

**TUGAS BESAR 2 IF 2123**  
**ALJABAR LINIER DAN GEOMETRI**  
**APLIKASI NILAI EIGEN DAN VEKTORI EIGEN DALAM**  
**KOMPRESI GAMBAR**



Disusun Oleh :

Kelompok 29

13520028 Timothy Stanley Setiawan

13520054 Farrel Farandieka Fibriyanto

13520082 Jeremy Rionaldo Pasaribu

PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT  
TEKNOLOGI BANDUNG  
Semester I Tahun 2021/2022

## Daftar Isi

<b>Daftar Isi .....</b>	<b>1</b>
<b>BAB 1: Deskripsi Masalah .....</b>	<b>2</b>
<b>BAB 2: Teori Dasar .....</b>	<b>6</b>
<b>BAB 3: Implementasi Library dan Program .....</b>	<b>10</b>
<b>BAB 4: Eksperimen .....</b>	<b>21</b>
<b>BAB 5: Kesimpulan, Saran, dan Refleksi .....</b>	<b>28</b>
<b>BAB 6: Pembagian Tugas .....</b>	<b>30</b>
<b>Referensi .....</b>	<b>31</b>

## BAB 1: Deskripsi Masalah

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*

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 1. 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  $A^TA$ . 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  $k$*

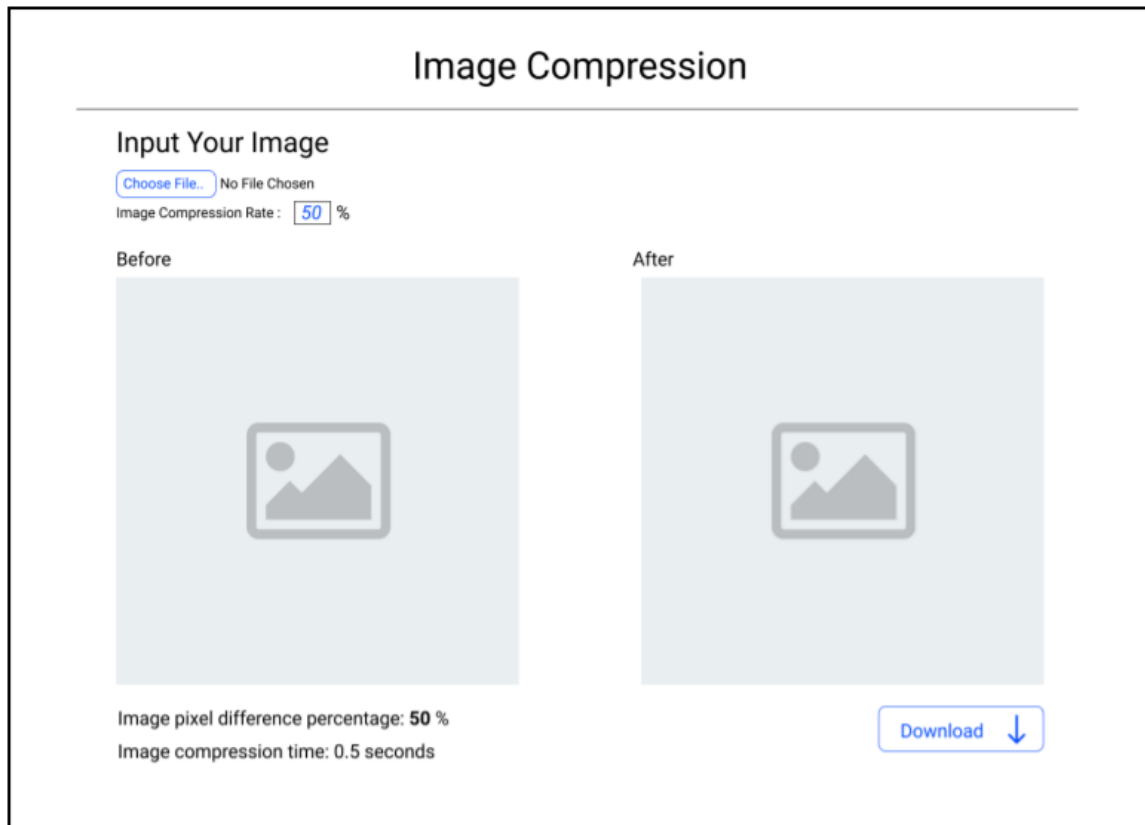
Dapat dilihat di gambar di atas bahwa dapat direkonstruksi gambar dengan banyak singular values  $k$  dengan mengambil kolom dan baris sebanyak  $k$  dari  $U$  dan  $V$  serta singular value sebanyak  $k$  dari  $S$  atau  $\Sigma$  terurut dari yang terbesar. Kita dapat mengaproksimasi suatu gambar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

Pada kesempatan kali ini, kalian mendapatkan tantangan untuk membuat website kompresi gambar sederhana dengan menggunakan algoritma SVD.

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.



*Gambar 3. Contoh tampilan layout dari aplikasi web yang dibangun.*

Anda dapat mengubah layout selama layout masih terdiri dari komponen yang sama. 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.

Membuatlah 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 2: Teori Dasar

### 2.1. Singular Value Decomposition (SVD)

Di dalam materi nilai eigen dan vektor eigen, pokok bahasan diagonalisasi, matriks bujursangkar  $A$  berukuran  $n \times n$  dapat difaktorkan menjadi :

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

dalam hal ini,

$E$  adalah matriks yang kolom-kolomnya adalah basis ruang eigen dari matriks  $A$ ,

$$E = (e_1 | e_2 | \dots | e_n)$$

$D$  adalah matriks diagonal sedemikian sehingga

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

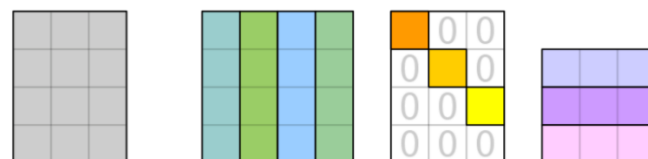
Namun, muncul permasalahan ketika memfaktorkan matrix non-bujursangkar berukuran  $m \times n$  yang tidak memiliki nilai eigen. Oleh sebab itu untuk matriks non-bujursangkar, pemfaktorrannya menggunakan metode singular decomposition value (SVD). SVD memfaktorkan matriks  $A$  berukuran  $m \times n$  menjadi matriks  $U$ ,  $\Sigma$ , dan  $V$  sedemikian sehingga

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

$U$  = matriks ortogonal  $m \times m$ ,

$V$  = matriks orthogonal  $n \times n$

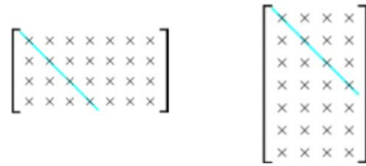
$\Sigma$  = matriks berukuran  $m \times n$  yang elemen-elemen diagonal utamanya adalah nilai-nilai singular dari matriks  $A$  dan elemen-elemen lainnya 0



$$\begin{matrix} \text{4x4 grid} & & \text{4x4 grid} & & \text{4x4 grid} & & \text{4x4 grid} \\ \mathbf{M} & = & \mathbf{U} & & \mathbf{\Sigma} & & \mathbf{V}^* \\ m \times n & & m \times m & & m \times n & & n \times n \end{matrix}$$

#### 2.1.1. Diagonal utama matriks $m \times n$

Diagonal utama sebuah matriks biasanya didefinisikan pada matriks persegi (matriks bujursangkar) berukuran  $n \times n$ . Untuk matriks bukan bujursangkar, yaitu matriks  $m \times n$ , diagonal utama matriks didefinisikan pada garis yang di mulai dari sudut kiri atas terus ke bawah matriks sejauh mungkin.



### 2.1.2. Matriks ortogonal

Matriks ortogonal adalah matriks yang kolom-kolomnya adalah vektor yang saling orthogonal satu sama lain (hasil kali titik sama dengan 0). Jika vektor-vektor kolom tersebut merupakan vektor satuan, maka matriks ortogonal tersebut dinamakan juga matriks ortonormal. Vektor satuan adalah vektor yang dinormalisasi dengan panjang atau magnitude-nya sehingga memiliki panjang atau magnitude = 1. Jika  $Q$  adalah matriks ortogonal  $m \times n$ , dan kolom-kolom matriks  $Q$  adalah vektor-vektor satuan  $v_1, v_2, \dots, v_m$ , maka  $v_i \cdot v_j = 0$  untuk  $i \neq j$ . Atau, dapat juga dikatakan bahwa  $Q$  adalah matriks ortogonal jika  $Q^T Q = I$ , dalam hal ini  $I$  adalah matriks identitas.

column vectors  $v_1, v_2, \dots, v_n$  of  $Q$  are orthogonal :

$$v_i^T v_j = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

The orthogonal matrix  $Q = [v_1, v_2, \dots, v_n]$

$$Q^T \cdot Q = \begin{bmatrix} v_1^T \\ v_2^T \\ \dots \\ v_n^T \end{bmatrix} \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix} = \begin{bmatrix} v_1^T v_1 & v_1^T v_2 & \dots & v_1^T v_n \\ v_2^T v_1 & v_2^T v_2 & \dots & v_2^T v_n \\ \dots & \dots & \dots & \dots \\ v_n^T v_1 & v_n^T v_2 & \dots & v_n^T v_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{bmatrix} = I$$

$$\therefore Q^{-1} = Q^T$$

### 2.1.3. Nilai-nilai singular matriks

Misalkan  $A$  adalah matriks  $m \times n$ . Jika  $\lambda_1, \lambda_2, \dots, \lambda_n$  adalah nilai-nilai eigen dari  $A^T A$ , maka  $\sigma_1 = \sqrt{\lambda_1}, \sigma_2 = \sqrt{\lambda_2}, \dots, \sigma_n = \sqrt{\lambda_n}$  disebut nilai-nilai singular dari matriks  $A$ . Diasumsikan  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$  sehingga  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$

#### Teorema

If  $A$  is an  $m \times n$  matrix, then:

(a)  $A^T A$  is orthogonally diagonalizable.

(b) The eigenvalues of  $A^T A$  are nonnegative.



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

Vektor-vektor singular kiri      Nilai-nilai singular      Vektor-vektor singular kanan

2.1.4. Langkah-langkah SVD mendekomposisi  $A_{m \times n}$  menjadi  $U$ ,  $\Sigma$ , dan  $V$ :

- Untuk vektor singular kiri, hitung nilai-nilai eigen dari  $AA^T$ .  $\text{Rank}(A) = k = \text{banyaknya nilai-nilai eigen tidak nol dari } AA^T$ .
- Tentukan vektor-vektor eigen  $u_1, u_2, \dots, u_m$  yang berkoresponden dengan nilai-nilai eigen dari  $AA^T$ . Normalisasi  $u_1, u_2, \dots, u_m$  dengan cara setiap komponen vektornya dibagi dengan panjang vektor. Diperoleh matriks  $U$ .
- Untuk vektor singular kanan, hitung nilai-nilai eigen dari  $A^T A$  lalu tentukan nilai-nilai singularnya.
- Tentukan vektor-vektor eigen  $v_1, v_2, \dots, v_n$  yang berkoresponden dengan nilai-nilai eigen dari  $A^T A$ . Normalisasi  $v_1, v_2, \dots, v_n$  dengan cara setiap komponen vektornya dibagi dengan panjang vektor. Diperoleh matriks  $V$ . Transpose-kan matriks  $V$  sehingga menjadi  $V^T$ .
- Bentuklah matriks  $\Sigma$  berukuran  $m \times n$  dengan elemen-elemen diagonalnya adalah nilai-nilai singular dari matriks  $A$  dengan susunan dari besar ke kecil. Nilai singular di dalam  $\Sigma$  adalah akar pangkat dua dari nilai-nilai eigen yang tidak nol dari  $A^T A$ .
- Maka,  $A = U \Sigma V^T$

## 2.2. Power Iteration

Untuk mendapatkan nilai dan vektor eigen kita dapat memanfaatkan metode Power iteration. Power iteration adalah metode aljabar linier untuk mendekati nilai eigen dan vektor eigen dari sebuah matrix. Berikut ini merupakan implementasi untuk mencari nilai dan vektor eigen menggunakan metode power iteration:

### 2.2.1. Orthogonal Iteration

Seperti yang kita tahu, vektor eigen itu memiliki sifat yang saling orthogonal sehingga kita dapat menggunakan power method dan membuat vektor eigen kedua orthogonal dengan vektor eigen pertama. Dengan demikian, kita dapat membuat dua vektor eigen berbeda

yang saling orthogonal. Hal ini tidak hanya berlaku untuk dua vektor, tetapi berlaku juga untuk banyak vektor eigen. Inilah yang disebut dengan orthogonal iteration. Orthogonal iteration adalah versi blok dari power iteration yang biasa disebut juga sebagai “Simultaneous Power”. Dalam versi ini, nilai  $A$  dikalikan tidak hanya pada satu vektor, tetapi pada banyak vektor sekaligus yang nantinya akan disimpan di suatu matrix. Setiap kali melakukan iterasi kita harus menormalisasi vektornya. Salah satu caranya dengan memanfaatkan algoritma QR.

## BAB 3: Implementasi Library dan Program

### 3.1. Implementasi untuk kompresi gambar (`imgcompression.py`)

#### 3.1.1. Fungsi `CompressChannel`

```
def CompressChannel(channelMatrix, k, eigen_total):  
    """  
    Menghasilkan matriks hasil kompresi menggunakan algoritma SVD.  
    """  
    U, S, VT = SVD(channelMatrix, eigen_total)  
    A = U[:, 0:k] @ S[0:k, 0:k] @ VT[0:k, :]  
    compressed = np.clip(A, 0, 255).astype('uint8')  
    return compressed
```

*Gambar 4. Fungsi `CompressChannel`*

Fungsi `CompressChannel` mengembalikan hasil matriks *channel* gambar yang sudah dilakukan kompresi menggunakan algoritma SVD. Fungsi ini memiliki parameter `k` sebagai tingkat kompresi gambar, parameter `channelMatrix` sebagai matriks *channel* gambar yang ingin dikompresi, serta parameter `eigen_total` sebagai jumlah total nilai eigen pada matriks *channel*. Hasil komponen SVD ( $U, S, V^T$ ) digabungkan kembali menjadi matriks *channel* gambar sesuai dengan nilai parameter `k`. Matriks tersebut kemudian dilakukan pemotongan nilai sehingga menghasilkan matriks yang elemennya hanya boleh memiliki nilai integer dengan range 0-255.

#### 3.1.2. Fungsi `SVD`

```
def SVD(matrix, k):  
    """  
    Menghasilkan komponen SVD dengan menggunakan konsep power iteration serta  
    konsep perkalian matriks invers.  
    """  
    # KAMUS LOKAL  
    baris, kolom = matrix.shape # baris dan kolom matriks  
  
    # ALGORITMA  
    if(baris <= kolom):  
        A = matrix @ matrix.T  
        n = baris  
    elif(baris > kolom):  
        A = matrix.T @ matrix  
        n = kolom
```

```
# Algoritma QR
Q = np.random.rand(n, k)
Q, _ = np.linalg.qr(Q)
# Skema power iteration
for i in range(150):
    Z = A.dot(Q)
    Q, R = np.linalg.qr(Z)
# Semua nilai singular values
diag = np.sqrt(np.abs(np.diag(R)))
# Mengubah nilai singular values agar memiliki invers
for i in range(diag.shape[0]):
    if diag[i] < 1:
        diag[i] = 0.1
S = np.diag(diag)

# Skema perkalian matriks invers untuk mendapatkan semua komponen
if(baris <= kolom):
    U = Q
    US = U @ S
    inv = np.linalg.inv(US)
    VT = inv @ matrix
elif(baris > kolom):
    VT = Q.T
    SVT = S @ VT
    inv = np.linalg.inv(SVT)
    U = matrix @ inv
return U, S, VT
```

Gambar 5. Fungsi SVD

Fungsi SVD mengembalikan komponen hasil SVD yaitu matriks  $U$ ,  $S$ , dan  $V^T$ . Fungsi ini memiliki parameter matrix untuk *input* matriks channel serta parameter  $k$  sebagai jumlah total nilai eigen. Fungsi ini menghasilkan komponen  $U$ ,  $S$ , dan  $V^T$  menggunakan konsep *power iteration* berupa *orthogonal iteration* serta konsep perkalian matriks invers. Algoritma *orthogonal iteration* dibantu dengan menggunakan algoritma QR untuk mencari matriks  $U/V$ . Jika kolom matriks lebih besar daripada baris matriks, matriks  $U$  dan  $S$  akan dicari terlebih dahulu dengan *orthogonal iteration* lalu komponen  $V^T$  akan dicari menggunakan perkalian invers. Jika baris matriks lebih besar daripada kolom matriks, matriks  $V$  dan  $S$  akan dicari terlebih dahulu dengan *orthogonal iteration* lalu komponen  $U$  akan dicari menggunakan perkalian invers.

### 3.1.3. Program Utama

```
# PROGRAM UTAMA
def mainCompress():
    # ALGORITMA
    file = str(input())
    ratio = float(input())
    sys_path = sys.path[0]
    load_path = "../test/testgambar/" + file
    path = os.path.join(sys_path, load_path)
    pic = Image.open(path)

    # Mengecek jenis gambar
    if pic.mode == "RGBA":
        alpha = pic.getchannel('A')
    elif len(pic.getbands()) == 1:
        pic = pic.convert("RGBA")
        alpha = pic.getchannel('A')

    # Ukuran width dan height gambar
    pixel = np.array(pic)
    width = pixel.shape[0]
    height = pixel.shape[1]
    # Tingkat kompresi gambar
    total_k = min(height, width)
    new_ratio = 100 - ratio
    k = round(total_k * new_ratio / 100)

    # Konversi channel gambar menjadi matriks
    red = np.asarray(pic.getchannel('R')).astype(float)
    green = np.asarray(pic.getchannel('G')).astype(float)
    blue = np.asarray(pic.getchannel('B')).astype(float)
    # Kompresi matriks channel
    redCompressed = CompressChannel(red, k, total_k)
    greenCompressed = CompressChannel(green, k, total_k)
    blueCompressed = CompressChannel(blue, k, total_k)
    # Konversi matriks menjadi channel gambar
    redImage = Image.fromarray(redCompressed)
    blueImage = Image.fromarray(blueCompressed)
    greenImage = Image.fromarray(greenCompressed)

    # Membuat gambar berdasarkan format
    if alpha:
        newImage = Image.merge(
            "RGBA", (redImage, greenImage, blueImage, alpha))
    else:
        newImage = Image.merge("RGB", (redImage, greenImage, blueImage))
    save_path = "../test/testgambar/" + "compressed_" + file
    path = os.path.join(sys_path, save_path)
    newImage.save(path)
```

*Gambar 6. Program Utama*

Program utama akan membaca gambar yang ingin dilakukan kompresi serta rasio kompresi yang diinginkan. Gambar tersebut kemudian dikonversi menjadi sebuah matriks berdasarkan channel gambar yang sesuai dengan formatnya. Program utama akan menghitung nilai k sebagai tingkat kompresi dari hasil rasio yang diinginkan. Metode kompresi yang digunakan akan menghilangkan beberapa persen dari gambar sesuai dengan rasio yang diinginkan. Selanjutnya matriks dilakukan kompresi menggunakan fungsi `CompressChannel` sesuai

dengan tingkat kompresi lalu dikonversi kembali menjadi beberapa channel gambar. Channel gambar tersebut digabungkan menjadi sebuah gambar berdasarkan format gambar sebelumnya. Gambar tersebut kemudian disimpan pada path folder yang telah disediakan.

### 3.2. Implementasi Front End dan Back End Website

#### 3.2.1 Program pada sisi peladen (api.py)

Sisi peladen pada program yang dibuat menggunakan bahasa pemrograman Python untuk menjalankan program.

##### 1. Mengimpor Library

```
import os
from flask import Flask, request, session, send_from_directory
from werkzeug.utils import secure_filename
import logging
import imgcompression as compress
```

*Gambar 7. Mengimpor library*

Baris pertama mengimpor os, os akan dipakai untuk menggabungkan *path* untuk menyimpan gambar yang akan dikompresi, dan mengecek apakah gambar tersebut sudah ada di folder yang ditetapkan. Baris kedua akan mengimpor Flask, request, session, dan send\_from\_directory. Flask akan digunakan untuk menginisialisasi kerangka situs berbahasa Python, request akan digunakan untuk memperoleh data yang dikirim oleh browser, session akan dipakai untuk mengatur *session data* (interval *browser* masuk dan keluar dari sesi), dan send\_from\_directory akan digunakan untuk mengirimkan file berupa gambar yang sudah dikompres. Baris ketiga akan mengimpor fungsi secure\_filename yang akan mengamankan nama file yang akan disimpan (contohnya dari “./src/index” ke “src\_index”). Baris keempat merupakan fungsi print untuk konsol yang sedang menjalankan api.py, kami pakai untuk mengecek apakah api ada masalah di kode bagian tertentu. Dan yang terakhir baris kelima akan mengimpor modul kompresi gambar yang sudah diterapkan dalam bahasa Python.

## 2. Inisialisasi API

```
app = Flask(__name__)
UPLOAD_FOLDER = '../public/user_uploaded/'
DOWNLOAD_FOLDER = '../public/user_uploaded/hasil/'
ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg'])
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['DOWNLOAD_FOLDER'] = DOWNLOAD_FOLDER
SESSION_TYPE = "redis"
PERMANENT_SESSION_LIFETIME = 1800
app.config.update(SECRET_KEY=b'BenciTubes2104819')
```

*Gambar 8. Inisialisasi API*

Pada awalnya, kami akan menginisialisasi fungsi Flask. Akan ditetapkan juga path untuk men-*download* dan meng-*upload* file (dilihat dari sisi user). Baris berikutnya akan ditetapkan ekstensi file apa saja yang diperbolehkan untuk di-*upload* ke *server*, di sini kami menetapkan ekstensi png, jpg, dan jpeg. Berikutnya ditetapkan tipe sesi dan maksimum waktu sesi permanen. Akan ditetapkan juga secret key yang akan membuat sesi klien aman, karena kliennya hanya user sendiri karena kami hanya mengimplementasikan di level localhost saja, kami tetapkan secret key yang statis.

## 3. Upload & Compress

Bagian ini digunakan untuk meng-*upload* dan men-*compress* gambar yang sudah disediakan oleh user dari halaman depan situs. User akan menetapkan rasio kompresi dan file yang akan dikompresi dan disimpan dalam variable file dan ratio, file tersebut kemudian akan disimpan di *path* target. Kemudian akan lanjut ke pemanggilan fungsi kompresi yang akan diberikan input nama file dan rasio yang sudah diberikan user. Setelah pemanggilan fungsi kompresi selesai, akan dikembalikan response status 200 dan header Access-Control-Allow-Origin untuk link manapun.

```
@app.route('/upload', methods=['POST'])
def fileUpload():
    target=os.path.join(UPLOAD_FOLDER)
    if not os.path.isdir(target):
        os.mkdir(target)
    file = request.files['file']
    ratio = request.form['ratio']
    filename = secure_filename(file.filename)
    logger.info(" [] Uploaded File " + str(filename))
    destination= target + filename
    print(destination)
    if not(os.path.exists(destination)):
        file.save(destination)
    session['uploadFilePath']=destination
    logger.info(" [] File downloaded, proceeding to compression stage")
    compress.mainCompress(filename, ratio)
    response = app.response_class(status=200)
    response.headers.add('Access-Control-Allow-Origin', '*')
    return response
```

Gambar 9. Upload dan Compress

#### 4. View File

```
@app.route("/view/<path:name>")
def view_file(name):
    return send_from_directory(
        app.config['DOWNLOAD_FOLDER'], name, as_attachment=False
    )
```

Gambar 10. View File

Bagian ini bila dipanggil dari *browser* akan mengembalikan gambar apabila ada dalam *directory* yang ditetapkan DOWNLOAD\_FOLDER berupa link gambar yang akan ditampilkan di *browser*. Gambar yang akan dikembalikan di sini merupakan gambar hasil kompresi berupa file

#### 5. Download File

```
@app.route("/download/<path:name>")
def download_file(name):
    return send_from_directory(
        app.config['DOWNLOAD_FOLDER'], name, as_attachment=True
    )
```

Gambar 11. Download File

Bagian ini dipanggil dari *browser* untuk mengembalikan gambar apabila ada dalam *directory* yang ditetapkan DOWNLOAD\_FOLDER berupa file gambar. File kemudian akan diperbolehkan untuk didownload oleh pengguna *browser*.



### 3.2.2 Program pada sisi peladen (FileUpload.jsx)

Pada kerangka situs React, dapat digunakan juga bahasa pemrograman typescript untuk fungsi yang mudah, di sini kami mengimplementasikan beberapa fungsi untuk tampilan depan situs.

#### 1. Mengubah rasio (handleValueChange)

```
handleValueChange(event){  
  event.preventDefault();  
  const inputValue = event.target.value;  
  this.setState({  
    value: inputValue  
  })  
}
```

*Gambar 12. Mengubah rasio*

Slider pada halaman depan dan input rasio yang bisa dimasukkan user saling terhubung. Untuk menghubungkannya diimplementasikan fungsi typescript untuk mengubah nilai yang akan disimpan, dan kemudian ditampilkan di *input box* rasio.

#### 2. Mengubah gambar yang akan dikompres (handleChange)

```
handleImageChange(event) {  
  event.preventDefault();  
  this.setState({  
    file: URL.createObjectURL(event.target.files[0]),  
    compressed: blur,  
    isCompressed: false,  
  })  
}
```

*Gambar 13. Mengubah gambar yang akan dikompres*

Terdapat tampilan *showcase* gambar mana yang telah dipilih user untuk di-*upload* dan kemudian dikompresi. Untuk mengubah tampilan tersebut dibutuhkan fungsi baru yang kita namakan *handleImageChange*. Fungsi ini juga akan mengubah tampilan gambar hasil kompresi dan menghilangkan kembali tombol unduh dan info hasil kompresi.

### 3. Mengunggah dan mengkompres gambar

```
handleUploadImage(ev) {  
  ev.preventDefault();  
  ev.stopPropagation();  
  ev.nativeEvent.stopImmediatePropagation();  
  
  if (this.uploadInput.files[0] !== undefined) {  
    this.setState({  
      isCompressing: true,  
    })  
    const filename = this.uploadInput.files[0].name  
    const data = new FormData();  
    data.append('file', this.uploadInput.files[0]);  
    data.append('filename', filename);  
    data.append('ratio', this.state.value);  
    var startDate = Date.now();  
    const config = {  
      method: 'POST',  
      body: data,  
    }  
    fetch('http://localhost:5000/compress', config)  
    .then(() => {  
      const postcompressionsrc = 'compress'.concat(filename);  
      var endDate = (Date.now() - startDate)/1000;  
      this.setState({  
        compressed: "http://127.0.0.1:5000/view/".concat(postcompressionsrc),  
        downloadlink: "http://127.0.0.1:5000/download/".concat(postcompressionsrc),  
        timeTaken : endDate,  
        pixelDiff: 100 - this.state.value,  
        isCompressed : true,  
        isCompressing: false,  
      });  
    })  
  }  
}
```

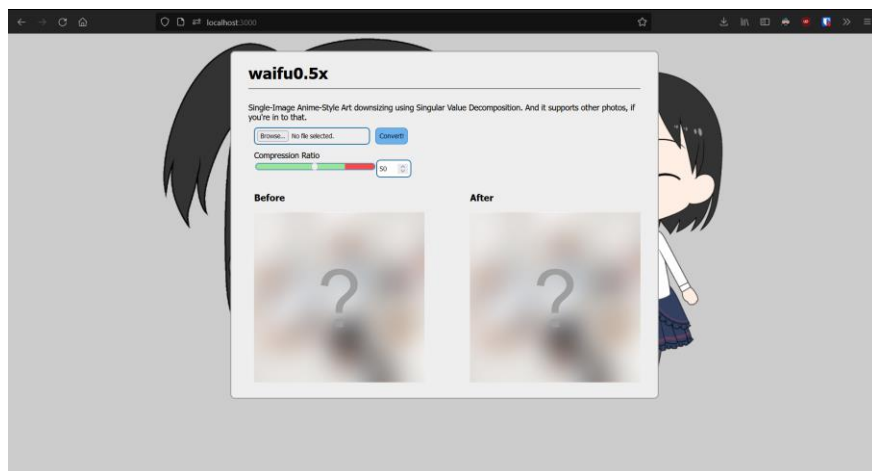
```
handleUploadImage(ev) {  
  ev.preventDefault();  
  ev.stopPropagation();  
  ev.nativeEvent.stopImmediatePropagation();  
  
  if (this.uploadInput.files[0] !== undefined) {  
    this.setState({  
      isCompressing: true,  
    })  
    const filename = this.uploadInput.files[0].name  
    const data = new FormData();  
    data.append('file', this.uploadInput.files[0]);  
    data.append('filename', filename);  
    data.append('ratio', this.state.value);  
    var startDate = Date.now();  
    const config = {  
      method: 'POST',  
      body: data,  
    }  
    fetch('http://localhost:5000/upload', config)  
    .then(() => {  
      const postcompressionsrc = 'compress'.concat(filename);  
      var endDate = (Date.now() - startDate)/1000;  
      this.setState({  
        compressed: "http://127.0.0.1:5000/view/".concat(postcompressionsrc),  
        downloadlink: "http://127.0.0.1:5000/download/".concat(postcompressionsrc),  
        timeTaken : endDate,  
        pixelDiff: 100 - this.state.value,  
        isCompressed : true,  
        isCompressing: false,  
      });  
    })  
  }  
}
```

Gambar 14. Mengubah gambar yang akan dikompres

Dibutuhkan fungsi untuk menangani kompresi, kita mengimplementasikannya dengan membuat fungsi `handleUploadImage`. Akan diambil file gambar yang telah dipilih dan rasio yang ditetapkan pengguna. Apabila gambarnya belum dipilih user, maka tidak akan terjadi apa-apa. Apabila ada, akan ditetapkan bahwa kita sekarang sedang mengkompresi sebuah gambar (`isCompressing: true`), dengan ini form pemilihan gambar, slider, dan tombol `convert` akan dihilangkan sementara. Kemudian siapkan form data yang akan dikirimkan ke `api.py` melalui `localhost:5000`, dengan data rasio, nama file, dan file yang akan dikompresi. Setelah api selesai, akan ditetapkan beberapa hal, seperti link gambar hasil kompresi, link download gambar hasil kompresi, waktu yang dibutuhkan untuk kompresi, bahwa kita tidak sedang mengkompresi gambar, dan bahwa kita sudah mengkompresi suatu gambar.

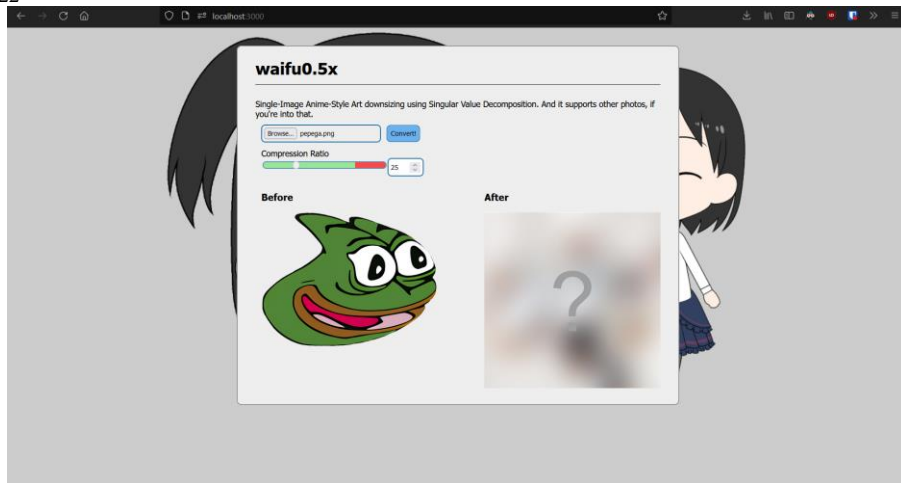
### 3.2.3 Program pada sisi klien (`index.html`)

kerangka situs React akan me-render semua komponen yang diberikannya pada `index.html`. Program hanya memiliki satu halaman saja yang merupakan halaman utama.



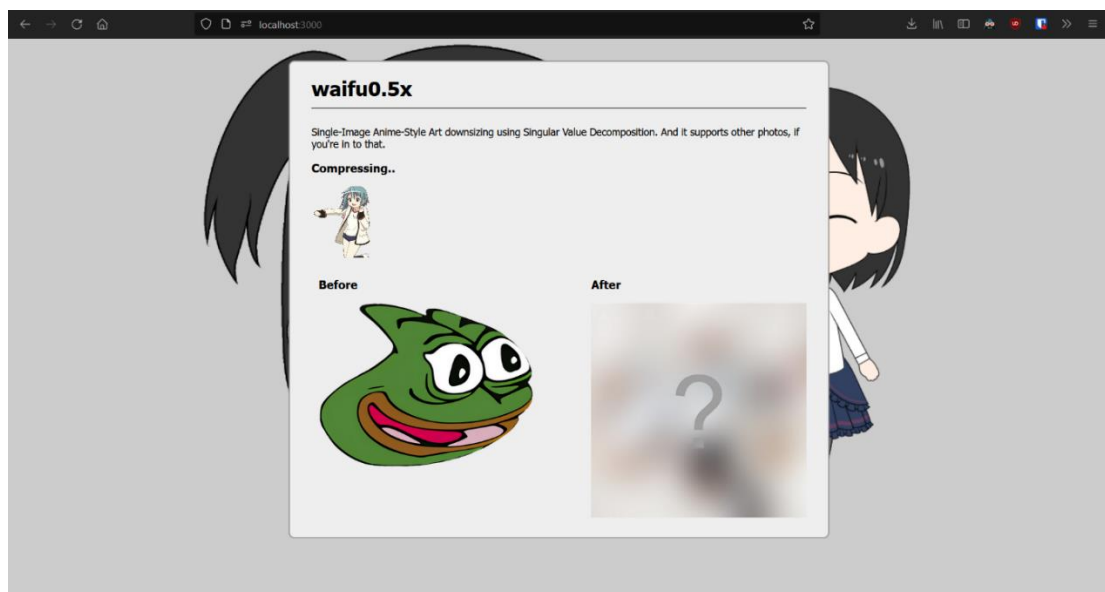
***Gambar 15.** Tampilan awal situs*

Pengguna dapat memasukkan gambar dari tombol “Browse” dan kemudian akan ditampilkan di bawah tulisan “Before”. Pengguna juga dapat memilih rasio kompresi melalui slider atau box di sebelahnya.

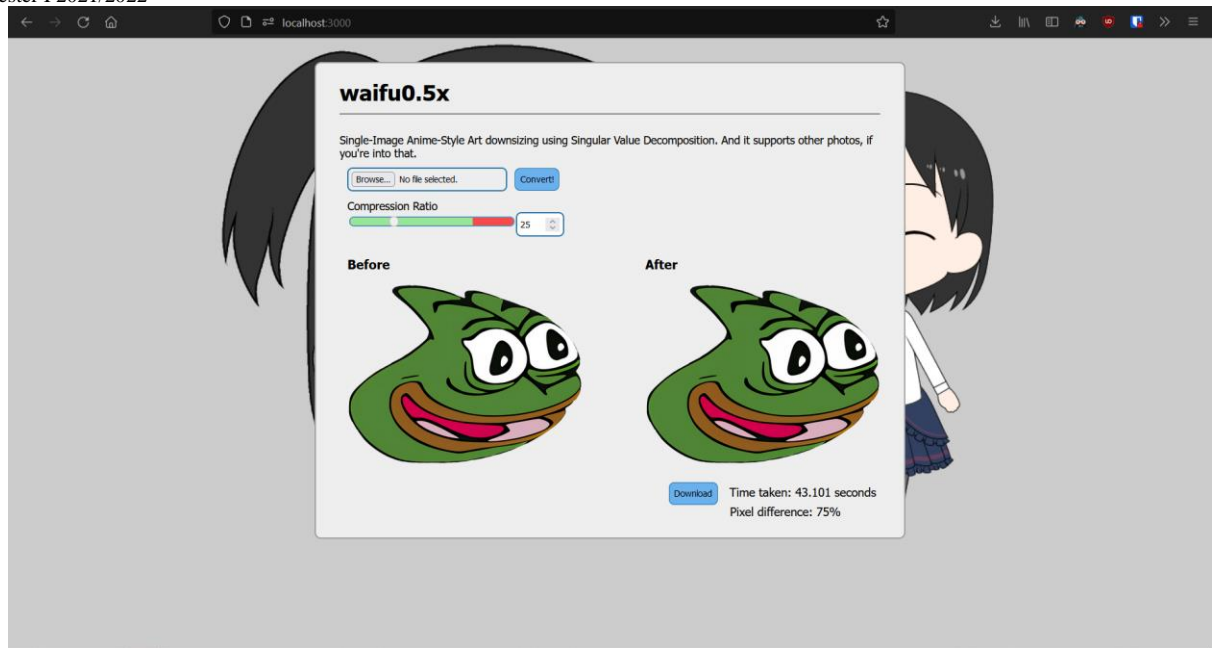


*Gambar 16. Tampilan situs setelah pengguna memilih gambar dan rasio kompresi*

Setelah memilih rasio kompresi dan gambar, pengguna dapat menekan tombol “Convert!” untuk memulai kompresi gambar.



*Gambar 17. Tampilan situs ketika sedang mengkompresi gambar*



*Gambar 18. Tampilan situs setelah kompresi selesai*

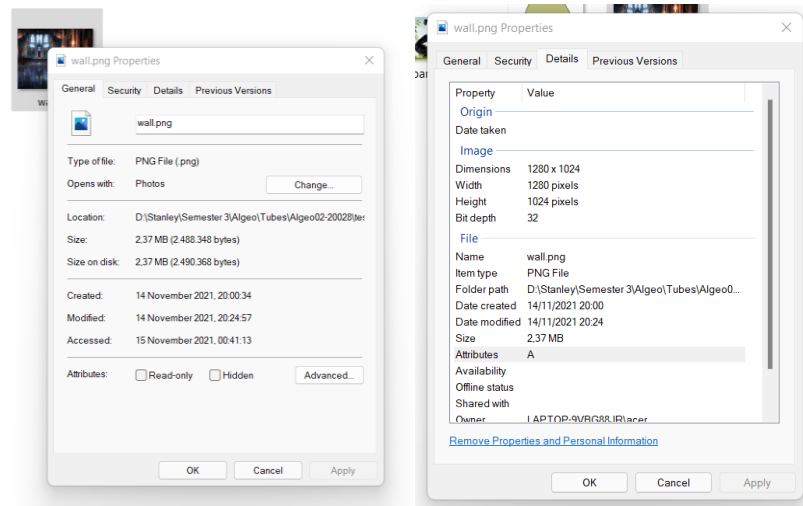
Website menampilkan *output* hasil kompresi gambar, *runtime* algoritma kompresi, serta perbedaan pixel (*pixel difference*) antara gambar awal dengan gambar hasil kompresi. Rumus *pixel difference* yaitu  $100\% - \text{rasio kompresi}$ , di mana rasio kompresi merupakan *input* user.

## BAB 4: Eksperimen

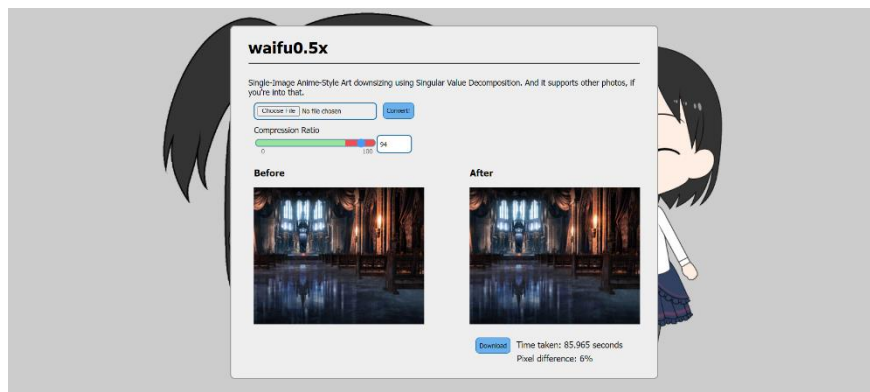
### 4.1. Pengujian

#### 1. Kasus A

Pada kasus A ini kami menggunakan gambar dengan nama “wall.png” dengan ukuran sekitar 2.37 Mb. Untuk informasi gambar selengkapnya bisa dilihat di bawah ini

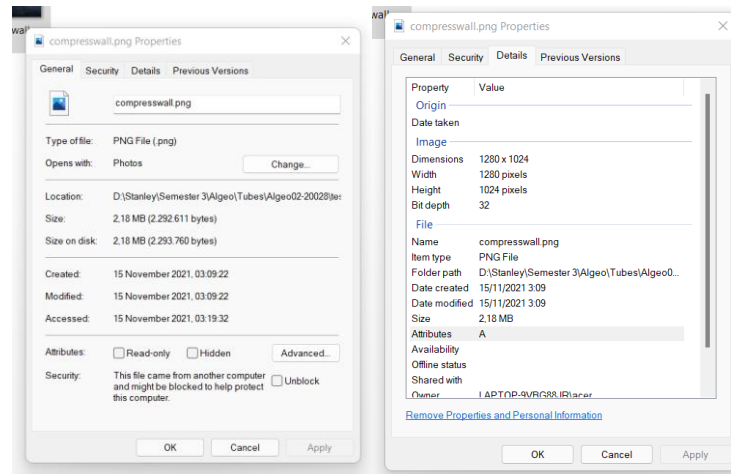


Gambar 19. Informasi gambar “wall.png”



Gambar 20. Hasil kompresi gambar “wall.png”

Untuk mengkompresi gambar “wall.png” dengan tingkat kompresi 6% dari gambar awal, diperlukan waktu selama sekitar 85,9 detik

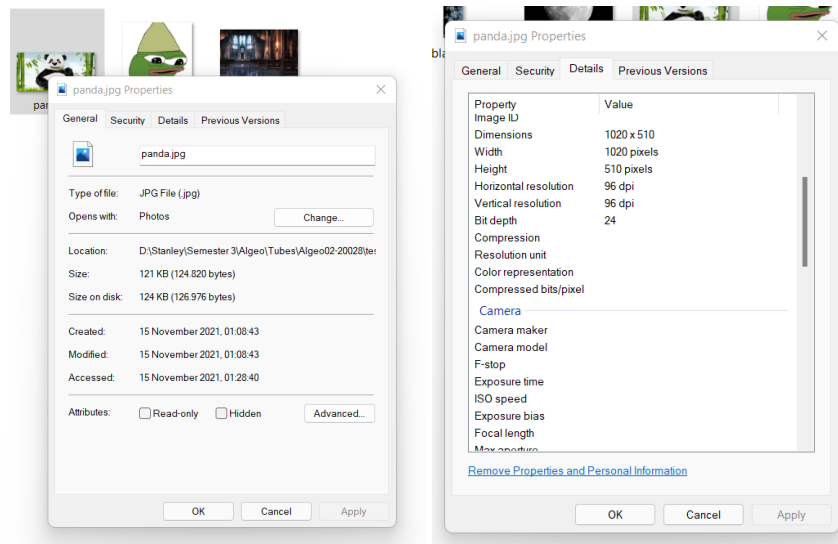


Gambar 21. Ukuran hasil kompresi gambar “wall.png”

Dari hasil kompresi, gambar “wall.png” kini turun ukurannya menjadi 2,18 Mb atau turun sekitar 190 kb.

## 2. Kasus B

Pada kasus B ini kami menggunakan gambar dengan nama “panda.jpg” dengan ukuran sekitar 121 kb. Untuk informasi gambar selengkapnya bisa dilihat di bawah ini

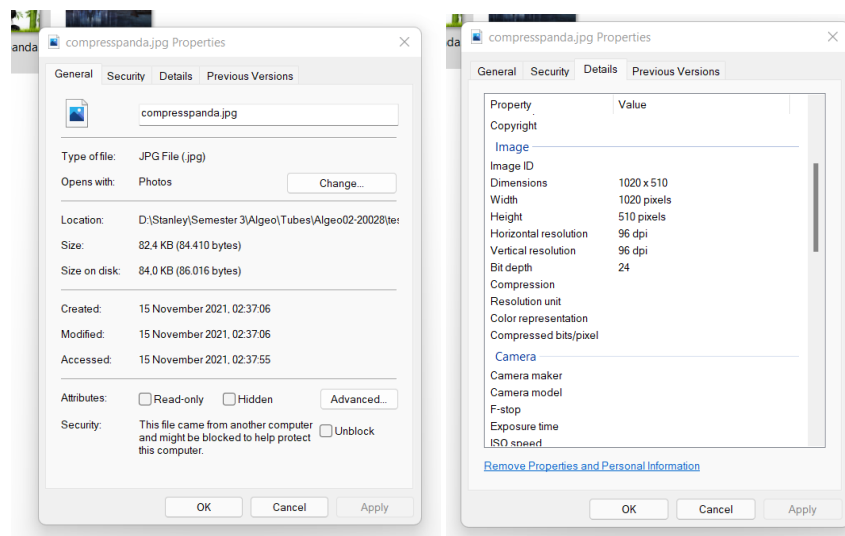


Gambar 22. Informasi gambar “panda.jpg”



*Gambar 23. Hasil kompresi gambar “panda.jpg”*

Untuk mengkompresi gambar “panda.jpg” dengan tingkat kompresi 63% dari gambar awal, diperlukan waktu selama sekitar 17 detik



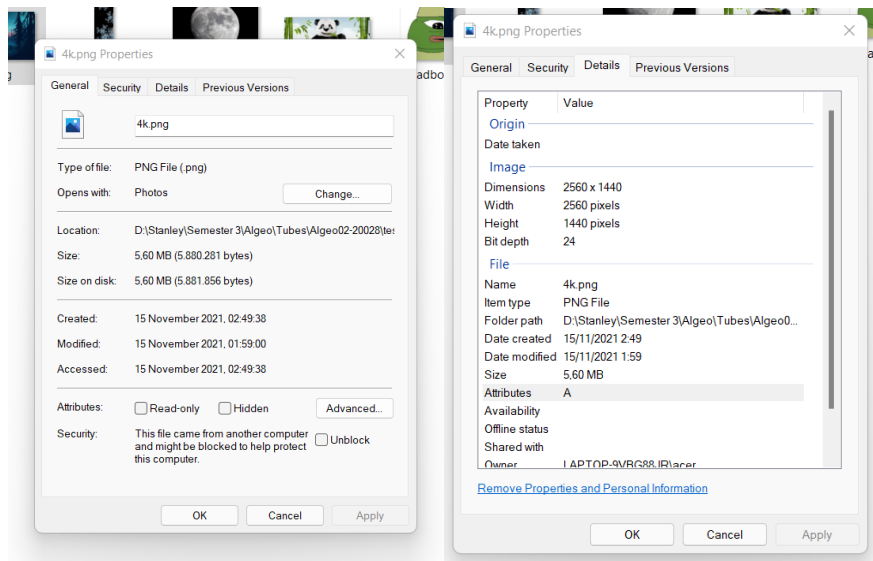
*Gambar 24. Ukuran hasil kompresi gambar “panda.jpg”*

Dari hasil kompresi, gambar “wall.png” kini turun ukurannya menjadi 82,4 kb atau turun sekitar 38,6 kb.



### 3. Kasus C

Pada kasus C ini kami menggunakan gambar dengan nama “4k.png” dengan ukuran sekitar 5,6 Mb. Untuk informasi gambar selengkapnya bisa dilihat di bawah ini

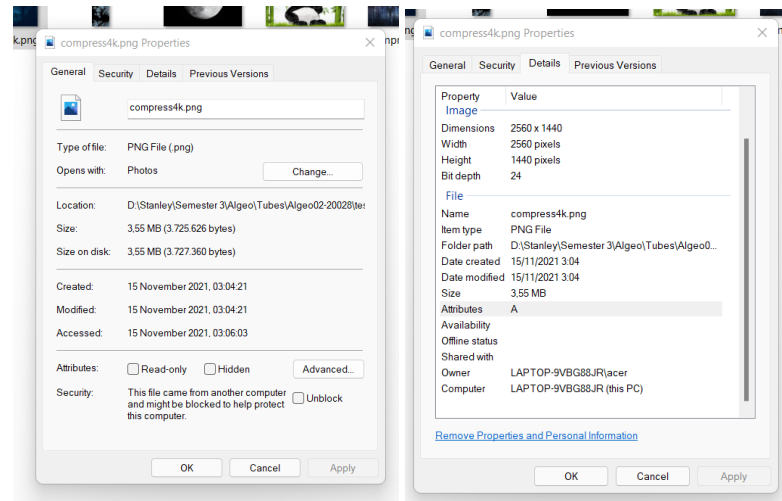


*Gambar 25. Informasi gambar “4k.png”*



*Gambar 26. Hasil kompresi gambar “4k.png”*

Untuk mengkompresi gambar “4k.png” dengan tingkat kompresi 4% dari gambar awal, diperlukan waktu selama sekitar 165,8 detik

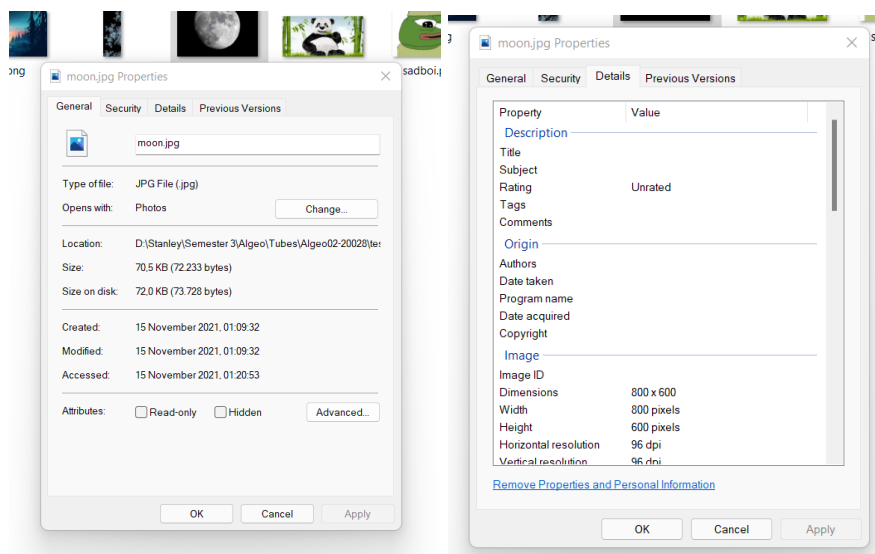


*Gambar 27. Ukuran hasil kompresi gambar “4k.png”*

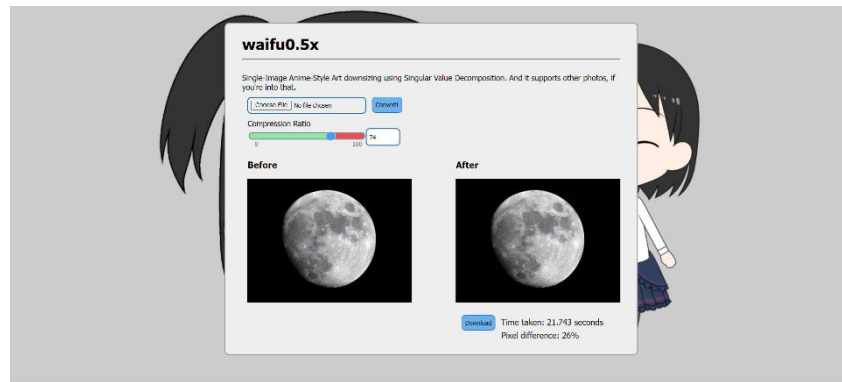
Dari hasil kompresi, gambar “wall.png” kini turun ukurannya menjadi 3,55 Mb atau turun sekitar 2,05 Mb.

#### 4. Kasus D

Pada kasus D ini kami menggunakan gambar dengan nama “moon.jpg” dengan ukuran sekitar 70,5 kb. Untuk informasi gambar selengkapnya bisa dilihat di bawah ini

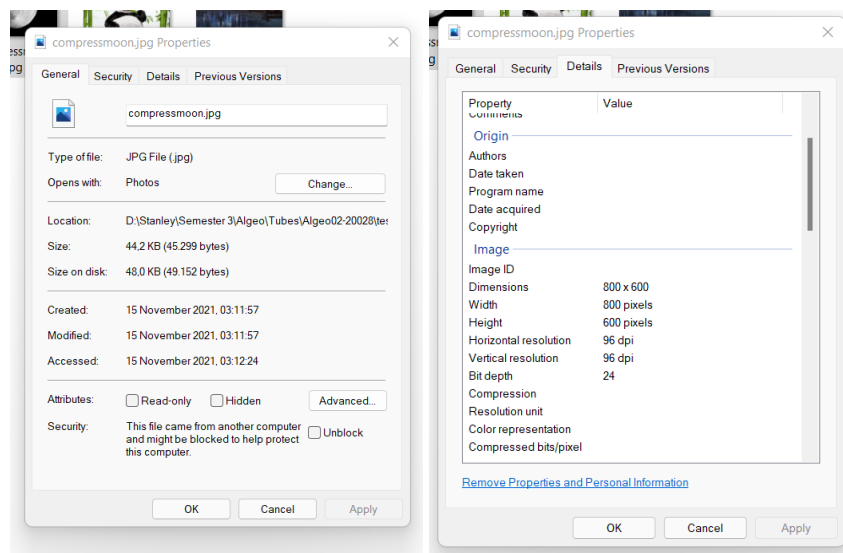


*Gambar 28. Informasi gambar “moon.jpg”*



*Gambar 29. Hasil kompresi gambar “moon.jpg”*

Untuk mengkompresi gambar “black.jpg” dengan tingkat kompresi 26% dari gambar awal, diperlukan waktu selama sekitar 21,7 detik



*Gambar 30. Ukuran hasil kompresi gambar “moon.jpg”*

Dari hasil kompresi, gambar “wall.png” kini turun ukurannya menjadi 44,2 kb atau turun sekitar 26,3 kb.

## 4.2. Analisis

Dari beberapa kasus di atas, kami dapatkan jika ukuran gambar semakin besar, waktu yang dibutuhkan untuk mengkompresinya juga semakin besar. Untuk ukuran 2650x1440 dibutuhkan waktu 165 detik, sedangkan untuk ukuran 800x600 hanya dibutuhkan waktu 21 detik. Untuk mengkompresi gambar dari jenis png, semakin besar ukuran gambarnya akan semakin besar pula perbedaan hasil kompresinya, walaupun tingkat rasionya sama. Sedangkan untuk gambar dari jenis jpg dan jpeg, hasil kompresi cenderung konsisten. Untuk mengkompresi gambar dari jenis png, semakin besar ukuran gambarnya akan

semakin besar pula perbedaan hasil kompresinya, walaupun tingkat rasionya sama.  
Sedangkan untuk gambar dari jenis jpg dan jpeg, hasil kompresi cenderung konsisten.

## BAB 5: Kesimpulan, Saran, dan Refleksi

### 5.1. Kesimpulan

1. Hal yang dicapai dalam tugas besar kedua IF2123 Aljabar Linear dan Geometri yaitu membuat program kompresi gambar dengan memanfaatkan algoritma SVD dalam bentuk website lokal sederhana.
2. Bahasa pemrograman yang dipilih merupakan Python dan JavaScript serta *framework* yang dipilih merupakan Flask dan React.
3. Kami telah berhasil membuat website yang dapat menerima *file* gambar beserta *input* tingkat kompresi gambar.
4. Kami telah berhasil menampilkan gambar *input*, *output*, *runtime* algoritma, dan persentase hasil kompresi gambar (perubahan jumlah pixel gambar) serta *file output* hasil kompresi dapat diunduh melalui website
5. Kami telah berhasil membuat program yang dapat mempertahankan transparansi dari gambar asli, contohnya gambar png dengan *background* transparan.

### 5.2. Saran

Untuk mengembangkan program lebih lanjut, kami memiliki beberapa saran. Pertama, membersihkan tampilan antarmuka situs. Karena keterbatasan waktu, kami belum sempat mempelajari lebih lanjut untuk pengaturan CSS sehingga memilih tema laman yang bisa dibilang simpel. Kedua, apabila memungkinkan, membuat algoritma kompresi yang lebih efisien. Karena keterbatasan waktu, algoritma kami bisa dibilang masih berjalan dengan waktu yang cukup lama dibanding standar internet.

### 5.3. Refleksi

#### 5.3.1. Timothy Stanley Setiawan (13520028)

Dalam pengerjaan tugas besar ini, saya merasa ada banyak hal yang saya perlu saya gali dalam hal membuat situs, terutama dalam pengerjaan *back-end*. Selama pengerjaan tubes, walaupun saya sudah bekerja sama dengan teman sekelompok, saya masih kesulitan untuk *debugging* yang terjadi.

#### 5.3.2. Farrel Farandieka Fibriyanto (13520054)

Dalam pengerjaan tugas besar ini, saya rasa ada banyak hal yang saya bisa tingkatkan. Terutama dalam pengerjaan *back-end* situs. Saya rasa saya masih awam dalam hal membaca dokumentasi kaskas dan dalam hal *debugging*.

### 5.3.3. Jeremy Rionaldo Pasaribu (13520082)

Ketika mengerjakan tugas besar ini, saya merasa masih terdapat banyak hal yang dapat ditingkatkan jika saya dapat membagi waktu dengan baik. Hal ini terutama pada pengerjaan *user interface* situs ataupun pengerjaan *back-end* situs yang memakan banyak waktu dalam pengerjaan tugas besar ini.

## **BAB 6: Pembagian Tugas**

NO	BAGIAN	PENANGGUNG JAWAB
1	Algoritma SVD	Stanley, Farrel F, Jeremy R
2	Front-end (React)	Stanley, Farrel F, Jeremy R
3	Back-end (Flask & Python)	Stanley, Farrel F, Jeremy R
4	Laporan	Stanley, Farrel F, Jeremy R

## Referensi

Anthonissen, Martijn. “Orthogonal Iteration”, diakses dari :

<https://www.dropbox.com/s/9mqp0fxrsze9zuw/4-5%20orthogonal%20iteration.pdf?dl=0>

Flask, diakses dari :

<https://flask.palletsprojects.com/en/2.0.x/>

Grinberg, Miguel, diakses dari :

<https://blog.miguelgrinberg.com/post/how-to-create-a-react--flask-project>

Munir, Rinaldi. “Nilai Eigen dan Vektor Eigen (Bagian 1)”, diakses dari :

[https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-  
Nilai-Eigen-dan-Vektor-Eigen-Bagian1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-<br/>Nilai-Eigen-dan-Vektor-Eigen-Bagian1.pdf)

Munir, Rinaldi. “Nilai Eigen dan Vektor Eigen (Bagian 2)”, diakses dari :

[https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-  
Nilai-Eigen-dan-Vektor-Eigen-Bagian2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18-<br/>Nilai-Eigen-dan-Vektor-Eigen-Bagian2.pdf)

Munir, Rinaldi. “Singular Value Decomposition (SVD)”, diakses dari :

[https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-19b-  
Singular-value-decomposition.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-19b-<br/>Singular-value-decomposition.pdf)

Pandey, Ashish. “File Upload with React & Flask”, diakses dari :

<https://medium.com/excited-developers/file-upload-with-react-flask-e115e6f2bf99>

Qiao, S. “Eigensystems and SVD”, diakses dari :

<http://www.cas.mcmaster.ca/~qiao/courses/cas708/eig.pdf>