Simulation of a 3-DOF Robotic Arm Pick and Place Task Based on Inverse Kinematics

Songyang Mei

College of Electronic Information and Automation, Tianjin University of Science and Technology, 300222, China

Miao Zhang

College of Electronic Information and Automation, Tianjin University of Science and Technology, 300222, China

E-mail: 18186314885@163.com, miaozhang@tust.edu.cn

Abstract

This paper proposes a simulation method for grasping and placing a 3-DOF robotic arm based on inverse kinematics. Through the MATLAB GUI, the user enters target coordinates, computes the joint angles by the inverse kinematics geometric approach and simulates the motion path to achieve the task operation. In this approach, the robotic arm can reach specified positions precisely, which reduces complexity and error. The position error is calculated through a comparison of the target and the actual position. The simulation results show that the robot arm achieves a small average error across all five experimental groups. The maximum error remains within a reasonable range, which confirms the accuracy of the method in grasping and placing tasks.

Keywords: Robotic arm, MATLAB GUI, Pick and place, Inverse kinematics

1. Introduction

As a critical component in the field of automation, robotic arms are widely utilized in industrial manufacturing, healthcare services, and education. With continuous technological advancements, the intelligence and autonomous operation of robotic arms have become prominent research directions. Precise kinematic analysis, particularly the application of inverse kinematics, is essential for accomplishing complex tasks such as object grasping and placement. Inverse kinematics is a mathematical process that determines the necessary joint angles or positions required for a robotic arm to reach a specific end-effector position in workspace [1]. By calculating the joint parameters, the robotic arm can position the end-effector accurately at the desired location. This applies to tasks such as object manipulation, assembly, or interaction with the environment. Inverse kinematics is a commonly used algorithm in robotic arm control, which transforms target positions into joint angle movements to achieve accurate control. Various methods are commonly used to solve inverse kinematics problems, such as the Denavit-Hartenberg (D-H) parameter method [2], screw theory, iterative techniques [3], and soft computing approaches. Fundamentally, the aim of these methods is to find the optimal joint angles to reach the desired target position. In existing research, many scholars have explored the application of inverse kinematics in robotic arms. For example, studies have demonstrated that inverse kinematics methods can effectively control the position and orientation of robotic arms during object-picking tasks [4].

The structure of this paper is organized as follows. The first section discusses the importance and necessity of inverse kinematics in addressing robotic arm pick and place tasks. The second section describes the MATLAB GUI design process, the forward kinematics model of a three-degree-of-freedom (3-DOF) robotic arm, the methods used to solve inverse kinematics, and the workflow for pick and place operations. The third section evaluates the performance of the robotic arm in pick and place tasks by analyzing the errors between the target and actual positions. The fourth section summarizes the conclusions drawn from the simulation experiments and confirms the effectiveness of the proposed methods.

2. Research Methodology

The method employed in this paper primarily utilizes mathematical geometry to solve the inverse kinematics problem, enabling the execution of pick and place tasks with the 3-DOF robotic arm.

2.1. MATLAB GUI design

The designed system utilizes a MATLAB graphical user interface (GUI) constructed with components such as "Ui figure" and " Ui label " to create an input and display interface. Users can input the X, Y, and Z coordinates for

the target pick and place positions. The interface includes buttons to trigger inverse kinematics calculations, as well as output fields or labels to display the calculated joint angles. The designed interface is shown in Fig. 1. By clicking the "Pick and Place" button, the system reads the user-input coordinates and outputs the results in the MATLAB console. The "Reset" button clears all input coordinate information. Upon entering the target position coordinates and pressing the corresponding button, the program calculates the rotational angles of each joint using the inverse kinematics based on the robotic arm's kinematic model. Once the joint angles are determined, the system controls the robotic arm to move to the specified position, completing the pick and place task. Additionally, the actual position coordinates of the end-effector are displayed in the output field.

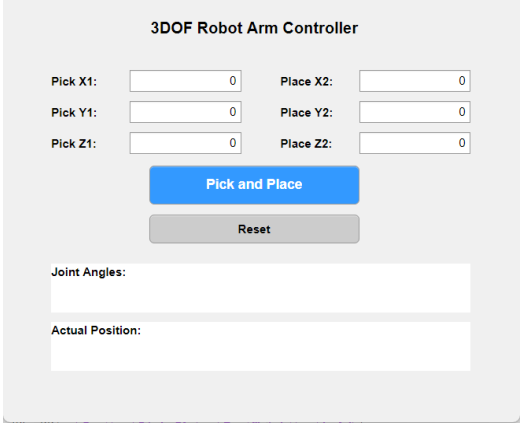

Fig. 1 MATLAB GUI design interface

2.2. Forward kinematic modeling of the 3-DOF robotic arm

The 3-DOF robotic arm is defined based on its link structure, where all links are described using the standard D-H parameter method. The D-H parameters for the link coordinate systems are summarized in Table 1Table 1. In the table, a_i refers to the link length, which represents the displacement along the X-axis of the current link coordinate system. α_i indicates the link twist, describing the rotation angle of the X-axis from the previous link to the current link about the Z-axis. d_i denotes the link offset, representing the displacement along the Z-axis of the previous link coordinate system. θ_i represents the joint angle, which corresponds to the rotation of the current joint.

Table 1. D-H parameters

	a_i	α_i	d_i	θ_i
1	0	$-\frac{\pi}{2}$	10	θ_1
2	15	0	0	θ_2
3	20	0	0	θ_3

The 3-DOF robotic arm model is shown in Fig. 2. Based on the link coordinate systems, the transformation relationship between adjacent links can be established, with equation (1) representing the transformation matrix.

$$ {}^{i-1}_{i}T = \begin{bmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1) $$

From the above equations, the forward kinematics equation can be expressed as equation (2).

$$ {}^{0}_{3}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \quad (2) $$

2.3. Inverse kinematics solution method

Given the target coordinates (x, y, z) of the end-effector, the inverse kinematics procedure is as follows.

First, the first joint angle θ_1 is calculated. This angle represents the rotation of the base joint in the horizontal plane, indicating how the base of the robotic arm should rotate to point toward the target (x, y). Specifically, θ_1 can be determined using the inverse tangent function of the x and y coordinates of the target position.

$$ \theta_1 = atan2(y, x) \quad (3) $$

Next, the effective horizontal distance r and effective height z_{eff} are computed. The effective horizontal distance r represents the distance of the target position in the plane and can be found by taking the square root of the sum of the squares of x and y. The effective height z_{eff} is the target height minus the length of the base link L_1, which gives the relative height of the target with respect to the robotic base.

$$ r = \sqrt{x^2 + y^2}; Z_{eff} = z - L_1 \quad (4) $$

Then, the second joint angle θ_2, which represents the shoulder angle, is calculated using the law of cosines. The

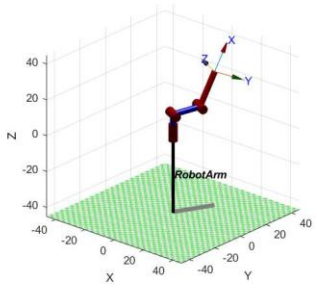

Fig. 2 Robotic arm model

known horizontal distance r, effective height z_{eff}, and the lengths of the two links L_2 and L_3 are used to determine this joint angle.

$$cos(\theta_2) = \frac{r^2 + z_{eff}^2 + L_2^2 - L_3^2}{2L_2\sqrt{r^2 + z_{eff}^2}} \quad (5)$$

Finally, the third joint angle θ_3, representing the elbow angle, is calculated. The previously computed horizontal distance, effective height, and the lengths of the arm links are used to derive this angle through geometric relationships.

$$\theta_3 = cos^{-1}\left(\frac{r^2 + z_{eff}^2 - L_2^2 - L_3^2}{2L_2L_3}\right) \quad (6)$$

Through these steps, all the joint angles of the robotic arm θ_1, θ_2, and θ_3 can be determined, enabling the robotic arm to reach the desired end-effector position.

2.4. Overall workflow of robotic arm pick and place operation

The workflow of the robotic arm system for completing pick and place tasks is illustrated in Fig. 3. The system starts with the user inputting the target pick and place coordinates via the MATLAB GUI. Once the coordinates are entered, the system performs inverse kinematics calculations to determine the joint angles required for the operation. Subsequently, the robotic arm executes the corresponding movements based on the calculated joint angles to complete the pick and place task. After the operation, the system displays the computed joint angles and the actual position of the robotic arm. At this point, the system prompts the user to decide whether additional target coordinates need to be entered. If the user provides new coordinates, the process repeats. If no additional coordinates are entered, the operation concludes.

Table 2. Pick joint angles

	θ_1	θ_2	θ_3
1	59.98	39.47	-106.70
2	41.96	51.21	-108.94
3	43.71	90.22	-135.01
4	-138.72	91.30	42.88
5	76.75	-0.51	-83.21

Table 3. Place joint angles

	θ_1	θ_2	θ_3
1	35.88	53.12	-88.74
2	63.72	-7.13	-58.60
3	-1.73	70.00	-66.57
4	60.20	50.21	-98.73
5	-82.27	178.25	-100.14

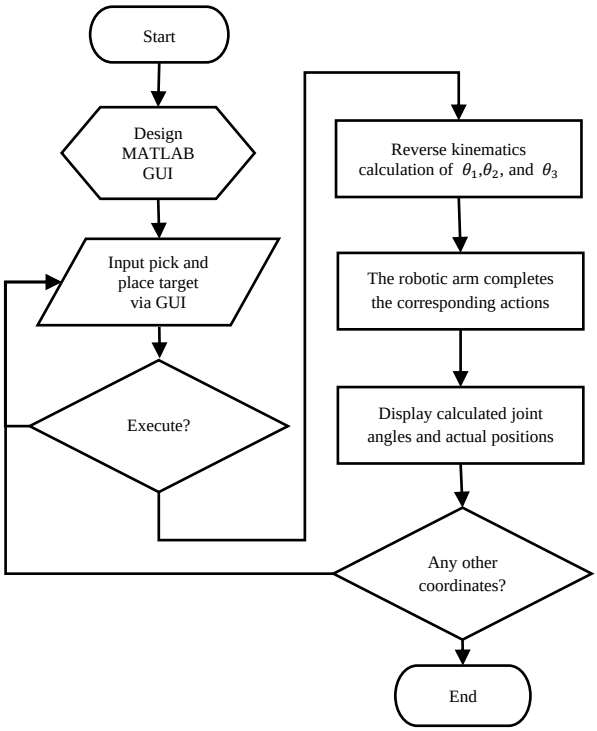

Fig. 3 Overall workflow diagram

3. Simulation Results

The simulation process involves the calculation of joint angles through inverse kinematics, interpolation of these angles, and the visualization of the robotic arm's movements during object grasping and placement. The 3-DOF robotic arm has the following link lengths, L_1 = 10 cm, L_2 = 15 cm, and L_3 = 20 cm. The test coordinates in the pick condition are (10,15,20), (15,10,18), (10,10,10), (12,8, -20), (5,15,30), the test coordinates in the place condition are (20,15,10), (10,20,30), (25,0, -5), (10,22,12), (0,10, -10). Table 2 and Table 3 present the joint angles for the corresponding target positions, which are calculated using inverse kinematics.

The comparison between the actual positions reached by the robotic arm and the target coordinates is illustrated in Fig. 4 and Fig. 5. The test is done by dividing the workspace into quadrant 1 for the negative end-effector X position and quadrant 2 for the positive end-effector X position. The simulation results indicate that the robotic arm end-effector successfully reached the specified target positions. The error is calculated using equation (7).

$$error = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2} \quad (7)$$

Where $(\Delta x, \Delta y, \Delta z)$ are the differences between the target coordinates of (x, y, z) and the actual coordinates.

The robotic arm managed to reach the coordinates based on the given target, despite the errors. The errors of the second and third sets are larger compared to the other three sets. This may be due to the spatial distribution of the target positions, the geometric structure of the robotic arm, and the accuracy limitations in the inverse kinematics calculation process. At some target positions, geometric constraints affect robotic arm movement. This increases joint angle errors and leads to higher positional errors for the end-effector. In the five test groups, the average positional error is 1.83 cm, and the maximum error is 4.41 cm.

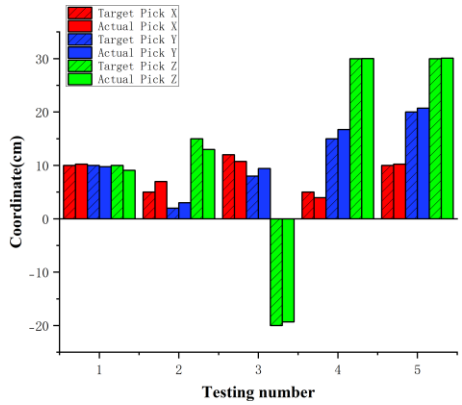

Fig. 4 Pick coordinate result

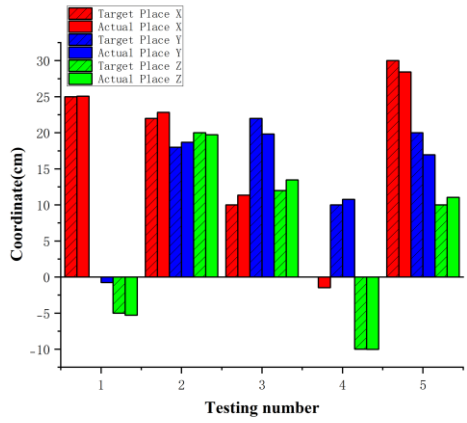

Fig. 5 Place coordinate result

4. Conclusion

This paper employs inverse kinematics to achieve grasping and placing tasks using a 3-DOF robotic arm. The inverse kinematics algorithm calculates the joint angles needed to position and orient the end-effector accurately in the workspace, ensuring precise task execution. Simulation results demonstrate the effectiveness of the proposed method. Across five experimental trials, the robotic arm achieved an average positioning error that remained minimal, with the maximum error maintained within an acceptable range. These findings validate the precision and reliability of the inverse kinematics approach in grasping and placing tasks, highlighting their potential in practical robot applications.

References

1. Al Tahtawi, Adnan Rafi, Muhammad Agni, et al., Small-scale robot arm design with pick and place mission based on inverse kinematics. *Journal of Robotics and Control (JRC)* 2.6 (2021): pp.469-475.
2. Becerra, Yeyson, Mario Arbulu, et al., A comparison among the Denavit-Hartenberg, the screw theory, and the iterative methods to solve inverse kinematics for assistant robot arm. *Advances in Swarm Intelligence: 10th International Conference*, ICSI 2019, Chiang Mai, Thailand, July 26–30, 2019, Proceedings, Part I 10. Springer International Publishing, 2019.
3. Patil, Apurva, Maithilee Kulkarni, et al., Analysis of the inverse kinematics for 5 DOF robot arm using DH parameters. *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. IEEE, 2017.
4. Kaur, Manpreet, Venkata Karteek Yanumula, et al., Trajectory planning and inverse kinematics solution of Kuka robot using COA along with pick and place application. *Intelligent Service Robotics* 17.2 (2024): pp.289-302.

Authors Introduction

Mr. Songyang Mei

He received his bachelor's degree in 2023 from the School of Electrical and Electronic Engineering at Wuhan Polytechnic University. Currently, he is a master's student at Tianjin University of Science and Technology, with a research focus on robotic kinematics.

Ms. Miao Zhang

She is a postgraduate tutor of Tianjin University of Science and Technology. In 2019, she received a doctorate from the University of Windsor, Ontario, Canada. The research direction is intelligent algorithms design filters, the control system design of industrial robots and control theory.