Optimalisasi Hasil Diagnostik: Implementasi *Singular Value Decomposition* (SVD) dalam Pencitraan Medis

Muhammad Raihan Nazhim Oktana - 13523021[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]m.raihannazhimoktana@gmail.com, 13523021@std.stei.itb.ac.id

Abstrak—Aljabar linier dan geometri adalah suatu cabang materi dalam matematika yang punya banyak aplikasi dalam dunia teknologi seperti pada informatika. Aljabar linier dan geometri mengandung dua cabang materi utama, yaitu aljabar linier dan aljabar geometri. Penggabungan dua cabang ini, menjadi satu materi yang cukup menarik dalam dunia informatika dan teknologi untuk dimanfaatkan dalam dunia nyata. Salah satu ilmu pada aljabar linier dan geometri adalah *Singular Value Decomposition* (SVD). Secara umum, SVD adalah cabang materi dari dekomposisi matriks yang memberikan suatu teknik untuk memecah sebuah matriks menjadi tiga buah komponen yang sangat penting untuk dikelola dan dianalisis lebih lanjut. Hingga saat ini, perkembangan dunia medis telah sampai pada tahap dimana perlu adanya perkembangan teknologi yang cukup pesat untuk menyelesaikan ragam persoalan yang muncul. Aljabar linier dan geometri punya peran penting dalam memajukan berbagai teknologi kedokteran yang ada. Penerapan SVD yang tepat akan sangat membantu pembuatan teknologi yang lebih baik lagi. Pada kesempatan kali ini, penulis ingin mencari tahu dan menganalisis implementasi SVD dalam pemrosesan citra medis untuk optimalisasi hasil diagnostik yang lebih baik. Dari hasil percobaan yang dilkaukan, ditemukan bahwa terbukti aplikasi SVD pada optimalisasi pencitraan dunia medis sangat baik dan bermanfaat.

Kata kunci—aljabar linier dan geometri, citra medis, optimalisasi diagnostik, SVD.

I. PENDAHULUAN

Aljabar linier dan geometri adalah suatu cabang materi dalam matematika yang punya banyak aplikasi dalam dunia teknologi seperti pada informatika. Aljabar linier dan geometri mengandung dua cabang materi utama, yaitu aljabar linier dan aljabar geometri. Pada materi aljabar linier, ada beberapa materi yang menjadi fokusnya, yaitu tentang vektor, matriks, dan sistem persamaan linier. Kegunaan aljabar linier pada umumnya adalah untuk proses analisis data. Pada materi aljabar geometri, ada beberaoa materi yang menjadi fokusnya, yaitu tentang bentuk, dimensi, dan ruang. Kegunaan aljabar geometri pada umumnya adalah untuk perluasan dalam pengembangan grafika komputer dan sebagainya. Penggabungan dua cabang ini, menjadi satu materi yang cukup menarik dalam dunia informatika untuk dimanfaatkan dalam dunia nyata.[1]

Pada materi aljabar linier dan geometri, terdapat banyak sekali cabang-cabang ilmu. Salah satunya adalah *Singular Value Decomposition* (SVD). Secara umum, SVD adalah cabang materi dari dekomposisi matriks yang memberikan suatu teknik untuk memecah sebuah matriks menjadi tiga buah komponen yang sangat penting untuk dikelola dan dianalisis lebih lanjut. Terdapat banyak sekali kegunaan SVD, beberapa yang paling umum adalah untuk analisis data dalam dimensi yang cukup tinggi, pengurangan dimensi, pengolahan sinyal, sistem kompresi, pemisahan informasi, pencarian informasi, dan masih banyak lagi.[9]

Hingga saat ini, perkembangan dunia medis telah sampai pada tahap dimana perlu adanya perkembangan teknologi yang cukup pesat untuk menyelesaikan ragam persoalan yang muncul. Dalam hal ini, perkembangan dunia informatika dan kaitannya terhadap dunia medis sangatlah penting. Dapat dilihat di berbagai rumah sakit atau fasilitas layanan kesehatan pada umumnya bahwa terdapat banyak teknologi modern seperti CT-*scan* atau MRI. Teknologi ini memungkinkan para dokter atau tenaga kesehatan untuk mendapatkan suatu visualisasi yang lebih tepat terkait struktur internal dari tubuh pasiennya yang tidak bisa dilihat secara langsung dengan kasat mata. Hal ini memungkinkan para tenaga Kesehatan untuk menganalisis dan mengidentifikasi hal yang terjadi pada pasiennya agar dapat memberikan diagnosis yang semakin tepat dan semakin efektif.[14]

Aljabar linier dan geometri punya peran penting dalam memajukan berbagai teknologi kedokteran yang ada. Penerapan SVD yang tepat akan sangat membantu pembuatan teknologi yang lebih baik lagi. Perkembangan teknologi di dunia tidak akan pernah berhenti. Masih banyak pengembangan untuk optimalisasi dan inovasi baru yang dapat dilakukan. Oleh karena itu, pada kesempatan kali ini, penulis ingin mencari tahu dan menganalisis implementasi SVD dalam pemrosesan citra medis untuk optimalisasi hasil diagnostik yang lebih baik. Penulis akan mencoba melakukan penerapan SVD dalam pemrosesan citra medis dengan menggunakan materi-materi pada aljabar linier dan geometri yang berkaitan.[1]

II. DASAR TEORI

A. Aljabar Linier

Aljabar linier adalah salah satu cabang matematika pecahan dari aljabar linier dan geometri. Keberadaan aljabar linier sangat penting karena punya fokus pada banyak hal dasar seperti vektor, ruang vektor, transformasi linier, dan sistem persamaan linier. Dasar

teori dalam aljabar linier banyak digunakan dalam aplikasi di bidang fisika, ilmu komputer, statistika, dan masih banyak lainnya. Berbagai konsep penting seperti matriks dan nilai eigen, menjadi kunci penting dalam aljabar linier karena memungkinkan melakukan analisis secara mendalam dengan berbagai teknik yang ada.[1]

B. Aljabar Geometri

Aljabar geometri adalah salah satu cabang matematika pecahan dari aljabar linier dan geometri. Keberadaan aljabar geometri sangat penting karena punya fokus pada banyak hal dasar seperti bidang-bidang geometri dan ruang. Penerapan aljabar geometri meliputi berbagai teknik dalam aljabar. Hal ini memungkinkan para pengguna teknik ini untuk menyelesaikan berbagai permasalahan terkait dengan geometri.[1]

C. Matriks

Matriks adalah susunan dari sekumpulan bilangan atau hal lainnya yang disusun dalam bentuk baris dan kolom. Matriks digunakan secara luas dalam aljabar linier untuk mewakili sistem persamaan linear, seperti dapat menjadi representasi suatu gambar, suatu graf, suatu system persamaan linier, hingga kernel di dalam metode deep learning. Matriks memiliki beberapa jenis, seperti matriks persegi, matriks identitas, matriks nol, dan lainnya. Dalam operasi matriks pun, ada banyak yang bisa dilakukan seperti penjumlahan matriks, pengurangan matriks, perkalian matriks, invers matriks, dan lainnya. Serta, ada juga sifat-sifat matriks yang bisa berguna dalam proses operasi matriks.

1. Jenis Matriks

Terdapat tiga jenis matriks yang umum untuk digunakan dalam pembahasan matriks, yaitu matriks persegi, matriks identitas, dan matriks nol. Matriks persegi merupakan matriks yang bebentuk persegi. Matriks identitas merupakan matriks yang berbentuk persegi dan seluruh elemen diagonalnya bernilai 1 dan selain itu bernilai nol. Matriks nol ialah sebuah matriks yang seluruh elemennya bernilai nol. Tentu masih banyak jenis lainnya lagi, hanya sebagian kecil saja yang dibahas di sini.

2. Sifat Matriks

Untuk memudahkan proses operasi matriks, terdapat beberapa sifat yang bisa digunakan untuk melakukan operasi matriks sebagai berikut.

(a) $A+B=B+A$ (**Commutative law for addition**)
(b) $A+(B+C)=(A+B)+C$ (**Associative law for addition**)
(c) $A(BC)=(AB)C$ (**Associative law for multiplication**)
(d) $A(B+C)=AB+AC$ (**Left distributive law**)
(e) $(B+C)A=BA+CA$ (**Right distributive law**)
(f) $A(B-C)=AB-AC$
(g) $(B-C)A=BA-CA$
(h) $a(B+C)=aB+aC$
(i) $a(B-C)=aB-aC$
(j) $(a+b)C=aC+bC$
(k) $(a-b)C=aC-bC$
(l) $a(bC)=(ab)C$
(m) $a(BC)=(aB)C=B(aC)$

Gambar 1. Sifat Operasi Aritmetika Matriks[2]

Pada operasi transpose matriks, terdapat beberapa sifat khusus tambahan yang dapat dipakai untuk membantu sebagai berikut.

(a) $\left(A^T\right)^T = A$
(b) $(A+B)^T = A^T + B^T$
(c) $(A-B)^T = A^T - B^T$
(d) $(kA)^T = kA^T$
(e) $(AB)^T = B^T A^T$

Gambar 2. Sifat Operasi Transpose Matriks[2]

3. Operasi Matriks

Operasi matriks yang pertama adalah penjumlahan dan pengurangan matriks, yaitu menjumlahkan atau mengurangkan elemen pada posisi yang sama antar kedua matriks. Pada operasi ini, diharuskan kedua matriks memiliki ukuran yang sama.

$$\mathbf{A}+\mathbf{B} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix}$$
$$= \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn} \end{bmatrix}$$

Gambar 3. Operasi Matriks Penjumlahan[2]

Operasi matriks yang kedua adalah perkalian antar matriks, yaitu mengalikan dan menjumlahkan elemen matriks pada baris dan/atau kolom yang bersesuaian. Pada operasi ini, diharuskan bahwa jika matriks A berukuran m x n, maka matriks B berukuran n x p, dengan matriks hasil perkalian berukuran m x p.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix}$$

$$\mathbf{C} = \begin{pmatrix} a_{11}b_{11}+\cdots+a_{1n}b_{n1} & a_{11}b_{12}+\cdots+a_{1n}b_{n2} & \cdots & a_{11}b_{1p}+\cdots+a_{1n}b_{np} \\ a_{21}b_{11}+\cdots+a_{2n}b_{n1} & a_{21}b_{12}+\cdots+a_{2n}b_{n2} & \cdots & a_{21}b_{1p}+\cdots+a_{2n}b_{np} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{11}+\cdots+a_{mn}b_{n1} & a_{m1}b_{12}+\cdots+a_{mn}b_{n2} & \cdots & a_{m1}b_{1p}+\cdots+a_{mn}b_{np} \end{pmatrix}$$

Gambar 4. Operasi Matriks Perkalian[2]

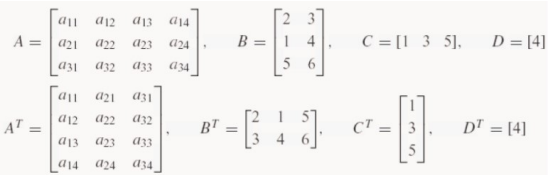
Operasi matriks yang ketiga adalah perkalian scalar dengan matriks, yaitu mengalikan setiap elemen pada matriks dengan nilai skalar yang diberikan.

• Contoh: Misakan $A = \begin{bmatrix} 2 & 3 & 4 \\ 1 & 3 & 1 \end{bmatrix}$, $B = \begin{bmatrix} 0 & 2 & 7 \\ -1 & 3 & -5 \end{bmatrix}$, $C = \begin{bmatrix} 9 & -6 & 3 \\ 3 & 0 & 12 \end{bmatrix}$

maka $2A = \begin{bmatrix} 4 & 6 & 8 \\ 2 & 6 & 2 \end{bmatrix}$, $(-1)B = \begin{bmatrix} 0 & -2 & -7 \\ 1 & -3 & 5 \end{bmatrix}$, $\frac{1}{3}C = \begin{bmatrix} 3 & -2 & 1 \\ 1 & 0 & 4 \end{bmatrix}$

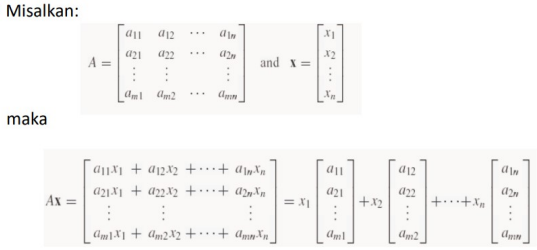
Gambar 5. Operasi Matriks Perkalian Skalar[2]

Operasi matriks yang keempat adalah transpose matriks, yaitu memutar matriks sehingga setiap elemen kolom menjadi elemen baris. Pada operasi ini, jika matriks awal berbentuk m x n, maka matriks hasil akan berbentuk n x m.



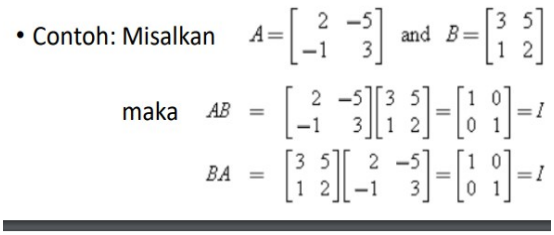
Gambar 6. Operasi Matriks Transpose[2]

Operasi matriks yang kelima adalah kombinasi linier matriks, yaitu memecah matriks dalam arti memandang perkalian matriks sebagai kombinasi linier.



Gambar 7. Operasi Matriks Kombinasi Linier[2]

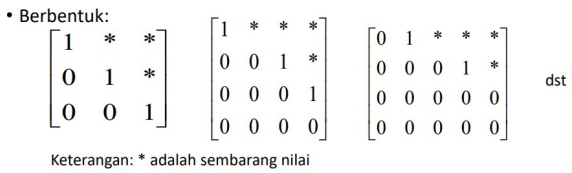
Operasi matriks yang keenam adalah invers matriks, yaitu proses membalik matriks sesuai dengan aturan yang ada, di mana matriks awal dikalikan dengan matriks invers akan menghasilkan matriks identitas.



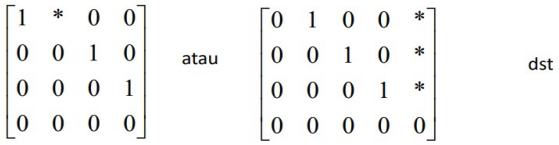
Gambar 8. Operasi Matriks Invers[2]

D. Operasi Baris Elementer

Operasi baris elementer pada matriks merupakan suatu bentuk operasi matriks yang bertujuan untuk membuat sebuah matriks menjadi sebuah matriks eselon baris. Sifat matriks eselon baris adalah punya 1 utama di depan, baris yang seluruhnya nol ditaruh di bawah, dan tidak ada yang punya 1 utama sejajar. Lebih spesifik, terdapat matriks eselon baris tereduksi juga yang seupa dengan matriks eselon baris, hanya saja seluruh elemen pada kolom 1 utama haruslah 0.



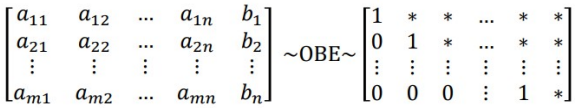
Gambar 9. Matriks Eselon Baris[3]



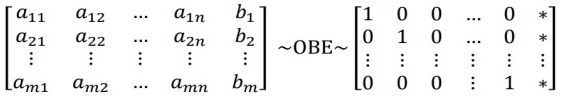
Gambar 10. Matriks Eselon Baris Tereduksi[3]

Operasi baris elementer pada umumnya digunakan untuk menyelesaikan sistem persamaan linier yang membentuk suatu matriks eselon baris atau matriks eselon baris tereduksi pada akhirnya. Jika hasilnya adalah suatu matriks eselon baris, maka metodenya disebut metode gauss, sedangkan jika hasilnya adalah suatu matriks eselon baris tereduksi, maka metodenya disebut metode gauss-jordan.

Operasi baris elementer akan melalui tiga tipe transformasi yang digunakan untuk menyederhanakan matriks, yaitu mengalikan sebuah baris dengan konstanta tidak nol, melakukan pertukaran dua buah baris, atau menambahkan sebuah baris dengan kelipatan baris lainnya. Dari hasil operasi ini, kita bisa dapatkan tiga jenis solusi sistem persamaan linier, yaitu punya solusi tunggal, punya solusi tak hingga, dan tidak punya solusi.[5]



Gambar 11. OBE Metode Eliminasi Gauss[4]



Gambar 12. OBE Metode Eliminasi Gauss-Jordan[6]

E. Nilai Eigen & Vektor Eigen

Secara definisi, untuk sebuah matriks persegi A berukuran n x n, maka vektor tidak-nol x di R^n disebut vektor eigen dari A jika Ax sama dengan perkalian suatu skalar λ dengan x. Skalar λ disebut nilai eigen dari A, dan x disebut vektor eigen yang berkoresponden dengan λ. Vektor eigen x menyatakan matriks kolom yang apabila dikalikan dengan sebuah matriks n x n akan menghasilkan vektor lain berupa kelipatan vektor itu sendiri.[7 dan 8]

$$Ax = \lambda x$$
$$IAx = \lambda Ix$$
$$Ax = \lambda Ix$$
$$(\lambda I - A)x = 0$$

Gambar 13. Rumus Nilai Eigen dan Vektor Eigen[7]

F. Dekomposisi Matriks

Secara definisi, dekomposisi matriks memiliki konsep memfaktrokan sebuah matriks menjadi perkalian dari matriks faktor-faktornya.

$$A = P_1 \times P_2 \times \ldots \times P_k$$

Gambar 14. Konsep Dekomposisi Matriks[9]

Dalam pembahasan dekomposisi matriks, terdapat tiga metode dekomposisi matriks yang umum untuk digunakan, yaitu metode dekomposisi LU, dekomposisi QR, dan dekomposisi nilai singular (SVD).

1. Dekomposisi LU

Dekomposisi LU (*Lower-Upper*) merupakan suatu konsep dekomposisi matriks yang memecah matriks menjadi matriks L (segitiga bawah), dan matriks U (segitiga atas) sedemikian sehingga hasil perkalian kedua matriks tersebut adalah matriks asalnya.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix} \times \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix}$$

A L U

Gambar 15. Konsep Dekomposisi LU[11]

2. Dekomposisi QR

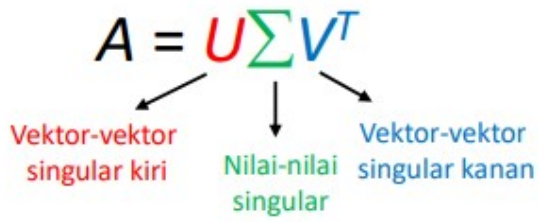
Dekomposisi QR merupakan suatu konsep dekomposisi matriks yang memecah matriks menjadi matriks 2 bagian, yaitu matriks Q (orthonormal), dan matriks R (segitiga atas) sedemikian sehingga hasil perkalian kedua matriks tersebut adalah matriks asalnya.

$$A = QR$$

Gambar 15. Konsep Dekomposisi QR[12]

3. Dekompisisi SVD

Dekomposisi yang akan digunakan pada pengamatan kali ini adalah SVD (*Singular Value Decomposition*) yang merupakan suatu konsep dalam dekomposisi matriks yang memecah matriks menjadi tiga bagian, yaitu matriks U (vektor singular kiri), matriks Σ (nilai-nilai singular), dan matriks V^T (vektor singular kanan) sedemikian sehingga hasil perkalian ketiga matriks tersebut adalah matriks asalnya.



Gambar 16. Konsep Dekomposisi SVD[10]

G. Pencitraan Medis

Citra medis adalah sebuah gambaran hasil dari suatu teknologi yang memungkinkan dokter atau tenaga medis untuk melihat gambaran internal tubuh seseorang tanpa perlu melakukan operasi seperti pembedahan atau semacamnya. Hal ini didukung oleh kemajuan teknologi yang ada seperti *computed tomography scan* (CT-*scan*) yang menggunakan teknologi X-*ray* yang diproses melalui suatu komputer untuk menghasilkan gambaran suatu potongan tubuh yang di-*scan* dengan sangat detil. Tak hanya itu, ada juga teknologi *magnetic resonance imaging* (MRI) yang menggunakan medan magnet kuat dan gelombang radio untuk menghasilkan suatu gambaran rinci dari organ-organ dan jaringan-jaringan di dalam tubuh seseorang.[14]

H. Aplikasi SVD pada Pencitraan Medis

Dua metode umum yang digunakan dalam pencitraan medis adalah CT-scan dan MRI. Kedua metode ini mempunyai suatu kelebihan dan kekurangannya masing-masing dalam aplikasinya. Namun, dari sudut pandang aljabar linier dan geometri, kedua metode tersebut akan menghasilkan data citra yang cukup besar untuk kemudian dianalisis dengan berbagai teknik. Salah satu tekniknya adalah SVD untuk mendapatkan informasi yang lebih spesifik dan detail. Dalam konteks ini, SVD akan berperan dalam reduksi noise, kompresi citra, dan peningkatan kualitas citra. Dengan melakukan dekomposisi SVD, dapat diperoleh elemen pentingnya dan mengurangi memori yang besar tadi.[14]

Gambar 17. Gambar Alat CT Scan
Sumber: GambarCTScan

I. Information Retrival

Temu-balik informasi merupakan suatu konsep menemukan informasi yang relevan berdasarkan pencarian pengguna dari suatu kumpulan informasi secara otomatis. Konsep ini sangat membantu pengamatan ini karena pencarian citra yang relevan dari *query* ke *database* yang dimiliki oleh sistem. Dengan adanya teknologi ini, proses pengelolaan data dan diagnostik menjadi lebih gampang dengan adanya sistem yang secara otomatis mencari data yang relevan dari *database*.

Gambar 18. Konsep Information Retrival[13]

III. PEMBAHASAN

A. Perhitungan SVD

Perhitungan SVD secara umum dilakukan dengan 2 tahap utama. Kedua tahap ini dapat digabungkan untuk menemukan ketiga komponen dekomposisi matriks SVD tersebut.

Pada tahap 1, cara menghitung dilakukan dengan mengubah matriks A menjadi A^TA. Hal ini punya tujuan

untuk mendapatkan nilai-nilai eigen dan vektor-vektor eigen dari matriks A. Selanjutnya, hitung nilai-nilai singular dengan mengakarkan nilai-nilai eigen dan susun nilai-nilai singular tersebut pada matriks Σ. Lalu, susun vektor-vektor eigen yang telah dinormalisasi pada V dan lakukan transpose menjadi V^T. Melalui tahap 1 ini, didapatkan matriks Σ dan V^T.

Pada tahap 2, akan dilakukan cara yang serupa dengan perhitungan pada tahap 1. Pada tahap 2 ini, matriks A akan diubah menjadi matriks AA^T. Selanjutnya, matriks tersebut akan dicari nilai-nilai yang serupa dengan eigen dan vektor-vektor yang serupa dengan eigennya seperti pada tahap 1. Lalu, normalisasi vektor-vektor tersebut dan susun pada matriks U. Melalui tahap 2 ini, didapatkan matriks U. Sehingga, melalui kedua tahap tersebut didapatkan ketiga matriks hasil dekomposisi, yaitu matriks U, Σ, dan V^T.

B. Pencarian Data Serupa

Proses pencarian data yang serupa pada database menggunakan konsep dasar dari sistem temu kembali. Proses ini dilakukan dengan cara menghitung jarak euclidean antara *query* yang sedang dicari dan seluruh database yang ada. Jarak yang lebih dekat dinilai memiliki kemiripan dengan query lebih baik. Dengan menggunakan pendekatan bahwa data dengan jarak paling jauh memiliki kemiripan 0%, dapat dihitung kemiripan setiap data dengan persentase dan mengelompokkannya berdasarkan tingkat kemiripannya. Perhitungan jarak euclidean menggunakan rumus sebagai berikut.

$$d(\mathbf{q}, \mathbf{z}_i) = \sqrt{\sum_{j=1}^{k} (q_j - z_{ij})^2}$$

Gambar 19. Perhitungan Jarak Euclidean
Sumber: RumusJarakEuclidean

C. Pemrosesan Citra yang Lebih Jelas

Dalam pemrosesan suatu citra terutama gambar, dalam, SVD dapat digunakan untuk salah satunya untuk memperbaiki citra, rekonstruksi citra, dan sebagainya. Dalam proses memperbaiki citra, ada proses denoising yang menyaring nilai-nilai singular kecil yang pada umumnya berkaitan dengan noise sehingga menghasilkan sisa nilai singular yang besar yang kemudian dilakukan rekonstruksi citra. Proses ini cenderung membuat citra menjadi lebih bersih dan lebih jelas karena menghilangkan noise yang ada. Implementasi ini sangat baik untuk dilakukan dalam dunia medis ataupun bukan. Dalam dunia medis, hal ini penting untuk memberikan detail citra yang jelas untuk membantu proses dokter dan tenaga medis untuk melakukan diagnosis.

D. Dampak Positif pada Dunia Medis

Penerapan aljabar linier dan geometri dalam dunia medis terutama pada pengembangan teknologi medis sangat besar. SVD sebagai salah satu teknik yang punya banyak manfaat, terbukti mampu melakukan pengolahan citra dengan sangat baik. Pada penerapannya, CT-scan dan MRI menggunakan SVD untuk melakukan *denoising* dan penajaman citra untuk menghasilkan citra dengan lebih baik. Pencitraan dalam dunia medis sangat penting karena detail struktur kecil yang bisa terlihat dengan jelas memungkinkan para dokter atau tenaga medis untuk memberikan diagnostik dengan lebih akurat.

Penerapan SVD juga memungkinkan penggunanya untuk melakukan kompresi dan dekompresi citra dengan baik tanpa menghilangkan informasi penting pada citra. Hal ini memungkinkan penyimpanan data citra medis dengan baik dan hemat memori sehingga mengurangi beban pada sistem. Ditambah dengan adanya fitur untuk penyesuaian pencocokan secara otomatis dengan gambar yang relevan pada databsase, memungkinkan proses diagnostik berjalan dengan lebih baik. Dengan itu, terbukti bahwa banyak dampak positif yang diberikan dari penerapan aljabar linier dan geometri pada dunia medis.

IV. IMPLEMENTASI PROGRAM

Proses implementasi program implementasi SVD dalam pencitraan medis untuk optimalisasi hasil diagnostik membutuhkan beberapa tahap implementasi. Tahapan yang akan dijelaskan di sini mengabaikan tahapan kecil seperti menginput citra, atau semacamnya. Secara umum, tahap implementasi ini meliputi proses konversi citra menjadi matriks, *data centering* pada *database*, perhitungan SVD, reduksi noise, rekonstruksi citra, dan perhitungan similaritas.

Untuk implementasi pertama, dapat dilihat seperti pada program berikut. Menggunakan bantuan library khusus dari PIL untuk melakukan konversi, melakukan penyesuaian menjadi berukuran n x n, dan menyimpan nilai setiap piksel dalam intensitas RGB nya. Perhitungan intensitas dilakukan dengan 0.2989 * R + 0.5870 * G + 0.1140 * B. Selanjutnya, setiap gambar pada database akan dijadikan kumpulan matriks dengan cara yang sama.

```
def convert_picture(path : str) -> list[float] :
    # SPESIFIKASI LOKAL
    # Melakukan konversi gambar menjadi pixel RGB dengan bantuan PIL, lalu menyimpan hasil intensitasnya pada list of float.

    # KAMUS LOKAL
    # img : ImageFile
    # pixels : list of tuple of integer
    # res : list of float
    # path : string
    # size , n : integer
    # i : integer (index)

    # ALGORITMA LOKAL
    n = 32
    img = Image.open(path)
    img = img.resize((n , n) , Image.Resampling.LANCZOS)
    img = img.convert('RGB')
    pixels = list(img.getdata())
    size = n * n
    res = [0.0 for i in range (size)]
    for i in range (size) :
        res[i] = 0.2989 * pixels[i][0] + 0.5870 * pixels[i][1] + 0.1140 * pixels[i][2]
    return res
```

Gambar 19. Perhitungan Jarak Euclidean
Sumber: Dokumen Penulis / Lampiran

Untuk implementasi kedua, dapat dilihat seperti pada program berikut. Proses *data centering* dilakukan dengan cara menghitung rata-rata dari nilai intensitas setiap piksel pada posisi yang sama pada semua citra di database. Selanjutnya, proses *data centering* dilanjutkan dengan mengurangi nilai setiap piksel pada setiap data di matriks tersebut dengan rata-rata yang telah dihitung sebelumnya. Hasilnya adalah suatu matriks yang memiliki nilai rata-rata di setiap pikselnya adalah 0. Hal ini bertujuan untuk membuat perhitungan lebih optimal dan tepat sasaran.

```python
def find_average(matrix : list[list[float]]) -> list[float] :
    # DESKRIPSI LOKAL
    # Sebuah fungsi untuk menghitung rata-rata intensitas di posisi yang bersesuaian.

    # KAMUS LOKAL
    # matrix : matrix of float
    # avg : list of float
    # row , col : integer
    # i , j : integer (index)

    # ALGORITMA LOKAL
    row = len(matrix)
    col = len(matrix[0])
    avg = [0.0 for i in range (col)]
    for i in range (row) :
        for j in range (col) :
            avg[j] = avg[j] + matrix[i][j]
    for i in range (len(avg)) :
        avg[i] = avg[i] / row
    return avg
```

Gambar 20. Perhitungan Rata-Rata
Sumber: Dokumen Penulis / Lampiran

```python
def data_centering(matrix : list[list[float]]) -> list[list[float]] :
    # DESKRIPSI LOKAL
    # Sebuah prosedur untuk mengurangi nilai setiap elemen intensitas dengan rata-rata yang bersesuaian

    # KAMUS LOKAL
    # matrix : matrix of float
    # avg : list of float
    # row , col : integer
    # i , j : integer (index)

    # ALGORITMA LOKAL
    avg = find_average(matrix)
    row = len(matrix)
    col = len(matrix[0])
    for i in range (row) :
        for j in range (col) :
            matrix[i][j] = matrix[i][j] - avg[j]
    return matrix
```

Gambar 21. Proses Data Centering
Sumber: Dokumen Penulis / Lampiran

Untuk implementasi ketiga, dapat dilihat seperti pada program berikut. Proses SVD dilakukan dengan membuat matriks kovarian terlebih dahulu dengan menghitung matriks awal A menjadi matriks A^TA yang kemudian dibagi dengan banyak database. Lalu, dilanjutkan dengan menghitung SVD dengan library bantuan dari numpy untuk mendapat matriks U, Σ, dan V^T.

```python
def make_covarians(matrix : list[list[float]]) -> list[list[float]] :
    # DESKRIPSI LOKAL
    # Membuat matriks covarians berdasarkan spesifikasi yang diberikan.

    # KAMUS LOKAL
    # matrix , covarians , transposed = matrix of float;
    # n , size : integer
    # i , j : integer (index)

    # ALGORITMA LOKAL
    n = len(matrix)
    transposed = transpose(matrix)
    covarians = multiply(transposed , matrix)
    size = len(covarians)
    for i in range (size) :
        for j in range (size) :
            covarians[i][j] = covarians[i][j] / n
    return covarians
```

Gambar 22. Proses Buat Matriks Kovarian
Sumber: Dokumen Penulis / Lampiran

```python
def svd_decomposition(covarians : list[list[float]]) -> tuple[list[list[float]] , list[list[float]] , list[list[float]]] :
    # DESKRIPSI LOKAL
    # Melakukan SVD (Singular Value Decomposition) menggunakan fungsi bantuan dari numpy.

    # KAMUS LOKAL
    # matrix , covarians , eigen_vector , diagonal_matrix , singular_value : matrix of float

    # ALGORITMA LOKAL
    matrix = np.array(covarians)
    eigen_vector , diagonal_matrix , singular_value = np.linalg.svd(matrix)
    diagonal_matrix = np.diag(diagonal_matrix)
    return (eigen_vector , diagonal_matrix , singular_value)
```

Gambar 23. Proses Perhitungan SVD
Sumber: Dokumen Penulis / Lampiran

Untuk impementasi keempat, dapat dilihat pada program berikut. Proses reduksi noise dilakukan dengan menggunakan bantuan library numpy. Proses ini dilakukan dengan cukup sederhana. Idenya adalah memotong dengan nilai k yang tepat dan ditentukan.

```python
def reduksi_noise(image) :
    # DESKRIPSI LOKAL
    # Subprogram ini akan melakukan penghilangan noise berdasarkan nilai-nilai singularnya.

    # KAMUS LOKAL
    # image_float , U , S , VT , S_k : matrix of float
    # denoised_image : img
    # k : integer

    # ALGORITMA LOKAL
    image_float = image.astype(float)
    U , S , VT = np.linalg.svd(image_float , full_matrices = False)
    k = 30
    S_k = np.zeros_like(S)
    S_k[:k] = S[:k]
    S_k_matrix = np.diag(S_k)
    denoised_image = U @ S_k_matrix @ VT
    denoised_image = np.clip(denoised_image , 0 , 255)
    denoised_image = denoised_image.astype(np.uint8)
    return denoised_image
```

Gambar 24. Proses Reduksi Noise
Sumber: Dokumen Penulis / Lampiran

Untuk implementasi kelima, dapat dilihat pada program berikut. Proses rekonstruksi citra dilakukan dengan menggunakan bantuan library cv2 dan matplotlib. Proses ini dilakukan cukup sederhana menggunakan fungsi bawaan untuk menampilkannya.

```python
def rekonstruksi_citra(path : str) :
    # DESKRIPSI LOKAL
    # Subprogram ini akan merekonstruksi citra berdasarkan hasil proses reduksi noise yang dilakukan.

    # KAMUS LOKAL
    # image , denoised_image : img

    # ALGORITMA LOKAL
    image = cv2.imread(path , cv2.IMREAD_GRAYSCALE)
    denoised_image = reduksi_noise(image)
    plt.figure(figsize = (10 , 5))
    plt.subplot(1 , 2 , 1)
    plt.imshow(denoised_image , cmap = 'gray')
    plt.title('Denoised Image')
    plt.axis('off')
    plt.show()
```

Gambar 25. Proses Rekonstruksi Image
Sumber: Dokumen Penulis / Lampiran

Untuk implementasi keeenam, dapat dilihat pada program berikut. Proses perhitungan similaritas dilakukan dengan cara seperti yang sudah dijelaskan pada bab sebelumnya. Prosesnya dimulai dengan proyeksi *query* dan *database* ke layar PCA. Selanjutnya, perhitungan similaritas dan penyortiran dilakukan dan menghasilkan sebuah kumpulan data yang memenuhi standar.

```python
def proyeksi_query(query : list[float] , k_eigen_vector : list[list[float]] , matrix : list[list[float]]) -> list[list[float]] :
    # DESKRIPSI LOKAL
    # Membuat fungsi untuk memproyeksikan query ke ruang komponen utama (PCA).

    # KAMUS LOKAL
    # query , avg : list of float
    # matrix , matrix_query , k_eigen_vector : matrix of float
    # size : integer
    # i , j : integer (index)

    # ALGORITMA LOKAL
    avg = find_average(matrix)
    size = len(query)
    matrix_query = [[0.0 for j in range (size)] for i in range (1)]
    for i in range (size) :
        matrix_query[0][i] = query[i] - avg[i]
    res = multiply(matrix_query , k_eigen_vector)
    return res
```

Gambar 26. Proses Proyeksi Query
Sumber: Dokumen Penulis / Lampiran

```python
def get_z(matrix : list[list[float]] , k_eigen_vector : list[list[float]]) -> list[list[float]] :
    # DESKRIPSI LOKAL
    # Menghitung matrix Z berdasarkan perkalian matrix yang sudah distandarisasi dan eigen_vector sebanyak rank.

    # KAMUS LOKAL
    # matrix , k_eigen_vector , res : matrix of float
    # i , j : integer (index)

    # ALGORITMA LOKAL
    matrix = data_centering(matrix)
    res = multiply(matrix , k_eigen_vector)
    return res
```

Gambar 27. Proses Proyeksi Database
Sumber: Dokumen Penulis / Lampiran

```python
def count_function(query : list[float] , data : list[float]) -> float :
    # DESKRIPSI LOKAL
    # Fungsi untuk menghitung proyeksi jarak euclidian.

    # KAMUS LOKAL
    # query , data : list of float
    # sumq , res : float
    # i : integer (index)

    # ALGORITMA LOKAL
    sumq = 0.0
    size = len(query)
    for i in range (size) :
        sumq = sumq + (query[i] - data[i]) * (query[i] - data[i])
    res = sqrt(sumq)
    return res
```

Gambar 28. Proses Hitung Similaritas
Sumber: Dokumen Penulis / Lampiran

```python
def minimum_similarity(data : list[list[float]] , target : float) -> tuple[list[list[float]] , list[float]] :
    # DESKRIPSI LOKAL
    # Membatasi similaritas diatas target yang ditentukan untuk

    # KAMUS LOKAL
    # data , res : matrix of float
    # target , bar , count : float
    # size : integer
    # i : integer (index)

    # ALGORITMA LOKAL
    res = []
    size = len(data)
    bar = data[size - 1][1]
    percent = []
    for i in range (size) :
        count = 1 - data[i][1] / bar
        if (count >= target) :
            res.append(data[i])
            percent.append(count)
    return (res , percent)
```

Gambar 29. Proses Pemilihan Standar Similaritas
Sumber: Dokumen Penulis / Lampiran

```python
def jarak_euclidean(query : list[float] , matrix : list[list[float]]) -> tuple[list[tuple[int , float]] , list[float]] :
    # DESKRIPSI LOKAL
    # Menghitung jarak euclidean antara query dengan setiap data di database dan mengurutkannya.

    # KAMUS LOKAL
    # matrix , matrix_z , k_eigen_vector , matrix_query : matrix of float
    # data , distance : list of tuple of integer and float
    # query : list of float
    # size : integer
    # i : integer (index)

    # ALGORITMA LOKAL
    k_eigen_vector = get_keigen(matrix)
    matrix_z = get_z(matrix , k_eigen_vector)
    matrix_query = proyeksi_query(query , k_eigen_vector , matrix)
    size = len(matrix_z)
    distance = [[0 , 0.0] for i in range (size)]
    for i in range (size) :
        distance[i][0] = i + 1
        distance[i][1] = count_function(matrix_query[0] , matrix_z[i])
    target = 0.55
    data = merge_sort(distance)
    res , percent = minimum_similarity(data , target)
    return (res , percent)
```

Gambar 30. Kumpulan Hasil Penyortiran
Sumber: Dokumen Penulis / Lampiran

Dengan seluruh implementasi tersebut, program ini dapat berjalan dengan baik sesuai harapan. Tentunya, perlu ada banyak penyesuaian tambahan yang menghubungkan antar subprogram fungsi-fungsi tersebut. Keseluruhan implementasi program lengkap dapat dilihat pada source code di lampiran.

V. KESIMPULAN

Setelah melakukan rangkaian penelitian, penulis dapat menyimpulkan pembahasan atas berbagai percobaan dan pengamatan yang telah dilakukan sebagai berikut:

1. Penerapan materi aljabar linier dan geometri untuk memberikan optimalisasi hasil diagnostik dengan mengimplementasi materi SVD pada pencitraan medis terbukti sangat banyak dan bermanfaat.
2. SVD terbukti merupakan suatu teknik yang dapat melakukan pengolahan citra dengan baik, seperti untuk melakukan kompresi, penghilangan noise, merekonstruksi citra, dan lainnya.
3. Impelementasi SVD dalam pencitraan medis telah membuat diagnosis yang dilakukan oleh para tenaga medis menjadi lebih optimal.
4. Perkembangan dunia teknologi medis kedepannya akan membuat dunia medis lebih baik secara perlahan dan terus menerus.

VI. UCAPAN TERIMA KASIH

Alhamdulillah, serangkaian proses penyusunan makalah yang berjudul " Optimalisasi Hasil Diagnostik: Implementasi Singular Value Decomposition (SVD) dalam Pencitraan Medis" di mata kuliah Aljabar Linier dan Geometri telah penulis selesaikan dengan baik.

Penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada Bapak Ir. Rila Mandala, M.Eng., Ph.D., sebagai dosen dan pembimbing yang telah memberikan arahan dan dukungan dalam penyusunan makalah ini. Terima kasih juga kepada semua pihak yang telah membantu dalam proses penyelesaian laporan ini, meskipun tidak dapat penulis sebutkan satu persatu.

Penulis berharap semoga hasil penulisan makalah ini dapat memberikan manfaat bagi kita semua dan mendorong kemajuan ilmu pengetahuan di masa yang akan datang. Di sisi lain, penulis menyadari bahwa makalah ini masih jauh dari sempurna. Oleh karena itu, penulis terbuka dan menghargai segala kritik dan saran yang konstruktif. Sekian dan terima kasih.

REFERENSI

[1] Munir, Rinaldi. (2024). Pengantar Aljabar Linier & Geometri. 2 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Algeo-00-Pengantar-Aljabar-Geometri-(2024).pdf

[2] Munir, Rinaldi. (2024). Review Matriks. 1 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-01-Review-Matriks-2023.pdf

[3] Munir, Rinaldi. (2024). Matriks Eselon. 1 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-02-Matriks-Eselon-2023.pdf

[4] Munir, Rinaldi. (2024). Sistem Persamaan Linier (SPL) Bagian 1. 1 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-03-Sistem-Persamaan-Linier-2023.pdf

[5] Munir, Rinaldi. (2024). Sistem Persamaan Linier (SPL) Bagian 2. 1 Januari 2025, dari 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 Linier (SPL) Bagian 3. 1 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-05-Sistem-Persamaan-Linier-2-2023.pdf

[7] Munir, Rinaldi. (2024). Nilai Eigen dan Vektor Eigen Bagian 1. 1 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian1-2023.pdf

[8] Munir, Rinaldi. (2024). Nilai Eigen dan Vektor Eigen Bagian 2. 1 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-20-Nilai-Eigen-dan-Vektor-Eigen-Bagian2-2023.pdf

[9] Munir, Rinaldi. (2024). Singular Value Decomposition (SVD) Bagian 1. 1 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-21-Singular-value-decomposition-Bagian1-2023.pdf

[10] Munir, Rinaldi. (2024). Singular Value Decomposition (SVD) Bagian 2. 1 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-22-Singular-value-decomposition-Bagian2-2023.pdf

[11] Munir, Rinaldi. (2024). Dekomposisi LU. 1 Januari 2025, dari

https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-23-Dekomposisi-LU-2023.pdf
[12] Munir, Rinaldi. (2024). Dekomposisi QR. 1 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Algeo-23b-Dekomposisi-QR-2024.pdf
[13] Munir, Rinaldi. (2024). Aplikasi Dot Product pada Sistem Temu-balik Informasi. 1 Januari 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-14-Aplikasi-dot-product-pada-IR-2023.pdf
[14] Adan, Abdijibar M. (2022). A study of evaluation and proper diagnosis of stroke in CT scan and MRI. 1 Januari 2025, dari https://doi.org/10.3306/AJHS.2022.37.06.130

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, 2 Januari 2025

Muhammad Raihan Nazhim Oktana
13523021

LAMPIRAN

[1] Source Code: https://github.com/RNXFreeze/Makalah-IF2123-AljabarLinierDanGeometri