Simulasi Pergerakan Manipulator Robotik di Ruang Euclidean dengan Pendekatan Geometri Vektor Menggunakan Algoritma *Cyclic Coordinate Descent*

Kurt Mikhael Purba - 13524065
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13524065@std.stei.itb.ac.id, kurtmikhael123@gmail.com

***Abstract*— Invers Kinematika adalah sebuah konsep yang menyelesaikan masalah lengan robot dapat bergerak mencapai posisi target secara otomatis di ruang 3D. Makalah ini membahas penyelesaian masalah invers kinematika menggunakan aljabar vektor yang lebih efisien dibandingkan metode matriks konvensional. Makalah ini memanfaatkan algoritma *Cyclic Coordinate Descent* (CCD), untuk menghitung pergerakan setiap sendi secara iteratif melalui operasi *dot product* untuk menentukan sudut belok dan *cross product* untuk menentukan sumbu putaran lengan. Setelah mendapatkan hasil operasi vektor, proses perhitungan dilanjutkan dengan perhitungan mencari vektor rotasi dengan rumus Rotasi Rodrigues untuk menghasilkan pergerakan lengan yang tidak menyebabkan masalah *Gimbal Lock*. Makalah ini juga memberikan simulasi berbasis Python untuk mendemonstrasikan lengan robot yang mamp bergerak menuju target secara *real-time*.**

Keywords—Gimbal Lock, dot product, cross product, CCD

I. LATAR BELAKANG

Dalam perkembangan teknologi di bidang robotika, kemampuan lengan robot untuk bergerak menuju posisi target tertentu merupakan salah satu tantangan fundamental yang dikenal sebagai masalah invers kinematika. Pada ruang tiga dimensi, kompleksitas masalah ini meningkat seiring bertambahnya derajat kebebasan pada sendi robot. Pendekatan secara konvesional sangat bergantung pada metode matriks yang rumit, seperti inversi Jacobian, yang menuntut biaya komputasi tinggi dan rentan terhadap singularitas matematika. Oleh karena itu, diperlukan metode alternatif yang lebih efisien dan stabil untuk menangani pergerakan robot secara *real-time* tanpa mengorbankan akurasi.

Untuk mengatasi masalah ini, makalah ini menawarkan solusi invers kinematika dengan pendekatan aljabar vektor menggunakan algoritma *Cyclic Coordinate Descent*. Berbeda dari metode yang menyelesaikan semua persamaan sekaligus, CCD bekerja secara bertahap dan dimulai dari ujung lengan ke pangkal. Metode ini menggunakan operasi vektor sederhana yang ringan secara komputasi, seperti *dot product* untuk menghitung sudut koreksi dan cross product untuk menentukan sumbu putar secara dinamis di ruang tiga dimensi.

Untuk menjamin stabilitas rotasi, pergerakan sendi dalam makalah menggunakan implementasi rumus rotasi rodrigues. Penggunaan rumus ini penting karena memungkinkan rotasi pada sumbu rotasi tanpa risiko terjadinya fenomena *Gimbal Lock* yang sering ditemui pada metode rotasi Euler. *Gimbal Lock* merupakan suatu fenomena ketika terdapat dua atau lebih sumbu rotasi yang menyebabkan suatu objek berotasi pada arah yang sama. Efektivitas dari penggabungan metode aljabar vektor dan rumus Rodrigues ini kemudian diimplementasikan melalui simulasi kode dengan bahasa Python, yang mendemonstrasikan kemampuan lengan robot dalam beradaptasi mengejar target bergerak dengan responsif dan efisien.

II. LANDASAN TEORI

A. *Definisi dan Operasi Vektor*

Vektor merupakan suatu kuantitas fisik yang memiliki nilai dan arah yang direpresentasikan dengan kumpulan beberapa angka [1]. Vektori dapat direpresentasikan pada ruang dua dimensi yang direpresentasikan oleh $\begin{bmatrix} x \\ y \end{bmatrix}$ dan matriks tiga dimensi yang direpresentasikan oleh $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ [1]. Vektor-vektor tersebut memiliki aturan-aturan operasi sebagai berikut [1],

1. Penjumlahan Vektor
   Dua buah vektor yang berdimensi sama dapat saling dijumlahkan.

$$x + y = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

2. Pengurangan Vektor
   Dua buah vektor yang berdimensi sama dapat saling dikurangkan.

$$x - y = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

3. Perkalian dengan Skalar
   Sebuah vektor dapat dikalikan dengan sebuah nilai skalar.

$$cx = \begin{bmatrix} cx_1 \\ \vdots \\ cx_n \end{bmatrix}$$

4. Norma sebuah vektor

$$||v|| = \sqrt[2]{{v_1}^2 + \cdots + {v_n}^2}$$

5. Jarak Euclidean
   Jarak antara vektor a dan b dapat dihitung melalui rumus

$$d(a,b) = \sqrt[2]{(a_1 - b_1)^2 + \cdots + (a_n - b_n)^2}$$

B. Sifat-sifat Aljabar Vektor

Selain itu, vektor juga memiliki beberapa sifat-sifat, misalkan u dan v Adalah vektor, maka kedua vektor tersebut memenuhi sifat-sifat berikut [1].

1. u + v = v + u
2. (u + v) + w = u + (v + w)
3. u + 0 = 0 + u = u
4. u + (-u) = 0
5. k (u + v) = ku + kv
6. (u + v) k = ku + kv
7. k(mu) = km(u)
8. 1u = u

C. Vektor Satuan

Vektor satuan adalah vektor yang panjangnya 1. Vektor satuan sering dilambangkan sebagai u. Cara menghitung vektor satuan adalah sebagai berikut [1],

$$u = \frac{v}{||v||}$$

D. Perkalian Dot

Jika u dan v adalah vektor tidak nol pada ruang dua dan tiga maka perkalian titik (*dot product*) u dan v adalah [1]:

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

Jika u = (u_1, u_2, u_3) dan v = (v_1, v_2, v_3) adalah dua vektor maka perkalian dot antara vektor u dan v juga dapat dihitung seperti ini [1]:

$$u \,.\, v = (u_1 v_1) + (u_1 v_2) + \cdots + (u_n v_n)$$

Jika hasil perkalian dot antara dua vektor bernilai 0, kedua vektor tersebut dapat dikatakan saling tegak lurus (ortogonal) [2]. Kedua vektor yang panjangnya masing masing satu dan saling ortogonla disebut juga vektor ortonormal [2].

- Sifat-sifat perkalian dot:
  a. u . v = v . u
  b. u . (v + w) = u . v + u . w
  c. k (u . v) = (ku) . v
  d. v . v ≥ 0 dan v . v = 0 jika dan hanya jika v = 0
  e. 0 . v = v . 0 = 0
  f. (u + v) . w = u . w + v . w
  g. u . (v - w) = u . v – u .w
  h. (u – v) . w = u . w - v . w
  i. k (u . v) = u . (kv)

E. Perkalian Silang

Jika u = (u_1, u_2, u_3) dan v = (v_1, v_2, v_3) adalah dua vektor di R3 maka perkalian silang antara u dan v [3].

$$u \times v = \begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix} i - \begin{vmatrix} u_1 & u_3 \\ v_1 & v_3 \end{vmatrix} j + \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix} k$$

- Sifat – sifat perkalian silang [3]:
  - u x v = - (v x u)
  - u x (v + w) = (u x v) + (u x w)
  - (u + v) x w = (u x w) + v x w
  - k (u x v) = ku x v = u x (kv)
  - u x 0 = 0 x u = 0
  - u x u = 0

F. Hubungan Perkalian Silang dengan Perkalian Titik

- u . (u x v) = 0
- v . (u x v) = 0
- $||u \times v||^2 = ||u||^2||v||^2 - (u \,.\, v)^2$
- $u \times (v \times w) = (u \,.\, v)v - (u.v)w$
- (u x v) x w = (u . w) v – (v . w) u

G. Forward Kinematika (FK)

Forward Kinematika atau sering disebut kinematika maju merupakan suatu proses perhitungan posisi dan orientasi *end effector* robot berdasarkan sudut sendi yang dikeathui. Jika diberikan kumpulan sudut $\Theta_1, \Theta_2, \Theta_3, \ldots, \Theta_n$ dan kumpulan lengan $L_1, L_2, L_3, \ldots, L_n$, maka dapat dicari nilai vektor akhirnya dengan rumus:

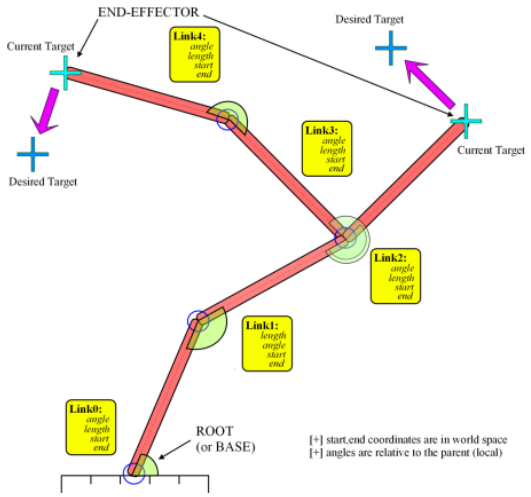
$$P_{end} = P_{base} + \sum_{i=1}^{n} v_i(\Theta_i)$$

H. Invers Kinematika (IK)

Dalam pembuatan lengan robot, Invers Kinematika berfungsi untuk menentukan konfigurasi sudut pada setiap sendi agar ujung lengan atau sering disebut *end effector* dapat mencapai koordinat target. Konsep ini berbeda dengan konsep Kinematika Maju yang menghitung posisi akhir berdasarkan input sudut yang diketahui [4].

Invers Kinematika membalik alur tersebut t dengan menyelesaikan persamaan matematis untuk mencari parameter sudut yang tidak diketahui berdasarkan posisi tujuan [4]. Representasi matematis sistem ini dalam ruang tiga dimensi sangat bergantung pada penggunaan vektor. Posisi setiap komponen sendi, P, didefinisikan sebagai vektor koordinat p = (x, y, z). Untuk melakukan perhitungan rotasi yang akurat pada setiap iterasi, posisi absolut tersebut harus terlebih dahulu dikonversi menjadi vektor relatif terhadap poros putar atau *pivot*, yang dirumuskan sebagai berikut:

$$v = P_{ujung} - P_{pivot}$$



Gambar 2.1 Ilustrasi masalah invers kinematika sederhana [4]

I. *Cyclic Coordinate Descent (CCD)*

Dalam menyelesaikan masalah IK, makalah ini menggunakan algoritma CCD [4]. Sebuah algoritma heuristik dan iteratif yang mengoptimalkan orientasi sendi satu per satu mulai dari ujung lengan sampai ke pangka *pivot*. Terdapat dua komponen utama yang terlebih dahulu dihitung oleh algoritma ini untuk menentukan pergerakan sebuah vektor. Komponen pertama yang dihitung adalah besar sudut rotasi antara vektor lengan ($\hat{u}_{current}$) saat ini dengan vektor target ($\hat{u}_{target}$) yang dapat dihitung dengan menggunakan perkalian titik [4].

$$\theta = \cos^{-1}(\hat{u}_{current} \cdot \hat{u}_{target})$$

Komponen kedua adalah vektor sumbu rotasi k yang tegak lurus terhadap bidang yang dibentuk oleh lengan dan target.

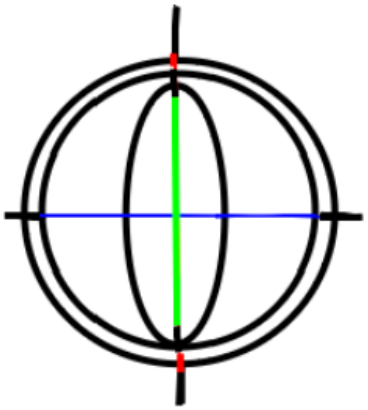
$$k = \hat{u}_{current} \times \hat{u}_{target}$$

$$k_{satuan} = \frac{k}{||k||}$$

J. *Gimbal Lock*

Gimbal lock adalah sebuah peristiwa hilangnya satu atau lebih derajat kebebasan pada suatu system rotasi tiga dimensi. Dalam representasi rotasi sudut Euler konvensional, rotasi total R dinyatakan sebagai hasil perkalian tiga matriks rotasi pada sumbu utama. Ketiga sumbu itu adalah sumbu X(*roll*),Y(*pitch*),dan Z(*yaw*).[4]

$$R = R_z(\delta).R_y(\delta).R_x(\delta)$$



Gambar 2.2 Ilustrasi *Gimbal Lock* [4]

Gimbal Lock terjadi ketika rotasi pada sumbu tengah (Dalam kasus di atas adalah sumbu y) bernilai $\frac{\pi}{2}$. Pada kondisi ini, $\cos(\frac{\pi}{2}) = 0$, yang menyebabkan sumbu rotasi pertama dan ketiga menjadi sejajar. Secara aljabar, matriks rotasi akan mengalami degenerasi sehingga perubahan pada *roll* dan *yaw* memberikan efek rotasi pada bidang yang sama. Hal ini menyebabkan sistem tidak mampu melakukan rotasi ke arah tertentu tanpa mengubah orientasi sumbu tengah terlebih dahulu, yang berakibat pada pergerakan robot yang tidak wajar dan diskontinu. Pendekatan *Axis-Angle* yang digunakan dalam rumus Rodrigues mengatasi hal ini dengan mendefinisikan rotasi sebagai putaran tunggal di sekitar vektor arbitrer *k*, sehingga tidak bergantung pada penyusunan sumbu yang hierarkis.

K. *Rotasi Rodrigues*

Dalam proses eksekusi rotasi pada sumbu k_{satuan}, makalah ini menggunakan rumus Rotasi Rodrigues [5]. Rumus ini digunakan karena algoritmanya yang efisien dan tidak menyebabkan terjadinya *Gimbal Lock* seperti algoritma euler. Rumus rotasi euler adalah:

$$v_{rot} = v \cos\theta + (k_{satuan} \times v) \sin\theta + k_{satuan}(k_{satuan} \cdot v)(1 - \cos\theta)$$

Setelah mendapatkan hasil berupa vektor rotasi, posisi vektor lengan dalam ruang tiga dimensi akan diperbaharui dengan menjumlahkan hasil rotasi dengan titik *pivot* awal dengan rumus:

$$P_{baru} = P_{pivot} + v_{rot}$$

III. IMPLEMENTASI

A. Fungsi Normalize

```
def normalize(v):
    norm = np.linalg.norm(v)
    if norm == 0:
        return v
    return v / norm
```

Gambar 3.1 Implementasi fungsi Normalize

Fungsi *normalize* merupakan fungsi yang menghitung vektor satuan dari sebuah vektor yang kemudian akan me-*return* vektor satuan tersebut.

B. Fungsi Rodrigues_rotation

```
def rodrigues_rotation(v, k, theta):
    v_rot = (v * np.cos(theta) +
             np.cross(k, v) * np.sin(theta) +
             k * np.dot(k, v) * (1 - np.cos(theta)))
    return v_rot
```

Gambar 3.2 Implementasi fungsi Rodrigues

Fungsi Rodrigues_rotation merupakan fungsi yang menghitung hasil vektor setelah dirotasikan menggunakan Rumus Rodrigues.

C. Kelas RobotArm3D

```
class RobotArm3D:
    def __init__(self, segment_lengths):
        self.n_joints = len(segment_lengths) + 1
        self.lengths = segment_lengths

        self.points = np.zeros((self.n_joints, 3))

        current_z = 0
        for i in range(1, self.n_joints):
            current_z += segment_lengths[i-1]
            self.points[i] = [0, 0, current_z]

    def solve_ik(self, target, max_iterations=5, tolerance=0.1):
        target = np.array(target)

        for _ in range(max_iterations):
            end_effector = self.points[-1]
            if np.linalg.norm(end_effector - target) < tolerance:
                break

            for i in range(self.n_joints - 2, -1, -1):
                pivot = self.points[i]
                end_pos = self.points[-1]

                v_current = end_pos - pivot
                v_target = target - pivot

                dist_curr = np.linalg.norm(v_current)
                dist_targ = np.linalg.norm(v_target)

                if dist_curr == 0 or dist_targ == 0:
                    continue

                u_current = v_current / dist_curr
                u_target = v_target / dist_targ

                dot_val = np.clip(np.dot(u_current, u_target), -1.0, 1.0)
                angle = np.arccos(dot_val)

                if angle < 1e-4:
                    continue


                axis = np.cross(u_current, u_target)
                axis = normalize(axis)


                for j in range(i + 1, self.n_joints):
                    rel_vec = self.points[j] - pivot

                    rot_vec = rodrigues_rotation(rel_vec, axis, angle)

                    self.points[j] = pivot + rot_vec
```

Gambar 3.3 Implementasi kelas RobotArm3D

Kelas ini merepresentasikan sebuah lengan robot 3D yang dapat bergerak dan menyesuaikan posisi untuk mencapai target menggunakan algoritma *Cyclic Coordinate Descent* (CCD). Pada kelas ini terdapat beberapa subfungsi yang akan digunakan selama eksekusi kode berlangsung. Fungsi __init_ berfungsi untuk menginisialisasi lengan robot dengan menerima parameter berupa daftar panjang setiap segmen. Fungsi ini menciptakan sejumlah joint yang sama dengan jumlah segment ditambah satu, kemudian menyimpan panjang setiap segmen dalam atribut *lengths*, dan membuat *array points* berukuran (n_joints, 3) untuk menyimpan koordinat tiga dimensi setiap joint. Inisialisasi dilakukan dengan menempatkan joint pertama di titik asal [0, 0, 0] dan joint-joint berikutnya ditempatkan secara vertikal sepanjang sumbu Z sesuai dengan akumulasi panjang segmen.

Selain itu, terdapat juga fungsi solve_ik. Fungsi ini berguna untuk menyelesaikan invers kinematika secara iteratif menyesuaikan setiap joint agar *end effector* dapat mencapai posisi target yang diberikan. Dalam setiap iterasi, fungsi ini melakukan loop dari joint terakhir ke belakang, dan untuk setiap joint, menghitung vektor dari joint tersebut ke *end effector* saat ini dan vektor dari joint ke target, kemudian mencari sudut rotasi yang diperlukan menggunakan arcosinus dari perkalian titik kedua vektor ternormalisasi. Setelah menemukan sudut rotasi, fungsi menghitung axis rotasi menggunakan perkalian silang dan selanjutnya merotasi semua joint yang berada di belakang pivot menggunakan formula Rodrigues untuk menyesuaikan posisi mereka. Proses ini berulang hingga end effector mencapai jarak tolerance dari target atau mencapai jumlah iterasi maksimum.

D. Fungsi Run_simulation_3D

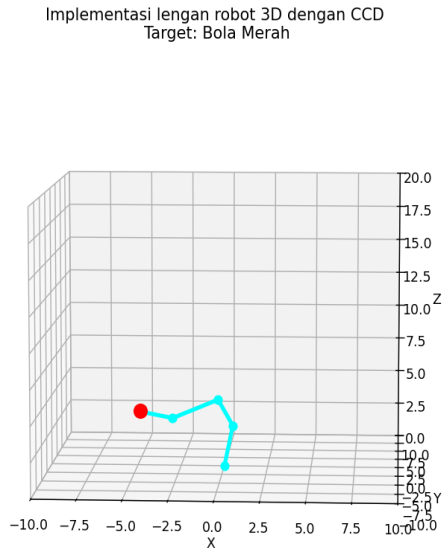
Fungsi run_simulation_3d() adalah fungsi utama yang menjalankan simulasi visual lengan robot 3D dengan animasi *real time*. Fungsi ini membuat lengan robot dengan 4 segmen, inisiasi plot 3D matplotlib dengan *line* untuk lengan robot dan target, serta mengatur batas-batas plot dengan *range* 10x10x20 satuan. Fungsi ini menjalankan animasi menggunakan FuncAnimation dengan 200 frame dan interval 50 milidetik.

Fungsi get_moving_target(t) adalah subfungsi yang menghitung posisi target yang bergerak mengikuti lintasan spiral 3D berdasarkan nilai frame t. Posisi X dan Y mengikuti lingkaran dengan radius 6, sementara Z bergerak naik turun berdasarkan fungsi sinus di sekitar ketinggian 5 satuan. Hasilnya merupakan target yang membentuk pola heliks yang dinamis untuk diikuti lengan robot.

```python
def run_simulation_3d():
    # Setup
    arm = RobotArm3D(segment_lengths=[3, 3, 3, 2]) # 4 Segmen

    fig = plt.figure(figsize=(10, 8))
    ax = fig.add_subplot(111, projection='3d')
    ax.set_title("Implementasi lengan robot 3D dengan CCD \nTarget: Bola Merah")


    line, = ax.plot([], [], [], 'o-', lw=3, markersize=6, color='cyan')
    target_point, = ax.plot([], [], [], 'ro', markersize=10) # Target Merah

    limit = 10
    ax.set_xlim(-limit, limit)
    ax.set_ylim(-limit, limit)
    ax.set_zlim(0, limit*2)
    ax.set_xlabel('X'); ax.set_ylabel('Y'); ax.set_zlabel('Z')

    def get_moving_target(t):
        x = 6 * np.cos(t * 0.05)
        y = 6 * np.sin(t * 0.05)
        z = 5 + 2 * np.sin(t * 0.1)
        return [x, y, z]

    def init():
        line.set_data([], [])
        line.set_3d_properties([])
        target_point.set_data([], [])
        target_point.set_3d_properties([])
        return line, target_point

    def update(frame):
        target_pos = get_moving_target(frame)

        arm.solve_ik(target_pos, max_iterations=5)

        xs = arm.points[:, 0]
        ys = arm.points[:, 1]
        zs = arm.points[:, 2]

        line.set_data(xs, ys)
        line.set_3d_properties(zs)

        target_point.set_data([target_pos[0]], [target_pos[1]])
        target_point.set_3d_properties([target_pos[2]])

        return line, target_point

    ani = FuncAnimation(fig, update, frames=np.arange(0, 200), init_func=init, interval=50, blit=False)
    plt.show()
```

Gambar 3.3 Implementasi fungsi run_animation_3d

IV. HASIL DAN DISKUSI



Gambar 4.1 Hasil Eksekusi Kode

Berdasarkan **Gambar 4.1** dapat dilihat bahwa program berhasil dieksekusi dan berhasil menampilkan implemenasi lengan robot tiga dimensi. Sesuai dengan gambar, lengan berhasil bergerak secara melingkar menuju titik target yang sudah ditentukan. Dengan memanfaatkan rumus Rodrigues, tidak ada dua atau lebih poros sumbu rotasi yang menggerakan lengan ke arah yang sama.

Lebih lanjut, hasil simulasi juga memvalidasi efektivitas algoritma *Cyclic Coordinate Descent* (CCD) dalam menyelesaikan permasalahan jarak secara iteratif. Dapat diamati bahwa konfigurasi sendi membentuk sudut-sudut yang optimal untuk meminimalkan *error* jarak antara *end effector* dan target bola merah. Hal yang krusial dari visualisasi ini adalah terjaganya integritas struktural system, seperti panjang setiap segmen lengan (garis berwarna *cyan*) tetap konstan dan tidak mengalami deformasi atau perubahan panjang selama pergerakan. Ini membuktikan bahwa penerapan konsep *distance constraint* berbasis vektor telah berhasil diimplementasikan, memastikan lengan robot berperilaku sebagai benda kaku (*rigid body*) yang realistis saat melakukan manuver rotasi 3D.

V. KESIMPULAN

Kesimpulan dari makalah ini adalah sistem lengan robot 3D dimensi dapat direpresentasikan oleh kumpulan vektor kinematik. Setiap titik pada lengan merepresentasikan sebuah titik pivot yang menghubungan setiap segmen lengan. Penggunaan algoritma CCD pada makalah ini juga menunjukan bahwa algoritma ini sangat membantu dalam proses menentukan vektor rotasi sendi dalam mencapai titik target. Dengan kombinasi CCD dan rumus Rotasi Rodrigues, proses rotasi lengan robot dapat berjalan dengan stabil dan tahan terhadap kasus *Gimbal Lock*.

VI. LAMPIRAN

Tautan Video:
https://youtu.be/i5XnVyPvByQ?si=QDqbVuRhcxJSxor1

Tautan *Source Code*:
https://github.com/Kurt-Mikhael/Implementasi-Lengan-Robot-3D-dengan-CCD-Cyclic-Coordinate-Descent-

VII. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Kuasa karena atas berkat dan karuniaNya Penulis dapat menyelesaikan tugas makalah ini. Penulis juga berterimakasih kepada Bapak Ir. Rila Mandala, M.Eng., Ph.D. sebagai Dosen Pengampu Mata Kuliah IF2123 Aljabar Linear dan Geometri atas ilmu dan bimbingan yang diberikan. Penulis juga berterimakasih kepada keluarga Penulis yang selalu memberikan dukungan tiada henti selama masa studi Penulis sampai saat ini.

DAFTAR PUSTAKA

[1] R. Munir, " Vektor di Ruang Euclidean (bagian 1)," *Bahan Kuliah IF2123 Aljabar Linier dan Geometri*. Bandung, Indonesia: Program Studi Teknik Informatika STEI-ITB, 2025. [Daring]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-11-Vektor-di-Ruang-Euclidean-Bag1-2025.pdf. [Diakses: 21 Desember 2025].

[2] R. Munir, " Vektor di Ruang Euclidean (bagian 2)," *Bahan Kuliah IF2123 Aljabar Linier dan Geometri*. Bandung, Indonesia: Program Studi Teknik Informatika STEI-ITB, 2025. [Daring]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-12-Vektor-di-Ruang-Euclidean-Bag2-2025.pdf. [Diakses: 21 Desember 2025].

[3] R. Munir, " Vektor di Ruang Euclidean (bagian 3)," *Bahan Kuliah IF2123 Aljabar Linier dan Geometri*. Bandung, Indonesia: Program Studi Teknik Informatika STEI-ITB, 2025. [Daring]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-13-Vektor-di-Ruang-Euclidean-Bag3-2025.pdf. [Diakses: 21 Desember 2025].

[4] B. Kenwright, "Inverse kinematics–cyclic coordinate descent (CCD)," *Journal of Graphics Tools*, vol. 16, no. 4, hal. 177–217, 2012. [Daring]. Tersedia: https://alogicalmind.com/res/inverse_kinematics_ccd/paper.pdf. [Diakses: 21 Desember 2025].

[5] [1] A. G. Valdenebro, "Visualizing rotations and composition of rotations with Rodrigues' vector," *European Journal of Physics*, vol. 37, no. 6, art. no. 065001, Nov. 2016. [Daring]. Tersedia: https://iopscience.iop.org/article/10.1088/0143-0807/37/6/065001. [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

Kurt Mikhael Purba (13524065)