Deteksi Objek Dominan pada Citra dengan Menggunakan Konsep Aljabar Linear dan Geometri

Nathan Adhika Santosa dan 13524041[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13524041@std.itb.ac.id, [2]nathan.santosa37@gmail.com

***Abstrak*—Deteksi objek utama adalah tantangan paling fundamental dalam bidang *computer vision*. Aljabar Linear dan Geometri sangat berhubungan erat dengan *computer vision*, sehingga diperlukan pemahaman lebih mendalam mengenai deteksi objek dengan konsep aljabar linear dan geometri. Makalah ini membahas hubungan dan implementasi dari konsep aljabar linear dan geometri pada program deteksi objek dengan sepenuhnya menggunakan aljabar linear dan geometri, model deteksi objek *classical* (HOG + SVM), dan model deteksi objek *modern* (YOLO). Meskipun tidak dikatakan secara eksplisit, konsep aljabar linear dan geometri sangat penting untuk keberhasilan program atau model deteksi objek apapun.**

***Kata Kunci*—Deteksi Objek, Matriks, SVD, Computer vision**

I. Pendahuluan

Perkembangan teknologi pengolahan citra digital dan visi komputer semakin pesat seiring berjalannya waktu, sebab terdapat peningkatan kebutuhan untuk sistem cerdas yang mampu memahami informasi visual secara otomatis. Salah satu permasalahan penting dalam bidang ini adalah deteksi objek dominan pada citra, yaitu proses mengidentifikasi objek utama yang paling signifikan dalam sebuah gambar. Deteksi objek dominan banyak diterapkan pada berbagai bidang, seperti pengenalan pola baru, sistem pengawasan, identifikasi penyakit, kendaraan otonom, hingga analisis gambar pada satelit.

Dalam pencarian deteksi objek dominan, citra digital seringkali direpresentasikan dalam bentuk matriks, sehingga deteksi objek dominan berkaitan erat dengan konsep aljabar linear. Selain itu, operasi-operasi aljabar linear, seperti transformasi matriks, vektor, dekomposisi matriks, serta analisis ruang vektor juga berperan penting untuk mengekstraksi, perhitungan, dan merepresentasikan karakteristik objek dalam citra.

Oleh karena itu, penelitian ini mengenai deteksi objek dominan pada citra bertujuan untuk melihat dan mengimplementasikan konsep aljabar linear dan geometri pada deteksi objek. Penelitian ini diharapkan dapat memberikan pemahaman yang lebih kuat mengenai peran konsep matematika dalam pengolahan citra.

II. Landasan Teori

A. Matriks

Matriks adalah kumpulan angka yang disusun berdasarkan baris dan kolom, dimana ditempatkan di dalam kurung. Matriks banyak digunakan dalam bidang informatika, seperti graf, citra digital, dll.

- Operasi Matriks
  Operasi Penjumlahan dan pengurangan
  Jika matriks A dan matriks B berukuran sama, yaitu mxn(m jumlah kolom dan n jumlah baris). Maka, penjumlahan kedua matriks tersebut akan menghasilkan matriks baru.

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

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

Gambar 1.1 dan 1.2 : Dua matriks dan hasil penjumlahan matriks (https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-01-Review-Matriks-2025.pdf)

Operasi Perkalian
Jika sebuah matriks A berukuran mxn dikalikan dengan sebuah bilangan x, maka perkalian tersebut akan menghasilkan matriks baru dengan ukuran sama dengan matriks A. Isi dari matriks baru adalah isi dari matriks A namun dikalikan dengan x. Jika matriks A dengan ukuran mxn dikalikan dengan matriks B berukuran nxp, maka perkalian tersebut akan menghasilkan matriks baru berukuran mxp.

$$AB = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$= \begin{bmatrix} 1(5) + 2(7) & 1(6) + 2(8) \\ 3(5) + 4(7) & 3(6) + 4(8) \end{bmatrix}$$

$$= \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

Gambar 1.3 : Hasil perkalian matriks (https://www.kompas.com/skola/read/2022/12/06/073000769/perkalian-matriks)

- Matriks Transpose
  Operasi Transpose yang dilakukan pada matriks A berukuran mxn akan menghasilkan A^T berukuran nxm

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}$$

$$A^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \\ a_{14} & a_{24} & a_{34} \end{bmatrix},$$

Gambar 1.4 : Hasil transpose sebuah matriks (https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-01-Review-Matriks-2025.pdf)

B. Determinan

Notasi dari determinan adalah |a|, dimana a adalah matriks dengan ukuran mxn.

- Determinan Matriks 2x2
  Jika terdapat sebuah matriks A berukuran 2x2 yang berisi

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Gambar 1.5 : Matriks A (https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-01-Review-Matriks-2025.pdf)
, maka determinan dari matriks A adalah a.d - b.c.

- Determinan Matriks mxm
  Jika terdapat sebuah matriks A berukuran mxm dan m > 2

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

Gambar 1.6: Sebuah matriks (https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-01-Review-Matriks-2025.pdf)
, maka determinan matriks A adalah sebagai berikut

$$\det(A) = a_{11}C_{11} + a_{12}C_{12} + \dots + a_{1n}C_{1n}$$
$$\det(A) = a_{21}C_{21} + a_{22}C_{22} + \dots + a_{2n}C_{2n}$$
$$\vdots$$
$$\det(A) = a_{n1}C_{n1} + a_{n2}C_{n2} + \dots + a_{nn}C_{nn}$$

Secara baris

$$\det(A) = a_{11}C_{11} + a_{21}C_{21} + \dots + a_{n1}C_{n1}$$
$$\det(A) = a_{12}C_{12} + a_{22}C_{22} + \dots + a_{n2}C_{n2}$$
$$\vdots$$
$$\det(A) = a_{1n}C_{1n} + a_{2n}C_{2n} + \dots + a_{nn}C_{nn}$$

Secara kolom

Gambar 1.7 : Rumus determinan sebuah matriks (https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-08-Determinan-bagian1-2025.pdf)
Determinan dapat dihitung dengan salah satu rumus tersebut. Dimana,

$$C_{ij} = (-1)^{i+j}.M_{ij}$$

M_{ij} disini artinya determinan dari submatriks, dimana elemennya adalah elemen selain baris i dan kolom j.

C. Vektor

Vektor adalah konsep pada matematika, fisika, dan ilmu komputer yang merepresentasikan besaran dan arah. Vektor biasanya memiliki notasi huruf yang dimiringkan dan memiliki tanda panah diatas huruf tersebut.

- Penjumlahan Vektor
  Jika terdapat vektor $v = (v_1, v_2, \dots, v_n)$ dan $u = (u_1, u_2, \dots, u_n)$, maka penjumlahan vektor tersebut akan menghasilkan vektor $w = (v_1+u_1, v_2+u_2, \dots, v_n+u_n)$

- Norma Vektor
  Norma Vektor disebut juga sebagai norma euclidean. Norma menghasilkan panjang dari sebuah vektor. Jika terdapat sebuah vektor $v = (v_1, v_2, \dots, v_n)$, maka norma dari vektor v adalah

$$||v|| = \sqrt{(v_1^2+v_2^2+...+v_n^2)}$$

- Jarak Euclidean
  Perhitungan Jarak Euclidean menghasilkan jarak antara dua vektor. Jika terdapat vektor $A = (a_1, a_2, \dots, a_n)$ dan vektor $B = (b_1, b_2, \dots, b_n)$, maka jarak euclidean dari vektor A dan vektor B adalah

$$||A-B|| = \sqrt{((a_1-b_1)^2+(a_2-b_2)^2+...+(a_n-b_n^2))}$$

- Perkalian Titik
  Perkalian titik atau disebut juga dot product digunakan untuk mencari hasil skalar dari perkalian dua vektor. Jika terdapat vektor $u = (u_1, u_2, \dots, u_n)$ dan vektor $v = (v_1, v_2, \dots, v_n)$, maka

$$u.v = ||u||||v||.\cos\theta$$

atau

$$u.v = u_1v_1+u_2v_2+...+u_nv_n$$

- Perkalian Silang
  Perkalian silang atau disebut juga cross product digunakan untuk mencari vektor yang bertegak lurus dengan dua vektor. Jika terdapat vektor u = (u_1, u_2, u_3) dan vektor v = (v_1, v_2, v_3), maka

$$\mathbf{u} \times \mathbf{v} = (\begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix}, -\begin{vmatrix} u_1 & u_3 \\ v_1 & v_3 \end{vmatrix}, \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix})$$

Gambar 1.8 : Hasil dari perkalian silang (https://informatika.stei.itb.ac.id/~rinaldi.munir/)

- Vektor Satuan
  Jika terdapat vektor u = (u_1, u_2, …, u_n), maka v yang merupakan vektor satuan dari vektor u adalah

$$v = u/||u||$$

- Ortogonal
  Sebuah vektor u dan v dikatakan saling ortogonal jika u.v menghasilkan nol

D. Nilai Eigen & Vektor Eigen

Jika terdapat matriks A dengan ukuran nxn, maka vektor tak nol x adalah vektor eigen dari A. Vektor eigen didefinisikan sebagai suatu matriks kolom yang apabila dikalikan dengan matriks *A* akan menghasilkan vektor yang merupakan kelipatan vektor itu sendiri dengan nilai eigen.

$$A.x = \lambda.x$$

Eigen berasal dari bahasa Jerman yang memiliki arti asli atau karakteristik, sehingga nilai eigen mengatakan karakteristik dari sebuah matriks nxn.

Dari persamaan eigen, dapat dikalikan kedua ruas dengan I yang merupakan matriks identitas berukuran nxn, lalu kedua ruas dikurangi dengan AxI sehingga menjadi persamaan berikut

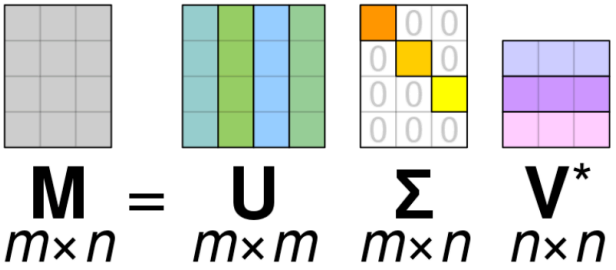
$$(\lambda.I - A)x = 0$$

Dari persamaan baru tersebut, dapat dibuat persamaan karakteristik dengan menggunakan determinan kedua ruas. Persamaan karakteristik ini digunakan untuk mencari nilai eigen dan dari nilai eigen tersebut dapat digunakan juga untuk mencari vektor eigen.

E. Singular Value Decomposition (SVD)

Singular Value Decomposition adalah salah satu metode untuk memfaktorkan sebuah matriks menjadi hasil kali dari sejumlah matriks. Jika terdapat sebuah matriks A berukuran mxn, maka SVD akan memfaktorkan matriks A menjadi matriks U, Σ, dan V sehingga

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



Gambar 1.9 : Dekomposisi matriks M dengan SVD (https://informatika.stei.itb.ac.id/~rinaldi.munir/)

U = matriks ortogonal mxm
Σ = matriks ukuran mxn, dimana berisi nilai singular dari matriks A^TA
V^T = matriks ortogonal nxn

Matriks ortogonal disini berarti setiap vektor pembentuk matriks saling ortogonal dengan vektor lain. Salah satu cara untuk mencari matriks dekomposisi adalah sebagai berikut

1. Mencari nilai eigen dan vektor eigen dari matriks A^TA
2. Mencari nilai singular dari nilai eigen, dimana nilai singular adalah akar kuadrat setiap nilai eigen tak nol
3. Membentuk matriks Σ, dimana setiap slot terisi angka nol kecuali slot diagonal yang terisi nilai singular mulai dari nilai terbesar
4. Setiap vektor eigen yang dibentuk oleh nilai eigen dinormalisasikan dan digunakan untuk membentuk matriks, dimana vektor eigen dengan nilai eigen terbesar diletakkan kiri matriks. Matriks tersebut ditranspose untuk membentuk V^T
5. Mencari vektor-vektor pembentuk matriks U, dimana vektor u_i adalah Av_i dibagi dengan σ_i. Lalu setiap vektor u dinormalisasikan dan diurutkan untuk membentuk matriks U

SVD tereduksi secara sederhananya adalah dekomposisi sebuah matriks dengan ukuran yang lebih kecil dari SVD biasa. Matriks Σ direduksi menjadi matriks persegi Σ_k dengan membuang baris atau kolom yang tidak mengandung nilai singular. Matriks U dan V^T direduksi dengan membuang vektor-vektor pembentuk matriks dari indeks terbesar jika jumlah vektor-vektor pembentuk melebihi jumlah nilai singular.

F. Object Detection

Object detection merupakan aktivitas *computer vision* yang memiliki tugas untuk mendeteksi objek-objek di dalam citra digital. Deteksi objek memiliki banyak aplikasi di dunia nyata. Salah satunya adalah robot, dimana deteksi objek digunakan sehingga robot dapat beroperasi secara otonom dengan mengidentifikasi objek-objek dan melakukan kalkulasi dari objek-objek yang telah diidentifikasi. Selain itu, deteksi objek digunakan juga untuk sistem deteksi kejahatan otomatis

dengan mendeteksi objek yang dibawa manusia sehingga dapat memprediksi dan mencegah kejahatan.

Untuk melakukan deteksi objek, dilakukan terlebih dahulu langkah *image processing*. *Image processing* mengubah masukan citra menjadi format yang sesuai dengan model atau algoritma yang akan digunakan. Sebab, citra digital merupakan matriks berisi angka yang merepresentasikan warna, kecerahan, dan intensitas.

Setelah melakukan *image preprocessing*, model atau algoritma yang dipilih akan menerima citra digital yang telah dilakukan *image processing* sehingga model dapat mendeteksikan objek yang terdapat pada citra digital.

Terdapat banyak model atau algoritma yang dapat digunakan untuk deteksi objek, seperti YOLO (You Only Look Once) dan Histogram of Oriented Gradient (HOG) + Support Vector Machine (SVM). Model atau algoritma deteksi objek terbagi menjadi *classical object detection* dan *modern object detection*.

G. Classical Object Detection

Classical Object Detection menggunakan prinsip-prinsip matematika, pengolahan citra, serta teknik pembelajaran mesin klasik untuk mendeteksi objek. Terdapat banyak macam model untuk deteksi objek klasik, seperti Histogram of Oriented Gradient (HOG) + Support Vector Machine (SVM). HOG + SVM memerlukan dataset untuk bahwa ini adalah objek yang kamu carikan dalam foto agar bisa melakukan deteksi objek. Dataset ini, terdiri dataset positif yang hanya mengandung objek yang ingin dideteksi dan dataset negatif yang mengandung objek selain objek yang dicarikan.

1. Histogram of Oriented Gradient (HOG)
   HOG adalah deskriptor fitur, dimana digunakan untuk menangkap struktur dengan menganalisis distribusi orientasi gradien. Berikut adalah langkah-langkah untuk melakukan HOG
   1. Preprocessing
      Pada langkah ini, citra digital diubah menjadi grayscale dan ukurannya diubah menjadi 64x32 pixel
   2. Komputasi Gradient
      Seluruh sel diubah menjadi matriks dengan setiap sel matriks berisi pixel 4x4. Magnitude gradient dicarikan dengan menggunakan rumus

      $$\sqrt{(G_x^2 + G_y^2)}$$

      Dimana, G_x adalah pengurangan sel sesudah dan sebelum sel pada axis-x. Serta, begitu juga untuk G_y.
      Lalu, dicarikan juga orientasi dari gradient dengan rumus

      $$\text{Orientasi} = \arctan(G_y/G_x)$$

   3. Pembuatan Histogram Orientasi
      Citra digital dibagi menjadi sel, dimana setiap sel memiliki 8x8 pixel. Lalu, histogram dibangun dengan membagi histogram menjadi 9 bagian (0, 20, .., 160 derajat), dimana setiap bagian adalah akumulasi magnitude dengan orientasi mendekati nilai setiap bagian
   4. Normalisasi
      Normalisasikan setiap vektor hasil dari pembuatan histogram
   5. Kontanisasi Fitur
      Setiap vektor yang telah dinormalisasikan akan di kontanisasi menjadi satu vektor.

2. Support Vector Machine (SVM)
   SVM digunakan untuk membedakan objek di dalam sebuah citra dengan mencari sebuah garis yang dapat membedakan kedua objek. Garis ini memiliki rumus

   $$w^T x + b = 0$$

   dimana, w adalah vektor normal terhadap garis dan b adalah bias dengan hyperplane. Untuk mencari jarak antara hyperplane dengan data, gunakan rumus berikut

   $$d_i = \frac{w^T x_i + b}{||w||}$$

   Gambar 2.1 : Jarak titik dengan garis (https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/)

   Lalu, untuk menentukan objek ini adalah objek yang ingin dicari, maka digunakan rumus berikut

   $$\hat{y} = \begin{cases} 1 & : w^T x + b \geq 0 \\ -1 & : w^T x + b < 0 \end{cases}$$

   Gambar 2.2 : Klasifikasi objek (https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/)

H. Modern Object Detection

Modern object detection memanfaatkan pendekatan *deep learning* dan pelatihan berbasis data dalam skala besar untuk menghasilkan sistem deteksi objek yang lebih kompleks dan akurat. Salah satu contoh terkenal dari *Modern Object Detection* adalah YOLO (You Only Look Once).

YOLO menggunakan CNN (Convolutional Neural Network) dan dataset(untuk melatih model mengenali objek-objek) sehingga bisa mendeteksi dan klasifikasi objek. CNN akan mengubah sebuah citra menjadi sebuah peta fitur sehingga dapat digunakan oleh YOLO untuk mendeteksi objek. CNN lakukan ini dengan melakukan operasi Convolutional, yaitu

$$(I*K)(i,j)=\Sigma_m\Sigma_n I(i+m,j+n).K(m,n)$$

dimana I adalah gambar input, K adalah kernel atau melakukan filter(sub matrix pada gambar) selama transversal gambar, i dan j adalah koordinat output peta fitur.

Selain itu, CNN menggunakan stride dan pooling untuk mereduksi gambar menjadi peta fitur. Slide mengatakan berapa pixel pergerakan selama filter, dimana slide yang lebih besar akan menghasilkan peta fitur yang lebih kecil. Pooling mencari angka maksimal dari submatriks pada hasil operasi Convolutional. Operasi Convolutional dan pooling dilakukan berulang-ulang untuk menghasilkan peta fitur.

Setelah melakukan CNN, YOLO akan membagi gambar menjadi beberapa kotak dan mencari kotak yang memiliki objek. Satu kotak hanya bisa berisi dengan satu objek jika ada. YOLO menggunakan fitur map untuk mencari objek di kotak-kotak ini. YOLO setelah melakukan pencarian akan mendapatkan probabilitas, titik pusat, dan ukuran *bounding box*. *Bounding box* dengan probabilitas di atas batasan yang ditentukan akan digunakan. Setelah itu, YOLO akan melakukan operasi *Intersection over Unions* untuk mendapatkan satu *bounding box* untuk satu objek. IoU memiliki rumus berikut.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Gambar 2.3 : Jarak titik dengan garis (https://www.v7labs.com/blog/yolo-object-detection)

Setelah kalkulasi IoU, *bounding box* diatas batas yang ditentukan yang akan digunakan untuk satu objek.

III. Pembahasan

Untuk melakukan analisis mengenai penggunaan aljabar linear dan geometri dalam deteksi objek utama, diperlukan contoh untuk mempermudah penjelasan

A. Deteksi Objek dengan konsep Aljabar Linear dan Geometri penuh

```python
import numpy as np
import math
from PIL import Image

# Fungsi-fungsi
def mean(M):
    s = 0.0
    cnt = 0
    for row in M:
        for x in row:
            s += x
            cnt += 1
    return s / cnt
def euclidean(v1, v2):
    s = 0.0
    for i in range(len(v1)):
        d = v1[i] - v2[i]
        s += d * d
    return math.sqrt(s)
def svd(A, k):
    ATA = A.T @ A
    eigenvalues, eigenvectors = np.linalg.eig(ATA)
    idx = np.argsort(eigenvalues)[::-1]
    eigenvalues = eigenvalues[idx]
    eigenvectors = eigenvectors[:, idx]

    U, S, V = [], [], []

    for i in range(k):
        lambda_i = eigenvalues[i]
        if lambda_i < 1e-9:
            break
        sigma = math.sqrt(lambda_i)
        v = eigenvectors[:, i]
        u = (A @ v) / (sigma + 1e-9)
        U.append(u)
        S.append(sigma)
        V.append(v)
    return np.array(U), np.array(S), np.array(V)
```

Gambar 3.1 : Kode bagian Fungsi

Untuk metode deteksi objek dengan menggunakan konsep aljabar linear dan geometri penuh, saya membuat implementasi dengan menggunakan bahasa pemrograman Python. Pada gambar di atas, saya membuat beberapa fungsi untuk mencari nilai rata-rata, jarak euclidean, dan mencari hasil dari dekomposisi SVD reduksi, berupa matriks U_k, Σ_k, dan V_k. Pencarian nilai eigen dan vektor eigen dilakukan menggunakan library numpy, sebab kesulitan pencarian nilai eigen dan vektor eigen tanpa library.

```python
img = Image.open("image.jpg").convert("L")
A = np.asarray(img, dtype=float)
h, w = A.shape
A_list = A.tolist()
A_mean = mean(A_list)
A_centered = A - A_mean

k = 8
U, S, V = svd(A_centered, k)

features = np.zeros((h, w, k))
for i in range(len(S)):
    features[:, :, i] = S[i] * np.outer(U[i], V[i])

F = features.reshape(-1, k)
centroid = [0.0] * k

for j in range(k):
    for i in range(F.shape[0]):
        centroid[j] += F[i][j]
    centroid[j] /= F.shape[0]

centroid = np.array(centroid)
dist = []
for i in range(F.shape[0]):
    dist.append(euclidean(F[i], centroid))

dist = np.array(dist)
dist_img = dist.reshape(h, w)
threshold = sum(dist) / len(dist)
mask = dist_img < threshold

objek1 = np.zeros_like(A)
objek2 = np.zeros_like(A)
objek1[mask] = A[mask]
objek2[~mask] = A[~mask]

Image.fromarray(np.array(objek1, dtype=np.uint8)).save("objek lain.png")
Image.fromarray(np.array(objek2, dtype=np.uint8)).save("objek dominan.png")

print("Selesai: Deteksi Objek Utama")
```

Gambar 3.2 : Kode bagian perhitungan dan hasil

Pada bagian ini kode python, image.jpg sebagai masukan untuk citra diubah terlebih dahulu dari RGB menjadi grayscale sehingga setiap sel di gambar hanya mengandung tingkat kecerahan dalam foto. Lalu, setiap sel dalam foto dikurangi dengan rata-rata dari seluruh sel untuk memastikan bahwa tidak terdapat bias terhadap intensitas cahaya.

Setelah itu, dicarikan U_k, Σ_k, V_k dengan menggunakan fungsi SVD. Lalu, dilakukan pembangunan ulang matriks foto dari ketiga matriks. Centroid akan berisi nilai rata-rata dari struktur global gambar, dimana menjadi penanda piksel mengikuti pola gambar. Jarak euclidian dicari antara setiap piksel dengan centeroid. Threshold dihitung dan setiap piksel ditentukan apakah piksel tersebut termasuk piksel yang mengandung objek utama.

Hasil akhir dari pemisahan objek dengan lainnya disimpan pada file objek dominan.jpg dan objek lain.jpg.

Gambar 3.3, 3.4, dan 3.5 : Gambar tes1, objek dominan, objek lain

Namun, perlu diperhatikan bahwa deteksi objek dengan menggunakan konsep aljabar linear dan geometri penuh hanya dapat dilakukan dengan akurat apabila variasi warna tidak terlalu banyak, seperti gambar yang ada di atas. Sebab, teknik yang digunakan disini tidak menggunakan machine learning, sehingga algoritma ini tidak mengetahui objek itu apa.

Gambar 3.6, 3.7, 3.8 : Gambar test 2, objek dominan 2, objek lain 2

Gambar pertama diambil dari pinterest dengan sengaja untuk tes kode terhadap gambar dengan variasi yang cukup tinggi. Diharapkan kakek yang sedang duduk di kursi dapat dideteksi oleh kode penggunaan penuh aljabar linear dan geometri, namun hanya sebagian saja yang terdeteksi.

B. Deteksi Objek dengan HOG + SVM

HOG dan SVM tidak sepenuhnya menggunakan konsep aljabar linear dan geometri. HOG menggunakan nilai gradient dan arah gradient untuk menentukan fitur yang terdapat di dalam sebuah gambar. Meskipun begitu, HOG masih menggunakan beberapa konsep aljabar linear. HOG menggunakan matriks untuk merepresentasikan gambar dan mengubah matriks tersebut menjadi matriks yang terdiri sel yang merupakan submatriks awal citra berukuran 4x4. Selain itu, digunakan juga vektor untuk merepresentasikan histogram dan kontanisasi fitur dari citra dan vektor satuan pada proses normalisasi.

SVM menggunakan konsep vektor untuk menerima vektor fitur dari HOG, serta menggunakan konsep tranpose pada vektor sebab vektor adalah matriks berukuran ax1 atau 1xa. Lalu, untuk mencari jarak antara titik dengan garis pemisah, digunakan norma vektor untuk membagi garis pemisah.

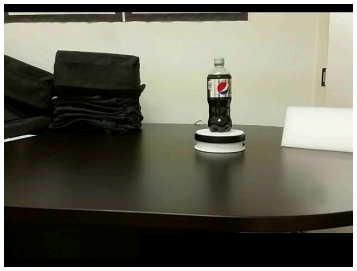
Untuk implementasi deteksi objek dengan HOG + SVM, penulis menggunakan github hist milik tejus-gupta. Namun, karena kode milik tejus-gupta sudah tidak mengikuti standar baru julia, maka saya mengubah beberapa bagian dari kodenya sehingga masih berfungsi

```julia
using Images, ImageFeatures, LIBSVM

pos_examples = "dataset/positive_images/"
neg_examples = "dataset/negative_images/"
pos_files = readdir(pos_examples)
neg_files = readdir(neg_examples)
n_pos = length(pos_files)
n_neg = length(neg_files)
n = n_pos + n_neg
examples = Array{Float64}(undef, n, 3780)
labels = Vector{Int}(undef, n)
all_files = [pos_files; neg_files]
global i = 0
for file in all_files
    global i += 1
    if i <= n_pos
        filename = joinpath(pos_examples, file)
        labels[i] = 1
    else
        filename = joinpath(neg_examples, file)
        labels[i] = 0
    end
    img = load(filename)
    img_resized = imresize(img, (128, 64))
    examples[i, :] = create_descriptor(img_resized, HOG())
end
println("Melatih SVM")
model = svmtrain(examples', labels)
img_test = load("5111.jpg")
img_test_resized = imresize(img_test, (128, 64))
des = create_descriptor(img_test_resized, HOG())
des_matrix = hcat(des)
predicted_label, _ = svmpredict(model, des_matrix)
println(predicted_label[1] == 1 ? "Diet Cola" : "Bukan Diet Cola")
```

Gambar 3.9 : Jarak titik dengan garis

Pada kode diatas, setiap gambar pada dataset positif dan negatif diubah ukurannya sehingga bisa digunakan HOG agar setiap gambar menghasilkan satu vektor berisi fitur gambar tersebut. Setelah itu, setiap gambar dataset diberikan label menandai apakah objek yang ada di gambar adalah objek yang dicarikan. Lalu, gambar yang diuji dilakukan hal yang sama dengan dataset sehingga dapat diketahui gambar uji mengandung objek yang dicarikan. *predicted_label* akan bernilai satu jika objek terdapat dalam gambar.



Gambar 3.10 : Gambar tes 1 diet cola (kaggle.com)

Gambar uji pertama diatas merupakan gambar diet cola diatas sebuah meja. Gambar tersebut adalah gambar uji pertama. Dataset positif berisi 850 gambar diet cola dan dataset negatif berisi 1001 gambar bukan diet cola, dimana dataset ini diambilkan dari kaggle. Ketika gambar uji pertama di tes, hasil akhir mengatakan bahwa gambar tersebut mengandung diet cola. Tes selanjutnya saya menggunakan gambar botol dan gambar kakek di taman yang sama dengan deteksi objek penuhnya menggunakan aljabar linear dan geometri. Untuk gambar botol, HOG+SVM mengatakan bahwa gambar tersebut terdapat diet cola, padahal tidak ada. Saat gambar kakek dites, model mengatakan bahwa tidak ada objek diet cola di dalam gambar, dimana itu adalah fakta.

Algoritma HOG+SVM memberi hasil yang lebih akurat daripada algoritma pertama, namun akurasinya masih akurasi rendah.

C. Deteksi Objek dengan YOLO

Untuk melakukan deteksi objek, saya menggunakan kode yang disediakan oleh website geeksforgeeks.org, dimana mereka menggunakan YOLO versi 5 untuk melakukan deteksi objek.

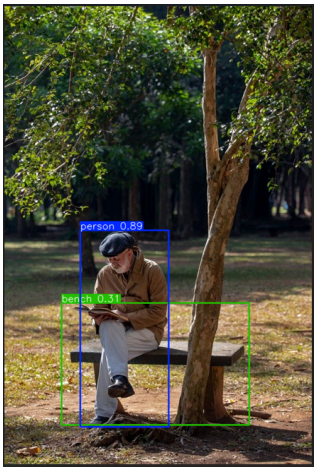
YOLO versi apapun menggunakan CNN, dimana CNN ini menggunakan matriks 3D untuk merepresentasikan bentuk gambar dan hasil CNN, serta matriks digunakan untuk filter setiap sel gambar. Lalu, operasi utama CNN, operasi Convolutional, menggunakan perkalian dot untuk mengalikan kernel dengan $I(i+m,j+n)$. YOLO sendiri hanya menggunakan matriks dari aljabar linear dan geometri, dimana YOLO menggunakan matriks untuk merepresentasikan gambar masukan dan keluarkan dan untuk menerima peta fitur dari CNN.

```python
import torch
from pathlib import Path
import cv2
import matplotlib.pyplot as plt
import numpy as np
# Load YOLOv5 model
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)
# Load image
img_path = 'image2.jpg'
img = cv2.imread(img_path)
# Convert BGR to RGB
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# Perform inference
results = model(img)
# Print results
results.print()  # Print results to console
results.show()   # Display results
# Get the results
detections = results.pandas().xyxy[0]  # Results as pandas dataframe

# Print detections
print(detections)
```

Gambar 3.11 : Jarak titik dengan garis

Diatas adalah kode yang telah diambil dari website geeksforgeeks. Di dalam kode tersebut, dilakukan load model terlebih dahulu. Lalu, diikuti dengan pembacaan gambar dan konversi gambar dari BGR menjadi RGB yang perbedaannya letak pada urutan warna saja. Setelah itu, gambar yang telah diubah menjadi RGB akan diberikan ke model sehingga mendapatkan gambar dengan objek yang telah dideteksi. Gambar yang digunakan untuk deteksi objek YOLO sama dengan gambar yang digunakan untuk deteksi objek penuh dengan konsep aljabar linear dan geometri. Berikut adalah hasilnya.



Gambar 3.12 dan 3.13 : Jarak titik dengan garis

Dari dua gambar diatas, dapat dibilang bahwa pendeteksian objek, baik di kondisi tanpa variasi banyak dan di kondisi variasi banyak, menghasilkan deteksi objek secara akurat. Botol, Kakek, dan kursi dapat dideteksi dan diidentifikasikan dengan baik oleh YOLO dibandingkan dengan deteksi kode pertama. Namun, YOLO masih salah mengidentifikasi botol sebagai vas.

IV. Kesimpulan

Berdasarkan penelitian yang telah dilakukan, aljabar linear dan geometri memiliki peran yang sangat penting dalam deteksi objek utama. Algoritma deteksi objek utama dengan penggunaan prinsip aljabar linear dan geometri penuh memberikan hasil yang baik jika variasi lingkungan tidak banyak, namun kurang di lingkup variasi banyak. Jenis model classical dan modern untuk deteksi objek utama memberikan akurasi yang lebih tinggi dan tidak menggunakan aljabar linear dan geometri sepenuhnya. Meskipun begitu, kedua jenis tersebut masih memiliki hubungan erat dengan aljabar linear dan geometri, terutama materi matriks dan vektor.

V. Lampiran

Berikut adalah tautan untuk program deteksi objek berbasis aljabar linear dan geometri murni.

https://github.com/N8NAS/Deteksi-Objek-Dominan-pada-Citra-dengan-Menggunakan-Konsep-Aljabar-Linear-dan-Geometri.git

Berikut adalah tautan untuk kode deteksi objek HOG+SVM oleh tejus-gupta.

https://gist.github.com/tejus-gupta/05e3392658461894f28ccd980328d9f8

Berikut adalah tautan untuk kode deteksi objek YOLO versi 5 dari website geeksforgeeks.

https://www.geeksforgeeks.org/computer-vision/how-does-yolo-work-for-object-detection/

VI. Ucapan Terima Kasih

Terima kasih kepada Tuhan Yang Maha Esa karena telah memberkati penulis sehingga penulis dapat menyelesaikan makalah ini tanpa mengalami kendala yang terlalu berat. Penulis juga ingin mengucapkan terima kasih kepada Dr. Ir. Rinaldi Munir, M.T. selaku dosen pengampu mata kuliah IF2123 kelas 1 karena telah membimbing dan mengajarkan kelas kami selama satu semester ini. Terakhir penulis juga ingin mengucapkan terima kasih kepada orang tua penulis karena telah memberi semangat kepada penulis selama proses pembuatan makalah ini.

Referensi

[1] Ruangguru, “Mengenal matriks dalam matematika: pengertian, jenis, dan transpose,” Ruangguru Blog. [daring]. https://www.ruangguru.com/blog/mengenal-matriks-dalam-matematika-pengertian-jenis-dan-transpose. Diakses: 17 Desember 2025.

[2] Ruangguru, “Konsep dasar vektor,” Ruangguru Blog. [daring]. https://www.ruangguru.com/blog/konsep-dasar-vektor. Diakses: 17 Desember 2025.

[3] BYJU’S, “Eigen values,” BYJU’S Maths. [daring]. https://byjus.com/maths/eigen-values/. Diakses: 18 Desember 2025.

[4] [4] Kompas.com, “Perkalian matriks,” Kompas Skola, Dec. 2022. [daring]. https://www.kompas.com/skola/read/2022/12/06/073000769/perkalian-matriks. Diakses: 18 Desember 2025.

[5] GeeksforGeeks, “Histogram of Oriented Gradients,” GeeksforGeeks. [daring]. https://www.geeksforgeeks.org/computer-vision/histogram-of-oriented-gradients/. Diakses: 23 Desember 2025.

[6] GeeksforGeeks, “Support Vector Machine algorithm,” GeeksforGeeks. [daring]. https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/. Diakses: 23 Desember 2025.

[7] GeeksforGeeks, “How does YOLO work for object detection?” GeeksforGeeks. [daring]. https://www.geeksforgeeks.org/computer-vision/how-does-yolo-work-for-object-detection/. Diakses: 23 Desember 2025.

[8] GeeksforGeeks, “How do convolutional neural networks (CNNs) work?” GeeksforGeeks. [daring]. https://www.geeksforgeeks.org/computer-vision/how-do-convolutional-neural-networks-cnns-work/. Diakses: 23 Desember 2025.

[9] [I. H. Gicheru, “Object detection,” Medium, 2020. [daring]. https://medium.com/@innohgicheru/object-detection-8da11f53dabf. Diakses: 23 Desember 2025.

[10] IBM, “What is object detection?” IBM Think. [daring]. https://www.ibm.com/think/topics/object-detection. Diakses: 23 Desember 2025.

[11] V7 Labs, “YOLO object detection,” V7 Labs Blog. [daring]. https://www.v7labs.com/blog/yolo-object-detection. Diakses: 23 Desember 2025.

[12] [1DataCamp, “YOLO object detection explained,” DataCamp Blog. [daring]. https://www.datacamp.com/blog/yolo-object-detection-explained. Diakses: 23 Desember 2025.

[13] GeeksforGeeks, “What is object detection in computer vision?” GeeksforGeeks. [daring]. https://www.geeksforgeeks.org/computer-vision/what-is-object-detection-in-computer-vision/. Diakses: 23 Desember 2025.

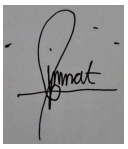
[14] R. Munir, “Review matriks,” Institut Teknologi Bandung, 2025. [daring]. https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/20

25-2026/Algeo-01-Review-Matriks-2025.pdf. Diakses: 20 Desember 2025.
[15] R. Munir, “Determinan (Bagian 1),” Institut Teknologi Bandung, 2025. [daring]. https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-08-Determinan-bagian1-2025.pdf. Diakses: 20 Desember 2025.
[16] R. Munir, “Vektor di ruang Euclidean (Bagian 1),” Institut Teknologi Bandung, 2025. [daring]. https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-11-Vektor-di-Ruang-Euclidean-Bag1-2025.pdf. Diakses: 20 Desember 2025.
[17] R. Munir, “Vektor di ruang Euclidean (Bagian 3),” Institut Teknologi Bandung, 2025. [daring]. https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-13-Vektor-di-Ruang-Euclidean-Bag3-2025.pdf. Diakses: 20 Desember 2025.
[18] [18] R. Munir, “Nilai eigen dan vektor eigen (Bagian 1),” Institut Teknologi Bandung, 2025. [daring]. https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian1-2025.pdf. Diakses: 20 Desember 2025.
[19] R. Munir, “Singular Value Decomposition (Bagian 1),” Institut Teknologi Bandung, 2025. [daring]. https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-21-Singular-value-decomposition-Bagian1-2025.pdf. Diakses: 20 Desember 2025.
[20] R. Munir, “Singular Value Decomposition (Bagian 2),” Institut Teknologi Bandung, 2025. [daring]. https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-22-Singular-value-decomposition-Bagian2-2025.pdf. Diakses: 21 Desember 2025.

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, 24 Desember 2025



Nathan Adhika Santosa
13524041