Optimalisasi Kompresi *Video Conference* Menggunakan SVD sebagai Solusi Jaringan dengan *Bandwidth* Terbatas

Azfa Radhiyya Hakim - 13523115[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]aradhihakim@gmail.com, 13523115@std.stei.itb.ac.id

***Abstract*—SVD adalah suatu konsep dalam aljabar, yang memfaktorkan suatu matriks menjadi tiga matriks lainnya yang lebih sederhana. Suatu gambar atau rekaman dapat direpresentasikan sebagai matriks dengan komposisi dan bobot tertentu. Dalam melakukan *video conference*, kadang terdapat jaringan yang memiliki *bandwidth* terbatas, berakibat sulitnya mengakses video yang membutuhkan *bandwidth* yang lebih tinggi. SVD dimanfaatkan untuk mengoptimalkan kompresi video secara efisien.**

***Keywords*—Video, SVD, *Bandwidth*, Optimasi**

I. PENDAHULUAN

Video Conference adalah salah satu metode yang telah dimanfaatkan banyak orang maupun kelompok dalam berkomunikasi jarak jauh, terutama pada dan setelah era Covid-19. Kebutuhan terkait komunikasi visual yang lancar dan berkualitas melalui internet semakin meningkat, baik untuk keperluan pendidikan, bisnis, silaturahmi, atau lain-lain. Di Indonesia, tidak semua tempat memiliki koneksi internet yang stabil, ada yang memiliki kecepatan yang sangat tinggi, ada pula koneksi dengan *bandwidth* yang terbatas. Perbedaan ini dapat menyebabkan ketidaknyamanan dalam berkomunikasi. Dalam konteks *video conference*, akan lebih baik jika komunikasi terus berjalan, walaupun dengan kualitas video yang lebih rendah.

Singular Value Decompotition (SVD), adalah salah satu konsep dalam aljabar linear dan geometri yang dapat memproses gambar atau video secara efektif. SVD memungkinkan kita untuk memfaktorkan suatu matriks menjadi matriks-matriks lainnya yang lebih sederhana, baik dari segi pemakaian memori maupun waktu pemrosesan. Video terbentuk atas beberapa *frame*, yang jika setiap *frame* tersebut dinyatakan sebagai matriks, lalu disederhanakan matriks tersebut, maka akan terbentuk video baru yang menyimpan lebih sedikit memori.

Makalah ini bertujuan untuk mengembangkan optimalisasi kompresi video menggunakan SVD, yang dapat menyelesaikan permasalahan jaringan dengan *bandwidth* terbatas. Dengan mengimplementasikan algoritma kompresi berbasis SVD, diharapkan akan tercapai keseimbangan optimal antara efisiensi *bandwidth* dengan kualitas video yang dihasilkan.

II. TEORI DASAR

A. Matriks

Misalkan K adalah suatu bidang. Misalkan pula terdapat $m, n \in \mathbb{Z}^+$. Matriks berukuran $n \times m$ dengan elemen yang berada di K adalah pemetaan dari $\{1,2,\dots,n\} \times \{1,2,\dots,m\}$ dengan nilai-nilai di K. Secara umum, jika dipunyai matriks M dengan ukuran $n \times m$, M dapat direpresentasikan dalam bentuk

$$M = \begin{pmatrix} m_{11} & \cdots & m_{1m} \\ \vdots & \ddots & \vdots \\ m_{n1} & \dots & m_{nm} \end{pmatrix}$$

Sebagai contoh,

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Contoh tersebut adalah sebuah matriks A yang berukuran 3×3, dengan elemen-elemen yang bersesuaian. Pada baris satu kolom satu, A memiliki elemen 1. Fakta ini dapat dituliskan dalam notasi berikut.

$$A_{11} = 1$$

Secara umum, untuk suatu elemen X pada matriks M yang berada pada baris i dan kolom j, kita dapat menotasikannya sebagai berikut.

$$M_{ij} = X$$

Matriks dapat dimanfaatkan untuk menyelesaikan permasalahan matematika, seperti Sistem Persamaan Linear (SPL). Maka dari itu, akan dikenalkan beberapa operasi matriks sederhana yang akan digunakan di tahap-tahap berikutnya.

- Penjumlahan dan pengurangan matriks

$$M = \begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix} \pm \begin{pmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{pmatrix} = \begin{pmatrix} a_1 \pm b_1 & a_2 \pm b_2 & a_3 \pm b_3 \\ a_4 \pm b_4 & a_5 \pm b_5 & a_6 \pm b_6 \\ a_7 \pm b_7 & a_8 \pm b_8 & a_9 \pm b_9 \end{pmatrix}$$

- Perkalian matriks
  Misalkan terdapat 2 buah matriks A dengan ukuran

$m \times n$ dan matriks B dengan ukuran $n \times r$. Hasil perkalian kedua matriks tersebut adalah matriks berukuran $m \times r$. Adapun untuk setiap elemennya, dapat dinyatakan sebagai berikut.

$$M_{ij} = \sum_{p=0}^{n} A_{mp} \times B_{pr}$$

- Transpose Matriks
  Transpose Matriks adalah matriks yang dioperasikan dengan melakukan pertukaran elemen baris menjadi kolom, dan sebalkinya. Berikut adalah contohnya.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

B. Sistem Persamaan Linear

Sistem Persamaan Linear adalah persamaan aljabar yang mengandung beberapa variable dengan derajat satu. Secara umum, persamaan linear dapat dituliskan dalam bentuk berikut.

$$a_1x_1 + a_2x_2 + a_3x_3 + \cdots + a_nb_n = C_1$$

C. Operasi Baris Elementer

OBE adalah suatu metode mengubah matriks menjadi bentuk yang lebih sederhana yang seringkali digunakan untuk menyelesaikan sistem persamaan linear. Terdapat tiga operasi utama yang dilakukan dalam menggunakan metode ini.

1. Kalikan sebuah baris dengan konstanta tidak nol
2. Pertukarkan dua buah baris
3. Tambahkan sebuah baris dengan kelipatan baris lainnya

D. Metode Eliminasi Gauss dan Gauss Jordan

Kedua metode ini sangat sering digunakan di dunia informatika karena sangat efisien dalam pemakaian memori dan penggunaan waktu. Terdapat 3 proses utama dalam pemakaiannya. Pertama, nyatakan SPL dalam bentuk matriks *augmented*. Selanjutnya, terapkan OBE pada matriks *augmented* hingga terbentuk matriks eselon baris. Matriks eselon baris adalah matriks yang memiliki 1 utama (leading one) pada setiap baris, kecuali baris yang seluruhnya bernilai 0. Selain itu, dikenalkan pula istilah matriks eselon baris tereduksi, yang memiliki sifat yang sama dengan matriks eselon baris, namun setiap kolom yang memiliki 1 utama, memiliki 0 di tempat lain. Target penggunaan eliminasi gauss adalah agar menghasilkan matriks eselon baris, sedangkan eliminasi Gauss jordan menghasilkan eselon baris tereduksi.

$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{pmatrix} \Rightarrow matriks\ eselon\ baris$$

$$\begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \end{pmatrix} \Rightarrow matriks\ eselon\ baris\ tereduksi$$

E. Matriks Balikan

Balikan dari suatu matriks dapat dihitung menggunakan konsep eliminasi Gauss Jordan. Misalkan A adalah matriks persegi berukuran $n \times n$. Balikan dari matriks A adalah A^{-1} sedemikian sehinggan $AA^{-1} = A^{-1}A = I$. Untuk mengetahui matriks A^{-1}, dapat digunakan persamaan

$$[A|I] \sim [I|A^{-1}]$$

yaitu mengubah matriks M menjadi matriks identitas dengan menggunakan Gauss Jordan. Metode Gauss Jordan diterapkan secara simultan untuk A maupun I.

F. Nilai Eigen

Misalkan M adalah suatu matriks berukuran $n \times n$, maka vektor tidak-nol x di $\mathbb{R}^+$ disebut vektor eigen dari M jika mx sama dengan perkalian suatu kalar λ dengan x, yaitu

$$\lambda x = Mx$$

Skalar λ disebut nilai eigen dari M, dan x dinamakan vektor eigen yang berkoresponden dengan λ. Adapun cara yang digunakan untuk menentukan nilai eigen adalah sebagai berikut.

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

solusi dari persamaan tersebut menghasilkan nilai eigen dari matriks A.

G. SVD

Singular Value Decomposition (SVD), merupakan suatu metode pada aljabar yang memfaktorkan suatu matriks M menjadi tiga buah matriks lainnya yang lebih sederhana. Secara aljabar, hal ini dapat dituliskan dalam bentuk berikut.

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

dengan, U adalah matriks ortogonal $m \times m$, V adalah matriks orthogonal $n \times n$, dan $\sum$ adalah matriks berukuran $m \times n$ yang elemen-elemen diagonal utamanya adalah nilai-nilai *singular* dari matriks A dan elemen lainnya adalah 0. Berikut adalah langkah utama dalam menentukan matriks U dan V.

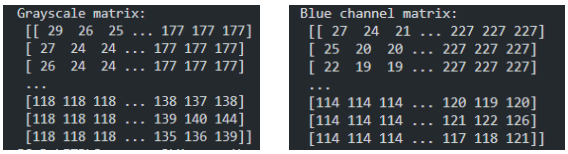
- Tentukan vektor-vektor eigen $\boldsymbol{u_1, u_2, u_3, \ldots, u_m}$ yang berkoresponden dengan nilai eigen dari AA^T. Normalisasi $\boldsymbol{u_1, u_2, u_3, \ldots, u_m}$ dengan membagi setiap komponen vektor dengan panjang vektor. Proses ini akan menghasilkan matriks U.
- Matriks V dihasilkan dengan cara yang sama dalam menghasilkan matriks U, namun nilai eigen dicari bedasarkan A^TA. Kemudian, transpose kan hasilnya untuk memperoleh V^T

Kemudian , dikenalkan istilah SVD tereduksi, yaitu mengurangi banyak baris dan/atau kolom dari matriks $U, \sum$, dan V. Baris-baris dan kolom nol pada matriks $\sum$ dihilangkan. Jika baris i pada $\sum$ dihilangkan, maka kolom i pada matriks U dihilangkan. Sedangkan jika kolom j pada $\sum$ dihilangkan, maka kolom j pada matriks V dihilangkan. Dengan mengompresi, akan didapat matriks yang lebih sederhana dan membutuhkan memori yang lebih sedikit.

H. Video Digital

Video adalah serangkaian gambar statis (*frame*) yang ditampilkan dalam kecepatan tertentu. Setiap *frame* pada suatu video memiliki informasi visual dalam waktu tertentu. *Frame* terdiri atas kumpulan *pixel* yang dapat direpresentasikan sebagai nilai numerik. Untuk setiap *pixel*, jika *frame* tersebut dalam bentuk monokrom, maka nilai yang ditampilkan adalah intensitas warna (0-255), sedangkan jika *frame* tersebut dalam keadaan berwarna (RGB), informasi yang ditampilkan adalah nilai intensitas dari tiga warna (*Red, Green, Blue*).

Gambar 1. Frame ITB



Gambar 2. Contoh matriks grayscale dan blue channel dari Frame ITB

I. Bandwidth

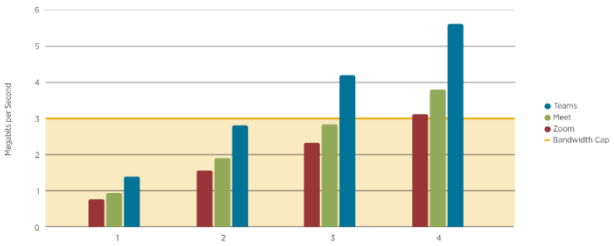
Bandwidth adalah sebuah ukuran kapasitas yang menampung besarnya jumlah data yang dilewati *traffic* dalam jumlah tertentu. *Bandwidth* dipengaruhi oleh beberapa faktor, yaitu medium transmisi yang digunakan, teknologi jaringan yang dimplementasikan, dan infrastruktur jaringan yang tersedia.

Terdapat pengaruh antara kualitas *bandwidth* terhadap video yang diakses. *Bandwidth* yang rendah akan meningkatkan *latency*. Selain itu, *bandwidth* berpengaruh besar terhadap kelancaran gerak pada video. Untuk suatu video, besar *bandwidth* yang dibutuhkan untuk mengaksesnya dengan lancar dapat dinyatakan dalam persamaan berikut.

$$RB = r \times FR \times B \times CR$$

dengan RB adalah *Required Bandwidth*, r adalah *resolution*, FR adalah *frame rate*, B adalah *bits per pixel*, dan CR adalah *compression ratio*. Dikenalkan istilah PSNR (*Peak Signal to Noise Radio*), yaitu metrik untuk menghitung kualitas kompresi video atau gambar. Dikenalkan pula MSE (*Mean Square Error*), suatu metrik yang digunakan untuk mengukur kualitas video atau gambar. Secara umum, PSNR dapat dihitung menggunakan rumus berikut.

$$PSNR = 20 \times \log_{10} \frac{PIXEL_{MAX}}{\sqrt{MSE}}$$



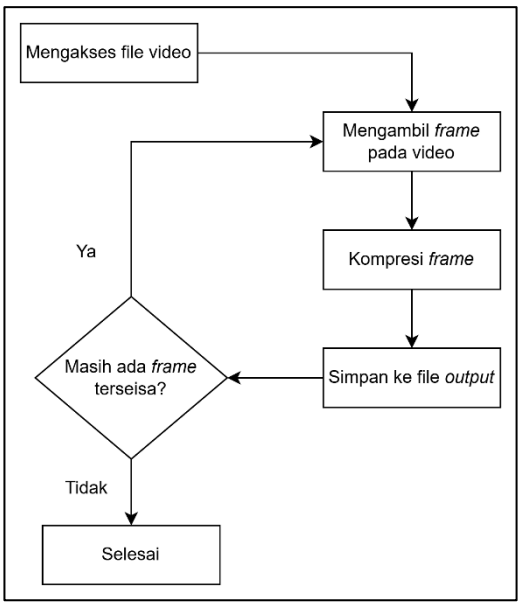
Gambar 3. Hubungan bandwidth terhadap software video conference

(Sumber: https://internetequity.uchicago.edu/data-story/how-much-internet-does-video-conferencing-need/)

III. IMPLEMENTASI DAN PEMBAHASAN

A. Diagram Alur Secara Keseluruhan

Berikut adalah diagram alur program secara keseluruhan.



Gambar 4. Diagram alur secara keseluruhan

B. Fungsi *compress_frame*

```python
def compress_frame(frame, k):
    #inisialisasi list
    compressed_channels = []
    for channel in cv2.split(frame):

        # menerapkan SVD
        U, s, Vt = svd(channel, full_matrices=False)

        # menyesuaikan dengan nilai k
        compressed = np.dot(U[:, :k], np.dot(np.diag(s[:k]), Vt[:k, :]))

        # normalisasi semua nilai agar dalam rentang 0-255
        compressed = np.clip(compressed, 0, 255).astype(np.uint8)

        # menggabungkan semua channel
        compressed_channels.append(compressed)
    return cv2.merge(compressed_channels)
```

Gambar 5. Fungsi compress_frame

Untuk menggunakan fungsi ini, digunakan *libraries* NumPy dan OpenCV. Fungsi pertama yang digunakan adalah *compress_frame*. Fungsi ini menerima dua parameter input, yaitu *frame* dalam bentuk matriks, dan k yang merupakan jumlah *singular values*. *Singular values* adalah elemen diagonal pada matriks diagonal bedasarkan hasil dekomposisi SVD. Semakin kecil nilai k, maka akan semakin tinggi kompresi yang dilakukan, namun kualitas yang dihasilkan akan semakin jelek. Semakin besar nilai k (mendekati nilai asli), maka *frame* yang terkompress akan hanya sedikit dan menghasilkan *frame* dengan kualitas yang baik. Pada program ini, digunakan nilai $k = 50$ untuk menyeimbangkan kualitas dan hasil kompresi. Diterapkan SVD pada *frame*, menghasilkan tiga faktor matriks, yaitu $U, s,$ dan vt. Langkah selanjutnya adalah menyesuaikan hasil yang didapat dengan nilai k yang telah diatur. Hasil dari langkah ini adalah hasil terkompresi dari *frame* yang diberikan. Nilai matriks yang dihasilkan tidak selalu berada dalam *range* yang serupa, ada yang sangat kecil, ada yang sangat tinggi. Untuk menyesuaikan, digunakan normalisasi dengan nilai 0-255. Selanjutnya digabungkan kembali ketiga matriks tersebut dan diperoleh *output* berupa *frame* dengan dimensi yang sama dengan *frame* pada *input*, namun dengan ukuran data yang lebih kecil. Berikut adalah contoh kompresi *frame*.

Gambar 6. Sebelum dan sesudah kompresi

C. Fungsi *process_video*

Fungsi ini menerima dua buah parameter *input*, yaitu file video yang akan dikompresi, dan file video tempat hasil kompresi akan disimpan. Fungsi ini diimplementasikan menggunakan *libraries* opencv

```python
def process_video(input_path, output_path):

    # membuka video
    cap = cv2.VideoCapture(input_path)

    # mengambil data yang dibutuhkan
    fps = int(cap.get(cv2.CAP_PROP_FPS))
    width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

    # setup video output tempat hasil kompresi disimpan
    newvideo = cv2.VideoWriter_fourcc(*'mp4v')
    out = cv2.VideoWriter(
        output_path,
        newvideo, fps,
        (width, height)
    )

    frame_count = 0
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        try:
            # kompres frame menggunakan fungsi compress_frame
            compressed_frame = compress_frame(frame, 50)

            # menuliskan hasil kompresi pada file output
            out.write(compressed_frame)
            frame_count += 1

        # antisipasi jika terjadi error
        except Exception as e:
            print(f"Error processing frame {frame_count}: {str(e)}")
            continue

    # membersihkan cap dan out setelah selesai procesing
    cap.release()
    out.release()
    return
```

Gambar 7. Fungsi process_video

Tahap pertama adalah membuka file video yang diinput. Kemudian, akan dicari nilai fps (*frame per second)*, lebar *frame*, dan tinggi *frame*. Hal ini untuk menyesuaikan file hasil kompresi nantinya.

Kemudian, dibuat *"newvideo"*, yaitu sebuah setup yang digunakan untuk menulis *frame* hasil kompresi ke dalam file video baru, yang membutuhkan *fps*, lebar *frame*, dan tinggi *frame* yang telah diketahui sebelumnya. *Mp4v* digunakan untuk mendecode file video.

Kemudian, setiap *frame* yang ada pada video akan diproses. Bagian kode berikut,

```python
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
```

Gambar 8. Pembacaan frame

adalah tahap pembacaan frame, dimana *ret*

mengindikasikan keberhasilan pembacaan, sedangkan *frame* merupakan hasil *frame* yang diperoleh. Setelah diperoleh *frame*, langkah berikutnya adalah mengaplikasikan fungsi *compress_frame* terhadap *frame* tersebut. Ini dilakukan secara *looping*, hingga semua *frame* telah berhasil di kompresi dan cap tertutup. Untuk setiap *frame* yang telah berhasil di kompresi, hasil tersebut akan langsung disimpan pada *out* yaitu sebuah *videowriter* yang dapat membuat video dari kumpulan frame.

D. fungsi main

Fungsi selanjutnya adalah fungsi main, yaitu *interface* dimana pengguna dan program berinteraksi. Penulis berencana untuk membuat *interface* berbasis *web application*, namun karena keterbatasan waktu, *interface* hanya akan dibuat dalam terminal. Tentu saja kedepannya, penulis akan membuat suatu *website* yang dapat mengaplikasikan fungsi-fungsi yang telah diperoleh ini. Berikut adalah fungsi *main*.

```python
def main():
    print("Made by: Azfa Radhiyya Hakim")
    print("Selamat datang di program kompresi video!")
    inputfile = input("Masukkan nama file video yang ingin Anda kompresi: ")
    outputfile = input("Masukkan nama file hasil kompresi: ")

    print(".")
    print(".")
    print(".")

    print("Memulai kompresi video...")
    process_video(inputfile, outputfile)

    print("Kompresi video selesai!")
    print("Sampai jumpa kembali!")
```

Gambar 9. Fungsi main

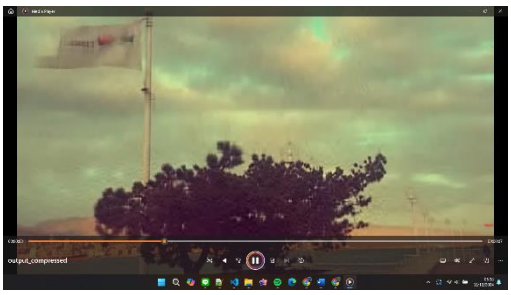
Jika *source code* tersebut di jalankan, akan menghasilkan sebagai berikut.

```
Made by: Azfa Radhiyya Hakim
Selamat datang di program kompresi video!
Masukkan nama file video yang ingin Anda kompresi: input.mp4
Masukkan nama file hasil kompresi: output_compressed.mp4
.
.
.
Memulai kompresi video...
Processed 301 frames.
Kompresi video selesai!
Sampai jumpa kembali!
```

Gambar 10. Interface fungsi main

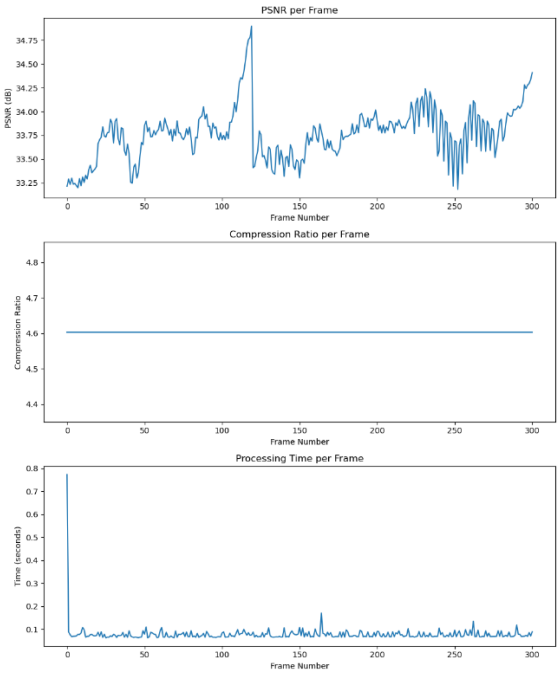
Besar file video sebelum dan sesudah kompresi secara berturut-turut adalah 4 MB dan 1,5 MB. Berikut adalah tangkapan layar dari hasil kompresi video, yang dapat menggambarkan seberapa jauh video berubah.

Gambar 11. Video sebelum kompresi



Gambar 12. Video setelah kompresi

Berikut adalah grafik hasil kompresi metrik.



Gambar 13. Grafik kompresi metrik

Dari grafik tersebut, terdapat beberapa informasi yang dapat diperoleh

- Program menampilkan performa yang stabil, ditunjukkan dengan rasio kompresi yang tetap
- Waktu pemoresesan yang konsisten menunjukan algoritma yang efisien
- Rasio kompresi yang tetap menghasilkan PSNR yang baik (33.25 dB – 34.75 dB)

V. KESIMPULAN

Berdasarkan analisis tersebut, telah dirancang sebuah solusi konkret dengan memanfaatkan SVD, yang mampu mengompresi video menjadi ukuran yang lebih kecil. Perbedaan ukuran hasil kompresi ini sangat signifikan dalam konteks efisiensi *bandwidth*, terutama ketika ketersediaan *bandwidth* menurun atau meningkat.

VI. UCAPAN TERIMA KASIH

Saya mengucapakan rasa syukur kepada Allah SWT,

yang karena Rahmat-Nya, saya diberi kemudahan dalam menyelesaikan makalah ini dengan baik dan tepat waktu. Selain itu, saya mengucapkan terima kasih kepada Ir. Rila Mandala, M.Eng., Ph.D. selaku dosen mata kuliah IF 2123 Aljabar Linear yang sudah mengajarkan materi semester 3 ini dengan sangat baik dan menyenangkan. Tak lupa saya ucapkan terima kasih kepada kedua orang tua saya yang telah banyak membantu dan mendukung saya selama proses pengerjaan makalah. Saya juga ingin mengucapkan terima kasih kepada teman-teman yang tidak dapat saya sebutkan satu-satu, atas bantuan dan dukungannya dalam pembuatan makalah ini.

DAFTAR PUSTAKA

[1] CBT&P. "Pengertian Bandwidth dan Fungsinya." CBT&P Knowledge Base. Diakses 31 Desember 2024. https://cbtp.co.id/kb/knowledge-base/pengertian-bandwidth-dan-fungsinya/

[2] Munir, Rinaldi, 2023. “Nilai Eigen dan Vektor Eigen (Bagian 1)”. https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian1-2023.pdf

[3] Munir, Rinaldi, 2023. “Singular Value Decomposition (Bagian 2)”. https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-22-Singular-value-decomposition-Bagian2-2023.pdf

[4] Munir, Rinaldi, 2023. “Sistem persamaan linear (Bagian 1)”. https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-03-Sistem-Persamaan-Linier-2023.pdf

[5] Munir, Rinaldi, 2023. “Sistem persamaan linear (Bagian 2)”. https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-04-Tiga-Kemungkinan-Solusi-SPL-2023.pdf

[6] Munir, Rinaldi, 2024. “Sistem persamaan linear (Bagian 3)”. https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-05-Sistem-Persamaan-Linier-2-2023.pdf

[7] Serre, Denis, 2001. “Matrices: Theory and Applications”. Springer.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 1 Januari 2025

Azfa Radhiyya Hakim, 13523115