ingegneria sismica
International Journal of Earthquake Engineering
Anno XLIII - Num. 1 - 2026

Research on automated calibration technology for manual position of moment-aware rotary machinery based on artificial intelligence and mathematical modeling

Xiongwei Jia[1,*] *and Mei Kang*[2]

[1] School of Mechanical and Electrical Engineering, Xi'an Traffic Engineering University, Xi'an, Shaanxi, 710300, China
[2] Shaanxi HanDe Axle Co., Ltd., Xi'an, Shaanxi, 710200, China

SUMMARY: *The research improves the ability of robots in their perceptions in terms of robotic arms by developing a torque-aware multi-station alignment approach to help tackle the problem of obtaining accurate multi-station alignment. The kinematics model of the robot arm is defined using a better Denavit–Hartenberg (D-H) approach for the establishment of mapping the trajectories from the Cartesian space to the joint space. This study proposes an enhanced rapid random tree (RRT) algorithm for conducting path planning of the rotary manipulator by taking advantage of the benefits of bidirectional and target-biased RRT methods. A simulation control system for the robot arm is designed using computational torque approaches. Based on the above, this study develops an advanced nine-point calibration approach. Using multiple pose data obtained from the same spatial point, the length of the robot sixth axis is calculated; thus, the sixth axis can be determined after the end-effector installation. Experiments showed that using the enhanced RRT algorithm resulted in the robot achieving minimum path costs, runtimes, and nodes of 1701.7 mm, 7.8 seconds, and 366, respectively. Nonlinear moment controller achieved improved performance in trajectory errors as well as control speeds. Using the three-stage alignment approach, angular and positional errors could be less than 0.06° and 0.20 mm, respectively.*

KEYWORDS: *D-H method; RRT algorithm; path planning; computational moment method; nine-point calibration method; manual position automation calibration*

1 Introduction

With the development of manufacturing techniques using artificial intelligence, artificial intelligent robot arms play an important part in the industry's manufacture process [1]. Among these, machine vision stands as a key representative technology in AI development, requiring robots to possess a certain level of intelligence while observing and judging image objects in ways that surpass human vision [2, 3]. Simultaneously, machine vision calibration stands as a core technology for industrial robots. Through the machine vision system, robots identify and locate workpieces within the camera's field of view. They then convert the coordinate values from the image coordinate system into those of the robotic arm's coordinate system, feeding this data back to the robot's motion control system to achieve workpiece grasping [4, 5]. The calibration of the machine vision system is precisely to establish the relationship between the image coordinate system and the manipulator coordinate system. Therefore, the calibration of

*jxw_0328@163.com
https://doi.org/10.65102/is2026370
Published: 30 April, 2026

the vision system plays a crucial role [6].

The accuracy of manipulator calibration depends on the precision of motion control and the accuracy of the kinematic parameters used in the control model [7]. Kinematics parameters consist of nominal constants and joint variables. Deviations between nominal and actual values of these constants arise from machining, installation, and assembly of mechanical components. Direct use of nominal values in control significantly compromises robotic accuracy [8]. To enhance precision, mathematical modeling can identify kinematics parameters and establish theoretical relationships.

The modern scientific community seeks solutions in self-calibrations. Self-calibration can be performed by means of artificial intelligence algorithms and mathematical modeling [9]. For example, Kong et al. used a long-short term memory extended Kalman filter (LSTM-EKF) algorithm in robot calibration based on the use of laser tracking and pose estimation of the robots. It helps to reveal the distribution and relations between calibration parameters. The presented approach is rather effective when it comes to solving the problem of high-load and high-speed robot calibration [10]. Moreover, Yang et al. introduced the combination of extended Kalman filtering and residual neural networks for constructing a kinematic model of robotic manipulators that could help reduce geometric and non-geometric errors and provide significant advances in calibration accuracy [11]. Sayour et al. proved that the approach of using a MATLAB-based convolutional neural network (CNN) for calibrating robotic arms is a source of innovation in grasping and autonomous manipulation because of the positive results obtained during validation experiments [12]. Nussibaliyeva et al. suggested the integration of artificial vision systems and machine-to-machine communications in order to speed up sorting and packaging tasks. These authors managed to break new ground concerning the accurate positioning of robots that work with moving objects [13]. Taken together, all the sources reviewed above have accumulated significant data for the use of artificial intelligence algorithms in robotic and manipulator self-calibration research.

There have been many research studies done on calibration errors and accuracy. Li, Z. et al. employed the Levenberg-Marquardt method and the extended Kalman filtering technique to accurately calibrate and adjust the kinematics parameters of robots, thus minimizing errors and achieving high industrial standards[14]. In order to solve the issue of poor accuracy in positioning of robotic arms, Li, Z. et al. developed a mathematical model with regularization to minimize the overfitting of the model. The results achieved were highly satisfying through several experiments conducted [15].

In an attempt to mitigate deficiencies in motion, and relative pose estimation of mobile manipulators, Fan et al. came up with a visual rapid base frame calibration technique that would enable computation of the coordinate systems of marked points through images taken at various positions and robotic joints [16]. Then, the relative pose between the two robot base frames was estimated in real time based on the coordinate systems obtained above [16]. Cao et al. developed a novel approach for robot calibration, which combines several calibration algorithms. Through geometric error optimization, improved position and orientation were attained in robot operations. The feasibility of this algorithm was verified using a Stewart platform, and there was noted an improved accuracy of robot poses [17]. Using the error correction matrix as a basis, Liu et al. came up with a calibration technique to compensate for multi-source errors, such as misassembly, in robots [18]. Using differential matrices and left-multiplied perturbation, this technique ensures accurate joint calibration in the presence of multi-source errors, and this was proved through simulation experiments [18]. Inspired by previous works, Balanji et al. designed a computer vision-based calibration approach, relying on one camera and ArUco markers, and also modeling the kinematic model of the robot. With real-life data, the results showed positional errors of 2.5 mm and orientation errors of 0.2°, and

therefore proving to be feasible [19].

The research focuses on the application of artificial intelligence and mathematical modeling in developing automation calibration and robotic arms motion planning control in rotating equipment workstations. Specifically, an improved rapid traversing tree (RRT) algorithm for avoiding obstacles in robotic arms motion planning is put forward. In this algorithm, the advantages of both bidirectional RRT and target-biased RRT algorithms are taken into consideration, through assigning virtual targets and setting step size when expanding the random trees. The redundant nodes along the obstacle-avoiding path are eliminated by bidirectional pruning, and the path is smoothed using cubic B-spline. For the interpolation of the robot arm position and orientation, linear interpolation and SLERP (spherical linear interpolation) are applied. Meanwhile, an optimized D-H methodology is used to build up the robot arm kinematics model, realizing the transition between Cartesian coordinate and joint coordinate. Afterwards, Lagrangian and Newton-Euler methods are adopted to establish the robot arm's dynamic model by considering the forward and inverse dynamic characteristics of the robot arm. A simulation system for the whole robotic system can be built based on the computed torque controller. As a solution for the problem that accuracy in traditional nine-point calibration depends highly on the precision of robot arm parameters and camera, an enhanced nine-point calibration method is put forward, where multiple sets of poses are obtained around the reference point in the workspace to obtain the length of the sixth axis, hence obtaining more accurate parameters for nine-point calibration.

2 Kinematic Modeling and Motion Trajectory Control of Torque-Sensitive Rotating Manipulators

2.1 Kinematic Modeling of Rotating Robotic Arms

The six-axis robotic arm was modeled using the D-H parameter method [20], establishing the joint coordinate system shown in Figure 1. Here, a_j denotes the link length, α_j represents the link twist angle, d_j indicates the link offset distance, and θ_j signifies the joint angle of the robotic arm. x_i, y_i, z_i denotes the coordinate system axes at each joint of the robotic arm.

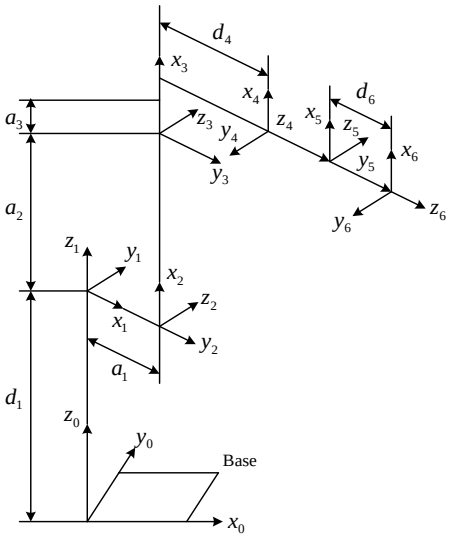


Figure 1: Coordinate system of the robotic arm joint

The D-H parameter table can be derived from the structural parameters of the robotic arm and its joint coordinate system. The specific joint parameters are shown in Table 1.

Table 1: Robotic arm D-H parameter

Arthrosis	a_{j-1}	a_{j-1}/(°)	d_j	θ_j	Joint limit/(°)
1	0	0	d_1	θ_1	180
2	a_1	-90	0	θ_2	145
3	a_2	0	0	θ_3	110
4	a_3	-90	d_4	θ_4	320
5	0	90	0	θ_5	200
6	0	-90	d_6	θ_6	360

Based on the D-H parameter table, the expression for the D-H matrix T is obtained, namely:

$$T=\begin{bmatrix} cos\theta_j & -sin\theta_j cos\alpha_j & sin\theta_j sin\alpha_j & a_j cos\theta_j \\ sin\theta_j & cos\theta_j cos\alpha_j & -cos\theta_j sin\alpha_j & a_j sin\theta_j \\ 0 & sin\alpha_j & cos\alpha_j & d_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Given the joint angles $[\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6]$ of the robotic arm, the end-effector's pose $[x,y,z,\alpha,\beta,\gamma]$ can be determined. Thus:

$${}^0_6T=\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

In the equation: n_x,o_x,a_x represents the first part of the rotation matrix, denoting rotation about the x axis; n_y,o_y,a_y represents the second part of the rotation matrix, denoting rotation about the γ axis; n_z,o_z,a_z represents the third part of the rotation matrix, denoting rotation about the z axis; p_x,p_y,p_z is the translation vector, corresponding to the translation amount along the x,y,z axis.

2.2 Path Planning for an Improved RRT Algorithm

2.2.1 Improving the RRT Algorithm

1) Setting Virtual Goal Points:

To enhance the search efficiency of the algorithm, virtual goal points are established during the extension of the bidirectional RRT algorithm [21]. This allows the random tree to treat these virtual points as actual goal points during the goal-bias process. The steps for generating virtual goal points are as follows:

(1) If the line segment connecting the initial point and the goal point does not intersect any

obstacles, the midpoint of this line segment is selected as the virtual goal point.

(2) When the line connecting the starting point and the target collides with an obstacle, assuming the starting point's coordinates are (x_1, y_1, z_1) and the target point's coordinates are (x_2, y_2, z_2), the equation of line $L1$ is:

$$\frac{(x-x_1)}{(x_1-x_2)} = \frac{(y-y_1)}{(y_1-y_2)} = \frac{(z-z_1)}{(z_1-z_2)} \tag{3}$$

Draw perpendicular lines $L2$ and $L3$ to line $L1$ through the midpoint $A(x_m, y_m, z_m)$ of the initial point and target point. Lines $L2$ and $L3$ are parallel to the horizontal plane and plumb plane, respectively. Let the direction vectors of lines $L2$ and $L3$ be (n_1, m_1, p_1) and (n_2, m_2, p_2), respectively. The equations of lines $L2$ and $L3$ are:

$$\begin{cases} \dfrac{x-x_0}{n_1} = \dfrac{y-y_0}{m_1} = \dfrac{z-z_0}{p_1} \\ z-z_0 = 0 \end{cases} \tag{4}$$

$$\begin{cases} \dfrac{x-x_0}{n_2} = \dfrac{y-y_0}{m_2} = \dfrac{z-z_0}{p_2} \\ y-y_0 = 0 \end{cases} \tag{5}$$

With center $A(x_n/2, y_n/2, z_n/2)$ and radius r, the equation of the sphere is:

$$(x-\frac{x_m}{2})^2 + (y-\frac{y_m}{2})^2 + (z-\frac{z_m}{2})^2 = (r)^2 \tag{6}$$

By solving equations (4) to (6) simultaneously, the intersection point between sphere O and line $L2, L3$ can be determined. If the intersection points between sphere O and line $L2, L3$ are both within the obstacle, increase the sphere's radius by $1/\rho$ and recalculate until at least one intersection point between sphere $L2, L3$ and line 6 lies outside the obstacle. If only one intersection point lies outside the obstacle, designate that point as the virtual target point. If multiple intersection points lie outside the obstacle, select the point farthest from the obstacle as the virtual target point.

2) Adjusting Step Size:

To further enhance the algorithm's search efficiency, a spherical safety zone with a radius of R is established around obstacles. The step size for producing a new node depends on whether the node falls within the predefined safety zone. If it does, then the step size is modified to create a new node without checking for any collision. Otherwise, the same step size is used to create a new node and then check for any collision.

The specific process for generating new nodes is as follows:

If node X_{newl} generated from parent node X_{nearst} is within the safe region, then:

Change the step size to regenerate new node X_{new2}, Retain node X_{new2}, Discard node X_{new1}. If the newly generated node X_{new3} is in the unsafe region, then: Connect node X_{new3}. Perform collision detection on node X_{nearat}. If a collision occurs, discard node X_{new3};

otherwise, retain node X_{new3}. Regenerate node X_{new} as shown in Equation (7):

$$X_{new} = X_{new} + p1 * X_{new} \tag{7}$$

In the formula, $p1$ represents the step size increment coefficient.

2.2.2 Shortest Path

Even with the enhancements in the RRT algorithm through the inclusion of target bias points and different stride lengths, there still seems to be a problem with the number of redundant nodes used in the obstacle-avoiding paths. In order to solve this problem, a two-way node deletion scheme is used to determine the shortest path by deleting unnecessary nodes from both starting points and ending points. After eliminating unnecessary nodes on both sides, the lengths of the two generated paths are compared, and the shorter one is kept. The principle of redundant node deletion is illustrated in Figure 2.

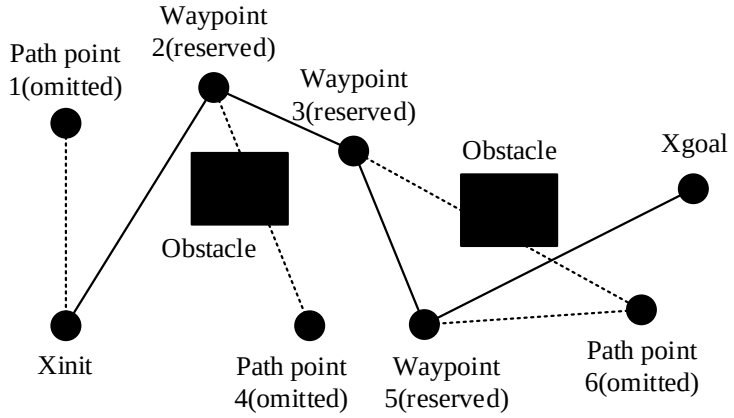


Figure 2: Principle of deleting Redundant nodes

The detailed steps are as follows:

(1) Obtain a collision-free path point set $M1$ from the initial point to the target point by improving the RRT algorithm.

(2) First connect the initial point to path point 1. If no collision occurs between the initial point and path point 1, then connect the initial point to path point 2, and so on, until a collision occurs between two points. Retain the node preceding the colliding path point and use it as the parent node to repeat the above operation until reaching the target point.

(3) Starting from the destination point, connect to the preceding node and repeat step (2) until reaching the starting point.

(4) Compare the path lengths after removing redundant nodes from both the starting point and destination point. Retain the nodes from the shorter path and connect the retained path nodes.

2.2.3 Path Smoothing

Removing redundant nodes from the obstacle-avoidance path shortens its length but introduces smoothness issues at turning points. To address this, cubic B spline curves are employed to smooth the obstacle-avoidance path. The equation for the B spline curve is as follows:

$$C(u) = \sum_{i=0}^{n} N_{i,} k(u) P_i \tag{8}$$

In the formula, $N_i, k(u)(i = 0,1,\cdots,n)$ represents the k nd-degree B -spline basis function, and $P_i(i = 0,1,\cdots,n)$ denotes the control vertex determining the control polygon of the B -spline.

The recursive formula for the k th-degree B -spline basis function is:

$$N_{i,0}(u) = \begin{cases} 1 & if\ u_i \le u \le u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$N_{i,k}(u) = \frac{u - u_i}{u_{i+k} - u_i} N_{i,k-1}(u) + \frac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} N_{i,k-1}(u) \tag{10}$$

In the formula, k represents the degree of the B nd-order spline function; i denotes the sequence number of the B th-order spline function, where 0/0=0.

By utilizing corresponding position-time sequence points in the joint space as type-value points for back-calculating control vertices, and then employing cubic B -spline interpolation for trajectory fitting, the motion trajectory is ensured to pass through the corresponding workspace positions. The functional expression for the i segment curve is:

$$k_i(t) = \frac{1}{6}[t^3 \quad t^2 \quad t \quad 1] \begin{vmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{vmatrix} \begin{bmatrix} P_i \\ P_{i+1} \\ P_{i+2} \\ P_{i+3} \end{bmatrix} \tag{11}$$

2.3 Dynamic Simulation Control of Six-Axis Robotic Arm Based on Computed Torque Method

This paper establishes a computational moment method simulation control system as shown in Figure 3.

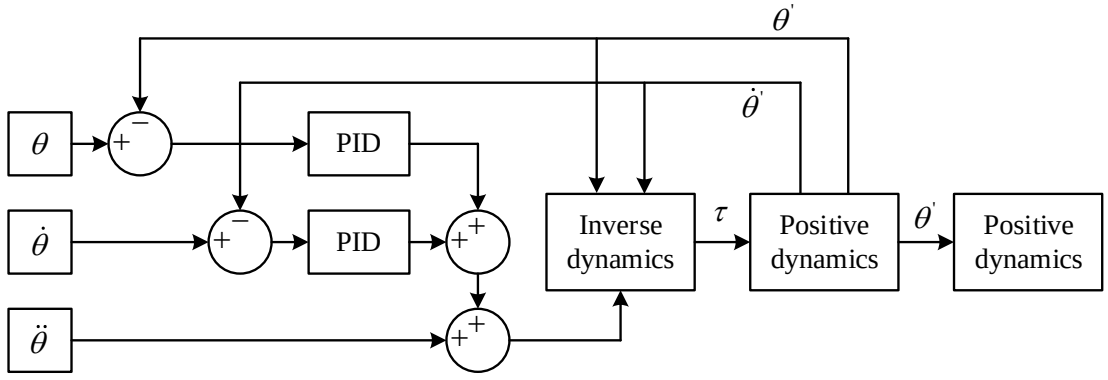


Figure 3: Schematic diagram of the control by the calculated torque method

Figure $\theta, \dot{\theta}, \ddot{\theta}$ shows the desired joint angle, angular velocity, and angular acceleration; Figure τ shows the desired torque; Figure τ' shows the actual torque; Figure $\theta', \dot{\theta}'$ shows the actual joint angle and angular velocity.

The torque-based control method drives the robotic arm by controlling joint torques. It uses the actual angles and angular velocities obtained from the forward dynamics module as

feedback signals. Through a PID controller, it adjusts the inputs to the inverse dynamics module to regulate the magnitude of the output torques.

The inverse dynamics and forward dynamics modules in Figure 3 are modeled using the Lagrange method and the Newton-Euler method, respectively.

2.3.1 Inverse Kinematics Model of a Six-Axis Robotic Arm

The Newton-Euler method constructs a model that enables fast computation and facilitates real-time control. It involves two steps: outward propagation and inward propagation. First, starting from the base and propagating outward toward the end effector, it calculates the angular velocity, linear acceleration, angular acceleration, inertial forces, and moments for each link's center of mass. Subsequently, starting from the end effector and propagating inward toward the base, it calculates the moments for each joint.

The outward propagation of angular velocity can be expressed as:

$$\omega_{i+1}^{i+1} = {}_{i}^{i+1}R\omega_{i}^{i} + \dot{\theta}_{i+1}\hat{z}_{i+1}^{i+1} \tag{12}$$

In the equation: ω_{i+1}^{i+1} is the angular velocity of link $i+1$ in coordinate system $i+1$; ${}_{i}^{i+1}R$ is the rotation matrix between coordinate systems i and $i+1$; ω_{i}^{i} is the angular velocity of link i in coordinate system i; $\dot{\theta}_{i+1}$ is the angular velocity of joint $i+1$; $\hat{z}_{i+1}^{i+1}$ is the Z-axis vector of coordinate system $i+1$ in coordinate system $i+1$. The angular acceleration can be expressed recursively outward as:

$$\dot{\omega}_{i+1}^{i+1} = {}_{i}^{i+1}R\dot{\omega}_{i}^{i} + {}_{i}^{i+1}R\omega_{i}^{i} \times \dot{\theta}_{i+1}\hat{z}_{i+1}^{i+1} + \ddot{\theta}_{i+1}\hat{z}_{i+1}^{i+1} \tag{13}$$

In the equation: $\dot{\omega}_{i+1}^{i+1}$ represents the coordinate system; $i+1$ denotes the angular acceleration of link $i+1$; $\ddot{\theta}_{i+1}$ denotes the angular acceleration of joint $i+1$.

The linear acceleration can be expressed by outward recursion as:

$$\dot{v}_{i+1}^{i+1} = {}_{i}^{i+1}R(\dot{\omega}_{i}^{i} \times p_{i+1}^{i} + \omega_{i}^{i} \times (\omega_{i}^{i} \times p_{i+1}^{i}) + \dot{v}_{i}^{i}) \tag{14}$$

In the equation:

$\dot{v}_{i+1}^{i+1}$ and $\dot{v}_{i}^{i}$ denote the coordinate systems;

$i+1$ and i denote the linear acceleration of the origin;

p_{i+1}^{i} denotes the vector pointing from the origin of coordinate system i to the origin of coordinate system $i+1$.

The acceleration of the center of mass can be expressed as an outward recursive process:

$$\dot{v}_{c_{i+1}}^{i+1} = \dot{\omega}_{i+1}^{i+1} \times p_{c_{i+1}}^{i+1} + \omega_{i+1}^{i+1} \times (\omega_{i+1}^{i+1} \times p_{c_{i+1}}^{i+1}) + \dot{v}_{i+1}^{i+1} \tag{15}$$

In the equation:

$\dot{v}_{c_{in}}^{i+1}$ is the linear acceleration of the center of mass of link $i+1$;

$p_{c_{in}}^{i+1}$ is the vector from the origin of coordinate system $i+1$ pointing to the center of mass of link $i+1$.

The resultant force acting on the center of mass F_{i+1}^{i+1} is:

$$F_{i+1}^{i+1} = m_{i+1}\dot{v}_{c_{i+1}}^{i+1} \tag{16}$$

In the equation: m_{i+1} represents the mass of connecting rod $i+1$.

The resultant torque N_{i+1}^{i+1} acting at the center of mass can be expressed as:

$$N_{i+1}^{i+1} = I_{c_{i+1}}^{i+1}\dot{\omega}_{i+1}^{i+1} + \omega_{i+1}^{i+1} \times I_{c_{i+1}}^{i+1}\omega_{i+1}^{i+1} \tag{17}$$

In the equation: $I_{c_{i,1}}^{i+1}$ represents the moment of inertia matrix of the rod $i+1$ in its center-of-mass coordinate system.

The internal force equilibrium equation can be expressed as:

$$f_i^i = {}_{i+1}^{i}R f_{i+1}^{i+1} + F_i^i \tag{18}$$

In the equation:

f_i^i represents the connecting rod;

$i-1$ denotes the force acting on connecting rod i;

${}_{i+1}^{i}R$ is the rotation matrix from coordinate system $i+1$ to coordinate system i;

$f_i + 1^{i+1}$ represents the force acting on connecting rod $i+1$ from connecting rod i.

The moment equilibrium equation can be expressed as:

$$n_i^i = N_i^i + {}_{i+1}^{i}R n_{i+1}^{i+1} + p_{c_i}^i \times F_i^i + p_{i+1}^i \times {}_{i+1}^{i}R f_{i+1}^{i+1} \tag{19}$$

In the equation:

n_i^i represents the torque exerted by link $i-1$ on link i;

$p_{c_i}^i$ denotes the vector from the origin of coordinate system i pointing toward the center of mass of link i.

Since the motor torque aligns with the axis of coordinate system Z and the remaining external forces are borne by the mechanical components of the robotic arm, the joint torque is:

$$\tau_i = n_i^i z^T \tag{20}$$

In the formula: τ_i represents the joint; i represents the torque; $z = [000]^T$.

2.3.2 Six-Axis Robotic Arm Forward Kinematics Model

Compared to Newton-Euler methods, Lagrange methods build models that clearly describe the dynamic characteristics and motion effects of robotic arms under specific joint torque driving conditions.

The expressions for kinetic energy T and potential energy V of the robotic arm are as follows:

$$T = \frac{1}{2}\dot{q}^T M \dot{q} \tag{21}$$

$$V = \sum_{i=1}^{6} m_i T_{mi}^0(3,4) \tag{22}$$

In the formula:

M represents the mass matrix;

m_i denotes the mass of link i;

T_{mi}^0 is the homogeneous transformation matrix of the center of mass of link i;

$T_{mi}^0(3,4)$ represents the z coordinates of the center of mass of link i.

The mass matrix M can be expressed as:

$$M = \sum_{i=1}^{6} m_i J_{Vi}^T J_{Vi} + J_{\omega i}^T R_{mi}^0 I_i R_{mi}^{0T} J_{\omega i} \tag{23}$$

In the equation: J_{Vi} represents the Jacobian matrix of the linear velocity of the center of mass of link i; $J_{\omega i}$ represents the Jacobian matrix of the angular velocity of the center of mass of link i; $R_{mi}^0 = T_{mi}^0(1:3,1:3)$. Substituting T and V into the Lagrange equations yields:

$$M\ddot{q} + \dot{M}\dot{q} - \frac{1}{2}\left[\frac{\partial M}{\partial q}\dot{q}\right]^T \dot{q} + \frac{\partial V}{\partial q} = Q \tag{24}$$

In the equation: Q represents the joint torque. Equation (24) can be rearranged and simplified as:

$$\ddot{q} = M^{-1}(Q - C - G) \tag{25}$$

In the equation: $\ddot{q}$ represents the joint angular acceleration; C denotes the velocity vector in Cartesian space; G signifies the gravity vector in Cartesian space. Their respective expressions are:

$$\begin{cases} C = \dot{M}\dot{q} - \frac{1}{2}\left[\frac{\partial M}{\partial q}\dot{q}\right]^T \dot{q} \\ G = \frac{\partial V}{\partial q} \end{cases} \tag{26}$$

Based on Equation (26), a forward dynamics module for the robotic arm is established. The angular accelerations at each joint are calculated from the driving torque, and then integrated to obtain the actual angles and angular velocities. These values are fed back to the PID controller and the inverse dynamics module to achieve closed-loop control.

3 Automated Calibration Technology for Manual Workstations in Rotary Machinery

3.1 Principle of the Improved Nine-Point Calibration Method

3.1.1 Nine-Point Calibration Method

In this research, an eye-on-hand approach is used to perform hand-eye calibration on a robotic manipulator using the nine-point calibration technique. The nine-point calibration technique helps achieve a common transformation between the coordinate systems of the camera, robotic arm, and world. To accomplish this goal, it requires moving the robotic arm's end-effector to nine predetermined points to calculate its coordinates relative to the robotic arm coordinate system. Simultaneously, the camera identifies corresponding target points to derive nine sets of pixel coordinates.

The calibration equation for the transformation between the robotic arm coordinate system and the camera coordinate system is:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = R \begin{pmatrix} x \\ y \end{pmatrix} + M \tag{27}$$

In the formula:

(x', y') represents the robotic arm coordinate;

(x, y) represents the pixel coordinate;

R represents the deflection angle;

M represents the displacement.

The homogeneous transformation expression is:

$$\begin{cases} x' = ax + by + c \\ y' = a'x + b'y + c' \end{cases} \tag{28}$$

Assuming the coordinates of the three points are $(x_0, y_0), (x_1, y_1), (x_2, y_2)$, substituting into equation (29) yields:

$$\begin{cases} x_0' = ax_0 + by_0 + c \\ x_1' = ax_1 + by_1 + c \\ x_2' = ax_2 + by_2 + c \end{cases} \tag{29}$$

$$\begin{cases} y_0' = a'x_0 + b'y_0 + c' \\ y_2' = a'x_1 + b'y_1 + c' \\ y_2' = a'x_2 + b'y_2 + c' \end{cases} \tag{30}$$

Substituting equations (29) and (30) into equation (28) yields:

$$\begin{bmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} x_0^{'} \\ x_1^{'} \\ x_2^{'} \end{bmatrix} \tag{31}$$

$$\begin{bmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{bmatrix} \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix} = \begin{bmatrix} y_0' \\ y_1' \\ y_2' \end{bmatrix} \tag{32}$$

In the equation: a, b represents the coordinate in the x direction when mapping the x, y nd coordinate of the camera coordinate system to the robotic arm coordinate system; $a^{'}, b^{'}$ represents the coordinate in the $y^{'}$ th direction when mapping the x, y th coordinate of the camera coordinate system to the robotic arm coordinate system; c, c' represents the displacement (translation) in the x, y th direction.

Solve for R and M in the calibration equation to derive the transformation relationship between robot arm coordinates and camera coordinates. This also reveals that the primary sources of error in the nine-point calibration method stem from end-effector pose errors of the robot arm and camera calibration inaccuracies. In industrial production, robotic arms may swap different grippers based on varying tasks. During assembly, errors may occur, leading to imprecise measurements of the robotic arm's sixth axis length. This inaccuracy affects the calculation of the end-effector's pose, ultimately introducing errors in the nine-point calibration process.

3.1.2 Improved Nine-Point Calibration Method

The errors in the nine-point calibration method stem from end-effector pose errors and camera calibration errors. To address this issue, an improved nine-point calibration method is proposed. A calibration rod is placed within the robot arm's workspace, with its apex designated as fixed point p. The robot arm is moved to align its end-effector with the tip of the calibration rod, recording the current pose of the robot arm. The robot arm is then moved again, capturing four distinct poses of its end-effector as it approaches this point. By approaching the calibration rod with the end effector in different poses, the joint angles $[\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]$ of each joint when the robotic arm reaches the calibration rod and the coordinates (x, y, z) of the end effector are obtained. These values are then substituted into Equation (33) to solve the inverse kinematics of the robotic arm and determine the length of the sixth axis.

Multiplying both sides of equation (33) by the inverse matrix of ${}_1^0T$ on the left and by the inverse matrix of ${}_6^5T$ on the right yields:

$$[{}_1^0T]^{-1}{}_6^0T[{}_6^5T]^{-1} = {}_2^1T\,{}_3^2T\,{}_4^3T\,{}_5^4T \tag{33}$$

To make the elements at positions (2, 4) on both sides equal, we have:

$$p_y c_1 - d_6(a_y c_1 + a_x s_1) - p_x s_1 = d_3 \tag{34}$$

In the formula:

p_x, p_y, a_x, a_y represents the parameter variable in the end-effector pose matrix of the

robotic arm;

$c_1 = \cos\theta_1, s_1 = \sin\theta_1$ and d_6 denote the length of the sixth axis after installing the robotic arm end-effector.

Given the joint angles and axis lengths of the robotic arm, it can be determined that:

$$d_6 = \frac{p_y c_1 - p_x s_1 - d_3}{a_y c_1 + a_x s_1} \tag{35}$$

After obtaining d_6, substituting into equation (35) allows for the calculation of more precise coordinates for each point in the nine-point calibration method, thereby enhancing its accuracy.

3.2 Pose Estimation Based on Point Cloud Template Matching

In order to solve the issues caused by the incorrect grasp and reduce the production efficiency that stems from the incapability to estimate the workpiece's pose information in the robotic arm grasping process, the following three-dimensional point cloud template matching solution is adopted. The process of workpiece pose estimation is shown in Figure 4. The point cloud data for the scene is acquired with the help of Intel D455i depth camera. The statistical and bilateral filtering processes are employed to eliminate noise data and retain the shape and edges of the workpiece at the same time. Next, RANSAC algorithm is implemented to separate the scene point cloud and the workpiece point cloud data. The workpiece point cloud and the template point cloud are registered roughly through the use of SAC-IA method, in which the template point cloud is created with the help of SolidWorks.

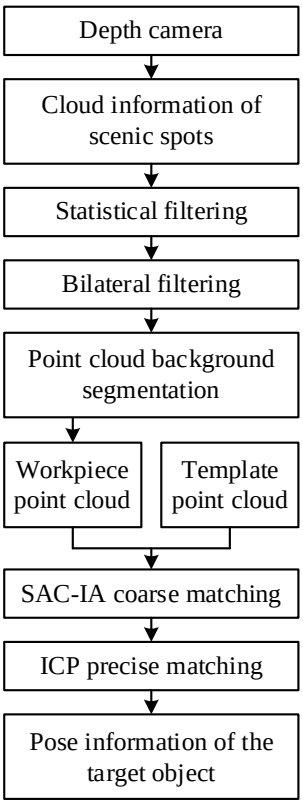


Figure 4: Workpiece pose estimation

3.3 Analysis of Automated Labeling Effectiveness

3.3.1 Feature Point Matching Performance Verification

Point cloud template matching pose estimation pose estimation algorithm has been successfully used to identify targets under circumstances of rotation, tilting, occlusion, background cluttering, excessive lighting, and inadequate lighting, but still retains its fast computational speed. For a clearer understanding of the characteristics of the pose estimation algorithm, we conducted feature matching experiments with a set of images with the same resolution by applying not only the point cloud template matching pose estimation algorithm but also an enhanced version of SIFT algorithm through RANSAC. Feature matching rates and computing times of the two algorithms under the above conditions are shown in Table 2. It can be seen that our proposed algorithm is much more efficient than SIFT, where both complete their tasks within 50 ms. The reason for this high efficiency is the fact that we did a lot of offline training work before recognition. In addition, our algorithm can achieve fast recognition of feature points and matching work. However, due to its high efficiency in identifying features, SIFT recognizes too many feature points. After removing mismatched points using the RANSAC algorithm, its matching rate remains low. In contrast, the feature points used by this algorithm are stable points selected during the training phase, resulting in fewer points and a higher matching rate.

Table 2: Feature point matching rate and operation time

Sample	Our algorithm				SIFT algorithm			
	Template feature points	Matching feature points	Matching rate/%	Match time/ ms	Template feature points	Matching feature points	Matching rate/%	Match time/ ms
Rotations	355	288	81.13	22	1525	695	45.57	888
Slant	355	220	61.97	22	1525	438	28.72	721
Occlusion	355	199	56.06	20	1525	465	30.49	856
Overshine	355	285	80.28	45	1525	725	47.54	1221
Overdark	355	256	72.11	23	1525	705	46.23	785
Clutter	355	171	48.17	18	1525	35	2.30	565

3.3.2 Target Positioning Verification

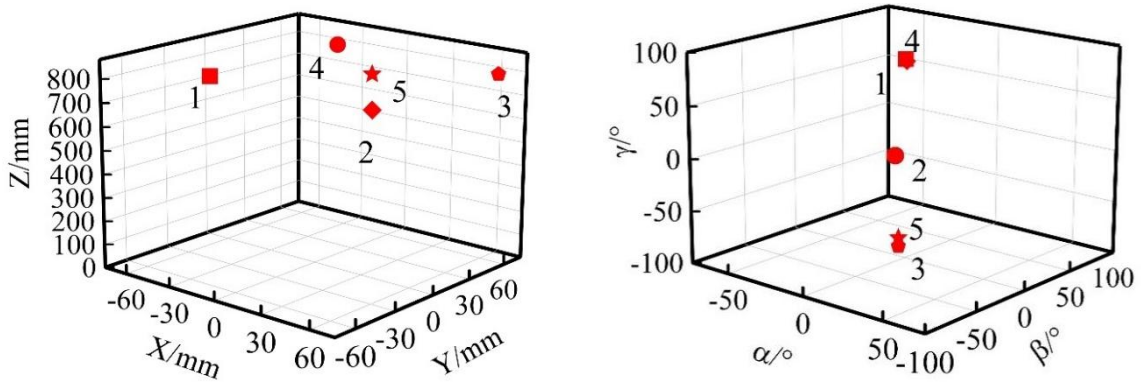
Based on the principles of the nine-point calibration method, after completing target recognition, positioning experiments were conducted on the objects. A world coordinate system for the object was established with the center of its surface as the origin. Feature point coordinates were substituted into the camera-image-object coordinate relationship to solve for the object's position and orientation. Positioning was performed on flat objects, stacked objects, and randomly arranged objects, respectively, to analyze their relative positioning accuracy. The accuracy of visual localization depends on the precision of the target object's pose data within the camera coordinate system. Since it is difficult to directly measure the positioning error of an object's pose in the camera coordinate system, the following localization experiments indirectly measure visual positioning errors using the relative poses between objects and analyze these errors.

For flat object localization, objects 1–5 were placed at the corners of a predefined square, spaced approximately 100.0 mm apart. Table 3 shows the pose of the flat objects in the camera coordinate system. Based on the visual positioning data in Table 3 and the distances between objects, the in-plane relative positioning error does not exceed 2.50 mm, indicating accurate

positioning. It is noticeable that there are significant differences in the Z-coordinates of objects 15 and 13, mainly because the camera is not installed at an angle of 90 degrees to the imaging plane. Figure 5 illustrates a 3D coordinate diagram created based on the camera coordinate system.

Table 3: The position of the flat object in the camera coordinate system

Number	X /mm	Y /mm	Z /mm	α /°	β /°	γ /°
1	-35.2	-38.8	808.9	-0.2	0.75	90.5
2	66.9	-39.2	808.3	0.9	0.32	88.9
3	-35.5	60.5	804.8	-3.9	-3.85	-0.82
4	68.2	60.1	803.5	-4.4	0.25	-90.75
5	20.0	20.0	801.7	-4.5	0.44	-82.48



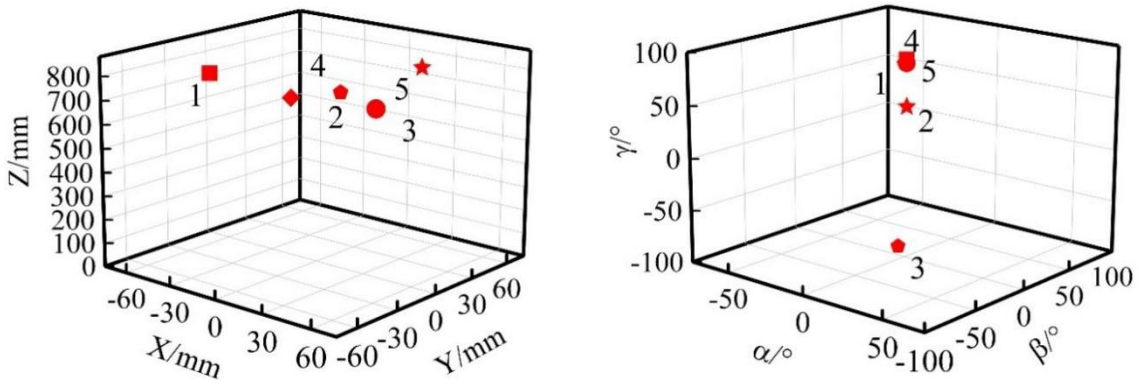
(a) The position of the flat object (b) The posture of the flat object

Figure 5: Three-dimensional display of flat object position

In the positioning of stacked objects, Object 5 is at the bottom layer, Objects 1 and 3 are at the relatively lower layer on the same plane, Object 2 is in the middle, and Object 4 is at the top layer. Table 4 shows the pose of stacked objects in the camera coordinate system. The Z-axis data clearly reveals differences in depth distances: the relative depth positioning errors between Objects 12, 23, 34, and 14–35 are 0.60, 0.33, 0.1, 0.33, and 0.60 mm, respectively, indicating high positioning accuracy. 14, and 35 exhibit relative depth positioning errors of 0.60, 0.33, 0.1, 0.33, and 0.60 mm, respectively, indicating high positioning accuracy. The poses of objects 1–5 are also illustrated in the specific measurement data and three-dimensional coordinate diagram. Figure 6 presents the three-dimensional display of the stacked objects' poses.

Table 4: The position of the stack object in the camera coordinate system

Number	X /mm	Y /mm	Z /mm	α /°	β /°	γ /°
1	-36.9	-37.6	807.7	0.3	1.3	90.2
2	15.1	-37	774.2	1.8	-2.4	89.2
3	68.6	-40	807.5	1	0.2	87.1
4	15.5	-0.5	740.2	-3.9	-0.4	-91.1
5	35.78	38.25	811.8	0.8	0.8	45.8



(a) The position of the stack object (b) The posture of the stack object

Figure 6: Three-dimensional display of the stack object position

4 Experimental Verification and Analysis

4.1 Robotic Arm Path Planning Simulation Test

4.1.1 Two-Dimensional Simulation Experiments

To validate the effectiveness and superiority of the improved RRT algorithm, simulation comparison experiments were first conducted in a two-dimensional space. The experiments compared RRT, RRT*, RRT-Connect, and the improved RRT algorithm proposed in this paper. The four algorithms shared identical basic parameters, with a simulation map size of 1000×1000. The path starting point was [50, 50], with the endpoint at [800, 800]. The initial step size Sinitial was set to 50, with 1000 iterations. The distance gain factor and environment gain factor were set to 1 and 3, respectively. Each algorithm ran 25 times. Due to the random nature of RRT-based algorithms, the average results were used for comparison. The comparison between RRT algorithm and other algorithms under two dimensional space is shown in Fig. 7. Experimental outcomes can be seen in Table 5. As opposed to the previous three methods, the new enhanced RRT algorithm produces significantly fewer nodes than before. The reason behind this improvement can be attributed to the use of target biased sampling as well as the better near point selection method, both of which improve the goal-oriented property of the algorithm. Moreover, the ability to adopt a variable step method enables efficient search of random trees in areas of densely populated obstacles. As mentioned in the table, compared with RRT, RRT* and RRT-Connect methods, the enhanced RRT algorithm produces 27.8%, 10.5%, and 22.3% lower path cost; at the same time, run time of 59.1%, 53.7%, and 41.5% reduction and number of nodes of 80.5%, 90.4%, and 61.3% are achieved, respectively.

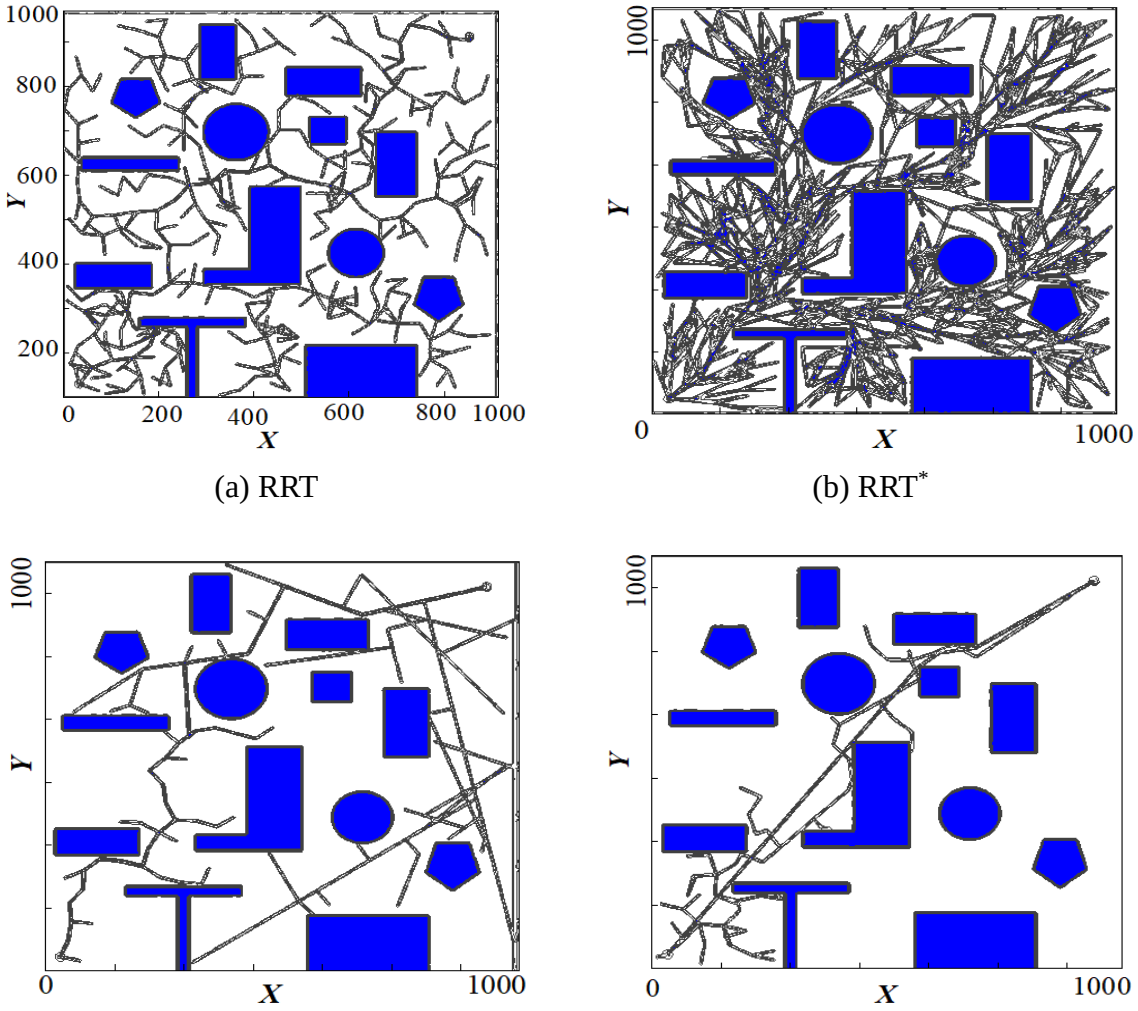


Figure 7: The two-dimensional space RRT algorithm is compared

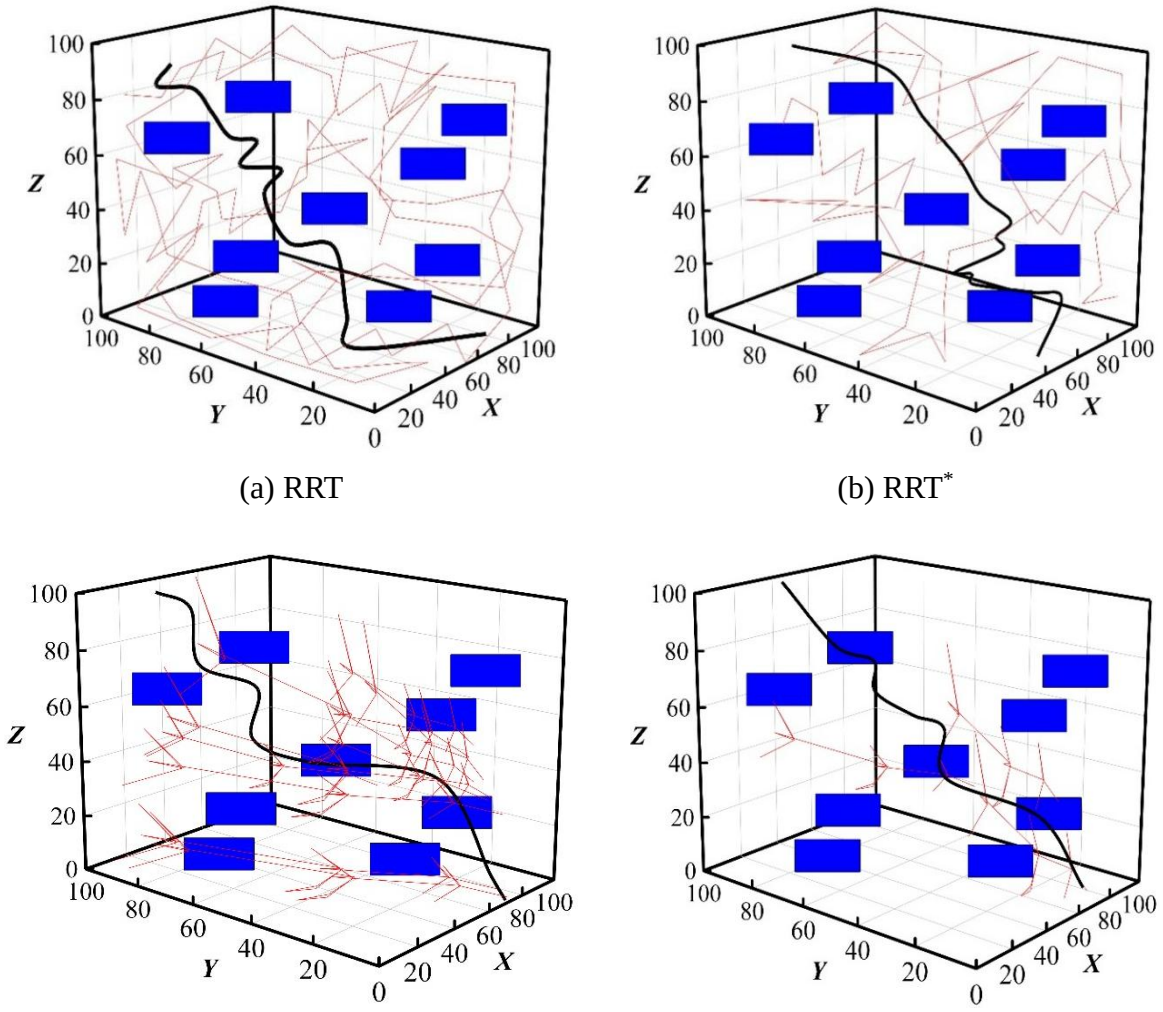
Table 5: Two-dimensional simulation data

Algorithm	Path cost/mm	Running time/s	Node number
RRT	1452.3	0.656	308
RRT*	1171.1	0.579	625
RRT-Connect	1355.5	0.458	155
Our	1048.2	0.268	60

4.1.2 Three-Dimensional Simulation Experiments

For testing the performance of the enhanced algorithm in three-dimensional environment, the following Figure 8 shows the path planning results of the four algorithms in three-dimensional space. From the figure, the obstacle is shown as a hexagonal prism, the sparse random tree as thin lines and the planned trajectory as thick lines. According to the results, the RRT algorithm generates sampling points with high density where the generation of many random numbers makes the final generated trajectory very tortuous with low efficiency. On the other hand, the RRT* algorithm reduces the amount of sampling points and simplifies the expansion process of the random tree; hence, there is improvement in convergence efficiency, but the resultant path still is relatively tortuous. The RRT-Connect algorithm performs better than RRT when

considering the rate of convergence efficiency. Compared to RRT*, its random tree expansion has a high directional trend, and thus it leads to a relatively straight path and lower path cost. As seen from the graph, the enhanced algorithm has the lowest number of sampling points compared to the other algorithms. Moreover, the random tree expansion process of the enhanced algorithm is simplified, and the path cost attained is similar to that of the RRT-Connect algorithm. This suggests that the optimized algorithm has the highest convergence efficiency without affecting the path cost. Table 6 gives the average time consumption of the four algorithms for 50 trials. The table indicates that in similar complex environments, the enhanced algorithm takes less time with the best efficiency with the lowest values of path cost, time and nodes being 1701.7 mm, 7.8 s and 366, respectively.



(a) RRT (b) RRT*

(c) RRT-Connect (d) Our

Figure 8: The three-dimensional space RRT algorithm is compared

Table 6: The data comparison of four algorithm path planning in 3D scenario

Algorithm	Path cost/mm	Running time/s	Node number
RRT	1945.5	71.5	2255
RRT*	1777.7	28.3	833
RRT-Connect	1710.8	20.5	750
Our	1701.7	7.8	366

4.2 Motion Control Analysis

As part of robotic arm motion control, the joint_states node collects data concerning the joint angle, velocities, and other factors involved in the operation. It is possible to obtain these data from the node and analyze them using MATLAB in order to find out about the relationship between the joints. In Figure 9, one can see a plot depicting the theoretical joint angle of Joint 1 compared to the practical joint angles that were recorded for two different controllers. As can be seen, both controllers managed to move the robotic arm through the required path; however, there were some errors that occurred during control.

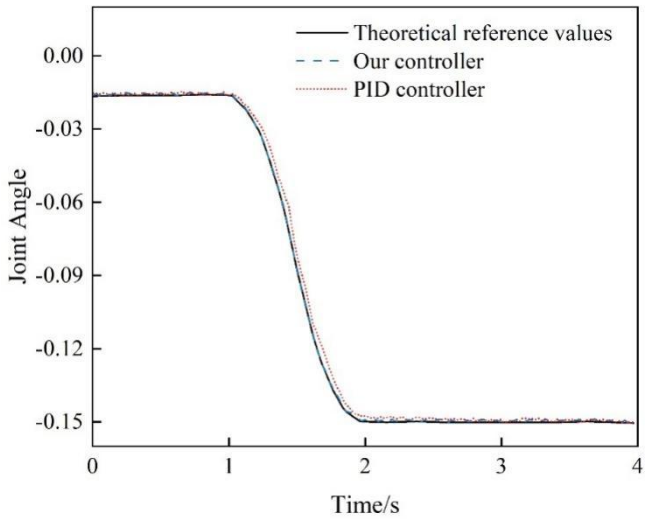


Figure 9: Angle of joint 1 in three cases

Figure 10 shows the joint angle error of Joint 1 under two control schemes. The figure indicates that the trajectory tracking error of the computational torque controller is significantly reduced compared to the PID controller. Under CTC control, the maximum joint angle error of Joint 1 is $e_1 = 1.83\mathrm{E}-3 rad$ °, while under PID control, it reaches $e_2 = 3.56\mathrm{E}-3 rad$ °. Calculations reveal that the maximum joint angle error under CTC control is reduced by 48.60%.

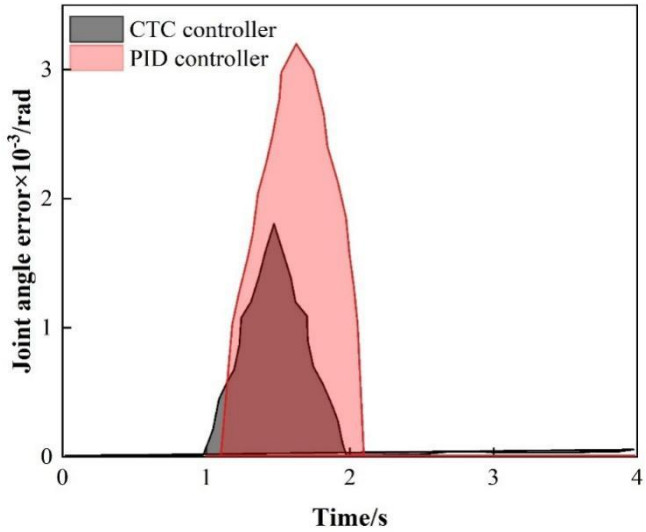
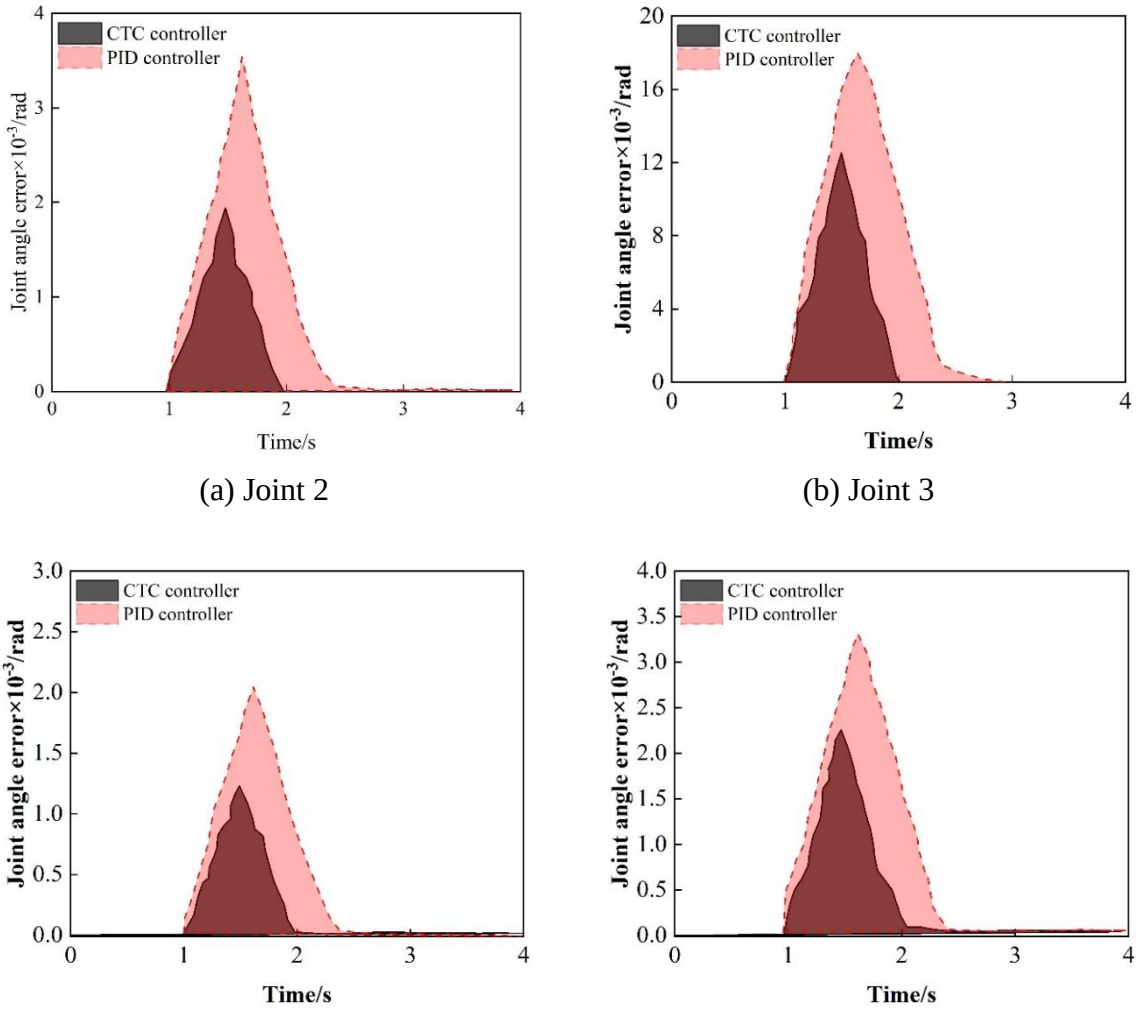


Figure 10: Joint angle errors of joint 1 under two control schemes

Similarly, the joint angular errors for joints 2 to 5 under the two control schemes are

analyzed, with results shown in Figure 11. It can be seen that when the robotic arm moves along the pre-planned trajectory, the CTC control achieves smaller errors compared to the PID controller. As the robotic arm approaches the target position, the joint angular errors are summarized in Table 7. It is evident that under CTC control, the angular errors of all joints exhibit a significant reduction compared to PID control. Among these, Joint 2 demonstrates the most substantial reduction in angular error, decreasing by 89.16%, while Joint 4 shows the smallest reduction, yet still achieving a decrease of 25.94%.



(a) Joint 2 (b) Joint 3

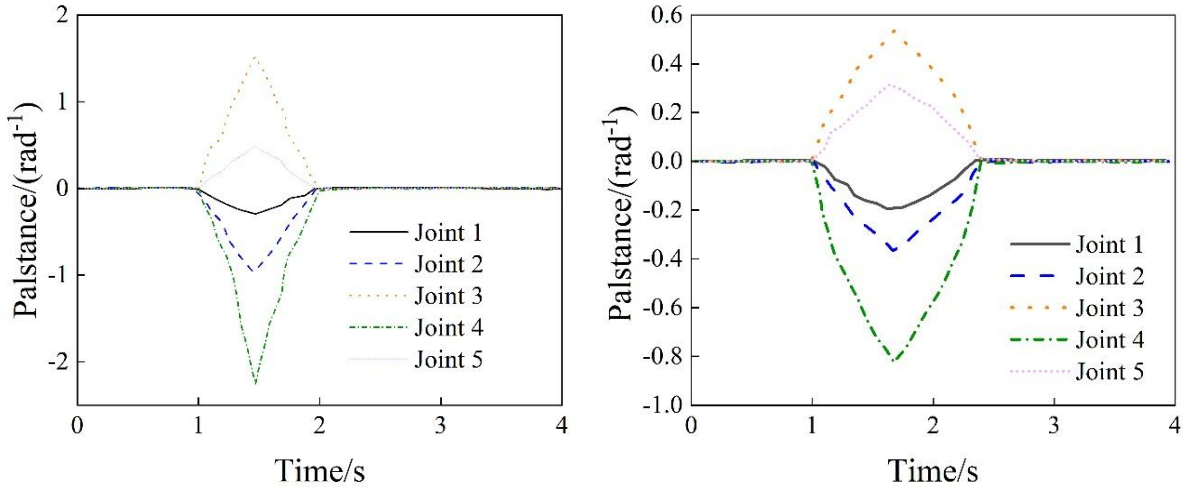
(c) Joint 4 (d) Joint 5

Figure 11: Joint angle errors of each joint under two schemes

Table 7: Manipulator joint errors at target position

Joint serial *i*	Joint error×10^{-5}/rad		Error reduction/%
	Our controller	PID controller	
1	4.412	6.356	44.06
2	2.445	4.625	89.16
3	2.824	4.787	69.51
4	5.012	6.312	25.94
5	5.995	8.879	48.11
6	0.000	0.000	0.00

The angular velocity of the robotic arm under two control schemes is shown in Figure 12. The implementation of the nonlinear computational torque controller demonstrates compatibility with more advanced controllers and provides a valuable reference for developing other sophisticated control systems.



(a) CTC controller (b) PID controller

Figure 12: Joint velocities of manipulator under two control schemes

4.3 Verification of Multi-Station Operation Methods for Robotic Arms

4.3.1 Verification of Multi-Station Operation Method for Robotic Arms

The global camera and the local camera use the same domestically made Daheng Mercury series camera, model MER-502-79U3M, with a resolution of 2048 × 2048 pixels. The global camera has a 5 mm lens, while the local camera has a 9 mm lens. The lighting system consists of a forward lighting source and strobe light source, where the frequency of projection matches the frequency of the camera's acquisition. There are three workstations set up around the robotic arm; each workstation contains a target aligned at different distances and angles, generating the greatest angular difference of 90°. The robotic arm has a working radius of about 1.1 m and can rotate a maximum of 145°. The distance between the stereo vision system and the alignment holes is initially between 1 m to 1.7 m. Measurement using calipers shows that the diameter of the alignment shaft is 9.95 mm, while the diameters of the alignment holes 1, 2, and 3 are 10.01 mm, 10.18 mm, and 10.95 mm, respectively.

The deviation between the current pose ${}_{M}^{P}T_{i}$ and the desired pose ${}_{M}^{P}T_{e}$ throughout the entire alignment process is quantified as ΔT. The pose deviation results are shown in Table 8. Specifically, $\Delta\alpha$, $\Delta\beta$, and $\Delta\gamma$ represent angular deviations around the X, Y, and Z axes, respectively, while ΔX, ΔY, and ΔZ denote positional deviations along the X, Y, and Z axes, respectively. Step 0 in Table 7 represents the starting posture error. Step 1 refers to the result of positioning based on the global camera, in which case the error becomes greatly diminished but fails to meet the required specifications for alignment. Based on the local camera's guidance, three fine-tuning steps are carried out by the robot arm, whereupon the angular and linear errors become smaller gradually. Finally, the axis of the hole becomes aligned, with an angular error of 0.06° and a linear error of 0.19 mm.

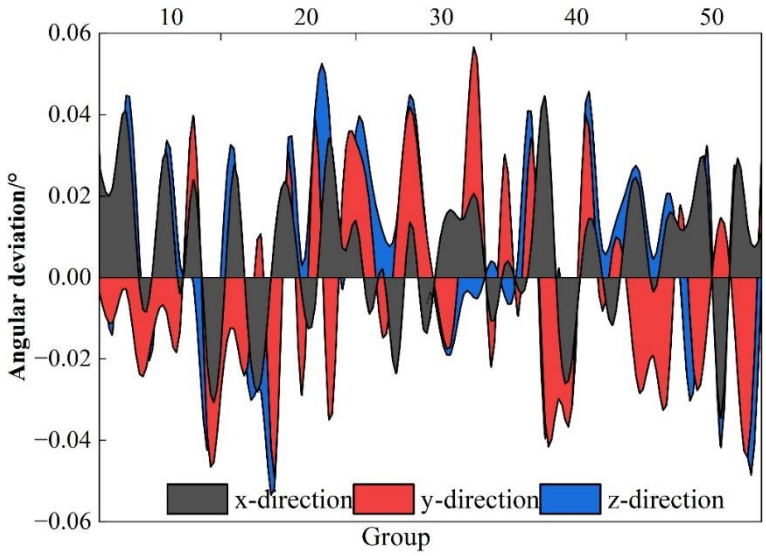
Table 8: Pose deviation of alignment process of hole 1

Step1	$\Delta\alpha$ /°	$\Delta\beta$ /°	$\Delta\gamma$ /°	ΔX /mm	ΔY /mm	ΔZ /mm
0	-23.85	4.43	30.88	-211.52	-210.28	1235.89
1	-1.44	0.25	1.55	-21.22	-19.88	55.55
2	-0.15	0.08	0.09	-1.58	-1.15	2.11
3	0.08	-0.06	0.01	0.22	0.33	0.08
4	-0.05	-0.05	0.01	0.22	-0.08	-0.03

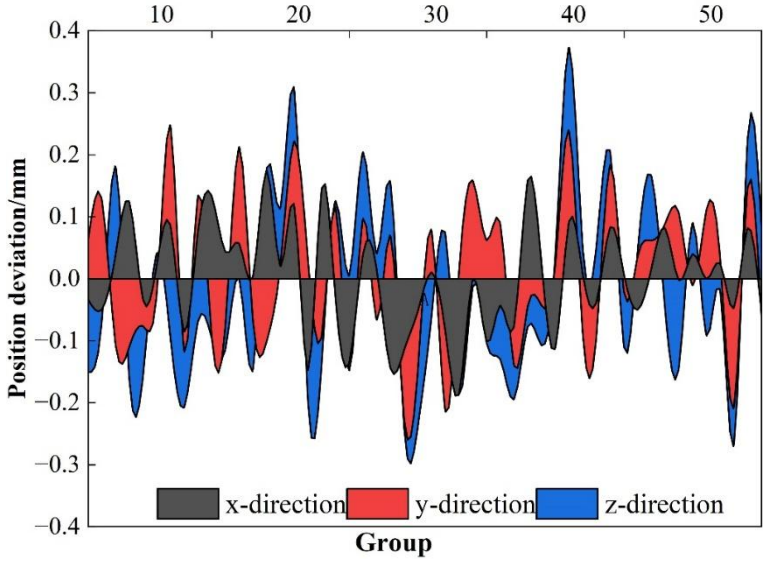
Alignment of holes 2 and 3 was done using the same technique used for aligning hole 1, leading to accurate results. For instance, hole 2 shows angular displacement at 0.06° and positional displacement at 0.11 mm, whereas hole 3 shows angular displacement at 0.03° and positional displacement at 0.11 mm. In this experiment, it should be mentioned that due to some experimental limitations, three workstations were used to validate the procedure of validation; however, this system has the capability of aligning and grasping items at other stations as well.

4.3.2 Repeatability Alignment Accuracy Verification

To validate the repeatability of the alignment accuracy for the proposed method, 50 alignment experiments were conducted on target hole 1. The pose deviations during hole axis alignment were statistically analyzed, including angular deviations around the X, Y, and Z axes and positional deviations along the X, Y, and Z axes. The results are shown in Figure 13. The figure indicates that positional deviations along the X, Y, and Z axes are comparable, while angular deviations around the Z-axis are significantly smaller than those around the X and Y axes. This is attributed to the point cloud template matching algorithm employed for pose estimation, which demonstrates higher accuracy in estimating in-plane rotations (rotations around the Z-axis). Additionally, Table 9 presents the mean, maximum value, and standard deviation of the absolute deviation quantities. The mean angular deviation is less than 0.05°, and the mean positional deviation does not exceed 0.075 mm, indicating that the alignment strategy proposed in this paper exhibits good stability.



(a) Angle deviations of alignment results



(b) Position deviations of alignment results

Figure 13: Deviation of 50 alignment experiments

Table 9: Results of 50 alignment experiments

Parameter	$\Delta\alpha$ /°	$\Delta\beta$ /°	$\Delta\gamma$ /°	ΔX /mm	ΔY /mm	ΔZ /mm
Mean	0.03	0.03	0.005	0.055	0.055	0.05
Maximum	0.05	0.07	0.02	0.21	0.18	0.15
Standard deviation	0.012	0.012	0.005	0.055	0.048	0.042

5 Conclusion

This paper developed a motion control and calibration system for rotary manipulators. An improved D-H parameter approach was employed to establish its forward and inverse kinematic models. Trajectory planning was achieved using an enhanced RRT algorithm. A computational torque method control module for the robotic arm was constructed, and an improved nine-point calibration method for automated workstation calibration was proposed. Experimental results from different modules demonstrated that:

(1) In a two-dimensional environment, compared to the RRT, RRT*, and RRT-Connect algorithms, the proposed method reduces path cost by 27.8%, 10.5%, and 22.3%, respectively, while decreasing runtime by 59.1%, 53.7%, and 41.5%, and reducing the number of nodes by 80.5%, 90.4%, and 61.3%, respectively. In a three-dimensional environment of equivalent complexity, the proposed method achieves a shortest path cost of 1701.7 mm, a runtime of 7.8 seconds, and utilizes 366 nodes. This capability to obtain optimal paths while saving time demonstrates the superiority and practicality of the improved RRT algorithm presented herein.

(2) With CTC, the angular error for each joint of the robotic arm was greatly reduced compared to PID. For Joint 2 and Joint 4, the reduction of angular error was greatest at 89.16% and least at 25.94%, respectively. The result suggests that the torque controller provides an excellent level of control performance, as well as small trajectory tracking errors, leading to higher performance levels.

(3) The applicability of the method is confirmed using the hole-shaft alignment experiment. Based on the results obtained after the three stages of alignment, the angular error was less than 0.06°, while the positional error was less than 0.20 mm. Future works based on this study will

be focused on two aspects, namely, the improvement of global camera searching for feature points and the target tracking algorithm.

Funding

This work was supported by Key Scientific Research Project in 2024 of Xi'an Traffic Engineering University (Project No.: 2024KY-14): AI-Enabled Torque-Sensing Multi-Station Automated Calibration Techniques for Rotary Manipulators.

References

[1] Suwarno, I., Cakan, A., Raharja, N. M., Baballe, M. A., & Mahmoud, M. S. (2023). Current trend in control of artificial intelligence for health robotic manipulator. Journal of Soft Computing Exploration, 4(1), 1-12.

[2] Mahadevkar, S. V., Khemani, B., Patil, S., Kotecha, K., Vora, D. R., Abraham, A., & Gabralla, L. A. (2022). A review on machine learning styles in computer vision—techniques and future directions. Ieee Access, 10, 107293-107329.

[3] Zhao, X., Wang, L., Zhang, Y., Han, X., Deveci, M., & Parmar, M. (2024). A review of convolutional neural networks in computer vision. Artificial Intelligence Review, 57(4), 99.

[4] Zhou, L., Zhang, L., & Konz, N. (2022). Computer vision techniques in manufacturing. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 53(1), 105-117.

[5] Li, Z., Li, S., & Luo, X. (2021). An overview of calibration technology of industrial robots. IEEE/CAA Journal of Automatica Sinica, 8(1), 23-36.

[6] Enebuse, I., Foo, M., Ibrahim, B. S. K. K., Ahmed, H., Supmak, F., & Eyobu, O. S. (2021). A comparative review of hand-eye calibration techniques for vision guided robots. IEEE Access, 9, 113143-113155.

[7] Ibaraki, S., Theissen, N. A., Archenti, A., & Alam, M. M. (2021). Evaluation of kinematic and compliance calibration of serial articulated industrial manipulators. International Journal of Automation Technology, 15(5), 567-580.

[8] Ruiz-Ruiz, F. J., Gandarias, J. M., Pastor, F., & Gomez-De-Gabriel, J. M. (2021). Upper-limb kinematic parameter estimation and localization using a compliant robotic manipulator. IEEE Access, 9, 48313-48324.

[9] Podder, I., Fischl, T., & Bub, U. (2024). Smart calibration and monitoring: leveraging artificial intelligence to improve MEMS-based inertial sensor calibration. Complex & Intelligent Systems, 10(6), 7451-7474.

[10] Kong, Y., Yang, L., Chen, C., Zhu, X., Li, D., Guan, Q., & Du, G. (2024). Online kinematic calibration of robot manipulator based on neural network. Measurement, 238, 115281.

[11] Yang, W., Li, S., Li, Z., & Luo, X. (2023). Highly accurate manipulator calibration via

extended Kalman filter-incorporated residual neural network. IEEE Transactions on Industrial Informatics, 19(11), 10831-10841.

[12] Sayour, M. H., Kozhaya, S. E., & Saab, S. S. (2022). Autonomous robotic manipulation: real-time, deep-learning approach for grasping of unknown objects. Journal of Robotics, 2022(1), 2585656.

[13] Nussibaliyeva, A., Sergazin, G., Tursunbayeva, G., Uzbekbayev, A., Zhetenbayev, N., Nurgizat, Y., ... & Yussupova, S. (2024). Development of an artificial vision for a parallel manipulator using machine-to-machine technologies. Sensors, 24(12), 3792.

[14] Li, Z., Li, S., & Luo, X. (2021, December). Data-driven industrial robot arm calibration: a machine learning perspective. In 2021 IEEE International Conference on Networking, Sensing and Control (ICNSC) (Vol. 1, pp. 1-6). IEEE.

[15] Li, Z., Li, S., Bamasag, O. O., Alhothali, A., & Luo, X. (2022). Diversified regularization enhanced training for effective manipulator calibration. IEEE Transactions on Neural Networks and Learning Systems, 34(11), 8778-8790.

[16] Fan, Q., Gong, Z., Zhang, S., Tao, B., Yin, Z., & Ding, H. (2021). A vision-based fast base frame calibration method for coordinated mobile manipulators. Robotics and Computer-Integrated Manufacturing, 68, 102078.

[17] Cao, H. Q., Nguyen, H. X., Tran, T. N. C., Tran, H. N., & Jeon, J. W. (2021). A robot calibration method using a neural network based on a butterfly and flower pollination algorithm. IEEE Transactions on Industrial Electronics, 69(4), 3865-3875.

[18] Liu, Z., Liu, X., Cao, Z., Gong, X., Tan, M., & Yu, J. (2022). High precision calibration for three-dimensional vision-guided robot system. IEEE Transactions on Industrial Electronics, 70(1), 624-634.

[19] Balanji, H. M., Turgut, A. E., & Tunc, L. T. (2022). A novel vision-based calibration framework for industrial robotic manipulators. Robotics and Computer-Integrated Manufacturing, 73, 102248.

[20] Peiqi He, Yaqin Zhao, Min Tan & Bingquan Liao. (2025). Path optimization model of manipulator based on d-h parameter method and genetic algorithm. Journal of Physics: Conference Series,2964(1),012018-012018.

[21] Li Yucai & Chen Naijian. (2025). Improved RRT algorithm for robotic arm path planning based on reward strategy. Proceedings of the Institution of Mechanical Engineers,239(17),6983-6995.