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



Kelompok 7
Dzaky Fattan Rizqullah (13520003)
Bariza Haqi (13520018)
Jeremy S.O.N. Simbolon (13520042)

TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG

2021

BAB 1
DESKRIPSI MASALAH

Gambar adalah suatu hal yang sangat dibutuhkan pada dunia modern ini. Kita sering kali 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. Sering kali 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 *file*-nya dengan baik tanpa mempengaruhi kualitas gambar secara signifikan. Terdapat berbagai metode dan algoritma yang digunakan untuk kompresi gambar pada zaman modern ini.

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 Σ, dan transposisi dari matriks ortogonal V. Pada penugasan kali ini, penulis bermaksud untuk membuat *website* kompresi gambar sederhana dengan menggunakan algoritma SVD.

BAB 2
TEORI SINGKAT

A. Nilai Eigen dan Vektor Eigen

Nilai eigen λ dan vektor eigen $\boldsymbol{x}$ dari sebuah matriks A berturut-turut adalah nilai skalar dan vektor yang memenuhi persamaan berikut.

$$\boxed{A\boldsymbol{x} = \lambda\boldsymbol{x}}$$

Persamaan 1. Persamaan nilai eigen dan vektor eigen

Untuk menghitung nilai eigen λ dari matriks A, kita dapat menyelesaikan persamaan berikut.

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

Persamaan 2. Mencari nilai eigen dari matriks *A*

Nilai-nilai eigen yang diperoleh dari persamaan di atas dapat digunakan untuk mencari vektor eigen yang bersesuaian. Vektor eigen $\boldsymbol{x}$ adalah vektor tidak nol yang memenuhi persamaan berikut.

$$(\lambda I - A)\boldsymbol{x} = 0$$

Persamaan 3. Mencari vektor eigen dari matriks *A*

B. *Singular Value Decomposition* (SVD)

Ketika dihadapkan kepada sebuah matriks, kita dapat menganalisis matriks tersebut untuk mendapatkan nilai-nilai spesial yang membantu kita dalam mengelompokkan serta memanipulasi matriks tersebut. Salah satu dari manipulasi tersebut adalah proses dekomposisi matriks yang disebut *Singular Value Decomposition* (SVD). SVD adalah salah satu metode faktorisasi matriks menjadi submatriks penting U, Σ, dan V^T yang membantu kita merepresentasikan data di matriks awal. SVD mengikuti persamaan berikut.

$$A_{m\times n} = U_{m\times m}\,\Sigma_{m\times n}\,V^T_{n\times n}$$

Persamaan 4. Persamaan *Singular Value Decomposition* dari matriks *A*

Matriks U merupakan matriks yang kolomnya terdiri atas vektor-vektor ortonormal dari matriks AA^T. Matriks Σ merupakan matriks diagonal yang berisi akar dari nilai-nilai eigen tidak nol dari matriks U dan V yang terurut mengecil. Matriks V merupakan matriks yang kolomnya terdiri atas vektor-vektor ortonormal dari matriks A^TA.

Gambar 1. Ilustrasi *Singular Value Decomposition* dari matriks A

Hal yang menarik dari dekomposisi ini adalah data terpenting pada tiap matriks tersimpan di atas. Matriks U menyimpan informasi mengenai baris matriks A dan informasi terpentingnya berada di kolom pertama. Matriks Σ menyimpan nilai eigen yang terurut kepentingannya dari atas. Matriks V^T menyimpan informasi mengenai kolom matriks A dan informasi terpentingnya berada di baris pertama. Akibatnya, kita dapat menggunakan ketiga submatriks tersebut untuk membentuk matriks A' yang merupakan pendekatan dari matriks A.

Untuk membentuk matriks A', ditentukan terlebih dahulu sebuah nilai skalar k pada rentang 1 hingga banyaknya nilai *singular* A sebagai mode pendekatan. Selanjutnya, kita ambil k kolom pertama dari matriks U sebagai matriks U', submatriks persegi kiri atas berukuran $k \times k$ dari Σ sebagai matriks Σ′, dan k baris pertama dari matriks V^T sebagai matriks $V^{T\prime}$. Maka, matriks A' dapat dibentuk melalui persamaan berikut.

$$A'_{m\times n} = U'_{m\times k}\, \Sigma'_{k\times k}\, V^{T\prime}_{k\times n}$$

Persamaan 5. Membentuk matriks A'

Dapat kita perhatikan bahwa matriks A' yang terbentuk memiliki ukuran yang sama dengan matriks A, tetapi memerlukan penyimpanan nilai matriks yang lebih sedikit. Fakta ini dapat kita manfaatkan untuk merealisasikan sebuah algoritma kompresi gambar dengan pendekatan SVD.

BAB 3
IMPLEMENTASI PROGRAM

A. *Frontend*

Untuk bagian Frontend, kami menggunakan framework Vue. Untuk komponennya terdiri dari App.vue yang merupakan bagian utama dari aplikasinya, Header untuk bagian headernya, Main untuk bagian utama dalam pemrosesannya dan Footer untuk bagian footernya. Untuk pengembangan aplikasi Vuenya kami menggunakan Webpack. Image akan diupload dalam bentuk base64 kemudian menginput besar compressnya dan diperlihatkan hasil compressnya yang bisa didownload.

B. *Backend*

Untuk bagian Backend, kami menggunakan framework Flask yang berdasarkan Python, dengan demikian Backend dapat digunakan untuk menjalankan kompresi setelah menerima input image hanya dengan meng-import `img_compress.py`. Mekanisme penyimpanan image sementara menggunakan `image.json` dan menggunakan modul `json-server`. Image diterima di backend dalam bentuk dictionary berisi file image yang di-convert menjadi base64, nilai compression rate, dan nama file. Image dalam base64 ini di-convert kembali menjadi image file dan selanjutnya di pass dalam fungsi `compress()` beserta nilai compression rate-nya untuk dilakukan kompresi. Hasil kompresi kemudian di-save secara local di `src/backend/img/` sebagai `image.jpg` dan dapat diakses dan ditampilkan dari frontend dengan mengakses link `http://localhost:5000/compress`.

C. Algoritma Kompresi

Algoritma kompresi gambar terdiri atas dua modul, yakni `img_compress.py` serta `svd_matrix.py`. Modul `img_compress.py` terdiri atas dua buah fungsi, yakni compress dan recompose. Fungsi `compress` menerima sebuah objek gambar dari *backend*, memisahkan kanal gambar berwarna merah, hijau, dan biru, memanggil fungsi `decompose` dan modul `svd_matrix.py`, menggabungkan kembali hasil dekomposisi kanal warna menggunakan fungsi `recompose`, menggabungkan kembali ketiga kanal warna ke dalam satu objek gambar, dan mengembalikannya ke *backend*. Fungsi `recompose` menerima *tuple* yang terdiri atas matriks ortogonal kiri, matriks *singular*, dan matriks ortogonal kanan, mengambil potongan matriks sesuai permintaan *backend*, menggabungkannya, dan mengembalikan hasil skala ulangnya.

Modul `svd_matrix.py` terdiri atas enam buah fungsi, yakni `decompose`, `eigenvalue`, `solve_homogeneous`, `ortho_singular_left`, `ortho_diagonal`, dan `ortho_singular_right`. Fungsi `decompose` menerima sebuah matriks dan mengembalikan hasil dekomposisinya menggunakan metode *singular value decomposition* yang melibatkan fungsi `ortho_singular_left`, `ortho_diagonal`, dan `ortho_singular_right` serta mengembalikan *tuple* yang terdiri atas ketiga matriks tersebut. Fungsi `eigenvalue` menerima sebuah matriks dan mengembalikan *array* yang beranggotakan pendekatan atas nilai-nilai eigen matriks tersebut menggunakan dekomposisi QR. Fungsi `solve_homogeneous` menerima sebuah matriks dan mengembalikan pendekatan atas ruang *null* dari matriks tersebut menggunakan algoritma SVD yang telah ada. Fungsi `ortho_singular_left` menerima sebuah matriks dan mengembalikan matriks persegi berukuran banyak baris matriks masukan yang kolom-kolomnya berisikan vektor eigen ternormalisasi yang bersesuaian dengan nilai eigen

matriks tersebut. Fungsi `ortho_diagonal` menerima sebuah matriks dan mengembalikan matriks diagonal dengan ukuran yang sama dengan matriks masukan yang diagonal utamanya berisi nilai-nilai *singular* dari matriks tersebut. Fungsi `ortho_singular_right` menerima sebuah matriks dan mengembalikan hasil transposisi matriks persegi berukuran banyak kolom matriks masukan yang kolom-kolomnya berisikan vektor eigen ternormalisasi yang bersesuaian dengan nilai eigen matriks tersebut. Matriks hasil ini diperoleh menggunakan hasil balikan dari `ortho_singular_left` dan `ortho_diagonal`.

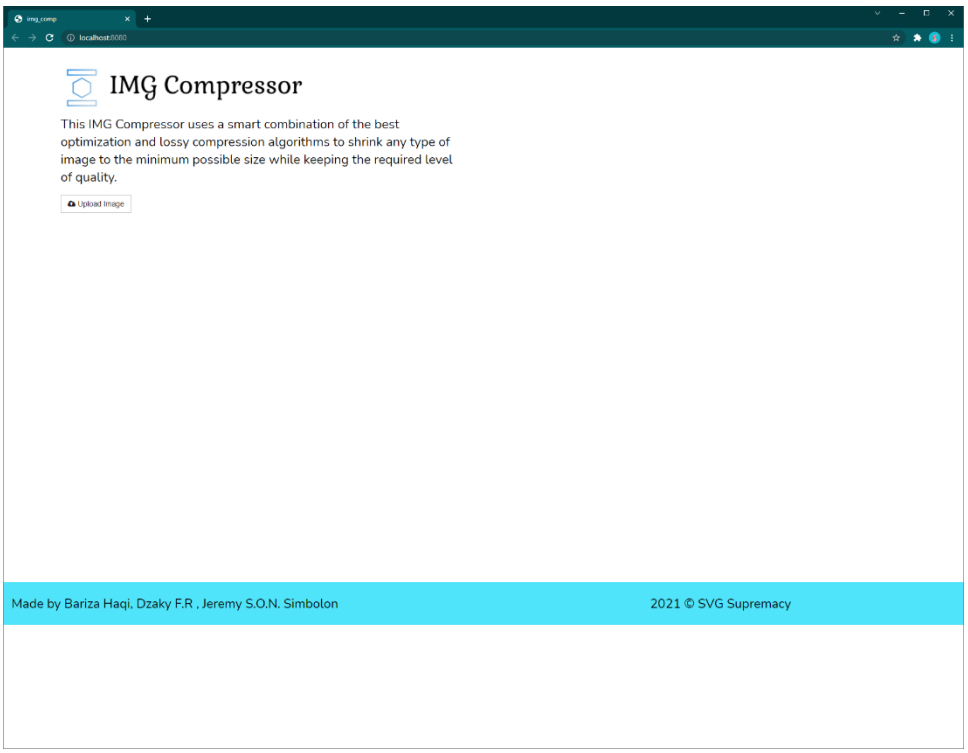
BAB 4
EKSPERIMEN

A. *Setup*

Eksperimen dimulai dengan men-download .zip dari repository. Selanjutnya mengikuti tutorial yang diberikan pada `README.md`, yang melingkupi instalasi dependencies, pengecekan apakah `pip` sudah ter-install, kemudian menjalankan API, server, dan frontendnya dalam cmd yang terpisah.

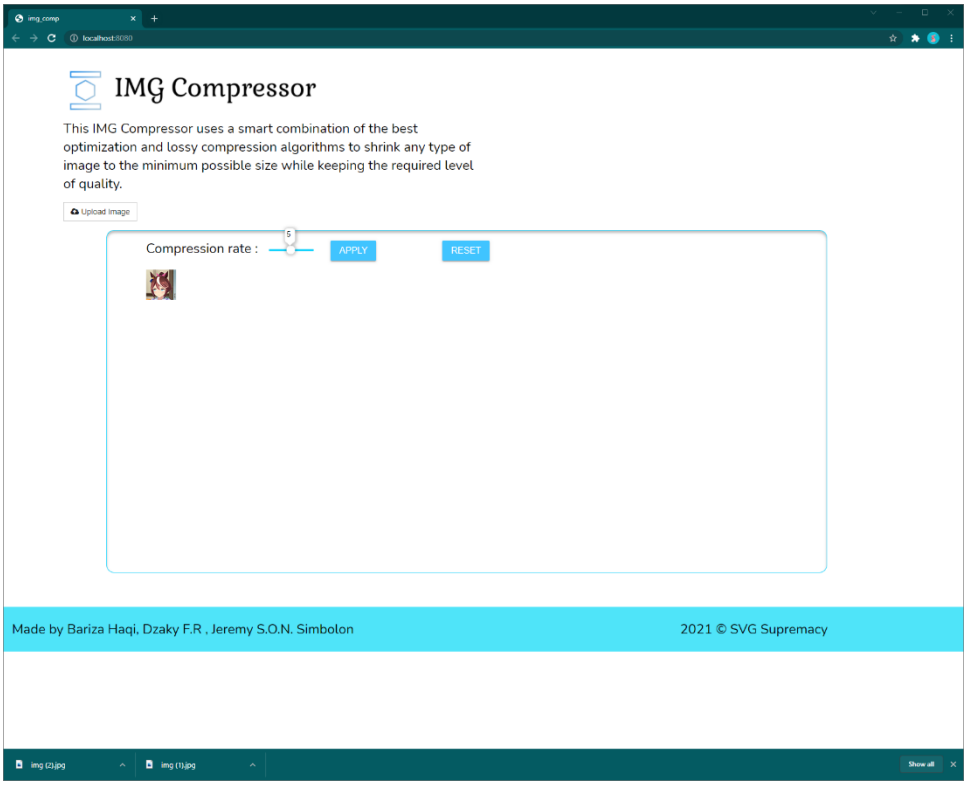
B. *Penggunaan Website*

Bila setup pada `README.md` dijalankan dengan benar, maka website sudah akan dapat diakses pada link http://localhost:8080/. Tampilan antarmuka web adalah sebagai berikut (tampilan in-development, hasil akhir bisa saja terdapat sedikit perubahan pada antarmuka).



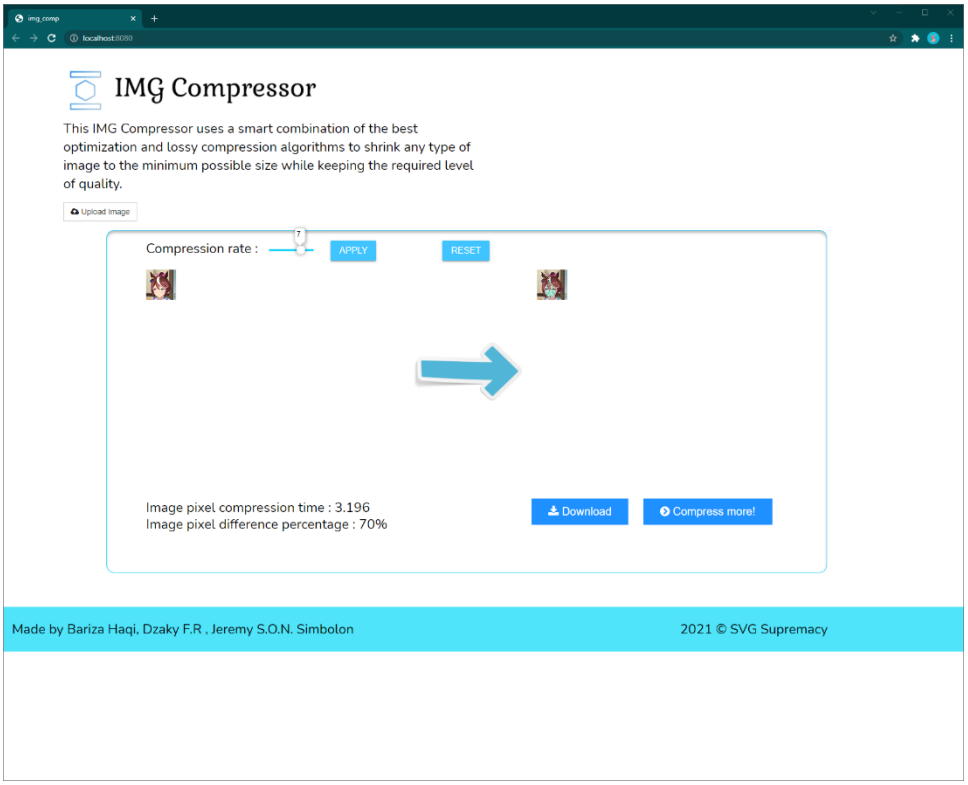
Gambar 2. Tampilan antarmuka web image compressor (in-dev)

Website menerima input gambar (hanya menerima format .jpg) dengan klik tombol `Upload Image`. Selanjutnya akan muncul tampilan image yang akan di-compress beserta slider untuk menentukan nilai kompresi dan tombol `APPLY`, serta terdapat pula tombol `RESET` yang fungsinya sudah dijelaskan pada `README.md`.



Gambar 3. Antarmuka setelah gambar di-upload (in-dev)

Geser slider untuk menentukan tingkat kompresi gambar. Nilai kompresi antara 1-10 yang artinya kompresi antara 10% - 100%. Setelah menentukan nilai kompresi, klik tombol `APPLY` dan tunggu hasilnya.



Gambar 4. Antarmuka setelah kompresi selesai (in-dev)

Gambar hasil kompresi dapat di-download dengan klik tombol download dan akan langsung tersimpan dengan nama file “img.jpg”. Untuk kompresi gambar selanjutnya dapat dilakukan dengan klik tombol “Compress more!” dan web akan direload kembali seperti semula.

Ada kalanya Web tidak mengupdate gambar hasil kompresi setelah menekan “Compress more!”. Hal ini disebabkan gagalnya `image.json` dibersihkan, yang bisa terjadi bila pengguna melakukan reload halaman tanpa klik tombol `RESET` (bila tombol tersebut di-klik, akan memunculkan notis bahwa halaman siap di-reload). Solusinya ialah dengan membuka file `image.json` yang terletak pada direktori `src/frontend/img/` dan menghapus segala hal yang berada di dalam square bracket yang merupakan atribut `”image”` .

BAB 5
KESIMPULAN, SARAN, DAN REFLEKSI

A. Kesimpulan

Transmisi informasi, khususnya gambar, merupakan salah satu kegiatan yang tak dapat dilepaskan dari perkembangan dan peningkatan pemakaian internet di masa modern. Upaya kompresi data merupakan salah satu solusi yang dikemukakan untuk mengatasi masalah-masalah umu yang berkaitan dengan transmisi data. Dalam melakukan kompresi data, khususnya gambar, ada banyak metode yang tersedia yang dapat kita manfaatkan. Salah satu metode yang dapat kita gunakan yakni kompresi gambar menggunakan algoritma *singular value decomposition* (SVD). *Website* kompresi gambar yang telah dipaparkan sebelumnya dapat digunakan untuk memenuhi kebutuhan yang telah diuraikan sebelumnya.

B. Saran

Kendala utama yang dihadapi dalam pengembangan *website* ini adalah masalah pada performa algoritma yang memerlukan waktu lama seiring meningkatnya ukuran gambar. Pemilihan algoritma lain dalam melakukan dekomposisi SVD, khususnya perhitungan nilai eigen diharapkan dapat menyelesaikan kendala ini ke depannya.

C. Refleksi

Pengerjaan tugas ini secara umum berjalan dengan baik. Salah satu hambatan utama yang kami hadapi adalah tidak familiernya kami dengan kakas *backend* dan *frontend*. Akibatnya, proses pengerjaan kami cukup terhambat secara signifikan di awal pengerjaan. Walaupun begitu, kami tetap berhasil menuntaskan penugasan ini.

REFERENSI

Anton, H., Rorres, C. and Kaul, A., 2019. *Elementary Linear Algebra: Applications Version*. Hoboken, New Jersey: Wiley.

Baumann, T., 2020. *SVD-Demo: Image Compression*. [online] timbaumann.info. Available at: <https://timbaumann.info/svd-image-compression-demo/> [Accessed 5 Nov. 2021].

Mathews, B., 2014. *Image Compression using Singular Value Decomposition (SVD)*. [online] Available at: <https://www.math.utah.edu/~goller/F15_M2270/BradyMathews_SVDImage.pdf> [Accessed 5 Nov. 2021].

Rufflewind, 2009. *How to solve homogeneous linear equations with NumPy?* [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/1835246/how-to-solve-homogeneous-linear-equations-with-numpy>.

Wojdyło, J., n.d. *The QR Method for Finding Eigenvalues*. [online] Available at: <https://cstl-csm.semo.edu/jwojdylo/MA345/Chapter6/qrmethod/qrmethod.pdf> [Accessed 5 Nov. 2021].