IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 17 May 2022, accepted 12 June 2022, date of publication 20 June 2022, date of current version 29 June 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3184451

A Novel Solution to the Inverse Kinematics Problem of General 7R Robots

SHUXIN XIE[1], LINING SUN[2], GUODONG CHEN[2], ZHENHUA WANG[2], AND ZIXIANG WANG[2]

[1]Robotics and Microsystems Center, School of Mechanical and Electrical Engineering, Soochow University, Suzhou 215123, China
[2]Jiangsu Provincial Key Laboratory of Advanced Robotics, School of Mechanical and Electrical Engineering, Soochow University, Suzhou 215123, China

Corresponding authors: Guodong Chen (guodongxyz@163.com) and Lining Sun (lnsun@suda.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB1310201.

ABSTRACT Task and Motion Planning for a 7-DOF serial robot containing only revolute joints needs a fast and numerically stable inverse kinematics solution. The analytical methods are real-time, but they are robot dependent. Although the numerical methods based on inverse differential kinematics can solve the inverse kinematics problem (IKP) of 7R robots with arbitrary geometric parameters, their numerical stability is insufficient. Moreover, it is time-consuming for them to converge to a single solution that depends on the initial guess. The main innovation of this article is the development of a practical method for solving the IKP of general 7R robots. The proposed method can find multiple solutions and is also faster than the numerical method. It combines symbolic preprocessing, Sylvester dialytic elimination, and eigendecomposition. First, according to the Denavit-Hartenberg convention of geometric representation, a general kinematics model of a 7R manipulator is established. Secondly, the joint variables are separated by symbolic preprocessing, and then two symbolic matrices are obtained. After numerical substitution, a square matrix of polynomial coefficients is computed through Gaussian and Sylvester dialytic elimination. Finally, utilizing the eigendecomposition of the matrix restructured from the square matrix, the values of some variables are obtained, and then the remaining variables are solved by back-substitution. The simulation and experimental data are analyzed. The comparison results verify that the proposed method applies to solving the IKP of general 7R robots and is almost real-time, effective, and numerically stable.

INDEX TERMS Dialytic elimination, inverse kinematics, robot kinematics, serial manipulators, symbolic computation.

I. INTRODUCTION

As far as one pose in the three-dimensional workspace for the end-effector of a 7R robot is concerned, an infinite number of inverse kinematics (IK) solutions can make the end-effector reach it. This feature affords a robot with avoiding workspace obstacles [1], refraining from singular configurations [2], overcoming joint limits [3], and improving dexterity [4]. Solving the inverse kinematics problem (IKP) of redundant robots is a nonlinear problem of mapping from the robot workspace to the joint space, seeking the numerical or analytical solutions of a system of nonlinear transcendental equations.

The associate editor coordinating the review of this manuscript and approving it for publication was Yangmin Li.

A. RELATED WORK

At present, the methods to solve the IKP for redundant degrees of freedom (DOF) robots are mainly numerical, such as pseudo-inverse of Jacobian, extended Jacobian, augmented Jacobian, gradient projection, weighted least-norm, damped least square, weighted least square, sequential quadratic programming, etc.

To resolve motion rate control on the IK of multi-body mechanisms, Liegéois [5] proposed the pseudo-inverse method. However, this method has the weakness of causing extraneous motion in null space. Using a composite Jacobian, Oh *et al.* [6] developed the iterative numerical method of the Newton-Raphson procedure. It can be applied to solve the IKP for redundant or non-redundant manipulators without having to derive beforehand a closed-form solution. Using a modified Newton-Raphson technique for solving a system of nonlinear equations, Goldenberg *et al.* [7] presented

a complete generalized solution to the IK of robots with any arbitrary number of DOF.

The method based on the Newton-Raphson algorithm has the disadvantage of uncertain iteration numbers and computing the pseudo-inverse of the Jacobian matrix for singular configurations. To solve the singularity problem, Nakamura and Hanafusa [8] introduced the singularity robust inverse of the Jacobian matrix as an alternative to the pseudo-inverse of the Jacobian matrix. Baillieul [9] imposed a constraint task to the mapping between joint displacements and poses of the end-effector and presented the extended Jacobian technique, which was viewed as a generalized inverse technique. By projecting the constraint Jacobian onto the null space of the end-effector Jacobian, Chiacchio *et al.* [10] developed a closed-loop IK scheme to solve the IK for redundant manipulators when additional constraints were imposed. Employing numerical estimation techniques, Tevatia and Schaal [11] introduced a computationally efficient version of the extended Jacobian that had better performance than the original version. The simplified extended Jacobian algorithm was capable of reducing the irrelevant null-space motion for certain optimization criteria. To solve the approximation problem of the IK algorithms for redundant manipulators, Ratajczak [12] introduced the approximation of dynamically consistent Jacobian. Using the Cholesky decomposition and the Ritz method, he proposed an optimal extended Jacobian IK algorithm. To keep robots far from singularities, Simas and Gregorio [13] proposed the algorithm of adaptive coefficients in constraint functions to obtain a good conditioning index of the extended Jacobian.

The sequential quadratic programming (SQP) method efficiently solves a series of related nonlinear optimization problems. As an iterative algorithm for nonlinear optimization, the SQP method is adequate for solving the problems when the objective function with constraints is twice consecutively differentiable. Leger and Angeles [14]proposed a method to solve the forgoing redundancy of 5-DOF tasks based on SQP. Jiang *et al.* [15] used the SQP method to improve the efficiency and trajectory tracking accuracy of a robot and eliminate its jerks. After parameterizing the predefined geometric path, the trajectory generation problem can be transformed into a nonlinear optimization problem with a single state variable. Lyu *et al.* [16] applied the SQP method to generate a time-optimal and energy-efficient trajectory for manipulators. By the SQP method, Taniai and Naniwa [17] computed the optimal arm angles within the constraints of the joint angles. Huang and Chen [18] used the SQP method to obtain the optimal obstacle avoidance pose of a 7-axis robotic manipulator.

Unlike redundancy resolution schemes based on differential kinematics which lack repeatability, the augmented Jacobian method is always cyclic. Using the systematically enlarged direct kinematics, Sciavicco and Siciliano [19] proposed a dynamic solution technique to solve the IKP for constrained redundant manipulators. The kinematics of a robotic manipulator was appropriately augmented by including the constraints of obstacle avoidance and limited joint range, called task-space augmentation. It introduced a constraint task to be fulfilled along with the end-effector task. Then, an augmented Jacobian matrix was set up whose inverse gave the sought joint velocity solution. The concept of task-space augmentation was also independently proposed by Egeland [20], to avoid singularities and the loss of DOF for generating a position reference of the positioning part online. Incorporating the kinematic constraints and the end-effector motion, Seraji [21] revisited the concept and developed a simple framework for the configuration control of redundant manipulators. By parameterizing the arm angle and constructing its Jacobian matrix, Cao *et al.* [22] proposed an iteration algorithm based on augmented Jacobian to solve the IKP for 7-DOF manipulators with arbitrary offsets.

Using the gradient projection optimization, Dubey *et al.* [23] presented a computationally efficient kinematic control scheme for a 7-DOF robot with a spherical wrist. This scheme was modified for general 7-DOF robots [24] and did not need to determine the pseudo-inverse of the Jacobian. Zghal *et al.* [25] extended the scheme presented by Dubey *et al.* and developed an efficient gradient projection optimization scheme for robotic manipulators with multiple degrees of redundancy. In order to avoid joint limits for redundant joint manipulators, the gradient projection method provides the optimal direction for the joint velocity vector within the null space. Nevertheless, its magnitude is not unique and is adjusted by a scalar coefficient chosen by trial and error. Using a weighted least-norm solution, Chan and Dubey [26] proposed a manipulation scheme that automatically chose an appropriate magnitude of the self-motion all through the workspace. Unlike the gradient projection method, this scheme guarantes joint limit avoidance and minimizes unnecessary self-motion.

The IK formulations based on the Jacobian matrix are inefficient and fail near kinematic singularities. To overcome kinematic singularities, Wampler [27] reformulated the exact inverse problem as a damped least-squares problem and presented an efficient vector formulation of the IKP in terms of a partial velocity matrix. Using the Gauss-Newton model of the direct kinematics function with the Levenberg-Marquardt iteration, Deo and Walker [28] proposed an adaptive nonlinear least-squares algorithm to solve the IKP for general robotic manipulators. The algorithm was insensitive to the reachability of the desired position, and convergence was ensured even if the goal position corresponds to a singular configuration. Adopting the damped least-squares technique to provide numerical robustness of the solution in the neighborhood of kinematic singularities, Caccavale *et al.* [29] utilized closed-loop IK algorithms and a two-stage algorithm to realize real-time kinematic control on a seven-joint manipulator. Buss and Kim [30] introduced the selectively damped least squares method for the IK of multi-body with multiple end-effectors. It adjusted the damping factor separately for each singular vector of the Jacobian singular value decomposition based on the difficulty of reaching the target positions.

It had advantages in convergence with fewer iterations and in not requiring interim damping constants. Le *et al.* [31] proposed two strategies based on the damped least square method to effectively deal with singularities and minimize error due to the introduction of damping. Unlike other work in which the same damping factor was used for all singular vectors, the genetic algorithm was implemented to search for a different damping coefficient for each singular vector based on the corresponding singular value of the Jacobian. To solve the problem of multi-solution caused by the redundancy of a 7-DOF manipulator, Yang *et al.* [32] introduced the rule of best compliance based on the weighted least square method. Applying a genetic algorithm to search for all global optimum solutions made the multi-solution a mono-one. Sugihara [33] proposed a robust numerical IK solution based on the Levenberg–Marquardt algorithm. The solution employed the squared norm of residual of the original equation as the damping factor and avoided heavily computing the manipulability and the minimum singular value of the coefficient matrix.

Since redundant robots are highly nonlinear, coupled multi-variable systems, it is challenging to obtain the IK solution in an analytic way that can obtain all the feasible solutions in the global joint space. Analytical methods are robot dependent and only applicable to redundant robots with simple configurations or unique geometric properties. Mostly, analytical solution algorithms employ screw theory and geometric features analysis. Hemami [34] presented solutions to four different classes of robotic manipulators obtained by locking the redundant joints. When the extra joints are randomly locked at arbitrary angles, the resultant will be non-redundant manipulators of various structures for each class. For the 7-DOF robotic manipulators of which all links length is zero, Shimizu *et al.* [35] derived a closed-form inverse kinematic solution based on a parameterization method and developed an analytical method for computing feasible solutions under the joint limits. Singh and Claassens [36] provided an analytical solution to the IKP for the Barrett WAM manipulator and addressed an analytical method to identify a set of feasible poses for some joint-angle constraints. Wang *et al.* [37] presented an analytic IK solution for a 7-DOF space manipulator with an S-R-S structure and provided an optimization approach to get a near-optimal IK solution.

In order to realize the efficient search of IK solutions for redundant DOF robots, intelligent algorithms are widely used, for example, genetic algorithms, artificial neural networks, and particle swarm optimization algorithms. Parker *et al.* [38] used genetic algorithms to position a redundant robot at a target location by minimizing the maximum joint displacement in a point-to-point positioning task. Considering the point-to-point motion of redundant manipulators working in environments with obstacles, Nearchou and Aspragathos [39] took the problem as a constrained optimization problem and solved the IKP using a method based on a simple genetic algorithm. Although the method could be successfully applied to any redundant or non-redundant manipulators with obstacles in their workspace, it produced a rather sizeable positional error of the end-effector when the environment was cluttered with obstacles or when some obstacles were too close to the desired location. To solve this problem, Nearchou [4] applied a modified genetic algorithm and presented a new approach. The new approach searched for successive robot configurations in the entire free space so that the robot moved its end-effector from initial placement to a final desired one. It showed better performance than the pseudo-inverse method and simple genetic algorithm. Tabandeh *et al.* [40] proposed a modified genetic algorithm to find multiple solutions of the IK through an adaptive niching strategy. The algorithm used a few preset parameters and could be generalized to solve the IKP of a robot with an unknown number of DOF and an unknown present configuration.

Artificial Neural network is one of the most important techniques to solve the IKP with the help of different neural network architectures. Since the neural networks work with an acceptable error, the error at the end of IK learning should be minimized. Köker [41] used a simulated annealing algorithm to improve the result obtained from the neural-network-based IK solution of robotic manipulators. A simulated annealing algorithm was used to find the best-fitting ten digits for the decimal part of the best solution selected by three Elman neural networks. Using an artificial neural network, Srivastava and Kumar [42] presented an optimization technique to get the best results for the path of the end-effector with respect to the joint angles. Unlike traditional implementations of neural networks, Feng *et al.* [43] proposed a hybrid approach to train the neural network using the collected training data from joint subspace and an efficient learning algorithm, the electromagnetism-like method. The hybrid approach could significantly reduce computation time and improve precision.

The particle swarm optimization algorithm is one of the biologically-inspired optimization techniques that can be applied to solve optimization problems with a multimodal function. Huang *et al.* [44] applied a particle swarm optimization algorithm to solve the IKP of 7-DOF robotic manipulators. The algorithm was capable of finding an optimal configuration and solving the problem more efficiently. By merging genetic algorithms and particle swarm optimization algorithm, Starke *et al.* [45] developed a novel biologically-inspired approach to solving the IKP efficiently for arbitrary joint chains. The hybridization approach performed more robustly and adaptively than traditional or related methods. Liu *et al.* [46] proposed an IK calculation method based on an improved particle swarm optimization algorithm. The method is applied to general robots. Compared to the traditional particle swarm optimization algorithm, it solved the local convergence problem and improved stability and convergence accuracy, and operation speed.

Many of the methods mentioned above have been directed to minimize a criterion function. Taking additional

constraints into account implies a time-consuming optimization process. Many numerical methods are based on inverse differential kinematics. They involve the computation pseudo-inverse of the Jacobian matrix, which has the problems of large computation amount and long computation time [47]. These methods can only get one approximate solution that depends on the initial guess and has a cumulative error owing to many iterations. The common disadvantage of numerical methods is the lack of repeatability. In other words, the joint space trajectory corresponding to a repeated task space trajectory can not repeat itself, which means that the behavior of a robot is unpredictable when periodic tasks are carried out [48]. Furthermore, numerical methods require the current joint positions of controllers and a well-planned motion path on which the poses of the end-effector in a specific interval are continuously computed. However, robot motion planning and task planning expect the joint configuration of the desired pose to be computed first, and then plan the motion path and trajectory of the end-effector. Therefore, it is difficult for general numerical methods based on inverse differential kinematics to realize them.

Analytical methods have the advantages of fast solving speed, high precision, and obtaining several reliable solutions. However, analytical methods are only applicable to 7R robots with unique geometric configurations, such as redundant humanoid arms with S-R-S structure, a robotic manipulator with all zero of joint offset or link length [49], or a robot with three consecutive parallel joints and spherical wrist joints [50]. They lack universality and portability. Even though the search methods based on intelligent algorithms can obtain multiple solutions or an optimal one, they are time-consuming, and the obtained solutions are approximate.

B. WORK OF THIS PAPER

Although solutions to the IKP have been expressed in closed form for some 7R robots with a particular configuration, it is not easy to directly obtain analytical solutions to the IKP for 7R robots with general geometric parameters due to the highly nonlinear and complex coupling properties. Whereas numerical methods based on inverse differential kinematics can solve the IKP for 7R robots with arbitrary geometric parameters, it is time-consuming to compute one solution governed by the initial guess without numerical stability. In addition, search methods based on intelligent algorithms are time-consuming to obtain approximate solutions.

It is difficult to derive the kinematic and dynamic equations of robotic systems with high DOF. There are some useful works for a wheeled mobile robotic manipulator [51], planner multi-body systems [52], tree-type robotic systems [53], and mechanisms of single-loop or multi-loop linkages [54], flexible link manipulators [55], [56], etc.

To solve the IKP of 7R robots with general geometric parameters, we propose a new method combining symbolic preprocessing, Sylvester dialytic elimination [57] and eigendecomposition to find the IK solutions, called SDE method for short. Given the geometric parameters of a 7R robot and the end-effector's desired position and direction, the displacements of all seven joints should be computed. The IKP boils down to solving a system of multivariate equations, in terms of whose algebraic properties, according to the method proposed by Raghavan and Roth [58], 14 symbolic equations are generated. After symbolic computation to separate variables, we obtain two equal polynomial matrices and then break them down into two systems of polynomial matrices equations. Setting a joint as the discrete variable, inputting numerical values, and thereby applying the Gaussian elimination method, six equations are selected from the 14 reconstructed equations to form a system of linear equations whose coefficient matrix contains only one variable. A 12×12 matrix is structured, which contains only one variable by variable substitution of the half-angle tangent function and Sylvester dialytic elimination. Rearranging the 12×12 matrix to form a 24×24 numerical matrix, we compute its eigenvalues and eigenvectors. The position of each joint is computed successively through back-substitution, and finally, several IK solutions are obtained.

Although the SDE method is not an analytical method, it has the novelty of obtaining several IK solutions at once by almost real-time computation and further obtaining the unique solution under given constraint conditions. The proposed method for the IKP of 7R robots is a combination of algebraic and numerical techniques. It involves symbolic computation, linear algebra, numerical analysis, and matrix operation. It applies to the kinematics model of a 7R robot established according to Denavit-Hartenberg (D-H) parameters convention [59] of Waldron [60] and Paul [61], without considering joint limits. Compared with the works mentioned above, the SDE method has several contributions:

- Several solutions at position level are obtained in one call.
- By contrast with numerical methods, solving time of the SDE method is shorter and numerical stability is better. It is beneficial for online motion planning and can be employed in a multi-objective optimization approach [62] for general 7R robots.
- The SDE method can solve the IKP of 7R robots with arbitrary geometric parameters and can be modified to apply to the IK of all 7-DOF serial robotic manipulators.

II. 7R ROBOT KINEMATICS MODEL

In order to solve the IKP for a 7R serial robot, the forward kinematics model should be established first. According to the D-H parameters convention of Waldron and Paul, as shown in Fig. 1, we model the geometric representation of a 7R manipulator. Seven links in a 7R manipulator are numbered from 0 to 7, so the same is the coordinate system attached to each link. The base link is 0, and the outermost link is 7. Seven joints in the robot mechanism are numbered from 1 to 7.

The four D-H parameters are joint angle θ_i, joint offset d_i, link length a_i, and link twist α_i. For articulated robots, the last three parameters are definite values, and the first one is taken

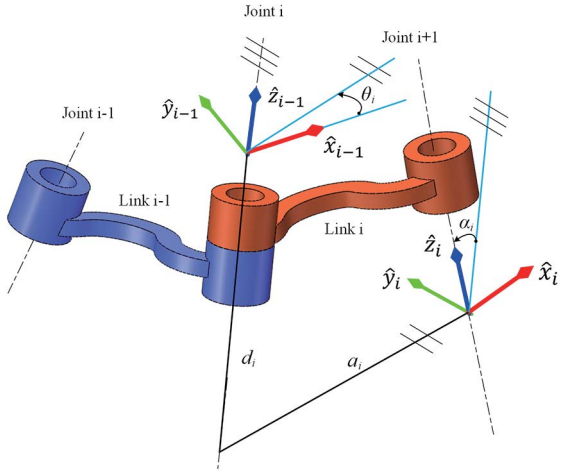

FIGURE 1. D-H parameter convention of Waldron and Paul.

as the joint variable. The general transformation of coordinate system $\{i\}$ relative to coordinate system $\{i-1\}$ is shown as (1), by multiplying four orderly independent elementary transformations based on the above four parameters, where $c\theta_i = \cos\theta_i$, $s\theta_i = \sin\theta_i$, $c\alpha_i = \cos\alpha_i$, $s\alpha_i = \sin\alpha_i$.

$$ {}^{i-1}T_i = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

Then, for a 7R manipulator, multiplying the transformation matrix of each link sequentially results in the transformation matrix of the coordinate system {7} attached to the outermost link relative to coordinate system {0} attached to the base link,

$$ {}^0T_7 = {}^0T_1\,{}^1T_2\,{}^2T_3\,{}^3T_4\,{}^4T_5\,{}^5T_6\,{}^6T_7, \tag{2} $$

where,

$$ {}^0T_7 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3} $$

Equation (2) is the formulation of forward kinematics for a 7R manipulator, which expresses the mapping relationship between joint variables and a pose of the end-effector. In other words, given values of all joint variables for a robot, we can use (2) to describe the pose of its end-effector with respect to its base. Conversely, the IKP corresponds to computing the seven joint angles satisfying (2) for the desired pose.

III. SDE METHOD

In this section, we present the process of SDE method to solve the IKP for 7R robots in detail. First, we rearrange (2), as

$$ {}^1T_2^{-1}\,{}^0T_1^{-1}\,{}^0T_7\,{}^6T_7^{-1} = {}^2T_3\,{}^3T_4\,{}^4T_5\,{}^5T_6. \tag{4} $$

As a result, each entry of the right-hand matrix of (4) is a function of θ_3, θ_4, θ_5 and θ_6, and each entry of the left-hand matrix is a function of θ_1, θ_2 and θ_7. We represent a manipulator as a chain of characters that symbolizes its position of revolute joints and general geometry. Equation 4 helps to decrease the degree and the symbolic complexity of the resulting expressions.

A. SYMBOLIC PREPROCESSING FOR VARIABLES SEPARATION

Taking the D-H parameter link length a_i, joint offset d_i, sines and cosines of link twist α_i and 1-3 rows of elements in ${}^0T_7$ as symbolic constants, the entries of 3rd and 4th columns among left-hand matrix in (4) do not contain θ_7. Thus, comparing the entries of the third and fourth columns among the corresponding right-hand matrix, we can obtain six equations with six joint variables, in which all constant terms are moved to the right side (see equations EQ01-EQ06 in Appendix A). We rearrange them to structure four vectors,

$$ \vec{p}_{left} = \begin{bmatrix} h_1c\theta_2 + h_2s\theta_2 \\ h_1s\theta_2 - h_2c\theta_2 \\ h_3 \end{bmatrix}, \vec{q}_{left} = \begin{bmatrix} n_1c\theta_2 + n_2s\theta_2 \\ n_1s\theta_2 - n_2c\theta_2 \\ n_3 \end{bmatrix}, $$

$$ \vec{p}_{right} = \begin{bmatrix} a_2 + f_1c\theta_3 + f_2s\theta_3 \\ f_3s\alpha_2 + f_2c\alpha_2c\theta_3 - f_1c\alpha_2s\theta_3 \\ d_2 + f_3c\alpha_2 - f_2s\alpha_2c\theta_3 + f_1s\alpha_2s\theta_3 \end{bmatrix}, $$

$$ \vec{q}_{right} = \begin{bmatrix} r_1c\theta_3 + r_2s\theta_3 \\ r_3s\alpha_2 + r_2c\alpha_2c\theta_3 - r_1c\alpha_2s\theta_3 \\ r_3c\alpha_2 - r_2c\alpha_2c\theta_3 + r_1s\alpha_2s\theta_3 \end{bmatrix}. $$

Considering $\vec{p}_{left}$, $\vec{p}_{right}$, $\vec{q}_{left}$ and $\vec{q}_{right}$ to be 3×1 vectors, according to the operation rules of dot product and cross product for vectors, the following four expressions are valid [63],

$$ \begin{aligned} &\vec{p}_{left} \cdot \vec{p}_{left} = \vec{p}_{right} \cdot \vec{p}_{right}, \\ &\vec{p}_{left} \cdot \vec{q}_{left} = \vec{p}_{right} \cdot \vec{q}_{right}, \\ &\vec{p}_{left} \times \vec{q}_{left} = \vec{p}_{right} \times \vec{q}_{right}, \\ &\left(\vec{p}_{left} \cdot \vec{p}_{left}\right)\vec{q}_{left} - 2\left(\vec{p}_{left} \cdot \vec{q}_{left}\right)\vec{p}_{left} \\ &\qquad = \left(\vec{p}_{right} \cdot \vec{p}_{right}\right)\vec{q}_{right} - 2\left(\vec{p}_{right} \cdot \vec{q}_{right}\right)\vec{p}_{right}. \end{aligned} $$

From the above four expressions of vectors relationship, we can derive the other eight equations employing symbolic operation (see equations EQ07-EQ14 in Appendix A). Thus, we get 14 equations containing symbolic constants and variables, in which left-hand polynomials are a linear combination of $s\theta_1s\theta_2$, $s\theta_1c\theta_2$, $c\theta_1s\theta_2$, $c\theta_1c\theta_2$, $s\theta_1$, $c\theta_1$, $s\theta_2$ and $c\theta_2$, and right-hand polynomials are a linear combination of $s\theta_4s\theta_5$, $s\theta_4c\theta_5$, $c\theta_4s\theta_5$, $c\theta_4c\theta_5$, $s\theta_4$, $c\theta_4$, $s\theta_5$, $c\theta_5$ and 1. We express them as,

$$ Q\Theta_{12} = P\Theta_{45}, \tag{5} $$

where, Θ_{12}=$(s\theta_1s\theta_2, s\theta_1c\theta_2, c\theta_1s\theta_2, c\theta_1c\theta_2, s\theta_1, c\theta_1, s\theta_2, c\theta_2)^T$, $\Theta_{45} = (s\theta_4s\theta_5, s\theta_4c\theta_5, c\theta_4s\theta_5, c\theta_4c\theta_5, s\theta_4, c\theta_4, s\theta_5, c\theta_5, 1)^T$, Q is a 14×8 symbolic constants matrix and P is a 14×9 symbolic polynomials matrix. Q is rearranged from the left-hand polynomials of the 14 equations, and P is derived from the right-hand of the 14 equations. All entries of P are linear functions of $s\theta_3$, $c\theta_3$, $s\theta_6$ and $c\theta_6$. Each entry of Q and P is a symbolic function structured from the

D-H parameter, and the relationship in (5) helps to eliminate 4 of the 6 joint variables. Symbolic preprocessing is used to separate variables, and is applicable to 7R robots with arbitrary geometry as a universal method.All entries of P are linear functions of $s\theta_3$, $c\theta_3$, $s\theta_6$ and $c\theta_6$. Each entry of Q and P is a symbolic function structured from the D-H parameter, and the relationship in (5) helps to eliminate 4 of the six joint variables. Symbolic preprocessing is used to separate variables and is applicable to 7R robots with arbitrary geometry as a universal method.

B. SYMBOLIC MATRIX ANALYSIS

The symbolic constants matrix Q in (5) has a special structure, of which many entries are zeros. In rows 3, 6, 7, 8, 9 and 14, the coefficients corresponding to $s\theta_1 s\theta_2$, $s\theta_1 c\theta_2$, $c\theta_1 s\theta_2$, $c\theta_1 c\theta_2$, $s\theta_2$ and $c\theta_2$ are all zeros, whereas in rows 1, 2, 4, 5, 10, 11, 12 and 13, the coefficients of $s\theta_1$ and $c\theta_1$ are also zeros. Thus, we break down Q into two matrices Q_1 and Q_2, and accordingly, do the same to P. They are expressed as two different system of equations,

$$Q_1\Theta_1 = P_1\Theta_{45} \tag{6}$$
$$Q_2\Theta_2 = P_2\Theta_{45} \tag{7}$$

where $\Theta_1 = (s\theta_1, c\theta_1)^T$, $\Theta_2 = (s\theta_1 s\theta_2, s\theta_1 c\theta_2, c\theta_1 s\theta_2, c\theta_1 c\theta_2, s\theta_2, c\theta_2)^T$, Q_1, Q_2, P_1 and P_2 are 6×2, 8×6, 6×9 and 8×9 matrices, respectively. After analyzing the symbolic matrices, we get two systems of matrices with lower dimension, which help to improve computing speed and effectively reduce workload of matrix computation.

C. NUMERICAL SUBSTITUTION AND MATRIX COMPUTATION

As for the polynomial coefficient matrix P in (5), which contains two joint variables, θ_3 and θ_6, it is necessary to eliminate one joint variable to form a matrix with only one joint variable. The solution is to choose one of them as a discrete variable and assign an initial value θ_{ini}, such as zero or a nominal angle value, to it within the joint position limit first. If there are no IK solutions found, it is assigned a new value in accordance with a certain increment $\delta\theta$. In this paper, we choose θ_3 and θ_6 as the discrete fixed joint parameter, respectively, and express it as ξ. Thus, given a specific configuration of a 7R robot, we substitute numerical values of D-H parameters and the sine and cosine values of discrete joint ξ into matrices Q and P, so that (5) is converted into,

$$\bar{Q}\Theta_{12} = \bar{P}\Theta_{45}. \tag{8}$$

As a result, we obtain a numerical matrix $\bar{Q}$ and a numerical polynomial matrix $\bar{P}$. All entries of $\bar{P}$ are linear combinations of $s\zeta$, $c\zeta$ and 1 (when $\xi = \theta_3$, $\zeta = \theta_6$, or when $\xi = \theta_6$, $\zeta = \theta_3$. $s\zeta = \sin\zeta$, $c\zeta = \cos\zeta$). Based on the description of Section III-B,we can obtain numerical matrices $\bar{Q}_1$ and $\bar{Q}_2$, the minors of $\bar{Q}$, and numerical polynomial matrices $\bar{P}_1$ and $\bar{P}_2$, the minors of $\bar{P}$. Since numerical matrices $\bar{Q}_1$ and $\bar{Q}_2$ are derived from symbolic operation, and each entry of them is computed at one time by numerical substitution, they are definite given geometric parameters of a robot and a pose of its end-effector. Also, each entry of $\bar{P}_1$ and $\bar{P}_2$ is a linear combination of $s\zeta$, $c\zeta$ and 1, whose numerical coefficients are definite. Thus, matrices $\bar{Q}_1$, $\bar{Q}_2$, $\bar{P}_1$ and $\bar{P}_2$ can be operated according to algorithms of numerical matrix.

1) GAUSSIAN ELIMINATION

We rearrange (8), and obtain two equations. They are expressed as

$$\bar{Q}_1\Theta_1 = \bar{P}_1\Theta_{45}, \tag{9}$$
$$\bar{Q}_2\Theta_2 = \bar{P}_2\Theta_{45}. \tag{10}$$

As a system of linear equations with respect to variables $s\theta_1$ and $c\theta_1$, (9) has two variables but six equations, and by Gaussian elimination, four equations can be eliminated. Similarly, two equations can be eliminated in (10). We perform Gaussian elimination with partial pivoting method on (9) and (10) respectively. After employing elementary row operations on $\bar{Q}_1$, $\bar{Q}_2$, $\bar{P}_1$ and $\bar{P}_2$, we obtain two equations,

$$\bar{Q}'_1\Theta_1 = \bar{P}'_1\Theta_{45}, \tag{11}$$
$$\bar{Q}'_2\Theta_2 = \bar{P}'_2\Theta_{45}. \tag{12}$$

Since row rank of $\bar{Q}'_1$ may be 0, 1, or 2, at least there are 4 rows whose elements are zeros in $\bar{Q}'_1$, and the all elements of last four rows are zeros. Since row rank of $\bar{Q}'_2$ is at most 6, at least there are 2 rows whose elements are all zeros, and all elements in the last two rows are zeros. Each entry of numerical coefficient polynomial matrices $\bar{P}'_1$ and $\bar{P}'_2$ is linear combination of variables $s\zeta$, $c\zeta$ and 1, like the form

$$\eta c\zeta + \mu s\zeta + \delta. \tag{13}$$

Matrices $\bar{P}'_1$ and $\bar{P}'_2$ are expressed as,

$$\bar{P}'_1 = c\zeta\left(\bar{A}_1\right) + s\zeta\left(\bar{B}_1\right) + \left(\bar{C}_1\right), \tag{14}$$
$$\bar{P}'_2 = c\zeta\left(\bar{A}_2\right) + s\zeta\left(\bar{B}_2\right) + \left(\bar{C}_2\right), \tag{15}$$

where $\bar{A}_1$, $\bar{B}_1$ and $\bar{C}_1$ are 6×9 numerical matrices, and $\bar{A}_2$, $\bar{B}_2$ and $\bar{C}_2$ are 8×9 numerical matrices.

2) SYLVESTER DIALYTIC ELIMINATION

After performing Gaussian elimination, left-hand polynomials of the last four equations in (11) are equal to 0, whereas the left-hand polynomials of the last two equations of in (12) are equal to 0. We select the last four rows of elements from $\bar{P}'_1$ and the last two rows from $\bar{P}'_2$ to structure a 6×9 matrix Σ. Each entry of Σ is a linear combination of $s\zeta$, $c\zeta$ and 1. As a result, we obtain equation (16),

$$\Sigma\Theta_{45} = 0. \tag{16}$$

Replacing the trigonometric function in (16) with,

$$s\zeta = \frac{2y}{1+y^2},\quad c\zeta = \frac{1-y^2}{1+y^2},\quad s\theta_4 = \frac{2x_4}{1+x_4^2},$$
$$c\theta_4 = \frac{1-x_4^2}{1+x_4^2},\quad s\theta_5 = \frac{2x_5}{1+x_5^2},\quad c\theta_5 = \frac{1-x_5^2}{1+x_5^2},$$

where $y = \tan\left(\frac{\zeta}{2}\right)$, $x_4 = \tan\left(\frac{\theta_4}{2}\right)$, $x_5 = \tan\left(\frac{\theta_5}{2}\right)$, and multiplying each equation by $1+y^2$, $1+x_4^2$ and $1+x_5^2$ to clear out the denominators, (16) is converted into the form shown as,

$$\Sigma' X_{45} = 0, \tag{17}$$

where $X_{45} = (x_4^2x_5^2,\ x_4^2x_5,\ x_4^2,\ x_4x_5^2,\ x_4x_5,\ x_4,\ x_5^2,\ x_5,\ 1)^T$. In (17), Σ' is 6×9 matrix, each entry of which is a quadratic polynomial of y. In order to perform eigendecomposition later, it is necessary to construct a new square matrix from Σ' using Sylvester dialytic elimination. Multiplying (17) by x_4, we obtain a square system of the form,

$$\Sigma'' X'_{45} = 0, \tag{18}$$

where $X'_{45} = (x_4^3x_5^2,\ x_4^3x_5,\ x_4^3,\ x_4^2x_5^2,\ x_4^2x_5,\ x_4^2,\ x_4x_5^2,\ x_4x_5,\ x_4,\ x_5^2, x_5,\ 1)^T$, $\Sigma'' = \begin{bmatrix} \Sigma' & 0_{6\times 3} \\ 0_{6\times 3} & \Sigma' \end{bmatrix}$. Σ'' is a 12×12 matrix, each entry of which is a quadratic polynomial of y. The condition for (18) to exist non-zero solutions is that the determinant of Σ'' is zero. Therefore, a degree 24 characteristic polynomial in y is equal to 0. Computing the determinant of Σ'' can introduce significant numerical errors, and the computation of real roots of the polynomial can be ill-conditioned [64], [65]. Furthermore, even though the roots of y can be obtained by computing the determinant of Σ'', the next step we back substitute the numerical value of y into Σ' in (17) and then we have to compute determinant for x_4. As a result, the numerical errors accumulate in the intermediate steps of the computing determinant and the roots of polynomials. Therefore, how to compute y from Σ'' has a significant impact on the computational efficiency and numerical stability. It is difficult to solve the roots of a high degree univariate polynomial, but in this paper, we reduce the problem of computing roots to the problem of eigenvalues computation. Since each entry of Σ'' is a quadratic polynomial of y, we express it as

$$\Sigma'' = My^2 + Ny + L, \tag{19}$$

where, M, N and L are all 12×12 numerical matrices.

3) EIGENDECOMPOSITION AND BACK-SUBSTITUTION

The condition for inverse matrix M^{-1} of matrix M to exist is that its condition number is not infinite. That is to say, M is a well-conditioned matrix. When this condition is satisfied, the unary polynomial matrix $Iy^2 + M^{-1}Ny + M^{-1}L$ in y exists. If M is ill-conditioned, we can set a new value to ξ according to $\xi = \xi_{ini} + \delta\xi n_{cnt}$ in Algorithm 1, where n_{cnt} is an integer between the maximum negative and positive increments computed from the lower limit and the upper limit of the free joint. Considering the case that M is well-conditioned, we structure a 24×24 matrix Ω from (19),

$$\Omega = \begin{bmatrix} 0_{12\times 12} & I_{12\times 12} \\ -M^{-1}L & -M^{-1}N \end{bmatrix}. \tag{20}$$

According to the research of Gohberg *et al.* [66], the 24 eigenvalues of matrix Ω are exactly the roots of determinant$(\Sigma'') = 0$. That is to say, the values of variable y correspond exactly to the eigenvalues of matrix Ω. In addition, eigenvectors of matrix Ω compose the following structure [67],

$$V = \begin{bmatrix} \vec{v} \\ y\vec{v} \end{bmatrix}, \tag{21}$$

where $\vec{v}$ happens to correspond to X'_{45}. As a result, we can compute all the roots of y, x_4 and x_5 in (18) by means of the eigendecomposition of Ω. We pick out the real roots to compute ζ, θ_4 and θ_5. Back substituting sines and cosines of ζ, θ_4 and θ_5 into (8), θ_1, θ_2 can be solved. At last, θ_7 can be solved using (2). Since V contains several terms consisting of x_4 and x_5, it is necessary to select a reasonable pair from them to compute x_4 and x_5. Each term of $\vec{v}$ has the same bound on the maximum error according to the study of Wilkinson [68]. The term with the maximum magnitude generally has a smaller relative error, and this property is used to compute x_4 and x_5 accurately. Dividing V into two parts, $\vec{v}$ and $y\vec{v}$, let

$$\vec{v}_1 = \begin{cases} \vec{v} & |y| \le 1 \\ y\vec{v} & |y| > 1 \end{cases}, \tag{22}$$

where, $\vec{v}_1 = (v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12})^T$, which corresponds to X'_{45}. thus, we can compute x_4 and x_5 from $\vec{v}_1$, whose relative error is low. In vector $\vec{v}_1$, the last element is usually not equal to 1, so we can not directly use the 9th and 11th elements to compute x_4 and x_5, respectively. To reduce the error, x_4 or x_5 is computed by selecting elements with larger magnitude from $\vec{v}_1$. That is to say, if v_1 has the maximum magnitude, then x_4 is equal to the ratio of v_1 and v_4, and so on. Two elements are selected, the ratio of which is x_4. A similar operation is performed to compute x_5. After obtaining $s\zeta$, $c\zeta$, $s\theta_4$, $c\theta_4$, $s\theta_5$ and $c\theta_5$, back substituting them into (8) and (2), the values of other variables are solved.

Algorithm 1 To Generate a New Guess for the Free Joint

Require:

ξ_{ini}, the initial guess for the free joint
ξ_{upper}, the upper limit of the free joint
ξ_{lower}, the lower limit of the free joint
$\delta\xi$, the search discretization increment

Ensure:

ξ_{new}, a new guess;

1: Initializing: $\xi_{new} \leftarrow 0$, $n_{cnt} \leftarrow 0$
2: $n_{pos} \leftarrow$ FLOOR$((\xi_{upper} - \xi_{ini})/\delta\xi)$, getting the step number from the initial guess to the upper limit
3: $n_{neg} \leftarrow$ CEIL$((\xi_{lower} - \xi_{ini})/\delta\xi)$, getting the step number from the lower limit to the initial guess
4: $n_{cnt} \leftarrow$ GET_COUNT$(n_{pos}, n_{neg}, n_{cnt})$, to generate a new counter between n_{pos} and n_{neg}
5: $\xi_{new} \leftarrow \xi_{ini} + \delta\xi n_{cnt}$, to generate a new guess for the free joint
6: **return** ξ_{new}

TABLE 1. D-H parameters of Baxter kinematics model based on the D-H parameters convention of Waldron and Paul.

Link i	θ_i(rad)	d_i(m)	a_i(m)	α_i(rad)	offset(rad)
1	θ_1	0.27035	0.069	$-\pi/2$	0
2	θ_2	0	0	$\pi/2$	$\pi/2$
3	θ_3	0.36442	0.069	$-\pi/2$	0
4	θ_4	0	0	$\pi/2$	0
5	θ_5	0.37429	0.01	$-\pi/2$	0
6	θ_6	0	0	$\pi/2$	0
7	θ_7	0.115975	0	0	0

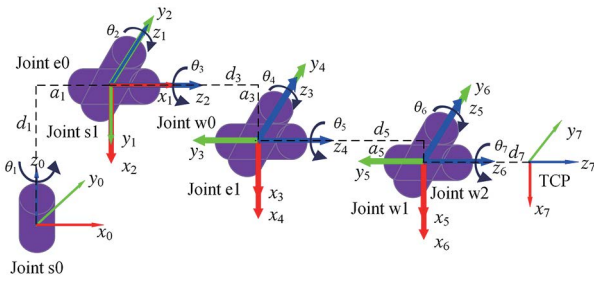

FIGURE 2. Kinematics model of left arm of Baxter based on D-H parameters convention of Waldron and Paul.

D. CONVERGENCE ANALYSIS

In the previous sections, we have presented a practical method for the IK of general 7R serial manipulators. Currently, the method is restricted to manipulators consisting of seven revolute joints, and it does not assume the geometry of the manipulator. Based on the algebraic formulation in Section III-A, we reduce the problem to computing zeros of matrix polynomials and matrix pencils. The matrix polynomial Σ'' is regular if its symbolic determinant is non-zero. Otherwise, it is a singular matrix polynomial. However, it is difficult to check whether Σ'' is singular. As an alternative approach, we compute the condition number of numerical matrix M. If the matrix is singular, its condition number is infinity. We set the rational maximum condition number as n_{con} and take it as a prerequisite for the eigendecomposition of Ω. If the condition number of M is less than n_{con}, the next computation will carry on. Otherwise, according to Algorithm 1, a new M is generated corresponding to the new value of the free joint. Because the free joint has the upper and lower limit, the amount of total iteration between them is determined, and it is the sum of n_{pos} and n_{neg}. The sum relates to the search discretization increment, $\delta\xi$. For a smaller $\delta\xi$, the SDE method needs more iterations, but it obtains better performance in success. In addition, if the solution result does not meet the final error tolerance, the iterative process will also continue. In any case, the SDE method ends after a finite number of iterations, regardless of whether the solving operation is successful. But in most cases, this method succeeds in computing the IK solution, as shown in Table 5 and Table 6.

IV. SDE METHOD IMPLEMENTATION

In this section, combining the Robotics Toolbox for MATLAB developed by Corke [69], we realize the SDE method proposed in Section III by programming in the mathematical computing software of MathWorks, MATLAB R2019B.

We use Baxter, a dual-arm robot in our laboratory, as an example to illustrate the solution to the IKP with the SDE method. Analyzing the URDF files in the control system of Baxter, we can obtain its D-H parameters. When the upper shoulder link is taken as the base link, its left arm and right arm have the same D-H parameters, and their installation positions have the same coordinate transformation, transl(0.055695, 0, 0.011038)rotz(0)roty(0)rotx(0). In this case, the SI unit of angle is rad and the SI unit of length is m. We take the left arm as the research object and list its D-H parameters in Table 1. The corresponding D-H parameter kinematics model is shown in Fig. 2.

We consider the joint positions $(-0.08, -1, -1.19, 1.94, 0.67, 1.03, -0.5)$ as the desired configuration. The pose of the left end-effector related to the mount link is computed using forward kinematics formulation,

$$T_{eef} = \begin{bmatrix} -0.4807 & 0.8755 & 0.0492 & 0.3534 \\ 0.8756 & 0.4823 & -0.0268 & -0.4109 \\ -0.0472 & 0.0302 & -0.9984 & 0.2659 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

According to

$$T_{eef} = \text{transl}(0.055695, 0, 0.011038)^0T_7,$$

it is easy to compute ${}^0T_7$,

$${}^0T_7 = \begin{bmatrix} -0.4807 & 0.8755 & 0.0492 & 0.2977 \\ 0.8756 & 0.4823 & -0.0268 & -0.4109 \\ -0.0472 & 0.0302 & -0.9984 & 0.2549 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

A. SPECIFIC EXAMPLE

Let the free joint variable be θ_6 and $\theta_6 = 1.83$, $\zeta = \theta_3$. Substituting D-H parameters and the elements of ${}^0T_7$ into symbolic matrices Q and P, after operation according to Section III, we can obtain eight real eigenvalues about θ_3. The results is shown in Table 2. And the 8 solutions of joint variables for the specific pose of the left end-effector and the corresponding pose errors are computed and given in Table 3.

B. GENERAL EXAMPLE

Let the free joint variable be θ_3 and θ_6, respectively. The initial guess θ_{ini} is zero, and the discretization value $\delta\theta$ for searching the valid free joint displacement is 5 degrees. In Robotics Toolbox for MATLAB, tr2delta(T_0, T_1) is the function to compute the differential motion corresponding to infinitesimal motion from pose T_0 to T_1 which are homogeneous 4×4 transformations. We can use it to compute the error of the pose corresponding to the computed joint values relative to the desired pose T_{eef}. The error is expressed as a vector $(\delta x, \delta y, \delta z, \delta R_x, \delta R_y, \delta R_z) \times 10^{-12}$, which represents

TABLE 2. Eight real eigenvalues and their condition numbers.

No.	1	2	3	4	5	6	7	8
Real Eigenvalue	1.3839	1.3620	-1.3284	-1.3250	0.7508	0.7597	-0.7303	-0.7294
Condition Number	2.1718	2.3962	2.1746	2.3885	2.0117	2.2191	2.0153	2.2166

TABLE 3. Joint angels of 8 solutions.

No.	θ_1(rad)	θ_2(rad)	θ_3(rad)	θ_4(rad)	θ_5(rad)	θ_6(rad)	θ_7(rad)	δx	δy	δz	δR_x	δR_y	δR_z
1	1.6225	-3.0983	1.8901	-1.2159	1.4879	1.8300	0.7573	0.0222	-0.0048	-0.0173	-0.0185	-0.0049	-0.0018
2	-1.8743	0.0651	1.8750	1.9494	-1.5496	1.8300	1.1324	-0.0473	0.0021	0.0088	-0.0127	-0.0184	0.0045
3	2.7691	-3.0766	-1.8510	-1.2170	-1.4830	1.8300	-0.5116	.0833	0.0030	-0.0589	0.0591	-0.0279	0.0070
4	-0.0138	0.0193	-1.8486	1.9470	1.5426	1.8300	-0.8866	0.2983	0.0677	0.0469	.0430	-0.1602	-0.0234
5	-3.0734	-3.1817	1.2881	1.6693	1.6354	1.8300	-0.5116	0.0160	-0.0015	0.0365	0.0033	-0.0164	0.0030
6	-0.3177	-0.0833	1.2994	-1.6269	-1.6148	1.8300	-0.8866	0.1283	-0.0205	-0.0659	-0.0095	0.0473	0.0107
7	1.1861	-3.2152	-1.2616	1.6667	-1.6282	1.8300	0.7573	0.1887	-0.0252	0.4440	-0.0593	-0.1739	-0.0455
8	-1.5735	-0.0494	-1.2604	-1.6282	1.6095	1.8300	1.1324	0.4621	-0.0347	-0.2522	0.0677	0.1280	-0.0328

infinitesimal translation and rotation. The final error tolerance for the pose of a valid IK solution is expressed as ϵ. It is computed on the norm of the error between the current and desired pose, and $\epsilon = 10^{-10}$. Failure may happen when the SDE solver runs at the initial guess for the free joint. In this case, the value of the free joint can be set according to Algorithm 1. In this way, the success rate will improve markedly.

We substitute D-H parameters and the pose data into the computation program. When $\zeta = \theta_6$, 8 valid eigenvalues are computed after one iteration, which are 4.4597, -3.9620, 2.5621, -2.3731, 0.5757, 0.4520, -0.5407 and -0.4203. And when $\zeta = \theta_3$, 4 valid eigenvalues are computed after 20 iterations, which are 6.4390, -5.0230, 0.3467 and -0.2783. At last, 12 solutions of joint variables for the specific pose of the left end-effector and the corresponding manipulability indices are computed as given in Table 4. We can choose the No.5 solution as the best one of the 12 solutions, which has the max manipulability index. In addition, for several special configurations, such as zero joint angle configuration, vertical configuration, stretched configuration, and nominal configuration, the SDE method can find the IK solution, too.

C. MORE EXAMPLES

We test the SDE method on seven types of 7R robots which are selected from the robots models library of RoboDK V5.2.2, a simulation software. The seven robots are Daihen OTC FD-V6LS, Franka Emika Panda, Kinova Gen3, KUKA LBR iiWA 14 R820, Motoman SIA20D, Siasun SCR3 and Yumi IRB 14050. The standard D-H parameters of these seven robots are imported into the MATLAB program of the SDE method, and the pose of the end-effector is computed according to the configuration data of the parameter panel for each robot in Fig. 6 to Fig. 12 (see Appendix B). For each robot, we solve the IKP corresponding to the pose of its end-effector, and we list the IK solutions in Table 12 to Table 18 (see Appendix B), respectively.

Generally, the wrist structures of 7-DOF manipulators are designed in two different types: the Euler wrist and the offset wrist. Offset wrists are used with the robot manipulators that are needed to get long horizontal reach while maintaining the appropriate angle in their workspace, while the Euler wrists provide regularly shaped workspace [70]. Solving the IKP is very difficult for the robot manipulators that have an offset wrist. We test the SDE method on the YASKAWA VA1400II robot [22] that has the offset wrist (see Appendix C).

V. EXPERIMENTS AND COMPARATIVE ANALYSIS

In this section, we carry out simulation experiments and one practical experiment. The simulation experiment is a comparative study on the performance of the SDE method and the IK tools in the Robotics Toolbox for MATLAB (version 10.4).

A. PERFORMANCE

Robotics Toolbox for MATLAB is a MATLAB-based robot modeling and simulation toolbox developed by Peter Corke from Australia. It contains three built-in functions, ikine, ikunc, and ikcon, to solve general IKP. Function ikine applies the Gauss-Newton iterative algorithm [71] optimized by the Levenberg-Marquardt method [72] to compute IK solutions. Function ikcon searches the minimum of nonlinear multivariate objective function considering joint limits, whereas function ikunc does the same but does not consider joint limits. Function ikunc applies the Quasi-Newton method [73], whereas function ikcon applies the SQP method [74]. We take the kinematics model of the left arm in Section IV as the simulation object. At first, 1000 sets of joint positions are generated by random function within the joint limits. Each configuration corresponds to a pose of the end-effector, so we have 1000 configurations and 1000 corresponding poses. The final error tolerance are $\epsilon = 10^{-2}$, $\epsilon = 10^{-3}$, $\epsilon = 10^{-4}$, $\epsilon = 10^{-5}$, $\epsilon = 10^{-6}$, $\epsilon = 10^{-7}$, respectively. We compute the solutions corresponding to each pose using the SDE method and the three IK functions, respectively. The configuration of zero joint displacements is considered the

TABLE 4. Joint angels of 12 solutions and manipulability indices.

No.	θ_1(rad)	θ_2(rad)	θ_3(rad)	θ_4(rad)	θ_5(rad)	θ_6(rad)	θ_7(rad)	manipulability index
1	-0.9505	0.5238	0	-1.7046	0.0574	2.7004	0.1700	0.0834
2	-0.9484	0.5238	0	-1.6512	-3.0902	-2.6471	-2.9761	0.0849
3	2.1915	-3.8389	0	1.5746	-3.1054	2.3974	0.1451	0.0866
4	2.1929	-3.8389	0	1.5212	0.0341	-2.3440	-2.9979	0.0881
5	2.1916	-2.4704	0	-1.1470	-3.1133	1.0447	0.1044	0.0905
6	-0.9500	-1.3539	0	2.0254	0.0326	0.8491	0.0970	0.0681
7	2.1928	-2.4704	0	-1.2004	0.0292	-0.9913	-3.0378	0.0899
8	-0.9487	-1.3539	0	2.0788	-3.1074	-0.7957	-3.0456	0.0656
9	-0.7700	-0.9591	2.8334	-1.6480	-2.8752	0.8727	-0.1212	0.0786
10	-1.1782	-0.9455	-2.7486	-1.6454	2.8710	0.8727	0.3790	0.0790
11	-1.4713	-1.2750	0.6675	1.9916	-0.2137	0.8727	0.3721	0.0642
12	-0.5234	-1.3046	-0.5428	2.0029	0.2168	0.8727	-0.1159	0.0654

TABLE 5. SDE method vs the three inverse kinematics functions.

Tol	SDE($\xi = \theta_3$)		SDE($\xi = \theta_6$)		IKINE		IKUNC		IKCON	
	time(s)	successes num	time(s)	successes num	time(s)	successes num	time(s)	successes num	time(s)	successes num
1e-02	0.023755	1000	0.017399	995	0.27372	572	0.45456	991	0.42577	555
1e-03	0.022796	1000	0.017018	995	0.2733	572	0.45478	987	0.42588	537
1e-04	0.023092	1000	0.016995	995	0.27285	572	0.45454	974	0.42594	523
1e-05	0.023014	1000	0.016977	995	0.27313	572	0.45466	968	0.42649	520
1e-06	0.022909	1000	0.016972	995	0.27309	572	0.4549	694	0.42651	510
1e-07	0.022932	1000	0.017043	995	0.27316	572	0.45545	40	0.42658	184

TABLE 6. SDE method vs IKINE.

method to IK	average cost time(s)	rate of success
SDE($\xi = \theta_3$)	0.02253	100%
SDE($\xi = \theta_6$)	0.01691	99.40%
IKINE	0.26543	58.39%

initial guess for these four IK solution methods. Computing results are listed in Table 5. From Table 5 we can see that the numbers of successes for ikunc and ikcon decrease when the convergence condition gets stricter. For $\epsilon = 10^{-10}$ ikunc and ikcon will fail, so Table 6 only compares the SDE method to ikine for 10000 random poses. When $\xi = \theta_3$, the average number of iterations is 1.0004 before the SDE method finds the IK solution, whereas the average number of iterations is 13.8042 when $\xi = \theta_6$.

To study the data in Table 5, we can see that functions ikine, ikunc, and ikcon cost more time to search IK solutions. Compared to the numerical methods in Robotics Toolbox for MATLAB, the running time of the SDE method is very short, and meantime, it ensures a good success rate for the IKP. On a personal computer, configuring with CPU of Intel® Core™ i7-4712MQ@2.3GHz and 16G memory, the average running time is reduced to 20ms, whereas the other three inverse methods take ten to twenty times as long as SDE method.

B. EXPERIMENTAL VERIFICATION

Fig. 3 shows the experimental platform, and it is composed of a Baxter robot, a Leica laser tracker, a PC workstation with the Robot Operating System (ROS) to control the robot, and a PC station to collect the data generated by the laser tracker.

At first, in the MoveIt tool, we choose 15 groups of different joint positions according to Fig. 4. Each group of joint positions corresponds to a pose of the end-effector, and the SDE method is used to compute the IK solutions. We select a reasonable one conforming to the joint limit conditions from all IK solutions for each pose. Then, the workstation configured with ROS and Baxter SDK is connected to Baxter through the network protocol, and the reflector ball of the laser tracker is mounted at the end of the left arm. The terminal executes two Python scripts containing the joint positions generated by the MoveIt tool and computed IK solutions by the SDE method. The robot is controlled online to reach 15 configuration states successively. The laser tracker captures the Cartesian space positions of the end-effector, and two sets of point cloud data are obtained. The former set of data is taken as the reference data, and the deviation of the latter set of data is measured.

The data in Table 7 are joint positions generated in the MoveIt tool, and the spatial position of the end-effector corresponding to each configuration measured by the laser tracker is listed in Table 9. The data in Table 8 are selected from IK solutions solved by the SDE method, and the measured spatial position of the end-effector for each configuration is listed in Table 10. We regard the data in Table 9 as reference data, and compute deviation of the data in Table 10. The deviation results are listed in Table 11. Fig. 5 shows the positions of two sets of data in the same coordinate system. The red points are

TABLE 7. Joint positions from MoveIt.

No. of configuration	θ_1(rad)	θ_2(rad)	θ_3(rad)	θ_4(rad)	θ_5(rad)	θ_6(rad)	θ_7(rad)
1	-0.0837	-1.0786	-0.9590	1.1413	0.1401	0.1029	-0.3897
2	-0.0837	-1.0786	-0.9590	1.1413	-0.4824	0.2587	-0.0673
3	-0.2267	-0.8442	-0.8912	0.9929	-1.0138	0.2667	-0.5806
4	-0.2267	-0.5816	-0.9779	1.0308	-1.2046	0.3704	-0.0606
5	0.0242	-0.7084	-0.9779	1.1517	-0.8926	0.2151	-0.5635
6	0.3808	-0.6359	-1.3927	1.5447	-0.5286	0.2462	-1.4646
7	0.5255	-0.7263	-1.2027	1.5900	-0.9446	0.8066	-0.5635
8	0.1399	-0.4641	-0.7874	1.1896	-1.4646	0.8689	0.5635
9	-0.1879	-0.5816	-0.5278	1.0156	-1.7589	0.7754	-0.3035
10	-0.5449	-0.7988	-0.0434	1.0535	-1.6727	0.9520	0.1297
11	-0.2845	-0.7988	-0.3891	1.2501	-1.7938	0.6614	0.7195
12	-0.1494	-0.8442	-0.5107	1.4465	-1.3949	0.8689	0.7537
13	0.6317	-0.4008	-0.0434	1.2048	-1.1698	1.8239	0.7537
14	-0.4870	-0.8349	-0.1991	1.2576	-1.4646	0.8377	0.0434
15	-0.4772	-0.8078	0.1643	1.0233	-1.8629	1.0660	0.8406

TABLE 8. Joint positions computed by SDE.

No. of configuration	θ_1(rad)	θ_2(rad)	θ_3(rad)	θ_4(rad)	θ_5(rad)	θ_6(rad)	θ_7(rad)
1	-0.1151	-1.1062	-0.8813	1.1479	-0.2264	0.1029	-0.0701
2	0.0270	-0.8955	-1.3231	1.1160	0.4298	0.2587	-0.7097
3	-0.1039	-0.4373	-1.6582	0.9634	0.9917	0.2667	-1.8802
4	-0.1818	0.0444	-1.9967	1.0205	1.2031	0.3704	-1.4532
5	0.1267	-0.4094	-1.4703	1.1278	0.8647	0.2151	-1.8750
6	0.3957	-0.4135	-1.6079	1.5413	0.5232	0.2462	-2.3095
7	0.4669	0.2433	-2.0818	1.6019	0.9437	0.8066	-1.7190
8	-0.0654	0.7167	-2.7644	1.2168	1.4327	0.8689	-0.6493
9	-0.3894	0.5537	-3.0263	1.0319	1.7139	0.7754	-1.5101
10	-0.3729	0.3952	-2.7678	1.0444	1.7127	0.9520	-0.9352
11	-0.1314	0.4489	-2.4620	1.2313	1.8188	0.6614	-0.9840
12	0.0697	0.4429	-2.2250	1.4139	1.4212	0.8689	-0.6610
13	0.9431	0.9204	-2.5596	1.1821	1.2314	1.8239	-0.2640
14	-0.1243	0.3347	-2.2403	1.2104	1.5198	0.8377	-1.2609
15	-0.3813	0.3954	-3.1230	1.0251	1.8867	1.0660	-0.1278

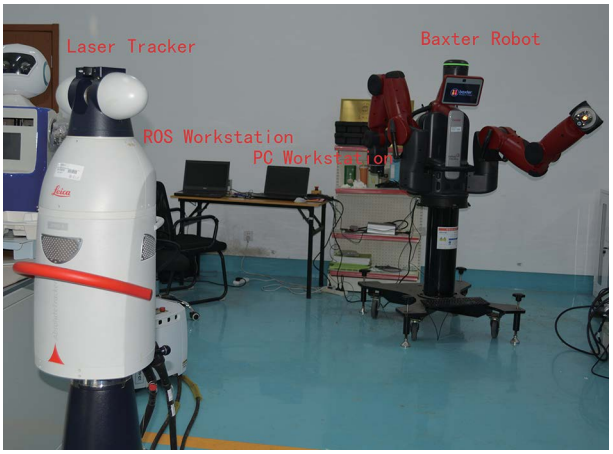

FIGURE 3. Experiment scene to test SDE method.

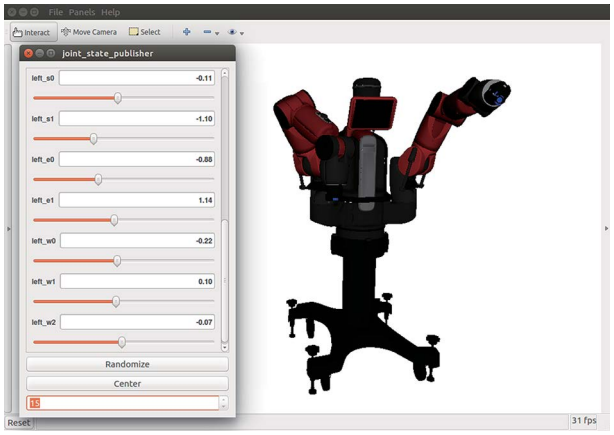

FIGURE 4. Configuration of Baxter in MoveIt tool.

reference data, and the blue points are the data generated by the SDE method. In Fig. 5, the measured data are very close to the observational data.

C. ANALYSIS AND DISCUSSION

In Section V-A, the simulation shows that the SDE method can solve the IKP in a shorter time than general numerical methods in Robotics Toolbox for MATLAB, and it can successfully solve the IKP of any poses in a workspace. Moreover, it has good numerical stability, better accuracy, and computational efficiency. Due to Algorithm 1, the SDE has a nearly 100% success rate, and its numerical stability is improved. For the same initial guess, the SDE method spends several iterations before it finds the solutions, and

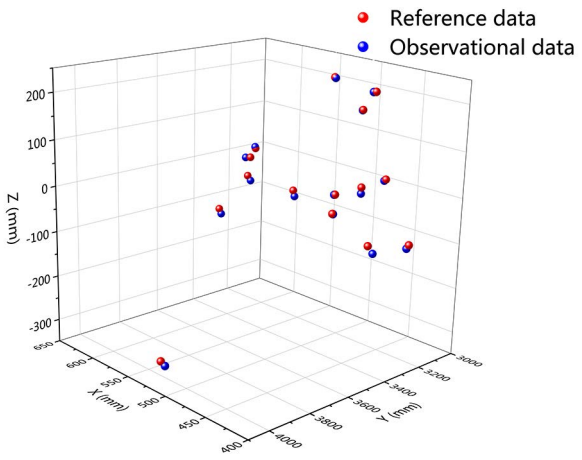

FIGURE 5. Measured data by laser tracker.

its computational complexity is considered to be $O(n)$. Even though the error tolerance is getting smaller, the average time to compute the joint angles does not change significantly. By contrast, ikine has an almost 40% probability of failing because it reaches the max iteration number, and ikunc and ikcon have the worse performance in searching for a higher precision solution. The proposed method is more efficient than the three numerical methods due to the following reasons:

1) **Convergence** The SDE method has a certain number of iterations. Although the three numerical methods set a maximum number of iterations, they can not guarantee convergence before meeting the end condition.
2) **Time Cost** In Table 5, the SDE method spends the shortest time among the four IK methods.
3) **Tollerance** The SDE method can compute the IK solution at higher precision and does not affect the solving success rate. The numerical methods fail probably to obtain higher precision.
4) **Success Rate** The SDE method has a better performance in solving the success rate than the three numerical methods.

To compare the configuration data provided by RoboDK and the corresponding IK solutions, we can see that the SDE method is suitable for 7R robots with different geometry. Given the desired pose of the end-effector, the SDE method can obtain several IK solutions, some of which accord to the joint limits and some do not. Imposing limit constraints can reject the IK solutions that do not satisfy the robot structure. In addition, the value of joint 6 in Table 12 to 18 has an offset plus the joint position provided by RoboDK, respectively. If the offset is zero, there must be one among computed IK solutions almost consistent with the data in RoboDK. Thus, it proves that, under the condition of the existence of IK solutions for a 7R robot, the SDE method can solve the IKP successfully.

Considering that the position accuracy of the Baxter robot is about ± 0.5 mm and the manipulator is not stationary when it has been in the desired state because of the series elastic actuators, the result in Table 10 is acceptable. The average deviation is 7.942 mm, and we can see that the IK solution obtained by the SDE method can make the robot reach a desired spatial position. The main interest of our work is on 7-DOF serial robotic manipulators with only revolute joints. We hope our work can contribute to the IK solution for general 7R robots.

TABLE 9. Measured positions corresponding to configurations in MoveIt.

No. of Postion	X(mm)	Y(mm)	Z(mm)
1	505.087463	3242.273438	230.599899
2	461.20874	3220.952148	216.284302
3	487.842712	3157.281494	161.720428
4	465.391693	3115.535156	17.238596
5	604.035339	3266.799072	9.197744
6	480.194702	3333.934326	-2.032487
7	460.752472	3436.831055	-21.194323
8	622.513123	3363.721191	-113.29878
9	612.273315	3246.015869	-40.626881
10	482.735809	3170.289551	-4.431376
11	555.16803	3226.474609	-45.879261
12	452.657715	3267.641846	-105.9828
13	542.495117	3968.130371	-331.0466
14	424.609802	3171.694336	-97.588051
15	594.58374	3278.513428	35.879925

TABLE 10. Measured data corresponding to solutions computed by SDE.

No. of Postion	X(mm)	Y(mm)	Z(mm)
1	504.240112	3242.141113	229.459244
2	463.723389	3222.563965	214.984726
3	488.484589	3156.164307	160.714371
4	467.364777	3114.633789	13.239909
5	609.969788	3269.420898	6.057378
6	481.143005	3334.086426	-2.716518
7	460.729828	3434.337158	-21.899275
8	621.022644	3358.265137	-124.93841
9	609.846069	3239.131592	-51.666603
10	483.836639	3166.0271	-19.404547
11	554.078308	3223.032959	-59.460335
12	447.082092	3265.818604	-119.62746
13	539.131348	3959.597656	-340.1683
14	427.325775	3170.226563	-107.32658
15	595.02002	3279.355957	39.190998

TABLE 11. Deviation results.

mean deviation	standard deviation	maximum	minimum	RMS
7.942 mm	5.592 mm	15.67 mm	1.179 mm	9.65 mm

VI. CONCLUSION

1) To solve the IKP of a 7R robot with arbitrary geometry, analytical methods are not applicable. In contrast, the numerical methods based on inverse differential kinematics can only find an approximate solution given an initial guess, and they are numerically unstable. This paper proposes a general method called SDE, applicable to 7R robots with arbitrary geometry. The SDE

method can compute several IK solutions by combining symbolic preprocessing, Sylvester dialytic elimination, and eigendecomposition. It employs symbolic preprocessing to separate variables, Sylvester dialytic elimination to construct a square matrix of coefficients, and eigendecomposition to compute roots of the specific variables. Finally, the IKP is solved by matrix operation and back substitution.

2) Comparing to the analytical methods only capable of solving the IKP of 7R robots with a particular configuration, the SDE method has general applicability to 7R robots with arbitrary geometric parameters.
3) Comparing to the solution methods based on inverse differential kinematics, the SDE method is almost real-time to obtain solutions and is more effective and numerically stable. Given the pose of the end-effector and an initial discrete value of the free joint, the SDE method can compute several IK solutions that are suitable to be taken as initial states for motion planning.
4) SDE method is proposed for general 7R robots without considering joint limits. Although it applies to the kinematics model of 7R robots established according to the D-H parameter convention of Waldron and Paul, after parameters converting, it is also applicable to the kinematics model of 7R robots established according to the D-H parameter convention of Craig [75]. This method has the potential to provide a feasible way to solve the IKP of hyper redundant robots and redundant robots with prismatic joints.
5) According to the basic D-H parameter description, we know that each joint has four parameters, namely θ, d, a, and α. So for a 7R robot, there are 28 design variables in total. Seven of them vary, and the rest fix and determine the robot's geometry. However, due to manufacturing errors, the number of geometry parameters of each joint will be expanded to six. The two added are the offset in the $\hat{y}$ axis direction and the rotation around the $\hat{y}$ axis. In fact, after the calibration process, the values of the two extra parameters can be determined. In this way, although 42 design variables are generated, in the symbol processing stage, by considering six θs as variations and the rest as constants, 14 equations similar to those in Appendix VI can still be derived. Of course, these equations are a bit more complicated, but the derivation process is similar. Then, in the following procedure of symbolic matrix analysis, numerical substitution, and matrix calculation, at least six equations with the right side of zeros can be obtained. Thus, an algebraic matrix Σ' with only one variable y is formed. Finally, Σ' is transformed to a 12×12 matrix Σ'' through the Sylvester dialytic elimination, and its dimension corresponds to X'_{45}, neither increasing nor decreasing.
6) The algebraic matrix Σ'' contains only one variable y, the equation whose determinant is zero is of a high order, and it is not easy to find all the roots. Even if the value of a joint angle variable can be found by constructing a univariate higher-order equation, substitute it back to (17), and then construct a higher-order equation about θ_4 or θ_5, and calculate the value of each joint angle in turn. The whole calculation progress will generate a more significant calculation error. By calculating the condition number of the matrix M in (19), M is judged whether ill-conditioned or not. If well-conditioned, the matrix Ω is constructed, and three joint variables are calculated at one time through eigendecomposition. Otherwise, a new matrix M is generated by changing the angle of the free joint without complicating the calculation process. Therefore, although there are unreasonable coefficients, the calculation of the three joint angle variables with significant errors can be avoided by the condition number calculation. In addition, the SDE method can filter out unnecessary inverse solutions by setting an error threshold after obtaining multiple inverse solutions.
7) For the Baxter robot, the SDE method can find the IK solution after fewer iterations when $\xi = \theta_3$. Nevertheless, a failure situation happens when the free joint variable is θ_3 for robots Daihen OTC FD-V6LS, Kinova Gen3, KUKA LBR iiWA 14 R820, Motoman SIA20D, and Siasun SCR3. In many cases, when the free joint variable is θ_6, the SDE method can find the IK solution faster but with more iterations. Further research is needed before we can explain why the above situation occurs.

APPENDIX A
DERIVED EQUATIONS

EQ01: $h_1c\theta_2 + h_2s\theta_2 = a_2 + f_1c\theta_3 + f_2s\theta_3$
EQ02: $h_1s\theta_2 - h_2c\theta_2 = f_3s\alpha_2 + f_2c\alpha_2c\theta_3 - f_1c\alpha_2s\theta_3$
EQ03: $h_3 = d_2 + f_3c\alpha_2 - f_2s\alpha_2c\theta_3 + f_1s\alpha_2s\theta_3$
EQ04: $n_1c\theta_2 + n_2s\theta_2 = r_1c\theta_3 + r_2s\theta_3$
EQ05: $n_1s\theta_2 - n_2c\theta_2 = r_3s\alpha_2 + r_2c\alpha_2c\theta_3 - r_1c\alpha_2s\theta_3$
EQ06: $n_3 = r_3c\alpha_2 - r_2c\alpha_2c\theta_3 + r_1s\alpha_2s\theta_3$
EQ07: $h_1^2 + h_2^2 + h_3^2 = (2a_2f_1 - 2d_2f_2s\alpha_2)c\theta_3 + (2a_2f_2 + 2d_2f_1s\alpha_2)s\theta_3 + 2d_2f_3c\alpha_2 + f_1^2 + f_2^2 + f_3^2 + d_2^2 + a_2^2$
EQ08: $h_1n_1 + h_2n_2 + h_3n_3 = (a_2r_1 - d_2r_2s\alpha_2)c\theta_3 + (a_2r_2 + d_2r_1s\alpha_2)s\theta_3 + f_1r_1 + f_2r_2 + f_3r_3 + d_2r_3c\alpha_2$
EQ09: $h_2n_1 - h_1n_2 = (f_1r_3s\alpha_2 - f_3r_1s\alpha_2 + a_2r_2c\alpha_2)c\theta_3 + (f_2r_3s\alpha_2 - f_3r_2s\alpha_2 - a_2r_1c\alpha_2)s\theta_3 + a_2r_3s\alpha_2 + (f_1r_2 - f_2r_1)c\alpha_2$
EQ10: $(h_3n_2 - h_2n_3)c\theta_2 + (h_1n_3 - h_3n_1)s\theta_2 = (f_2r_3 - f_3r_2s\alpha_2 - d_2r_2c\alpha_2)c\theta_3 + (f_3r_1 - f_1r_3 + d_2r_1c\alpha_2)s\theta_3 - d_2r_3s\alpha_2$
EQ11: $(h_3n_1 - h_1n_3)c\theta_2 + (h_3n_2 - h_2n_3)s\theta_2 = (d_2r_1 + a_2r_2s\alpha_2 - f_1r_3c\alpha_2 + f_3r_1c\alpha_2)c\theta_3 + (d_2r_2 - a_2r_1s\alpha_2 - f_2r_3c\alpha_2 + f_3r_2c\alpha_2)s\theta_3 - a_2r_3c\alpha_2 + (f_1r_2 - f_2r_1)s\alpha_2$
EQ12: $(-n_1h_1^2 + n_1h_2^2 + n_1h_3^2 - 2n_2h_1h_2 - 2n_3h_1h_3)c\theta_2 + (n_2h_1^2 - n_2h_2^2 + n_2h_3^2 - 2n_1h_1h_2 - 2n_3h_2h_3)s\theta_2 = (-r_1a_2^2 + 2a_2d_2r_2s\alpha_2 + r_1d_2^2 - 2d_2f_1r_3c\alpha_2 + 2d_2f_3r_1c\alpha_2 - r_1f_1^2 + r_1f_2^2 + r_1f_3^2 - 2r_2f_1f_2 - 2r_3f_1f_3)c\theta_3 + (-r_2a_2^2 - 2a_2d_2r_1s\alpha_2 + r_2d_2^2 - 2d_2f_2r_3c\alpha_2 + 2d_2f_3r_2c\alpha_2 + r_2f_1^2 - r_2f_2^2 + r_2f_3^2 - 2r_1f_1f_2 - 2r_3f_2f_3)s\theta_3 + 2d_2(f_1r_2 - f_2r_1)s\alpha_2 - 2a_2d_2r_3c\alpha_2 - 2a_2(f_1r_1 + f_2r_2 + f_3r_3)$

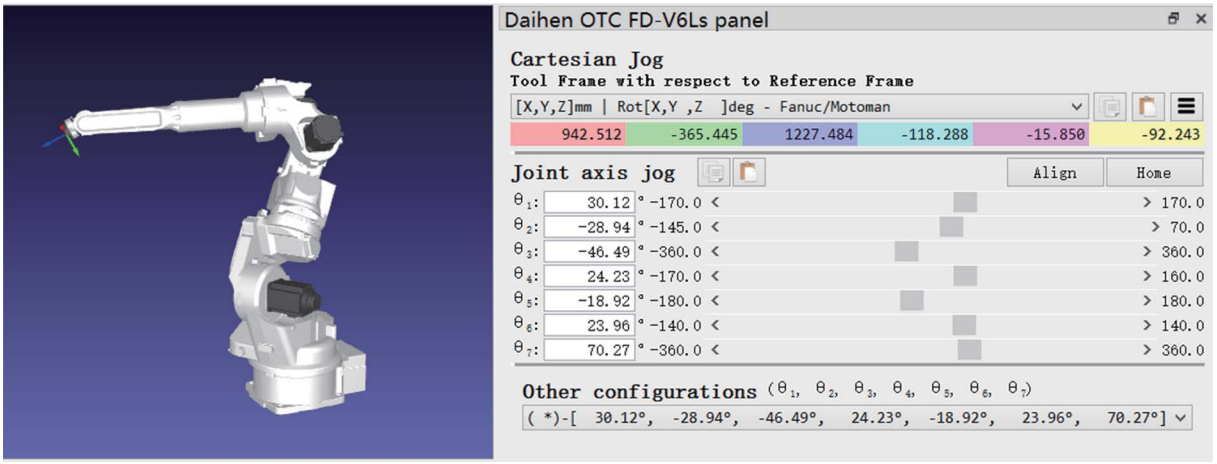

FIGURE 6. Configuration of Daihen OTC FD-V6Ls in RoboDK.

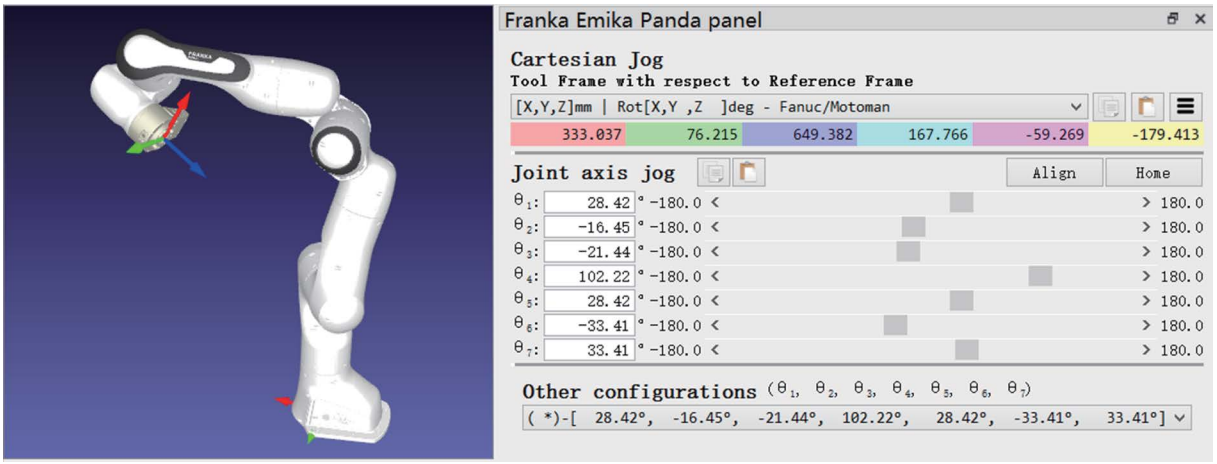

FIGURE 7. Configuration of Franka Emika Panda in RoboDK.

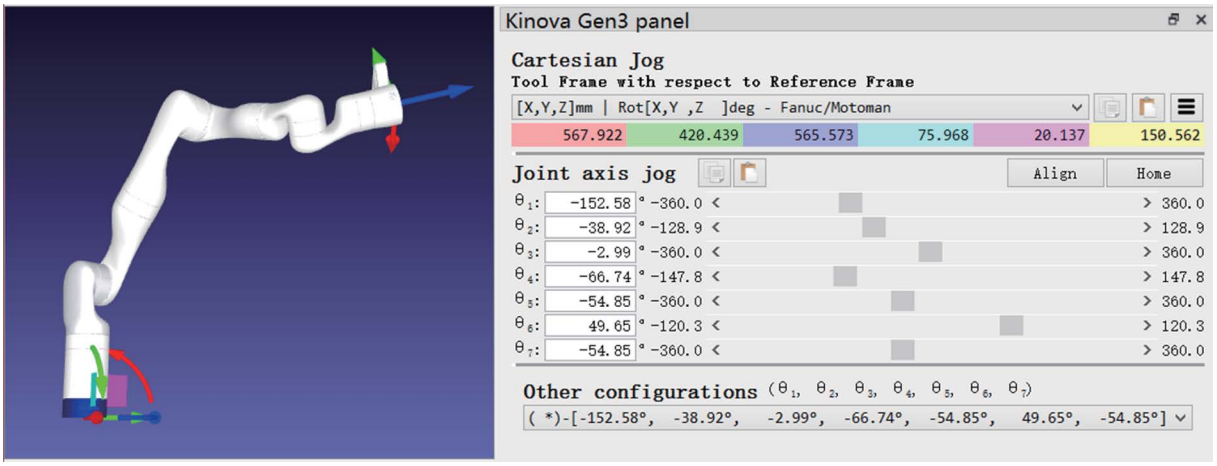

FIGURE 8. Configuration of Kinova Gen3 in RoboDK.

EQ13: $(-n_2h_1^2 + n_2h_2^2 - n_2h_3^2 + 2n_1h_1h_2 + 2n_3h_2h_3)c\theta_2 + (-n_1h_1^2 + n_1h_2^2 + n_1h_3^2 - 2n_2h_1h_2 - 2n_3h_1h_3)s\theta_2 = (r_2a_2^2c\alpha_2 + 2a_2f_1r_3s\alpha_2 - 2r_1a_2f_3s\alpha_2 + r_2d_2^2c\alpha_2 - 2r_3f_2d_2 + 2r_2d_2f_3 + r_2f_1^2c\alpha_2 - 2r_1f_1f_2c\alpha_2 - r_2f_2^2c\alpha_2 - 2r_3f_2f_3c\alpha_2 + r_2f_3^2c\alpha_2)c\theta_3 + (-r_1a_2^2c\alpha_2 + 2a_2f_2r_3s\alpha_2 - 2a_2f_3r_2s\alpha_2 - r_1d_2^2c\alpha_2 + 2r_3d_2f_1 - 2r_1d_2f_3 + r_1f_1^2c\alpha_2 + 2r_2f_1f_2c\alpha_2 + 2r_3f_1f_3c\alpha_2 - r_1f_2^2c\alpha_2 - r_1f_3^2c\alpha_2)s\theta_3 + r_3a_2^2s\alpha_2 + 2a_2f_1r_2c\alpha_2 - 2a_2f_2r_1c\alpha_2 + r_3d_2^2s\alpha_2 + r_3f_1^2s\alpha_2 - 2r_1f_1f_3s\alpha_2 + r_3f_2^2s\alpha_2 - 2r_2f_2f_3s\alpha_2 - r_3f_3^2s\alpha_2$

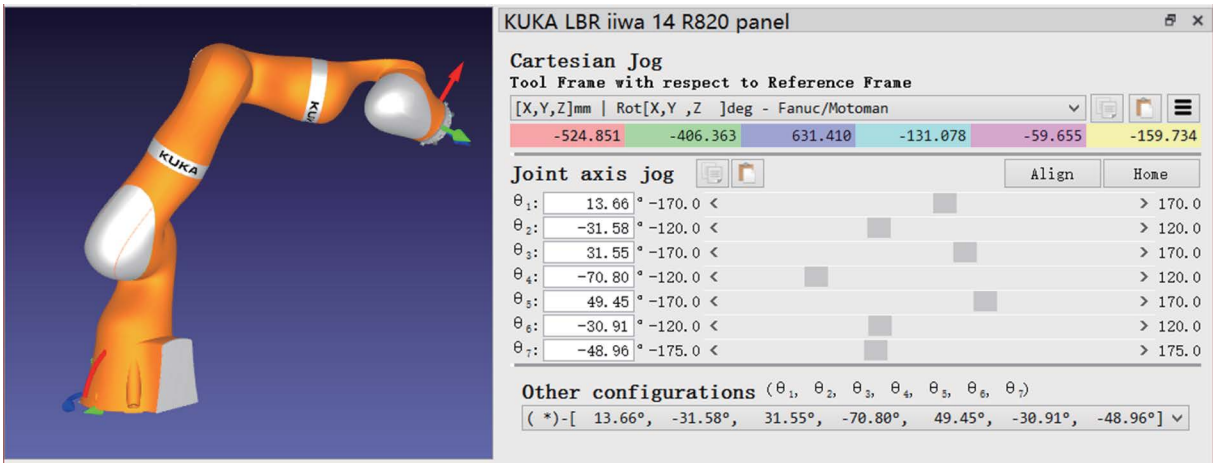

FIGURE 9. Configuration of KUKA LBR iiwa 14 R820 in RoboDK.

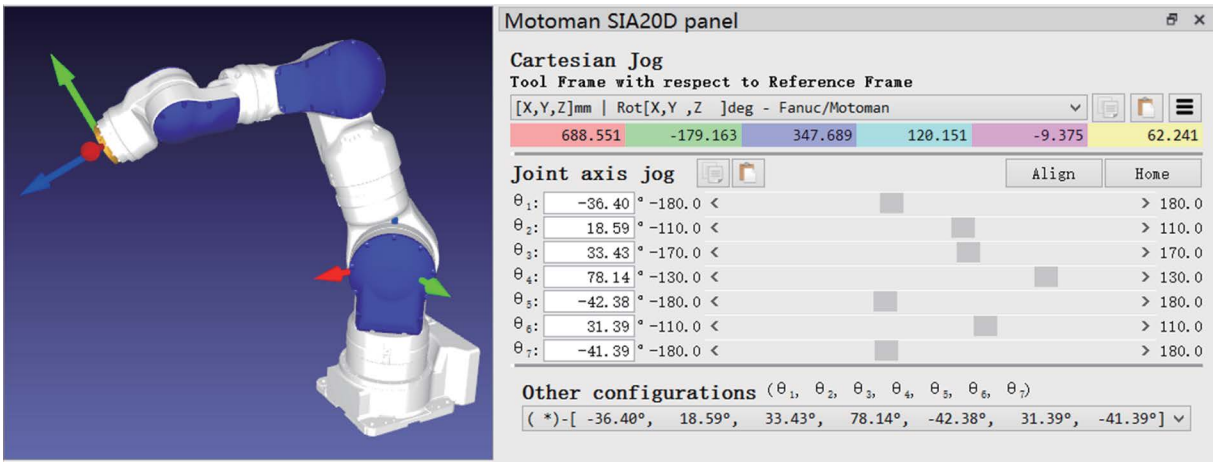

FIGURE 10. Configuration of Motoman SIA20D in RoboDK.

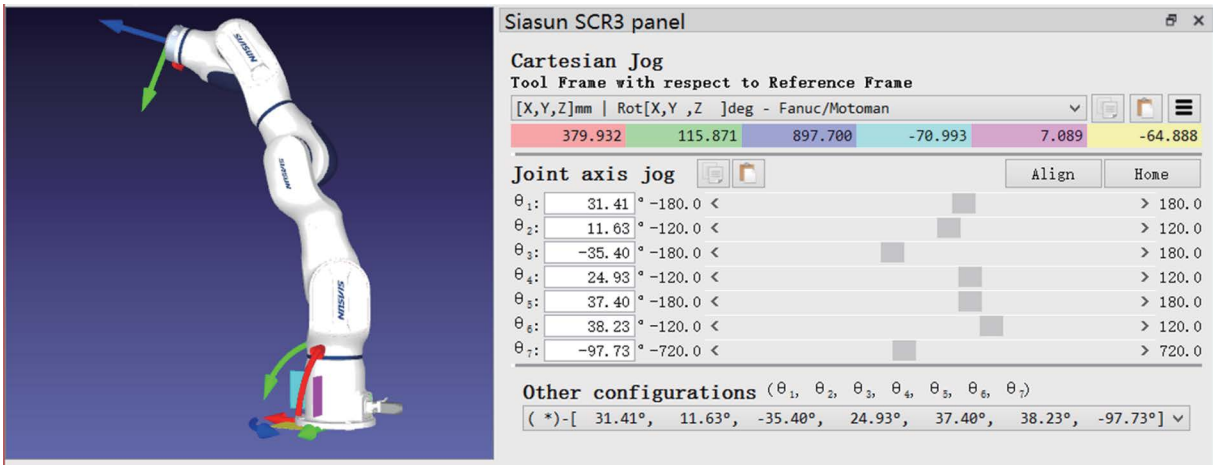

FIGURE 11. Configuration of Siasun SCR3 in RoboDK.

EQ14: $n_3h_1^2 + n_3h_2^2 - n_3h_3^2 - 2n_1h_1h_3 - 2n_2h_2h_3 = (-r_2a_2^2s\alpha_2 - 2r_1a_2d_2 + 2a_2f_1r_3c\alpha_2 - 2r_1a_2f_3c\alpha_2 + r_2d_2^2s\alpha_2 - r_2f_1^2s\alpha_2 + 2r_1f_1f_2s\alpha_2 + r_2f_2^2s\alpha_2 + 2r_3f_2f_3s\alpha_2 - r_2f_3^2s\alpha_2)c\theta_3 + (r_1a_2^2s\alpha_2 - 2r_2a_2d_2 + 2a_2f_2r_3c\alpha_2 - 2a_2f_3r_2c\alpha_2 - r_1d_2^2s\alpha_2 - r_1f_1^2s\alpha_2 - 2r_2f_1f_2s\alpha_2 - 2r_3f_1f_3s\alpha_2 + r_1f_2^2s\alpha_2 + r_1f_3^2s\alpha_2)s\theta_3 + r_3a_2^2c\alpha_2 - 2a_2f_1r_2s\alpha_2 + 2a_2f_2r_1s\alpha_2 - r_3d_2^2c\alpha_2 - 2r_1d_2f_1 - 2r_2d_2f_2 - 2r_3d_2f_3 + r_3f_1^2c\alpha_2 - 2r_1f_1f_3c\alpha_2 + r_3f_2^2calpha_2 - 2r_2f_2f_3c\alpha_2 - r_3f_3^2c\alpha_2$

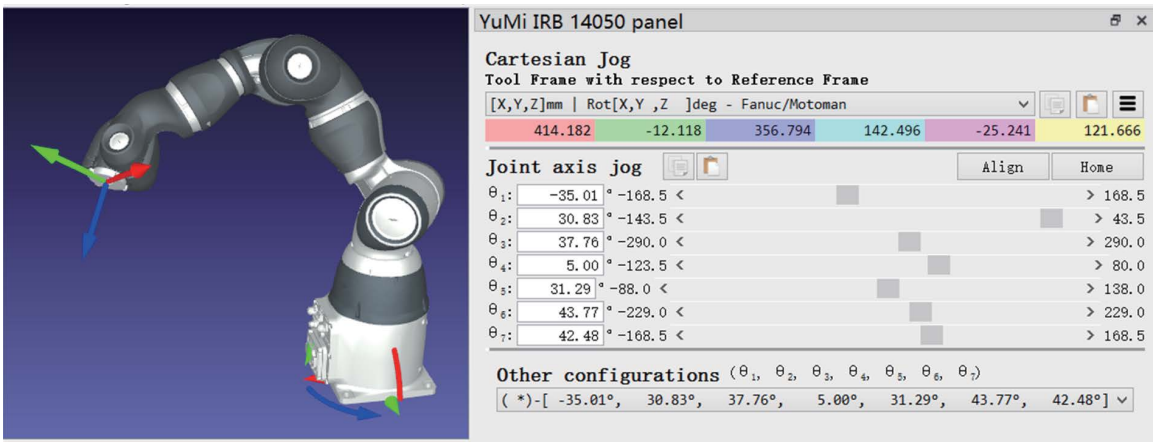

FIGURE 12. Configuration of YuMi IRB 14050 in RoboDK.

TABLE 12. Joint angles corresponding to solutions for Daihen OTC FD-V6Ls.

No.	θ_1(rad)	θ_2(rad)	θ_3(rad)	θ_4(rad)	θ_5(rad)	θ_6(rad)	θ_7(rad)
1	1.4389	-1.1929	2.1147	-3.0887	1.7505	1.0182	-4.8029
2	-1.6064	0.9741	2.1307	0.3639	-1.6215	1.0182	-4.0893
3	-2.0229	-0.9739	-2.0516	-3.1294	-1.6513	1.0182	-5.6162
4	1.0285	0.6419	-2.0564	0.3222	1.5552	1.0182	-6.2577
5	1.0568	0.5975	1.0907	-3.1708	-1.8079	1.0182	-5.9175
6	-2.0206	-0.9713	1.0904	0.2753	1.6658	1.0182	-5.9538
7	1.4613	-1.2181	-1.0162	0.2295	-1.6010	1.0182	-4.4644
8	-1.6154	0.9587	-1.0155	-3.2167	1.7133	1.0182	-4.4272

TABLE 13. Joint angles corresponding to solutions for Franka Emika Panda.

No.	θ_1(rad)	θ_2(rad)	θ_3(rad)	θ_4(rad)	θ_5(rad)	θ_6(rad)	θ_7(rad)
1	0.5506	0.3258	2.2705	-0.8723	-2.0263	-0.0831	0.7459
2	-2.4445	-1.1714	-2.0282	-0.8722	2.0260	-0.0831	-2.7924
3	-2.5910	-0.3259	-0.8710	-0.8723	-2.0263	-0.0831	0.7459
4	0.6969	1.1714	1.1134	-0.8722	2.0257	-0.0831	-2.7928

TABLE 14. Joint angles corresponding to solutions for Kinova Gen3).

No.	θ_1(rad)	θ_2(rad)	θ_3(rad)	θ_4(rad)	θ_5(rad)	θ_6(rad)	θ_7(rad)
1	-3.1017	-1.4330	2.1473	-1.1880	1.2096	0.3666	1.4416
2	0.0628	1.3812	2.1289	1.1440	-1.9054	0.3666	1.4502
3	0.0529	0.8963	-2.0909	-1.1412	-1.2423	0.3666	-1.4264
4	-3.1228	-0.8443	-2.0821	1.1912	1.9389	0.3666	-1.4314
5	3.1231	-0.8968	1.1070	-1.1908	-1.2698	0.3666	-1.4283
6	0.0929	0.8444	0.9989	1.1455	1.9746	0.3666	-1.4368
7	-3.1355	-1.3811	-1.0148	1.1918	-1.8749	0.3666	1.4521
8	0.0957	1.4325	-0.9900	-1.1434	1.1718	0.3666	1.4476

TABLE 15. Joint angles corresponding to solutions for KUKA LBR iiwa 14 R820.

No.	θ_1(rad)	θ_2(rad)	θ_3(rad)	θ_4(rad)	θ_5(rad)	θ_6(rad)	θ_7(rad)
1	0.1433	-1.5373	2.4175	-1.2357	-1.8644	-1.1395	0.4010
2	-2.9983	1.5373	2.4175	1.2356	1.2771	-1.1395	0.4010
3	-1.9541	0.7373	2.0058	-1.2356	1.8643	-1.1395	-1.0209
4	1.1874	-0.7373	2.0058	1.2357	-1.2772	-1.1395	-1.0209
5	1.1874	-0.7373	-1.1358	-1.2357	1.8643	-1.1395	-1.0209
6	-1.9542	0.7373	-1.1358	1.2356	-1.2772	-1.1395	-1.0209
7	-2.9983	1.5373	-0.7241	-1.2357	-1.8644	-1.1395	0.4010
8	0.1433	-1.5373	-0.7241	1.2357	1.2772	-1.1395	0.4010

TABLE 16. Joint angles corresponding to solutions for Motoman SIA20D.

No.	θ_1(rad)	θ_2(rad)	θ_3(rad)	θ_4(rad)	θ_5(rad)	θ_6(rad)	θ_7(rad)
1	2.1664	-0.9994	1.9414	-1.3638	1.2558	1.3479	-0.4706
2	-0.9752	0.9994	1.9414	1.3638	-1.8857	1.3479	-0.4706
3	0.4441	1.1660	-2.1598	1.3639	1.8859	1.3479	-2.0006
4	-2.6973	-1.1659	-2.1598	-1.3641	-1.2561	1.3479	-2.0006
5	2.1664	-0.9994	-1.2002	1.3638	-1.8857	1.3479	-0.4706
6	-0.9752	0.9994	-1.2002	-1.3638	1.2559	1.3479	-0.4706
7	0.4442	1.1659	0.9818	-1.3639	-1.256	1.3479	-2.0006
8	-2.6975	-1.1659	0.9818	1.3638	1.8858	1.3479	-2.0005

TABLE 17. Joint angles corresponding to solutions for Siasun SCR3.

No.	θ_1(rad)	θ_2(rad)	θ_3(rad)	θ_4(rad)	θ_5(rad)	θ_6(rad)	θ_7(rad)
1	2.8995	-0.4915	2.2014	-0.4351	1.3923	0.8672	-1.2432
2	-0.2420	0.4915	2.2014	0.4351	-1.7494	0.8672	-1.2431
3	-2.3600	-0.4068	-1.8576	-0.4351	-1.3923	0.8672	-1.8171
4	0.7816	0.4068	-1.8576	0.4351	1.7492	0.8672	-1.8170
5	0.7816	0.4068	1.2839	-0.4351	-1.3923	0.8672	-1.8171
6	-2.3600	-0.4068	1.2839	0.4351	1.7493	0.8672	-1.8171
7	-0.2421	0.4915	-0.9402	-0.4351	1.3923	0.8672	-1.2432
8	2.8995	-0.4915	-0.9402	0.4351	-1.7493	0.8672	-1.2431

TABLE 18. Joint angles corresponding to solutions for YuMi IRB 14050.

No.	θ_1(rad)	θ_2(rad)	θ_3(rad)	θ_4(rad)	θ_5(rad)	θ_6(rad)	θ_7(rad)
1	-2.9898	-0.4782	2.8192	0.1785	1.1386	0.9639	0.3426
2	2.1200	-1.5338	-1.4147	0.2678	-1.2949	0.9639	1.9458
3	-1.0345	1.2451	1.6548	0.0307	-0.8498	0.9639	1.6506
4	2.4022	-1.3328	1.5134	-2.8082	2.2860	0.9639	1.5191
5	0.0371	0.4404	-0.1786	0.1116	1.0147	0.9639	0.4683
6	2.9085	-0.7837	0.2177	-2.7498	-2.4634	0.9639	0.7337

TABLE 19. D-H parameters for YASKAWA VA1400II manipulator.

Link i	α_i(rad)	a_i(m)	d_i(m)	θ_i(rad)	upper	lower
1	0	0	0	θ_1	-2.9671	2.9671
2	$-\pi/2$	0.15	0	θ_2	-3.0543	2.6180
3	$\pi/2$	0	0.614	θ_3	-1.5708	1.5708
4	$-\pi/2$	0	0	θ_4	-1.2217	2.5831
5	$\pi/2$	-0.20	0.64	θ_5	-2.6180	2.6180
6	$-\pi/2$	0	0	θ_6	-0.7854	3.1416
7	$\pi/2$	-0.03	0	θ_7	-3.4907	3.4907

where,

$h_1 = pc\theta_1 + qs\theta_1 - a_1$

$h_2 = s\alpha_1(s - d_1) - c\alpha_1(ps\theta_1 - qc\theta_1)$

$h_3 = c\alpha_1(s - d_1) + s\alpha_1(ps\theta_1 - qc\theta_1)$

$f_1 = a_3 + g_1c\theta_4 + g_2s\theta_4$

$f_2 = g_3s\alpha_3 - c\alpha_3(g_1s\theta_4 - g_2c\theta_4)$

TABLE 20. Joint angels of 6 solutions for IK of YASKAWA VA1400II.

No.	θ_1(rad)	θ_2(rad)	θ_3(rad)	θ_4(rad)	θ_5(rad)	θ_6(rad)	θ_7(rad)	δx	δy	δz	δR_x	δR_y	δR_z
1	-0.0117	2.1259	0	-0.9708	0.3656	1.8652	0.0091	0.0289	-0.0461	-0.0385	-0.2614	-0.1742	0.0008
2	-0.0072	2.0362	0	-0.8045	-2.7838	-1.7920	3.1267	0.0115	0.0182	0.0102	0.0872	0.0453	-0.0026
3	-0.0172	0.8757	0	1.4157	-2.6505	-0.8133	2.6821	0.0235	0.0099	-0.0278	0.0404	-0.1908	-0.0028
4	-0.0000	0.7854	0	1.5708	0.5236	0.7505	-0.4887	-0.0491	-0.0134	0.0686	-0.0479	0.4561	0.0037
5	-0.3348	0.8914	0.6231	1.5524	-0.2075	0.6807	0.0010	-0.0015	-0.0006	-0.0010	-0.0017	0.0002	0.0008
6	-0.1679	0.8085	0.3008	1.5704	0.2052	0.6807	-0.0023	-0.0022	-0.0007	-0.0006	-0.0010	0.0007	0.0006

$$
\begin{aligned}
f_3 &= d_3 + g_3 c\alpha_3 + s\alpha_3(g_1 s\theta_4 - g_2 c\theta_4)\\
n_1 &= uc\theta_1 + vs\theta_1\\
n_2 &= ws\alpha_1 - c\alpha_1(us\theta_1 - vc\theta_1)\\
n_3 &= wc\alpha_1 + s\alpha_1(us\theta_1 - vc\theta_1)\\
r_1 &= m_1 c\theta_4 + m_2 s\theta_4\\
r_2 &= m_3 s\alpha_3 - c\alpha_3(m_1 s\theta_4 - m_2 c\theta_4)\\
r_3 &= m_3 c\alpha_3 + s\alpha_3(m_1 s\theta_4 - m_2 c\theta_4)\\
p &= p_x - a_7 r_{11} - d_7 r_{12} s\alpha_7 - d_7 r_{13} c\alpha_7\\
q &= p_y - a_7 r_{21} - d_7 r_{22} s\alpha_7 - d_7 r_{23} c\alpha_7\\
s &= p_z - a_7 r_{31} - d_7 r_{32} s\alpha_7 - d_7 r_{33} c\alpha_7\\
g_1 &= a_4 + a_5 c\theta_5 + a_6 c\theta_5 c\theta_6 + d_6 s\alpha_5 s\theta_5 - a_6 c\alpha_5 s\theta_5 s\theta_6\\
g_2 &= d_5 s\alpha_4 + d_6(s\alpha_4 c\alpha_5 + c\alpha_4 s\alpha_5 c\theta_5) + a_6 s\theta_6(s\alpha_4 s\alpha_5 - c\alpha_4 c\alpha_5 c\theta_5) - a_5 c\alpha_4 s\theta_5 - a_6 c\alpha_4 s\theta_5 c\theta_6\\
g_3 &= d_4 + d_5 c\alpha_4 + d_6(c\alpha_4 c\alpha_5 - s\alpha_4 s\alpha_5 c\theta_5) + a_6 s\theta_6(c\alpha_4 s\alpha_5 + s\alpha_4 c\alpha_5 c\theta_5) + a_5 s\alpha_4 s\theta_5 + a_6 s\alpha_4 s\theta_5 c\theta_6\\
u &= r_{12} s\alpha_7 + r_{13} c\alpha_7\\
v &= r_{22} s\alpha_7 + r_{23} c\alpha_7\\
w &= r_{32} s\alpha_7 + r_{33} c\alpha_7\\
m_1 &= s\alpha_6 c\theta_5 s\theta_6 + s\alpha_5 c\alpha_6 s\theta_5 + c\alpha_5 s\alpha_6 s\theta_5 c\theta_6\\
m_2 &= c\alpha_6(s\alpha_4 c\alpha_5 + c\alpha_4 s\alpha_5 c\theta_5) - s\alpha_6 c\theta_6(s\alpha_4 s\alpha_5 - c\alpha_4 c\alpha_5 c\theta_5) - c\alpha_4 s\alpha_6 s\theta_5 s\theta_6\\
m_3 &= c\alpha_6(c\alpha_4 c\alpha_5 - s\alpha_4 s\alpha_5 c\theta_5) - s\alpha_6 c\theta_6(c\alpha_4 s\alpha_5 + s\alpha_4 c\alpha_5 c\theta_5) + s\alpha_4 s\alpha_6 s\theta_5 s\theta_6
\end{aligned}
$$

APPENDIX B
SIMULATION EXPERIMENTS DATA

See Figs. 6–12 and Tables 12–18.

APPENDIX C
IK FOR 7-DOF MANIPULATOR VA1400II

The YASKAWA VA1400II manipulator is a fast and highly flexible welding robot. The D-H parameters are shown in Table 19. A given configuration in degrees is $(0, 45, 0, 90, 30, 43, -28)$ and its corresponding expression in radian is $(0, 0.7854, 0, 1.5708, 0.5236, 0.7505, -0.4887)$. By the forward kinematics, the pose of the end-effector for the given configuration is

$$
\begin{bmatrix}
-0.9872 & -0.1245 & 0.0995 & 1.206\\
-0.0837 & 0.9363 & 0.3410 & -0.01097\\
-0.1356 & 0.3283 & -0.9348 & 0.122\\
0 & 0 & 0 & 1
\end{bmatrix}.
$$

Let the free joint is the third and the sixth joint, respectively. The IK solutions computed by the SDE method are listed in Table 20.

REFERENCES

[1] K.-K. Lee, Y. Komoguchi, and M. Buss, "Multiple obstacles avoidance for kinematically redundant manipulators using Jacobian transpose method," in *Proc. SICE Annu. Conf.*, Sep. 2007, pp. 1070–1076.

[2] T. Yoshikawa, *Foundations of Robotics: Analysis and Control*. Cambridge, MA, USA: MIT Press, 1990.

[3] S. F. M. Assal, K. Watanabe, and K. Izumi, "Neural network-based kinematic inversion of industrial redundant robots using cooperative fuzzy hint for the joint limits avoidance," *IEEE/ASME Trans. Mechatronics*, vol. 11, no. 5, pp. 593–603, Oct. 2006.

[4] A. C. Nearchou, "Solving the inverse kinematics problem of redundant robots operating in complex environments via a modified genetic algorithm," *Mechanism Mach. Theory*, vol. 33, no. 3, pp. 273–292, Apr. 1998.

[5] A. Liégeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. SMC-7, no. 12, pp. 868–871, Dec. 1977.

[6] S.-Y. Oh, D. Orin, and M. Bach, "An inverse kinematic solution for kinematically redundant robot manipulators," *J. Robotic Syst.*, vol. 1, no. 3, pp. 235–249, 1984.

[7] A. Goldenberg, B. Benhabib, and R. Fenton, "A complete generalized solution to the inverse kinematics of robots," *IEEE J. Robot. Autom.*, vol. RA-1, no. 1, pp. 14–20, Mar. 1985.

[8] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *J. Dyn. Syst., Meas., Control*, vol. 108, no. 3, pp. 163–171, Sep. 1986.

[9] J. Baillieul, "Kinematic programming alternatives for redundant manipulators," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 2, Jun. 1985, pp. 722–728.

[10] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano, "Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy," *Int. J. Robot. Res.*, vol. 10, no. 4, pp. 410–425, 1991.

[11] G. Tevatia and S. Schaal, "Inverse kinematics for humanoid robots," in *Proc. IEEE Int. Conf. Robot. Autom. Symp.*, vol. 1, Apr. 2000, pp. 294–299.

[12] J. Ratajczak, "Design of inverse kinematics algorithms: Extended Jacobian approximation of the dynamically consistent Jacobian inverse," *Arch. Control Sci.*, vol. 25, no. 1, pp. 35–50, Mar. 2015.

[13] H. Simas and R. Di Gregorio, "A technique based on adaptive extended Jacobians for improving the robustness of the inverse numerical kinematics of redundant robots," *J. Mech. Robot.*, vol. 11, no. 2, pp. 1–56, Apr. 2019.

[14] J. Leger and J. Angeles, "A redundancy-resolution algorithm for five-degree-of-freedom tasks via sequential quadratic programming," in *Proc. TrC-IFToMM Symp. Theory Mach. Mech.*, Izmir, Turkey, 2015, pp. 1–8.

[15] L. Jiang, S. Lu, Y. Gu, and J. Zhao, "Time-jerk optimal trajectory planning for a 7-DOF redundant robot using the sequential quadratic programming method," in *Intelligent Robotics and Applications*, Y. Huang, H. Wu, H. Liu, and Z. Yin, Eds. Cham, Switzerland: Springer, 2017, pp. 343–353.

[16] H. Lyu, X. Song, D. Dai, J. Li, and Z. Li, "Time-optimal and energy-efficient trajectory generation for robot manipulator with kinematic constraints," in *Proc. 13th IEEE Conf. Autom. Sci. Eng. (CASE)*, Aug. 2017, pp. 503–508.

[17] Y. Taniai and T. Naniwa, "Joint trajectory planning based on minimum Euclidean distance of joint angles of a seven-degrees-of-freedom manipulator for a sequential reaching task," *J. Adv. Comput. Intell. Intell. Inform.*, vol. 23, no. 6, pp. 997–1003, Nov. 2019.

[18] T. Y. Huang and W. J. Chen, "Application of sequential quadratic programming for obtaining optimal obstacle avoidance posture and end position of a 7-axis robotic manipulator," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 1113, no. 1, Mar. 2021, Art. no. 012002.

[19] L. Sciavicco and B. Siciliano, "A dynamic solution to the inverse kinematic problem for redundant manipulators," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 4, Mar./Apr. 1987, pp. 1081–1087.
[20] O. Egeland, "Task-space tracking with redundant manipulators," *IEEE J. Robot. Autom.*, vol. RA-3, no. 5, pp. 471–475, Oct. 1987.
[21] H. Seraji, "Configuration control of redundant manipulators: Theory and implementation," *IEEE Trans. Robot. Autom.*, vol. 5, no. 4, pp. 472–490, Aug. 1989.
[22] Y. Cao, Y. Gan, X. Dai, and J. Cao, "Inverse kinematics of 7-DOF redundant manipulators with arbitrary offsets based on augmented Jacobian," in *Proc. 33rd Youth Academic Annu. Conf. Chin. Assoc. Autom. (YAC)*, May 2018, pp. 33–42.
[23] R. V. Dubey, J. A. Euler, and S. M. Babcock, "An efficient gradient projection optimization scheme for a seven-degree-of-freedom redundant robot with spherical wrist," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 1, Apr. 1988, pp. 28–36.
[24] R. V. Dubey, J. A. Euler, and S. M. Babcock, "Real-time implementation of an optimization scheme for seven-degree-of-freedom redundant manipulators," *IEEE Trans. Robot. Autom.*, vol. 7, no. 5, pp. 579–588, Oct. 1991.
[25] H. Zghal, R. V. Dubey, and J. A. Euler, "Efficient gradient projection optimization for manipulators with multiple degrees of redundancy," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, May 1990, pp. 1006–1011.
[26] T. F. Chan and R. V. Dubey, "A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators," *IEEE Trans. Robot. Autom.*, vol. 11, no. 2, pp. 286–292, Apr. 1995.
[27] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, no. 1, pp. 93–101, Jan. 1986.
[28] A. S. Deo and I. D. Walker, "Adaptive non-linear least squares for inverse kinematics," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 1, May 1993, pp. 186–193.
[29] F. Caccavale, P. Chiacchio, S. Chiaverini, and B. Siciliano, "Experiments of kinematic control on a redundant robot manipulator with nonspherical wrist," *Lab. Robot. Autom.*, vol. 8, no. 1, pp. 25–36, 1996.
[30] S. R. Buss and J.-S. Kim, "Selectively damped least squares for inverse kinematics," *J. Graph. Tools*, vol. 10, no. 3, pp. 37–49, Jan. 2005.
[31] L. M. Phuoc, P. Martinet, S. Lee, and H. Kim, "Damped least square based genetic algorithm with Ggaussian distribution of damping factor for singularity-robust inverse kinematics," *J. Mech. Sci. Technol.*, vol. 22, no. 7, pp. 1330–1338, Jul. 2008.
[32] Y. Yang, G. Peng, Y. Wang, and H. Zhang, "A new solution for inverse kinematics of 7-DOF manipulator based on genetic algorithm," in *Proc. IEEE Int. Conf. Autom. Logistics*, Aug. 2007, pp. 1947–1951.
[33] T. Sugihara, "Solvability-unconcerned inverse kinematics by the Levenberg–Marquardt method," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 984–991, Oct. 2011.
[34] A. Hemami, "A more general closed-form solution to the inverse kinematics of mechanical arms," *Adv. Robot.*, vol. 2, no. 4, pp. 315–325, Jan. 1987.
[35] M. Shimizu, H. Kakuya, W. K. Yoon, K. Kitagaki, and K. Kosuge, "Analytical inverse kinematic computation for 7-DOF redundant manipulators with joint limits and its application to redundancy resolution," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1131–1142, Oct. 2008.
[36] G. K. Singh and J. Claassens, "An analytical solution for the inverse kinematics of a redundant 7DoF manipulator with link offsets," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 2976–2982.
[37] Y. Wang, L. Sun, W. Yan, and J. Liu, "An analytic and optimal inverse kinematic solution for a 7-DOF space manipulator," *Robot*, vol. 36, no. 5, pp. 592–599, 2014.
[38] J. K. Parker, A. R. Khoogar, and D. E. Goldberg, "Inverse kinematics of redundant robots using genetic algorithms," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 1, May 1989, pp. 271–276.
[39] A. C. Nearchou and N. A. Aspragathos, "Application of genetic algorithms to point-to-point motion of redundant manipulators," *Mechanism Mach. Theory*, vol. 31, no. 3, pp. 261–270, Apr. 1996.
[40] S. Tabandeh, W. W. Melek, and C. M. Clark, "An adaptive niching genetic algorithm approach for generating multiple solutions of serial manipulator inverse kinematics with applications to modular robots," *Robotica*, vol. 28, no. 4, pp. 493–507, Jul. 2010.
[41] R. Köker, "A neuro-simulated annealing approach to the inverse kinematics solution of redundant robotic manipulators," *Eng. Comput.*, vol. 29, no. 4, pp. 507–515, Oct. 2013.
[42] A. Srivastava and A. Kumar, "An optimization technique to solve the inverse kinematics of robot manipulator," in *Proc. Int. Conf. Manuf. Excellence*, 2013, pp. 77–83.
[43] F. Yin, Y.-N. Wang, Y. Yang, and Y.-M. Yang, "Inverse kinematics solution for robot manipulator based on neural network under joint subspace," *Int. J. Comput. Commun. Control*, vol. 7, no. 3, pp. 459–472, Mar. 2012.
[44] H.-C. Huang, C.-P. Chen, and P.-R. Wang, "Particle swarm optimization for solving the inverse kinematics of 7-DOF robotic manipulators," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2012, pp. 3105–3110.
[45] S. Starke, N. Hendrich, S. Magg, and J. Zhang, "An efficient hybridization of genetic algorithms and particle swarm optimization for inverse kinematics," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2016, pp. 1782–1789.
[46] L. Yiyang, J. Xi, B. Hongfei, W. Zhining, and S. Liangliang, "A general robot inverse kinematics solution method based on improved PSO algorithm," *IEEE Access*, vol. 9, pp. 32341–32350, 2021.
[47] X. Fu, Y. Shang, Z. Su, and G. Yan, "Study on the inverse kinematics of redundant scanning robot based on RBF network," *China Mech. Eng.*, vol. 17, no. 17, pp. 1765–1768, 2006.
[48] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Germany: Springer, 2016.
[49] J. Fang, T. Mei, J. Chen, and J. Zhao, "An iteration method for inverse kinematics of redundancy robot," in *Proc. IEEE Int. Conf. Mechatronics Autom.*, Aug. 2014, pp. 1005–1010.
[50] Z. Yang, Z. Chen, C. Zhao, and Y. Li, "Inverse kinematics algorithm of 7-DOF manipulator based on self-motion," *J. Mech. Eng.*, vol. 55, no. 23, pp. 75–82, 2019.
[51] M. H. Korayem and A. M. Shafei, "Motion equation of nonholonomic wheeled mobile robotic manipulator with revolute–prismatic joints using recursive Gibbs–Appell formulation," *Appl. Math. Model.*, vol. 39, nos. 5–6, pp. 1701–1716, Mar. 2015.
[52] M. Saura, A. Celdran, D. Dopico, and J. Cuadrado, "Computational structural analysis of planar multibody systems with lower and higher kinematic pairs," *Mechanism Mach. Theory*, vol. 71, pp. 79–92, Jan. 2014.
[53] A. M. Shafei and H. R. Shafei, "Dynamic modeling of tree-type robotic systems by combining 3×3 rotation and 4×4 transformation matrices," *Multibody Syst. Dyn.*, vol. 44, no. 4, pp. 367–395, 2018.
[54] A. Müller, "Recursive higher-order constraints for linkages with lower kinematic pairs," *Mechanism Mach. Theory*, vol. 100, pp. 33–43, Jun. 2016.
[55] M. H. Korayem, H. Rahimi, and A. Nikoobin, "Path planning of mobile elastic robotic arms by indirect approach of optimal control," *J. Adv. Robot. Syst.*, vol. 8, no. 1, pp. 10–20, 2011.
[56] M. H. Korayem, A. M. Shafei, and S. F. Dehkordi, "Systematic modeling of a chain of N-flexible link manipulators connected by revolute–prismatic joints using recursive Gibbs–Appell formulation," *Arch. Appl. Mech.*, vol. 84, no. 2, pp. 187–206, Feb. 2014.
[57] L.-W. Tsai, *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*. New York, NY, USA: Wiley, 1999.
[58] M. Raghavan and B. Roth, "Inverse kinematics of the general 6R manipulator and related linkages," *J. Mech. Des.*, vol. 115, no. 3, pp. 502–508, Sep. 1993.
[59] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *J. Appl. Mech.*, vol. 22, no. 2, pp. 215–221, Jun. 1955.
[60] K. J. Waldron, "A study of overconstrained linkage geometry by solution of closure equations—Part 1. Method of study," *Mechanism Mach. Theory*, vol. 8, no. 1, pp. 95–104, Mar. 1973.
[61] R. P. Paul, *Robot Manipulators: Mathematics, Programming and Control*. London, U.K.: MIT Press, 1981.
[62] W. Wiedmeyer, P. Altoe, J. Auberle, C. Ledermann, and T. Kroger, "A real-time-capable closed-form multi-objective redundancy resolution scheme for seven-DoF serial manipulators," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 431–438, Apr. 2021.
[63] M. Raghaven and B. Roth, "Kinematic analysis of the 6R manipulator of general geometry," in *Proc. Int. Symp. Robot. Res.* Cambridge, MA, USA: MIT Press, 1991, pp. 263–269.
[64] J. H. Wilkinson, "The evaluation of the zeros of ill-conditioned polynomials. Part I," *Numerische Math.*, vol. 1, no. 1, pp. 150–166, Dec. 1959.
[65] J. H. Wilkinson, "The evaluation of the zeros of ill-conditioned polynomials. Part II," *Numerische Math.*, vol. 1, no. 1, pp. 167–180, Dec. 1959.
[66] I. C. Gohberg, P. Lancaster, and L. Rodman, *Matrix Polynomials*. New York, NY, USA: Academic, 1982.
[67] D. Manocha and J. F. Canny, "Real time inverse kinematics for general 6R manipulators," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 1, May 1992, pp. 383–389.

[68] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*. Oxford, U.K.: Clarendon Press, 1965.
[69] P. Corke, *Robotics, Vision and Control*, vol. 118. Cham, Switzerland: Springer, 2017.
[70] S. Kucuk and Z. Bingul, "Inverse kinematics solutions for industrial robot manipulators with offset wrists," *Appl. Math. Model.*, vol. 38, nos. 7–8, pp. 1983–1999, 2014.
[71] W. H. Lai, S. L. Kek, and K. G. Tay, "Solving nonlinear least squares problem using Gauss-Newton method," *Int. J. Innov. Sci., Eng. Technol.*, vol. 4, no. 1, pp. 258–262, 2017.
[72] C. Kanzow, N. Yamashita, and M. Fukushima, "Levenberg–Marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints," *J. Comput. Appl. Math.*, vol. 172, no. 2, pp. 375–397, Dec. 2004.
[73] C.-C. Yih and P. I. Po, "Near-optimal motion planning for nonholonomic systems with state/input constraints via quasi-Newton method," in *Proc. Int. Conf. Robot. Automat.*, vol. 2, Apr. 1997, pp. 1430–1435.
[74] W. Sun and Y.-X. Yuan, *Sequential Quadratic Programming*. Boston, MA, USA: Springer, 2006, pp. 523–560.
[75] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Upper Saddle River, NJ, USA: Pearson, 2005.

SHUXIN XIE was born in 1980. He received the B.E. degree from the School of Material Engineering and the M.S. degree from the School of Mechanical and Electrical Engineering, Soochow University, Suzhou, China, in 2003 and 2010, respectively. He is currently pursuing the Ph.D. degree in intelligent robot technology. His research interests include robot vision and motion planning.

LINING SUN was born in 1964. He received the Ph.D. degree from the Harbin Institute of Technology, Harbin, China, in 1993. He is currently a Professor with Soochow University. His research interests include industrial robot and intelligent manufacturing.

GUODONG CHEN was born in 1983. He received the Ph.D. degree from the Harbin Institute of Technology, Harbin, China, in 2011. He is currently an Associate Professor with Soochow University. His research interests include robot vision and intelligent industrial robots.

ZHENHUA WANG was born in 1974. He received Ph.D. degree from the Harbin Institute of Technology, Harbin, China. He is young and middle-aged academic leaders of the "Blue Project" in Jiangsu Province. His research interests include industrial robots and intelligent automation equipment.

ZIXIANG WANG was born in 1997. He received the B.E. degree from Yangzhou University, China, in 2019. He is currently pursuing the M.E. degree with the School of Mechanical and Electrical Engineering, Soochow University. His research interests include robot vision and robotics.