

**TUGAS BESAR 2 IF 2123**  
**ALJABAR LINIER DAN GEOMETRI**

oleh

<b>Bryan Amirul Husna</b>	<b>13520146</b>
<b>Mohamad Hilmi Rinaldi</b>	<b>13520149</b>
<b>Thirafi Najwan Kurniatama</b>	<b>13520157</b>



**PROGRAM STUDI TEKNIK INFORMATIKA**  
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**2021**

## DAFTAR ISI

DAFTAR ISI .....	1
BAB I DESKRIPSI MASALAH .....	2
1.1. Abstraksi .....	2
1.2. Penggunaan Program .....	3
1.3. Spesifikasi Tugas .....	4
BAB II DASAR TEORI .....	6
2.1. Perkalian Matriks .....	6
2.2. Nilai Eigen dan Vektor Eigen .....	6
2.3. Matriks SVD .....	7
BAB III IMPLEMENTASI .....	9
3.1. svdgolub.js .....	9
3.2. process.js .....	9
3.3. index.html dan style.css .....	10
BAB IV EKSPERIMEN .....	11
4.1. Gambar Landscape .....	11
4.1.1. High – TruePixel .....	11
4.1.2. Normal .....	11
4.1.3. Low .....	12
4.2. Gambar Portrait .....	12
4.2.1. High .....	12
4.2.2. Normal – TruePixel .....	13
4.2.3. Low .....	13
4.3. Gambar Transparan .....	14
4.3.1. High .....	14
4.3.2. Normal .....	14
4.3.3. Low – TruePixel .....	15
BAB V PENUTUPAN .....	16
5.1. Kesimpulan .....	16
5.2. Saran .....	16
5.3. Refleksi .....	16
DAFTAR PUSTAKA .....	17

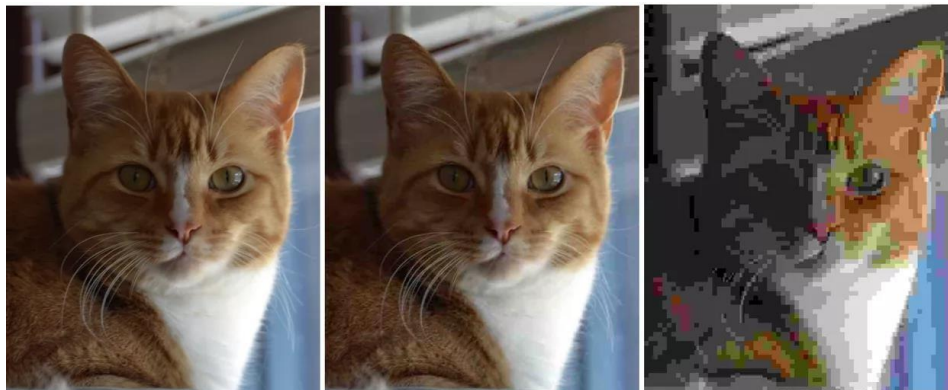
## BAB I

### DESKRIPSI MASALAH

#### 1.1. Abstraksi

Gambar adalah suatu hal yang sangat dibutuhkan pada dunia modern ini. Kita seringkali berinteraksi dengan gambar baik untuk mendapatkan informasi maupun sebagai hiburan. Gambar digital banyak sekali dipertukarkan di dunia digital melalui file-file yang mengandung gambar tersebut. Seringkali dalam transmisi dan penyimpanan gambar ditemukan masalah karena ukuran file gambar digital yang cenderung besar.

Kompresi gambar merupakan suatu tipe kompresi data yang dilakukan pada gambar digital. Dengan kompresi gambar, suatu file gambar digital dapat dikurangi ukuran filenya dengan baik tanpa mempengaruhi kualitas gambar secara signifikan. Terdapat berbagai metode dan algoritma yang digunakan untuk kompresi gambar pada zaman modern ini.



Three levels of JPG compression. The left-most image is the original. The middle image offers a medium compression, which may not be immediately obvious to the naked eye without closer inspection. The right-most image is maximally compressed.

*Gambar 1. Contoh kompresi gambar dengan berbagai tingkatan*

Sumber : [Understanding Compression in Digital Photography \(lifewire.com\)](https://lifewire.com/understanding-compression-in-digital-photography)

Salah satu algoritma yang dapat digunakan untuk kompresi gambar adalah algoritma SVD (Singular Value Decomposition). 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 2. Algoritma SVD*

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  $A^T A$ . Matriks ini menyimpan informasi yang penting terkait kolom-kolom matriks awal, dengan informasi terpenting disimpan dalam baris pertama.



Gambar 3. Ilustrasi Algoritma SVD dengan rank  $k$

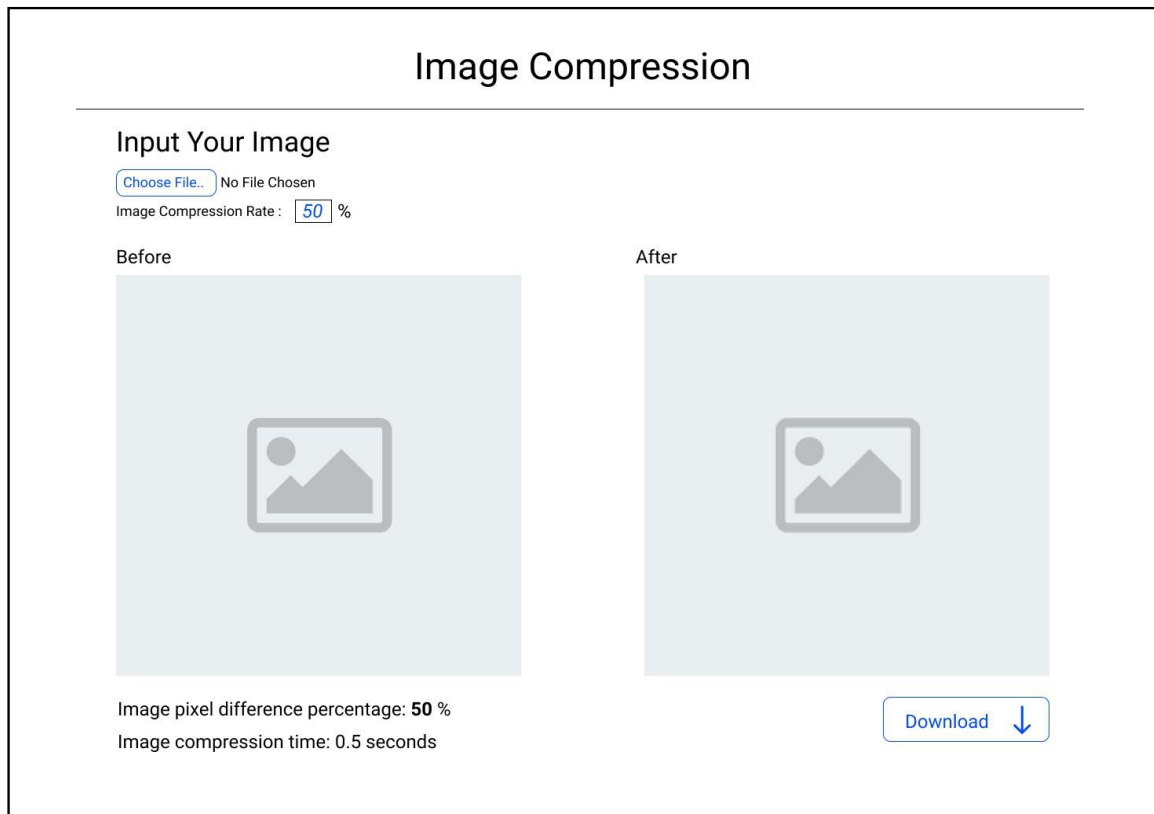
Dapat dilihat di gambar di atas bahwa dapat direkonstruksi gambar dengan banyak *singular values*  $k$  dengan mengambil kolom dan baris sebanyak  $k$  dari  $U$  dan  $V$  serta *singular value* sebanyak  $k$  dari  $S$  atau  $\Sigma$  terurut dari yang terbesar. Kita dapat mengaproksimasi suatu gambar 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.

## 1.2. Penggunaan Program

Berikut ini adalah input yang akan dimasukkan pengguna untuk eksekusi program.

1. **File gambar**, berisi *file* gambar input yang ingin dikompresi dengan format *file* yang bebas selama merupakan format untuk gambar.
2. **Tingkat kompresi**, berisi tingkat kompresi dari gambar (formatnya dibebaskan, cth: Jumlah *singular value* yang digunakan)

Tampilan *layout* dari aplikasi web yang akan dibangun kurang lebih adalah sebagai berikut. Anda dapat mengubah *layout* selama *layout* masih terdiri dari komponen yang sama.



Gambar 4. Contoh tampilan layout dari aplikasi web yang dibangun.

Catatan: Warna biru menunjukkan komponen yang dapat di klik.

Anda dapat menambahkan menu lainnya, gambar, logo, dan sebagainya. Tampilan *front end* dari *website* dibuat semenarik mungkin selama mencakup seluruh informasi pada layout yang diberikan di atas. Tampilan program merupakan bagian dari penilaian.

### 1.3. Spesifikasi 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

#### 2.1. Perkalian Matriks

Matriks merupakan suatu susunan bilangan yang berbentuk persegi atau persegi panjang yang terdiri dari baris dan kolom. Salah satu operasi dasar dari matriks ini yaitu perkalian matriks. Dalam perkalian antara dua matriks, jumlah kolom pada matriks pertama harus sama dengan jumlah baris pada matriks kedua sehingga menghasilkan matriks yang memiliki jumlah baris matriks pertama dan jumlah kolom matriks kedua.

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

Misal  $A = [a_{ij}]$  dan  $B = [b_{ij}]$

Maka  $C = A \times B$

$$= [c_{ij}]$$

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{in}b_{nj}$$

Contoh :

$$\begin{aligned} AB &= \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \times \begin{bmatrix} 5 & 2 & 1 \\ 5 & 3 & 5 \end{bmatrix} \\ &= \begin{bmatrix} 20 & 11 & 16 \\ 30 & 16 & 22 \end{bmatrix} \end{aligned}$$

Selain perkalian antara dua matriks, terdapat juga perkalian matriks dengan skalar.

Misal  $A = [a_{ij}]$  dan  $c$  adalah skalar

Maka  $cA = [ca_{ij}]$

Contoh :

$$\begin{aligned} 2A &= 2 \times \begin{bmatrix} 5 & 2 & 1 \\ 5 & 3 & 5 \end{bmatrix} \\ &= \begin{bmatrix} 10 & 4 & 2 \\ 10 & 6 & 10 \end{bmatrix} \end{aligned}$$

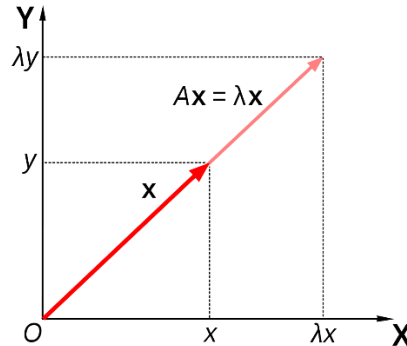
#### 2.2. Nilai Eigen dan Vektor Eigen

Kata eigen berasal dari bahasa Jerman yang memiliki arti asli atau karakteristik. Dengan begitu, nilai eigen dapat diartikan sebagai nilai karakteristik dari sebuah matriks yang berukuran  $m \times m$ .

Misalkan terdapat sebuah matriks  $C$  yang berukuran  $m \times m$  maka vektor tidak nol  $x$  di  $R^n$  disebut vektor eigen dari  $C$  jika terdapat perkalian  $Ax$  yang sama dengan perkalian suatu skalar

$\lambda$  dengan  $x$ , yaitu  $Ax = \lambda x$ . Skalar  $\lambda$  disebut nilai eigen dari  $A$ , dan  $x$  disebut dengan vektor eigen yang berhubungan dengan  $\lambda$ .

Vektor eigen  $x$  merepresentasikan vektor kolom yang apabila dikalikan dengan sebuah matriks persegi akan menghasilkan vektor lain yang merupakan kelipatan vektor itu sendiri.



Gambar 5. Ilustrasi  $Ax = \lambda x$

Untuk menentukan nilai eigen dan vektor eigen dari suatu matriks  $A$  yang berukuran  $m \times m$  yaitu dapat dihitung melalui persamaan karakter dari matriks tersebut yaitu  $\det(\lambda I - A) = 0$ . Melalui persamaan tersebut, dapat diambil akar-akar persamaan  $\lambda$  yang disebut dengan nilai-nilai eigen. Setelah mendapatkan nilai-nilai eigen, nilai-nilai tersebut disubstitusikan ke dalam persamaan  $\lambda I - A = 0$  untuk mendapatkan vektor eigennya.

Contoh :

$$A = \begin{bmatrix} 4 & 1 \\ 2 & 3 \end{bmatrix}$$

a) Nilai eigen

$$\det(\lambda I - A) = 0 \rightarrow \begin{vmatrix} 4 - \lambda & 1 \\ 2 & 3 - \lambda \end{vmatrix} = 0 \rightarrow (\lambda - 5)(\lambda - 2) = 0 \rightarrow \lambda_1 = 5 \text{ dan } \lambda_2 = 2$$

b) Vektor eigen

$$\lambda_1 = 5 \rightarrow \begin{bmatrix} -1 & 1 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow \text{solusi : } x_1 = x_2, x_2 = t, t \in \mathbb{R}$$

$$\text{Vektor eigen : } x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} t \\ t \end{bmatrix} = t \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\lambda_2 = 2 \rightarrow \begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow \text{solusi : } x_1 = -\frac{1}{2}x_2, x_2 = t, t \in \mathbb{R}$$

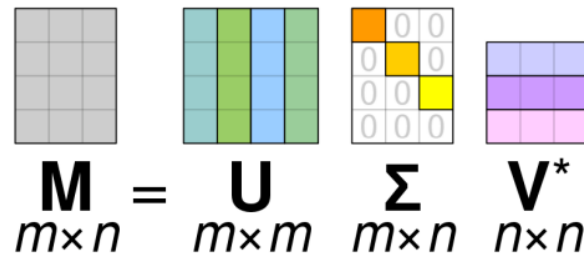
$$\text{Vektor eigen : } x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}t \\ t \end{bmatrix} = t \begin{bmatrix} -\frac{1}{2} \\ 1 \end{bmatrix}$$

### 2.3. Matriks SVD

SVD (*Singular Value Decomposition*) merupakan salah satu metode dalam mendekomposisi (memfaktorkan) suatu matriks menjadi hasil kali dari beberapa matriks lain. Dalam SVD ini sebuah matriks didekomposisi menjadi hasil perkalian dari 3 matriks yaitu matriks ortogonal  $U$ , matriks diagonal  $S$ , dan transpose dari matriks ortogonal  $V$ . Matriks ortogonal merupakan



matriks yang kolom-kolomnya adalah vektor yang saling orhogonal satu sama lain (hasil kali titiknya yaitu 0). Dekomposisi SVD ini dapat dinyatakan sebagai persamaan berikut.



$$\begin{matrix} \text{3x3 grid} & \text{3x3 grid} & \text{3x3 grid} & \text{3x3 grid} \\ \mathbf{M} & = & \mathbf{U} & \mathbf{\Sigma} & \mathbf{V}^* \\ m \times n & & m \times m & m \times n & n \times n \end{matrix}$$

Gambar 6. Dekomposisi SVD

Dari gambar di atas, matriks U merupakan matriks yang kolomnya terdiri dari vektor eigen ortonormal dari matriks  $MM^T$ . Matriks U menyimpan informasi terkait baris-baris matriks awal, dengan yang paling terpenting terdapat di dalam kolom yang pertama. Matriks  $\Sigma$  merupakan matriks diagonal yang berisi nilai singular (akar dari nilai eigen) dari matriks U atau V yang berurut mengecil. Sedangkan matriks V merupakan matriks yang kolomnya terdiri dari vektor eigen ortonormal dari matriks  $M^T M$ . Matriks V menyimpan informasi terkait kolom-kolom matriks awal, dengan yang paling terpenting terdapat di dalam baris yang pertama.

Untuk mendapatkan komponen-komponen dari SVD ini, terdapat berbagai metode. Salah satunya yang terkenal dan banyak digunakan adalah algoritma Golub-Reinsch yang efisien dalam perhitungan numerik SVD. Algoritma ini terdiri atas dua tahap, yaitu reduksi M ke bentuk bidiagonal dengan pembangunan dua rangkaian berhingga transformasi Householder dan dilanjutkan metode QR untuk mencari nilai singular matriks bidiagonal tersebut. Dua tahap tersebut akan menghasilkan *singular value decomposition* dari matriks M.

## BAB III

### IMPLEMENTASI

Dalam pembuatan website mengenai kompresi gambar sederhana, kami menggunakan bahasa Javascript dalam pembuatan algoritma SVD dan kompresi gambar. Matriks piksel direpresentasikan sebagai array dua dimensi dan pengolahan citra dilakukan dengan pustaka OpenCV JavaScript. Selain itu, kami menggunakan framework HTML, CSS, dan Bootstrap untuk tampilan dalam pembuatan website. Struktur program kami terdiri atas 4 file utama yang terdapat di folder src yaitu `svdgolub.js`, `process.js`, `style.css`, `index.html`.

#### 3.1. `svdgolub.js`

File ini berisi implementasi algoritma Golub-Reinsch dan beberapa fungsi pembantu operasi pada matriks. Karena algoritma yang digunakan hanya dapat menghitung matriks dengan jumlah baris yang lebih besar atau sama dengan jumlah kolomnya, langkah pertama implementasi adalah pengecekan kondisi tersebut dan melakukan transpose jika jumlah kolomnya melebihi jumlah baris. Selanjutnya dilakukan berbagai tahapan algoritma Golub-Reinsch, yaitu reduksi matriks ke bentuk bidiagonal, akumulasi transformasi sisi kanan, akumulasi transformasi sisi kiri, dan diagonalisasi ke bentuk bidiagonal. Setelah proses berakhir, dihasilkan objek baru yang berisi matriks U, Q (atau Sigma), dan V yang kemudian dapat digunakan untuk melakukan kompresi pada gambar. Sebagai catatan, matriks U dan V perlu ditukar jika pada langkah pertama sebelumnya dilakukan transpose karena jumlah kolom melebihi jumlah baris.

#### 3.2. `process.js`

File ini merupakan program *backend* dari website yang berisi berbagai fungsi dalam proses kompresi gambar. Pada file ini kami menggunakan library `math.js` untuk pengoperasian matriks, `gpu.js` untuk operasi perkalian matriks agar komputasinya berjalan lebih cepat, dan `opencv.js` untuk mengolah gambar.

Fungsi-fungsi yang terdapat di dalam file ini diantaranya berfungsi untuk menerima inputan gambar dari user. Lalu gambar tersebut akan dibaca dan diolah ke dalam matriks dengan pustaka OpenCV. Lalu matriks yang sudah dibaca tersebut akan dipisah menjadi 3 channel yaitu RGB channel yang masing-masing channelnya akan didekomposisi terlebih dahulu menggunakan `svdgolub` yang sudah diimplementasikan sebelumnya. Selanjutnya masing-masing komponen dari `svd` tersebut akan diolah dengan menyederhanakan baris dan kolomnya sesuai dengan jumlah singular value ( $k$ ) yang diinginkan. Matriks U akan diambil kolomnya sebesar  $k$  dan matriks V akan diambil barisnya sebesar  $k$ . Lalu masing-masing komponen tersebut akan dikalikan kembali agar menjadi sebuah matriks hasil rekonstruksi yang menyimpan informasi dari gambar yang sudah dibaca sebelumnya. Setelah masing-masing channel sudah diolah, maka channel RGB tersebut akan digabung kembali dengan pustaka OpenCV dan gambar hasil kompresi akan ditampilkan dalam website sehingga bisa didownload oleh user.

Dalam kompresi gambar, kami membuat dua opsi dalam prosesnya yang pertama yaitu kompresi biasa yang sistemnya yaitu gambar yang diinput akan *diresize* lebih kecil terlebih dahulu lalu akan diproses dengan `svd`, lalu gambar hasil kompresinya akan dikembalikan ke *size* awal. Sedangkan opsi yang kedua kami sebut dengan “True Pixel Compression” yang

sistemnya yaitu *size* dari gambar aslinya langsung diproses dengan svd sehingga kecepatan prosesnya relatif lebih lama dibandingkan yang pertama.

### **3.3. index.html dan style.css**

File ini berisikan mengenai informasi-informasi yang akan ditampilkan di halaman website. Secara garis besar, di dalam file html terdiri dari head yang berisikan mengenai title dan link ke stylesheets dari website terkait. Lalu di dalam bagian body berisikan mengenai informasi-informasi yang akan ditampilkan di website yang akan dikses. Sedangkan pada file style yang akan digunakan sebagai stylesheet pada file index.

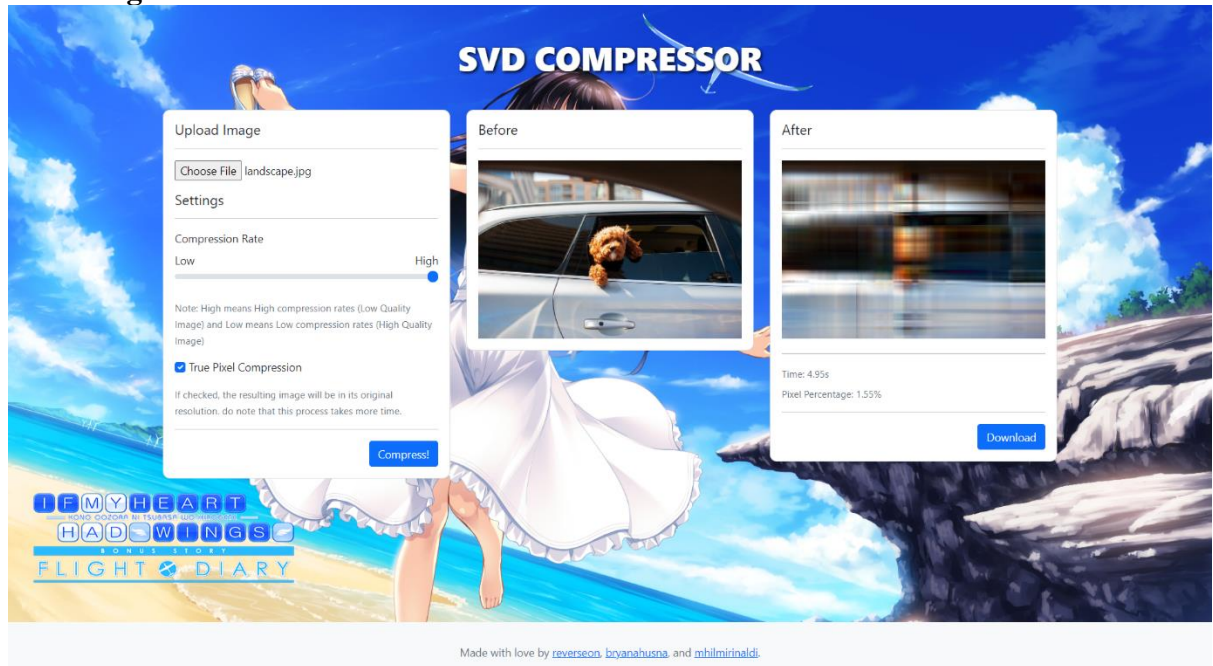
## BAB IV

### EKSPERIMEN

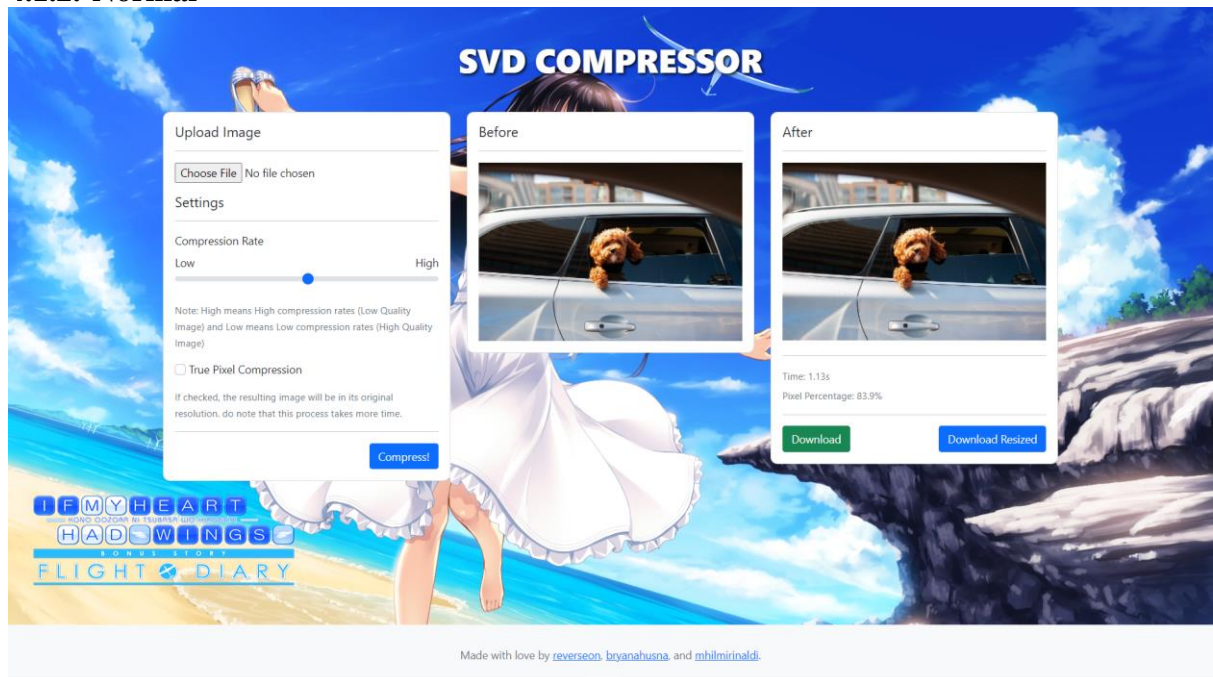
#### 4.1. Gambar Landscape

Gambar yang dites memiliki ukuran 640 x 433 *pixels* dengan format file JPEG.

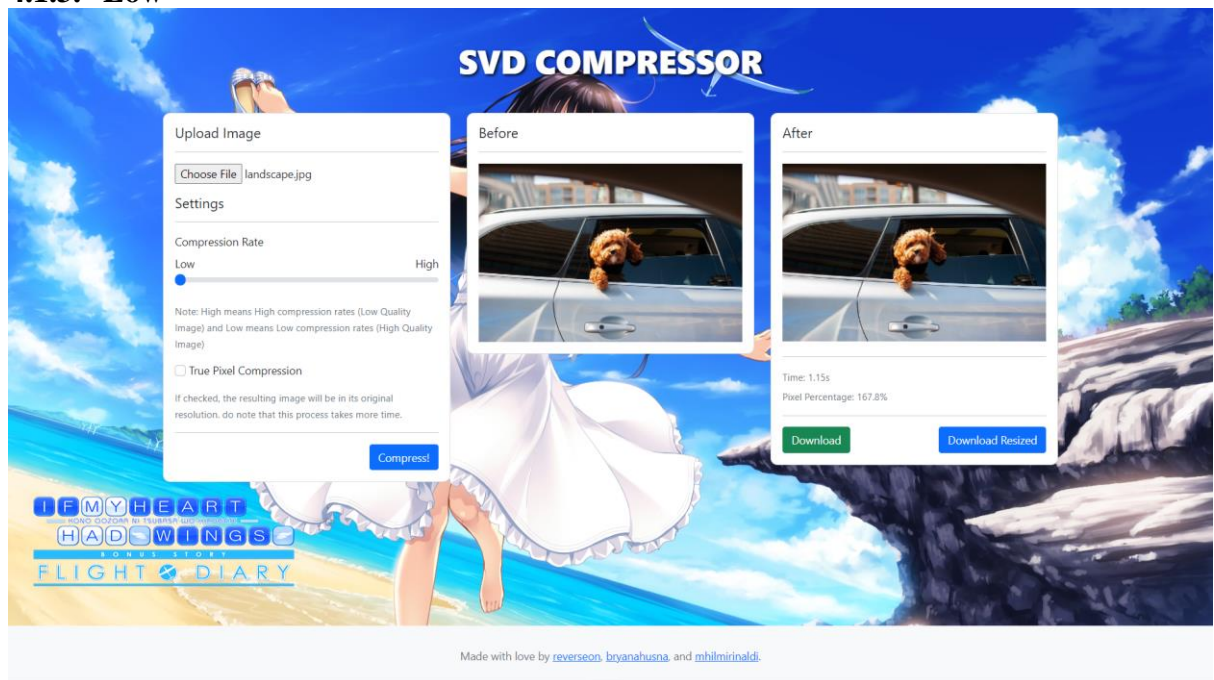
##### 4.1.1. High – TruePixel



##### 4.1.2. Normal



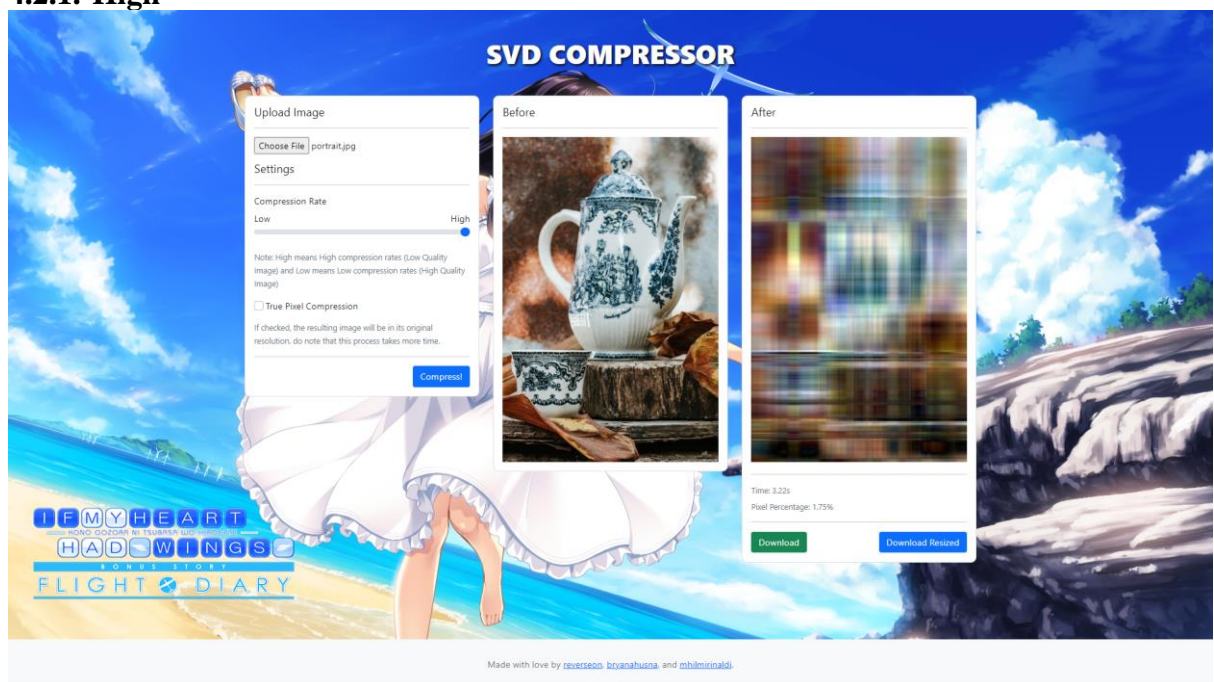
### 4.1.3. Low



## 4.2. Gambar Portrait

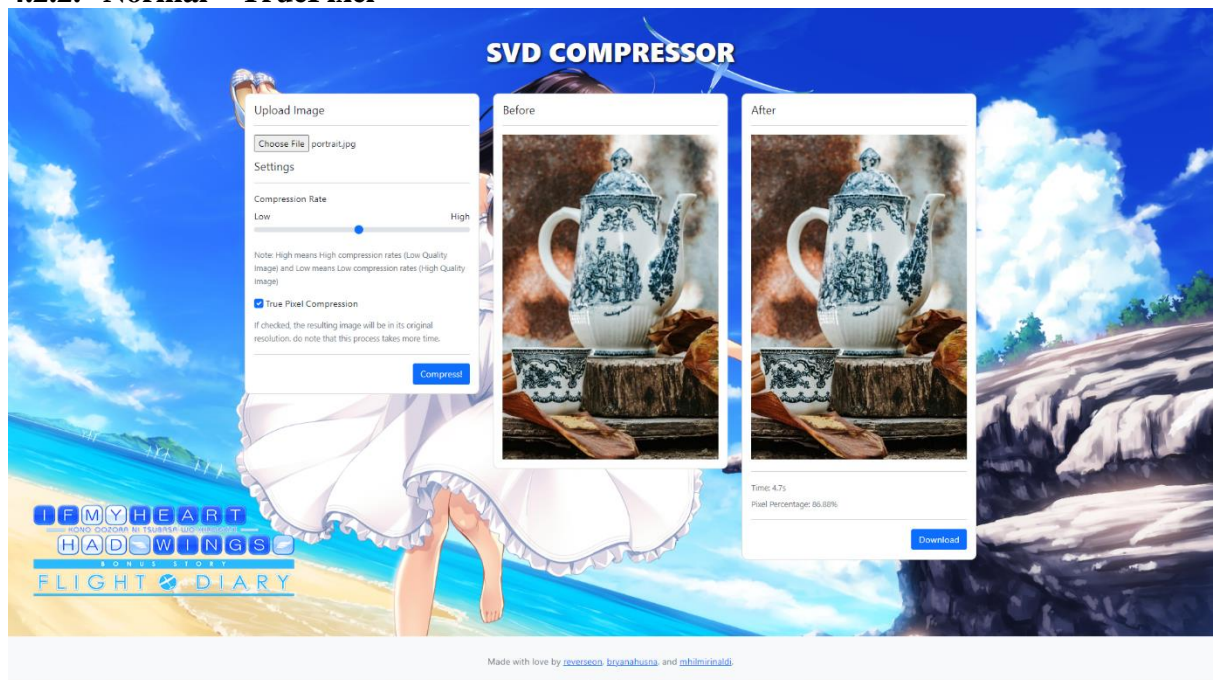
Gambar yang dites memiliki ukuran 424 x 640 *pixels* dengan format file JPEG.

### 4.2.1. High

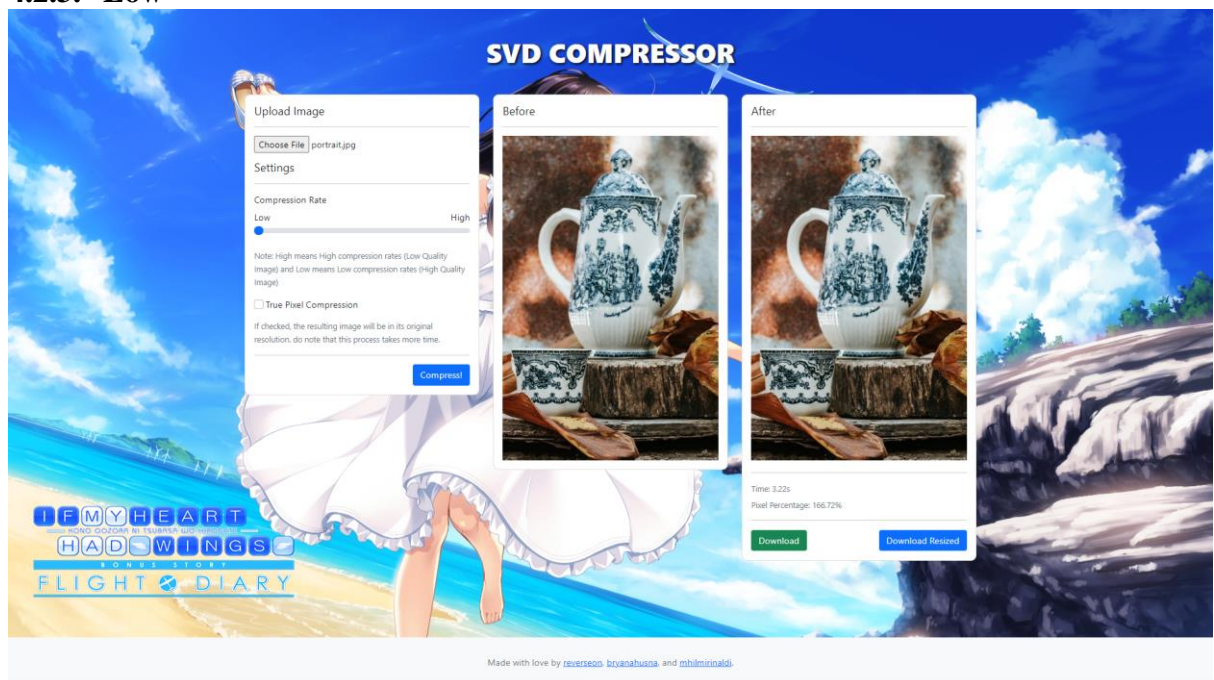




#### 4.2.2. Normal – TruePixel



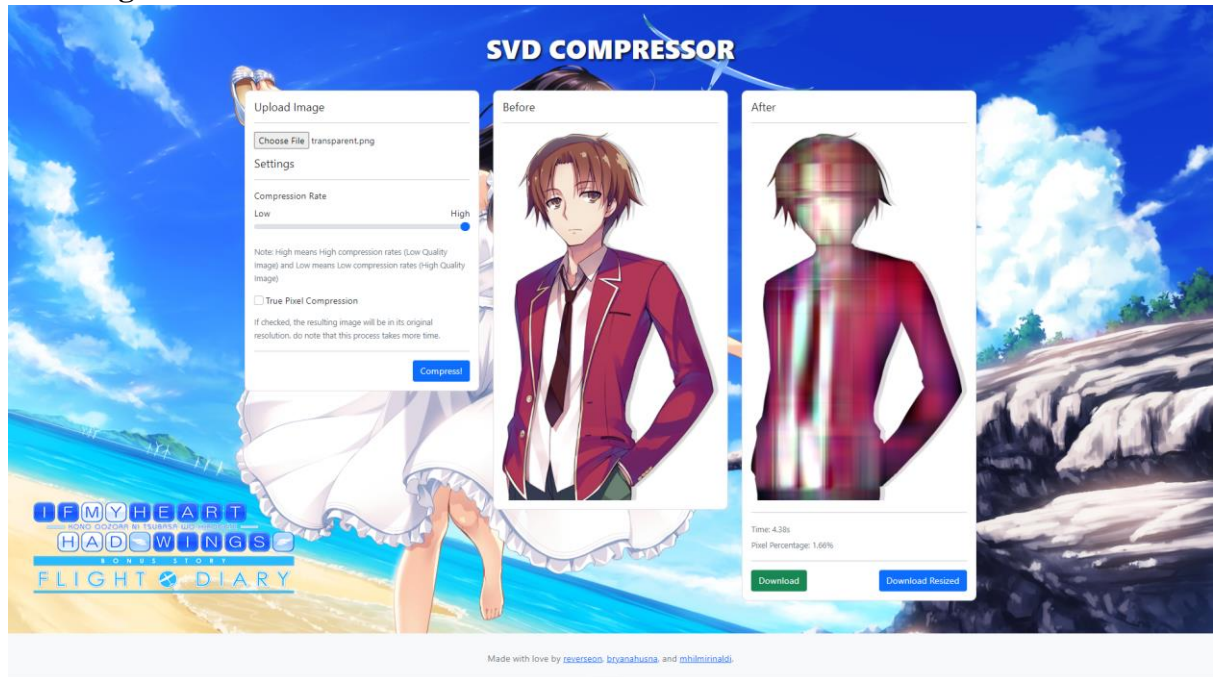
#### 4.2.3. Low



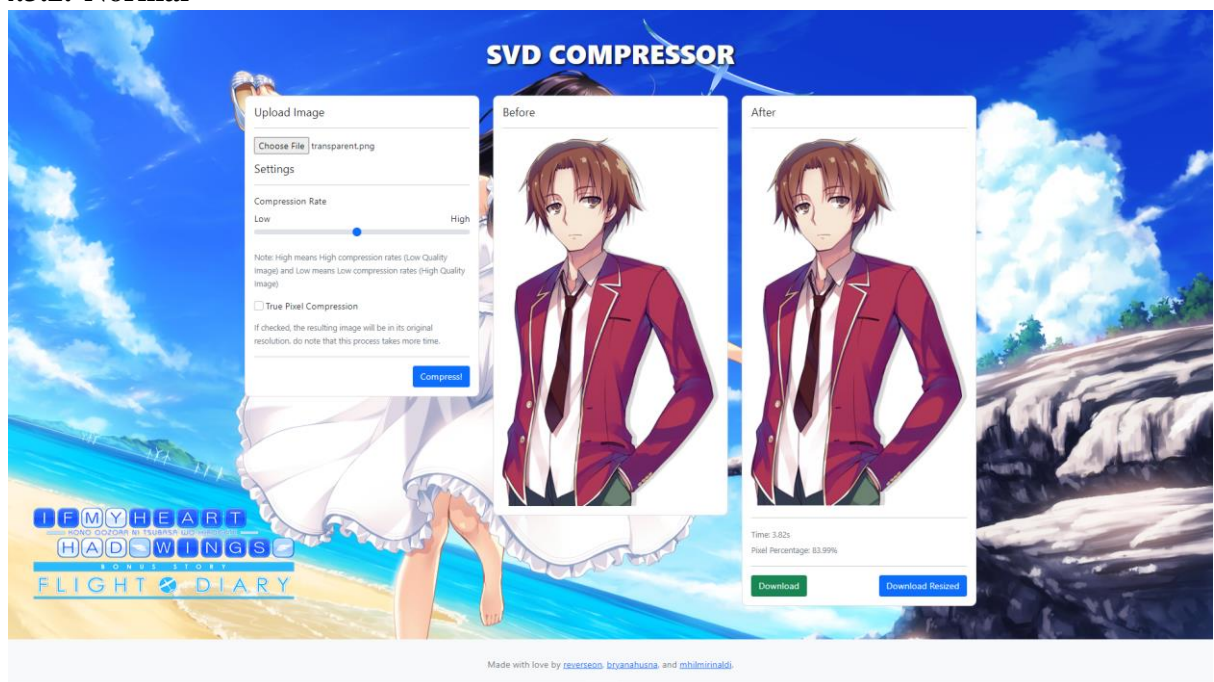
### 4.3. Gambar Transparan

Gambar yang dites memiliki ukuran 429 x 732 *pixels* dengan format file PNG.

#### 4.3.1. High

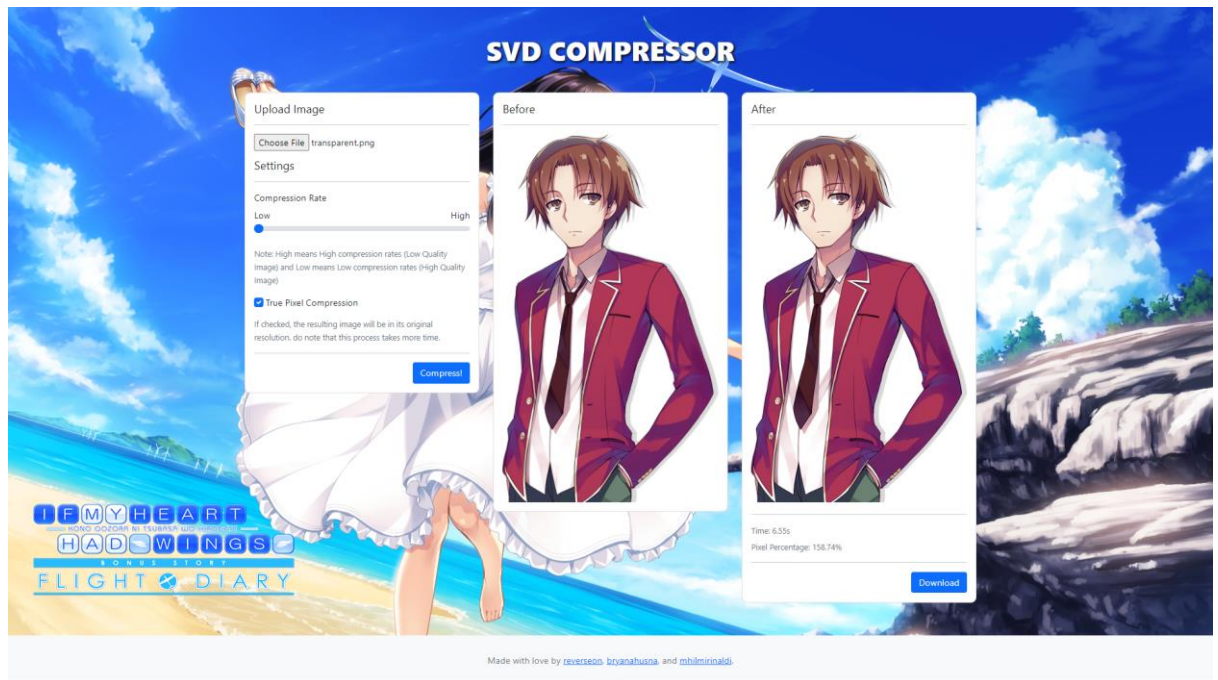


#### 4.3.2. Normal





### 4.3.3. Low – TruePixel





## BAB V

### PENUTUPAN

#### 5.1. Kesimpulan

Pada tugas kali ini, kami berhasil Membuat program *web* yang dapat melakukan kompresi pada gambar dalam berbagai format *file*. Gambar hasil yang dapat diatur tingkat kompresinya mirip dengan gambar asli dan program kami mampu mempertahankan transparansi gambar asal.

#### 5.2. Saran

Karena dekomposisi masih lama untuk berkas berukuran besar, peningkatan dapat dilakukan dengan menggunakan algoritma Golub-Reinsch yang sudah ditingkatkan dan dioptimisasi sehingga kompresi lebih cepat. Selain itu, perlu dilakukan *post processing* agar ukuran berkas yang diunduh dapat berkurang tidak hanya kualitasnya yang turun.

#### 5.3. Refleksi

Dalam tugas besar kedua Aljabar Linier dan Geometri ini, kami telah mempelajari dan memahami salah satu metode kompresi pada gambar dengan menerapkan materi yang telah dipelajari dalam kuliah, yaitu *Singular Value Decomposition*. Kami juga berkenalan dengan *web development* yang dapat menjadi *skill* tambahan dan pengantar perkuliahan web lebih lanjut.

## DAFTAR PUSTAKA

- Golub, G.H., Reinsch, C. Singular value decomposition and least squares solutions. *Numer. Math.* 14, 403–420 (1970). <https://doi.org/10.1007/BF02163027>
- Tony F. Chan. 1982. An Improved Algorithm for Computing the Singular Value Decomposition. *ACM Trans. Math. Softw.* 8, 1 (March 1982), 72–83. <https://doi.org/10.1145/355984.355990>
- Munir, Rinaldi. 2020. “Nilai Eigen dan Vektor Eigen Bagian 1” <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-Nilai-Eigen-dan-Vektor-Eigen-Bagian1.pdf>
- Munir, Rinaldi. 2021. “Singular Value Decomposition (SVD)” <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-19b-Singular-value-decomposition.pdf>