HiLL PUBLISHING

Advances in Computer and Communications, 2023, 4(6), 363-372
https://www.hillpublisher.com/journals/acc/
ISSN Online: 2767-2875

Design and Application Research on Control System of 3D Printer Based on Multi-DOF Robotic Arm

Hailu Xu*, Xinping Rong

Faculty of Electrical and Mechanical, Nanjing Tech University Pujiang Institute, Nanjing, Jiangsu, China.

How to cite this paper: Hailu Xu, Xinping Rong. (2023) Design and Application Research on Control System of 3D Printer Based on Multi-DOF Robotic Arm. *Advances in Computer and Communication*, 4(6), 363-372.
DOI: 10.26855/acc.2023.12.004

Received: November 25, 2023
Accepted: December 22, 2023
Published: January 18, 2024

***Corresponding author:** Hailu Xu, Faculty of Electrical and Mechanical, Nanjing Tech University Pujiang Institute, Nanjing, Jiangsu, China.

Abstract

This study combined 3D printing technology to design a control system for a small robotic arm and developed a compact and multi-degree-of-freedom robotic arm 3D printer. Firstly, the forward kinematics and inverse kinematics of the manipulator are solved in detail, and then the path is interpolated to realize the high-precision trajectory planning of the manipulator. This robotic 3D printer can perform complex 3D printing tasks with precision. The research focuses on optimizing the motion trajectory of the manipulator to ensure the stability of motion and the precision of printing. The results show that the system significantly improves the dynamic response and print quality of the printer, makes the printing attitude more flexible, and expands the diversity of printed products. This research aims to improve the traditional 3D printer to achieve a multi-purpose machine, improve work efficiency, and reduce costs. Under the background of "New Engineering Course", combining theory with practice can improve engineering practice ability and scientific research innovative thinking.

Keywords

3D printing technology, robotic arm, control system

1. Introduction

Traditional 3D printers have evolved from basic tools to advanced manufacturing systems, leading to three main printer architectures. The first, a gantry structure 3D printer, uses a conventional mechanical motion framework, known as the i3 structure [1]. This printer controls the print head's position via three perpendicular axes (X, Y, Z), stacking material to form three-dimensional objects. Its design, simple and cost-effective, is popular in personal and low-cost markets. However, due to its simplicity, it faces limitations in print speed and stability, particularly for larger objects.

The Delta 3D printer, inspired by Prof. Reymond Clavel's Delta-style parallel robotic arm from the École Polytechnique Fédérale de Lausanne, is designed for precision. Originally intended for fast manipulation of light objects, it uses a series of parallelograms for precise print head positioning [2]. Unique in structure, it controls the print head via three top-mounted motors connected by rods, enabling movement along set trajectories. While its parallel mechanism offers stability and speed, making it ideal for high-speed, high-precision printing, its large vertical design limits the horizontal print area and its complexity could pose maintenance challenges [3].

The XYZ box structure 3D printer marks a significant advancement in 3D printing technology. In this design, the X and Y axes move in a horizontal plane, while the print base on the Z-axis moves vertically. This efficient use of space allows for a compact design with a larger print volume. Products like the Einstart series are popular among schools, designers, and small businesses for their structural stability and print accuracy [4].

(a) Gantry structure printer (b) Delta printer (c) XYZ box printer

Figure 1. 3D printer.

Gantry and XYZ Cartesian structure 3D printers utilize a Cartesian coordinate system for print head movement, with X, Y, and Z motion axes perpendicular to each other. This intuitive design enables straightforward motion control. Motor rotations are transformed into linear movements via a screw or belt, moving the print head or bed along the specified axis. Axis travel is controlled by motor steps, with each step equating to one pulse signal, allowing for precise position control [5].

In practical use, Cartesian coordinate system printers commonly use micro stepping for finer stepper motor control, enabling smaller steps and smoother motion for higher printing resolution [6]. XYZ printers, with precise control over three independent axes, generally offer higher accuracy and better surface finishes, especially for complex geometries or fine details.

Delta-style 3D printers use a polar coordinate system with a parallel mechanical structure, where three arms, typically motors and rods, control the print head's position. This design allows for fast, smooth movement in three dimensions, ideal for high-speed printing [7]. However, Delta printers involve complex trigonometric calculations for coordinate transformation, requiring sophisticated control algorithms. They necessitate forward and inverse kinematic calculations at each print task's start, demanding high-performance microprocessors or specialized hardware for fluid printing execution.

When printing circular arcs, Delta printers' control systems often break the arc into numerous straight segments to approximate the path, increasing computational demands and potentially impacting print smoothness and accuracy. The arc approximation's quality hinges on the segmentation's granularity; finer segments yield smoother arcs but require more computational power and faster control system response [8].

2. Three-Axis Serial Robotic Arm 3D Printer

This study presents a design scheme for a three-axis serial robotic arm 3D printer, as illustrated in Figure 2. Upon setting the target coordinates, the system first calculates the position deviation (ΔX, ΔY, ΔZ) between the target position (X, Y, Z) and the current position (X_0, Y_0, Z_0) of the robotic arm's end effector. To precisely control the movement trajectory of the robotic arm, this positional difference is decomposed into multiple discrete micropath segments. For each segment, the system continuously updates the current and target position coordinates and then employs the forward and inverse kinematics algorithms of the robotic arm to calculate the corresponding joint angle values.

To guarantee smooth motion and prevent vibrations in the robotic arm, the system uses advanced kinematic planning [9]. It adjusts acceleration and velocity curves for smooth joint movements and precisely converts joint angles into motor pulses for accurate motion control.

During printing, the control system constantly checks the actuator's real-time position against the target position. If the target isn't reached, it continues outputting control signals to guide the robotic arm along the set trajectory. This closed-loop control secures path accuracy until the end effector precisely reaches the final target position [10].

Through this method, not only is the positioning accuracy during the printing process improved, but also the accumulation of errors caused by dynamic responses is effectively reduced. In summary, the proposed three-axis serial robotic arm 3D printer structure can not only precisely control the movement of the print head along a predetermined three-dimensional path, but also significantly enhance the overall print quality and the smoothness of the robotic arm's movement through optimized control strategies for acceleration and velocity [11]. The implementation of this

technology provides an effective solution for the 3D printing of complex shapes and fine features.

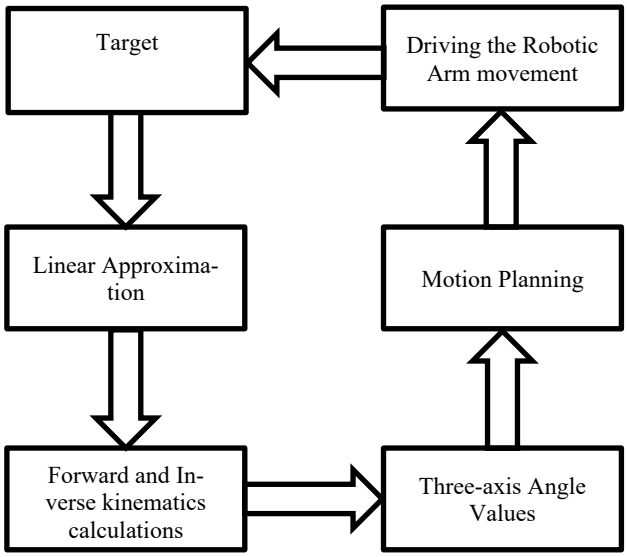
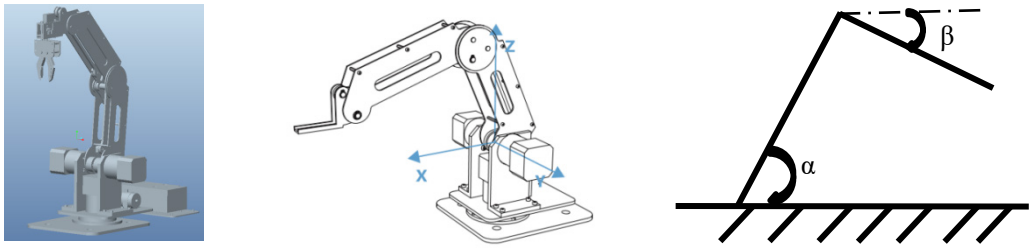


Figure 2. Technical line of research for tandem robotic arm 3D printers.

3. Forward and inverse solution calculation for a robotic arm 3D printer

As shown in Figure 3, to achieve the movement of the robotic arm to the set target coordinates, these spatial coordinates must be converted into corresponding angle values for each joint axis of the robotic arm. This process involves complex kinematic calculations, namely forward and inverse kinematic analysis. Initially, the target position is defined as a point in the coordinate system, and the forward and inverse kinematics of the robotic arm translate these points into joint angles. Forward kinematics involves calculating the spatial position of the robotic arm's end effector based on known joint angles. Then, inverse kinematics is the reverse operation of this process, which involves deducing the required angular configuration of the robotic arm's joints based on the desired target position.



(a) 3D Robot arm (b) Coordinates (c) Angle solution diagram

Figure 3. Forward and inverse solution calculation for robotic arms.

To solve inverse kinematics, it is first necessary to establish a geometric model of the robotic arm's joints and link lengths. Then, by applying trigonometry and matrix operations, a set of solutions for joint angles can be obtained, which will guide the movements of the robotic arm [12]. Based on this, path planning is also required to ensure that the acceleration and velocity of the robotic arm during movement meet specific dynamic constraints, thus avoiding unnecessary vibrations in joint movements [13]. The following are the specific steps needed to solve for α, β and the angle of horizontal motion γ for the positive inverse solution:

1) Given the Cartesian coordinates (T_x, T_y, T_z) of a target point in 3D space, the projection distance of the robotic arm on the horizontal plane can be calculated, which then allows the computation of the horizontal motion angle γ required for the rotation of the robotic arm base.

$$\gamma = arc\tan\left(T_y / T_x\right) \tag{1}$$

2) Calculate the straight-line distance r from the base of the robotic arm to the target point on the horizontal

plane:

$$r = \sqrt{T_x^2 + T_y^2} \tag{2}$$

3) The vertical distance h from the base of the robotic arm to the target point is:

$$h = T_z \tag{3}$$

4) Assuming the robotic arm has two segments, with the first segment length from the shoulder to the elbow being d_1 and the second segment length from the elbow to the wrist being d_2, the angles of the two joints α and β can be calculated within the vertical plane passing through the target point:

$$\alpha = \arctan\left(h/r\right) + \arccos\left[\left(d_1^2 + R^2 - d_2^2\right)/\left(2d_1R\right)\right] \tag{4}$$

$$\beta = \arccos\left[\left(d_1^2 + d_2^2 - R^2\right)/\left(2d_1d_2\right)\right] \tag{5}$$

In the formula, $R = \sqrt{r^2 + h^2}$ represents the straight-line distance from the first joint of the robotic arm to the target point.

In this context, α and β represent the angles that need to be reached by the shoulder and elbow joints of the robotic arm, respectively, while the base's rotation angle is also considered. The two linkages, d_1 and d_2, play a crucial role in the structural design of the robotic arm as they determine the arm's spatial reach and flexibility in movement.

By integrating the application of forward and inverse kinematics solutions with dynamic planning, precise and stable control of the robotic arm can be achieved, meeting the requirements of complex 3D printing tasks. This method significantly improves printing accuracy, reduces printing time, and enhances the stability of the printing process, making it vitally important for advancing the application of robotic arms in the field of 3D printing [14].

In solving the robotic arm's inverse kinematics, iterative algorithms adjust joint angles to align the end effector with the target position. Techniques like the Jacobian inverse method use the relationship between joint and end effector velocities to refine angles [15]. Each iteration involves calculating the actuator's position, adjusting it based on the Jacobian matrix, and repeating until the error is minimal. This approach balances algorithm speed with computational stability and accuracy [16]. A portion of the code is as follows:

```
import numpy as np

    def forward_kinematics(joint_angles, link_lengths):
        # This function should calculate the position of the end effector based on joint angles and link lengths
        # Returns the position of the end effector
        pass

    def compute_jacobian(joint_angles):
        # This function should calculate the Jacobian matrix
        # Returns the Jacobian matrix
        pass
    def inverse_kinematics(target_position, initial_joint_angles, link_lengths, tolerance=1e-3, max_iterations=1000):
        joint_angles = np.array(initial_joint_angles)

        for i in range(max_iterations):
            current_position = forward_kinematics(joint_angles, link_lengths)
            position_error = target_position - current_position

            # If the error is already small, we can stop iterating
            if np.linalg.norm(position_error) < tolerance:
                break
```

```
            # Calculate the Jacobian matrix and its inverse
            J = compute_jacobian(joint_angles)
            J_inv = np.linalg.pinv(J)    # Use pseudo-inverse in case the Jacobian is not a square matrix

            # Update joint angles
            joint_angles += J_inv.dot(position_error)

    return joint_angles if np.linalg.norm(position_error) < tolerance else None

# Example: Assumed link lengths and initial joint angles
link_lengths = [1.0, 1.0]    # Can be modified according to the actual robotic arm
initial_joint_angles = [0, 0]

# Target position
target_position = np.array([1.5, 1.5])

# Run inverse kinematics solution
solution = inverse_kinematics(target_position, initial_joint_angles, link_lengths)
print("Solution joint angles:", solution)

    if np.linalg.norm(position_error) < tolerance:
        break

    # Compute the Jacobian matrix and its inverse
    J = compute_jacobian(joint_angles)
    J_inv = np.linalg.pinv(J)    # Use pseudo-inverse in case the Jacobian is not a square matrix
        # Update joint angles
    joint_angles += J_inv.dot(position_error)

return joint_angles if np.linalg.norm(position_error) < tolerance else None

# Assumed link lengths and initial joint angles
link_lengths = [1.0, 1.0]    # Can be adjusted according to the actual robotic arm
initial_joint_angles = [0, 0]

# Target position
target_position = np.array([1.5, 1.5])

# Run inverse kinematics solution
solution = inverse_kinematics(target_position, initial_joint_angles, link_lengths)
print("Solution joint angles:", solution)
```

In this code, the `inverse_kinematics` function attempts to find a set of joint angles such that the robotic arm's end effector approaches the target position. Starting from an initial set of joint angles, it iteratively uses the inverse (or pseudo-inverse) of the Jacobian matrix to calculate the amount by which to update the joint angles, thereby reducing the error between the current position and the target position. This process is repeated until the position of the end effector is within a very small threshold of the target position or the maximum number of iterations is reached.

4. Motion Interpolation for Robotic Arm 3D Printer

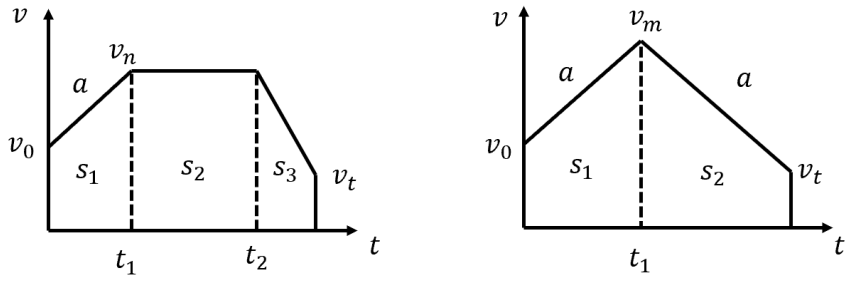
4.1 Motion Planning

To optimize the motion of a robotic arm along a predetermined trajectory, thereby achieving high precision and

stability in the 3D printing process, the implementation of precise motion planning is crucial. This planning involves two key aspects: path planning and trajectory planning. Path planning is responsible for defining a sequence of spatial points that the robotic arm must traverse, while trajectory planning matches these spatial points with time points, ensuring that the robotic arm moves along the path in a predetermined sequence [17].

A critical component in both path and trajectory planning is motion interpolation. It involves generating additional intermediate points between the defined path points to achieve continuous and smooth motion of the robotic arm [18]. The choice of interpolation strategy directly impacts the avoidance of vibrations caused by frequent starts and stops of the robotic arm during the printing process, thus being vital for the surface quality of the final product, mechanical stability of the printer, and its long-term durability.

In this solution, velocity planning is implemented using a trapezoidal acceleration and deceleration strategy [19]. This strategy linearly increases or decreases the speed to avoid sudden changes in velocity, as shown in Figure 4. This model consists of three stages: acceleration, constant speed, and deceleration. During the acceleration phase, the speed linearly increases from zero to a set maximum value; in the constant speed phase, this speed is maintained; and finally, in the deceleration phase, the speed gradually decreases until it stops. This trapezoidal acceleration and deceleration curve not only helps to reduce mechanical shocks and vibrations but also enhances the accuracy and repeatability of the printing process.



(a) Motion Curve Trajectory (b) Interpolation Points Diagram

Figure 4. Trapezoidal Acceleration-Deceleration Graph.

As shown in Figure 4(a), *a* trapezoidal acceleration curve is used for velocity curve planning. The acceleration v_{max} is the rate required for the robotic arm to reach its maximum speed from a stationary state, the initial velocity v_0 is the speed at which the robotic arm starts moving (usually 0), the final velocity v_t is the speed before the robotic arm stops (ideally also 0), the intermediate running speed v_n is the speed during the uniform motion phase of the robotic arm, and the total distance S is the overall distance the robotic arm needs to cover during the entire motion cycle.

Acceleration Distance:

$$s_1 = \left(v_n^2 - v_0^2\right) / (2a) \tag{6}$$

Deceleration Distance:

$$s_3 = \left(v_n^2 - v_t^2\right) / (2a) \tag{7}$$

Constant Speed Distance:

$$s_2 = s - s_1 - s_3 \tag{8}$$

If $s_2 > 0$, the robotic arm has a uniform motion segment, and the total distance of the acceleration and constant speed segments is $s_k = s_1 + s_2$.

When $s_2 < 0$, the robotic arm needs to decelerate immediately after acceleration, and there is no constant speed segment, as shown in Figure 4(b). Assuming the speed at the intersection of the acceleration and deceleration segments is v_m, the relationship is:

$$s_1 = \left(v_m^2 - v_0^2\right) / (2a) \tag{9}$$

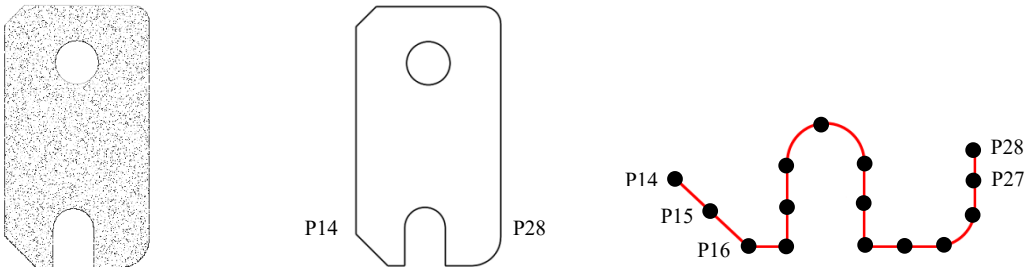
$$s_2 = \left(v_m^2 - v_t^2\right)/(2a) \Rightarrow v_m^2 = \left(2as + v_0^2 + v_t^2\right)/2 \Rightarrow s_1 = \left(2as - v_0^2 + v_t^2\right)/(4as) \tag{10}$$

$$s = s_1 + s_2 \tag{11}$$

In practical applications, such dynamic speed adjustment allows for the implementation of different velocity strategies in various printing segments. For instance, reducing speed while printing complex details enhances fineness and accuracy. This type of velocity planning is crucial for 3D printing robotic arms, as it ensures smooth motion of the arm during the printing process, prevents vibrations caused by sudden speed changes, and thus enhances the quality of printing and the reliability of the robotic arm [20].

4.2 Motion Interpolation

To ensure precise curvilinear motion of the robotic arm during the 3D printing process, advanced motion planning is indispensable. The planning in this solution employs the Linear Approximation method, as shown in Figure 5. It calculates interpolation points between the start point $p_1(x_1, y_1, z_1)$ and endpoint $p_n(x_n, y_n, z_n)$, facilitating the smooth motion of the robotic arm along the trajectory. This strategy implements a trapezoidal velocity profile plan on each micro-segment path.



(a) Motion Trajectory (b) Interpolation Points (c) Local Interpolation Points Diagram

Figure 5. Motion Interpolation Diagram.

1) First, calculate the differences x_{diff}, y_{diff} and between the target coordinates and the current position coordinates to determine the movement requirements of the robotic arm on each coordinate axis.

X Coordinate:

$$x_{\text{diff}} = x_{\text{destination}} - x_{\text{current}} \tag{12}$$

Y Coordinate:

$$y_{\text{diff}} = y_{\text{destination}} - y_{\text{current}} \tag{13}$$

Z Coordinate:

$$z_{\text{diff}} = z_{\text{destination}} - z_{\text{current}} \tag{14}$$

In the formula: $x_{\text{destination}}$ is the x-value of the target point, x_{current} is the x-coordinate value of the current point. The same applies to the other coordinates.

2) Calculate the Euclidean Distance D_{diff} between two points.

$$D_{diff} = \sqrt{x_{diff}^2 + y_{diff}^2 + z_{diff}^2} \tag{15}$$

3) Given the number of segments *n* of the line segment, calculate the length L_{length} of each segment.

$$L_{length\_x} = x_{diff}/n \tag{16}$$

$$L_{length\_y} = y_{diff}/n \tag{17}$$

$$L_{length\_z} = z_{diff} / n \tag{18}$$

In the formula: L_{length_x}, L_{length_y} and L_{length_z} represent the distance that the robotic arm should cover in each micro-movement, corresponding to the projection lengths on the XYZ axes, respectively.

4) For each segment of the path, calculate the inverse kinematics of the robotic arm to determine the angles for the three axes. These angles will then be converted into corresponding electric motor pulse counts to achieve precise control of the robotic arm.

As shown in Figure 6, solving the inverse kinematics for segment L_1 yields the angles for the three axes, and similarly, solving for segment L_2 provides another set of three-axis angles. The pulse counts for the X, Y, and Z directions between points p14 and p15 correspond to the pulse counts for the respective three-axis movement angles. By using this method, fine control over the path and speed of the robotic arm can be achieved, significantly enhancing the quality of 3D printing operations. However, increasing the number of subdivisions directly affects the computation time of the algorithm, thereby impacting the speed of 3D printing. Therefore, a balance between accuracy and printing efficiency needs to be found.

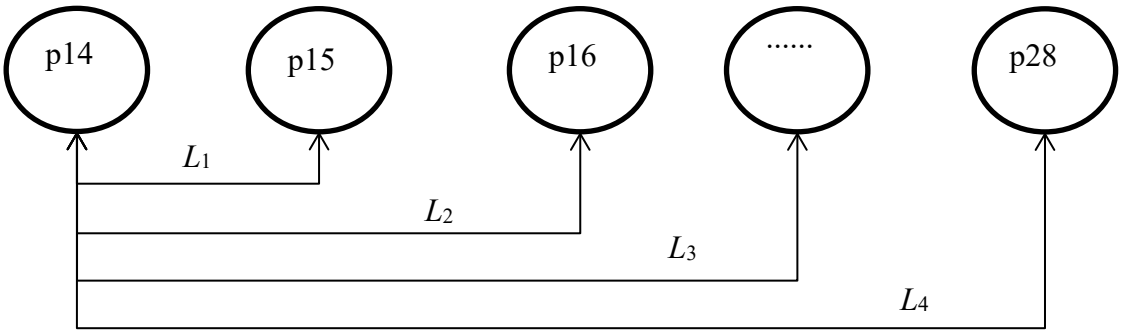


Figure 6. Interpolation Point Calculation Diagram.

5. Experimental Results

In this study, the experimental platform is configured with a Windows 10 x64 operating system, a high-performance computing environment equipped with 16GB RAM, and an Intel(R) Core (TM) i7-6500U CPU @2.5GHz. The development environment is based on the Visual Studio 2013 x64 Integrated Development Environment (IDE). The control algorithm used in the experiments was compiled and uploaded into the Arduino MEGA2560 microcontroller unit (MCU).

In the initial experimental phase, a robotic arm was used for basic shape drawing to verify the accuracy of circular trajectory planning. The experiments showed high precision in drawing circles. Subsequent writing experiments demonstrated the robotic arm's ability to mimic human writing, with the program execution and results illustrated in Figure 7, highlighting its potential for complex tasks.

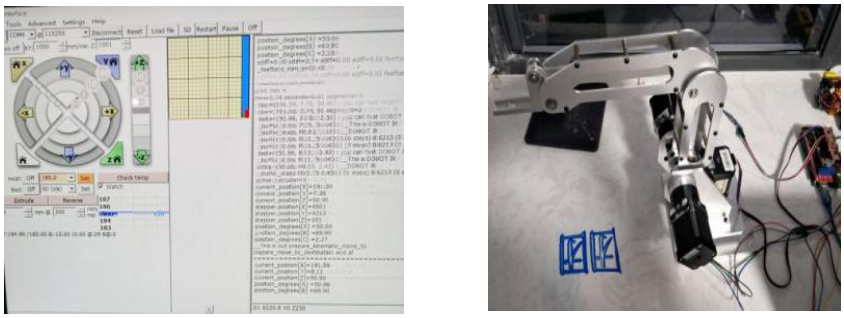


Figure 7. Diagram of the Robotic Arm Writing and its Operating Program.

In the final experiment phase, we focused on applying a robotic arm in 3D printing. The arm was used to print precise mechanical parts like gear waterwheels, showcased in Figure 8. Throughout the printing process, the arm demonstrated impressive stability and accuracy, producing high-quality prints with detailed features and precise

dimensions. This validated our control algorithm's effectiveness and the hardware's efficiency, also indicating the robotic arm's potential in broader manufacturing applications.

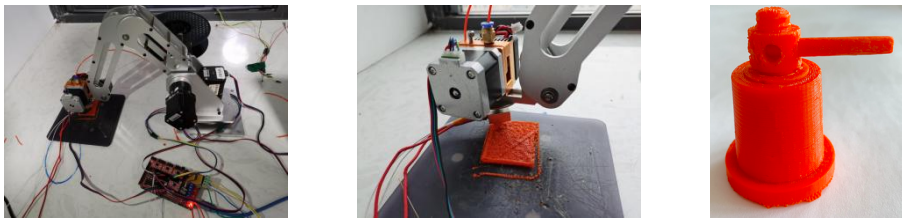


Figure 8. Robotic Arm-Based 3D Printer and Printed Models.

6. Conclusion

This research project successfully developed and integrated an advanced control system for a multi-degree-of-freedom robotic arm, greatly enhancing its 3D printing capabilities. By conducting detailed forward and inverse kinematics analyses, along with intricate path interpolation calculations, the project team was able to significantly enhance the robotic arm's motion stability and printing accuracy. These improvements were pivotal in elevating the performance of the arm in complex 3D printing tasks. The trajectory planning strategies implemented were not only efficient but also adaptable, allowing the robotic arm to execute precise movements and intricate designs, which is a critical requirement in high-quality 3D printing.

Moreover, the project innovatively enhanced traditional 3D printing mechanisms, which enabled the robotic arm to perform multiple functions. This multi-functionality has proven to be a game-changer, effectively increasing work efficiency and reducing operational costs. This breakthrough is particularly relevant in the realm of "New Engineering Courses" where the blend of theoretical knowledge and practical application is paramount. The study sets a remarkable precedent in engineering practice, demonstrating how theoretical concepts can be applied to develop innovative solutions. Additionally, it fosters research innovation thinking, encouraging exploration and creativity in solving engineering challenges, and thus contributing significantly to the advancement of modern engineering solutions.

Funding

This paper is supported by the Nanjing University of Technology Pujiang College Research Project (No. njpj2021-2-02).

References

[1] Wang, Y. P. & Xu, H. L. (2022). Rapid Prototyping Technology and Applications. Xi'an University of Electronic Science and Technology Press, Beijing.

[2] Rodriguez-Labra, J. I., Narakathu, B. B., & Atashbar, M. Z. (2019). Development of a Wireless Robotic Arm Control System Using Piezoelectric Sensors and Neural Networks. 2019 IEEE SENSORS. Presented at the 2019 IEEE SENSORS, Montreal, QC, Canada. https://doi.org/10.1109/sensors43011.2019.8956669.

[3] Daniel G, Bastian H, Franz G, et al. Continuous 3D-Printing for Additive Manufacturing [J]. Rapid Prototyping Journal, 2014, 20(4): 320-327.

[4] Wu, G. Q. (2018). 3D Printing Forming Processes and Materials. Higher Education Press, Beijing.

[5] Al-Dulimi, Z., Wallis, M., Tan, D. K., Maniruzzaman, M., & Nokhodchi, A. (2021). 3D Printing Technology as Innovative Solutions for Biomedical Applications. Drug Discovery Today, 26(2), 360-383. https://doi.org/10.1016/j.drudis.2020.11.013.

[6] Chen, X. & Li, K. (2021). Robotic arm control system based on augmented reality brain-computer interface and computer vision. Journal of Biomedical Engineering.

[7] Shahrubudin, N., Lee, T. C., & Ramlan, R. (2019). An Overview on 3D Printing Technology: Technological, Materials, and Applications. Procedia Manufacturing, 1286-1296. https://doi.org/10.1016/j.promfg.2019.06.089.

[8] Derossi, A., Caporizzi, R., Paolillo, M., & Severini, C. (2021). Programmable Texture Properties of Cereal-Based Snack Mediated by 3D Printing Technology. Journal of Food Engineering, 289, 110160. https://doi.org/10.1016/j.jfoodeng.2020.110160.

[9] Yao, W. L. & Chen, H. C. (2021). Intelligent Color Image Recognition and Mobile Control System for Robotic Arm. 2021 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS). Presented at the 2021 International

Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Hualien City, Taiwan. https://doi.org/10.1109/ispacs51563.2021.9651124.

[10] Shi, Y. S., Zhang, L. C., & Bai, Y. (2015). The Development of 3D Printing Technology and Its Software Implementation. Scientia Sinica Informationis, 45, 197-203.

[11] Kadyroldina, A. T., Orazova, A. Zh., Krasavin, A. L., Kazantsev, I. G., Dyomina, I. A., & Alontseva, D. L. (2022). Development of New Control Algorithms for A Robotic Arm Equipped with A 3d Scanning or Machine Vision System. Bulletin of D. Serikbayev EKTU, 39-59. https://doi.org/10.51885/1561-4212_2022_1_39.

[12] Liu, M., Zhou, H., & Pang, A. (2020). Research on Motion Control System of 6-DOF Robotic Arm. In Proceedings of the 11th International Conference on Modelling, Identification and Control (ICMIC2019). Lecture Notes in Electrical Engineering (pp. 53-61). https://doi.org/10.1007/978-981-15-0474-7_6.

[13] Pal, K., Champaty, B., Nayak, S. K., Anis, A., Banerjee, I., Mohapatra, B., & Ray, S. S. (2019). Wireless Speech Control System for Robotic Arm. International Journal of Biomedical Engineering and Technology, 30(4), 344. https://doi.org/10.1504/ijbet.2019.10022443.

[14] Takacs, K., Koniar, D., Hargas, L., & Hock, O. (2021). Control of Robotic Arm with Visual System. 2021 25th International Conference Electronics. Presented at the 2021 25th International Conference Electronics, Palanga, Lithuania. https://doi.org/10.1109/ieeeconf52705.2021.9467427.

[15] Chen, L., Sun, H., Zhao, W., & Yu, T. (2021). Robotic Arm Control System Based on AI Wearable Acceleration Sensor. Mathematical Problems in Engineering, 1–13. https://doi.org/10.1155/2021/5544375.

[16] Smirnova, M. A., Smirnov, M. N., & Smirnov, N. V. (2022). Multi-Purpose Robotic Arm Control System. Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes, 18(4), 621-630. https://doi.org/10.21638/11701/spbu10.2022.415.

[17] Kong, Z., Rao, S., Yang, H., Lan, W., Leng, Y., & Ge, S. (2022). Eye-Tracking-Based Robotic Arm Control System. 2022 International Conference on Computer Engineering and Artificial Intelligence (ICCEAI). Presented at the 2022 International Conference on Computer Engineering and Artificial Intelligence (ICCEAI), Shijiazhuang, China. https://doi.org/10.1109/icceai55464.2022.00141.

[18] Pribluda, B., Uglovskiy, A., & Korol, V. (2019). Development and Design of the Intellectual Bionic Robotic Arm Control System. 2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM). Presented at the 2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), Sochi, Russia. https://doi.org/10.1109/icieam.2019.8742781.

[19] Aka Driver. (2020). Understanding of the Motion Control Part of the Open-Source CNC Project Marlin 2.0 [Blog post]. Retrieved October 22, 2020, from https://blog.csdn.net/liuzhijun301/article/details/104477187.

[20] Cheng, X. & Kimoto, T. (2022). Kinect-based Data Processing Noncontact Robotic Arm Control System. 2022 16th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS). Presented at the 2022 16th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Dijon, France. https://doi.org/10.1109/sitis57111.2022.00099.