

**Tugas Besar 2 IF 2123 Aljabar Linier dan Geometri**  
**Aplikasi Nilai Eigen dan Vektor Eigen dalam Kompresi Gambar**



Laporan ini dibuat untuk memenuhi tugas  
Mata Kuliah IF 2123 Aljabar Linier dan Geometri

Disusun Oleh:  
Kelompok 5

Vionie Novencia Thanggestyo (13520006)

Vincent Ho (13520093)

Vincent Christian Siregar (13520136)

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**SEMESTER I TAHUN 2021/2022**

## DAFTAR ISI

<b>DAFTAR ISI</b>	<b>2</b>
<b>BAB I</b>	<b>3</b>
<b>DESKRIPSI TUGAS</b>	<b>3</b>
<b>BAB II</b>	<b>4</b>
<b>DASAR TEORI</b>	<b>4</b>
Perkalian Matriks	4
Nilai Eigen dan Vektor Eigen	4
Singular Value Decomposition (SVD)	5
<b>BAB III</b>	<b>7</b>
<b>IMPLEMENTASI PUSTAKA DAN PROGRAM</b>	<b>7</b>
Tech Stack	7
Nilai Eigen dan Vektor Eigen	7
SVD	7
Proses membaca gambar dan menyimpan gambar	8
<b>BAB IV</b>	<b>9</b>
<b>EKSPERIMEN</b>	<b>9</b>
Test case 1	9
Test case 2	10
Test case 3	11
<b>BAB V</b>	<b>12</b>
Kesimpulan	12
Saran	12
Refleksi	12
<b>Referensi</b>	<b>13</b>

## **BAB I**

### **DESKRIPSI TUGAS**

Buatlah program kompresi gambar dengan memanfaatkan algoritma SVD dalam bentuk website lokal sederhana. Spesifikasi website adalah sebagai berikut:

1. Website mampu menerima file gambar beserta input tingkat kompresi gambar (dibebaskan formatnya).
2. Website mampu menampilkan gambar input, output, runtime algoritma, dan persentase hasil kompresi gambar (perubahan jumlah pixel gambar).
3. File output hasil kompresi dapat diunduh melalui website.
4. Kompresi gambar tetap mempertahankan warna dari gambar asli.
5. (Bonus) Kompresi gambar tetap mempertahankan transparansi dari gambar asli, misal untuk gambar png dengan background transparan.
6. Bahasa pemrograman yang boleh digunakan adalah Python, Javascript, dan Go.
7. Penggunaan framework untuk back end dan front end website dibebaskan. Contoh framework website yang bisa dipakai adalah Flask, Django, React, Vue, dan Svelte.
8. Kalian dapat menambahkan fitur fungsional lain yang menunjang program yang anda buat (unsur kreativitas diperbolehkan/dianjurkan).
9. Program harus modular dan mengandung komentar yang jelas.
10. Diperbolehkan menggunakan library pengolahan citra seperti OpenCV2, PIL, atau image dari Go.
11. Dilarang menggunakan library perhitungan SVD dan library pengolahan eigen yang sudah jadi.

## BAB II

### DASAR TEORI

#### Perkalian Matriks

Misalkan matriks  $A = (a,b,c,d)$  berukuran  $2 \times 2$  dikalikan dengan matriks  $B = (e,f,g,h)$  berukuran  $2 \times 2$ , sehingga rumusnya akan menjadi:

$$A \times B$$
$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} j & k \\ l & m \end{pmatrix}$$
$$A \times B = \begin{pmatrix} aj + bl & ak + bm \\ cj + dl & ck + dm \end{pmatrix}$$

Syarat dua matriks dapat dioperasikan perkalian yaitu banyak kolom matriks pertama harus sama dengan banyak baris matriks kedua, sebagai berikut:

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

#### Nilai Eigen dan Vektor Eigen

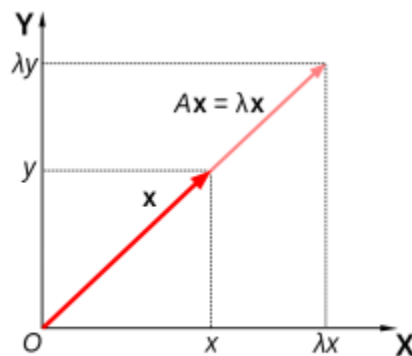
Jika  $A$  adalah matriks  $n \times n$  maka vektor tidak nol  $x$  di  $R^n$  disebut vektor eigen dari  $A$  jika  $Ax$  sama dengan perkalian suatu skalar  $\lambda$  dengan  $x$ , yaitu

$$Ax = \lambda x$$

Skalar  $\lambda$  disebut nilai eigen dari  $A$ , dan  $x$  dinamakan vektor eigen yang berkorespondensi dengan  $\lambda$ .

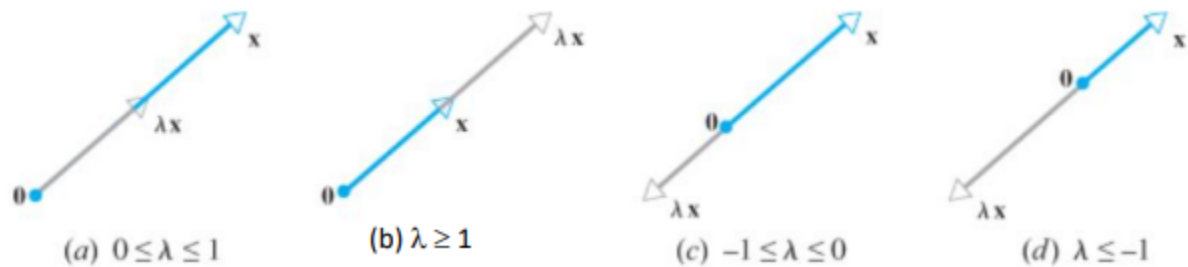
Nilai eigen menyatakan nilai karakteristik dari sebuah matriks yang berukuran  $n \times n$ .

Vektor eigen  $x$  menyatakan vektor kolom yang apabila dikalikan dengan sebuah matriks  $n \times n$  menghasilkan vektor lain yang merupakan kelipatan vektor itu sendiri.



Sumber gambar: Wikipedia

Dengan kata lain, operasi  $Ax = \lambda x$  menyebabkan vektor  $x$  menyusut atau memanjang dengan faktor  $\lambda$  dengan arah yang sama jika  $\lambda$  positif dan arah berkebalikan jika  $\lambda$  negatif.



### ***Singular Value Decomposition (SVD)***

Algoritma SVD didasarkan pada teorema dalam aljabar linier yang menyatakan bahwa sebuah matriks dua dimensi dapat dipecah menjadi hasil perkalian dari 3 sub-matriks yaitu matriks ortogonal  $U$ , matriks diagonal  $S$ , dan transpose dari matriks ortogonal  $V$ . Dekomposisi matriks ini dapat dinyatakan sesuai persamaan berikut.

$$A_{m \times n} = U_{m \times m} S_{m \times n} V_{n \times n}^T$$

Gambar x. Algoritma SVD

Matriks  $U$  adalah matriks yang kolomnya terdiri dari vektor eigen ortonormal dari matriks  $A^T A$ . Matriks ini menyimpan informasi yang penting terkait baris-baris matriks awal, dengan informasi terpenting disimpan di dalam kolom pertama. Matriks  $S$  adalah matriks diagonal yang berisi akar dari nilai eigen matriks  $U$  atau  $V$  yang terurut menurun. Matriks  $V$  adalah matriks yang kolomnya terdiri dari vektor eigen ortonormal dari matriks  $A A^T$ . Matriks ini menyimpan informasi yang penting terkait kolom-kolom matriks awal, dengan informasi terpenting disimpan dalam baris pertama.



Gambar 2. Ilustrasi Algoritma SVD dengan rank  $k$

Dapat dilihat di gambar di atas bahwa dapat direkonstruksi gambar dengan banyak singular values  $k$  dengan mengambil kolom dan baris sebanyak  $k$  dari  $U$  dan  $V$  serta singular value

sebanyak  $k$  dari  $S$  atau  $\Sigma$  terurut dari yang terbesar. Kita dapat mengaproksimasi suatu gambar yang mirip dengan gambar aslinya dengan mengambil  $k$  yang jauh lebih kecil dari jumlah total singular value karena kebanyakan informasi disimpan di singular values awal karena singular values terurut mengecil. Nilai  $k$  juga berkaitan dengan rank matriks karena banyaknya singular value yang diambil dalam matriks  $S$  adalah rank dari matriks hasil, jadi dalam kata lain  $k$  juga merupakan rank dari matriks hasil. Maka itu matriks hasil rekonstruksi dari SVD akan berupa informasi dari gambar yang terkompresi dengan ukuran yang lebih kecil dibanding gambar awal.

## BAB III

### IMPLEMENTASI PUSTAKA DAN PROGRAM

#### Tech Stack

Program ini dibuat dengan bahasa pemrograman Python dan menggunakan HTML untuk tampilan frontend website.

Library yang digunakan sebagai berikut

- Flask (2.0.2) -> Untuk pembuatan web application
- imageio (2.10.1) -> Untuk mengubah gambar menjadi array
- numpy (1.21.3) -> Untuk mengolah array dan matriks
- Pillow (8.4.0) -> Untuk merekonstruksi array menjadi gambar
- Werkzeug (1.0.1) -> Menghasilkan filename yang secure
- Bootstrap 5 -> Framework CSS

#### Nilai Eigen dan Vektor Eigen

Pada program ini, kami menggunakan pendekatan qr algorithm untuk mencari nilai eigen dan vektor eigen. Misalkan matriks yang akan dicari nilai eigen dan vektor eigennya adalah matriks A. Tahapan awal yang dilakukan adalah menduplikat matriks A ke dalam variabel baru misalkan M dan membuat matriks identitas dengan ukuran yang sama dengan matriks A. Setelah itu, cari nilai qr dari matriks yang sudah duplikat dan disimpan dalam matriks Q dan R. Matriks R dan Q dikalikan dengan perkalian matriks biasa. Hasil dari perkalian matriks ini akan membentuk matriks diagonal dengan nilai diagonalnya berupa nilai eigen dari matriks awal. Matriks identitas dikalikan dengan Q dan menghasilkan matriks *normalized* vektor eigen.

#### SVD

Tahapan dalam mendapatkan SVD dari matriks A pada program ini adalah mencari matriks singular kanan dengan melakukan perkalian matriks A transpose dengan matriks A. matriks yang dicari adalah matriks U(matriks singular kiri), sigma(matriks singular), VT (transpose matriks singular kanan). Terapkan metode qr yang sudah dijelaskan sebelumnya untuk mencari nilai *normalized* vektor eigen dan nilai eigen. *Normalized* vektor eigen ini merupakan nilai V. Nilai eigen yang bukan 0 diakarkan untuk menghasilkan nilai singular yaitu matriks Sigma. Matriks U dihasilkan dari mengalikan matriks awal dikalikan dengan V lalu dikalikan dengan invers dari Sigma. V ditranspose untuk menghasilkan VT

### **Proses membaca gambar dan menyimpan gambar**

Pada program ini menggunakan library imageio untuk membaca gambar dan menggunakan pillow atau PIL untuk menyimpan gambar. Imageio digunakan untuk membaca file gambar (jpg, jpeg, atau png) dan menghasilkan matriks channel (RGB) dari gambar yang dibaca. Matriks tersebut didekomposisikan menggunakan metode svd untuk setiap channel. Matriks yang sudah dilakukan svd di *construct* menjadi matriks baru dan diubah ke image dengan menggunakan PIL.

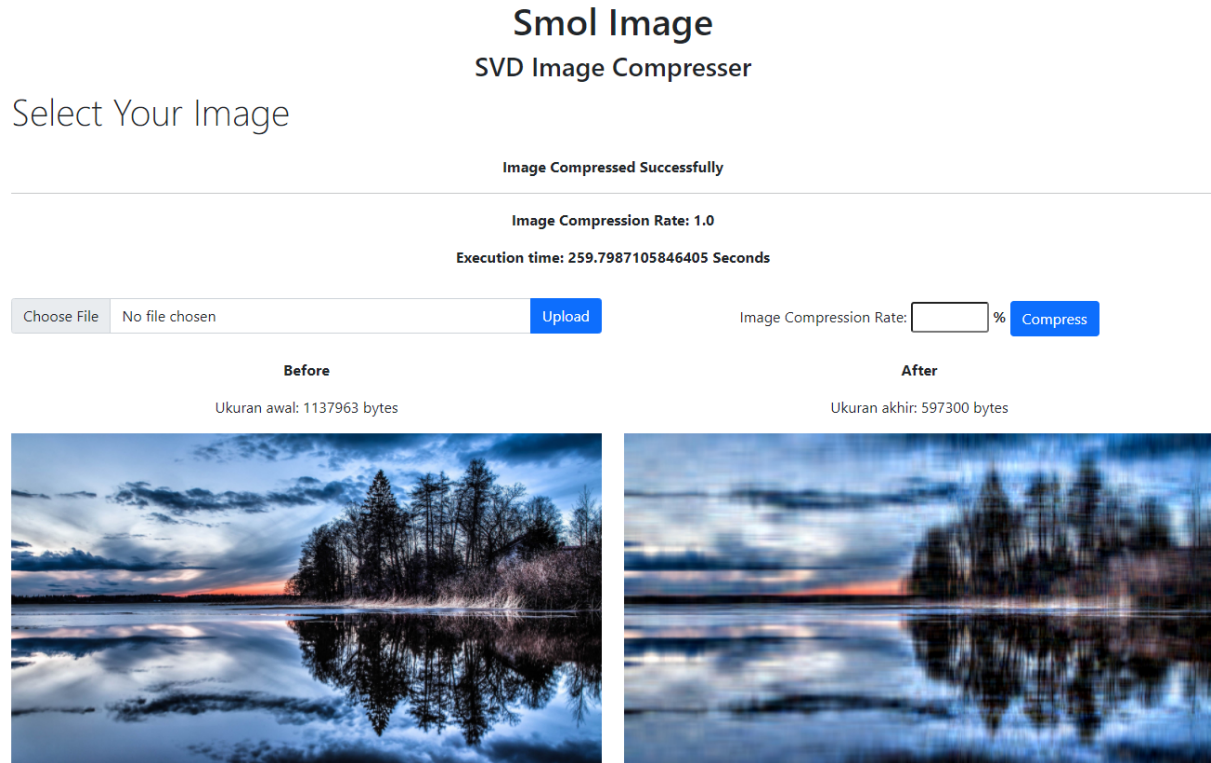


## BAB IV

### EKSPERIMEN

#### Test case 1

Gambar yang diinput adalah gambar landscape resolusi 4K *full color* dengan ukuran awal 1MB yang kemudian direkonstruksi dengan tingkat kompresi 1%. Program kemudian menghasilkan gambar yang sudah direkonstruksi dengan ukuran 584 kb seperti gambar dibawah ini. Waktu yang diperlukan untuk memproses gambar tersebut adalah 259.79 detik.



Gambar 4.1 Gambar Hasil Test Case 1

## Test case 2

Gambar yang diunggah adalah gambar grayscale dengan ukuran awal 249kb. Gambar kemudian direkonstruksi dengan tingkat konstruksi 20%. Program kemudian menghasilkan gambar yang sudah direkonstruksi dengan ukuran 131 kb. Waktu yang diperlukan untuk memproses gambar tersebut adalah 10.79 detik.

### Smol Image

#### SVD Image Compressor

Select Your Image

Image Compressed Successfully

Image Compression Rate: 20.0

Execution time: 10.791983127593994 Seconds

Choose File

No file chosen



Upload

Image Compression Rate:  % 

Compress

**Before**  
Ukuran awal: 254217 bytes

**After**  
Ukuran akhir: 133636 bytes



Download Image

Gambar 4.2 Gambar Hasil Test Case

### Test case 3

Gambar awal yang diunggah adalah gambar dengan ekstensi png dengan ukuran awal 4130 kb. Kemudian, gambar ini di kompres dengan tingkat kompresi 17% sehingga didapatkan hasil seperti pada gambar dibawah ini. Ukuran file setelah proses kompresi sebesar 3201 kb yang dapat dilihat pada gambar dibawah ini. Waktu yang diperlukan untuk memproses gambar tersebut adalah 66.15 detik.

## SVD Image Compressor

### Select Your Image

Image Compressed Successfully

Image Compression Rate: 17.0

Execution time: 66.15126156806946 Seconds

Choose File

No file chosen


Upload

Image Compression Rate:  % 

Compress


Before

Ukuran awal: 4228227 bytes



After

Ukuran akhir: 3277021 bytes



Download Image

Gambar 4.3 Gambar Hasil Test Case 3

## **BAB V**

### **PENUTUP**

#### **Kesimpulan**

Program kami mampu menyelesaikan masalah-masalah yang didefinisikan, yakni:

1. Website mampu menerima file gambar beserta input tingkat kompresi gambar.
2. Website mampu menampilkan gambar input, output, runtime algoritma, dan persentase hasil kompresi gambar.
3. File output hasil kompresi dapat diunduh melalui website
4. Hasil kompresi gambar tetap mempertahankan warna dari gambar asli

#### **Saran**

- a. Untuk memperindah tampilan dapat ditambahkan progress bar saat image sedang melakukan proses kompresi.
- b. Memulai pengerjaan tugas lebih awal agar tidak kewalahan di akhir saat mendekati *deadline* pengumpulan

#### **Refleksi**

Tugas ini mendorong penulis untuk belajar memanfaatkan berbagai macam library pada bahasa pemrograman python dan juga belajar membuat website menggunakan flask yang dapat menerima input gambar dan mengembalikan gambar yang telah di proses. Penulis juga belajar mengenai cara mengimplementasikan pengetahuan yang didapatkan dari mata kuliah Aljabar Linear dan Geometri kedalam sebuah program, berkolaborasi menggunakan platform Github, dan melakukan *debugging* terhadap kode yang dihasilkan.

Selama proses pengerjaan tugas besar ini, kelompok kami tentu saja menemukan berbagai kendala. Salah satunya adalah kami telat menyadari bahwa terdapat bug pada fungsi eigen vektor yang kami buat sehingga eigen vektor yang dihasilkan tidak sesuai dengan yang diinginkan. Walaupun demikian kami akan terus berusaha memberikan yang terbaik dalam setiap tugas yang kami kerjakan.

## Referensi

1. <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-Nilai-Eigen-dan-Vektor-Eigen-Bagian1.pdf>
2. [http://www.math.utah.edu/~goller/F15\\_M2270/BradyMathews\\_SVDImage.pdf](http://www.math.utah.edu/~goller/F15_M2270/BradyMathews_SVDImage.pdf)
3. <https://youtu.be/I9BBGulrOmo>
4. <https://ristohinno.medium.com/qr-decomposition-903e8c61eaab>