

LAPORAN TUGAS BESAR 2
IF2123 ALJABAR LINIER DAN GEOMETRI
APLIKASI NILAI EIGEN DAN VEKTOR EIGEN DALAM KOMPRESI
GAMBAR



oleh

Firizky Ardiansyah	13520095
Ikmal Alfaozi	13520125
Bayu Samudra	13520128

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

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

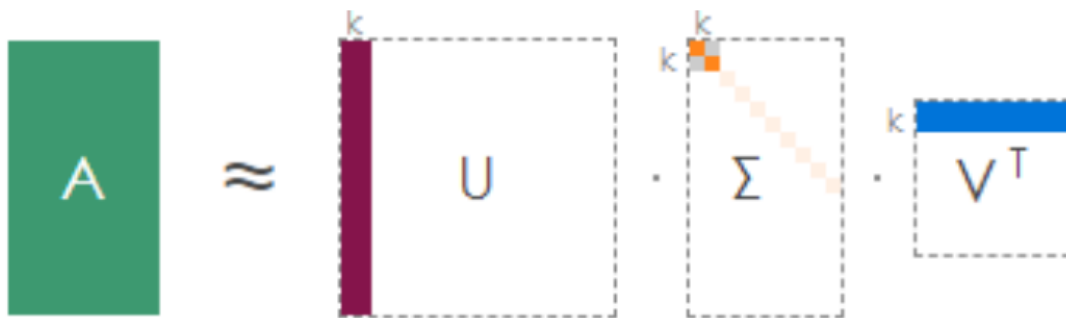
Sumber : Understanding Compression in Digital Photography (lifewire.com)

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 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. 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.

BAB 2

TEORI SINGKAT

A. Perkalian Matriks

Syarat dua buah matriks dapat dikalikan adalah jumlah baris matriks pertama dan jumlah kolom matriks kedua harus sama. Perkalian dua buah matriks berukuran $A_{m \times r} B_{r \times n} = C_{m \times n}$.

Misalkan $A = [a_{ij}]$ dan $B = [b_{ij}]$, maka $C = AB = [c_{ij}]$ dengan $c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{in}b_{nj}$

B. Nilai 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 dengan x , yaitu

$$Ax = \lambda x$$

Skalar disebut nilai eigen dari A , dan x dinamakan vektor eigen yang berkorespondensi dengan λ . Vektor eigen dan nilai eigen dari matriks A dapat dihitung seperti berikut.

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

Solusi trivial dari $(A - \lambda I)x$ adalah $x = 0$. Agar $(A - \lambda I)x = 0$ memiliki solusi tidak-nol, haruslah $\det(A - \lambda I) = 0$. Persamaan $\det(A - \lambda I) = 0$ disebut persamaan karakteristik dari matriks A , yaitu λ , dan akar-akar persamaan tersebut merupakan akar-akar karakteristik atau nilai-nilai eigen.

C. Vektor Eigen

Vektor Eigen adalah solusi dari $(A - \lambda I)x = 0$. Dengan x adalah vektor eigen itu sendiri yang berkorespondensi dengan nilai eigen.

D. Matriks SVD

Misalkan A adalah matriks persegi berukuran $n \times n$, matriks persegi A dikatakan dapat didiagonalisasi jika ia mirip dengan matriks diagonal, yaitu terdapat matriks E sedemikian sehingga $E^{-1}AE$ adalah matriks diagonal. Dalam hal ini E mendiagonalisasi matriks A .

Matriks diagonal adalah matriks yang semua elemen di atas dan di bawah diagonal utama adalah nol. E adalah matriks yang kolom-kolomnya adalah basis ruang eigen dari matriks A, yaitu:

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

Misalkan D adalah matriks diagonal, maka

$$A = EDE^{-1} \rightarrow D = E^{-1}AE$$

Metode SVD merupakan salah satu metode untuk memfaktorkan matriks non-bujur sangkar berukuran $m \times n$ yang tidak memiliki nilai eigen. Metode SVD dilakukan dengan memfaktorkan matriks A berukuran $m \times n$ menjadi matriks U, Σ , dan V sedemikian sehingga

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

U = matriks ortogonal $m \times m$,

V = matriks ortogonal $n \times n$

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

1. Diagonal utama dari matriks $m \times n$ adalah garis dari elemen pertama sudut kiri atas sampai ke kanan bawah sejauh mungkin.
2. Matriks ortogonal adalah matriks yang kolom-kolomnya merupakan vektor yang saling ortogonal atau tegak lurus satu sama lain.
3. Nilai-nilai singular matriks adalah akar pangkat dua nilai-nilai eigen dari matriks $A^T A$, yaitu $\sigma_1 = \sqrt{\lambda_1}, \sigma_2 = \sqrt{\lambda_2}, \dots, \sigma_n = \sqrt{\lambda_n}$, dengan $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$.

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

4. Untuk vektor singular kiri U, hitung nilai-nilai eigen dari AA^T . $\text{Rank}(A) = k =$ banyaknya nilai-nilai eigen tidak nol dari AA^T .
5. 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.
6. Untuk vektor singular kanan, hitung nilai-nilai eigen dari $A^T A$ lalu tentukan nilai-nilai singularnya.

7. 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 .
8. Bentuklah matriks 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 adalah akar pangkat dua dari nilai-nilai eigen yang tidak nol dari $A^T A$.
9. Maka, $A = U \Sigma V^T$.

BAB 3

IMPLEMENTASI PUSTAKA

A. Kakas Program

Pada pembuatan program ini, kami memisahkan dua bagian, yaitu *frontend* dan *backend*. Kakas *frontend* yang kami gunakan diantaranya adalah sebagai berikut:

1. ReactJs
2. Axios
3. Bootstrap 5
4. Socket.IO Client

Pada bagian *backend*, berikut ini adalah kakas yang kami gunakan:

1. FastAPI
2. Socket.IO

Kami melakukan *deployment* program pada Heroku untuk backend dan Github Pages untuk frontend. Kami melakukan deploy dari program kami pada alamat <http://compress.bayusamudra.my.id/>.

B. Endpoint Backend

Backend program ini memiliki beberapa API Endpoint, yaitu:

1. Endpoint “/” digunakan untuk menampilkan keberjalanan server.
2. Endpoint “/status” digunakan untuk menampilkan keadaan server. Pada endpoint ini juga ditampilkan informasi server apakah dapat digunakan atau tidak oleh user yang memanggil program ini.
3. Endpoint “/upload” digunakan untuk mengunggah gambar yang akan dikompresi oleh server. User haruslah telah melakukan emit pada event subscribe pada websocket sebelum melakukan hal ini.
4. Endpoint “/compress” digunakan untuk menampilkan hasil kompresi gambar. Hasil dapat dilihat apabila matriks SVD telah selesai dibentuk.

C. Event Websocket

Dalam implementasi backend, terdapat 5 event yang terdapat dalam interaksi websocket:

1. Event “subscribe” digunakan untuk mendaftarkan user agar bisa memakai *resources* pada *backend*. Setiap backend hanya dapat digunakan oleh satu user dikarenakan untuk menghemat resources dari backend.
2. Event “build-matrix” digunakan untuk memulai proses pembuatan matriks SVD. Hasil dari proses SVD akan disimpan pada variabel global dan dapat dipakai selagi user masih tersambung dengan server. Bila user telah disconnect dari server, semua hasil proses tersebut akan direset.
3. Event “unsubscribe” digunakan untuk mengembalikan resources pada server. User yang melakukan unsubscribe akan kehilangan semua hasil proses yang telah dilakukan sebelumnya.
4. Event “progress” digunakan untuk memberikan aliran data yang berisi status proses SVD kepada user.
5. Event “response” yang digunakan untuk memberikan aliran data mengenai status koneksi

D. Algoritma yang digunakan

Pada program ini, kami menggunakan algoritma pencarian nilai eigen QL implisit. Pada Algoritma eksplisit, Proses ini dilakukan dengan cara melakukan dekomposisi sebuah matriks menjadi dalam bentuk Q dan L , yang dalam hal ini Q sebagai matriks yang ortogonal sedangkan L berperan sebagai matriks segitiga bawah. Proses selanjutnya adalah melakukan perkalian LQ untuk mendapatkan matriks yang baru. Hal ini dilakukan hingga ditemukan matriks yang mendekati dengan matriks segitiga, yang mana dalam hal ini, matriks segitiga memiliki nilai eigen yang sama dengan nilai-nilai pada diagonalnya.

Nilai Eigen pada proses ini dipastikan tidak berubah, karena proses pada algoritma ini memanfaatkan sifat kemiripan matriks. Dua buah matriks dikatakan mirip apabila keduanya memenuhi sifat berikut:

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

Yang mana dalam hal ini matriks C dan matriks A memiliki nilai eigen yang sama. Pada Proses pencarian nilai eigen pada QL, dapat kita lihat, kita melakukan proses dekomposisi matriks sebagai berikut

$$Q_k(A_k - c_k I) = L_k$$

Untuk mendapatkan matriks selanjutnya, kita dapat melakukan matriks sebagai berikut

$$A_{k+1} = L_k Q_k^T + C_k I$$

Apabila kita substitusikan kedua persamaan diatas diperoleh

$$A_{k+1} = Q_k A_k Q_k^T$$

Dikarenakan Q merupakan matriks ortogonal, maka

$$A_{k+1} = Q_k A_k Q_k^{-1}$$

Oleh karena itu, matriks A_{k+1} dan matriks A_k merupakan matriks yang mirip.

Dalam proses sebenarnya, kita dapat melakukan proses ini dengan melakukan pergeseran pada diagonal utama. Pergeseran ini tidak akan mengubah nilai eigen dan juga vektoreigen yang dibentuk pada matriks A_{k+1} dan A_k . Proses shifting ini akan mempercepat proses konvergensi.

Dalam perhitungan SVD, kita perlu mencari nilai singular dari sebuah matriks beserta matriks singular kanan V. Hal ini dapat dicapai dengan cara menghitung nilai eigen $A^T A$. Hasil perkalian dari kedua matriks tersebut merupakan matriks simetris. Sifat kesimetrisan dari perkalian kedua matriks tersebut akan sangat bermanfaat dalam proses optimasi.

Optimasi yang dapat dilakukan untuk mempercepat proses perhitungan nilai eigen adalah mencari matriks tridiagonal yang mirip. Matriks tridiagonal ini dapat dibentuk dengan menghitung matriks hessenberg dari matriks $A^T A$. Proses perhitungan matriks hessenberg dapat dengan mudah kita lakukan dengan memanfaatkan transformasi householder. Pada program ini, kami menghitung matriks hessenberg dengan memanfaatkan pustaka yang terdapat pada scipy.

Pencarian matriks singular kiri dapat kita lakukan dengan memanfaatkan rumus berikut ini

$$u_i = \frac{1}{\sigma_i} A v_i$$

Proses penyusunan matriks harus didasarkan dengan urutan dari nilai singular. Nilai singular terbesar haruslah disusun dari yang terbesar hingga yang terkecil.

Setelah matriks komposisi tersebut didapatkan, kita dapat melakukan kompresi gambar dengan mengurangi rank dari matriks singular lalu menghitung hasil kali dari ketiga matriks tersebut.

E. Alur Program

User pertama kali akan ditampilkan dengan tampilan depan. Setelah itu, user diharuskan untuk mengambungkan ke server dengan menekan icon colokan. Saat user telah memberikan informasi backend yang diinginkan, program akan memeriksa apakah backend tersebut dapat digunakan oleh user.

Apabila user dapat menggunakan backend tersebut, user dapat mengupload file dengan cara meletakkan file gambar yang akan dikompresi pada area yang telah disediakan. Setelah gambar berhasil terupload, backend akan mengubah gambar tersebut menjadi matriks gambar. Mode gambar yang digunakan pada matriks adalah RGBA. Matriks tersebut akan disimpan pada state global dari program.

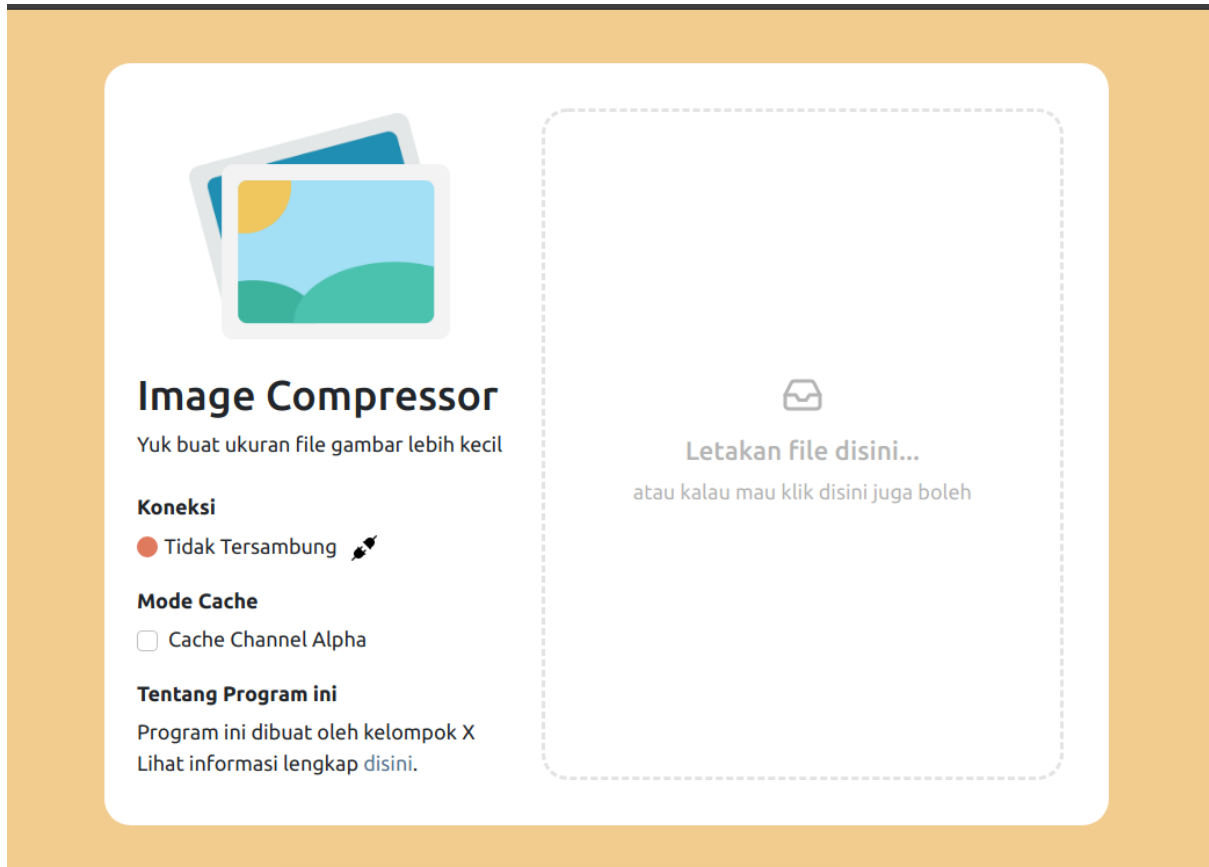
Setelah proses tersebut berhasil, program akan melakukan dekomposisi dari matriks gambar yang sudah dilakukan sebelumnya. Hasil dari dekomposisi gambar ini akan disimpan pada state global dengan tujuan untuk digunakan kembali pada proses selanjutnya.

Setelah program berhasil melakukan dekomposisi, user akan dapat memilih besar kompresi yang diinginkan. Gambar yang dihasilkan memanfaatkan hasil cache pada state global. Gambar yang dihasilkan akan menggunakan mode asal dari gambar.

State global tersebut akan disimpan hingga user terputus dari server. Selama user belum terputus dari server, user lain dipastikan tidak dapat mengakses server tersebut.

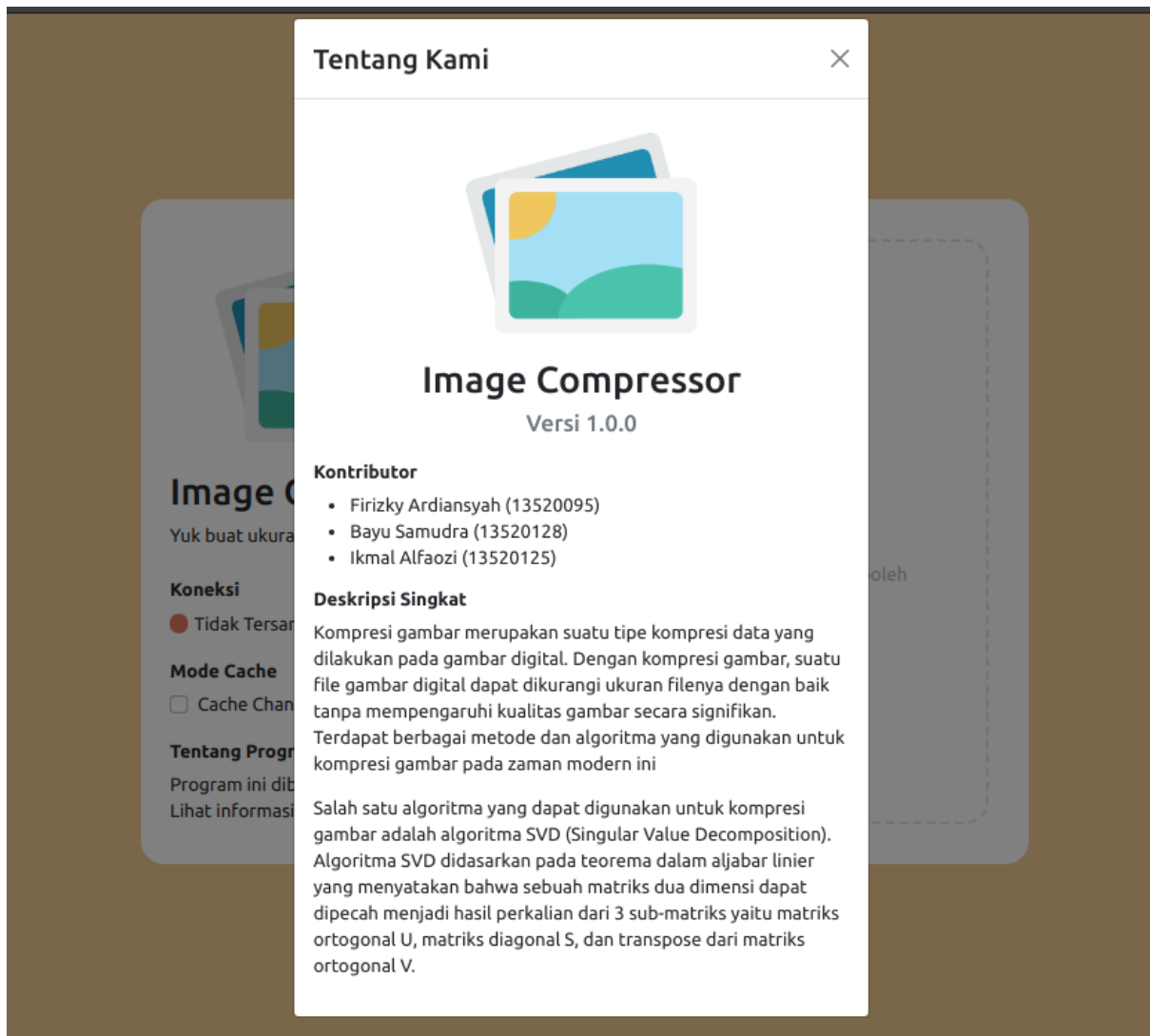
BAB 4 EKSPERIMEN

A. Tampilan Depan



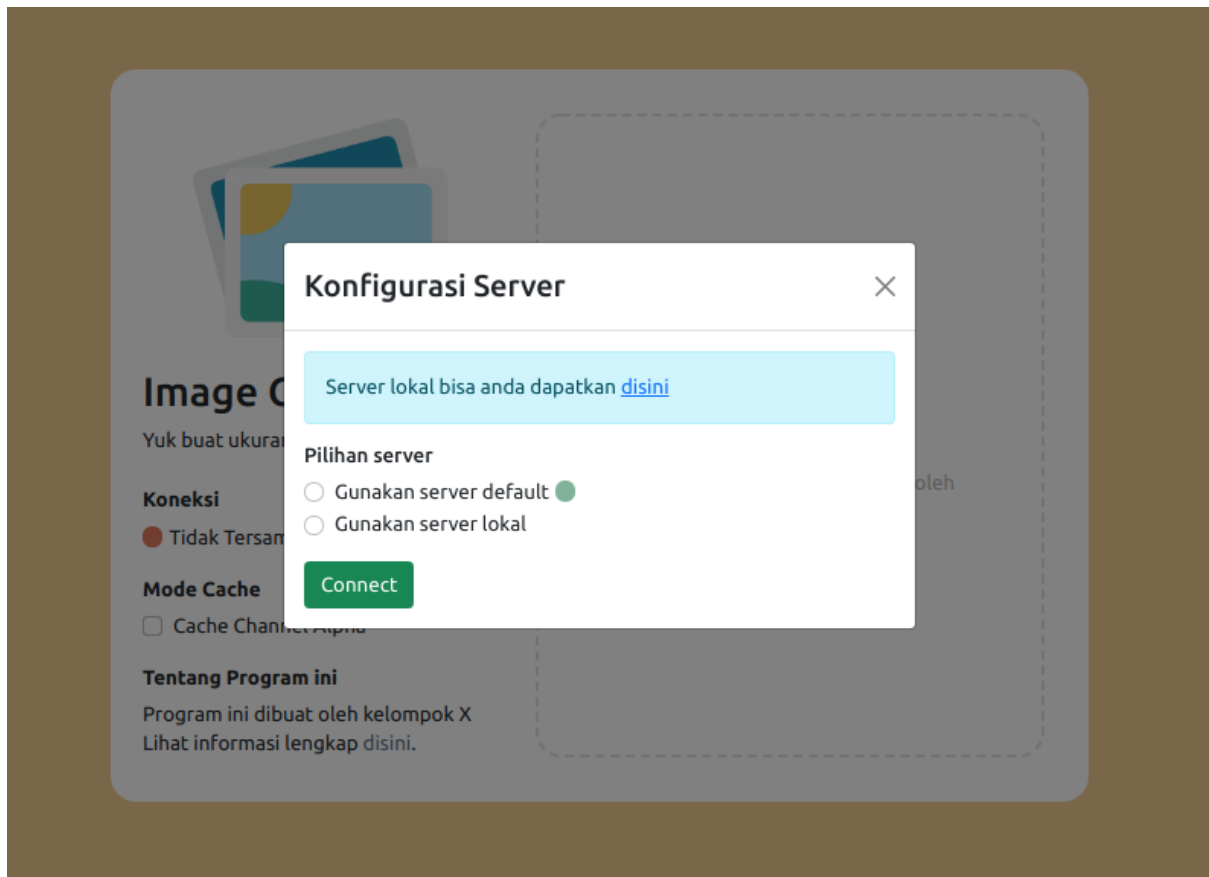
Gambar 4 Tampilan Depan Situs

Bagian depan terdiri atas judul situs, gambaran umum situs, informasi koneksi, dan input file yang akan dikompresi. Kotak centang mode *cache* digunakan sebagai parameter dekomposisi kanal warna transparant atau disebut sebagai kanal *Alpha*. Gambran umum situs memuat informasi pembangun situs dan informasi mengenai situs itu sendiri, berikut adalah antarmuka informasi lengkap pada situs.



Gambar 5 Tentang Kami

Adapun untuk menghubungkan situs ke server dapat dilakukan dengan menekan *icon* di sebelah kanan informasi koneksi.



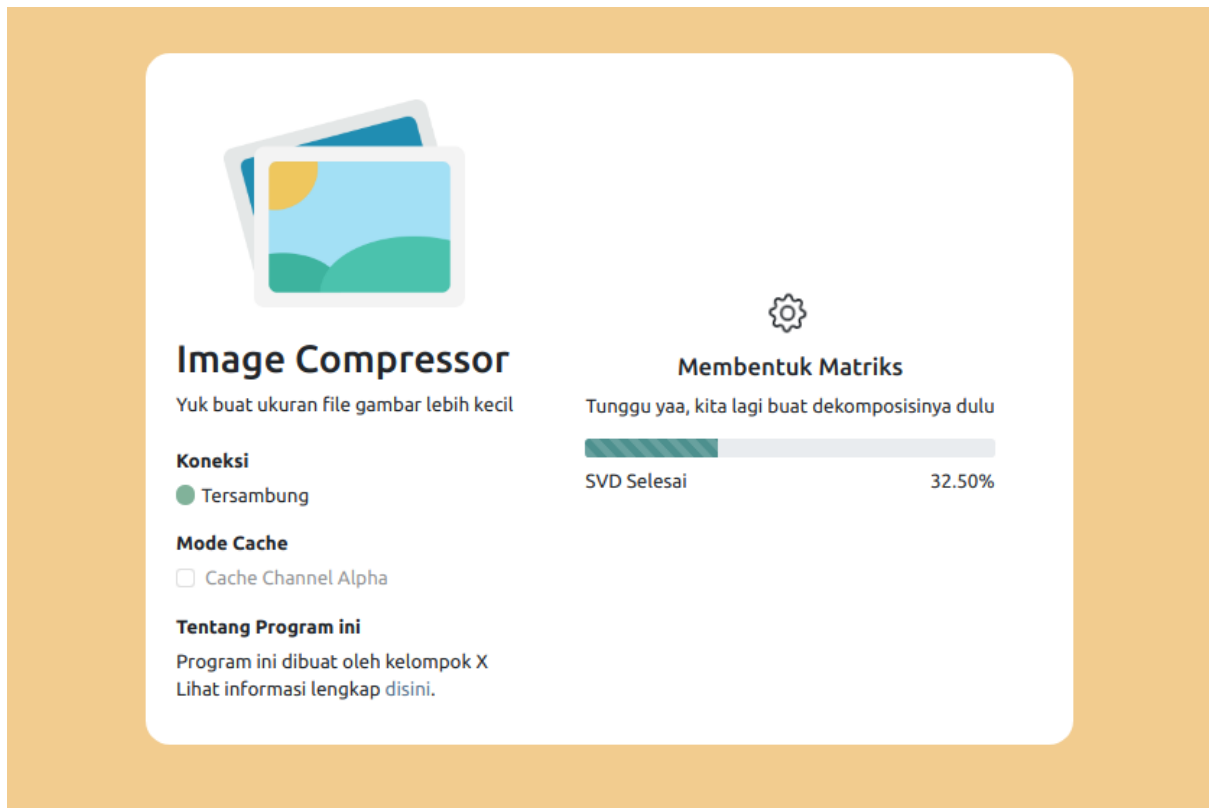
Gambar 6 Konfigurasi Server

Server default yang digunakan adalah platform heroku sedangkan server lokal membutuhkan pengguna untuk memasukkan server info sesuai dengan host lokalnya masing-masing ketika melakukan eksekusi program python pada source code.

Setelah terhubung ke server, pengguna bisa melakukan input file pada kotak yang disediakan.

B. Fitur Utama

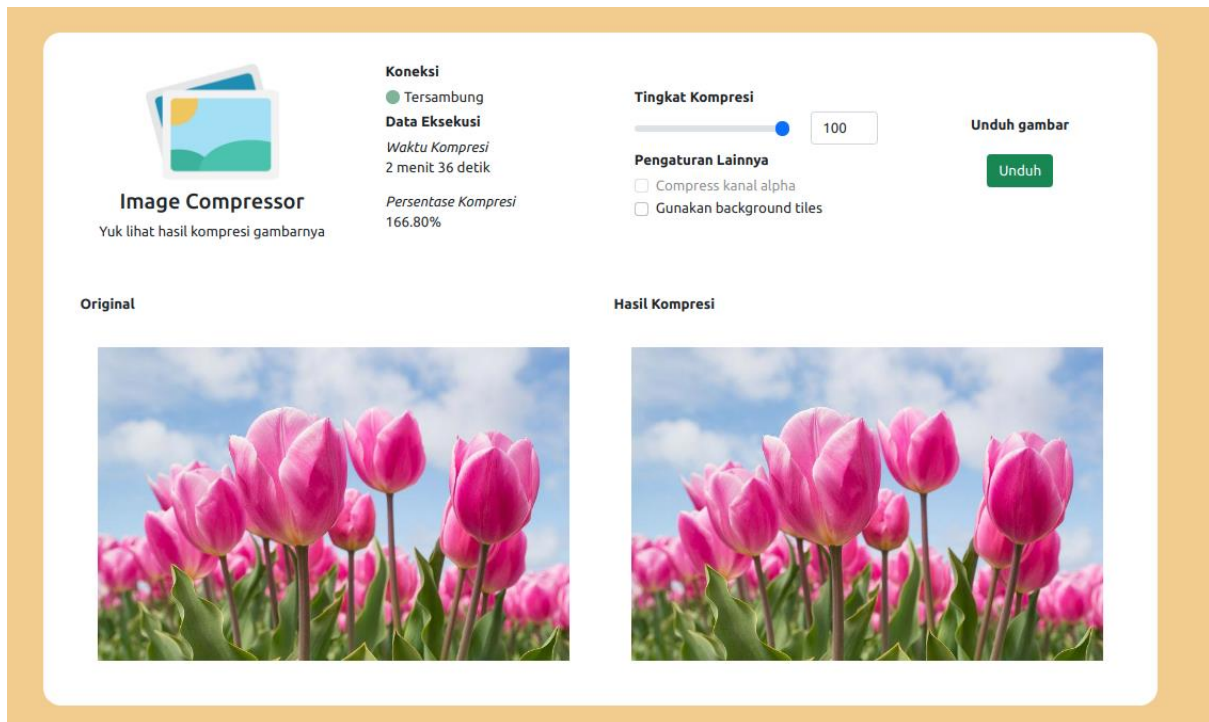
Setelah pengguna melakukan input pada kotak, akan muncul persentase eksekusi saat memuat data dan melakukan proses dekomposisi.



Gambar 7 Proses Muat dan Eksekusi

Lamanya eksekusi bergantung pada kecepatan server, CPU, dan ukuran gambar. Semakin besar ukuran pixel dari gambar, semakin lama eksekusinya.

Setelah proses berhasil dieksekusi, hasil kompresi akan muncul sebagai berikut



Gambar 8 Hasil Eksekusi

Tingkat kompresi adalah rank matriks baru yang akan dibangun dari matriks awal (dalam persentase), kotak centang kanal alpha dapat digunakan jika di awal pengguna memilih untuk melakukan dekomposisi kanal alpha, sedangkan kotak centang *background tiles* digunakan untuk membentuk *background* transparan di belakang input/output. Pergeseran nilai persentase tingkat kompresi bisa dilakukan tanpa waktu eksekusi yang signifikan dan gambar akan otomatis beradaptasi dengan parameter yang dimasukkan. Untuk mengunduh hasil kompresi, bisa dilakukan dengan menekan tombol unduh.

C. Analisis Kasus

Akan digunakan beberapa sampel uji sebagai analisis kasus untuk mengetes program. Variabel bebas kasus uji pada eksperimen ini diantaranya adalah ukuran pixel file, tipe file, persentase kompresi, dan tipe gambar. Sedangkan variabel terikatnya adalah waktu eksekusi dan gambar hasil.

Jika m, n adalah ukuran pixel gambar, sampel uji akan dilakukan pada:

6. $\min(m, n) < 500$. (ukuran kecil)
7. $500 \leq \min(m, n) < 1000$, dan (ukuran sedang)
8. $1000 \leq \min(m, n)$ (ukuran besar)

Adapun tipe file yang akan diuji adalah file berformat PNG, JPG, dan JPEG, persentase kompresi akan bergantung pada signifikansi perubahan output pada gambar, sedangkan tipe gambar yang akan diuji meliputi RGBA 32 bit/28 bit, RGB 28 bit/24 bit/16 bit, dan gambar dengan tipe *Grayscale* yang pada eksekusinya akan dianggap sebagai RGB *picture*.

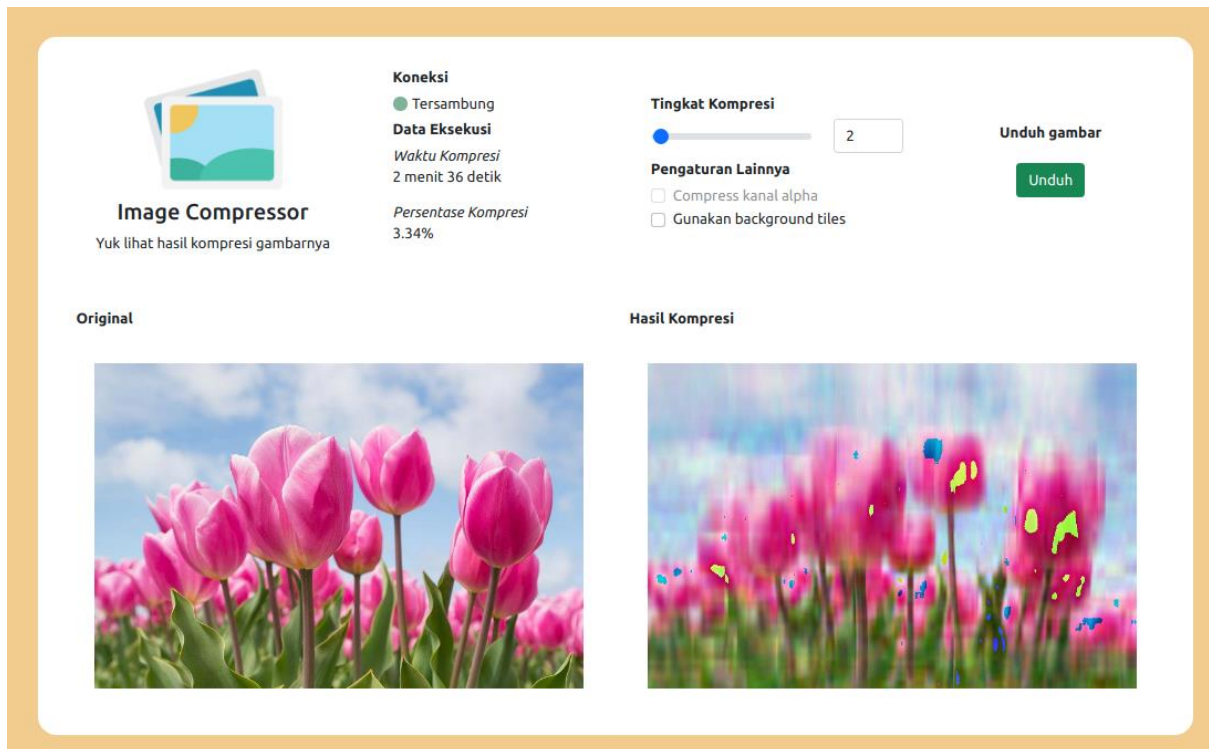
1. Gambar ukuran sedang RGB berekstensi JPEG

Gambar input yang digunakan adalah:



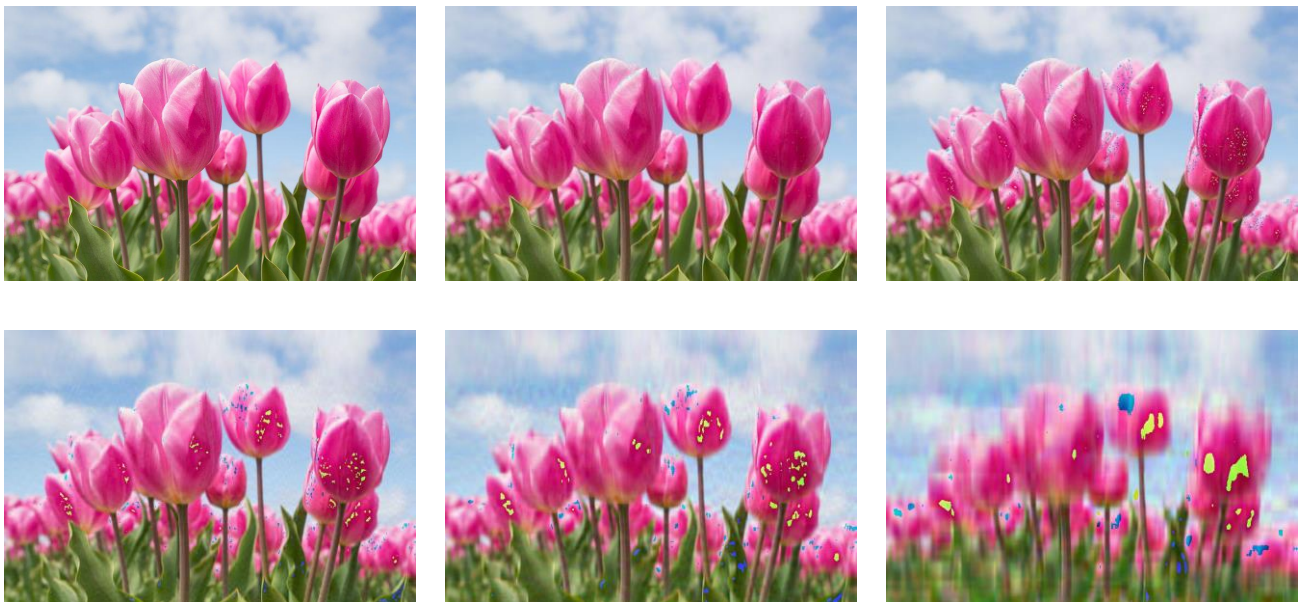
Gambar 9 Kasus Uji 1

Gambar ini memiliki ukuran pixel 750x500, berformat JPEG, dan memiliki 24 *bit depth* (RGB).



Gambar 10 Hasil Eksekusi Kasus Uji 1

Waktu total mendekomposisi adalah 2 menit 36 detik, berikut adalah hasil kompresi 50, 25, 10, 5, dan 2 persen



Gambar 11 Gambar original, 50, 25, 10, 5, 2 persen dekomposisi kasus uji 1

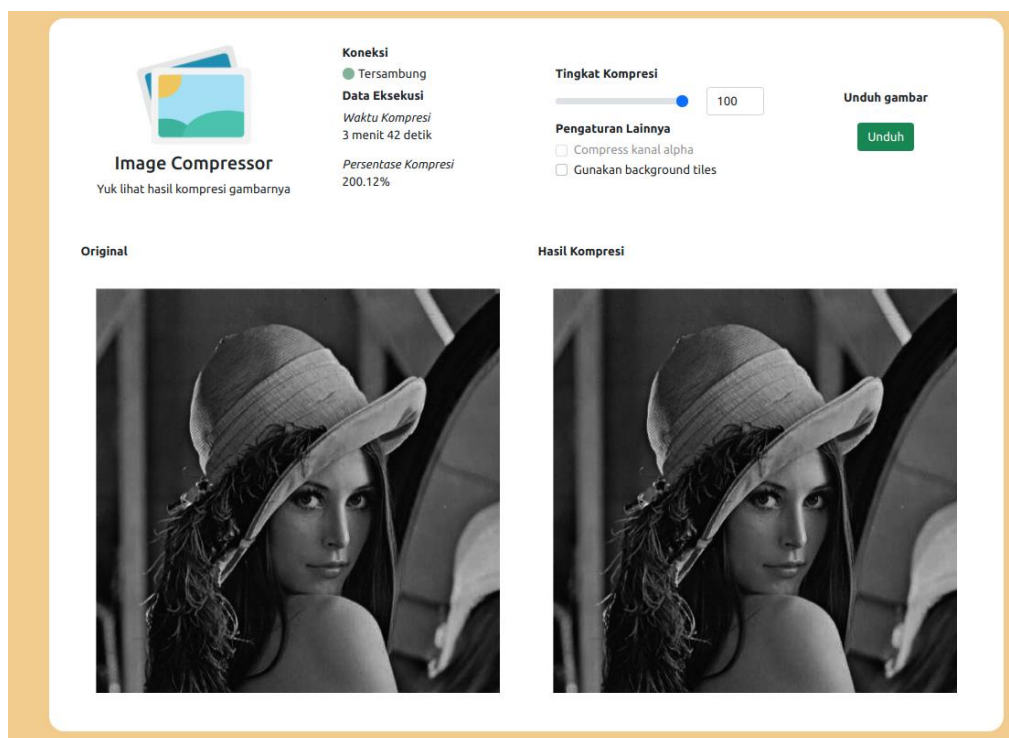
2. Gambar ukuran sedang grayscale berekstensi PNG

Ukuran pixel file adalah 850x850, berikut adalah gambar asli yang di-input:



Gambar 12 Kasus Uji 2

Gambar ini bertipe grayscale atau hanya memiliki kromatis hitam-putih. Hasil eksekusi menggunakan server Heroku, diperoleh 3 menit 42 detik.



Gambar 13 Hasil Eksekusi Kasus Uji 2

Waktu eksekusi yang diperoleh adalah 3 menit 42 detik, berikut adalah hasil kompresi 50, 25, 10, 5, 2



Gambar 14 Gambar original, 50, 25, 10, 5, 2 persen dekomposisi kasus uji 2

3. Gambar berukuran besar RGBA berekstensi png

Ukuran pixel file adalah 1024x1024, berikut adalah gambar asli yang di-input:



Gambar 15 Kasus Uji 3

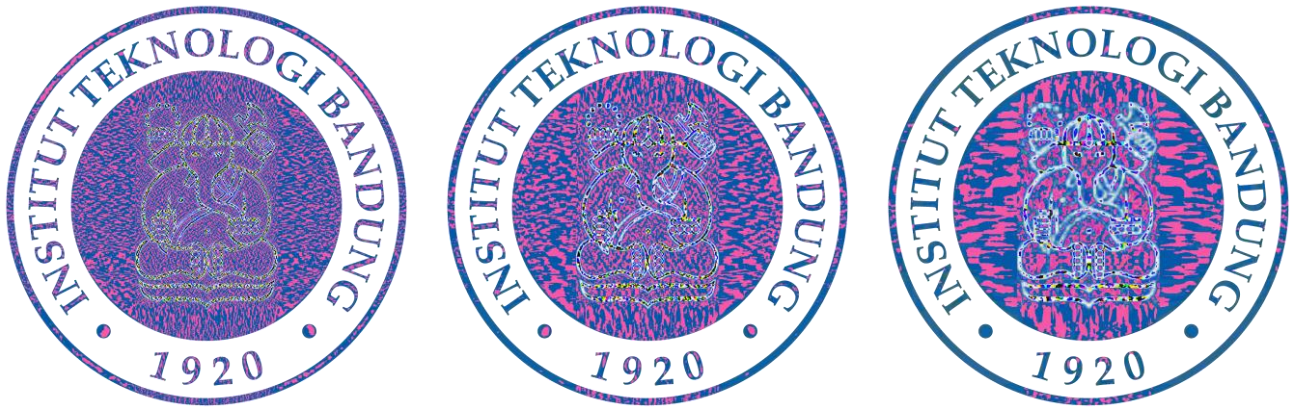
Hasil eksekusi diperoleh 5 menit 50 detik, tipe gambar berupa RGBA yang terpaletisasi (gambar indeks), sehingga ukuran gambar cenderung membesar, akan tetapi menurun seiring tingkat kompresi.



Gambar 16 Hasil Eksekusi Kasus Uji 3

Berikut adalah hasil kompresi 50, 25, 10, 5, 2 persen pada kasus uji





Gambar 17 Gambar original, 50, 25, 10, 5, 2 persen dekomposisi kasus uji 3

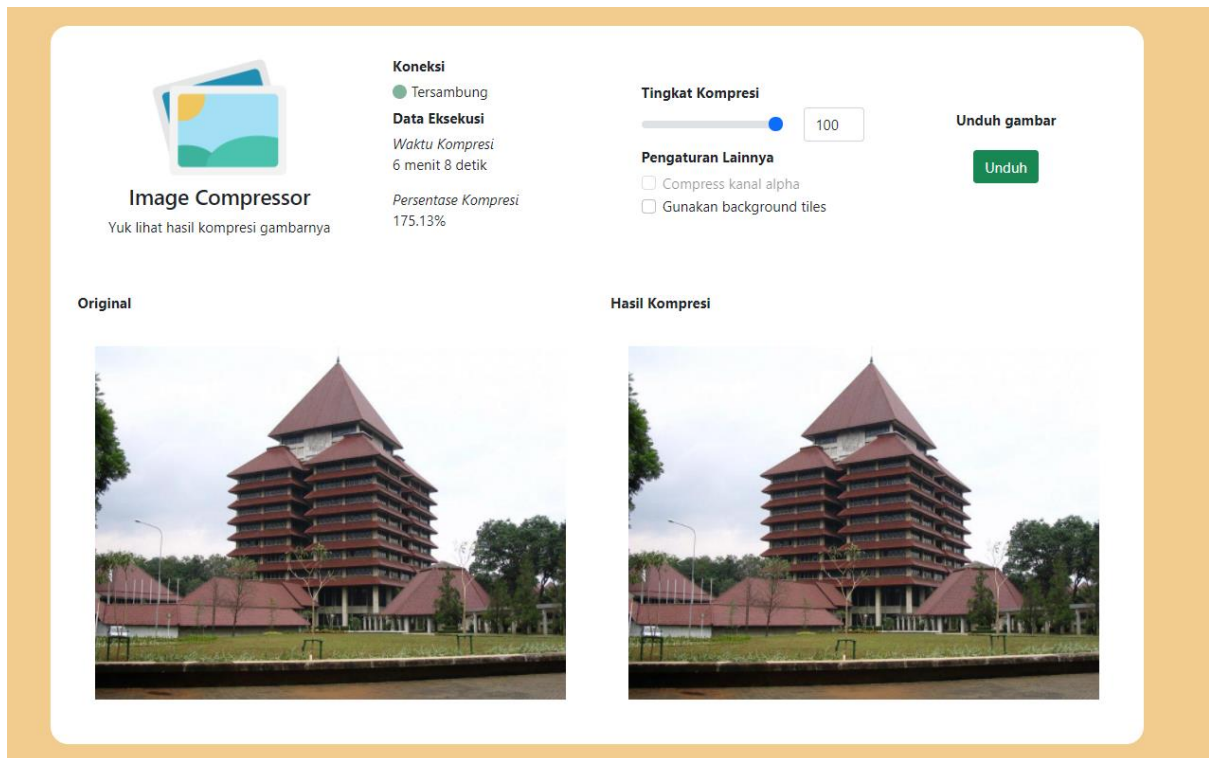
4. Gambar ukuran sedang RGB berekstensi JPG

Berikut adalah gambar asli dari kasus uji:



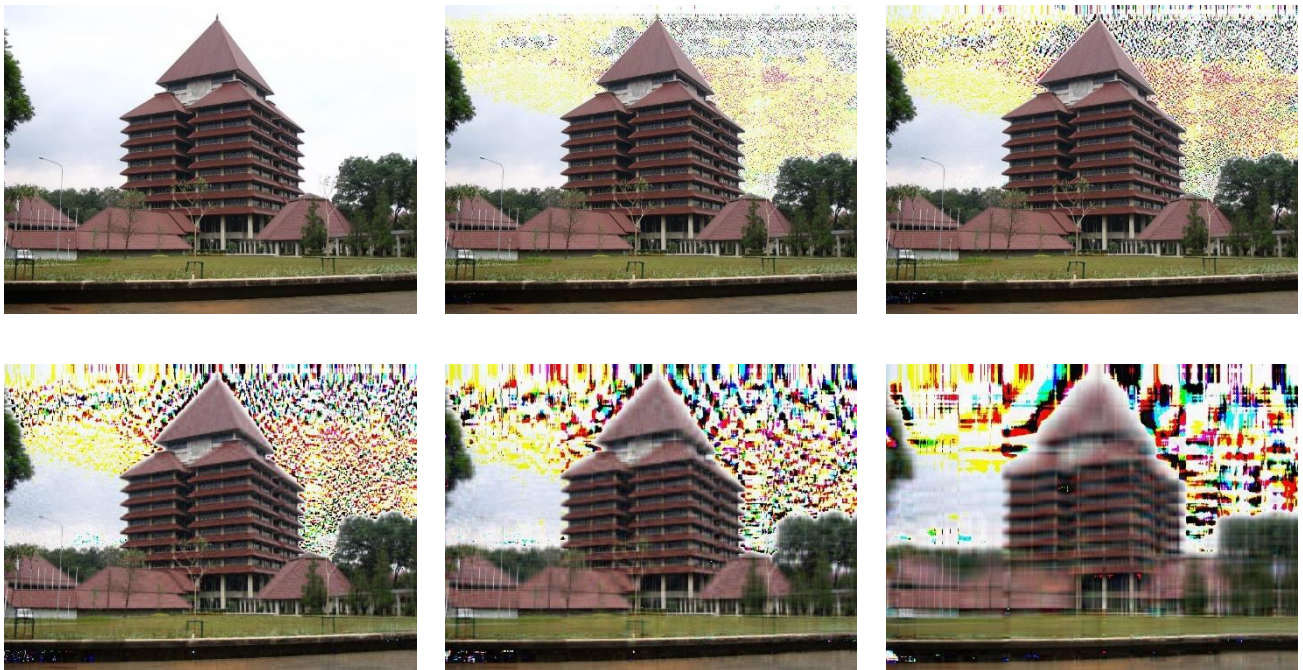
Gambar 18 Kasus Uji 4

Ukuran pixelnya adalah 768x576 pixel, dengan tipe gambar adalah RGB, waktu eksekusi yang diperoleh adalah 6 menit 8 detik.



Gambar 19 Hasil Eksekusi Kasus Uji 4

Hasil kompresi 50, 25, 10, 5, 2 persen diperoleh sebagai berikut



Gambar 20 Gambar original, 50, 25, 10, 5, 2 persen dekomposisi kasus uji 4

5. Gambar berukuran sedang RGB berekstensi JPG

Gambar input yang dipakai adalah sebagai berikut



Gambar 21 Kasus Uji 5

Gambar ini berformat JPG dengan ukuran pixel 800x588, tipe gambar adalah RGB dengan waktu eksekusi 2 menit 56 detik.




Image Compressor
Yuk lihat hasil kompresi gambarnya

Koneksi
● Tersambung

Data Eksekusi
Waktu Kompresi
2 menit 56 detik

Persentase Kompresi
173.63%

Tingkat Kompresi

100

Pengaturan Lainnya


☐ Compress kanal alpha

☐ Gunakan background tiles


Unduh gambar

Unduh

Original

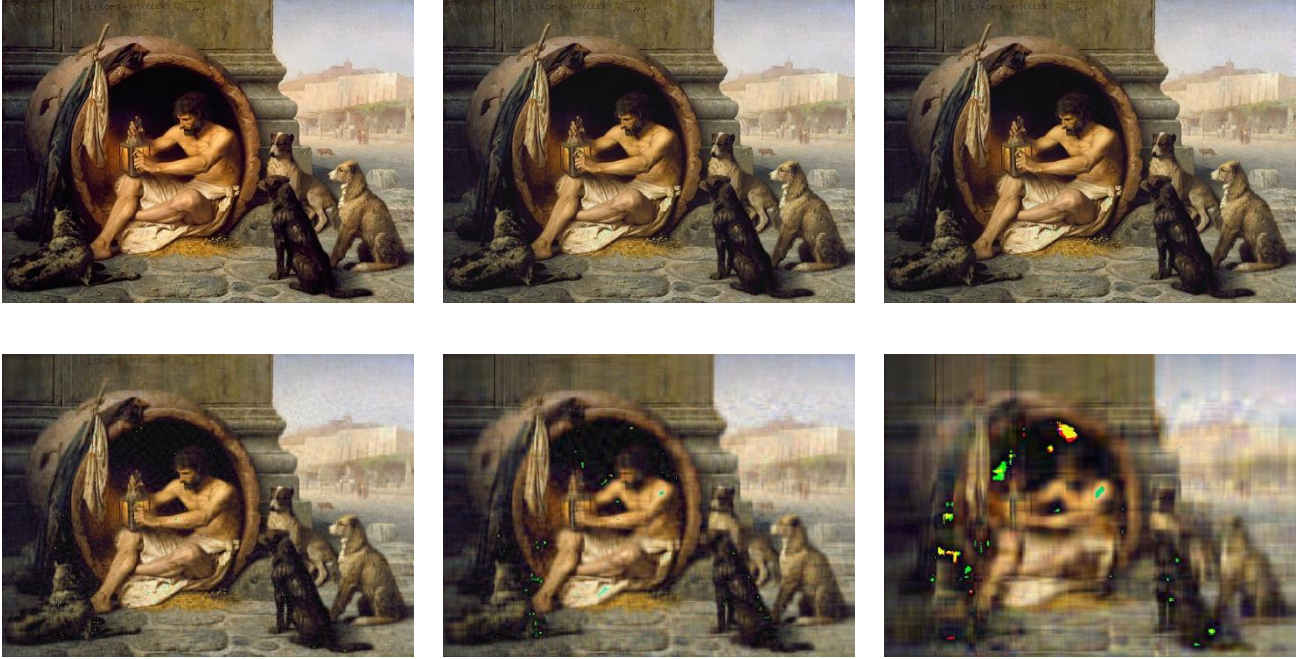


Hasil Kompresi



Gambar 22 Hasil Eksekusi Kasus Uji 5

Berikut adalah hasil kompresi 50, 25, 10, 5, 2 persen pada gambar input:



Gambar 23 Gambar original, 50, 25, 10, 5, 2 persen dekomposisi kasus uji 5

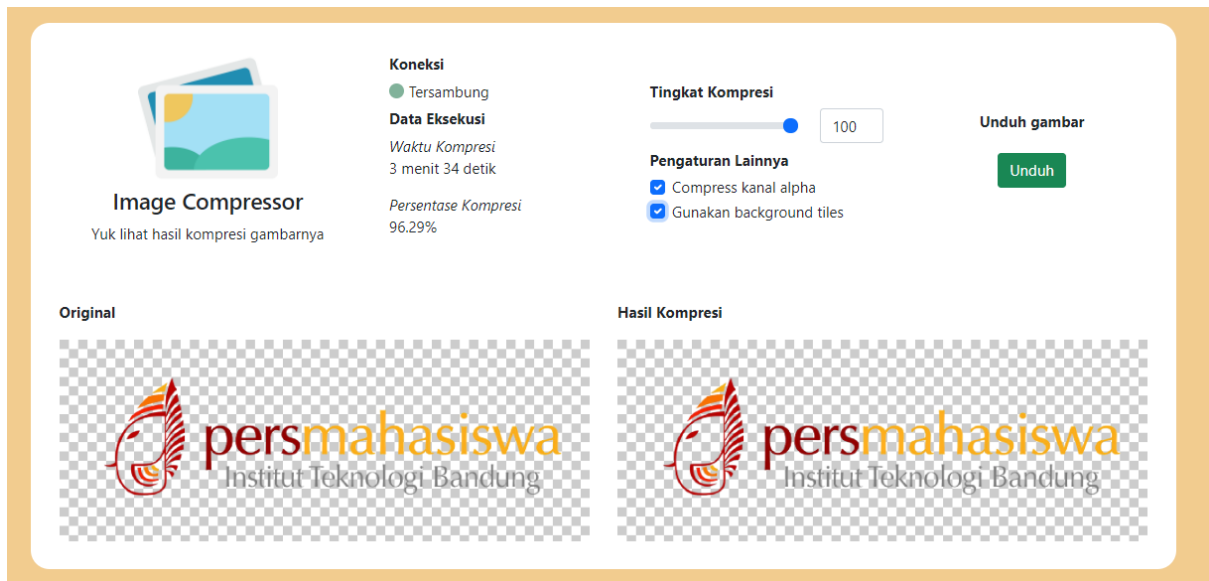
6. Gambar berukuran kecil RGBA berekstensi PNG

Gambar yang dijadikan input adalah sebagai berikut



Gambar 24 Kasus Uji 6

Gambar ini memiliki ukuran pixel 552x184, dengan tipe gambar RGBA dan berformat PNG. Waktu total eksekusi kompresi adalah 3 menit 34 detik,



Gambar 25 Hasil Eksekusi Kasus Uji 6

Berikut adalah hasil kompresi dengan tingkat kompresi 50, 25, 10, 5, dan 2 persen



Gambar 26 Gambar original, 50, 25, 10, 5, 2 persen dekomposisi kasus uji 6

D. Hasil Analisis

Berdasarkan keenam kasus uji diperoleh dekomposisi SVD yang cukup akurat, gambar hasil aproksimasi dekomposisi tidak jauh dengan input dari pengguna. Pemrosesan gambar juga bisa dilakukan dalam berbagai ekstensi dan tipe gambar. Kanal Alpha dari gambar dapat diikutsertakan dalam dekomposisi, membuat pengguna mudah dalam mengontrol sifat dekomposisi yang diinginkan.

Waktu eksekusi, seperti yang telah dipaparkan sebelumnya, sangat bergantung pada kecepatan server dan CPU. Meskipun demikian, kami menyadari bahwa waktu eksekusi dari program tidak cukup baik (orde yang didapatkan adalah ukuran menit), asumsi kami

adalah karena algoritma yang kami gunakan lebih cocok untuk mencari dekomposisi yang memerlukan keakuratan yang cukup baik.

Ukuran hasil file dari proses dekomposisi ini tidak konsisten, Ada beberapa gambar yang ukurannya menurun, ada juga yang justru malah membesar. Asumsi kami, hal ini disebabkan adanya penambahan palet warna pada gambar, meskipun rank matriksnya mengecil. Hal ini terutama terjadi pada gambar terpaletisasi (contoh: gambar logo ITB), yang mana preservasi jumlah palet tidak mudah dilakukan, jika menggunakan SVD. Gambar berekstensi JPG pun cenderung tidak menghasilkan kompresi yang ukurannya lebih kecil, asumsi kami, ini disebabkan modul PIL yang kami gunakan dalam program python untuk mengubah gambar menjadi matriks, mengakibatkan penambahan kanal yang tidak diperlukan (karena adanya konversi menjadi RGBA, kemudian kembali menjadi RGB).

BAB 5

KESIMPULAN, SARAN, DAN REFLEKSI

A. Kesimpulan

Kompresi gambar merupakan salah satu keuntungan dalam optimalisasi memori penyimpanan. Kompresi gambar dilakukan dengan hanya menyimpan informasi-informasi penting dari gambar sehingga tetap bisa dilihat dengan jelas walaupun ukuran memori dari gambar lebih kecil dari sebelumnya.

B. Saran

Sebagian besar masalah dalam kompresi gambar ini adalah efisiensi waktu dari algoritma yang dibuat terkadang sangat lambat. Oleh karena itu, diharapkan untuk pengembangan selanjutnya adalah optimalisasi algoritma tersebut sehingga efisiensinya lebih cepat.

C. Refleksi

Tugas besar ini membutuhkan eksplorasi yang lebih mendalam tentang pemfaktoran matriks dan penerapannya untuk kompresi gambar dengan metode SVD sehingga membutuhkan waktu yang cukup lama sebelum membuat algoritma yang sesuai untuk kompresi gambar.

DAFTAR PUSTAKA

- Anton, H., & Rorres, C. (2019). In *Elementary Linear Algebra: Applications Version*. essay, JOHN WILEY & SONS.
- Cullum, J. K., & Willoughby, R. A. (1996). A QL procedure for computing the eigenvalues of complex symmetric tridiagonal matrices. *SIAM Journal on Matrix Analysis and Applications*, 17(1), 83–109. <https://doi.org/10.1137/s0895479894137639>
- Munir, R. (2021). *Singular value decomposition - informatika.stei.itb.ac.id*. Singular value decomposition. Retrieved November 14, 2021, from <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-19b-Singular-value-decomposition.pdf>.

