan metode QR decomposition sebagai salah satu aplikasi materi nilai eigen dan vektor eigen dari mata kuliah Aljabar Linear dan Geometri. Materi yang kami implementasikan dari program tersebut adalah sebagai berikut:

1. Nilai eigen dan vektor eigen

Nilai eigen dan vektor eigen merupakan komponen paling penting dalam perhitungan eigenface dengan metode QR dan menjadi tulang punggung algoritma perhitungan.

2. Penggunaan dekomposisi QR

Metode dekomposisi QR adalah salah satu metode dekomposisi matriks yang kami pilih untuk program kali ini. Dekomposisi

3. Penggunaan library OpenCV, numpy, dan Mediapipe

Kami menggunakan OpenCV untuk melakukan pre-processing pada image untuk memudahkan perhitungan eigenface dengan cara mengganti background, mendeteksi wajah dari image, mengubah size menjadi 256 x 256, mengubah warna image ke grayscale, serta memasukkan hasilnya ke array numpy untuk diproses.

4. Pembuatan GUI

Kami menggunakan library tkinter dalam pembuatan tampilan aplikasi. GUI yang kami buat masih tergolong yang sederhana karena keterbatasan kemampuan. Dalam GUI aplikasi yang kami buat terdapat beberapa bagian yaitu set up untuk window, deklarasi fungsi-fungsi (functions), dan widgets yang berupa tombol-tombol untuk langsung memanggil perintah dari tombol tersebut serta tombol-tombol switch untuk mengubah suatu kondisi.

5. Perhitungan Eigenface

Perhitungan dari eigenface memanfaatkan konsep nilai eigen dan vektor eigen. Nilai eigenface didapat dari perhitungan vektor eigen dari matriks kovarian yang diperoleh dari dataset training. Eigenface inilah yang dijadikan sebagai komponen utama dalam prediksi gambar.

II. Saran

Implementasi dari program pengenalan wajah dapat dilakukan dengan berbagai cara dan metode, untuk bagian pre-processing kami menyarankan untuk mengubah background menjadi warna putih

agar mudah untuk dilakukan pendeteksian wajah, serta mengubah matriks image ke sebuah flattened matrix agar memudahkan perhitungan.

Pada bagian perhitungan eigenface kami menyarankan untuk menggunakan metode dekomposisi QR dari semua metode yang ada karena jauh lebih mudah untuk pemrosesan secara komputasional.

III. Refleksi

Dalam proses mengerjakan Tugas Besar 2 IF 2123 Aljabar Linier dan Geometri: Aplikasi Nilai Eigen dan EigenFace pada Pengenalan Wajah, kami mendapatkan banyak insight mengenai berbagai macam library yang tersedia di python dan cara implementasinya. Selain itu, kami juga memperoleh banyak pelajaran mengenai aplikasi nilai eigen, vektor eigen, dan dekomposisi matriks dalam kehidupan sehari-hari. Tentunya kami juga belajar banyak tentang cara membuat GUI dan cara implementasinya dalam bentuk web application. Selain ilmu, kami juga belajar banyak mengenai pengembangan diri dan kinerja kelompok. Berbagai kendala yang kami lalui seperti timeline ataupun deadline pengerjaan dapat menjadi sumber introspeksi untuk meningkatkan diri kedepannya. Terakhir, kami harap melalui tugas besar ini kami dapat terus belajar dalam mengimplementasikan materi kuliah Aljabar Linear dan Geometri maupun image processing.

Daftar Referensi

1. **https://saintif.com/perkalian-matriks/**
2. **https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2022-2023/algeo22-23.htm**
3. **https://rosettacode.org/wiki/QR_decomposition**
4. **Numerical Linear Algebra with Applications Using MATLAB – 2014 – William Ford**
5. **https://learnopencv.com/introduction-to-mediapipe/**
6. **https://www.youtube.com/watch?v=yQSEXcf6s2I&list=PLCC34OHNcOtoC6GglhF3ncJ5rLwQrLGnV**
7. **https://jurnal.untan.ac.id/index.php/jcskommipa/article/view/9727**
8. *Marijeta, Slavković, & Dubravka, Jevtić. 2012.* Face Recognition Using Eigenface Approach. *(Innovation Center of School of Electrical Engineering, Belgrade)* (**http://www.doiserbia.nb.rs/Article.aspx?ID=1451-48691201121S#.Y3zNUHZBxD8)**
9. **https://www.educative.io/answers/how-to-capture-a-single-photo-with-webcam-using-opencv-in-python**

Lampiran

Link Repo: https://github.com/hanifmz07/Algeo02-21043

Link Demo: FaceIT - Face Recognition