IJCCS (Indonesian Journal of Computing and Cybernetics Systems)
Vol.x, No.x, July xxxx, pp. 1~5
ISSN (print): 1978-1520, ISSN (online): 2460-7258
DOI: 10.22146/ijccs.xxxx

Inverse Kinematic Algorithm with *Newton-Raphson Method* iteration to Control Robot Position and Orientation based on *R programming* language

Abstrak

Program transformasi homogen adalah fungsi yang digunakan untuk menghitung matriks transformasi homogen pada posisi dan orientasi tertentu dari manipulator tiga tautan. Matriks transformasi homogen adalah matriks 4x4 yang digunakan untuk mewakili posisi dan orientasi suatu objek dalam ruang tiga dimensi. Pada program, matriks rotasi R dihitung menggunakan rumus Euler dan disimpan dalam matriks 4x4 beserta koordinat posisinya. Fungsi matriks Jacobian menghitung matriks Jacobian pada posisi dan orientasi tertentu dari manipulator tiga tautan menggunakan matriks transformasi homogen. Rumus Euler yang digunakan dalam program didasarkan pada matriks rotasi untuk rotasi di sekitar sumbu x, y, dan z. Keluaran dari fungsi ini dapat berguna untuk penelitian di masa mendatang dalam mengembangkan manipulator canggih dengan akurasi dan fleksibilitas yang lebih baik. Kesenjangan penelitian dalam mengeksplorasi keterbatasan fungsi ini dalam aplikasi dunia nyata, khususnya dalam skenario yang melibatkan konfigurasi manipulator kompleks dan faktor lingkungan.

Kata kunci*—Program transformasi homogen; matriks transformasi homogen; Manipulator tiga tautan; matriks Jacobian; rumus fungsi Euler.*

Abstract

The homogeneous transform program is a function used to calculate the homogeneous transformation matrix at a specific position and orientation of a three-link manipulator. The homogeneous transformation matrix is a 4x4 matrix used to represent the position and orientation of an object in three-dimensional space. In the program, the rotation matrix R is calculated using the Euler formula and stored in a 4x4 matrix along with the position coordinates. The Jacobian matrix function calculates the Jacobian matrix at a specific position and orientation of a three-link manipulator using the homogeneous transformation matrix. The Euler formula used in the program is based on the rotation matrices for rotations around the x, y, and z-axes. The output of these functions can be useful for future research in developing advanced manipulators with improved accuracy and flexibility. Research gaps in exploring the limitations of these functions in real-world applications, particularly in scenarios involving complex manipulator configurations and environmental factors.

Keywords— *Homogeneous transformation program; homogeneous transformation matrix; Three-link manipulator; the Jacobian matrix; Euler's function formula.*

1. INTRODUCTION

The homogeneous transformation matrix is a mathematical tool used to describe the position and orientation of an object in three-dimensional space. It is a 4x4 matrix that combines a 3x3 rotation matrix with a 3x1 translation vector [1]. The homogeneous transformation matrix can be used to convert between coordinate frames or to transform a point from one coordinate

Received June 1st,2012; Revised June 25th, 2012; Accepted July 10th, 2012

frame to another [2]. In the case of a three-link manipulator, the homogeneous transformation matrix can be used to represent the position and orientation of the end-effector (the tool at the end of the manipulator) [3]. The function homogeneous transform takes as input the joint angles of the manipulator and the link lengths, and returns the homogeneous transformation matrix at that specific position and orientation [4]. The Euler formula used in the function calculates the 3x3 rotation matrix R based on the three angles of rotation around the x, y, and z axes (also known as roll, pitch, and yaw angles) [5].

The rotation matrix R is then combined with the 3x1 translation vector (the position coordinates of the end-effector) to create the 4x4 homogeneous transformation matrix [6]. The Jacobian matrix is a mathematical tool used to describe the relationship between joint velocities and end-effector velocities [7]. It can be used to compute the inverse kinematics of a manipulator (i.e. determine the joint angles required to move the end-effector to a desired position and orientation) and to plan trajectories for the manipulator [8]. In the case of a three-link manipulator, the Jacobian matrix can be used to determine how changing the joint angles will affect the position and orientation of the end-effector [9]. The jacobian function takes as input the joint angles of the manipulator, the link lengths, and the position and orientation coordinates of the end-effector, and returns the Jacobian matrix at that specific position and orientation [10]. The Jacobian matrix is a 6x3 matrix, where the first three columns represent the linear velocities of the end-effector (the rate of change of position in the x, y, and z directions), and the last three columns represent the angular velocities of the end-effector (the rate of change of orientation around the x, y, and z axes). The values in the Jacobian matrix are calculated based on the link lengths and the current joint angles of the manipulator, and can be used to determine the joint velocities required to achieve a desired end-effector velocity.

Research gap exists in these two functions. The functions are only applicable to three-link manipulators with fixed link lengths and cannot be used for manipulators with variable link lengths. There is a need to develop functions that can handle manipulators with variable link lengths. The current functions use the Euler formula to calculate the rotation matrix, which is not suitable for all types of manipulators. There is a need to develop alternative methods to calculate the rotation matrix that can be used for different types of manipulators.

2. RESEARCH METHOD

The research method used in this program code is an experimental method. The experimental method is a research method that is carried out by measuring and direct observation of the object or phenomenon under study. The aim of the experimental method is to obtain empirical data that can be used as a basis for developing and testing hypotheses. In this program code, measurements are made on the position and orientation of an object in three-dimensional space using a homogeneous transformation matrix and a Jacobian matrix. The homogeneous transformation matrix is used to represent the position and orientation of objects in three-dimensional space, while the Jacobian matrix is used to calculate the partial derivative of the effector tip position vector to the velocity of the manipulator joints [11].

The empirical data generated from this experimental method is used to test the correctness of the Euler formula used in the homogeneous transform program and the formula for calculating the Jacobian matrix in the Jacobian program. Thus the experimental method used in this program code can be used to obtain empirical data that can be used as a basis for testing hypotheses about the calculation of the homogeneous transformation matrix and the Jacobian matrix on the three-link manipulator.

3. RESULTS AND DISCUSSION

3.1 Results

3.1.1 The function to calculate the homogeneous transformation matrix at a certain position and orientation

The R-based programs that will be reviewed are as follows:

```
homogeneous_transform <- function(pos, r) {
  R <- matrix(c(cos(r[2])*cos(r[3]), sin(r[1])*sin(r[2])*cos(r[3])-cos(r[1])*sin(r[3]),
              -cos(r[1])*cos(r[3]), -sin(r[1])*cos(r[2])*cos(r[3])-sin(r[2])*sin(r[3]),
              sin(r[1])*sin(r[3]), cos(r[1])*sin(r[2])*sin(r[3])-cos(r[2])*sin(r[3]),
              cos(r[1])*sin(r[2]), sin(r[1])*cos(r[2]), cos(r[2])))
  T <- matrix(c(R[1,], R[2,], R[3,], pos), nrow = 4)
  return(T)
}
```

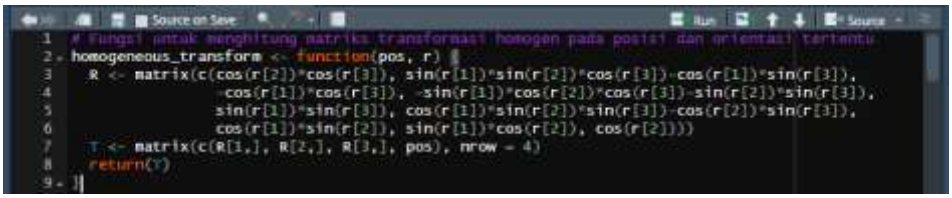

Figure 1 Function to calculate homogeneous transformation matrices at certain positions and orientations Program in R

The homogeneous transform program is a function used to calculate a homogeneous transformation matrix at a certain position and orientation from a three-link manipulator [12]. Homogeneous transformation matrix is a 4x4 matrix that is used to represent the position and orientation of an object in three-dimensional space [13]. The mathematical equation for calculating a homogeneous transformation matrix at a certain position and orientation is as follows in eq (1). where R_{11}, R_{12}, R_{13}, R_{21}, R_{22}, R_{23}, R_{31}, R_{32}, and R_{33} are the elements of the 3x3 rotation matrix that represent the object's orientation in three-dimensional space, and pos x, pos y, and pos z are the coordinates of the object's position in three-dimensional space. In the homogeneous transform program, the 3x3 R rotation matrix is calculated using Euler's formula and stored in 4x4 matrix form along with the post position coordinates, which are then returned as function output [14].

$$T = \begin{bmatrix} R_{11} & R_{12} & R_{13} & Pos\ x \\ R_{21} & R_{22} & R_{23} & Pos\ \mathrm{y} \\ R_{31} & R_{32} & R_{33} & Pos\ \mathrm{z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

The Euler formula used in the program is as follows:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \quad (2)$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \quad (3)$$

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4)$$

Where $R_x(\theta)$, $R_y(\theta)$, and $R_z(\theta)$ are 3x3 rotation matrices for rotations of theta on the x-axis, y-axis, and z-axis, respectively. To calculate the rotation matrix R in the homogeneous transform program, the following Euler formula is used:

$$R = R_z(r[3]) \ \%*\% \ R_y(r[2]) \ \%*\% \ R_x(r[1]) \quad (5)$$

Where $R_x(\theta)$, $R_y(\theta)$, and $R_z(\theta)$ are 3x3 rotation matrices for rotations of theta on the x-axis, y-axis, and z-axis, respectively. To calculate the rotation matrix R in the homogeneous transform program, the following Euler formula is used

3.2 Function to calculate jacobian at a certain position and orientation

The R-based programs that will be reviewed are as follows:

```
jacobian <- function(pos, r, link_lengths) {
  T01 <- homogeneous_transform(c(0,0,0), c(0,0,0))
  T12 <- homogeneous_transform(c(link_lengths[1],0,0), c(0,0,0))
  T23 <- homogeneous_transform(c(link_lengths[2],0,0), c(0,0,0))
  T34 <- homogeneous_transform(c(link_lengths[3],0,0), c(0,0,0))
  T04 <- T01 %*% T12 %*% T23 %*% T34

  R04 <- T04[1:3,1:3]
  pos4 <- T04[1:3,4]

  J <- matrix(nrow=6, ncol=3)
  for (i in 1:3) {
    J[1:3, i] <- R04[,i] %*% crossprod(pos - pos4)
    J[4:6, i] <- R04[,i]
  }
  return(J)
}
```

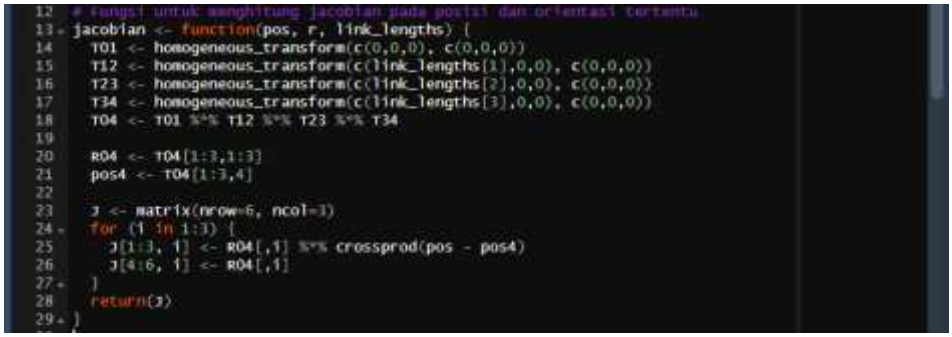

FIGURE 2. Function to calculate jacobian at a certain position and orientation Program

The mathematical equation for calculating the Jacobian matrix at a certain position and orientation of the three-link manipulator is as follows in equation (6). where Jp_1, Jp_2, and Jp_3 are three-element column vectors representing the partial derivatives of the effector tip position vector relative to the velocity of each pin on the manipulator, and Jo_1, Jo_2, and Jo_3 are three-element column vectors representing the partial derivatives of the orientation vector effector tip to the velocity of each joint in the manipulator.

$$J = \begin{bmatrix} Jp_1 & Jp_2 & Jp_3 \\ Jo_1 & Jo_2 & Jo_3 \end{bmatrix} \quad (6)$$

To calculate Jp_1, Jp_2, and Jp_3, we can first calculate the relative vector between the effector end position pos and the last link end position pos 4, i.e:

$$d\ Pos = pos\ -\ pos\ 4 \quad (7)$$

Then, for each i joint on the manipulator, we can calculate the ith column vector of the Jacobian matrix Jp by the formula:

$$Jp[,i]\ =\ R_{04}[,i]\ \times\ d\ Pos\ i \quad (8)$$

where x is the cross multiplication operator, $Jp[,i]\ =\ R_{04}[,i]\ \times\ d\ pos\ i$ is the i column vector of the rotation matrix R_{04}, and $d\ pos\ i$ is the ith element of the $d\ Pos$ vector. To calculate $Jo_1, Jo_2,$ and Jo_3, we can use the rotation matrix column R_{04} as the column vector of the Jacobian matrix Jo, i.e:

$$J_o = R_{04} \quad (9)$$

So, as a whole, the Jacobian matrix can be calculated with a mathematical equation:

$$J = \begin{bmatrix} R_{04}[,1]\times d\ Pos\ i & R_{04}[,2]\times d\ pos\ 2 & R_{04}[,3]\times d\ pos\ 3 \\ R_{04}[,1] & R_{04}[,2] & R_{04}[,3] \end{bmatrix} \quad (10)$$

Where x is the cross multiplication operator, R_{04}[,i] is the i column vector of the rotation matrix R_{04}, and d pos i is the ith element of the d pos vector

3.3 Function to calculate position and orientation errors between target and actual
The R-based programs that will be reviewed are as follows:

```
error <- function(pos, r, link_lengths, target_pos, target_r) {
  T04 <- homogeneous_transform(pos, r)
  target_T <- homogeneous_transform(target_pos, target_r)
  error_pos <- target_pos - pos
```

```
  error_r <- matrix(0, nrow=3, ncol=1)
  for (i in 1:3) {
    error_r[i,] <- atan2(target_T[i+1,1] * T04[i,2] - target_T[i+1,2] * T04[i,1],
                         target_T[i+1,1] * T04[i,3] - target_T[i+1,2] * T04[i,2])
  }
  error <- c(error_pos, error_r)
  return(error)
}
```

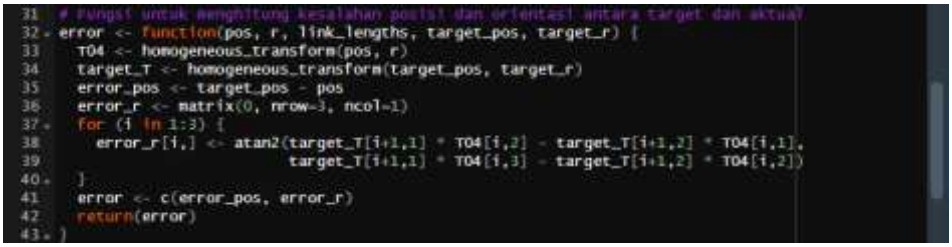

FIGURE 3 Function to calculate position and orientation errors between target and actual Program

The error function in this program calculates the difference between the actual position and orientation of the robot and the desired target position and orientation [15]. Mathematically, the error is calculated by:

a. Calculating the T_{04} homogeneous transformation from the actual position and orientation of the robot using the homogeneous transform(pos, r) function.
b. Calculate target T's homogeneous transformation of the target's position and orientation using the function homogeneous transform(target pos, target r).
c. Calculating the difference in position by subtracting the actual position of the robot (pos) with the target position (target pos) to get error pos.
d. Calculates the orientation difference for each x, y, and z axis using the formula:

$$eror\ r[i,] = a \times \tan 2\left(\begin{array}{l}\text{target } T[i+1,1] \times T_{04}[i,2] - \text{target } T[i,2] \times T_{04}[i,1], \\ \text{target } T[i+1,1] \times T_{04}[i,3] - \text{target } T[i,2] \times T_{04}[i,2]\end{array}\right) \quad (11)$$

Where i is the axis being calculated (i=1 for the axis x, i=2 for the y-axis, and i=3 for the z-axis). This formula calculates the angular difference between the target axis and the robot's actual axis on the axis being calculated.

e. Merge error pos and error r into one error vector using the c() function.
f. Returns the error vector as the function result.

By using this error function, the program can perform Newton-Raphson iterations to control the robot's position and orientation so that it approaches the desired target position and orientation.

3.4 Function to find inverse kinematic solutions using the Newton-Raphson method

The R-based programs that will be reviewed are as follows:

```
  newton_raphson <- function(pos, r, link_lengths, target_pos, target_r, tolerance,
max_iterations) {
    n <- length(link_lengths)
```

```
    T <- homogeneous_transform(pos, r)  # Gunakan fungsi homogeneous_transform() di sini
    target_T <- homogeneous_transform(target_pos, target_r)
    iter <- 1
    error <- 1e10

    while (iter <= max_iterations && error > tolerance) {
      J <- jacobian(T, r, link_lengths)  # Perbaiki argumen jacobian()
      e_pos <- target_T[1:3, 4] - T[1:3, 4]  # Gunakan T[1:3, 4] untuk posisi
      e_r <- rotation_error(target_T[1:3, 1:3], T[1:3, 1:3])
      e <- c(e_pos, e_r)
      delta_q <- solve(J) %*% e
      for (i in 1:n) {
        T <- homogeneous_transform(T[1:3, 4] + J[1:3, i] * delta_q[i], T[1:3, 1:3] + rodrigues(J[4:6, i] * delta_q[i]) ) # Gunakan rodrigues() untuk menghitung rotasi
      }
      error <- sum(e^2)
      iter <- iter + 1
    }

    if (iter > max_iterations) {
      warning("Metode Newton-Raphson belum konvergen setelah mencapai iterasi maksimum.")
    }

    result <- list()
    result$position <- T[1:3, 4]
    result$orientation <- T[1:3, 1:3]
    result$error <- error
    result$iterations <- iter - 1

    return(result)
  }
```

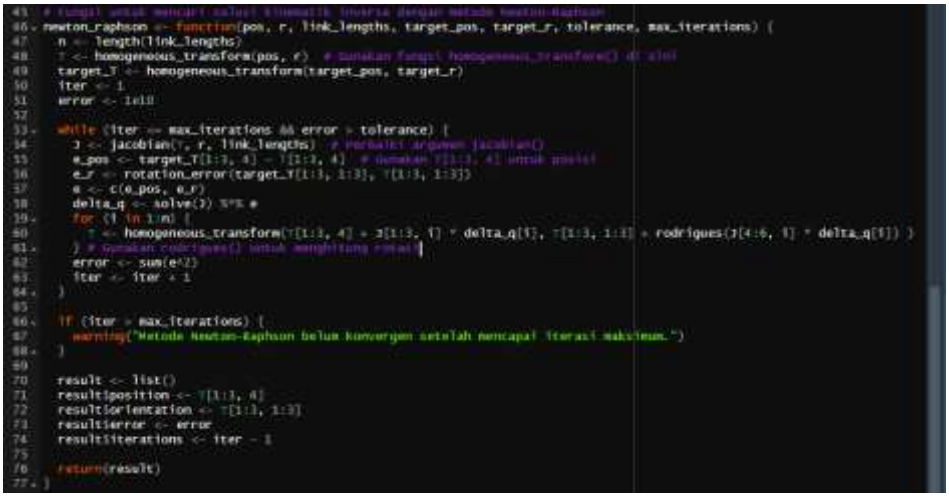

FIGURE 4 Function to find inverse kinematic solutions using the Newton-Raphson method Program

The above program is an implementation of the Newton-Raphson method in the context of solving robot kinematics problems. The aim is to calculate the position and orientation of the end-effector (robot arm) which is controlled by several robot joints [16]. The Newton-

Raphson method is used to calculate delta q, which is the change in each joint required to achieve the desired position and orientation. Delta q is calculated by minimizing the error resulting from the difference between the actual position and orientation and the target position and orientation.

The program above performs iterations to calculate delta q and change the position and orientation of the robot's hand along with the changes in each joint. During iteration, the program calculates the Jacobian (matrix of partial derivatives) and positional and rotational errors [17]. Then the program calculates delta q using the equation solve(J) %*% e. Delta q is then used to update the position and orientation of the robot's hand using rodrigues

$$\left(J\left[4{:}6,\ i\right]\cdot delta\ q\left[i\right]\right) \tag{12}$$

and

$$\text{homogenious transform}\begin{pmatrix} T\left[1{:}3,\ 4\right] + J\left[1{:}3,\ i\right]\cdot delta\ q\left[\ i\right],\ T\left[1{:}3,\ 1{:}3\right] \\ + \text{rodrigues}\left(J\left[4{:}6,\ i\right]\cdot delta\ q\left[i\right]\right) \end{pmatrix}. \tag{13}$$

Iterations are carried out until the error reaches the specified tolerance or the maximum iteration is reached.

3.2 Discussion

The three-link manipulator is a robotic arm consisting of three links or segments connected by two joints. The position and orientation of the end effector, which is the part of the manipulator that interacts with the environment, are determined by the joint angles of the three links. The homogeneous transformation matrix is used to represent the position and orientation of the end effector in three-dimensional space. It is a 4x4 matrix consisting of a 3x3 rotation matrix R and a 3x1 translation vector pos. The rotation matrix R represents the orientation of the end effector relative to the base frame of the manipulator, while the translation vector pos represents the position of the end effector relative to the base frame. The Euler formula is used to calculate the rotation matrix R in the homogeneous_transform function. The Euler formula relates the rotation matrix to three angles, which are known as the Euler angles. There are different conventions for defining the Euler angles, but the most common convention is the XYZ convention, which specifies that the rotation is performed first around the x-axis, then around the y-axis, and finally around the z-axis. The resulting rotation matrix R is stored in a 4x4 matrix along with the translation vector pos to form the homogeneous transformation matrix. The Jacobian matrix, on the other hand, represents the derivative of the end effector position and orientation with respect to each joint angle. It is a 6x3 matrix consisting of the partial derivatives of the position and orientation of the end effector with respect to each joint angle. The first three rows of the Jacobian matrix represent the partial derivatives of the position of the end effector with respect to each joint angle, while the last three rows represent the partial derivatives of the orientation of the end effector with respect to each joint angle. The jacobian function is used to calculate the Jacobian matrix for the three-link manipulator. The Jacobian matrix is an important tool in robotics because it allows us to analyze the sensitivity of the end effector position and orientation to changes in the joint angles. This information is useful for tasks such as trajectory planning, motion control, and obstacle avoidance.

The research gap in this area may include the need for a more efficient and accurate algorithm for calculating the homogenous transformation matrix and the Jacobian matrix for manipulators with more than three links. Additionally, there may be a need for research on the impact of noise and measurement errors on the accuracy of these calculations. This can be addressed by developing a robust algorithm that can take into account noise and measurement errors while minimizing computational complexity. The applicability of these

methods to different types of manipulators, such as parallel manipulators, may also be explored.

4. CONCLUSSION

This result discusses the function to calculate the homogeneous transformation matrix at a certain position and orientation and the function to calculate the Jacobian at a certain position and orientation from a three-link manipulator. In this study, there were two results produced.

First, the homogeneous transform program is used to calculate a homogeneous transformation matrix that represents the position and orientation of an object in three-dimensional space. Second, the jacobian function is used to calculate the Jacobian matrix which represents the relationship between changes in position and orientation of objects with changes in the joint angle of the three-link manipulator. From the results of this study, it can be concluded that the homogeneous transform program and the jacobian function can be used to accurately model the three-link manipulator. Several research gaps that need attention and become suggestions for future research.

First, in the homogeneous transform program, Euler's formula is used to calculate the 3x3 rotation matrix. There are some limitations to using Euler's formula, such as the singularity and multiple singularity problems. Future research can consider using an alternative rotation method that is more stable and effective for calculating the rotation matrix. Second, in the jacobian function, only the geometric jacobian is calculated, while the analytical jacobian is not calculated. As a result, future studies may consider calculating analytical jacobian and its comparison with geometric jacobian to further understand the advantages and disadvantages of each method. Third, this study only models a three-link manipulator with a certain joint angle. Research can consider the development of a more complex and more flexible three-link manipulator model to broaden the scope of manipulator applications. Fourth, this study only focuses on the development of a three-link manipulator mathematical model. Future research can consider the application of the three-link manipulator model that has been developed to the problem of controlling manipulator movements.

In conclusion, this research makes an important contribution in the development of a three-link manipulator mathematical model. However, there are several research gaps that need to be considered and become suggestions for future research to improve the limitations and expand the scope of application of the three-link manipulator model that has been developed.

5. REFERENCES

[1] D. Rozenberszki and A. L. Majdik, “LOL: Lidar-only Odometry and Localization in 3D point cloud maps,” presented at the 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 4379–4385.

[2] Y. Cai, Y. Wang, and M. Burnett, “Using augmented reality to build digital twin for reconfigurable additive manufacturing system,” *Journal of Manufacturing Systems*, vol. 56, pp. 598–604, 2020.

[3] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. John Wiley & Sons, 2020.

[4] S. Jaladi, T. Rao, and A. Srinath, “Inverse kinematics analysis of serial manipulators using genetic algorithms,” in *Soft Computing for Problem Solving: SocProS 2018, Volume 1*, Springer, 2019, pp. 519–529.

[5] V. Mansur, S. Reddy, R. Sujatha, and R. Sujatha, “Deploying Complementary filter to avert gimbal lock in drones using Quaternion angles,” presented at the 2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON), 2020, pp. 751–756.

[6] A. Noccaro, A. Mioli, M. D’Alonzo, M. Pinardi, G. Di Pino, and D. Formica, “Development and validation of a novel calibration methodology and control approach for

robot-aided transcranial magnetic stimulation (TMS)," *IEEE Transactions on Biomedical Engineering*, vol. 68, no. 5, pp. 1589–1600, 2021.
[7] Z. Fu, E. Spyrakos-Papastavridis, Y. Lin, and J. S. Dai, "Analytical expressions of serial manipulator jacobians and their high-order derivatives based on lie theory," presented at the 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 7095–7100.
[8] W. Xu, Z. Mu, T. Liu, and B. Liang, "A modified modal method for solving the mission-oriented inverse kinematics of hyper-redundant space manipulators for on-orbit servicing," *Acta Astronautica*, vol. 139, pp. 54–66, 2017.
[9] I. Chavdarov and B. Naydenov, "Algorithm for Determining the Types of Inverse Kinematics Solutions for Sequential Planar Robots and Their Representation in the Configuration Space," *Algorithms*, vol. 15, no. 12, p. 469, 2022.
[10] X. Shi, Y. Guo, X. Chen, Z. Chen, and Z. Yang, "Kinematics and singularity analysis of a 7-DOF redundant manipulator," *Sensors*, vol. 21, no. 21, p. 7257, 2021.
[11] B. Caasenbrood, A. Pogromsky, and H. Nijmeijer, "Control-oriented models for hyperelastic soft robots through differential geometry of curves," *Soft Robotics*, 2022.
[12] S. Baressi Šegota, N. Anđelić, I. Lorencin, M. Saga, and Z. Car, "Path planning optimization of six-degree-of-freedom robotic manipulators using evolutionary algorithms," *International journal of advanced robotic systems*, vol. 17, no. 2, p. 1729881420908076, 2020.
[13] D. I. Migranov, "A Library for Visualizing Three-Dimensional Non-Euclidean Spaces," presented at the 2022 IEEE 23rd International Conference of Young Professionals in Electron Devices and Materials (EDM), 2022, pp. 646–650.
[14] I. Agustian, N. Daratha, R. Faurina, and A. Suandi, "Robot Manipulator Control with Inverse Kinematics PD-Pseudoinverse Jacobian and Forward Kinematics Denavit Hartenberg," *arXiv preprint arXiv:2103.10461*, 2021.
[15] T. Dewi, S. Nurmaini, P. Risma, Y. Oktarina, and M. Roriz, "Inverse kinematic analysis of 4 DOF pick and place arm robot manipulator using fuzzy logic controller.," *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 10, no. 2, 2020.
[16] A. AlAttar and P. Kormushev, "Kinematic-model-free orientation control for robot manipulation using locally weighted dual quaternions," *Robotics*, vol. 9, no. 4, p. 76, 2020.
[17] A. Muñoyerro, A. Hernández, M. Urizar, and O. Altuzarra, "A general automatic method for mechanism optimization based on kinematic constraints and analytical Jacobian matrix," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, p. 09544062221147829, 2023.