

TUGAS BESAR 2
APLIKASI NILAI EIGEN DAN VEKTOR EIGEN DALAM KOMPRESI
GAMBAR

LAPORAN

Diajukan sebagai salah satu tugas mata kuliah
IF2123 Aljabar Linier dan Geometri pada Semester I
Tahun Akademik 2021-2022

Oleh

Amar Fadil	13520103
Owen Christian Wijaya	13520160
Fachry Dennis Herald	13520139



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

DAFTAR ISI

BAB I	3
BAB II.....	4
2.1 Abstraksi	4
2.2 Perkalian Matriks	4
2.3 Nilai Eigen dan Vektor Eigen	5
2.4 Matriks SVD	5
2.5 Citra/Gambar	6
2.6 Simultaneous Power Iteration	7
BAB III	8
3.1 Library	8
3.1.1 Algoritma EigenSolver.....	8
3.1.2 Algoritma SVDSolver.....	8
3.1.3 Algoritma CompressSVD	9
3.2 Back-End	11
3.3 Front-End.....	11
3.3.1 ReactJS	11
BAB IV	13
4.1 Percobaan 1: Kompresi Gambar Kecil	13
4.2 Percobaan 2: Kompresi Gambar Besar + Variasi Iterasi	14
4.3 Percobaan 3: Perbandingan Rasio Eigen	16
4.4 Percobaan 4: Layer Alpha	17
4.5 Percobaan 5: Gambar Greyscale Kotak.....	18
4.6 Percobaan 6: Gambar Resolusi 4K (3840 x 2160)	18
4.7 Kompresi menggunakan GPU	19
BAB V	20
5.1 Kesimpulan	20
5.2 Saran	20
5.3 Refleksi	20
REFERENSI	22

BAB I

DESKRIPSI MASALAH

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

TEORI SINGKAT

2.1 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 2.1.1 Contoh kompresi gambar dengan berbagai tingkatan (Sumber: *Understanding Compression in Digital Photography* (lifewire.com))

2.2 Perkalian Matriks

Perkalian matriks adalah operasi antara dua matriks yang menghasilkan sebuah matriks baru. Syarat perkalian dua matriks A dan B adalah jumlah kolom di matriks A sama dengan jumlah baris di matriks B. Perkalian matriks A berukuran $m \times n$ dan matriks B berukuran $n \times o$ akan menghasilkan matriks baru C berukuran $m \times o$, dengan elemen C_{mo} merupakan hasil *dot product* baris m matriks A dan kolom o matriks B.

Perkalian matriks tidak bersifat komutatif, berarti hasil perkalian matriks A dan B berbeda dengan hasil perkalian B dan A. Suatu matriks A dapat dikalikan dengan sebuah nilai skalar X, menghasilkan suatu matriks baru dengan elemennya merupakan hasil perkalian elemen A dengan nilai skalar X. Apabila matriks A adalah sebuah matriks orthogonal (kolom-kolom matriks tersebut saling

orthogonal satu sama lain), maka hasil perkalian transpos matriks A dengan matriks A akan menghasilkan sebuah matriks identitas.

2.3 Nilai Eigen dan Vektor Eigen

Nilai eigen (artinya “asli” dalam Bahasa Jerman) adalah nilai karakteristik dari sebuah matriks persegi. Nilai eigen adalah nilai yang menandakan bahwa hasil perkalian suatu matriks persegi A dengan sebuah vektor kolom X tertentu akan menghasilkan sebuah vektor kolom baru yang merupakan hasil kelipatan skalar dari kolom vektor X. Nilai skalar tersebut disebut dengan nilai eigen (λ), dan vektor kolom tersebut adalah vektor eigen yang berkorespondensi dengan nilai eigen tertentu.

$$AX = \lambda X$$

Operasi tersebut menyebabkan vektor kolom X menyusut atau memanjang dengan faktor λ dengan arah yang sama jika λ positif, dan arah berkebalikan dengan λ negatif.

Nilai-nilai eigen dari suatu matriks dapat dihasilkan dari hasil polynomial dari persamaan determinan yang diperoleh dari hasil pengurangan nilai λ dikalikan dengan matriks identitas dengan matriks A.

$$AX = \lambda X$$

$$IAX = \lambda IX$$

$$AX = \lambda IX$$

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

Supaya solusi dari persamaan diatas bukan nol, maka

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

Akan dicari nilai λ yang memenuhi persamaan diatas. Penyelesaian dapat diselesaikan dengan polinomial yaitu dengan mencari akar-akar penyelesaian persamaan.

2.4 Matriks SVD

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 2.4.1 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 2.4.2 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 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.

2.5 Citra/Gambar

Citra sering disebut juga gambar pada bidang dwimatra (2-D). Citra adalah sinyal dwimatra yang bersifat menerus (continue) yang dapat diamati oleh sistem visual manusia. Secara matematis, citra adalah fungsi dwimatra yang menyatakan intensitas cahaya pada bidang dwimatra. Citra digital direpresntasikan sebagai matriks berukuran $m \times n$ sebagai berikut.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \dots & \dots & \dots & \dots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix}$$

Gambar 2.5.1 Matriks Representasi Citra

$f(x,y)$ merupakan fungsi intensitas cahaya (*brightness*) pada titik (x,y) . Titik (x,y) merupakan koordinat pada bidang dwimatra. Setiap elemen matriks menyatakan sebuah *pixel* (picture element). Sebagai contoh, citra dengan resolusi 1200×1500 berarti memiliki $1200 \times 1500 \text{ pixel} = 1.800.000 \text{ pixel}$.

2.6 Simultaneous Power Iteration

Untuk mencari nilai eigen dan vektor eigen pada matriks yang ukurannya lebih kecil, kita mengkalkulasikannya secara bertahap, yaitu dengan menyelesaikan persamaan $\det(\lambda I - A) = 0$. setelah didapatkan nilai eigen (λ) selanjutnya kita dapat menentukan vektor eigen dengan menyulihkan nilai eigen tersebut ke dalam persamaan $\lambda I - A = 0$ lalu akan diselesaikan SPL yang terbentuk oleh persamaan tersebut dan barulah ditemukan vektor eigen dari suatu matriks. Bagaimana jika kita ingin mencari nilai eigen dan vektor eigen pada matriks yang lebih besar, seperti matriks representasi dari suatu gambar yang ukurannya bisa sangat besar, tentu dengan langkah-langkah yang telah disebutkan tadi akan membutuhkan waktu yang lama dan juga menyita ruang karena harus menyimpan hasil operasi-operasi matriks yang dilakukan. Alih-alih melakukan kalkulasi langkah demi langkah, kita dapat mencari nilai eigen dan vektor eigen sekaligus yaitu dengan Simultaneous Power Iteration.

Simultaneous Power Iteration, atau dikenal juga dengan nama Orthogonal Iteration adalah metode versi lanjut yang memanfaatkan power iteration sebagai salah satu metode di dalam Aljabar Linear untuk mengaproksimasi nilai eigen dan vektor eigen dari suatu matriks secara simultan. Idennya adalah setiap vektor eigen selalu ortogonal dengan vektor eigen dominan lainnya. Dengan power iteration, akan dicari vektor eigen yang ortogonal dan dapat dijamin nilainya konvergen untuk vektor eigen lainnya. Iterasi dilakukan dengan mengalikan tiap vektor-vektor $\{q_1, q_2, \dots, q_t\}$ dan nilainya disimpan pada matriks Q . Setiap operasi ini, akan dilakukan normalisasi pada tiap vektor menggunakan dekomposisi QR .

Dekomposisi QR adalah salah satu metode iteratif untuk mencari semua nilai eigen dari suatu matriks. Misalkan A adalah matriks yang akan didekomposisi, dengan metode QR , hasil dekomposisi dari matriks A adalah matriks Q dan matriks R . Matriks Q adalah matriks orthogonal dan matriks R adalah matriks segitigaatas. Matriks orthogonal adalah matriks yang jika dibalikkan (invers) akan menghasilkan nilai transpose dari matriks tersebut.

BAB III

IMPLEMENTASI PROGRAM

3.1 Library

Untuk membantu dalam melakukan operasi dasar dan manipulasi matriks, digunakan modul NumPy dalam proses kalkulasi. Pada bagian *library* terdapat tiga algoritma utama yang digunakan dalam pemrosesan kompresi gambar dari awal hingga akhir. *Library* tersebut juga dapat digunakan sebagai modul untuk program Python lainnya dengan menginstal *script setup* yang telah disediakan melalui pip (*package installer* for Python). Eksekusi program dapat didukung dengan akselerasi GPU via CUDA menggunakan backend JAX NumPy agar pemrosesan kompresi gambar menjadi lebih cepat.

3.1.1 Algoritma EigenSolver

Dikarenakan matriks yang diolah ukurannya besar, maka akan lebih efisien jika dilakukan kalkulasi nilai eigen dan vektor eigen dari matriks tersebut sekaligus. Langkah-langkahnya adalah sebagai berikut.

1. Misalkan matriks A adalah matriks persegi dengan ukuran $m \times m$ yang akan dicari nilai eigen dan vektor eigennya. Akan dibentuk suatu matriks Q_0 (Q awal) berisi nilai 0.5 yang ukurannya adalah $m \times m$.
2. Akan dilakukan iterasi sebanyak i kali. Dibentuk suatu matriks Z_i yang merupakan hasil perkalian matriks A dengan matriks Q_{i-1} . Matriks Z_i akan didekomposisi dengan metode QR sehingga menghasilkan matriks Q_i dan R_i . Iterasi dilakukan agar matriks Q_i dan R_i elemennya konvergen kepada suatu nilai yang nantinya akan menjadi nilai eigen dan vektor eigen. Iterasi dapat ditentukan di awal atau dapat diberikan batasan sehingga dapat berhenti ketika mencapai galat (epsilon) yang ditetapkan. Namun, pada program ini, digunakan batasan iterasi saja daripada galat.
3. Setelah iterasi selesai, akan didapatkan matriks Q dan R . Kolom-kolom matriks Q adalah vektor eigen dari matriks A , sementara itu diagonal dari matriks R yang merupakan matriks segitiga atas adalah nilai eigen dari matriks A . Tiap nilai eigen dan vektor eigen berkorespondensi pada kolom yang sama.

3.1.2 Algoritma SVDSolver

Fungsi SVD ini memanfaatkan *orthogonal iteration* dalam mencari vektor eigen. Untuk optimasi, hanya akan digunakan salah satu dari matriks. Matriks Q adalah matriks yang akan diiterasi. Matriks Q ini juga akan menjadi salah satu matriks ortogonal antara matriks U atau matriks V bergantung dari kondisi berikut:

Untuk matriks A berukuran $m \times n$,

- Jika $m > n$ maka matriks Q adalah matriks U
- Jika $m < n$ maka matriks Q adalah matriks V
- Jika $m = n$ maka matriks Q adalah matriks U dan juga matriks V

Jika salah satu nilai matriks singular baik kiri maupun kanan ditemukan, dan nilai singularnya diketahui, maka dapat ditemukan pula matriks singular yang lainnya. Hal tersebut didapatkan menggunakan properti:

$$A \times V = U \times S$$

$$V^{-1} = V^T = S^{-1} \times U^{-1} \times A = \frac{1}{S} \times U^T \times A$$

$$U = A \times V \times S^{-1} = A \times V \times \frac{1}{S}$$

Karena matriks S merupakan matriks diagonal, S^{-1} dapat dicari dengan mendapatkan hasil dari $\frac{1}{S}$ untuk setiap elemen diagonal dari S . U dan V merupakan matriks orthogonal, sehingga $U^{-1} = U^T$ dan $V^{-1} = V^T$.

Langkah-langkah (algoritma) :

1. Misalkan matriks A adalah matriks berukuran $m \times n$ yang akan didekomposisi. Akan dicari terlebih dahulu matriks singular kiri atau kanannya berdasarkan m dan n .
2. Lakukan orthogonal iteration pada matriks singular yang telah ditetapkan sehingga didapatkan matriks Q (vektor eigen) dan matriks R (matriks diagonal berisi nilai singular, yaitu nilai eigen yang telah diakar-kuadratkan).
3. Tentukan matriks U , nilai singular, dan matriks V^T berdasarkan penentuan awal dari ukuran matriks A .

3.1.3 Algoritma CompressSVD

Modul yang digunakan dalam pemrosesan gambar pada CompressSVD adalah Pillow (PIL).

Langkah-langkah (algoritma):

1. File gambar akan diakses terlebih dahulu dengan perintah open dan disimpan ke dalam suatu variabel, dengan module PIL informasi gambar disimpan dalam bentuk matriks. Akan diambil informasi mengenai format dan mode gambar dan disimpan ke dalam konstanta.
2. Atur banyak iterasi yang diinginkan untuk orthogonal iteration. Banyak iterasi akan berpengaruh kepada ketepatan nilai matriks, namun jika banyak iterasi tidak diatur, maka akan di-set secara default oleh algoritma menjadi 2.

3. Atur ukuran/dimensi pada gambar (*image scalling*) jika diperlukan. Apabila ingin mengubah dimensi gambar, maka lebar dan tinggi gambar akan di-*resize* sesuai *scale* yang ditetapkan dengan mengalikan nilai *scale* pada lebar dan tinggi gambar. Nilai *scale* berkisar antara 0 dan 1.
4. Gambar akan lebih cepat diproses jika gambar tersebut *landscape*. Pada gambar yang *potrait* akan dilakukan rotasi sebesar 90° sehingga gambar sekarang menjadi *landscape*.
5. Lakukan konversi mode gambar. Gambar yang memiliki mode tertentu akan dikonversi ke dalam dua mode, "RGB" atau "RGBA". Jika mode gambar yang terbaca adalah gambar yang memiliki mode "P" atau "PA" dan memiliki nilai transparansi maka akan dikonversi modenya menjadi mode "RGBA", selain mode "L" atau mode "L" dengan varian alpha maka akan dikonversi menjadi mode "RGB".
6. Matriks gambar akan dinormalisasi nilai tiap elemennya antara 0 sampai 1 dengan cara mengalikan tiap elemen pada matriks dengan $\frac{1}{255}$
7. Metadata dari suatu gambar akan disimpan kedalam suatu konstanta sebelum dilakukan kalkulasi. Jika transparansi dipertahankan, varian alpha tidak disertakan di dalam kalkulasi namun disimpan nilainya untuk digunakan kembali pada rekonstruksi gambar.
8. Elemen dari matriks gambar akan diratakan agar menjadi matriks yang berdimensi $height \times (width \times band_length)$
9. Lakukan proses dekomposisi SVD dengan memanggil fungsi SVDSolver sehingga matriks berhasil dipecah menjadi matriks U, matriks sigma, dan matriks V.
10. Hitung nilai k yang ditetapkan, nilai k digunakan untuk mereduksi matriks pada saat rekonstruksi matriks dengan k adalah banyaknya nilai eigen yang ingin digunakan.
11. Hitung rasio kompresi, yaitu aproksimasi ukuran matriks setelah dilakukan kompresi dibagi dengan ukuran matriks setelah dikompresi.
12. Rekonstruksi matriks dengan mengalikan matriks U, matriks sigma, dan matriks V yang tiap ukurannya dibatasi sebanyak k pada kolom atau baris yang bersesuaian.
13. Matriks hasil rekonstruksi dikembalikan bentuknya seperti semula. Apabila alpha dipertahankan, nilai alpha akan diambil dari gambar aslinya dengan fungsi *append*.
14. Denormalisasi elemen matriks agar nilai-nilainya kembali seperti bentuk semula (0 hingga 255). Kemudian atur tipe matriks kedalam bentuk array unsigned integer 8 bit.
15. Matriks dalam bentuk array dikonversi menjadi tipe bentukan modul PIL dengan mode gambar yang sesuai.
16. Apabila gambar asli tersebut *potrait* maka akan dirotasi kembali sebesar -90° agar kembali ke bentuk semula.

17. Apabila mode gambar tidak sesuai maka matriks gambar akan dikonversi ke semula. Jika telah sesuai maka proses kompresi berhasil dan fungsi mengembalikan gambar yang telah dikompresi, informasi rasio kompresi, dan informasi format gambar.

3.2 Back-End

3.2.1 Connexion

Connexion adalah library tambahan yang digunakan di Python untuk dapat membuat *factory* backend dari spesifikasi API dengan standar OAS 3.0. Program ini akan menggunakan Flask untuk hasil dari *factory* dan menerima *request* dan mengirimkan *response* ke bagian front-end. Flask akan menerima *request* dari front-end dalam bentuk tipe *multipart/form-data*. Setelah itu, *response* yang diterima akan diatur oleh Connexion sesuai spesifikasi sehingga menjadi bentuk data yang dapat diterima oleh backend. Apabila data yang diterima tidak sesuai spesifikasi, maka backend akan mengirimkan pesan error. Sebaliknya, jika data sudah benar, maka proses akan dimulai.

Program akan memasukkan file gambar ke dalam fungsi kompresi yang ada pada instance class CompressSVD yang sudah diinisialisasi di awal dijalanannya backend. Pada saat memanggil fungsi kompresi, program juga mulai menghitung waktu yang dibutuhkan untuk kompresi gambar. Setelah gambar selesai diproses, program akan memberikan respons balik ke front-end berupa gambar hasil kompresi, waktu kompresi dan persentase kompresi.

3.3 Front-End

3.3.1 ReactJS

ReactJS adalah salah satu *library* yang digunakan dalam pengembangan desain aplikasi web. ReactJS memanfaatkan sistem modularitas JavaScript dan fitur-fitur lainnya seperti *contexts* dan *hooks* untuk mempermudah pengembangan secara *front-end*. Menggunakan ReactJS, komponen-komponen dalam laman web dapat dibuat secara terpisah dan kemudian disatukan. Selain itu, fitur seperti *context* mampu mempermudah penyimpanan data secara global antar file. Komponen-komponen *custom* lainnya juga dapat dipasangkan menggunakan paket NodeJS, Selain menggunakan ReactJS, website juga dibangun menggunakan HTML5 dan CSS3.

Pada laman situs kompresi, pengguna akan diminta untuk mengunggah gambar yang ingin dikompres dengan format .png atau .jpeg. Pengguna dapat mengunggah gambar menggunakan tombol *Upload* yang ada, atau dengan menggunakan fitur *drag-and-drop*. Setelah itu, akan muncul halaman pengaturan kompresi dengan beberapa komponen:

- *Preview* dari gambar untuk memastikan gambar yang diunggah sudah sesuai keinginan

- *Slider* untuk menentukan rasio dari nilai Eigen. Rasio pada *slider* menentukan kualitas gambar akhir. Apabila *slider* mempunyai rasio rendah, maka hasil kompresi yang dihasilkan akan mempunyai kualitas kompresi rendah. Sebaliknya, rasio tinggi akan menghasilkan dengan kualitas kompresi yang lebih tinggi.
- *Input* skala resolusi gambar. Pengguna dapat memasukkan nilai dari 1 – 100 yang mewakili persentase kompresi gambar. Semakin tinggi angka yang dimasukkan, hasil kompresi akan mempunyai resolusi yang tinggi (angka 100 berarti resolusi gambar sama dengan resolusi gambar aslinya, hanya dengan kompresi kualitas). Semakin tinggi angka yang dimasukkan, waktu yang dibutuhkan untuk melakukan kompresi juga akan semakin lama.
- *Input* jumlah iterasi. Jumlah iterasi dilakukan untuk mendapatkan hasil matriks yang akurat.
- *Checkbox* untuk menjaga transparensi. Apabila dicentang, proses yang dilakukan akan memisahkan layer alpha (untuk transparensi) sehingga apabila gambar mempunyai unsur transparensi, unsur transparensi tidak akan terganggu.
- Tombol *Compress* untuk mengkompresi gambar apabila semua pilihan telah terpilih.

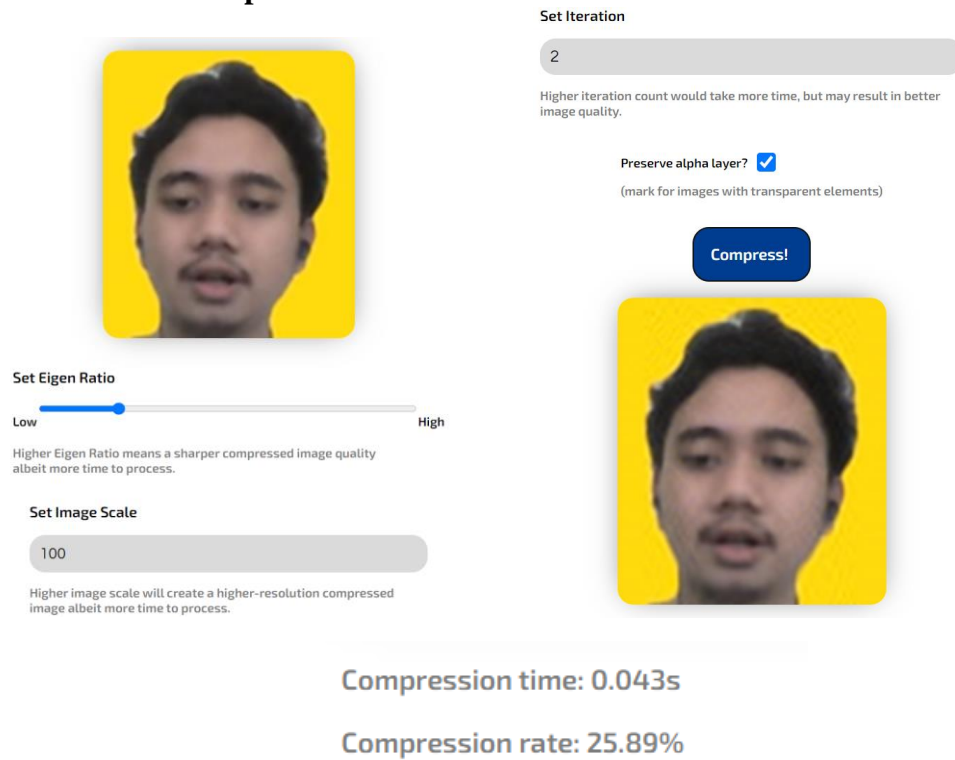
Selama menunggu gambar dikompres, akan muncul tulisan menunggu. Apabila gambar sudah selesai diproses, maka laman akan menunjukkan hasil kompresi beserta informasi mengenai waktu dan persentase kompresi.

BAB IV

EKSPERIMEN

Eksperimen dilakukan di device dengan OS Windows 10, CPU Intel i5-9400H

4.1 Percobaan 1: Kompresi Gambar Kecil



Set Iteration

2

Higher iteration count would take more time, but may result in better image quality.

Preserve alpha layer? ☒

(mark for images with transparent elements)

Compress!

Set Eigen Ratio

Low High

Higher Eigen Ratio means a sharper compressed image quality albeit more time to process.

Set Image Scale

100

Higher image scale will create a higher-resolution compressed image albeit more time to process.

Compression time: 0.043s

Compression rate: 25.89%

Rasio Eigen 20%, skala resolusi 100%, iterasi 2, layer alpha dipertahankan.

Waktu kompresi: 0.043s, persentase kompresi: 25.89%



Perbandingan gambar awal (kiri) dan gambar hasil (kanan)

4.2 Percobaan 2: Kompresi Gambar Besar + Variasi Iterasi

Set Eigen Ratio

Low High

Higher Eigen Ratio means a sharper compressed image quality albeit more time to process.

Set Image Scale

100

Higher image scale will create a higher-resolution compressed image albeit more time to process.

Set Iteration

2

Higher iteration count would take more time, but may result in better image quality.

Preserve alpha layer? ☒ (mark for images with transparent elements)

Compress!

Rasio Eigen 20%, skala resolusi 100%, iterasi 2, layer alpha dipertahankan.

Waktu kompresi: 1.351s, persentase kompresi 23.75%.



Compression time: 1.351s
Compression rate: 23.75%

Perbandingan gambar asli (atas) dan gambar hasil (bawah)

Set Iteration

20

Higher iteration count would take more time, but may result in better image quality.

Preserve alpha layer? ☒

(mark for images with transparent elements)

Compression time: 7.843s

Compression rate: 23.75%



Iterasi 20 kali, waktu kompresi 7.843s

Set Iteration

50

Higher iteration count would take more time, but may result in better image quality.

Preserve alpha layer? ☒

Compression time: 20.389s

Compression rate: 23.75%



Iterasi 50 kali, waktu kompresi 20.389s

Set Iteration

100

Higher iteration count would take more time, but may result in better image quality.

Preserve alpha layer? ☒

Compression time: 35.913s

Compression rate: 23.75%



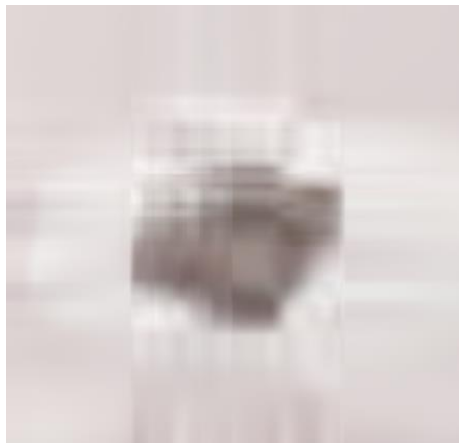
Iterasi 100 kali, waktu kompresi 35.913s.

Semakin tinggi banyaknya iterasi, kejernihan gambar meningkat namun waktu pemrosesan semakin lama.

4.3 Percobaan 3: Perbandingan Rasio Eigen



Gambar asli



Hasil rasio eigen tinggi (kiri) dibandingkan dengan rasio eigen rendah (kanan). Gambar dengan rasio eigen tinggi mempunyai kualitas kompresi yang lebih tinggi dari kualitas kompresi gambar dengan rasio eigen rendah. (Pengujian dilakukan di skala rasio resolusi 50%)

4.4 Percobaan 4: Layer Alpha



Gambar asli (400 x 372)



Compression time: 0.112s

Compression rate: 26.43%



Compression time: 0.120s

Compression rate: 24.86%

Kompresi Eigen Ratio rendah, kiri: layer alpha dipisah (waktu kompresi 0.112s), kanan: layer alpha tidak dipisah (waktu kompresi 0.120s).

Dapat dilihat bahwa pada gambar dengan layer alpha dipisah (layer transparansi), layer alpha ikut dikompresi sehingga menghasilkan artefak di bagian transparansi, sementara saat layer alpha dipisah, hanya bagian gambar yang bukan merupakan layer yang dikompresi.

4.5 Percobaan 5: Gambar Greyscale Kotak



Compression time: 0.065s

Compression rate: 40.699999999999996%

Ukuran gambar asli: 512 x 512 Waktu kompresi: 0.065 s, persentase kompresi: 40.7%

4.6 Percobaan 6: Gambar Resolusi 4K (3840 x 2160)

Set Eigen Ratio

Low  High

Higher Eigen Ratio means a sharper compressed image quality albeit more time to process.

Set Image Scale

100

Higher image scale will create a higher-resolution compressed image albeit more time to process.

Set Iteration

10

Higher iteration count would take more time, but may result in better image quality.

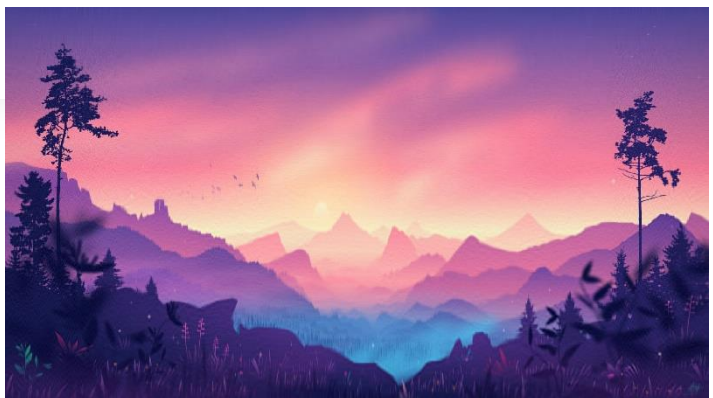
Preserve alpha layer? ☒

Preserve alpha layer? ☒

Compression time: 32.026s

Compression rate: 7.1499999999999995%

Iterasi 10 kali, waktu kompresi 32.026s,
persentase kompresi 7.15%



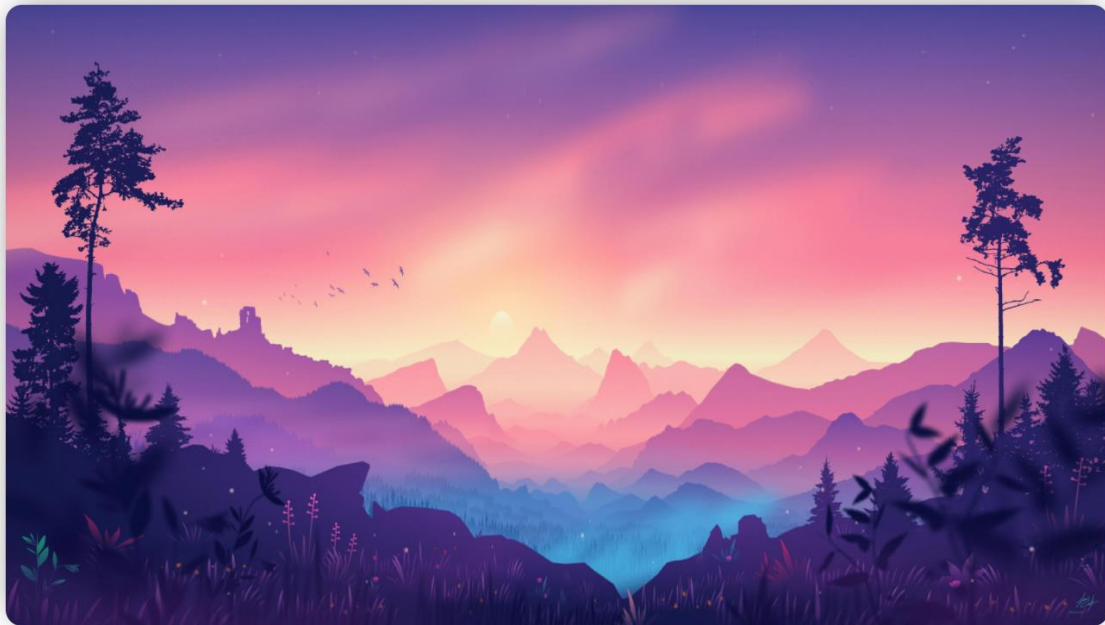
4.7 Kompresi menggunakan GPU

GPU yang digunakan adalah GPU *Google Colab* (Nvidia Tesla K80) yang disambungkan ke ngrok.io.



Compression time: 23.965s

Compression rate: 23.75%



Compression time: 1.387s

Compression rate: 23.75%

Iterasi 10 kali, rasio Eigen 20%

Menggunakan CPU (atas): waktu kompresi 23.965s, persentase kompresi 23.75%

Menggunakan GPU (bawah): waktu kompresi 1.387s, persentase kompresi 23.75%

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

5.1 Kesimpulan

Berdasarkan implementasi program dan eksperimen yang telah dilakukan, didapatkan kesimpulan sebagai berikut.

1. Telah dibuat suatu program kompresi gambar dengan memanfaatkan algoritma SVD dalam bentuk website lokal sederhana.
2. Kompresi gambar memanfaatkan algoritma SVD dan berbasis penggunaan matriks sebagai sebuah struktur data untuk mendapatkan nilai-nilai yang dibutuhkan untuk melakukan kompresi.

5.2 Saran

Berdasarkan implementasi program dan eksperimen yang telah dilakukan, adapun saran dari penulis sebagai berikut.

1. Memaksimalkan *framework* yang ada. *Framework-framework* seperti ReactJS dan Swagger UI membantu pemrosesan data secara front-end dan back-end dan mempermudah akses data. Meskipun membutuhkan waktu belajar lagi, penggunaan *framework* membantu pengerjaan tugas di implementasi algoritma. Pelajari *framework* yang akan digunakan sebelum menggunakan *framework* tersebut, karena mungkin ada perbedaan cara penggunaan Bahasa di *framework* dan bahasa aslinya atau ada fitur-fitur lainnya yang bisa memudahkan pengerjaan (seperti *context* dan *states* di React, perbedaan atribut elemen HTML biasa dan elemen ReactHTML)
2. Mencari banyak sumber di luar sumber di kelas. Ada beberapa opsi algoritma yang dapat digunakan, namun tidak semuanya mempunyai efisiensi yang setara. Oleh karena itu, eksplorasi algoritma di luar kelas dan riset dari sumber-sumber yang ada di Internet sangat dianjurkan untuk membantu pengerjaan algoritma.
3. Biasakan melakukan *testing* terhadap beragam contoh input (misalnya input warna *greyscale* atau format yang berbeda). Dengan demikian, kelemahan-kelemahan program dapat dikenali dan solusi-solusi baru dapat dibuat.

5.3 Refleksi

Dalam proses pengerjaan tugas besar ini tentu kami menemukan beberapa kendala, namun dengan kendala tersebut kami mencari solusi untuk menghadapinya. Adapun hal-hal yang menjadi perhatian bagi kami sebagai berikut.

1. Dibutuhkan waktu yang cukup banyak untuk mencari algoritma yang tepat guna dan efisien untuk melakukan kompresi gambar. Oleh karena itu, penulis menyarankan mengeksplorasi sumber-sumber yang ada selain referensi di kelas dan mengimplementasikan algoritma yang dapat dieksekusi dengan baik.
2. Jangan terlalu memaksakan kehendak untuk menggunakan satu algoritma tertentu. Teruslah bereksplorasi mencari algoritma-algoritma baru yang mungkin lebih efisien dari algoritma yang sedang dipakai. Terus-teruslah juga mengecek algoritma yang telah dibuat sehingga algoritma yang dibuat dapat lebih efisien.
3. Biasakan membagi waktu yang proporsional antara pengerjaan satu proyek dengan proyek lainnya. Penulis menyarankan menyiapkan waktu khusus untuk melakukan riset atau perbaikan ke program per harinya, dengan melakukan *check-up* dan *testing*. Jangan lupa juga untuk beristirahat dan menjaga kesehatan diri.

REFERENSI

- Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., ... Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs (Version 0.2.5). Opgehaal van <http://github.com/google/jax>
- Kong, Q., Siau, T., Bayen, A., (2020). Python Programming and Numerical Methods. MIT
- “Nilai Eigen dan Vektor Eigen Bagian 1” by Rinaldi Munir
<https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-18- Nilai-Eigen-dan-Vektor-Eigen-Bagian1.pdf>
- “Singular Value Decomposition (SVD)” by Rinaldi Munir
<https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-19b-Singular-value-decomposition.pdf>
- “Pengantar Pengolahan Citra Bagian 1” by Rinaldi Munir
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2020-2021/01-Pengantar-Pengolahan-Citra-Bag1-2021.pdf>
- “Simple SVD Algorithms – Naive ways to calculate SVD” by Risto Hinno
<https://towardsdatascience.com/simple-svd-algorithms-13291ad2eef2>
- “Power Iteration – ML Wiki” by Alexey Grigorev
http://mlwiki.org/index.php/Power_Iteration