Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 8174924, 14 pages
https://doi.org/10.1155/2020/8174924

Research Article

Two Optimized General Methods for Inverse Kinematics of 6R Robots Based on Machine Learning

Xiaoqi Wang, Jianfu Cao, Lerui Chen, and Heyu Hu

Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, Shanxi, China

Correspondence should be addressed to Xiaoqi Wang; wangxiaoqi1031@163.com

Received 9 October 2019; Revised 15 December 2019; Accepted 21 December 2019; Published 21 January 2020

Academic Editor: Luis Payá

For the 6R robot, there is no analytical solution for some configurations, so it is necessary to analyse inverse kinematics (IK) by the general solution method, which cannot achieve high precision and high speed as the analytical solution. With the expansion of application fields and the complexity of application scenarios, some robots with special configuration have become the research hotspot, and more high-speed and high-precision general algorithms are still being explored and studied. The present paper optimized two general solutions. Elimination is a numerical solution, which has high accuracy, but the solution process is complex and time-consuming. The present paper optimized the elimination method, derived the final matrix expression directly through complex coefficient extraction and simplifying operation, and realized one-step solution. The solving speed was reduced to 15% of the original, and the integrity of the method was supplemented. This paper proposed a new optimization method for the Gaussian damped least-squares method, in which the variable step-size coefficient is introduced and the machine learning method is used for the research. It was proved that, on the basis of guaranteeing the stability of motion, the average number of iterations can be effectively reduced and was only 4-5 times, effectively improving the solving speed.

1. Introduction

For 6R robots, most configurations satisfy the Pieper criterion; that is, the adjacent three axes are parallel or intersect at one point, and there is analytical solution, which can be solved with high speed and high precision. Part of the configuration does not meet the Pieper criterion and has no analytical solution due to special operating requirements, so it needs to be solved by the general solution method. To meet the requirements of smoothness, higher speed, and higher precision, some new general methods and optimization methods are still being explored and studied. Among numerous methods, the elimination method is time-consuming and complex, but it has the highest precision and can get all the solutions under the same terminal posture. This paper optimized the elimination method to improve the solving speed and make the algorithm more complete. On the basis of the Gaussian damped least-squares method, this paper introduced the variable step-size coefficient, studied the influencing factors of the coefficient, and determined the coefficient through the regression algorithm of machine learning. The speed is superior to other iterative optimization methods. The average number of iterations is only 4 to 5 times, which effectively improves the solving speed and basically approximates the speed of analytical solutions. For inverse kinematic problems, there are three main categories of algorithms: iterative algorithm, numerical and geometric methods, and soft computing methods.

The Jacobian matrix pseudoinverse method is proposed in [1], and it is the most commonly used method of robot inverse kinematics. The algorithm is unstable when the robot approaches the singular configuration. Four Jacobian-based methods of solving the inverse kinematics are evaluated in [2]. A novel formula for calculating the pseudoinverse is introduced for a class of redundant robots in [3]. The optimal approximation of the Jacobian pseudoinverse algorithm by the extended Jacobian algorithm is described in [4]. On the basis of the pseudoinverse method, the damped least-squares (DLS) method [5, 6] introduces the damping factor to balance the solution accuracy and the joint velocity near the singular configuration, so as to make the motion more stable. The DLS method is used to analyse the inverse kinematics of the 7R 6-DOF robots with the hollow nonspherical wrist in [7]. Based on different damping factor selection strategies, the selectively damped least-squares (SDLS) method is proposed in [8]. The Gaussian damped least-squares (GDLS) method is described in [9]. Based on the Gaussian distribution of the damping factor, the Gaussian function is applied to determine the damping factor, which can continuously transition from undamped to damped, making the motion more stable and reducing errors. A deeply learnt damped least-squares method is proposed for solving IK of the spatial snake-like robot in [10]. A deep network is built for prediction of the unique damping factor required for each target point. A hierarchical iterative inverse kinematic algorithm is proposed in [11].

An elimination method for inverse kinematics of a 6R robot is proposed in [12, 13], which simplifies the inverse kinematics into a 16-degree polynomial problem. Then, the matrix eigenvalue decomposition is used to replace the solving equation of higher degree. Inverse iterations of the Jacobian matrix and the elimination method are combined in [14] to study the 6-DOF manipulator and polymers. Literature [15] optimized the elimination method and verified it with an example. A new method for inverse kinematics while considering jerk limits is proposed in [16] to efficiently handle acceleration and jerk constraints. A generalized solution for a subproblem of inverse kinematics based on an exponential formula product is proposed in [16]. Conformal geometric algebra is used to develop analytical inverse kinematic solutions for the KUKA Agilus robot and the UR5 robot in [17]. In [18], to avoid the problem of singularity, the product of exponentials method based on screw theory is employed for kinematic modeling. In addition, the inverse kinematics is solved by adopting analytical, geometric, and algebraic methods combined with the Paden–Kahan subproblem. Various optimized numerical and iterative methods failed to achieve the same performance as analytical solutions.

Various soft computing methods [19–26] based on the artificial neural network and genetic algorithm are also the research hotspots in recent years. A comparative study between different soft computing-based methods (artificial neural network, adaptive neuro-fuzzy inference system, and genetic algorithms) applied to the problem of inverse kinematics is presented in [19]. A genetic algorithm based on extreme learning machine and sequential mutation is presented in [20]. A hybrid Taguchi deoxyribonucleic acid (DNA) swarm intelligence for solving the inverse kinematic redundancy problem is presented in [21]. In [22], an online adaptive strategy based on the Lyapunov stability theory is presented to solve the inverse kinematics of redundant manipulators. The neural network and genetic algorithms are used together to solve the inverse kinematics of a six-joint manipulator to minimize the error at the end effector in [23]. A neural-network committee machine (NNCM) is designed in [24] to solve the inverse kinematics of a 6R redundant manipulator to improve the precision of the solution. An approach for solving the inverse kinematics of manipulator robots based on soft computing algorithms is introduced in [25]. In [26], the inclusion of the current configuration of joint angles in the ANN significantly increased the accuracy of ANN estimation of the joint angle output. Various soft computing methods can achieve accuracy up to the micron level, but the solution speed is still a problem.

Some machine learning regression models are used in this paper. The decision tree (DT) [27] is a prediction model to establish a mapping between object attributes and object values. It has fast speed and easily visualizes the model. The K-nearest neighbor (KNN) [28] algorithm is based on the limit theorem. Decision-making depends on a very small number of nearest samples. The random forest (RF) [29] integrates the set of decision trees with control variance by using the idea of bagging. The gradient boosting decision tree (GBDT) [30] generates a weak classifier in each iteration through multiple iterations. Each classifier trains on the basis of the residual of the previous one. The bagging method constructs a series of predictive functions and combines them into a predictive function. ExtraTreeRegressor (ETR) is similar to the RF and consists of many decision trees. Each decision tree is obtained by using all training samples, and the bifurcation values are obtained completely randomly.

2. Optimized Gaussian Damped Least-Squares Method

2.1. Gaussian Damped Least-Squares Method. The Jacobian matrix pseudoinverse method, which was proposed by Whiteney [1], is most frequently used to solve inverse kinematics. Based on the known speed of the end effector, the robot joint speed is determined as

$$\dot{\theta} = J^{+}\dot{X}, \tag{1}$$

where J^{+} denotes the pseudoinversion of the Jacobian matrix. It can minimize the joint speed norm $\dot{\theta}^{2}$ and tracking error $\dot{X} - J\dot{\theta}^{2}$ by adopting the Jacobian matrix pseudoinverse method. However, this method is unstable near the singular points. The Jacobian matrix is $SV\,D$ decomposed as follows:

$$J = USV^{T}, \tag{2}$$

where S is a diagonal matrix comprising singular values of J, which are arranged in the descending order as $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_m$, and then the joint speed is

$$\dot{\theta} = J^{+}\dot{X} = \sum_{i=1}^{6} \frac{1}{\sigma_i} VU^{T}\dot{X}. \tag{3}$$

When the robot is close to the singular configuration, the minimum singular value is approximate to 0 and joint speed is infinitely great, which is the reason why the pseudoinverse method is unstable. The damped least-squares (DLS) method incorporates the damping factor λ on this basis to minimize $\dot{X} - J\dot{\theta}^{2} + \lambda^{2}\dot{\theta}^{*2}$, and as a result, as the robot approaches the singular configuration, the solving accuracy and joint speed are balanced, and then the joint speed is

$$\dot{\theta}^* = \sum_{i=1}^{6} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} V U^T \dot{X}. \quad (4)$$

Based on the DLS method, according to different strategies for damping factor selection, many methods have been developed. The Gaussian damped least-squares (GDLS) method allows for Gaussian distribution characteristic of the damping factor; the damping factor is determined through the following Gaussian function:

$$\lambda_i = \lambda_{\max} \cdot e^{-(\sigma_i/\varepsilon)^2}, \quad (5)$$

where $\lambda_{\max}$ is the maximum value of the damping factor and ε is a scalar quantity indicating the region of singularities. In comparison with other methods which take constant or piecewise function as the damping factor, the GDLS method includes a better strategy for selecting the damping factor. On the one hand, when the robot approaches the singular configuration, the joint speed transitions from the undamped into the damped state in a continuous way so that the motion is steadier. On the other hand, each singular value corresponds to a damping factor, so the damping only acts on the singular vectors, thus avoiding unnecessary damping and reducing the error. $\lambda_{\max}, \varepsilon$ can be determined through the genetic algorithm, the details of which are given in literature [9].

2.2. Variable Step-Size Coefficient

2.2.1. Concept and Distribution of Variable Step-Size Coefficient. In order to elevate the solving speed, the variable step-size coefficient k and optimal variable step-size coefficient k_optimal are introduced based on the GDLS method.

Definition. In the iteration process, $\theta_{\text{cur}} = \theta_{\text{cur}} + \mathrm{d}\theta$ is changed into $\theta_{\text{cur}} = \theta_{\text{cur}} + k\mathrm{d}\theta$ so that the iteration proceeds rapidly towards convergence and k is called the variable step-size coefficient. As shown in Figure 1, the number of iterations changes with k. When k is a certain value, the number of iterations is minimized, and the k value is defined as the optimal variable step-size coefficient k_optimal.

It can be seen from Figure 1 that when k is close to k_optimal, the number of iterations presents a continuous k-dependent change and gradually decreases. Hence, the number of iterations will approach the minimum value as long as k is close to k_optimal.

In order to investigate k_optimal distribution and the range of the minimum number of iterations, 10^4 poses are randomly selected, and they satisfy the following: D-H parameters are arbitrary, the iterative convergence error is $E_{rr} = 10^{-4}$, and initial and target joint angles are controlled within $|\theta_{\text{ini}} - \theta_{\text{end}}| \le 5^\circ$. The GDLS method is used to solve inverse solution, and k_optimal is obtained each time. Based on the statistics of data, the probability distribution of k_optimal and minimum number of iterations are shown in Figure 2.

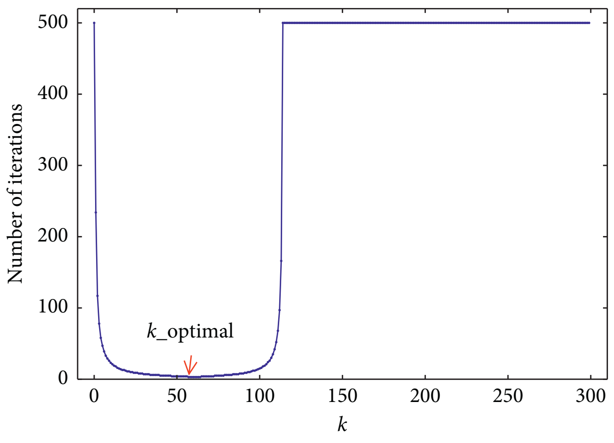


Figure 1: Number of iterations varying with k.

As shown in Figure 2, k_optimal is mainly distributed within [20, 120]. Statistical data show that, for 95.4% of the poses, the minimum number of iterations corresponding to k_optimal can be controlled under 20.

2.2.2. Influence Factors of k_optimal. Through the analysis of the iterative process, it is estimated that the possible influence factors of k_optimal include the iterative convergence error E_{rr}, the difference e between the current pose T_{cur} and the target pose T_{end}, the initial iterative value θ_i, and singularity.

The difference between the current pose T_{cur} and the target pose T_{end} is $e = f(e_x, e_y, e_z, e_\phi, e_\theta, e_\psi)$, where e_x, e_y, e_z belong to position errors, e_ϕ, e_θ, e_ψ belong to Euler angular direction errors, and $e = \sum_{i=1}^{6} |e_i|$.

Generally, the minimum singular value σ_m of the Jacobian matrix or the operand $w = \sqrt{\det|JJ^T|} = \sigma_1\sigma_1 \ldots \sigma_m$ is adopted to characterize the singularity.

When the other conditions remain unchanged, the changes of k_optimal and minimum number of iterations with e, E_{rr}, θ_i, σ_m, and w are worked out, as shown in Figure 3.

It can be seen from Figure 3 that when $E_{rr} \le 0.01$, E_{rr} does not affect k_optimal. When e is small, the effect on k_optimal is minor. Within a certain range, the effect on k_optimal can be great. When e exceeds a certain value, that is, when the current pose and target pose are quite different, the iteration will not converge. The initial iteration value θ_i has a great influence on k_optimal, and different joint angles can exert different effects on k_optimal. The minimum singular value and operand have no direct influence on k_optimal. On the whole, the main factors influencing k_optimal are θ_i and e.

2.3. Determination of k_optimal Based on Machine Learning. The analysis of influence factors of k_optimal shows that the goal is to determine k_optimal through θ_i and e. Though all kinds of methods are tried, it is very difficult to directly obtain their function expressions. Therefore, with $\theta_1 \sim \theta_6$ and

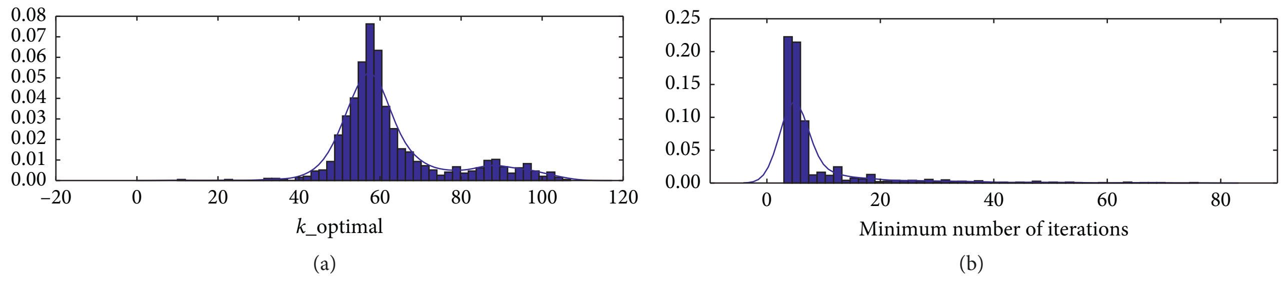


Figure 2: Distribution of k_optimal (a) and minimum iteration number (b).

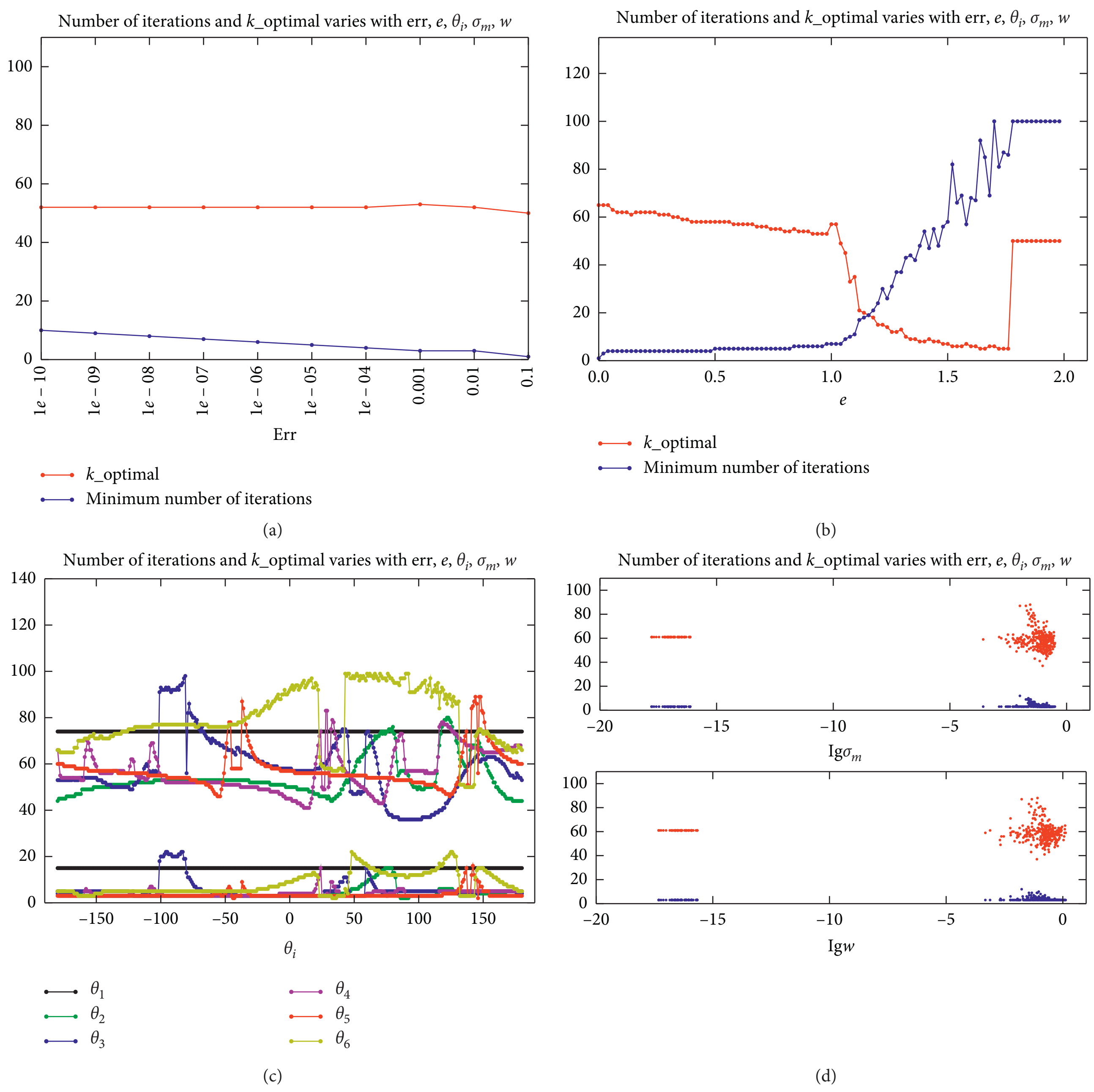


Figure 3: Influencing factors of k_optimal.

e as the input and k_optimal as the output, the machine learning regression method is selected.

10^5 poses are randomly selected. When other conditions are unchanged, $\theta_1 \sim \theta_6$ and e are changed to obtain the corresponding k_optimal which acts as original data for training and testing. Three frequently used statistical indicators, that is, root mean square error (RMSE), coefficient of determination (R^2), and mean absolute error (MAE), are used to evaluate the model performance:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_{i,m} - Y_{i,e}\right)^2}, \tag{6}$$

$$R^2 = \frac{\sum_{i=1}^{n}\left(Y_{i,m} - Y_{i,e}\right)^2}{\sum_{i=1}^{n}\left(Y_{i,m} - \overline{Y}_{i,m}\right)^2}, \tag{7}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|Y_{i,m} - Y_{i,e}\right|. \tag{8}$$

The regression models are frequently adopted in machine learning; 6 effective models are selected and compared, and the performance analysis is shown in Table 1. A total of 1,000 groups of data are selected by adopting four methods to generate the regressive fitting curves in Figure 4.

As shown in Table 1, training effects of the decision tree and ExtraTree on the training samples are good. The effects of the random forest and bagging on test samples are better. It can be intuitively seen from Figure 4 that the random forest and bagging generate better fitting results. Through a comprehensive comparison, the random forest brings the best result. Meanwhile, the trained random forest model is used to predict k_optimal, and the average prediction time is only 0.9735 μs, which can be neglected in comparison with solution time.

2.4. Flowchart of Optimized Method.

The flowchart of the optimized GDLS method is shown in Figure 5.

2.5. Case of Optimized GDLS Method.

A randomly selected 6R robot is taken as an example for the test, and D-H parameters are listed in Table 2. The optimized method is named "E-GDLS," and inverse solution is solved according to the flow in Figure 5. $\lambda_{\max} = 0.09$, $\varepsilon = 0.05$, and the iterative convergence error is set as $E_{rr} = 0.0001$.

In Table 3, IK results of 4 random data points of this method in the robot working space are summarized, and the number of iterations of the method before and after optimization is compared, as shown in Figure 6.

The test results of four arbitrary data points show that the E-GDLS method can effectively predict k_optimal; the solving accuracy meets the requirements; the number of iterations is all within 10. The GDLS method needs about 200 iterations to converge to the error threshold in the same situation. This method can greatly reduce the number of iterations to improve the solving speed.

Three trajectories are randomly selected in this study. Trajectory 1 only displays the position increment, while trajectory 2 displays increments of both the position and the Euler angular direction. Firstly, the number of iterations before and after the optimization is compared, and accuracy and stability of the optimized method are analysed through the changes of the spatial joint angle and trajectory tracking error. Trajectory 3 represents the status when the initial pose is approximate to a singular configuration, and it is mainly adopted to analyse the controlling effect of the optimized method on the joint speed relative to the Jacobian matrix pseudoinverse method.

Trajectory 1: the initial configuration is $\theta = (0.715, -2.914, 1.919, 2.897, -2.007, 2.059)^T$ rad, and the increment is $\Delta p = (0.31, -0.72, 0.43)^T$ m.

Trajectory 2. The initial configuration is $\theta = (-1.658, 0.244, 2.705, -0.925, 0.803, -2.251)^T$ rad, and the increment is $\Delta p = (-0.12, 0.57, 0.79, 1.8675, 0.5236, 2.304)^T$ m.

Trajectory 3. The initial configuration is $\theta = (0.734, 3.1708, 0.028, -3.184, 0.059, -0.044)^T$ rad, and the increment is $\Delta p = (0.19, -0.45, 0.33)^T$ m.

It can be seen from Figures 7 and 8 that the number of iterations of the optimized method in this paper is obviously reduced compared with that of the GDLS method, so it can effectively improve solving speed. The joint angle in the joint space changes steadily with a minor trajectory tracking error, so stability and accuracy of the motion are guaranteed. As shown in Figure 9, when the robot is approaching the singular configuration, compared with the Jacobian matrix pseudoinverse method, the optimized method can still effectively control joint speed just as other damping methods.

In order to further analyse the solving performances of different methods, 10^6 poses are randomly selected, the iterative convergence error is set as $E_{rr} = 0.0001$, the maximum number of iterations is $i\,\mathrm{limit} = 500$, and the difference value between the initial pose and the target pose is controlled within a certain range. The test environment is 3.2 GHz CPU, 8 GB RAM, and python. The Jacobian matrix pseudoinverse methods J^T, DLS, and GDLS and the optimized method E-GDLS in this paper are adopted to solve the inverse solution, respectively. The average number of iterations, the average solving time, and the proportion of failed convergence due to exceeding maximum number are calculated for each method. The results are listed in Table 4.

From Table 4, it can be found that the optimized method in this paper can effectively reduce the number of iterations and the average solving time and make iterative convergence faster so as to improve the method's convergence performance.

Table 1: Performance comparison of regression models.

Model	Training			Testing		
	R^2	RMSE	MAE	R^2	RMSE	MAE
Decision tree	1.000	0.000	0.000	0.931	0.721	0.463
K-nearest neighbor	0.848	1.100	0.803	0.796	1.237	0.919
Random forest	0.990	0.280	0.172	0.977	0.412	0.216
Gradient boosting	0.950	0.631	0.480	0.951	0.604	0.431
Bagging	0.988	0.309	0.185	0.975	0.432	0.245
ExtraTree	1.000	0.000	0.000	0.818	1.168	0.766

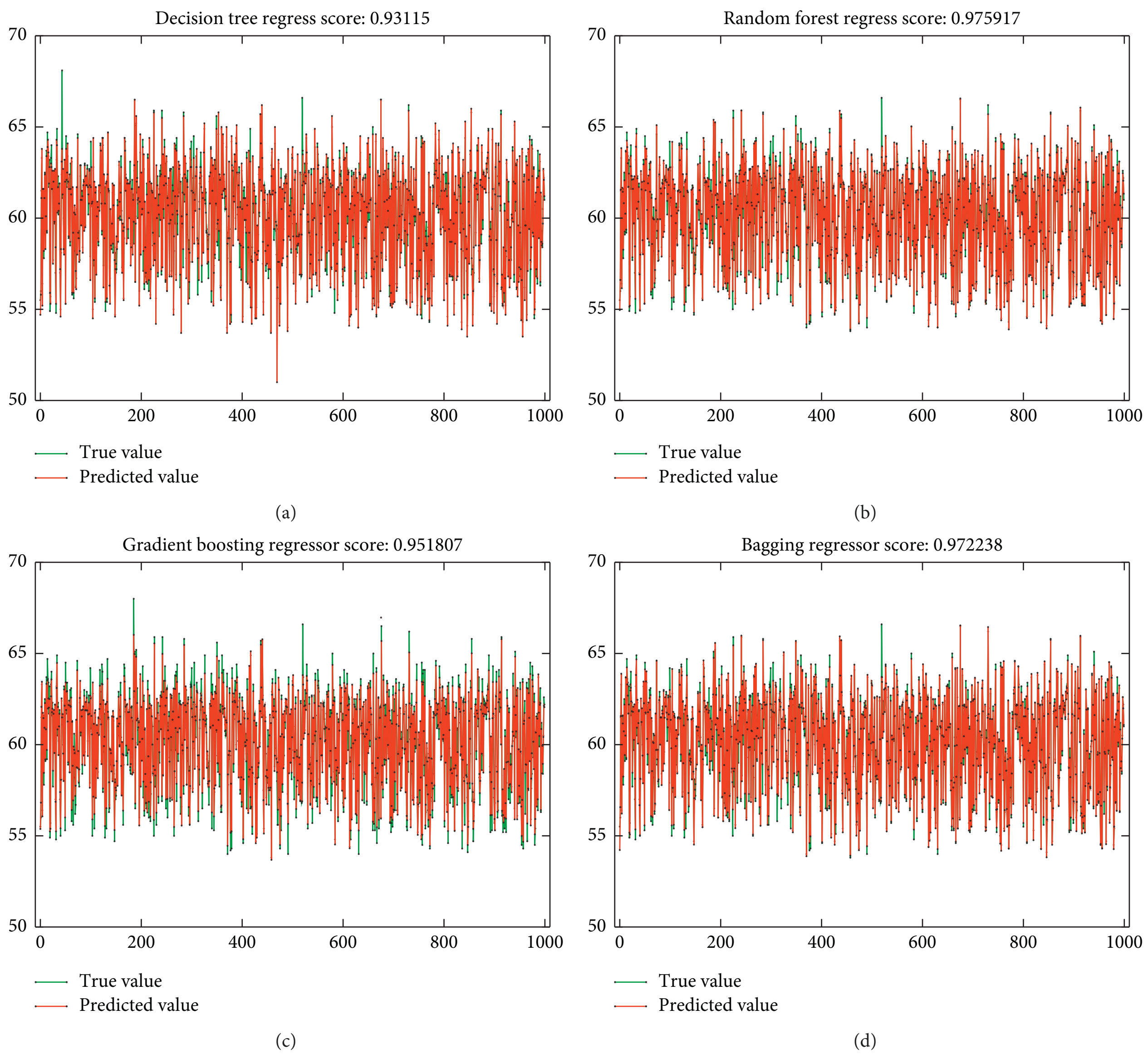


Figure 4: Regression fitting curves of four methods.

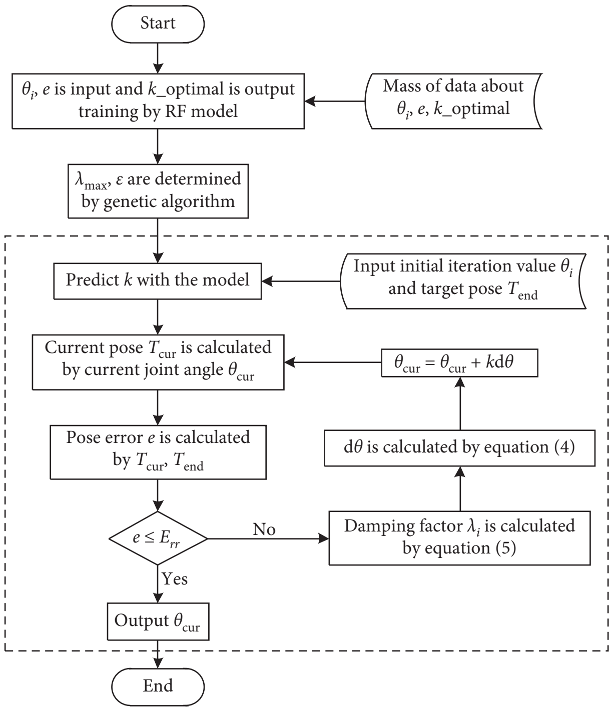

Start
θ_i, e is input and k_optimal is output training by RF model
Mass of data about θ_i, e, k_optimal
$\lambda_{\max}$, ε are determined by genetic algorithm
Predict k with the model
Input initial iteration value θ_i and target pose T_{end}
Current pose T_{cur} is calculated by current joint angle θ_{cur}
$\theta_{\text{cur}} = \theta_{\text{cur}} + k\mathrm{d}\theta$
Pose error e is calculated by T_{cur}, T_{end}
$\mathrm{d}\theta$ is calculated by equation (4)
$e \le E_{rr}$
No
Damping factor λ_i is calculated by equation (5)
Yes
Output θ_{cur}
End

Figure 5: Flowchart of the optimized algorithm.

Table 2: D-H parameters of a 6R robot in the case of the optimized GDLS method.

	a_{i-1} (m)	α_{i-1} (°)	d_i (m)	θ_i (°)
1 (S)	0.78	30	0.61	θ_1
2 (L)	0.38	40	0.72	θ_2
3 (U)	0.62	10	0.93	θ_3
4 (R)	0.53	60	0.79	θ_4
5 (B)	0.46	20	0.39	θ_5
6 (T)	0.12	80	0.17	θ_6

Table 3: IK results of the optimized method based on 4 arbitrary data points.

ID	Target position (m)			Joint angles (deg)						Predicted k_optimal	e (mm)	Iterations
	X	Y	Z	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6			
1	−0.569	−1.219	2.848	−17	−90	−170	66	−32	9	59	0.044	4
2	2.265	−0.364	2.044	72	−53	173	12	140	−97	65	0.035	5
3	1.522	−1.296	−0.424	54	−70	76	175	19	122	79	0.043	9
4	−2.397	0.168	1.698	−130	17	168	−126	−94	160	53	0.042	4

3. Optimized Elimination Method

3.1. Algorithm Description. The process of the elimination method is as follows: by reversible transformation of the matrix, it is obtained that

$$T_3T_4T_5 = (T_2)^{-1}\,(T_1)^{-1}T_{\text{end}}(T_6)^{-1}, \tag{9}$$

where $T_{\text{end}} = \begin{pmatrix} l_x & m_x & n_x & p_x \\ l_y & m_y & n_y & p_y \\ l_z & m_z & n_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$, in which the elements in columns 3 and 4 of the right and left matrices are equal, and 6 equations unrelated to θ_6 are obtained as

$$p_1\colon\ c_3f_1 + s_3f_2 + a_2 = c_2h_1 + s_2h_2 \quad (e_1), \tag{10}$$

$$p_2\colon\ c_3f_2 - s_3f_1 = s_2h_1 - c_2h_2 \quad (e_2), \tag{11}$$

$$p_3\colon\ f_3 = h_3 \quad (e_3), \tag{12}$$

$$l_1\colon\ c_3r_1 + s_3r_2 = c_2n_1 + s_2n_2 \quad (e_4), \tag{13}$$

$$l_2\colon\ s_3r_1 - c_3r_2 = c_2n_2 - s_2n_1 \quad (e_5), \tag{14}$$

$$l_3\colon\ r_3 = n_3 \quad (e_6), \tag{15}$$

where $s_i = \sin(\theta_i)$, $c_i = \cos(\theta_i)$, $s_{ij} = \sin(\theta_i + \theta_j)$, $and\, c_{\ ij} = \cos(\theta_i + \theta_j)$.

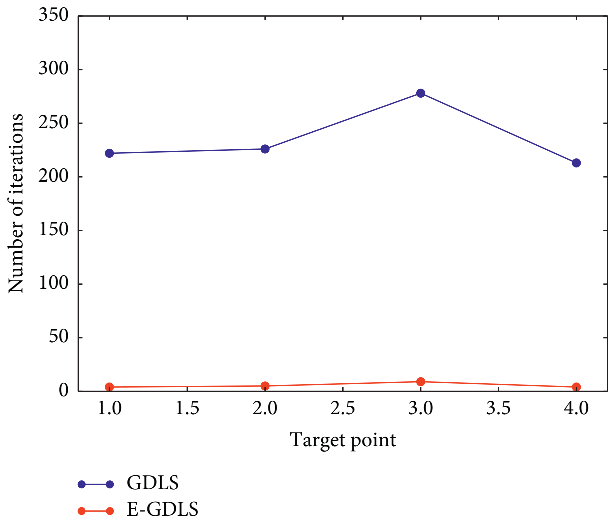


FIGURE 6: Comparison of the number of iterations.

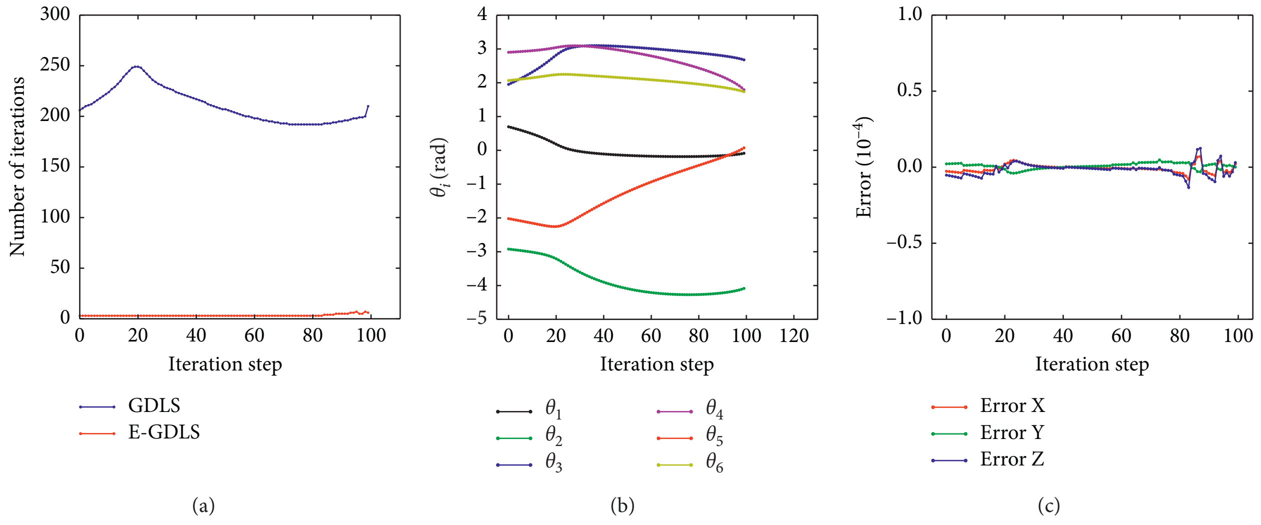


FIGURE 7: Comparison of the number of iterations (a), joint space trajectory (b), and tracking error of the optimized method (c).

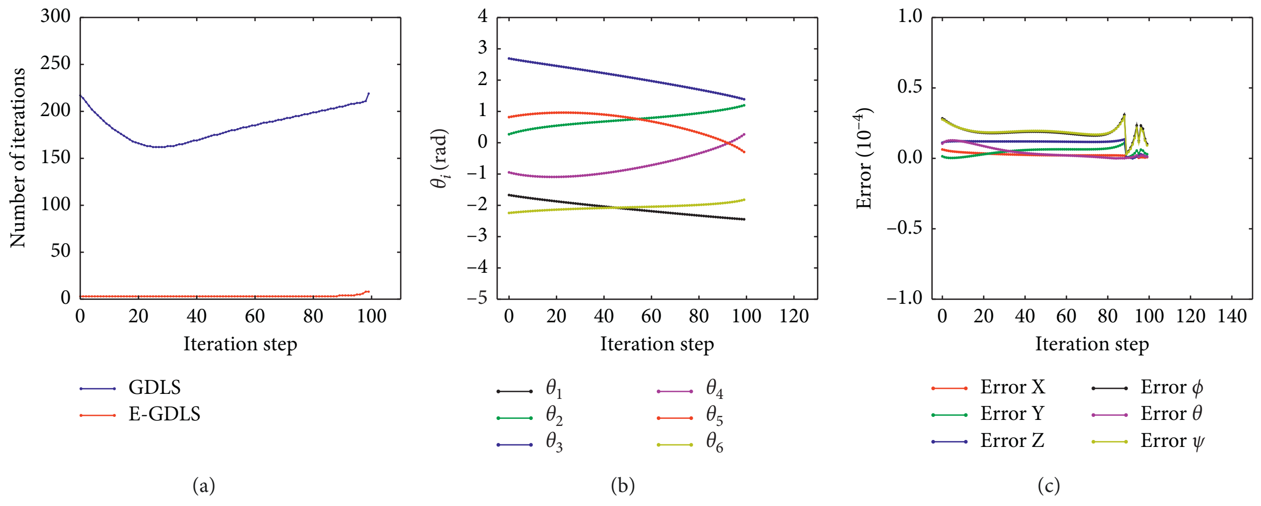
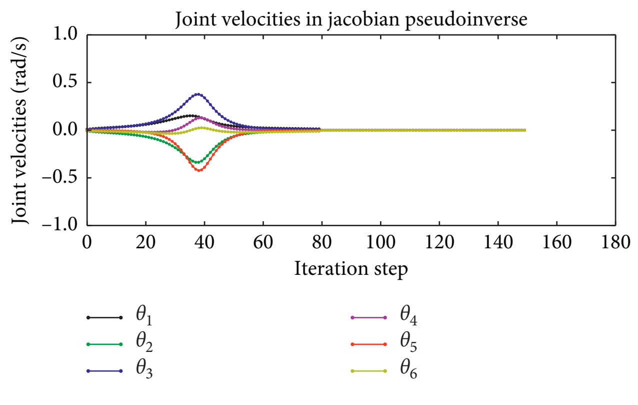
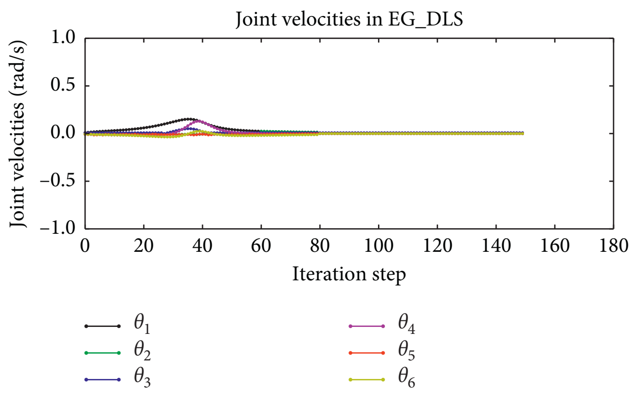


FIGURE 8: Comparison of the number of iterations (a), joint space trajectory (b), and tracking error of the optimized method (c).



(a)



(b)

FIGURE 9: Joint velocities of the two methods near the singular configuration.

TABLE 4: Performance comparison of four methods.

Method	Average number of iterations	Mean solution time (ms)	Proportion of nonconvergence beyond the maximum number of iterations (%)
J^T	161.63	5.254	3.9
DLS	187.34	6.918	5.3
GDLS	182.53	6.546	3.3
E-GDLS	4.23	0.229	0.17

Through the vector calculation of p and l, $p\cdot p$, $p\cdot l$, $p\times l$, $(p\cdot p)l-2(p\cdot l)p$, 8 equations are derived as

$$p_1^2+p_2^2+p_3^2 \quad (e_7), \tag{16}$$

$$p_1l_1+p_2l_2+p_3l_3 \quad (e_8), \tag{17}$$

$$p_1l_2-p_2l_1 \quad (e_9), \tag{18}$$

$$p_2l_3-p_3l_2 \quad (e_{10}), \tag{19}$$

$$p_3l_1-p_1l_3 \quad (e_{11}), \tag{20}$$

$$\left(p_1^2+p_2^2+p_3^2\right)l_1-2\left(p_1l_1+p_2l_2+p_3l_3\right)p_1 \quad (e_{12}), \tag{21}$$

$$\left(p_1^2+p_2^2+p_3^2\right)l_2-2\left(p_1l_1+p_2l_2+p_3l_3\right)p_2 \quad (e_{13}), \tag{22}$$

$$\left(p_1^2+p_2^2+p_3^2\right)l_3-2\left(p_1l_1+p_2l_2+p_3l_3\right)p_3 \quad (e_{14}). \tag{23}$$

Eliminating θ_1,θ_2 and simplifying the 14 equations $e_1\sim e_{14}$, 6 equations irrelevant to θ_1,θ_2 are obtained. The 6 equations are written in the form of a matrix equation as

$$\sum v_1=0, \tag{24}$$

where $v_1=\left[\,s_4s_5\ \ s_4c_5\ \ c_4s_5\ \ c_4c_5\ \ s_4\ \ c_4\ \ s_5\ \ c_5\ \ 1\,\right]^T$ and $\sum$ is a $6*9$ matrix containing only s_3,c_3. We convert s_3,c_3,s_4,c_4,s_5,c_5 into $s_3=2x_3/(1+x_3^2)$, $c_3=(1-x_3^2)\,/(1+x_3^2)$, $s_4=2x_4/(1+x_4^2)$, $c_4=(1-x_4^2)/(1+x_4^2)$, $s_5=2x_5\,/(1+x_5^2)$, $c_5=(1-x_5^2)/(1+x_5^2)$ and obtain the matrix equation

$$\left(\sum{}'\right)v_2=0, \tag{25}$$

where $v_2=\left[\,x_4^2x_5^2\ \ x_4^2x_5\ \ x_4^2\ \ x_4x_5^2\ \ x_4x_5\ \ x_4\ \ x_5^2\ \ x_5\ \ 1\,\right]^T$ and $\sum'$ is a $6*9$ matrix containing only x_3. We multiply each row of the matrix equation by x_4 and merge the equation obtained with the original one, to get the matrix equation

$$\left(\sum{}''\right)v=0, \tag{26}$$

where $v=\left[\,x_4^3x_5^2\,x_4^3x_5\,x_4^3\,x_4^2x_5^2\,x_4^2x_5\,x_4^2\,x_4x_5^2\,x_4x_5\,x_4\,x_5^2\,x_5\,1\,\right]^T$ and $\sum''$ is a $12*12$ matrix containing only x_3.

This is an overconstrained linear system. In order for this system to have a nontrivial solution, the coefficient matrix $\sum''$ must be singular. The determinant of the coefficient matrix is a 16th degree polynomial in x_3. The roots of this polynomial give the values of x_3 corresponding to the solutions of the inverse kinematic problem. And it is simplified to find the matrix eigenvalues and eigenvectors. $\sum''$ is written as $Ax_3^2+Bx_3+C$, and equation (26) is written as

$$\left(Ax_3^2+Bx_3+C\right)v=0. \tag{27}$$

A, B, and C are known $12*12$ matrices. Equation (27) is written as

$$\begin{gathered}\begin{bmatrix}0 & I\\ -A^{-1}C & -A^{-1}B\end{bmatrix}\begin{bmatrix}v\\ x_3v\end{bmatrix}=x_3\begin{bmatrix}v\\ x_3v\end{bmatrix},\\ \text{i.e., } M\begin{bmatrix}v\\ x_3v\end{bmatrix}=x_3\begin{bmatrix}v\\ x_3v\end{bmatrix}.\end{gathered} \tag{28}$$

The eigenvalue of M is x_3, and θ_3 is obtained from $\theta_3=2\tan^{-1}(x_3)$. The eigenvector is $V=\begin{bmatrix}v\\ x_3\,v\end{bmatrix}/\left\|\begin{bmatrix}v\\ x_3v\end{bmatrix}\right\|$.

3.2. Optimization and Supplement of Elimination Method. The simplification process is given in this paper for the first time. θ_1,θ_2 are eliminated and 6 equations are obtained through 14 equations $e_1\sim e_{14}$.

Expressions of s_1,c_1 are derived through e_3,e_6 and then substituted into e_7,e_8,e_9,e_{14} to obtain 4 equations. Equations $e_1,e_2,e_4,e_5,e_{10},e_{11},e_{12}$, and e_{13} are simplified through the following two formulas:

$$\frac{\mu_1^2}{2a_1}(e_{12}-\delta_1 e_4+\delta_2 e_1)-\lambda_1\mu_1 e_{10}+\mu_1 w e_2-\mu_1(r-d_1)e_5=0, \tag{29}$$

$$\frac{\mu_1^2}{2a_1}(e_{13}-\delta_1 e_5+\delta_2 e_2)-\lambda_1\mu_1 e_{11}-\mu_1 w e_1+\mu_1(r-d_1)e_4=0, \tag{30}$$

where $\delta_1 = p^2+q^2+ (r-d_1)^2-a_1^2$ and $\delta_2 = 2(pu+qv+(r-d_1)w)$.

The final expressions of matrices $A, B,$ and C are derived for the first time so that the one-step solving of the matrix M can be realized in this paper.

The complicated coefficient extraction process of the elimination method results in a large workload, so in most cases in other papers, the known quantities are substituted into the coefficient matrix, followed by extract variables, and the whole solving process should be implemented in multiple steps. In this paper, through the complicated coefficient extraction and simplified operation, the intermediate solving steps are streamlined, and the final expressions of matrices $A, B,$ and C are directly derived, so the whole solving process is simplified as much as possible. One-step solving of the matrix M is realized, and only 1 ms or so is needed. This is the greatest contribution that this paper has made to the simplification of the elimination method. The expressions are not presented because they are too long. For the specific expressions, refer to the uploaded MATLAB code at https://github.com/Wangxiaoqi1031/robot1.

Solving formulas for other joint angles $\theta_1, \theta_2, \theta_4, \theta_5,$ and θ_6 in the simplest forms are given for the first time in this paper, so as to improve the integrity of the whole method.

Through a large number of experiments, x_4 and x_5 are obtained by solving the ratio of large absolute values in the eigenvectors of the matrix M more accurately. Through analysis, it is possible that the maximum values are $v_1, v_3, v_{10}, and\, v_{12}$, and the calculation formulas are as follows:

$$\begin{cases}
\theta_4 = \tan^{-1}\left(\dfrac{v_1}{v_4}\right) & \theta_5 = \tan^{-1}\left(\dfrac{v_1}{v_2}\right) \quad \max(v_i) = |v_1|, \\
\theta_4 = \tan^{-1}\left(\dfrac{v_3}{v_6}\right) & \theta_5 = \tan^{-1}\left(\dfrac{v_2}{v_3}\right) \quad \max(v_i) = |v_3|, \\
\theta_4 = \tan^{-1}\left(\dfrac{v_7}{v_{10}}\right) & \theta_5 = \tan^{-1}\left(\dfrac{v_{10}}{v_{11}}\right) \quad \max(v_i) = |v_{10}|, \\
\theta_4 = \tan^{-1}\left(\dfrac{v_9}{v_{12}}\right) & \theta_5 = \tan^{-1}\left(\dfrac{v_{11}}{v_{12}}\right) \quad \max(v_i) = |v_{12}|,
\end{cases} \tag{31}$$

$$\begin{aligned}
s_1 &= \frac{\mu_1 v(d_2+f_3\lambda_2+\lambda_1(d_1-r)-\mu_2(\mathrm{c}_3\mathrm{f}_2-\mathrm{f}_1\mathrm{s}_3))+q\mu_1(w\lambda_1-r_3\lambda_2+\mu_2(c_3r_2-\mathrm{r}_1\mathrm{s}_3))}{p\mu_1^2 v-qu\mu_1^2}, \\
\theta_1 &= \sin^{-1}(s_1) \\
\text{or}\, \theta_1 &= \pi-\sin^{-1}(s_1),
\end{aligned} \tag{32}$$

$$\begin{aligned}
n &= \mu_6\big(n_z\lambda_1-c_1n_y\mu_1+n_xs_1\mu_1\big)-\lambda_6\big(m_z\lambda_1-c_1m_y\mu_1+m_xs_1\mu_1\big), \\
m &= l_z\lambda_1-c_1l_y\mu_1+l_xs_1\mu_1, \\
l &= s_5(\mu_4(\lambda_2\lambda_3-c_3\mu_2\mu_3)+c_4\lambda_4(\mu_3\lambda_2+c_3\mu_2\lambda_3)-s_3s_4\mu_2\lambda_4)+c_5(s_4(\mu_3\lambda_2+c_3\mu_2\lambda_3)+c_4s_3\mu_2), \\
x_6 &= \frac{n\pm\sqrt{m^2+n^2-l^2}}{l+m}, \\
\theta_6 &= 2\tan^{-1}(x_6),
\end{aligned} \tag{33}$$

$$\begin{aligned}
n' &= l_ys_1\mu_2+l_xc_1\mu_2,\ m' = l_xs_1\mu_2\lambda_1-l_z\mu_1\mu_2-l_yc_1\mu_2\lambda_1, \\
l' &= s_6(\mu_5(\lambda_3\lambda_4-c_4\mu_3\mu_4)+c_5\lambda_5(\mu_4\lambda_3+c_4\mu_3\lambda_4)-s_4s_5\mu_3\lambda_5)+c_6(s_5(\mu_4\lambda_3+c_4\mu_3\lambda_4)+c_5s_4\mu_3)-l_z\lambda_1\lambda_2+l_yc_1\mu_1\lambda_2-l_xs_1\mu_1\lambda_2, \\
x_2 &= \frac{n'\pm\sqrt{m'^2+n'^2-l'^2}}{l'+m'}, \\
\theta_2 &= 2\tan^{-1}(x_2).
\end{aligned} \tag{34}$$

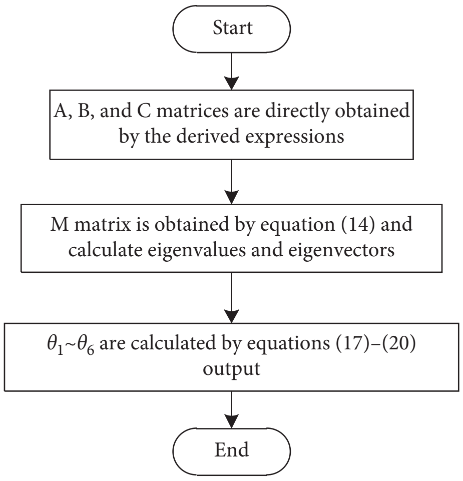

Start
A, B, and C matrices are directly obtained by the derived expressions
M matrix is obtained by equation (14) and calculate eigenvalues and eigenvectors
θ_1~θ_6 are calculated by equations (17)–(20) output
End

Figure 10: Flowchart of the optimized algorithm.

3.3. Flowchart of Optimized Method. The flowchart of the optimized elimination method is shown in Figure 10.

3.4. Case of Optimized Elimination Method. A randomly selected 6R robot is taken as an example for the test, and D-H parameters are listed in Table 5.

We set $\theta_1 = 14.5427°, \theta_2 = -44.5435°, \theta_3 = 130.5472°, \theta_4 = -70.2145°$, $\theta_5 = 60.2147°$, and $d\theta_6 = 160.9745°$. The matrix of the target pose is obtained as $T_{\text{end}} = [T_1, T_2, T_3, T_4]^T$, where

$$\begin{aligned} T_1 &= [-0.24811675957960,\ 0.85817093890060,\\ &\quad -0.44942264433624, 2.28037877048099],\\ T_2 &= [0.37824219259125, -0.34128340141828,\\ &\quad -0.86049897365432, 0.30849705188949],\\ T_3 &= [-0.8918357008774, -0.38349482335858,\\ &\quad -0.23991832588964,\ 0.58096198011612],\\ T_4 &= [0, 0, 0, 1]. \end{aligned} \tag{35}$$

The inverse solution is solved according to Figure 10. The real results are shown in Table 6.

We evaluate the performance of the methods with two indicators of accuracy and speed. We randomly select 10^5 data points in the working space of the robot and test them under the environment of 3.2 GHz CPU, 8 GB RAM, and MATLAB and use the elimination method before and after optimization to solve IK, respectively. The worst, best, and average cases are calculated separately. The statistical results are shown in Table 7.

The results show that there is not much difference in the accuracy of the method before and after optimization, and the solution speed of the method after optimization decreases from 11–13 ms to 1.4–1.8 ms. Therefore, it is concluded that the optimization of the elimination method in this paper can effectively improve the solution speed while ensuring high accuracy.

When the elimination method is adopted, the proportion of poses which cannot be accurately solved is 0.086% as the matrix A is singular or the robot is close to the singular configuration, and this can be solved through the following scheme:

If A is singular, inverse solving is changed into pseudoinverse solving. The singular region is evaded in the trajectory planning.

Poses which cannot be accurately solved can be processed through the iteration method.

Table 5: D-H parameters of a 6R robot in the case of the optimized elimination method.

	a_{i-1} (m)	α_{i-1} (°)	d_i (m)	θ_i (°)
1 (S)	0.25	20	0.19	θ_1
2 (L)	0.95	30	0.37	θ_2
3 (U)	0.30	−45	0.10	θ_3
4 (R)	0.55	80	1.55	θ_4
5 (B)	0.16	−120	0.21	θ_5
6 (T)	0.22	100	0.13	θ_6

4. Experiment

There is an ordinary 6R configurational robot experimental platform in the laboratory, as shown in Figure 11. The processor of the controller is OMAPL-138, a DSP + ARM dual-core processor, the demonstrator is based on WinCE 6.0, and the maximum speed can reach 300 °/s.

In order to compare ordinary analytical methods with the two optimized methods in this paper in terms of their solving performance, 1000 tracks are randomly selected, the iterative convergence error is set as $E_{rr} = 0.0001$, and the maximum number of iterations is $i\,\text{limit} = 500$; data calculated from the DSP are shown in Table 8.

It can be seen from Table 8 that the accuracy of the optimized elimination method is approximate to that of the analytical solution, and the solving time of the optimized Gaussian damping method is approximate to that of the analytical solution, too. There is no analytical solution for the robot configuration which does not

Table 6: All groups of real solutions.

	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	36.64732	55.0222	154.9653	77.53909	−136.2382	93.16235
2	36.64732	−50.54915	154.9653	77.53909	−136.2382	93.16235
3	36.64732	105.916	154.9653	77.53909	−136.2382	−9.495723
4	36.64732	−101.443	154.9653	77.53909	−136.2382	−9.495723
5	143.3527	110.5793	154.9653	77.53909	−136.2382	93.16235
6	143.3527	−59.60432	154.9653	77.53909	−136.2382	93.16235
7	143.3527	145.3161	154.9653	77.53909	−136.2382	−9.495723
8	143.3527	−94.34108	154.9653	77.53909	−136.2382	−9.495723
9	14.5427	20.82859	130.5472	−70.2145	60.2147	160.9745
10	14.5427	−44.5435	130.5472	−70.2145	60.2147	160.9745
11	14.5427	15.28996	130.5472	−70.2145	60.2147	−92.76167
12	14.5427	−39.00486	130.5472	−70.2145	60.2147	−92.76167
13	165.4573	98.19595	130.5472	−70.2145	60.2147	160.9745
14	165.4573	−59.02553	130.5472	−70.2145	60.2147	160.9745
15	165.4573	96.19785	130.5472	−70.2145	60.2147	−92.76167
16	165.4573	−57.02743	130.5472	−70.2145	60.2147	−92.76167

Table 7: All groups of real solutions.

		Best	Worst	Average
Accuracy	Elimination	$2.374e10^{-13}$	$7.324e10^{-11}$	$2.318e10^{-12}$
	Optimized elimination	$3.423e10^{-13}$	$4.012e10^{-11}$	$2.054e10^{-12}$
Speed (ms)	Elimination	11.253	12.454	11.632
	Optimized elimination	1.494	1.721	1.583

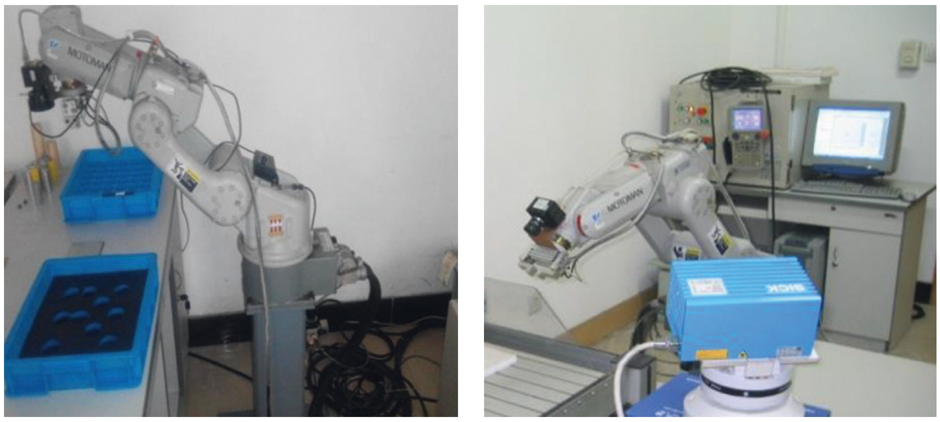

(a) (b)

Figure 11: Experimental platform.

Table 8: Performance comparison of three methods.

Method	Average solution time (μs)	Average solution accuracy
Analytic method	228	$1.063e10^{-12}$
Optimized elimination method	1504	$2.204e10^{-12}$
Optimized GDLS	234	0.0000989

conform to the Pieper criterion, and then it can be solved through the two optimized universal methods in this paper. The proper method can be selected, or the two methods can be combined according to different accuracy and speed requirements.

5. Conclusion

This paper optimized the elimination method and the Gaussian damped least-squares method. After optimization, for the elimination method, the average accuracy of 10^{-12}

can be achieved, and the solution time is 1 to 1.5 ms. For the Gaussian damped least-squares method, the average number of iterations is only 4 to 5, and the solving speed is close to that of the analytical solution. The precision and speed of these two optimized algorithms cannot be achieved by the other optimized algorithms. The elimination method is simplified to the extreme. Time is mainly spent in the calculation of the eigenvalues and eigenvectors of the M matrix. If this step can be improved, the speed of the algorithm can be further improved. The Gaussian damped least-squares method has the advantage of smooth motion itself. In this paper, by introducing the variable step-size coefficient, the number of iterations is greatly reduced, and the solving speed is effectively improved. Applying the method of machine learning to study the coefficient could achieve better results than the general fitting method. In practical application, if high speed is required, the iteration method can be applied to solve the problem. If high precision is required, the elimination method can be applied, or the two methods can be combined to solve the problem.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 61573272) and Shaanxi Science and Technology Project (no. 2019ZDLGY01-01-02).

References

[1] D. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on Man Machine Systems*, vol. 10, no. 2, pp. 47–53, 1969.

[2] I. Dulęba and M. Opałka, "A comparison of Jacobian-based methods of inverse kinematics for serial robot manipulators," *International Journal of Applied Mathematics and Computer Science*, vol. 23, no. 2, pp. 373–382, 2013.

[3] S. Miyata, S. Miyahara, and D. Nenchev, "Analytical formula for the pseudoinverse and its application for singular path tracking with a class of redundant robotic limbs," *Advanced Robotics*, vol. 31, no. 10, pp. 509–518, 2017.

[4] J. Karpińska and K. Tchoń, "Performance-oriented design of inverse kinematics algorithms: extended jacobian approximation of the jacobian pseudo-inverse," *Journal of Mechanisms and Robotics*, vol. 4, no. 2, Article ID 021008, 2012.

[5] C. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 93–101, 1986.

[6] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *The International Journal of Robotics Research*, vol. 6, no. 2, pp. 3–15, 1987.

[7] X. Wang, D. Zhang, and C. Zhao, "Inverse kinematics of a 7R 6-DOF robot with nonspherical wrist based on transformation into the 6r robot," *Mathematical Problems in Engineering*, vol. 2017, Article ID 2074137, 12 pages, 2017.

[8] S. R. Buss and J.-S. Kim, "Selectively damped least squares for inverse kinematics," *Journal of Graphics Tools*, vol. 10, no. 3, pp. 37–49, 2005.

[9] 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," *Journal of Mechanical Science and Technology*, vol. 22, no. 7, pp. 1330–1338, 2008.

[10] O. M. Omisore, S. Han, L. Ren et al., "Deeply-learnt damped least-squares (DL-DLS) method for inverse kinematics of snake-like robots," *Neural Networks*, vol. 107, pp. 34–47, 2018.

[11] J. Xu, K. Song, Y. He, Z. Dong, and Y. Yan, "Inverse kinematics for 6-DOF serial manipulators with offset or reduced wrists via a hierarchical iterative algorithm," *IEEE Access*, vol. 6, pp. 52899–52910, 2018.

[12] M. Raghavan and B. Roth, "Kinematic analysis of the 6R manipulator of general geometry," in *Proceedings of the 5th International Symposium on Robotics Research*, pp. 263–269, MIT Press, Cambridge, MA, USA, January 1990.

[13] D. Manocha and J. F. Canny, "Efficient inverse kinematics for general 6R manipulators," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 648–657, 1994.

[14] J. S. Kim and G. S. Chirikjian, "Inverse kinematic solutions of 6-D.O.F. biopolymer segments," *Robotica*, vol. 34, no. 8, pp. 1734–1753, 2016.

[15] Z. Fu, W. Yang, and Z. Yang, "Solution of inverse kinematics for 6R robot manipulators with offset wrist based on geometric algebra," *Journal of Mechanisms and Robotics*, vol. 5, no. 3, Article ID 031010, 2013.

[16] W. Suleiman, "On inverse kinematics with inequality constraints: new insights into minimum jerk trajectory generation," *Advanced Robotics*, vol. 30, no. 17-18, pp. 1164–1172, 2016.

[17] A. L. Kleppe and O. Egeland, "Inverse kinematics for industrial robots using conformal geometric algebra," *Modeling, Identification and Control: A Norwegian Research Bulletin*, vol. 37, no. 1, pp. 63–75, 2016.

[18] R. Zhao, Z. Shi, Y. Guan et al., "Inverse kinematic solution of 6R robot manipulators based on screw theory and the Paden–Kahan subproblem," *International Journal of Advanced Robotic Systems*, vol. 15, no. 6, Article ID 1729881418818297, 2018.

[19] A. El-Sherbiny, M. A. Elhosseini, and A. Y. Haikal, "A comparative study of soft computing methods to solve inverse kinematics problem," *Ain Shams Engineering Journal*, vol. 9, no. 4, pp. 2535–2548, 2017.

[20] Z. Zhou, H. Guo, Y. Wang et al., "Inverse kinematics solution for robotic manipulator based on extreme learning machine and sequential mutation genetic algorithm," *International Journal of Advanced Robotic Systems*, vol. 15, no. 4, Article ID 1729881418792992, 2018.

[21] H. C. Huang, S. S. D. Xu, and H. S. Hsu, "Hybrid taguchi DNA swarm intelligence for optimal inverse kinematics redundancy resolution of six-DOF humanoid robot arms," *Mathematical Problems in Engineering*, vol. 2014, Article ID 358269, 9 pages, 2014.

[22] H. Toshani and M. Farrokhi, "Real-time inverse kinematics of redundant manipulators using neural networks and quadratic programming: a Lyapunov-based approach," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 766–781, 2014.

[23] R. Köker, "A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization," *Information Sciences*, vol. 222, pp. 528–543, 2013.

[24] R. Köker, T. Çakar, and Y. Sari, "A neural-network committee machine approach to the inverse kinematics problem solution of robotic manipulators," *Engineering with Computers*, vol. 30, no. 4, pp. 641–649, 2014.

[25] C. Lopez-Franco, J. Hernandez-Barragan, A. Y. Alanis, and N. Arana-Daniel, "A soft computing approach for inverse kinematics of robot manipulators," *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 104–120, 2018.

[26] A. R. J. Almusawi, L. C. Dülger, and S. Kapucu, "A new artificial neural network approach in solving inverse kinematics of robotic arm (denso vp6242)," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 572016, 10 pages, 2016.

[27] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[28] P. Soucy and G. W. Mineau, "A simple KNN algorithm for text categorization," in *Proceedings of the 2001 IEEE International Conference on Data Mining*, pp. 647-648, San Jose, CA, USA, December 2001.

[29] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[30] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.