LAPORAN TUGAS BESAR II

“Aplikasi Nilai Eigen dan Vektor Eigen dalam Kompresi Gambar”

Laporan Ini Dibuat Untuk Memenuhi Tugas Perkuliahan

Mata Kuliah Aljabar Linier dan Geometri (IF2123)

KELAS 02

Dosen : Dr. Ir. Rinaldi Munir, MT.

DISUSUN OLEH:

Kelompok 28

“Radio”

Anggota:

Rava Naufal A (13520077)

Dimas Shidqi P (13520087)

Rio Alexander (13520088)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER I TAHUN 2021-2022

Daftar Isi

Daftar Tabel

Daftar Gambar

BAB I
Deskripsi Masalah

Membuat 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
Teori Singkat

2.1. Perkalian Matriks

Perkalian dari dua matriks akan menghasilkan suatu matriks baru dengan ukuran baris sama dengan matriks pertama dan ukuran kolom sama dengan matriks kedua. Jika A dan B adalah suatu matriks, maka hasil perkalian dari kedua matriks A dan B dapat dinyatakan sebagai C = AB.

Metode yang diterapkan di dalam rumus menghitung perkalian matriks ialah dengan memasangkan baris pada matriks pertama dengan kolom pada matriks kedua, tetapi perlu diketahui bahwa kedua nilai matriks ini dapat dikalikan jika banyak kolom pada matriks pertama mempunyai nilai yang sama dengan banyak baris pada matriks kedua dan hasil perkalian matriks akan mempunyai baris yang sama banyak dengan baris matriks pertama dan kolom yang sama banyak dengan kolom matriks kedua.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

Dapat dinyatakan sebagai, $C_{RxT} = A_{RxS} \times B_{SxT}$

$$C = C_{ij} = A_{i1}B_{1j} + \cdots + A_{iS}B_{Sj} = \sum_{k=1}^{S} A_{ik}B_{kj} ,$$

dengan S adalah banyaknya kolom matriks pertama atau banyaknya baris matriks kedua.

Sedangkan untuk penjelasan dari rumus perkalian skalar matriks dilakukan dengan cara konstanta, yang artinya nilai matriks bisa dikalikan dengan cara mengalikan setiap elemen atau komponen nilai matriks dengan skalar. Misalnya nilai Matriks A dikalikan dengan skalar K maka setiap elemen atau komponen Matriks A dikali dengan k.

$$k \times \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} ka & kb \\ kc & kd \end{pmatrix}$$

2.2. Nilai Eigen

Jika A adalah matriks n x n, maka vektor tidak-nol x di Rn disebut vektor eigen dari A jika Ax sama dengan perkalian suatu skalar λ dengan x, yaitu

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

dengan kata lain, nilai eigen menyatakan nilai karakteristik dari sebuah matriks yang berukuran n x n.

Untuk menghitung nilai eigen dari suatu matriks A dapat dihitung sebagai berikut,

$$\begin{aligned} A\mathbf{x} &= \lambda\mathbf{x} \\ IA\mathbf{x} &= \lambda I\mathbf{x} \\ A\mathbf{x} &= \lambda I\mathbf{x} \\ (\lambda I - A)\mathbf{x} &= 0 \end{aligned}$$

x = 0 adalah solusi trivial dari (λI – A)x = 0

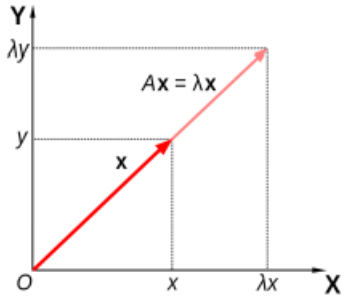
agar (λI – A)x = 0 memiliki solusi tidak-nol, maka haruslah

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

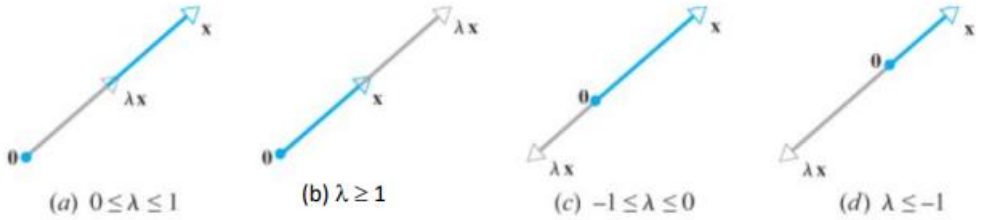
Persamaan det(λI – A) = 0 disebut persamaan karakteristik dari matriks A, dan akar-akar persamaan tersebut, yaitu λ, dinamakan akar-akar karateristik atau nilai-nilai eigen.

2.3. Vektor Eigen

Seperti yang telah dijelaskan sebelumnya, vektor eigen berkaitan dengan nilai eigen. Vektor eigen x menyatakan vektor kolom yang apabila dikalikan dengan sebuah matriks n x n menghasilkan vektor lain yang merupakan kelipatan vektor itu sendiri.



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



Dengan pemecahan solusi yang telah dijelaskan pada bagian sebelumnya, setelah didapat nilai dari λ (nilai eigen) kita dapat mensubstitusi nilai setiap λ yang ada pada persamaan (λI – A)x = 0 untuk mendapatkan vektor eigen.

Sebagai contoh, untuk

$$(\lambda I - A)x = 0 \rightarrow \begin{bmatrix} \lambda - 3 & 0 \\ -8 & \lambda + 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

didapatkan nilai-nilai eigen dari matrik A adalah λ = 3 dan λ = -1.

Untuk λ = 3,

$$\begin{bmatrix} 0 & 0 \\ -8 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow -8x_1 + 4x_2 = 0 \rightarrow x_1 = \frac{1}{2} x_2$$

sehingga didapat solusi

$$x_1 = \frac{1}{2} t, x_2 = t, t \in \boldsymbol{R}$$

maka vektor eigen

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} t \\ t \end{bmatrix} = t \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix}$$

Membentuk ruang eigen (*eigen space*).

Sehingga, $t \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix}$ adalah basis untuk ruang eigen dengan λ = 3, yang mana ruang eigen dapat ditulis sebagai berikut

$$E(3) = \{ x = t \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix},\ t \in \boldsymbol{R} \}$$

untuk λ = -1 dapat digunakan cara yang serupa untuk mendapatkan vektor eigennya.

2.4. Matriks SVD (Singular Value Decomposition)

Suatu matriks bujur sangkar A berukuran n x n dapat difaktorkan menjadi

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

E = Matriks yang mendiagonalisasi matriks A

D = Matriks diagonal yang memiliki kemiripan dengan matriks A

Dalam hal ini, E adalah matriks yang kolom-kolomnya adalah basis ruang eigen dari matriks A,

$$E = (e_1 \mid e_2 \mid \ldots \mid e_n)$$

D adalah matriks diagonal sedemikian sehingga

$$D = E^{-1}AE$$

Berbeda dengan matriks bujur sangkar, matriks bukan bujur sangkar dapat difaktorkan dengan menggunakan metode SVD (*Singular Value Decomposition*). SVD memfaktorkan matriks A berukuran m x n menjadi matriks U, ∑, dan V sedemikian sehingga

$$A = U \sum V^T$$

U = Matriks ortogonal m x m

V = Matriks ortogonal n x n

∑ = Matriks diagonal m x n dengan elemen-elemen diagonalnya adalah nilai-nilai singular dari matriks A

Matriks U adalah matriks yang kolomnya terdiri dari vektor eigen ortonormal dari matriks AA^T . 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^TA. Matriks ini menyimpan informasi yang penting terkait kolom-kolom matriks awal, dengan informasi terpenting disimpan dalam baris pertama.

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 Σ 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.

BAB III
Implementasi Program

Bahasa Pemrograman : Python, Javascript

Library : OpenCV, Numpy

Framework : 1. Frontend (Bootstrap)

2. Backend (FastAPI)

FastAPI digunakan sebagai *Backend* (server) untuk memproses gambar. Dalam proses mengirim file terdapat dua HTTP *request*, yaitu

1. POST *Request*
   - Untuk mengirim gambar dan skala kompresi yang diinginkan
   - *Return response* berupa JSON dengan atribut *time* (detik), *rate*, fileId, dan fileExt.
2. GET *Request*
   - Untuk meminta gambar yang telah dikompres
   - *Return* file gambar, *response* berupa file

Agar pengguna lain tidak dapat melakukan GET *Request* gambar yang telah dikompres, nama file diubah secara acak agar pengguna yang dapat melakukan GET *Request* hanya pengguna yang melakukan POST *Request* file tersebut. Setelah GET *Request* dilakukan, file gambar dihapus dari server agar privasi data pengguna tetap terjaga.

Penggunaan *library* OpenCV:

- Mengaplikasikan OpenCV untuk mengubah gambar menjadi matriks
- Matriks yang dihasilkan diubah menjadi matriks RGB terpisah
- Tiap matriks RGB diubah secara manual (terdapat pada algoritma SVD)
- Tiap matriks akan didekomposisi menggunakan metode SVD. Nilai singular dan vektor singunlar kanan akan didapat melalui *simultaneous power iteration*. Vektor singular kiri akan didapat dari nilai singular dan vektor singular kanan.
- Setelah didapat dekomposisi dari SVD, selanjutnya akan dilakukan kompresi matriks. Kompresi dilakukan dengan melakukan pemotongan k nilai dari matriks-matriks SVD. Nilai k didapat dari *compression rate* yang dimasukan
- Setelah kompresi, setiap matriks akan disatukan kembali menjadi gambar yang utuh

Penggunaan *library* Numpy:

- Pada awalnya, kami membuat *library* matriks dan polynom secara manual untuk meng-*handle* operasi matriks. Akan tetapi, waktu yang dibutuhkan selama pemrosesan sangatlah lama. Sehingga, diputuskan untuk menggunakan *library* numpy untuk meng-*handle* segala jenis operasi matriks
- Operasi matriks tersebut antara lain, instantiasi matriks, invers matriks, penjumlahan matriks, dll.

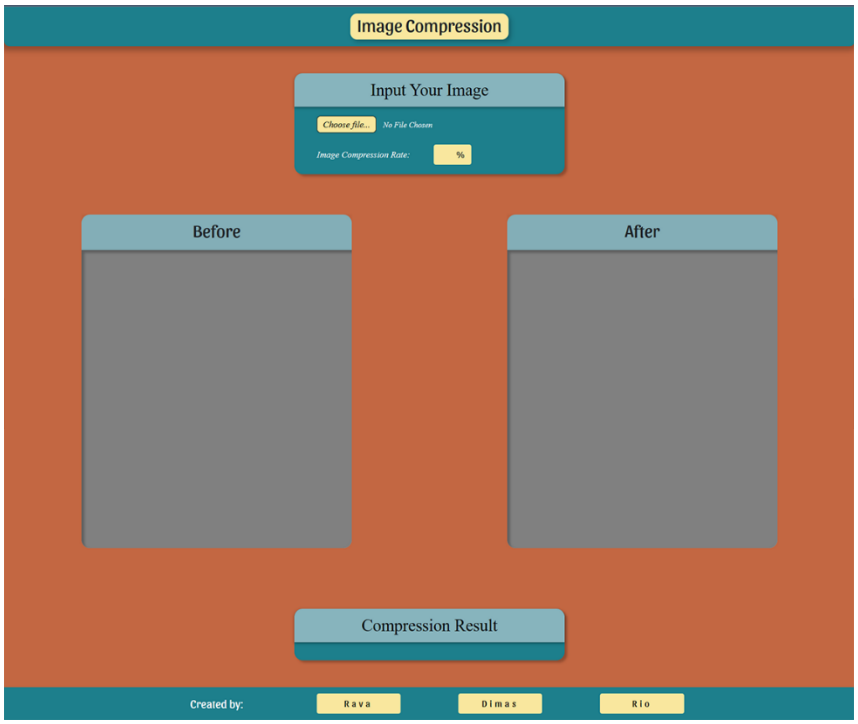
Tabel 1. Method-method yang digunakan pada program

Method	Deskripsi
`simultaneous_power_iteration(A, k, iterations)`	Mencari nilai eigen dan vektor eigen dari suatu matriks persegi A. • **A**: matriks yang ingin dicari • **k**: jumlah nilai singular • **iterations**: jumlah repetisi pengolahan matriks
`find_SVD(m, compRate, stat, decimal_places, iterations)`	Mencari dekomposisi suatu matriks m. • **m**: matriks yang ingin di dekomposisi • **compRate**: *compression rate* • **stat**: menampilkan statistik untuk keperluan *debugging* • **iterations**: jumlah repetisi pengolahan matriks
`compression(U, Zigma, Vt, compRate)`	Melakukan kompresi terhadap suatu matriks yang telah didekomposisi sebelumnya. • **U**: matriks singular kiri (U) • **Zigma**: matriks nilai singular • **Vt**: matriks singular kanan (Vt) • **compRate**: *compression rate*
`accuracy(A, ANew)`	Mencari akurasi kemiripan dua buah matriks A dan ANew dengan menghitung rerata selisih keduanya.

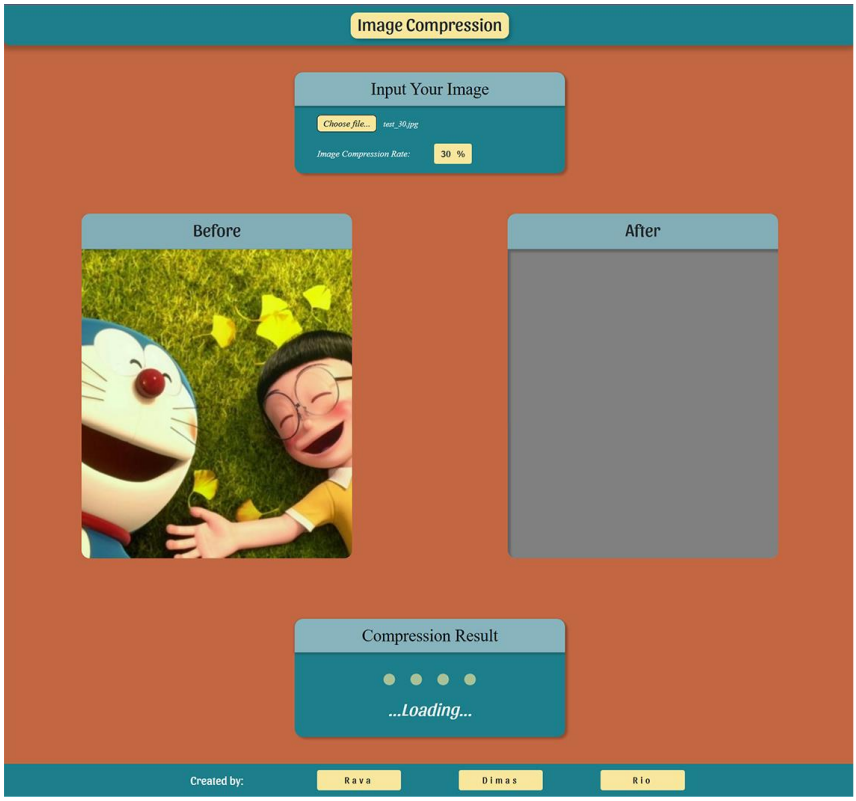
	• **A**: matriks pertama • **ANew**: matriks kedua
`compression(matrix, compRate, iterations)`	Melakukan kompresi terhadap suatu matrix. • **matrix**: matrix yang ingi dikompresi • **compRate**: *compression rate* • **Iterations**: jumlah repetisi pengolahan matriks
`compress_from_file(filePath, compRates)`	Melakukan kompresi terhadap suatu gambar dengan rate tertentu. • **filePath**: alamat direktori gambar • **compRate**: *compression rate*

BAB IV
Eksperimen

Tampilan *website* dan pada saat pemrosesan gambar sebagai berikut.



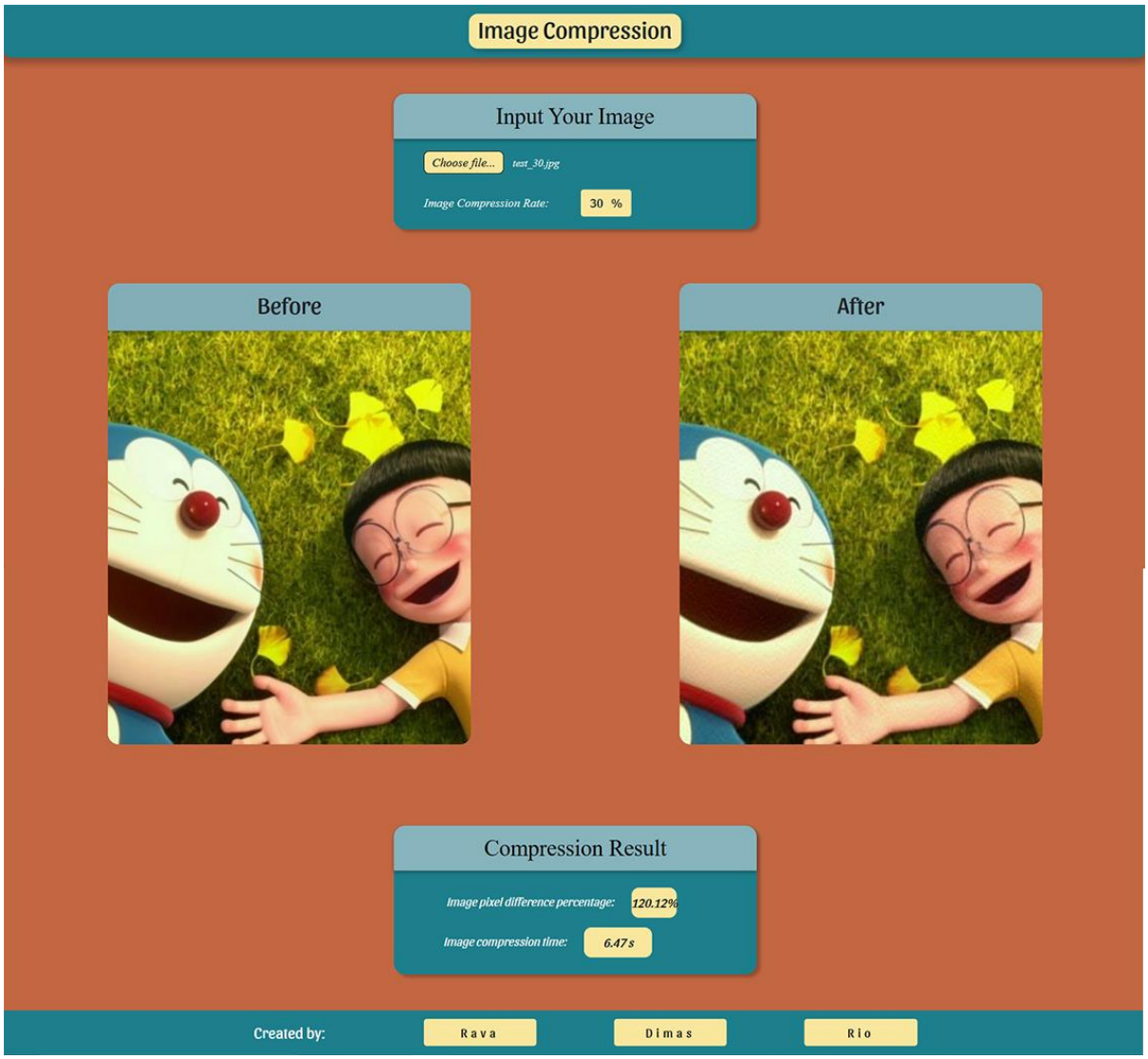
Gambar 1. Website compressing image



Gambar 2. Website saat compressing berlangsung

4.1. Eksperimen 1

Pada eksperimen pertama kami mencoba melakukan kompresi dengan file berukuran 72.9 KB dengan tingkat kompresi sebesar 30%.

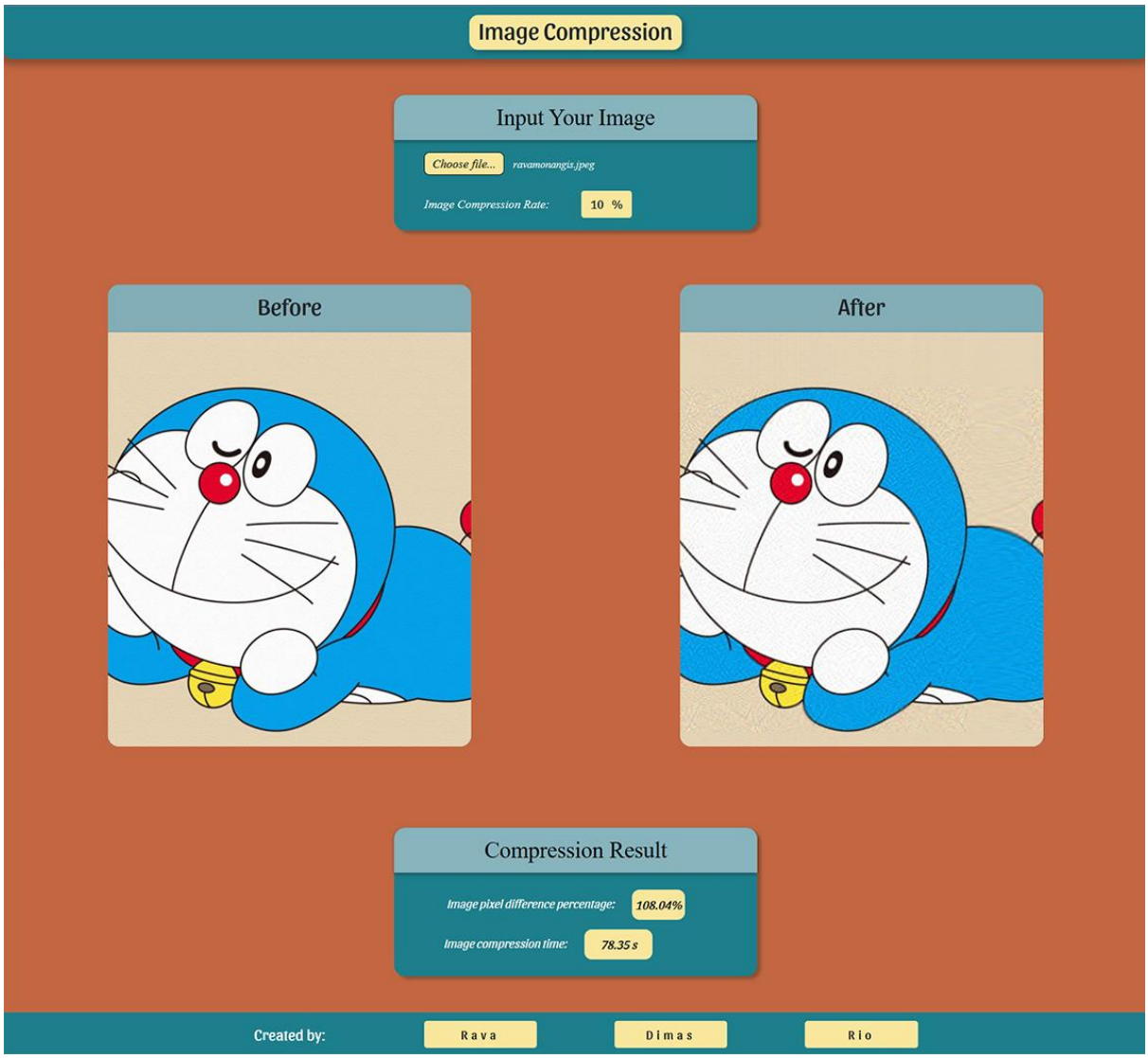


Gambar 3. Eksperimen 1

Didapatkan perubahan jumlah *pixel* gambar sebesar 120.12% dan waktu kompres gambar 6.47 detik. Untuk ukuran file awal sebesar 72.9 KB dan setelah dikompres menjadi 60.9 KB. Dari segi *pixel* tetap sama, yaitu 600 x 400 *pixels*. Pada eksperimen pertama ini dapat dilihat bahwa gambar hasil kompresi tidak begitu memperlihatkan perbedaannya dengan gambar awal.

4.2. Eksperimen 2

Pada eksperimen kedua kami mencoba melakukan kompresi dengan file berukuran 394 KB dengan tingkat kompresi sebesar 10%.

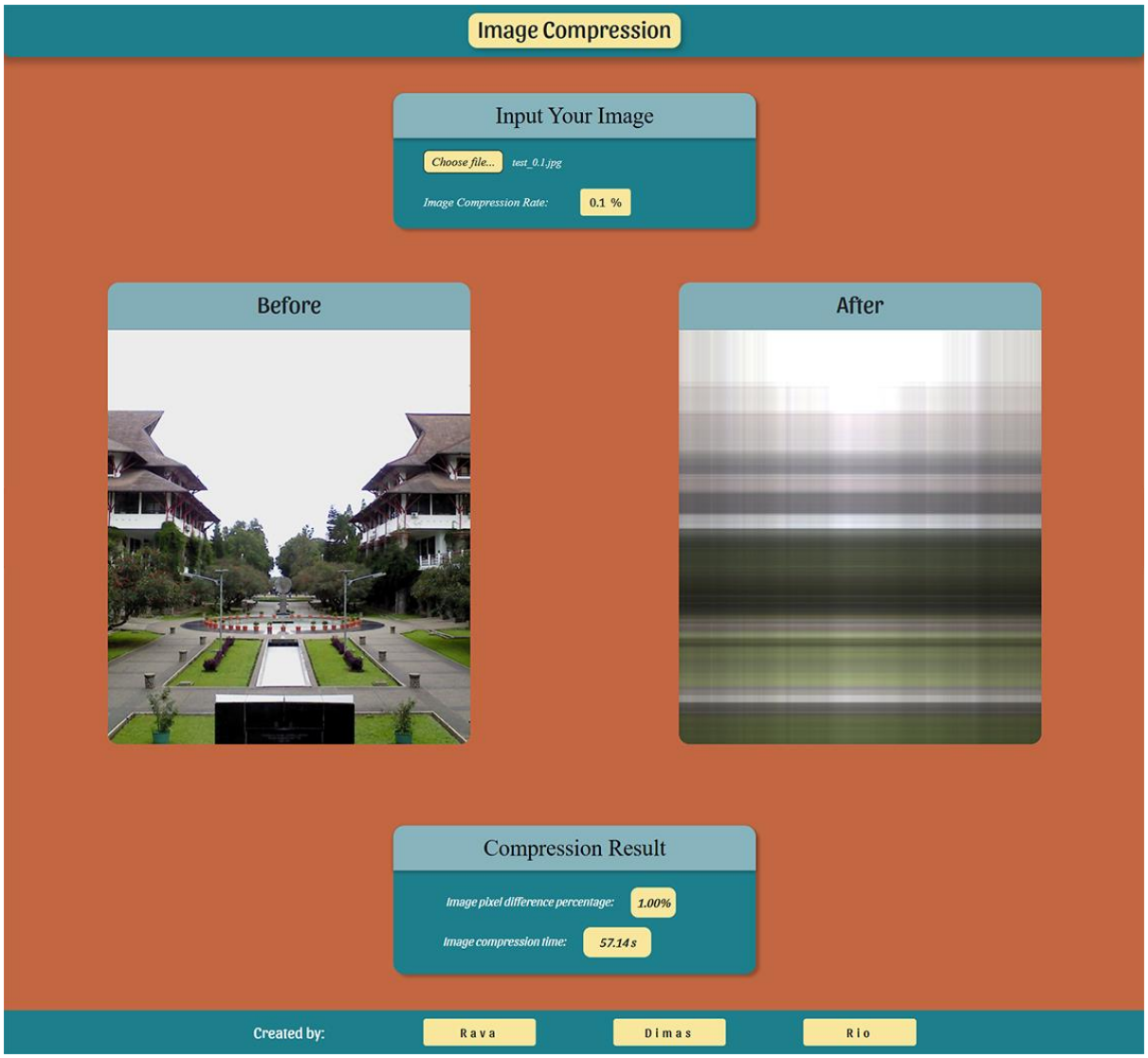


Gambar 4. Eksperimen 2

Didapatkan perubahan jumlah *pixel* gambar sebesar 108.04% dan waktu kompres gambar 78.35 detik. Untuk ukuran file awal sebesar 394 KB dan setelah dikompres menjadi 275 KB. Dari segi *pixel* tetap sama yaitu 1920 x 1080 *pixels*. Pada eksperimen kedua ini lebih terlihat efek setelah kompresi, gambar menjadi lebih *blurry*.

4.3. Eksperimen 3

Pada eksperimen ketiga kami mencoba melakukan kompresi dengan file berukuran 331 KB dengan tingkat kompresi sebesar 0.1%.



Gambar 5. Eksperimen 3

Didapatkan perubahan jumlah *pixel* gambar sebesar 1.00% dan waktu kompres gambar 57.14 detik. Untuk ukuran file awal sebesar 331 KB dan setelah dikompres menjadi 129 KB. Dari segi *pixel* tetap sama yaitu 1600 x 1200 *pixels*. Pada eksperimen ketiga ini dapat dilihat bahwasanya dengan tingkat kompresi 0.1%, gambar hasil kompresi menjadi sangat buram.

BAB V
Kesimpulan, Saran, dan Refleksi

5.1. Kesimpulan

Dari Tugas Besar IF2123 Aljabar Linier dan Geometri semester I 2021/2022 berjudul "Aplikasi Nilai Eigen dan Vektor Eigen dalam Kompresi Gambar", kami mendapati bahwa salah satu cara dalam kompresi gambar dapat dilakukan dengan metode SVD. Metode SVD tidak mengubah total pixel dari gambar, akan tetapi mampu menurunkan *size* dari file gambar tersebut bila dikompresi. Karena total matriks yang didekomposisi dengan SVD tidak sebesar matriks awal, sehingga memori yang dialokasikan-pun tidak sebesar matriks awal.

Selain itu, pada saat memproses matriks dengan SVD, ukuran file dipengaruhi oleh nilai k atau nilai *compression rate* yang dimasukkan oleh *user*, sehingga terdapat matriks yang dipotong. Pada saat menggabungkan hasil dekomposisi matriks oleh SVD menjadi gambar, *library* OpenCV memiliki kualitas bawaan untuk hasil penggabungan tersebut yang dapat membuat ukuran file menjadi lebih besar dari ukuran file *original*-nya, sehingga perlu dilakukan penyesuaian secara manual terhadap kualitas OpenCV agar ukuran file sesuai seperti file *original*.

5.2. Saran

Saran-saran yang dapat kami berikan untuk Tugas Besar IF2123 Aljabar Linier dan Geometri semester I 2021/2022 adalah:

1. Algoritma yang digunakan pada Tugas Besar kali ini masih memiliki banyak kelemahan sehingga masih dapat dikembangkan untuk dilakukan efisiensi, misalnya dengan penggunaan beberapa *library* bawaan yang telah tersedia. Oleh karena itu, dalam pengembangan program ini, masih bisa dilakukan efisiensi kinerja
2. Program ini dapat dikembangkan lebih lanjut dari segi UI/UX agar semakin *user-friendly*
3. Menimbang agar sebaiknya program ini dapat dipublikasikan setelah dikembangkan lebih lanjut. Supaya program ini memiliki kebermanfaatan yang lebih luas

5.3. Refleksi

Setelah menyelesaikan tugas besar IF2123 Aljabar Linier dan Geometri semester I 2021/2022, kami dapat merefleksikan beberapa hal, yaitu:

1. Komunikasi antar anggota kelompok berjalan dengan baik, sehingga tidak terjadi miskomunikasi atau kesalahpahaman selama pengerjaan
2. Selama pengerjaan Tugas Besar, telah dibuat beberapa *milestone* dan pembagian kerja untuk setiap anggota kelompok dengan baik
3. Selama proses pembuatan program, ketika ditemukan ketidaksesuaian saat proses eksperimen, temuan langsung dikomunikasikan kepada anggota kelompok lainnya dan bersama-sama mencari solusi
4. Perlunya untuk mempelajari *web development* agar kedepannya dapat membuat sebuah *website* yang lebih fungsional dan lebih *user-friendly* dari segi UI/UX

Daftar Referensi

https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-19b-Singular-value-decomposition.pdf

https://marksmath.org/classes/Fall2019LinearAlgebra/demos/null_space_basis.html

https://www.sparknotes.com/math/algebra2/polynomials/section4/

https://medium.com/mathadam/the-rational-root-theorem-62df4d43329c

https://numpy.org/doc/stable/index.html

http://mlwiki.org/index.php/Power_Iteration#Finding_Other_Eigenvectors