

SIMULATION OF CROSS-PLATFORM OFFLINE PROGRAMMING SYSTEM FOR A MECHATRONIC ROBOT USING THE SPACE ARC INTERPOLATION ALGORITHM

Zhang Hao

Guizhou Normal University Institute of Education, Guiyang 550001, Guizhou, China.

Email: zhhaowork@163.com

Abstract: The purpose is to apply the intelligent algorithm to the construction of a cross-platform offline programming system of a Mechatronics robot, and improve the efficiency of robot programming. First, the trajectory of the Mechatronics robot is planned, and the spatial arc interpolation algorithm, tool coordinate system alignment method, and fast search algorithm are introduced. Then, the cross-platform offline programming system of the Mechatronics robot is proposed. Based on the Unity3d platform, the virtual 3D (Three Dimensional) environment is constructed together with trajectory formulation, a program module, motion simulation, and auxiliary module. Finally, the proposed offline programming system is simulated through experiments. The results show that: the overall positioner angle of the robot increases unidirectionally, which is in line with the actual situation. Meanwhile, the rotation angle of the positioner has fallen in two places, causing the positioner to stop in the reverse direction, so the positioner cannot transit smoothly. The robot can operate continuously without machine jam or pause. The angle changes of different joints are quite different. The radian change of the fourth joint is small, while the radian change of the fifth and sixth joints is the largest, and the change interval is $[0^\circ, 62^\circ]$. The changing trend of the angular velocity of joints 2, 3, 5, and 6 is similar, but the angular velocity of joint 4 does not change. The changing trend of the angular velocity of joint 1 is opposite to that of joints 2 and 5. The results show that the simulation experiment of the Mechatronics robot cross-platform offline programming system based on spatial arc interpolation algorithm is successful, which can provide an experimental basis for the research and development of robot cross-platform offline programming system.

Keywords: Spatial Arc Interpolation Algorithm; Mechatronic Robot; Cross-platform Offline Programming System; Virtual 3D Environment.

1. Introduction

The manufacturing industry is the embodiment of a country's strength, and robot technology has become a crucial factor for the manufacturing industry. Robot technology is an emerging multidisciplinary science that integrates computer science, information science, mechanical science, control science, bionics, and artificial intelligence [1]. It represents the highest level of electromechanical technology today. Industrial robots have become cheaper and more stable, and the labor cost is rising. Industrial robots are getting more efficient, faster, easier to operate, more accurate, and more flexible. Hence, more and more industrial robots are applied to industrial automation production [2, 3].

In recent years, the domestic demand for industrial robots is increasing rapidly. The national sales volume exceeds 15,000 in 2010, reaches 27,000 in 2012, and increases more than 50% to

57,000 in 2014. In 2014, the output value of domestic robots and robotic systems exceeds 100 billion RMB, driving the sales of parts to exceed 300 billion RMB. Although China ranks first in total industrial robot usage, the use density is still very low, with only 30 robots per 10,000 people, less than one-tenth of 323 robots in Japan, and less than half of the global average of 62 robots. Thus, the robot business still has much room for growth in China [4,5]. An industrial robot is a kind of mechanical and electrical equipment with some intelligence and flexibility and can complete a variety of work through programming. Besides, the industrial robot is an important symbol of flexible manufacturing. The programmable ability of the robot determines the flexibility and intelligence of industrial robot functions, and its programming mode determines the task completion ability. The programming methods of the industrial robot include teaching copy programming and offline programming [6, 7].

The research of robot offline programming started earlier outside China. As early as the late 1970s and early 1980s, a lot of energy and funds are invested in the research of path planning and offline programming of industrial robots, which has promoted the continuous development of industrial robot technology [8]. In the 1990s, foreign countries have developed commercial offline programming and simulation software for industrial robots based on offline programming systems for general industrial robots. Companies, such as FANUC and KUKA, have commercial offline programming systems, which focus on different application fields and have been widely used in all walks of life today [9]. Since the 1990s, some research institutions in China have begun to study the offline programming and simulation technology of industrial robots. In recent years, offline programming software for manufacturers continuously emerges, yet there is still no common commercial software product [10]. The research and development of offline programming in China are mostly on campus, and there are many successful and influential experiments, such as ROBS (Rapid Optical Beam Steering) software developed by Tsinghua University, which can plan the kinematics and dynamics trajectory of PUMA (Programmable Universal Manipulation Arm) 560 and similar robots. Compared with foreign practical and commercial program production systems, there is still a big gap between domestic and international robot offline programming. For example, there is no professional process package, no data interface to connect with external CAD (Computer-Aided Design) system [11, 12].

Consequently, the spatial arc interpolation algorithm, tool coordinate alignment method, and fast search algorithm are introduced for robot trajectory planning. Then, a system architecture is proposed of cross-platform offline programming for mechatronic robots, which provides a realistic basis for efficiency improvement of cross-platform offline programming for robots.

2. Method

2.1 Trajectory Planning of Mechatronic Robot

For industrial robots, trajectory planning refers to two directions: one is the curved trajectory of the end of the manipulator, and the other is the curve contour of the displacement, velocity, and acceleration of the manipulator during motion [13]. The trajectory of the robot can be planned in joint space or rectangular space, and the trajectory function must be continuous and smooth for a smooth and steady movement. In joint space planning, the joint variable is expressed as a time

function to plan its first and second-order time derivatives. In rectangular space planning, the attitude, velocity, and acceleration of the hand are expressed as time functions. The corresponding joint displacement, velocity, and acceleration are obtained according to the hand information [14]. Specific algorithms should solve the corresponding parameters. The spatial arc interpolation algorithm is applied in robot trajectory planning together with the tool coordinate alignment and fast search algorithm.

First, the spatial arc interpolation algorithm is introduced [15]. The trajectory in the joint coordinate system refers to the whole movement of the robot from the starting point to the target point with a set percentage of the target point and the maximum joint velocity in the joint coordinate system space through the user or command. Another core of the robot offline programming system is the flowchart of trajectory planning algorithm, as shown in figure 1.

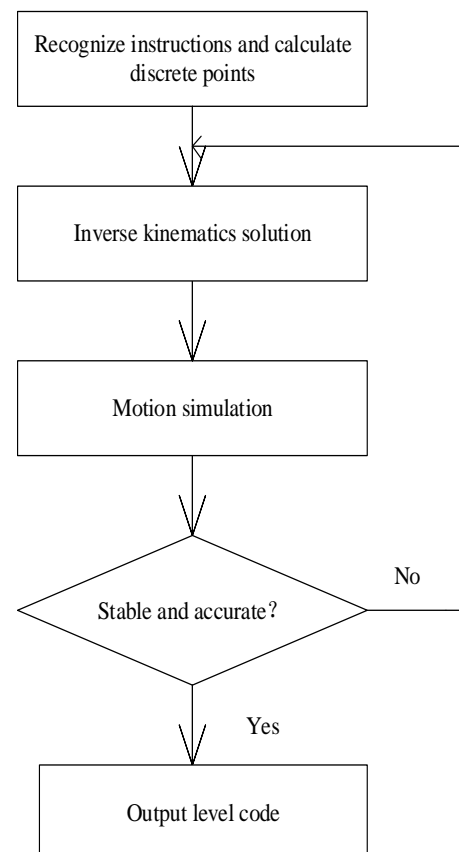


Figure. 1 Flowchart of the trajectory planning algorithm

For the spatial arc interpolation motion in a rectangular space, the motion constraint is different from the linear motion in position interpolation, and the attitude interpolation is consistent with the linear motion. Hence, only the position interpolation in arc motion is introduced below.

The position of the arc should be determined in arc motion interpolation in rectangular space. Since the circular motion command provides the coordinates of three points, the plane and circle positions of the arc can be determined according to these three points. Meanwhile, the collinearity of the given points should be checked. If collinear, the arc interpolation algorithm can be converted to a linear motion interpolation algorithm. In short, the robot program is decoded first, and different motion commands are matched for correct information storage. Afterward, the trajectory of each point is interpolated according to the speed requirement of the system, and then the joint angle of each axis is calculated through the inverse kinematics solution. Subsequently, the motion is simulated if the limit is not exceeded. If the path can be generated accurately, the program code is output for verification on real robot usage. If an error or interference occurs during motion simulation, the path should be optimized, and the program code should be modified until the motion simulation is accurate.

The core of robot trajectory planning is the calculation of discrete points through the program code [16]. The joint interpolation and linear interpolation algorithms are relatively simple since they only equal-divide the joint angle or interpolate according to the spatial line. The arc interpolation needs spatial arc interpolation. During space arc interpolation, the space arc is transformed into a plane arc through coordinate transformation, and then the interpolation result is transformed into the interpolation feed of a space arc through an inverse coordinate transformation. In the robot program, three points can represent a section of the arc. Suppose that the three points are $A(x_1, y_1, z_1)$, $B(x_2, y_2, z_2)$, and $C(x_3, y_3, z_3)$, and the center of the arc is $O(x', y', z')$. Then, the arc plane can be expressed as equation (1).

$$Dx + Ey + Fz + G = 0 \quad (1)$$

In equation (1), $D, E, F,$ and G represent constants, and then equation (2) can be obtained.

$$\begin{aligned} D_1 &= y_1z_2 - y_1z_3 - z_1y_2 + z_1y_3 + y_2z_3 - y_3z_2 \\ E_1 &= -x_1z_2 + x_1z_3 + z_1x_2 - z_1x_3 - x_2z_3 + x_3z_2 \\ F_1 &= x_1y_2 - x_1y_3 - y_1x_2 + y_1x_3 + x_2y_3 - x_3y_2 \\ G_1 &= -x_1y_2z_3 + x_1y_3z_2 + x_2y_1z_3 - x_3y_1z_2 - x_2y_3z_1 + x_3y_2z_1 \end{aligned} \quad (2)$$

Since the distance between each point and the center of the circle is equal, when the radius of the arc is set to be r , then equation (3) can be obtained.

$$\begin{aligned} r^2 &= (x_1 - x')^2 + (y_1 - y')^2 + (z_1 - z')^2 = (x_2 - x')^2 + (y_2 - y')^2 + (z_2 - z')^2 \\ &= (x_3 - x')^2 + (y_3 - y')^2 + (z_3 - z')^2 \end{aligned} \quad (3)$$

Afterward, the coordinates of B and C are brought into equation (1), and the coordinate of the center of the circle can be calculated through the above equations, as shown below.

$$(x', y', z') = - \begin{bmatrix} D_1 & E_1 & F_1 \\ D_2 & E_2 & F_2 \\ D_3 & E_3 & F_3 \end{bmatrix}^{-1} \begin{bmatrix} G_1 \\ G_2 \\ G_3 \end{bmatrix} \quad (4)$$

The coordinate system $\{S, R, T\}$ is established with the position of the center of the circle and the plane of the arc. The direction of the S axis is OP' , and the direction of the normal vector of the arc plane is along the T axis, as shown in figure 2.

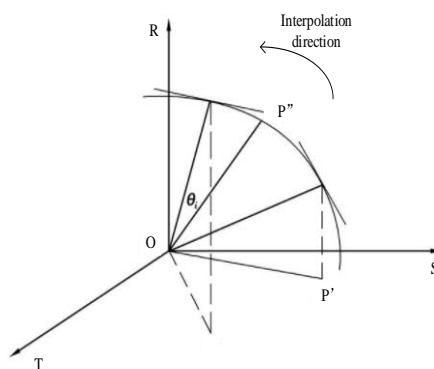


Figure 2 Spatial interpolation algorithm

The direction vectors of the S axis and the T axis can be calculated as below.

$$\vec{S} = \frac{\vec{OP}'}{r} = \left(\frac{x' - x''}{r}, \frac{y' - y''}{r}, \frac{z' - z''}{r} \right) \quad (5)$$

$$\vec{T} = \frac{\vec{OP}' \times \vec{OP}_i}{|\vec{OP}' \times \vec{OP}_i|} = \left(\frac{D}{r_T}, \frac{E}{r_T}, \frac{F}{r_T} \right) \quad (6)$$

In equations (5) and (6), $r_T = \sqrt{D^2 + E^2 + F^2}$, and after the vector R is calculated, the transformation matrix of coordinate system $\{S, R, T\}$ can be expressed as below.

$$\begin{bmatrix} S_j \\ R_j \\ 0 \end{bmatrix} = \begin{bmatrix} S_x & S_y & S_z \\ R_x & R_y & R_z \\ T_x & T_y & T_z \end{bmatrix} \begin{bmatrix} x_j & x' \\ y_j & y' \\ z_j & z' \end{bmatrix} \quad (7)$$

In equation (7), S_j and R_j represent the S and R axes in the transformation matrix of coordinate system {S, R, T}.

The radian ω of the arc can be calculated through the above equations. Suppose that the arc length is denoted as H , and the velocity is denoted as v . Then, $H = \omega \cdot r$, and $H' = H/v$. Equation (8) can be obtained.

$$\begin{cases} \omega' = \omega / v \\ S' = r \cdot \cos \omega' \\ R' = r \cdot \sin \omega' \end{cases} \quad (8)$$

Let the transformation matrix of the coordinate system {S, R, T} to {x, y, z} be the inverse value of equation (7), and then the value of each discrete point in the original coordinate system can be expressed as below.

$$\begin{bmatrix} x_j \\ y_j \\ z_j \end{bmatrix} = \begin{bmatrix} S_x & R_y & T_z \\ S_x & R_y & T_z \\ S_x & R_y & T_z \end{bmatrix} \begin{bmatrix} S_j \\ R_j \\ 0 \end{bmatrix} + \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \quad (9)$$

These discrete points can inversely solve each joint angle for motion simulation. If the simulation result is correct, the trajectory planning is completed and the correct robot program is output.

This algorithm has the following advantages. The host computer only needs to provide the coordinates of three points in space. The vector coordinates are all absolute coordinates without coordinate transformation. Theoretically, all the interpolation points can be dropped on the arc, determining the interpolation direction and the quadrant. Meanwhile, there is no accumulation error. The algorithm still has some shortcomings. The calculation amount is large before interpolation, the CPU (Central Processing Unit) processing time is very long, and fast and small distance movement may produce time error. Thus, the algorithm needs experimental verification [17].

The tool coordinate system alignment algorithm is introduced next. Industrial robots can install different fixtures.

$${}^C_T x = \begin{bmatrix} x_4 - x' \\ y_4 - y' \\ z_4 - z' \end{bmatrix} / \sqrt{(x_4 - x')^2 + (y_4 - y')^2 + (z_4 - z')^2} \quad (11)$$

$${}^C_T z = \begin{bmatrix} x_5 - x_4 \\ y_5 - y_4 \\ z_5 - z_4 \end{bmatrix} / \sqrt{(x_5 - x_4)^2 + (y_5 - y_4)^2 + (z_5 - z_4)^2} \quad (12)$$

The accuracy of the tool coordinate system affects the trajectory accuracy of the industrial robot. The calibration of the tool coordinate system is an essential function of the industrial robot. At present, the tool coordinate system calibration methods of domestic and foreign industrial robot manufacturers are slightly different, including the external reference method and multi-point calibration method [18]. The external reference calibration method only needs to align the tool with the external reference point for calibration. It is fast and simple and depends on the external reference of the robot.

Presently, most industrial robots can calibrate with multi-points through a tool coordinate system [19, 20]. The proposed offline programming system also can calibrate the tool coordinate system with multi-points and can be used in the educational field. Thus, learners can simulate and understand the tool coordinate system through real robots. The five-point method can calibrate the position and attitude of TCP (Tool Center Position).

The TCP attitude is calibrated through additional two points on the three-point method [21]. Here is the operation flow. The robot returns to the origin point where the origin tool and the Z-axis coincide. Then, it moves along the X-axis of TCP at any length, and the point in the X-direction is calibrated. Afterward, it moves along the Z-axis of TCP at any length, and the point in the Z direction is calibrated. With the coincidence point as the origin, the attitude of TCP is calculated through the X-direction point and Z-direction point, and the Euler angle is calculated. During TCP position calibration, the transformation matrix ${}^C_Z O$ can be obtained through the forward solution of the robot from the end link coordinate system to the calibration point base coordinate system. The principle of the algorithm is shown below.

$${}^C_Z r \cdot {}^Z_T r = {}^C_T r \quad (10)$$

In equation (10), r denotes the arc radius, ${}^C_T r$ can be decomposed into $\begin{bmatrix} {}^C_T x & {}^C_T y & {}^C_T z \end{bmatrix}$, and the following equations can be obtained.

According to the principle that the coordinate system is perpendicular to each other, the following equation can be obtained.

$${}^C_T y = {}^C_T x \times {}^C_T z \quad (13)$$

The attitude matrix of TCP can be obtained as below.

$${}^Z_T r = {}^C_T r^{-1} \cdot {}^C_T r \quad (14)$$

Then, the Euler angle of the coordinate system can be calculated [22].

Finally, the fast search algorithm is introduced [23]. The above explanation shows that the trajectory planning algorithm of the positioner and industrial robot axis linkage depends on the calculation of all cases and manual selection. For more trajectory points and higher accuracy, long time and large memory are required, so an automatic parallel algorithm is urgently needed for a quicker optimal path selection. Consequently, the fast search algorithm is proposed. A search algorithm can construct the solution tree according to the initial conditions and extension rules, and it can find the nodes in line with the target state. A search algorithm can be divided into two parts according to the algorithm implementation: control structure (the way of expanding nodes) and generation system (the way of expanding nodes). All algorithms are optimized and improved through modification of control structure. The initial state corresponds to the root node, and the target state corresponds to the target node. The first node is called the parent node, and the next node is called the child node. Nodes in the same layer are siblings, and the children generated by the parent node are called extensions. The search process is to find the path from the root node to the target node and find the optimal solution. The implementation of the search algorithm is similar to the traversal of a graph or tree.

The fast search algorithm is an improvement of the commonly used algorithm and can calculate the rotation angle of relatives.

First, the master-slave structure will be retained of the positioner, as the active arm and the industrial robot as the passive arm. After been imported, the trajectory should be fitting on small lines. Instead of a large amount of calculation, 30° point of the positioner is taken as the initial point of the robot, and then the point with the smallest curvature on the trajectory is taken as the starting path point. The reason is that the vector is close to 30 of the path point with the smallest curvature of No.30 positioner. Such is the best attitude for industrial robots, not easy to produce singularity and other

unusable points. Then, the other path points are transferred to the 30° line one by one for attitude calculation. Here, the length of the small segment is set as 30mm, and the rotation angle of each path point to the positioner is stored clockwise down the positioner, as shown in figure 3.

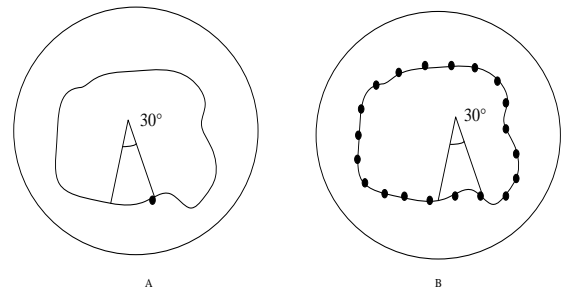


Figure. 3 Point selection of trajectory planning (A: Initial point B: One by one path point)

Moreover, because of the adopted cubic polynomial fitting, the fast search algorithm is likely to fail in complex path welding cases, such as multiple inversion of the positioner. Meanwhile, the algorithm is based on the assumption that the time interval between the path points is the same. In some cases, such as high-speed trajectory planning, the path points should be allocated within the time interval, so that the endpoint of the robot will not deviate the positioner 30° too much or lead to singularity and other unavailable points.

According to the fitted path points, the angle of the positioner is divided into four parts, and the attitude can be obtained of the path point under 0°, 90°, 180°, and 270° rotation angles of the positioner, as shown in ABCD in figure 4.

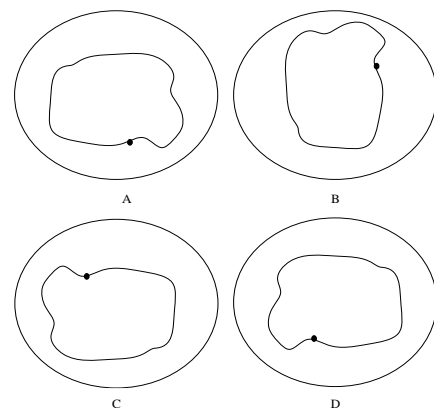


Figure. 4 Robot attitude under different rotation angles of positioner (A: 0°, B: 90°, C:180°, and D:270°)

2.2 Design of Cross-platform Offline Programming System for Mechatronic Robot

Unity [24] is the most professional, stable, efficient, and supporting cross-platform engine on the market.

The Unity3d platform is selected for a cross-platform robot offline programming system. The program adds joint motion properties, assembly constraint properties, and algorithm library according to the imported three-dimensional model.

An offline programming system consists of several parts, including a virtual three - dimensional

environment, trajectory formulation, kinematics control, program module, motion simulation, an operation detection module, an interaction interface, and auxiliary modules.

The functions of the platform components are shown in table 1.

Table 1 Composition of cross-platform offline programming system for mechatronic robot

Components	Primary contents
Virtual 3D (Three dimensional) environment	SolidWorks can construct a model and output an STL format file, and the file is input into 3dsmax. FBX format is generated through optimization and input into the Unity3D platform. Then, the virtual devices, including workshop scenes, robots, and artifacts, positioners are defined.
Auxiliary modules	In production, calibration technology and communication interface must be included in the offline programming system. In the simulation experiment, only the calibrated robot can move along the trajectory. The robot for the tool coordinate system alignment is also included in the auxiliary module. The simulation result is accurate only after TCP alignment. Communication interface connects offline programming system and robots, as an inseparable part.
Algorithm library	The algorithm library is the key technology to the offline programming system. The spatial arc interpolation algorithm is applied to the mobile platform with low power consumption and low CPU frequency. The robot motion control algorithm and trajectory planning algorithm should also be included in the algorithm library. The positive solution and inverse solution in robot motion control algorithms are the foundation of motion simulation. Trajectory planning designs the motions of 6R robots and the multi-axis coordinated motion that uses external axes such as positioners. An automatic search algorithm can find the optimal solution. The program conversion module is included in the program module. Since various controllers are used in different robot brands, the programming languages for execution are also diverse. A program conversion module should be developed for the universality of the offline programming system.

The system architecture is shown in figure 5 of the proposed cross-platform offline programming of mechatronic robots.

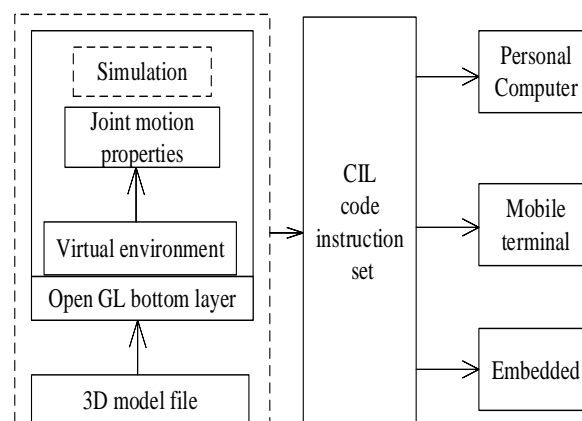


Figure. 5 The system architecture of cross-platform offline programming of the mechatronic robot

The STL file of the 3D robot model is segmented through the proposed robot offline programming system, and joint motion property and assembly constraint property are added to each rigid body module. Then, each virtual device and scene are displayed through the 3D virtual reality technology of Unity [25]. Afterward, the robot's motion is simulated and its trajectory is planned through the algorithm library.

The algorithm library is the core technology of the robot offline programming system, and the robot motion simulation and trajectory planning are the core of the software.

The forward and inverse solution algorithms and the space arc interpolation algorithm are the basis of the motion simulation. The fast search algorithm is the most difficult part of the proposed system design, and It can quickly plan the coordinated motion path between the robot and the positioner. Consequently, the robot and the positioner can move smoothly.

2.3 Simulation Environment and Parameters

(1) Easton industrial robot ER-16 and its controller are chosen as the hardware platform. The D-H parameters of the robot are shown in table 2.

Table 2 D-H parameters of the robot

Joint	1	2	3	4	5	6
$d_i(\text{mm})$	-408	0	0	-720	0	-90
$\alpha_i(\text{mm})$	150	660	120	0	0	0

(2) The Origin2018 64Bit is utilized for data visualization.

3. Results

3.1 Relationship between Path Point and Positioner Angle of the Mechatronic Robot

First, the relationship is analyzed between the path point of the mechatronic robot and the positioner angle. The results are shown in figure 6.

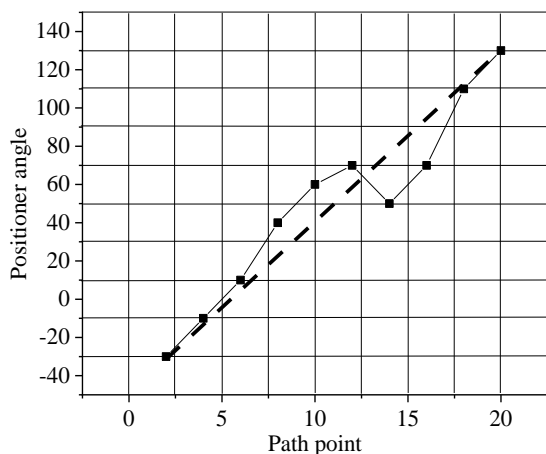


Figure. 6 Relationship between path point of robot and positioner angle

Figure 6 shows that the line chart of the positioner angle can be obtained by connecting each point. The overall positioner angles increase unidirectionally, which is in line with the actual situation. Yet, there are two places where the rotation angle of the positioner has fallen. This will cause the positioner to stop in reverse, and the positioner cannot transit smoothly. The broken line should be fit with a polynomial to get a smooth

curve. Meanwhile, the positioner angles corresponding to each path point are obtained. Since the shedding process requires as uniform welding as possible, it can be assumed that the time between each path point is equal. The position and attitude can be calculated of the path point relative to the positioner through the angle of the path point corresponding to the positioner. Then, the position and attitude are calculated of the path point relative to the industrial robot. Consequently, the joint angle of the industrial robot corresponding to the current path point is calculated through the inverse kinematics solution, such as the ordinary seven-axis linkage algorithm. This fast search algorithm is much faster than the whole algorithm.

3.2 Simulation Results of Mechatronics Robot Operation

The simulation process of the mechatronics robot is analyzed, as shown in figure 7.

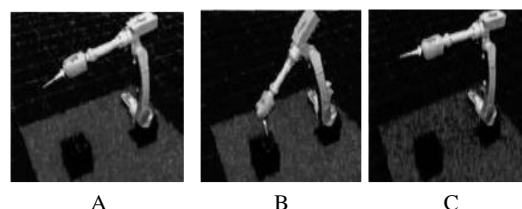


Figure. 7 Simulation process of the mechatronics robot (A: Initial attitude of robot B: Robot operation attitude C: Attitude of robot stop operation)

Figure 7 shows that the robot can operate continuously without machine jam or pause. This is because an offline programming system is constructed for mechatronics robots. The model and parameters of the robotic devices can be input into

the system, and the simulation model of the robot is constructed. Besides, the system is compatible with most serial robots on the market. The offline programming system is developed based on the Unity3D platform, which has good rendering abilities and a sound motion simulation effect.

3.3 Work of Each Joint of the Mechatronic Robot

The change of radian of each joint angle of the robot is visualized and analyzed. The results are shown in figure 8.

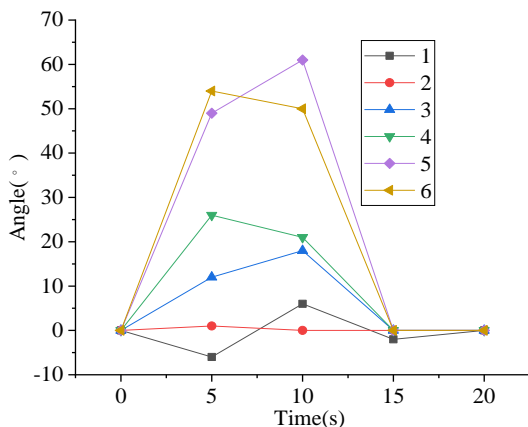


Figure 8 The change of each joint angle and radian of the robot

Figure 8 shows that the angle changes of different joints are quite different. The radian change of the fourth joint is medium, while that of the fifth and sixth joints is the largest, and the change interval is $[0^\circ, 62^\circ]$.

Then, the angular velocity change of each joint of the robot is analyzed visually, and the results are shown in figure 9.

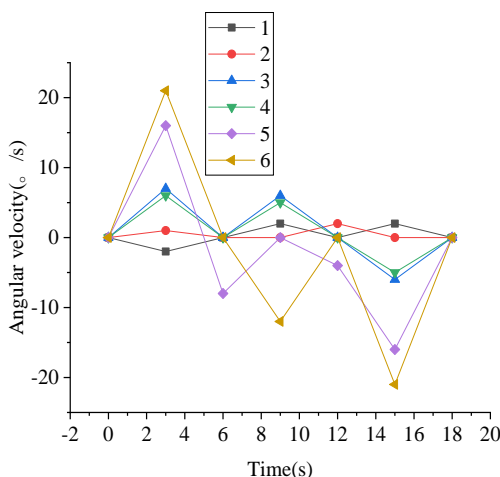


Figure 9 The change of angular velocity of each joint of the robot

Figure 9 shows that the changing trend of the angular velocity of joints 2, 3, 5, and 6 is similar, while the changing range of angular velocity of joint 6 is $[-22, 22]$. The angular velocity of joint 4 doesn't change. The changing trend of the angular velocity of joint 1 is opposite to that of joints 2 and 5

4. Conclusion

First, the robot trajectory planning method is introduced, including the spatial arc interpolation algorithm, tool coordinate system alignment method, and fast search algorithm. Then, the system architecture is proposed of cross-platform offline programming for mechatronics robot, containing virtual three-dimensional environment, algorithm library, and auxiliary modules. Afterward, the proposed system is simulated through the ER16 robot. The results show that the overall positioner angle of the mechatronics robot increases unidirectionally, which is in line with reality. The robot can operate continuously without machine jam or pause. The change is different in the radian and angular velocity of each joint angle of the robot, but the changing trend of the angular velocity of some joint angles is the same. There are also some shortcomings. First, the algorithms proposed in this study are relatively simple, and their inclusiveness and stability need to be improved. Second, the proposed offline programming system is only for a single robot and does not involve the cooperation between robots or assembly line operation. In future research, the focus should be on multi-robot cooperation, which is also the direction of cross-platform offline programming for robots.

Acknowledgment

This work was supported by Humanities and Social Sciences of Higher Education Institutions of Guizhou Provincial Education Department.

References

- [1] Johnson M, Shrewsbury B, Bertrand S, et al. (2018) Team IHMC's Lessons Learned from the DARPA Robotics Challenge: Finding Data in the Rubble. *Journal of Field Robotics*, 34(2),241-261.
- [2] Huang L Y, Lee Y P, Moon Y S, et al. (2017) Noble Implementation of Motor Driver with All Programmable SoC for Humanoid Robot or Industrial Device. *International Journal of Humanoid Robotics*, 14(4),1750028.
- [3] Tang G, Webb P, Thrower J. (2018) The development and evaluation of Robot Light Skin: A novel robot signalling system to improve communication in industrial human-robot

- collaboration. *Robotics and Computer-Integrated Manufacturing*, 56, 85-94.
- [4] Mahmud K, Ravishankar J, Hossain M J, et al. (2020) The Impact of Prediction Errors in the Domestic Peak Power Demand Management[J]. *IEEE Transactions on Industrial Informatics*, 16(7),4567-4579.
- [5] Poignant S, Laffon M. (2018) Impact of Moderate Hyperchloremia on Clinical Outcomes in Intracerebral Hemorrhage Patients. Is There Still Room for Continuous Infusion of 3% Hypertonic Saline?. *Critical Care Medicine*, 46(2), e178.
- [6] Van Niekerk J, Webb P. (2016) The effectiveness of brain-compatible blended learning material in the teaching of programming logic. *Computers & Education*, 103(dec.),16-27.
- [7] Vancza, Jozsef, Kemeny, et al. (2016) planning and offline programming for robotic remote laser welding systems. *International Journal of Computer Integrated Manufacturing*, 29(10/12),1287-1306.
- [8] Maiolino P, Woolley R, Branson D, et al. (2017) Flexible robot sealant dispensing cell using RGB-D sensor and off-line programming[J]. *Robotics and Computer-Integrated Manufacturing*, 48,188-195.
- [9] Schaal T P, Arango J, Wolc A, et al. (2016) Commercial Hy-Line W-36 pullet and laying hen venous blood gas and chemistry profiles utilizing the portable i-STAT@1 analyzer. *Poultry Science*, 95(2),466-471.
- [10] Geng J. (2016) Research and Review on the Java Multi-thread Programming and Its Further Development Tendency. *International Journal of Technology, Management*, 000(005), P.4-6.
- [11] Wang Y, Huang Q, Wang Q, et al. (2021) Analysis of the Structural Characteristics of the Online Social Network of Chinese Professional Athletes[J]. *Complexity*, 2021(6),1-10.
- [12] Fu, Chen, Baodan, et al. (2017) Enhanced recycling network for spent e-bicycle batteries: A case study in Xuzhou, China - *ScienceDirect. Waste Management*, 60,660-665.
- [13] Min H, Li S. (2016) Investigation on the Trajectory Planning of Spray Gun of Spray Robot and Its Spraying Effect. *International Journal of Smart Home*, 10(4),183-192.
- [14] Chen G, Guo W, Jia Q, et al. (2018) Failure treatment strategy and fault-tolerant path planning of a space manipulator with free-swinging joint failure. *Chinese Journal of Aeronautics*, 31(12),109-124.
- [15] Welch J, Kópházi, J, Owens A R, et al. (2017) A geometry preserving, conservative, mesh-to-mesh isogeometric interpolation algorithm for spatial adaptivity of the multigroup, second-order even-parity form of the neutron transport equation. *Journal of Computational Physics*, 347,129-146.
- [16] Privat C, Madurga S, Mas F. (2020) On the Use of the Discrete Constant pH Molecular Dynamics to Describe the Conformational Space of Peptides. *Polymers*, 13(1),99.
- [17] Charytanowicz M, Perzanowski K, Januszcak M, et al. (2020) Application of Complete Gradient Clustering Algorithm for analysis of wildlife spatial distribution. *Ecological Indicators*, 113,106216.
- [18] Ragay-Enot M, Lee Y H, Kim Y G. (2017) Fabrication of a mini multi-fixed-point cell for the calibration of industrial platinum resistance thermometers. *Measurement Science and Technology*, 28(7),075007.
- [19] Brookes V J, Barry S C, Hernández-Jover, M, et al. (2017) Point of truth calibration for disease prioritisation-A case study of prioritisation of exotic diseases for the pig industry in Australia.. *Preventive Veterinary Medicine*, 139,20-32.
- [20] Li X, Zhu J, Tian F, et al. (2020) An improved rapid prediction method of the milling tool point frequency response function. *The International Journal of Advanced Manufacturing Technology*, 110(3),841-852.
- [21] Wu J, Yuen C, Cheng B, et al. (2016) Streaming High-Quality Mobile Video with Multipath TCP in Heterogeneous Wireless Networks. *IEEE Transactions on Mobile Computing*, 15(9),2345-2361.
- [22] Joukov V, Esi J, Westermann K, et al. (2020) Estimation and Observability Analysis of Human Motion on Lie Groups. *IEEE Transactions on Cybernetics*, 50(3),1321-1332.
- [23] Zhai J, Gao L, Li S. (2016) Fast control optimization for switched linear systems based on harmony search algorithm. *Neurocomputing*, 185(apr.12),183-190.
- [24] Zhang Q, Chang N, Shang K. (2019) Design and exploration of virtual marine ship engine room system based on Unity3D platform. *Journal of Intelligent and Fuzzy Systems*, 38(5),1-7.
- [25] Chen Y, Wu Z. (2018) Study on the application of particle swarm optimization in the virtual reality of the modified wood furniture. *Journal of Intelligent and Fuzzy Systems*, 35(2),1-7.