
Tugas Besar 2 IF2123 Aljabar Linier dan Geometri
Singular Value Decomposition
Semester I Tahun 2021/2022



Disusun oleh:	
Jaya Mangalo Soegeng Rahardjo	13520015
Christopher Jeffrey	13520055
Aldwin Hardi Swastia	13520167

Institut Teknologi Bandung
Sekolah Teknik Elektro dan Informatika

BAB I

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 1. Algoritma SVD

Matriks U adalah matriks yang kolomnya terdiri dari vektor eigen ortonormal dari matriks $A^T A$. 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 $A A^T$. 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 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.

BAB II

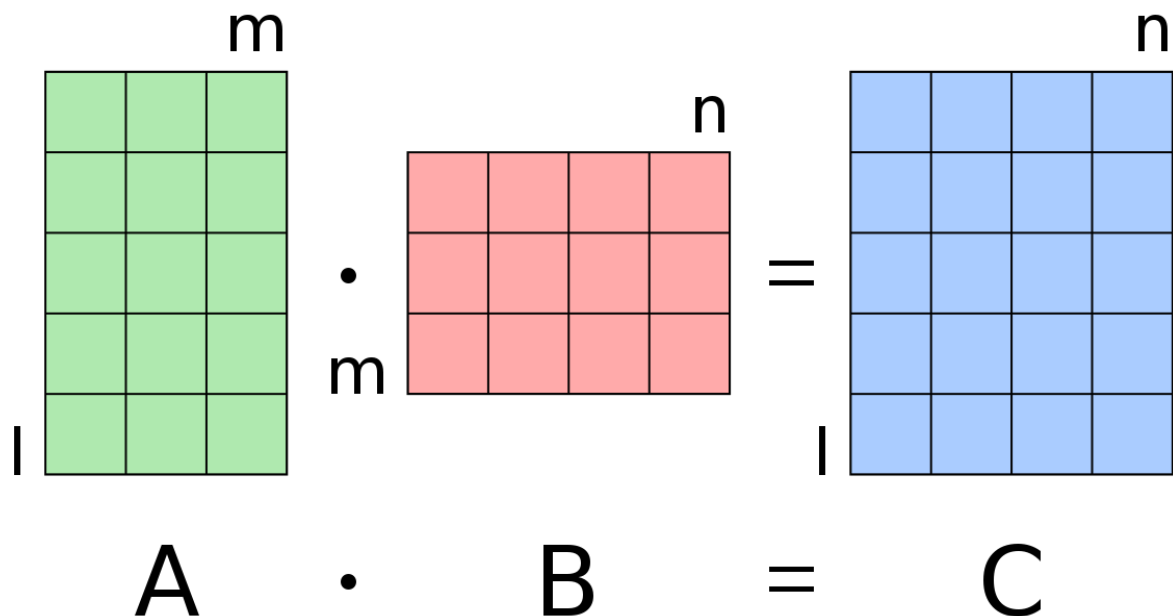
TEORI SINGKAT

A. Perkalian

Matriks

Perkalian matriks merupakan sebuah operasi binari fundamental dalam *linear algebra*. Perkalian matriks pertama kali ditemukan oleh sebuah ahli matematika dari perancis bernama Jacques Phillipe Marie Binet pada tahun 1812.

Perkalian matriks mengkalikan dua matriks menjadi sebuah matriks baru. Kedua matriks yang dikalikan tersebut perlu memenuhi syarat: Banyak kolom pada matriks pertama sama dengan banyak baris pada kolom kedua. Matriks hasil perkalian disebut matriks produk dan memiliki dimensi $M \times N$ dimana M adalah banyak baris dari matriks pertama dan N adalah banyak kolom dari matriks kedua.



B. Vektor Eigen dan Nilai Eigen

Vektor eigen adalah sebuah vektor tidak 0 dimana jika ia ditransformasi linear oleh sebuah matriks tertentu, vektor tersebut hanya akan berubah secara skalar dan akan tetap di *span*-nya.

Nilai eigen, biasanya dinotasikan oleh simbol lambda (λ), menyatakan seberapa besar dan arah dari vektor eigen akan meregang bila ditransformasi.

Untuk mendapatkan nilai eigen, dapat didapatkan dengan memasukan rumus tersebut dimana lambda adalah nilai eigen, I adalah matriks identitas, dan A adalah matriks transformasi linear.

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

Setelah mendapatkan nilai eigen, vektor eigen dapat dicari dengan menggunakan rumus tersebut dimana lambda adalah nilai eigen, I adalah matriks identitas, A adalah matriks transformasi linear, dan x adalah vektor eigen. Perlu dicatat apabila ada beberapa nilai eigen, perlu untuk mencari vektor eigen dari tiap lambda tersebut.

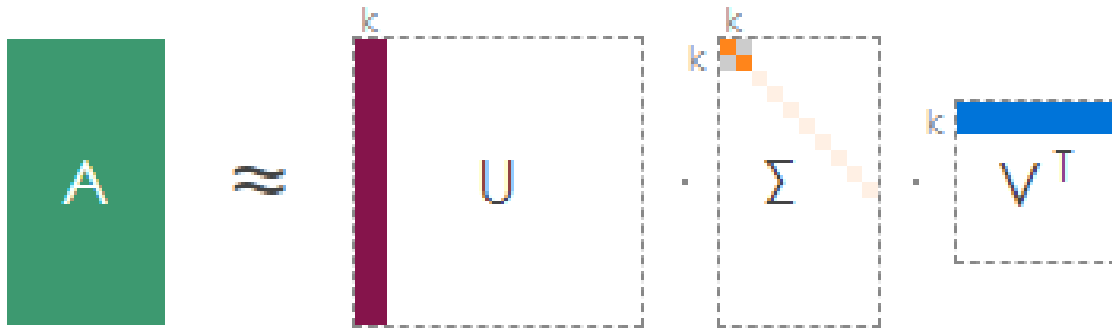
$$(\lambda I - A)\mathbf{x} = 0$$

Banyaknya dan nilai dari vektor eigen akan berbeda untuk tiap matriks transformasi linear, contohnya matriks transformasi linear skalar akan memiliki vektor eigen sebanyak infinit sementara matriks transformasi linear rotasi tidak akan memiliki vektor eigen.

C. Singular Value Decomposition

Singular Value Decomposition (SVD) adalah sebuah metode yang sering digunakan untuk mengecilkan data yang disimpan dalam matriks berdimensi sangat besar menjadi data yang lebih kecil. Metode ini sering dipakai untuk kompresi gambar, mengolah gambar, *machine learning*, *computer vision*, *digital watermarking*, dll.

SVD pertama dilakukan dengan memecah data atau matriks kita menjadi 3 bagian. Dimana A adalah matriks data original, U adalah *left singular vectors*, Σ adalah *singular values*, dan V^T adalah transpose dari *right singular vectors*.



Untuk mendapatkan U dan Σ pertama matriks A harus dikalikan dengan matriks transposenya seperti rumus berikut.

$$X = A \cdot A^T$$

U merupakan matriks yang diisi oleh vektor eigen X , sedangkan Σ terisi dengan *singular values* atau akar dari nilai eigen matriks X yang diurutkan dari besar ke kecil. Singular values tersebut diletakkan di diagonal matriks Σ .

Cara mendapatkan V mirip dengan cara mendapatkan U , bedanya adalah matriks dasarnya:

$$Y = A^T \cdot A$$

V merupakan matriks yang diisi oleh vektor eigen Y sedangkan V^T adalah transpos dari V .

Tetapi di aplikasi SVD, tidak digunakan SVD secara full karena memakan banyak sekali memori. Dalam aplikasinya, biasanya diambil U_t , Σ_t , V_t^T , yaitu $U \Sigma V^T$ yang hanya diambil k baris atau kolom pertama. Besarnya k ini dapat diatur dengan k yang besar akan menghasilkan data yang lebih mirip ke original data sedangkan k yang lebih kecil akan menggunakan memori yang lebih sedikit.

BAB III

IMPLEMENTASI PUSTAKA DAN PROGRAM

A. Gambaran Umum

Front-end menggunakan HTML, CSS, dan JavaScript *vanilla*, kami tidak menggunakan framework seperti react karena tidak diperlukan design *web* yang terlalu *advanced*.

Back-end menggunakan python dengan library: numPy, flask, cv2. Tugas yang perlu ditangani oleh backend adalah menyimpan file gambar yang di upload oleh user, melakukan kompresi terhadap gambar tersebut, dan menyimpan hasil dari kompresi tersebut agar bisa ditampilkan kembali (atau di download) di front end.

Untuk melakukan penyimpanan gambar, kami menggunakan 'request' dari flask. Selanjutnya, gambar akan dilakukan kompresi terhadap gambar dengan menggunakan dua library utama cv2 (library openCV untuk python) dan numPy. Cv2 digunakan untuk mengolah gambar menjadi matriks numPy, dan juga untuk melakukan penyimpanan dari gambar yang sudah diolah. NumPy digunakan untuk melakukan manipulasi matriks numPy dari gambar.

B. Algoritma Kompresi

Algoritma ini kami implementasikan dengan sebuah fungsi bernama algoKompresi. AlgoKompresi menerima dua buah argumen, *filename* yaitu sebuah string lokasi gambar (relatif terhadap file python tempat menyimpan algoKompresi) dan *percentage* yaitu tingkat kompresi.

Langkah pertama, image di *filename* akan di baca dan disimpan datanya dalam sebuah numPy array. Langkah kedua adalah menentukan k, yaitu jumlah data dominan yang ingin digunakan. Untuk lebih bisa mengerti k, kita perlu tahu mengenai sifat dari SVD. SVD akan membagi suatu matriks (untuk kasus ini, matriks tersebut adalah gambar yang ingin kita kompresi) menjadi 3 buah matriks lainnya. Matriks pertama disebut left singular value(u), matriks kedua disebut singular value(s), matriks ketiga disebut right singular

value(vh). Keistimewaan dari SVD adalah, data dari masing-masing matriks tersebut sudah terurut berdasarkan *dominansi* mereka. Semakin dominan suatu data, semakin besar andil mereka dalam mewakili data matriks yang dikenai SVD. Untuk left singular value, kolom pertama lebih dominan daripada kolom kedua, kolom kedua lebih dominan daripada kolom ketiga, dan seterusnya. Untuk singular value (berupa matriks yang hanya diisi pada bagian diagonalnya, sedangkan bagian lainnya nol. Pada algoritma ini di representasikan dengan matriks satu dimensi), data pertama lebih dominan daripada kedua, dan seterusnya. Untuk right singular value, baris pertama lebih dominan pada baris kedua, dan seterusnya. Semakin banyak data dominan yang kita ambil (misalkan semua data), maka semakin tepat data kita di representasikan. Jika kita hanya mengambil bagian dominan dari masing-masing matriks (left singular value, singular value, right singular value), maka matriks yang di representasikan akan lebih tidak detail, dengan masing-masing data sangat memiliki nilai yang sangat dekat dengan data utama. Jadi peran k adalah menentukan banyak data dominan yang akan digunakan. Semakin kecil nilai k , semakin kecil matriks u , s , dan vh . Detail gambar akan berkurang, ukuran matriks u , s , dan vh semakin kecil, sehingga dapat kita sebut sebagai kompresi (meskipun ketika ketiga matriks tersebut dikalikan kembali, memiliki ukuran yang sama dengan sebelumnya. Ukuran yang dimaksud adalah jumlah kolom dan jumlah baris dari pixel gambar).

Dengan asumsi gambar yang diberikan terdiri dari 3 color channel (red green blue), algoritma tersebut dilakukan sebanyak tiga kali terhadap masing-masing color channel. Sebelum digabungkan, hasil dibatasi antara 0 dan 255, dengan asumsi nilai dari pixel di semua color channel tidak bisa bernilai negatif, dan color depth dari foto adalah 8-bit. Setelah dipisah setiap colour channel, lalu di terapkan algoritma pengompresan, masing-masing color channel akan digabungkan kembali menjadi satu. Dapat dikatakan kompresi matriks gambar selesai dilakukan.

Setelah selesai dikompresi, program akan melakukan saving matriks tersebut sebagai gambar ke lokasi yang telah kami tentukan dan mengembalikan (return) statistik kompresi (terutama lama waktu yang dibutuhkan untuk melakukan algo tersebut).

C. Algoritma SVD

Kita ingat kembali SVD, yaitu suatu cara atau proses untuk menghasilkan tiga buah matriks (sebut saja u , s , dan v_h , sama seperti sebelumnya) yang jika dikalikan akan menghasilkan matriks awal. Analoginya seperti proses faktorisasi pada suatu persamaan kuadrat di matematika. Untuk mendapatkan matriks u , s , dan v_h , SVD sangat bergantung pada eigenvalue dan eigenvector. Matriks u dan v dibuat dari eigenvector, sedangkan matriks s dibuat dari eigenvalue. Kita bisa mendapatkan nilai eigenvalue dan eigenvector dengan menggunakan QR decomposition (sebut saja QR). Mirip seperti SVD, QR adalah algoritma yang ‘memfaktorkan’ matriks menjadi dua buah matriks lain, kita sebut q dan r . Penggunaan matriks QR sebagai cara untuk mendapatkan eigenvector didasari dari sifat matriks q , yang terdiri atas matriks kolom yang masing-masing orthogonal satu sama lain dan memiliki panjang satu (vektor satuan). Ini sangat mirip dengan matriks u dan v_h , matriks dengan panjang unit satu dan saling orthogonal. Sehingga kita bisa menggunakan QR untuk mendapatkan u dan v_h . Selanjutnya untuk eigenvalue (untuk s), nilai dari matriks diagonal dari matriks r adalah eigen value. Sehingga untuk menghasilkan s , kita cukup memutlak-an nilai diagonal dari r dan mengakarnya.

Nilai q dari QR tidak sepenuhnya persis dengan nilai yang kita harapkan, namun dengan melakukan algoritma tersebut beberapa kali, akan dihasilkan nilai yang semakin dekat dengan yang kita harapkan. Sehingga terdapat iterasi dalam mengaplikasikan QR. Kami memilih untuk melakukan iterasi sebanyak lima kali, alasannya adalah nilai yang dihasilkan sudah cukup dekat dengan yang diharapkan (berdasarkan pengamatan dari gambar yang dihasilkan) dan relatif cepat (semakin banyak pengulangan, semakin lama waktu yang dibutuhkan).

Selanjutnya, nilai eigen value yang dihasilkan dari QR memanglah nilai eigen value yang benar, namun ada sedikit kekurangan. Meskipun nilai eigen benar, belum tentu itu nilai SVD yang diharapkan, karena jika dikalikan kembali, belum tentu menghasilkan matriks semula. Hal ini disebabkan karena sebenarnya ada dua eigenvalue (yang memiliki panjang 1 unit vektor), dibedakan oleh arahnya (dalam matematika dapat dilakukan dengan mengalikannya -1). Sehingga untuk menghasilkan nilai matriks yang sesuai ketika dikalikan kembali, ada sifat dari SVD yang perlu kita manfaatkan.

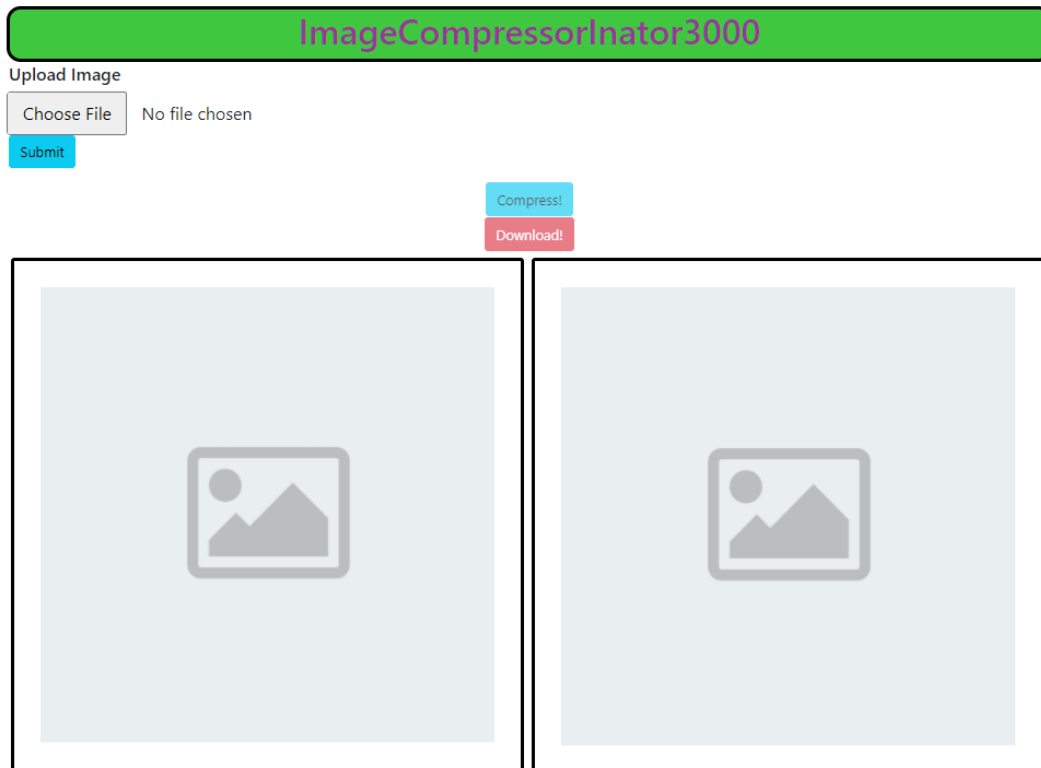
$$A v^t = \sigma u$$

Dengan menggunakan persamaan tersebut, kita dapat menghasilkan nilai v^h (dalam persamaan tersebut dilambangkan v^t) yang dapat menyelesaikan masalah tersebut. Salah satu keuntungan lain dari penerapan algoritma ini adalah, algoritma menjadi lebih cepat (karena tidak perlu melakukan penerapan algoritma QR dengan iterasi sebanyak dua kali). Setelah didapatkan nilai u , s , dan v^h , Selanjutnya adalah merapikan matriks agar lebih mudah digunakan. Ada beberapa nilai yang tidak akan digunakan, karena nilainya tidak akan terjangkau saat dilakukan perkalian kembali. Sehingga dalam penerapan algoritma SVD ini, juga dilakukan pemotongan data. Pemotongan data didasari nilai minimal dari baris dan kolom matriks input.

BAB IV

EKSPERIMEN

Ketika app.py initially di run:



Setelah foto disubmit:

ImageCompressorInator3000

Upload Image

Choose File

No file chosen


Submit

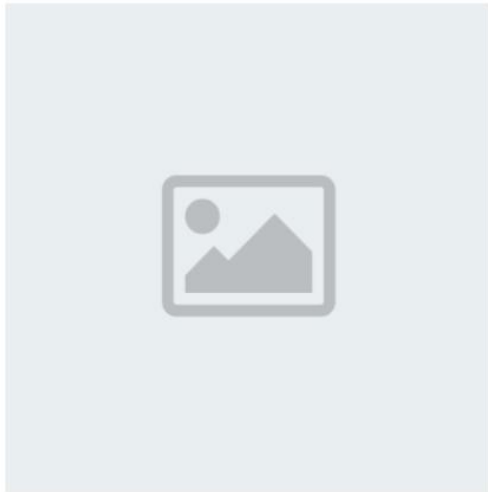
Compress Percentage

50

Compress!

Download!





Setelah foto dicompress:

ImageCompressorInator3000

Upload Image

Choose File

No file chosen

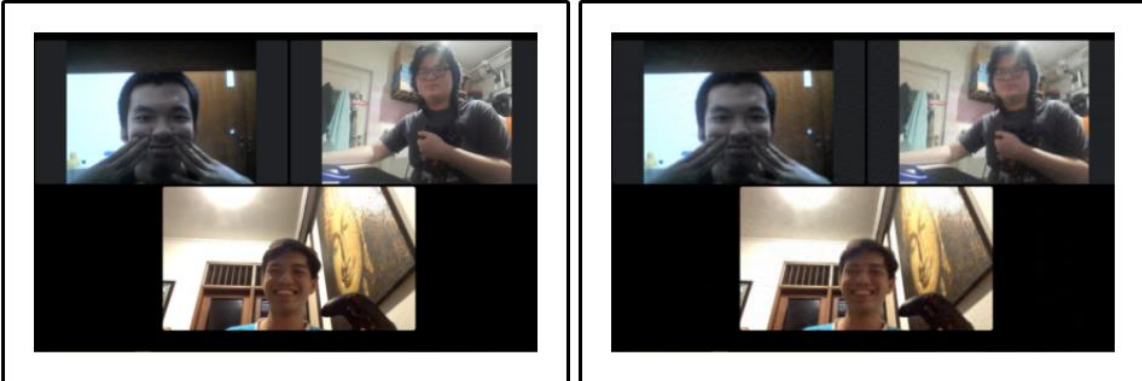
Submit

Compress Percentage

50

Compress!

Download!



Compress Log

image berhasil di read...
compressRatio = 0.12
kmax = 983
k = 72
ukuran asli = 1538395
ukuran compressed = 183528
splitting image and compressing...
processing channel 1 membutuhkan 3.120635986328125 detik
processing channel 2 membutuhkan 3.2075412273406982 detik
processing channel 3 membutuhkan 3.501063108444214 detik
combining...hasil kompresi akan disimpan dengan nama Foto_laporan_terbaik_burik_compressed.png
kompresi gambar berhasil
total waktu yang dibutuhkan untuk melakukan kompresi adalah 10.413895606994629 detik

Klarifikasi Teks:

Compress Log

image berhasil di read...

compressRatio = 0.5

kmax = 983

k = 301

ukuran asli = 1538395

ukuran compressed = 767249

splitting image and compressing...

processing channel 1 membutuhkan 3.0901172161102295 detik

processing channel 2 membutuhkan 3.1140003204345703 detik

processing channel 3 membutuhkan 3.0789997577667236 detik
combining...hasil kompresi akan disimpan dengan nama
Foto_laporan_terbaik_burik_compressed.png
kompresi gambar berhasil
total waktu yang dibutuhkan untuk melakukan kompresi adalah 9.799991130828857 detik

Note: Dengan kompresi 50%, size awal 1.56 MB dikompres menjadi 1.40 MB

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

Kompresi gambar dapat dilakukan dengan metode SVD (Singular Value Decomposition). Pertama-tama, nilai eigen dicari menggunakan metode QR Decomposition. Kemudian, matriks SVD dibentuk menggunakan nilai eigen yang telah didapat. Setelah itu, matriks singular kiri dan kanan dipotong hingga didapatkan matriks SVD yang lebih kecil sesuai dengan persentase kompresi. Lalu, ketiga matriks dikalikan kembali untuk mendapatkan gambar yang telah terkompresi.

Kami merasa bahwa kami kurang ambisius dalam mengerjakan tugas besar kali ini. Segala target dari pengerjaan tugas lebih mengarah pada keterselesain tugas dan tidak mengejar nilai-nilai bonus. Spesifikasi tugas dirasa masih terlalu abstrak karena SVD yang diajarkan di kelas berbeda dengan implementasi program. Hal ini menyebabkan waktu lebih banyak dihabiskan untuk mencari tahu cara implementasi algoritma. Saran kami, perbanyak referensi implementasi SVD. Selain itu,

Referensi:

- <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-14-Ruang-vektor-umum-Bagian1.pdf> (Diakses pada 10 November 2021, 18.13)
- https://www.youtube.com/watch?v=Z1RJmh_OqeA&t=2475s
- <https://www.youtube.com/watch?v=6WruncSoCdI&t=1386s>
- <https://stats.stackexchange.com/questions/20643/finding-matrix-eigenvectors-using-gr-decomposition>