LAPORAN TUGAS BESAR

ALJABAR LINIER DAN GEOMETRI

APLIKASI NILAI EIGEN DAN EIGENFACE PADA PENGENALAN WAJAH (*FACE RECOGNITION*)

Kelas Mahasiswa K03

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

Diajukan sebagai tugas besar Mata Kuliah IF2123 Aljabar Linier dan Geometri pada Semester I Tahun Akademik 2022/2023

Disusun Oleh:

Kelompok 10 (CJ ★)

Bintang Hijriawan	**13521003**
Jason Rivalino	**13521008**
Christophorus Dharma Winata	**13521009**

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

IF2123
Aljabar Linear dan Geometri

DAFTAR ISI

DAFTAR ISI 3

BAB I: DESKRIPSI MASALAH 4

BAB II: TEORI SINGKAT 5

- **I. Teori Singkat Perkalian Matriks** 5
- **II. Teori Singkat Nilai Eigen** 5
- **III. Teori Singkat Vektor Eigen** 6
- **IV. Teori Singkat Eigenface** 7

BAB III: IMPLEMENTASI PROGRAM 8

- **I. Penjelasan Tech-Stack pada Program** 8
- **II. Garis Besar Algoritma Program** 9

BAB IV: EKSPERIMEN 11

BAB V: KESIMPULAN, SARAN & REFLEKSI 12

- **I. Kesimpulan** 12
- **II. Saran** 12
- **III. Refleksi** 12

REFERENSI 14

LAMPIRAN 15

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.

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, yang akan diimplementasikan adalah untuk membuat sebuah program pengenalan wajah menggunakan metode *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.

BAB II

TEORI SINGKAT

I. Teori Singkat Perkalian Matriks

Perkalian Matriks adalah suatu operasi biner pada dua buah matriks untuk menghasilkan sebuah matriks. Perkalian dari dua buah matriks A dan B dapat dinyatakan dalam bentuk AB. Syarat dari perkalian dua buah matriks adalah banyaknya kolom dari matriks pertama harus sama banyak dengan banyak baris dari matriks kedua.[2] Bentuk perkalian matriks sendiri dapat dirumuskan sebagai berikut:

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

Matriks A merupakan matriks yang berukuran m x n dan matriks B merupakan matriks yang berukuran n x p. Apabila kedua matriks ini dikalikan, maka akan membentuk matriks C dengan bentuk sebagai berikut:

$$\mathbf{C} = \begin{pmatrix} a_{11}b_{11} + \cdots + a_{1n}b_{n1} & a_{11}b_{12} + \cdots + a_{1n}b_{n2} & \cdots & a_{11}b_{1p} + \cdots + a_{1n}b_{np} \\ a_{21}b_{11} + \cdots + a_{2n}b_{n1} & a_{21}b_{12} + \cdots + a_{2n}b_{n2} & \cdots & a_{21}b_{1p} + \cdots + a_{2n}b_{np} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{11} + \cdots + a_{mn}b_{n1} & a_{m1}b_{12} + \cdots + a_{mn}b_{n2} & \cdots & a_{m1}b_{1p} + \cdots + a_{mn}b_{np} \end{pmatrix}$$

II. Teori Singkat Nilai Eigen

Nilai Eigen adalah nilai karakteristik dari sebuah matriks yang memiliki ukuran n x n. Nilai Eigen sendiri dalam perhitungan dinyatakan dalam bentuk lambang λ. Nilai Eigen sendiri dinyatakan berasal dari persamaan Ax = λx, dengan λ menyatakan Nilai Eigen dari vektor A dan x menyatakan Vektor Eigen yang berkoresponden dengan λ.[5]

Berdasarkan persamaan Ax = λx, proses untuk mencari nilai-nilai eigen dapat dinyatakan dengan rumus:

$$Ax = \lambda x$$

$$IAx = \lambda Ix$$

$$Ax = \lambda Ix$$

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

Untuk mendapatkan nilai eigen dari vektor A, maka vektor x harus merupakan vektor tak nol. Sehingga, untuk mendapatkan nilai eigen harus memenuhi kondisi teorema berikut:

$$det\,(\lambda I - A) = 0$$

Notasi I dalam rumus ini berarti matriks identitas. Sedangkan notasi A sendiri merupakan matriks yang ingin dicari nilai eigennya. Operasi dilakukan dengan melakukan perkalian λ yang merupakan nilai eigen yang ingin dicari dengan matriks identitas, kemudian matriks hasil perkalian dikurangi dengan matriks yang ingin dicari nilai eigennya. Kemudian mencari determinan dari operasi yang telah dinyatakan sebelumnya dan kemudian mencari nilai eigen yang memenuhi dari hasil determinan yang ada.[7]

III. Teori Singkat Vektor Eigen

Vektor Eigen adalah vektor yang menyatakan matriks kolom yang apabila dikalikan dengan sebuah matriks n x n, akan menghasilkan vektor lain yang merupakan hasil kelipatan dari vektor itu sendiri. Vektor eigen secara geometris akan menyusut atau memanjang sesuai dengan faktor nilai eigen (λ). Jika nilai eigen positif, maka vektor akan memanjang ke arah yang sama dan memanjang arah berkebalikan jika nilai eigen adalah negatif.

Untuk menentukan nilai vektor eigen, cara yang dapat dilakukan adalah dengan mengalikan matriks bentuk $\lambda I - A$ dengan matriks x yang akan dicari nilai vektor eigennya. λ sendiri merupakan nilai eigen yang telah didapat pada proses pencarian nilai eigen sebelumnya. Vektor eigen yang didapatkan akan memiliki nilai berbentuk parametrik yang kemudian hasil ini akan dimanfaatkan untuk membentuk ruang eigen (*eigenspace*).[7]

IV. Teori Singkat *Eigenface*

Eigenface adalah algoritma yang digunakan untuk menyelesaikan permasalahan terkait dengan pengenalan wajah manusia. Algoritma ini sendiri berdasar pada *Principle Component Analysis* (PCA). Prinsip pengenalan wajahnya adalah dengan memindai informasi berupa matriks dari hasil foto wajah yang kemudian di *encode* dan dibandingkan dengan hasil *decode* yang sebelumnya telah dilakukan. *Decoding* yang dilakukan adalah dengan menghitung nilai *Eigenvector* yang kemudian direpresentasikan dengan matriks berukuran besar.[3] Untuk tahapan-tahapan yang ada pada algoritma *Eigenface* ini antara lain:

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

$$S = (\Gamma 1, \Gamma 2, \ldots, \Gamma M) \quad (1)$$

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)

$$L = A^T A \qquad L = \phi_m^T \phi_n$$

$$C = \frac{1}{M}\sum_{n=1}^{M}\phi_n \phi_n^T = AA^T \quad (4)$$

5. Langkah kelima menghitung eigenvalue (λ) dan eigenvector (v) dari matriks kovarian (C)

$$C \text{ x } \nu i = \lambda i \text{ x } \nu i \quad (5)$$

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

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

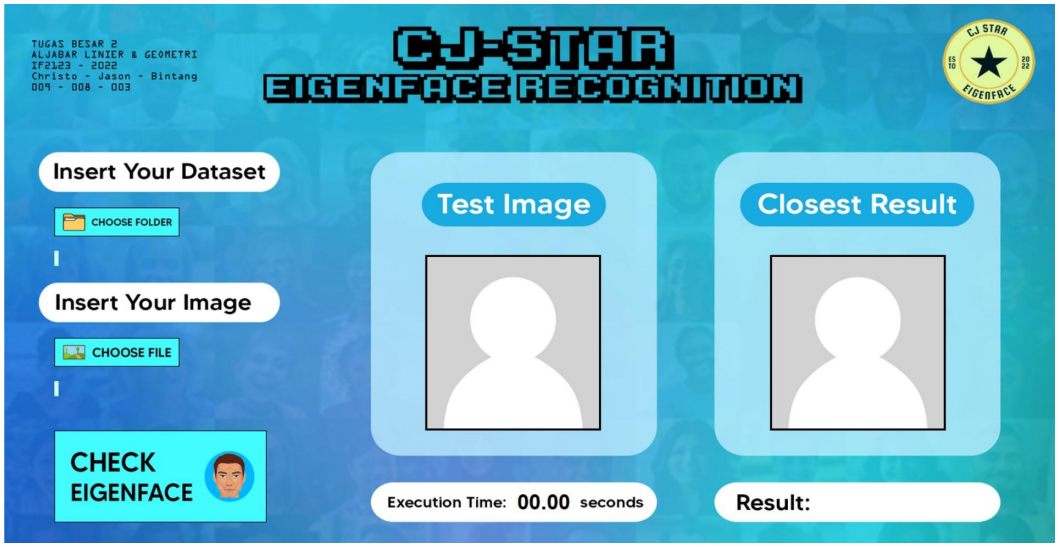
$l = 1, \ldots, M$

BAB III

IMPLEMENTASI PROGRAM

I. Penjelasan *Tech-Stack* pada Program

Tech-Stack dari program yang kami buat memanfaatkan Bahasa Pemrograman Python untuk pembuatan tampilan program (Front-End) dan untuk pembuatan struktur data dari program yang akan dijalankan (Back-End). Untuk bagian tampilan program, kelompok kami memanfaatkan Tkinter *library* untuk pembuatan aplikasi antarmuka pada Python. Dalam library ini terdapat berbagai macam fitur seperti fitur untuk mengimport gambar, fitur untuk membuat label tulisan, fitur untuk membuat tombol untuk memproses program, dll. Selain Tkinter, program kami juga memakai PIL *library* yang berfungsi untuk mengimport foto dari data komputer kedalam program. Untuk tampilan dari aplikasi antarmuka yang kelompok kami buat adalah sebagai berikut:



Pada tampilan GUI program yang kami buat, tampilan yang ada pada program memiliki tema berwarna biru. Pada bagian atas tengah aplikasi, terdapat nama kelompok dan judul “Eigenface Recognition”. Untuk bagian kanan atas, terdapat logo dari kelompok dan bagian kiri atas terdapat identitas dari anggota kelompok. Untuk bagian kiri dari tampilan program, terdapat tombol-tombol untuk menginput data, dan tombol untuk melakukan *run* menjalankan program. Pada bagian tengah, terdapat

tampilan dari foto yang akan dicari dan hasil foto yang terdekat dari foto tes serta ada waktu eksekusi dan hasil persentase yang didapat dari proses *run* program.

Untuk bagian struktur data dari program atau Back-End, kelompok kami membuat berbagai macam fungsi untuk melakukan operasi pada matriks dengan memanfaatkan *library* numpy. Operasi-operasi dari matriks ini yang berikutnya akan dipergunakan untuk melakukan perhitungan pada matriks yang ada pada foto untuk mendapatkan nilai-nilai yang diperlukan untuk proses pencarian. Selain itu, kami juga menggunakan *library* os yang berfungsi untuk membantu proses pembacaan data foto dalam *dataset*. Terakhir, ada library cv2 yang digunakan untuk menampilkan foto yang paling pas yang didapat dari berbagai macam proses perhitungan matriks foto.

II. Garis Besar Algoritma Program

1) Garis besar program bagian Front-End

Untuk penjelasan algoritma Front-End, dalam program, terdapat tombol untuk memilih folder *dataset* dan juga tombol untuk memilih *file* foto. Setelah folder *dataset* dan *file* foto sudah dipilih, foto yang dipilih akan langsung muncul dalam Test Image dan *direction* dari foto dan folder yang dipilih juga akan langsung muncul pada bagian bawah tombol. Untuk memeriksa kecocokan *file* dengan *dataset*, pengguna dapat mengklik tombol Check Eigenface untuk langsung melakukan pengecekan. Jika semua kebutuhan data sudah tersedia dan tombol diklik, waktu eksekusi akan langsung berjalan dan akan berhenti saat program sudah menampilkan foto yang paling sesuai antara foto dari *file* input dan juga foto yang terdapat pada *dataset*. Pada bagian bawah kanan program yaitu pada bagian Result, program juga akan menampilkan persentase tingkat kecocokan wajah dari proses pengecekan *file* foto dengan *dataset*. Jika pengguna mengganti foto atau folder dari dataset, maka waktu dan result akan langsung melakukan reset.

2) Garis besar program bagian Back-End

Untuk penjelasan algoritma Back-End, kelompok kami membuat berbagai fungsi matriks, seperti fungsi untuk membuat matriks, membuat matriks identitas, operasi matriks seperti pertambahan, pengurangan, dan perkalian skalar dan dot

matriks. Selain itu, ada fungsi untuk menyalin matriks, menggabungkan matriks, dan membuat matriks vektor beserta dengan operasi vektornya dan perhitungan ortogonal.

Selanjutnya, untuk pembuatan programnya, terdapat fungsi untuk akses foto dari folder-folder yang diinput oleh pengguna. Foto-foto ini kemudian akan dihitung jumlahnya dan diubah dalam bentuk *grayscale* terlebih dahulu dan kemudian akan dicari matriks dalam fotonya. Setelah foto diproses dan matriks didapatkan, langkah berikutnya adalah menghitung nilai *mean* dari total matriks dalam foto dengan menjumlahkan semua matriks lalu membaginya dengan total matriks, lalu nilai matriks pada foto akan dikurangi dengan nilai *mean* yang didapat dan melakukan perkalian dari bentuk matriks hasil pengurangan sebelumnya dengan transpose matriksnya hingga didapatkan nilai kovarian. Dari perhitungan nilai *mean* hingga mendapatkan matriks *covariant*, semua dijadikan satu dalam sebuah fungsi yaitu fungsi **deltaMeanandCovariant** dalam programnya (fungsi ini menerima input Matriks 3D dan mengembalikan nilai *mean* dan *covariant*). Langkah berikutnya adalah membuat fungsi untuk mencari nilai eigen dan vektor eigen. Untuk melakukan pencarian ini, kelompok kami menggunakan metode **QR Decomposition** dengan membuat fungsi untuk menghitung matriks QR (fungsi ini menerima input matriks dan mengembalikan nilai Q dan R). Dari matriks QR ini kemudian bisa dicari nilai eigen dan vektor eigennya dengan membuat fungsi perhitungan eigen. Setelah ditemukan vektor Eigen, langkah selanjutnya adalah membuat fungsi untuk mencari **Eigenface** dengan mengalikan nilai *mean* dengan ukuran kolom lalu hasilnya diappend hingga nilai Eigenface pun didapat (fungsi menerima vektor eigen dan juga nilai *mean*).

Untuk proses pencarian nilai terdekat pada foto, kelompok kami membuat fungsi untuk menghitung koefisien kombinasi linier dengan fungsi **omega** yang terdiri atas omega untuk *file* foto dan juga untuk *dataset* (fungsi ini menerima nilai Eigenface dan nilai *mean*). Hasil dari fungsi omega ini adalah berupa weight yang berikutnya akan dipakai untuk mencari nilai **Euclidean Distance.** Selain proses perhitungan dan pencarian foto, terdapat juga fungsi untuk memproses input, dataset, dan juga proses dari Face Recognition.

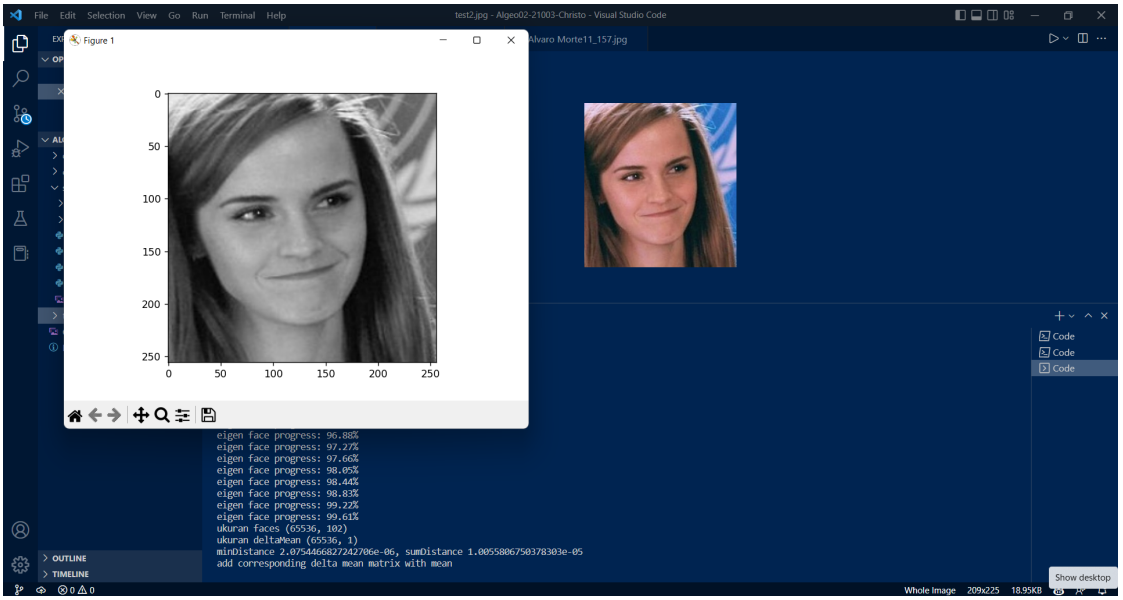
Untuk eksekusi pada bagian main, program kami akan menerima fungsi **deltaMean, meanTraining, datasetEigFaces, weightTraining** sebagai fungsi untuk

membaca dataset dan fungsi **inputEigFaces, weightInput** sebagai fungsi untuk membaca *file* foto. Berikutnya, program akan memproses nilai Euclidean Distance untuk mendapatkan nilai batas Euclidean. Setelah nilai batas Euclidean didapat, program akan menampilkan hasil foto dengan nilai Euclidean Distance yang paling kecil.

IF2123
Aljabar Linear dan Geometri

BAB IV

EKSPERIMEN



BAB V

KESIMPULAN, SARAN & REFLEKSI

I. Kesimpulan

Dari tugas besar ini dapat disimpulkan hal-hal sebagai berikut:

1) Operasi-operasi dasar pada matriks dan vektor yang sudah dipelajari dapat bermanfaat untuk membantu mencari kovarian, nilai eigen, vektor eigen, dan juga matriks Eigenface yang bermanfaat untuk membantu mendeteksi kemiripan wajah.
2) Pencarian nilai eigen dan juga vektor eigen dapat dilakukan dengan memanfaatkan metode QR Decomposition
3) Penggunaan *library* untuk operasi perhitungan seperti numpy dan matplotlib sangat membantu untuk mempermudah proses perhitungan pada matriks

II. Saran

Saran untuk penyelesaian masalah ini untuk kedepannya adalah sebagai berikut:

1) Pembuatan algoritma bisa dibuat menjadi lebih efisien lagi agar waktu *running* program menjadi lebih cepat dan efektif juga
2) Terdapat beberapa kesalahan dalam program yang dibuat sehingga kedepannya bisa dikembangkan menjadi lebih baik

III. Refleksi

Dari pengerjaan tugas besar ini, hal penting yang didapat adalah mengenai pentingnya komunikasi dan kerjasama yang baik antar anggota kelompok. Dalam tugas ini, setiap anggota kelompok memiliki tugas masing-masing seperti ada yang membuat tampilan program, ada yang membuat perhitungan matriks, ada yang membuat fungsi untuk mencari eigen dan euclidean, dll. Tentunya pekerjaan tersebut tidak dapat dilakukan sendiri dan harus membutuhkan kolaborasi bersama dalam kelompok. Tugas ini juga memberikan pelajaran mengenai aplikasi dari berbagai materi yang sudah didapat pada mata kuliah Aljabar Linier dan Geometri ini yang tentunya juga akan berguna untuk mempersiapkan kuliah yang lebih lanjut pada masa yang akan datang.

REFERENSI

[1] Amos, David. "Python GUI Programming With Tkinter – Real Python." *Real Python*, https://realpython.com/python-gui-tkinter/. (Diakses pada tanggal 4 November 2022)

[2] Nykamp, Duane. "Multiplying matrices and vectors." *Math Insight*, https://mathinsight.org/matrix_vector_multiplication (Diakses pada tanggal 13 November 2022)

[3] Muliawan, Muhammad Rizki. "IMPLEMENTASI PENGENALAN WAJAH DENGAN METODE EIGENFACE PADA SISTEM ABSENSI". https://drive.google.com/file/d/1Qxxg_M69foCOBkh6Qr9aRTrwC86zK_rH/view. (Diakses pada tanggal 13 November 2022)

[4] "ML | Face Recognition Using Eigenfaces (PCA Algorithm)." *GeeksforGeeks*, https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/. (Diakses pada tanggal 13 November 2022)

[5] "Definisi: Nilai Eigen dan Vektor Eigen." *JAGOSTAT.com*, https://jagostat.com/aljabar-linear/nilai-eigen-dan-vektor-eigen. (Diakses pada tanggal 20 November 2022)

[6] Valverde, Juan Miguel. "[Explanation] Face Recognition using Eigenfaces – Lipman's Artificial Intelligence Directory." *Lipman's Artificial Intelligence Directory*, http://laid.delanover.com/explanation-face-recognition-using-eigenfaces/. (Diakses pada tanggal 21 November 2022)

[7] "Face recognition using eigenfaces - Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Soci." *UCSB Computer Science*, https://sites.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf. (Diakses pada tanggal 22 November 2022)

[7] R. Munir (2022). Nilai Eigen dan Vektor Eigen Bagian 1 [Powerpoint Slides]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-Nilai-Eigen-dan-Vektor-Eigen-Bagian1.pdf

[8] "The QR Algorithm for Finding Eigenvectors [Powerpoint Slides]." *Department of Computer Science and Engineering*, https://cse.buffalo.edu/faculty/miller/Courses/CSE633/Eric-Mikida-Fall-2011.pdf.

IF2123
Aljabar Linear dan Geometri

LAMPIRAN

Pranala Github: https://github.com/bintang433/Algeo02-21003