Research Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
July-August 2018: 1–15

DOI: 10.1177/1729881418792992
journals.sagepub.com/home/arx
SAGE

Inverse kinematics solution for robotic manipulator based on extreme learning machine and sequential mutation genetic algorithm

Zhiyu Zhou[1], Hanxuan Guo[1], Yaming Wang[1], Zefei Zhu[2], Jiang Wu[1] and Xiangqi Liu[2]

Abstract

This article presents an intelligent algorithm based on extreme learning machine and sequential mutation genetic algorithm to determine the inverse kinematics solutions of a robotic manipulator with six degrees of freedom. This algorithm is developed to minimize the computational time without compromising the accuracy of the end effector. In the proposed algorithm, the preliminary inverse kinematics solution is first computed by extreme learning machine and the solution is then optimized by an improved genetic algorithm based on sequential mutation. Extreme learning machine randomly initializes the weights of the input layer and biases of the hidden layer, which greatly improves the training speed. Unlike classical genetic algorithms, sequential mutation genetic algorithm changes the order of the genetic codes from high to low, which reduces the randomness of mutation operation and improves the local search capability. Consequently, the convergence speed at the end of evolution is improved. The performance of the extreme learning machine and sequential mutation genetic algorithm is also compared with that of a hybrid intelligent algorithm, and the results showed that there is significant reduction in the training time and computational time while the solution accuracy is retained. Based on the experimental results, the proposed extreme learning machine and sequential mutation genetic algorithm can greatly improve the time efficiency while ensuring high accuracy of the end effector.

Keywords

Robotic manipulator, inverse kinematics, extreme learning machine, sequential mutation genetic algorithm, robotic control

Date received: 29 March 2018; accepted: 15 July 2018

Topic: Robot Manipulation and Control
Topic Editor: Andrey V Savkin
Associate Editor: Yongping Pan

Introduction

The inverse kinematics of a robotic manipulator can be used to describe the mapping of the manipulator and end effector from the Cartesian space to joint space. It is crucial to obtain highly accurate inverse kinematics solutions in an efficient manner in the design, motion planning, and control of the manipulator.[1–3] The main problem in determining the inverse kinematics solution of a robotic manipulator

[1] School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou, China
[2] School of Mechanical Engineering, Hangzhou Dianzi University, Hangzhou, China

Corresponding author:
Yaming Wang, School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China.
Email: wangyaming2018@126.com

is to compute each joint angle $\theta(\theta_1, \theta_2, \cdots, \theta_n)$ by algebraic, geometric, iterative optimization, or machine learning approaches based on the end effector pose $T(p_x, p_y, p_z, o_x, o_y, o_z, a_x, a_y, a_z, n_x, n_y, n_z)$.

Determining the inverse kinematics solution of a robotic manipulator with a high degree of freedom (DOF) is a complex process. There are three types of methods typically used to determine the inverse kinematics solution of a robotic manipulator, namely, geometric, algebraic, and iterative algorithms. Each method has its own disadvantages. For example, algebraic methods cannot guarantee a closed-form solution. Geometric methods can be used to solve the inverse kinematics problem only when the first three joints of the manipulator have a closed-form solution. The convergence speed of iterative algorithms is dependent on the initial position of the robotic manipulator. In addition, due to the complex geometry of the structure, most methods are unable to determine the inverse kinematics solution of the robotic manipulator in a fast, efficient manner. Hence, much effort has been made by researchers to develop intelligent algorithms in order to compute the inverse kinematics solutions of robotic manipulators. Among these intelligent algorithms, artificial neural networks (ANNs) and genetic algorithms (GAs) have gained much attention from researchers in order to solve inverse kinematics problems of robotic manipulators.

Recently, many researchers proposed neural network-based inverse kinematics solutions of robotic manipulators.[4–6] It has been proven theoretically that neural networks with three or more layers can fit any nonlinear system and this forms the basis that ANNs can be used to solve the inverse kinematics problems of robotic manipulators. Tejomurtula and Kak[7] presented a structured ANN to solve the inverse kinematics problem of a robotic manipulator, which prevents the shortcomings of the backpropagation algorithm in terms of the training time and computational accuracy. Karlik and Aydin[8] used six independent ANNs with two hidden layers to compute the inverse kinematics solution of a 6-DOF manipulator. Kalra et al.[9] proposed a new evolutionary approach to solve the multimodal inverse kinematics. This algorithm realizes high real-time and robust control of the inverse kinematics of the robotic manipulator. Bingul et al.[10] proposed an inverse kinematics algorithm based on backpropagation ANN. This method is disadvantageous because of the large joint angle errors and moreover, the method is incapable of handling multiple solutions. Rasit Köker and Tarık Cakar[11–15] proposed a variety of hybrid intelligent algorithms based on ANN, GA, and simulated annealing (SA). In their studies, ANN was used to compute a preliminary solution and the fractional part of the inverse kinematics solution was optimized by GA or SA. However, the algorithms are still lacking in terms of the time efficiency because the training process of the ANN is rather slow while the optimization process based on the improved GA and SA requires a significant number of iterations to obtain a converged solution with high accuracy. A number of researchers have also proposed inverse kinematics solutions based on genetic GAs[9,16] and artificial bee colony algorithms.[17] These optimization algorithms search in the solution space subject to certain rules and the convergence speed is unsatisfactory in most situations. All of the aforementioned algorithms suffer from the same fundamental problem: the iteration process is time-consuming.

In this article, an intelligent algorithm is proposed based on extreme learning machine (ELM) and sequential mutation genetic algorithm (SGA) to determine the inverse kinematics solution of a 6-DOF robotic manipulator. The main focus of this novel approach is to improve the time efficiency of the algorithm while ensuring high precision of the end effector inverse kinematics. In this algorithm, forward kinematics is used to compute a training set from the joint space $\theta(\theta_1, \theta_2, \cdots, \theta_n)$ of the manipulator to the pose $\overrightarrow{p}(x,y,z)$, $\overrightarrow{n}(x,y,z)$, $\overrightarrow{o}(x,y,z)$, $\overrightarrow{a}(x,y,z)$ of the end effector. Following this, the ELM is trained by this training set to obtain the initial prediction model. Finally, the preliminary inverse kinematics solution is optimized using SGA to reduce errors of the end effector.

The remainder of this article is organized as follows. The 6-DOF MT-ARM robotic manipulator used in the experiments and the fundamental theories pertaining to the forward kinematics of the robotic manipulator, ELM, and GA are presented in the "Preliminaries" section. In the "Proposed algorithm" section, we will describe in detail the proposed inverse kinematics solution of the robotic manipulator by integrating ELM with SGA, named the ELM-SGA algorithm. In the "Results and discussion" section, we present and compare the simulation and experimental results obtained using the proposed algorithm (ELM-SGA) and mainstream algorithm (Hybrid). Finally, the conclusions drawn based on the findings of the ELM-SGA algorithm are presented in the "Conclusions" section.

Preliminaries

Structure and Denavit–Hartenberg parameters of the 6-DOF Stanford MT-ARM robotic manipulator

The simulations and experiments are designed based on the 6-DOF Stanford MT-ARM robotic manipulator shown in Figure 1. The Denavit–Hartenberg method is used to analyze the kinematics of the manipulator. The basic strategy involves establishing the space coordinate system for each joint and then compute the conversion matrix between the adjacent joint coordinates. The final transformation matrix from the joint space of the manipulator to the Cartesian space of the end effector was calculated by the matrix transfer between each adjacent joint.[18] The four Denavit–Hartenberg parameters (θ_i, d, a, α) of the MT-ARM robotic manipulator represent the joint angle, link offset,

Figure 1. Six-DOF Stanford MT-ARM robotic manipulator. DOF: degree of freedom.

Table 1. Denavit–Hartenberg parameters of the 6-DOF Stanford MT-ARM robotic manipulator.

Joint no.	θ	d (mm)	a	α (rad)	Joint limit (rad)
1	θ_1	366.5	0	$-\pi/2$	$(-\pi, \pi)$
2	θ_2	242	0	$\pi/2$	$(-\pi, \pi)$
3	θ_3	0	0	$-\pi/2$	$(-2\pi/3, 2\pi/3)$
4	θ_4	356	0	$\pi/2$	$(-\pi, \pi)$
5	θ_5	0	0	$-\pi/2$	$(-2\pi/3, 2\pi/3)$
6	θ_6	280	0	0	$(-\pi, \pi)$

DOF: degree of freedom.

link length, and link twist, respectively, and the values are summarized in Table 1.

Forward kinematics of the 6-DOF Stanford MT-ARM robotic manipulator

The forward kinematics of a robotic manipulator can be described by the mapping from the joint space to the Cartesian space.[19] The forward kinematics of a robotic manipulator is usually analyzed by the Denavit–Hartenberg method, which establishes the coordinates for each joint and then calculates the position matrix by transformation between each adjacent joint coordinate.[20] The key aspect of the robotic manipulator forward kinematics is the matrix transformation between the adjacent joint coordinates. This is the well-known Denavit–Hartenberg transformation notation, which is given by

$$
\begin{aligned}
{}^{n}T_{n+1} = A_{n+1} &= R(z,\theta_{n+1}) \times T(0,0,d_{n+1}) \times T(a_{n+1},0,0) \times R(x,\alpha_{n+1}) \\
&= \begin{bmatrix} C_{n+1} & -S_{n+1}C_{\alpha_{n+1}} & S_{n+1}S_{\alpha_{n+1}} & a_{n+1}C_{n+1} \\ S_{n+1} & C_{n+1}C_{\alpha_{n+1}} & -C_{n+1}S_{\alpha_{n+1}} & a_{n+1}S_{n+1} \\ 0 & S_{\alpha_{n+1}} & C_{\alpha_{n+1}} & d_{n+1} \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{1}
$$

where ${}^{n}T_{n+1}$ is the transformation from coordinate n to $n+1$, denoted as A_{n+1}. $R(z,\theta_{n+1})$ and $R(x,\alpha_{n+1})$ denote the basic rotation transformation matrices around the z-axis and x-axis, respectively. $T(0,0,d_{n+1})$ and $T(a_{n+1},0,0)$ denote the basic translation transformation matrices along the z-axis and x-axis, respectively. S_{n+1} and C_{n+1} represent $\sin\theta_{n+1}$ and $\cos\theta_{n+1}$, respectively. When the matrix transformation between each two adjacent joint coordinate systems is determined, the total transformation matrix from the joint angle space to the Cartesian space of the end effector can be easily obtained as

$$
{}^{0}T_{6} = A_1A_2A_3A_4A_5A_6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}
$$

where ${}^{0}T_{6}$ is the pose of the end effector in the base coordinate system and $[n_x, n_y, n_z]^{\mathrm{T}}$ is the vector representation of the end effector's x-axis in the base coordinate system. Analogously, $[o_x, o_y, o_z]^{\mathrm{T}}$ and $[a_x, a_y, a_z]^{\mathrm{T}}$ represent the vector representations of the end effector's y-axis and z-axis, respectively. $[p_x, p_y, p_z]^{\mathrm{T}}$ is the position of the end effector in the base coordinate system. Each parameter in the matrix of equation (2) is calculated as follows

$$
p_x = -C_1\{C_2[C_3C_4S_5d_6 + S_3(C_5d_6 + d_4)] - S_2S_4S_5d_6\} + S_1[S_3C_4S_5d_6 - C_3(C_5d_6 + d_4) - d_2] \tag{3}
$$

$$
p_y = -S_1\{C_2[C_3C_4S_5d_6 + S_3(C_5d_6 + d_4)] - S_2S_4S_5d_6\} - C_1[S_3C_4S_5d_6 - C_3(C_5d_6 + d_4) - d_2] \tag{4}
$$

$$
p_z = S_2\{C_3[C_3C_4S_5d_6 + S_3(C_5d_6 + d_4)] + C_2S_4S_5d_6\} + d_1 \tag{5}
$$

$$
n_x = C_1\{C_2[C_3(C_4C_5C_6 - S_4S_6) - S_3S_5S_6] - S_2(S_4C_5C_6 + C_4S_6)\} - S_1[S_3(C_4C_5C_6 - S_4S_6) + C_3S_5C_6] \tag{6}
$$

$$n_y = S_1\{C_2[C_3(C_4C_5C_6 - S_4S_6) - S_3S_5S_6] - S_2(S_4C_5C_6 + C_4S_6)\} - C_1[S_3(C_4C_5C_6 - S_4S_6) + C_3S_5C_6] \quad (7)$$

$$n_z = -S_2[C_3(C_4C_5C_6 - S_4S_6) - S_3S_5S_6] - C_2(S_4C_5C_6 + C_4S_6) \quad (8)$$

$$o_x = -C_1\{C_2[C_3(C_4C_5C_6 + S_4S_6) - S_3S_5S_6] - S_2(S_4C_5C_6 - C_4C_6)\} + S_1[S_3(C_4C_5C_6 + S_4S_6) + C_3S_5C_6] \quad (9)$$

$$o_y = -S_1\{C_2[C_3(C_4C_5C_6 + S_4S_6) - S_3S_5S_6] - S_2(S_4C_5C_6 - C_4S_6)\} - C_1[S_3(C_4C_5C_6 + S_4S_6) + C_3S_5C_6] \quad (10)$$

$$o_z = S_2[C_3(C_4C_5C_6 + S_4S_6) - S_3S_5S_6] + C_2(S_4C_5C_6 - C_4S_6) \quad (11)$$

$$a_x = -C_1[C_2(C_3C_4S_5 + S_3C_5) - S_2S_4S_5] - S_1(S_3C_4S_5 - C_3C_5) \quad (12)$$

$$a_y = -S_1[C_2(C_3C_4S_5 + S_3C_5) - S_2S_4S_5] - C_1(S_3C_4S_5 - C_3C_5) \quad (13)$$

$$a_z = S_2(C_3C_4S_5 + S_3C_5) + C_2S_4S_5 \quad (14)$$

Extreme learning machine

ELM[21–23] is the learning algorithm for single hidden layer neural networks proposed by Huang et al.[21] in 2006. As a single hidden layer feedforward neural network (SLFN), the training phase of the ELM consists of two stages: (1) random feature mapping and (2) solution of the linear parameters. In the first stage, ELM randomly initializes the hidden layer to map the input data into a feature space using nonlinear mapping functions, which can be any nonlinear piecewise continuous functions.[24,25] In ELM, the weights of the input layer and biases of the hidden layer are randomly initialized and they are not adjusted during the training process. In order to achieve better results, the weights of the input layer and biases of the hidden layer are set within a range of [−1, 1] and [−1, 1], respectively, during random initialization. Hence, ELM can greatly improve the training speed compared with conventional neural networks. The fundamental theory of ELM is briefly described as follows.

For N arbitrary distinct samples (X_i, T_i), where $X_i = [x_{i1}, x_{i2}, \cdots, x_{in}]^{\mathrm{T}}$ is the input and $T_i = [t_{i1}, t_{i2}, \cdots, t_{im}]^{\mathrm{T}}$ is the output, the standard SLFNs with N hidden nodes and activation function $g(x)$ are mathematically modeled as

$$\sum_{i=1}^{N} \beta_i g(W_i \cdot X_j + b_i) = O_j,\ j = 1, \cdots, N \quad (15)$$

where $W_i = [w_{i1}, w_{i2}, \cdots, w_{in}]^{\mathrm{T}}$ is the weight vector connecting the ith hidden node and the input nodes, $\beta_i = [\beta_{i1}, \beta_{i2}, \cdots, \beta_{im}]^{\mathrm{T}}$ is the weight vector connecting the ith hidden node and the output nodes, b_i is the threshold of the ith hidden node, and O_j is the output vector. Ideally, the main goal of the training process is to obtain β_i for the function $g(x)$ such that the output vector will have no error approximation to the target value of the training sample, that is, $\sum_{j=1}^{N} ||O_j - T_j|| = 0$. Hence, equation (15) can be rewritten as

$$\sum_{i=1}^{N} \beta_i g(W_i \cdot X_j + b_i) = T_j,\ j = 1, \cdots, N \quad (16)$$

This equation can be expressed in matrix form as follows

$$H\beta = T \quad (17)$$

where

$$H(W_1, W_2, \cdots, W_N, b_1, b_2, \cdots, b_N, X_1, X_2, \cdots, X_N) = \begin{bmatrix} g(W_1 \cdot X_1 + b_1) & \cdots & g(W_N \cdot X_1 + b_N) \\ \vdots & \vdots & \vdots \\ g(W_1 \cdot X_N + b_1) & \cdots & g(W_N \cdot X_N + b_N) \end{bmatrix} \quad (18)$$

Algorithm 1. Steps involved in the ELM algorithm.

1: **Input:** Training dataset $\{(X_i,\ T_i)|\ X_i \in R^n,\ T_i = R^m,\ i = 1,\ 2,\ \cdots,\ N\}$, activation function $g(x)$, and hidden layer size L.
2: Randomly initialize the input weights W_i and hidden bias b_i.
3: Compute the hidden layer output matrix H.
4: Compute the output weight matrix $\beta : \widehat{\beta} = H^{\dagger}T$.
5: **Output:** Trained ELM model.

$$\beta = \begin{bmatrix} \beta_1^{\mathrm{T}} \\ \vdots \\ \beta_N^{\mathrm{T}} \end{bmatrix}_{N\times m} \quad \text{and} \quad T = \begin{bmatrix} T_1^{\mathrm{T}} \\ \vdots \\ T_N^{\mathrm{T}} \end{bmatrix}_{N\times m} \tag{19}$$

where H is the output matrix of the hidden layer, β is the output weight matrix, and T is the target matrix. It shall be noted that W_i and b_i are randomly initialized. Thus, the sole target of the training process is to compute the output weight matrix β, which can be easily determined using the least squares method. The formula is expressed as

$$\widehat{\beta} = H^{\dagger}T \tag{20}$$

where $H^{\dagger}$ is the Moore–Penrose generalized inverse matrix H. Algorithm 1 shows the steps involved in the ELM algorithm.

Genetic algorithm

GA is a stochastic search optimization algorithm derived from the survival rules in the biological evolution theory. GA was first proposed by Holland[26] in 1975. The basic idea of GA is to optimize the object by genetic operations, crossover, and mutation, and then select the optimal genetic solutions according to the fitness function.[27] GA has been used extensively in optimization problems. However, conventional GA is usually less efficient than other optimization methods. To improve its computational efficiency, GA is integrated with other evolutionary algorithms. Köker and Cakar[11] used SA as an operator in GA to optimize the inverse kinematics solution of a robotic manipulator. Garg[28] developed a hybrid technique by integrating GA with particle swarm optimization to solve the constrained optimization problems. In this work, a sequential mutation operation is proposed to improve local search ability of GA in order to optimize the inverse kinematics solution of the 6-DOF Stanford MT-ARM robotic manipulator. The adaptive crossover rate and mutation rate are selected for the GA. First, numbers within a range of [0, 1] are randomly generated and then compared with the crossover rate and mutation rate. When the crossover rate and mutation rate are smaller than the randomly generated number, the number will cross and vary.

Proposed algorithm

The proposed ELM-SGA algorithm used to determine inverse kinematics solution of the 6-DOF Stanford MT-ARM robotic manipulator is presented in this section. The main problem in inverse kinematics of robotic manipulators is to determine the joint angle $\theta(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ according to the pose $T(p_x, p_y, p_z, o_x, o_y, o_z, a_x, a_y, a_z, n_x, n_y, n_z)$ of the end effector. Figure 2 shows the flowchart of the proposed ELM-SGA algorithm and it can be seen that the algorithm consists of two stages. In the first stage, ELM is used to compute the preliminary inverse kinematics solution and in the second stage, SGA is used to optimize the preliminary inverse kinematics solution. Unlike conventional neural networks, ELM randomly initializes the weights of the input layer and biases of the hidden layer and the weights of the output layer are updated only during the training process, which significantly

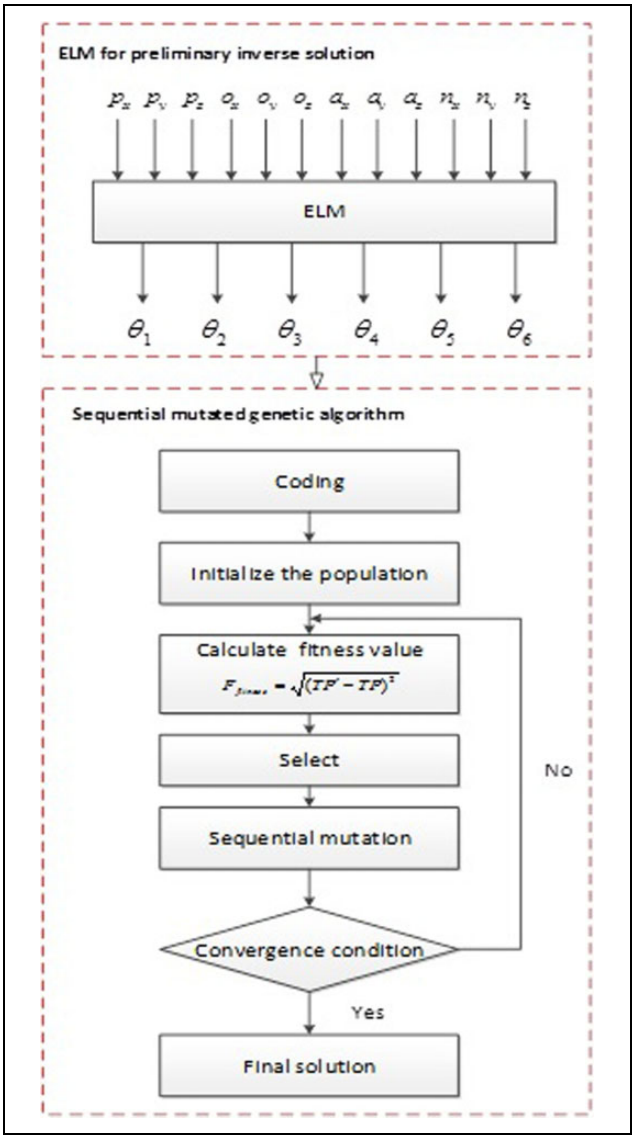

Figure 2. Flowchart of the proposed ELM-SGA algorithm used to compute and optimize the inverse kinematics solution of the robotic manipulator. ELM: extreme learning machine; SGA: sequential mutation genetic algorithm.

reduces the training time. ELM sacrifices the accuracy of the preliminary inverse kinematics solution to a small extent in order to improve the time efficiency. SGA is then used to optimize the preliminary inverse kinematics solution and therefore, it is not necessary for the ELM to attain a highly accurate inverse kinematics solution. It is deemed reasonable to sacrifice the precision of the preliminary inverse kinematics solution to a small extent in order to significantly reduce the training time. The roles of ELM and SGA in the ELM-SGA algorithm will be described in detail in the following subsections.

Computation of the preliminary inverse kinematics solution for 6-DOF Stanford MT-ARM robotic manipulator using ELM

As a single hidden layer neural network, ELM is designed to boost training speed. Unlike conventional neural networks, ELM randomly initializes the weights of input layer and biases of hidden layer only during the training process. This in turn reduces the training process. It is deemed reasonable that ELM sacrifices the accuracy of the preliminary inverse kinematics solution to a small extent in order to improve time efficiency. Solving the inverse kinematics problem using ELM consists of two stages: (1) training and (2) prediction. The training process is more crucial compared with the prediction process because it determines the success or failure of the prediction model. There are two issues that need to be considered during the training process. The first issue is to generate a suitable training data set, whereas the second issue is to select the appropriate number of hidden layers for the ELM model. Once the kinematic parameters of the robotic manipulator are known, a forward kinematics model can be formulated using equations (1) to (14). To ensure rationality of the training set, the joint angles $\theta(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ of the manipulator are randomly initialized. Following this, the forward kinematics model is used to determine the pose of the end effector $T(p_x, p_y, p_z, o_x, o_y, o_z, a_x, a_y, a_z, n_x, n_y, n_z)$ in response to the joint angles $\theta(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ of the manipulator. The number of hidden layer nodes is changed for each training process in order to determine the suitable ELM architecture. The mean square error (MSE) of the test data set is recorded to determine the best ELM architecture. Figure 3 shows the training and prediction processes involved in the ELM model in order to compute the preliminary inverse kinematics solution of the robotic manipulator. Algorithm 2 shows the steps involved in the ELM algorithm in computing the preliminary inverse kinematics solution.

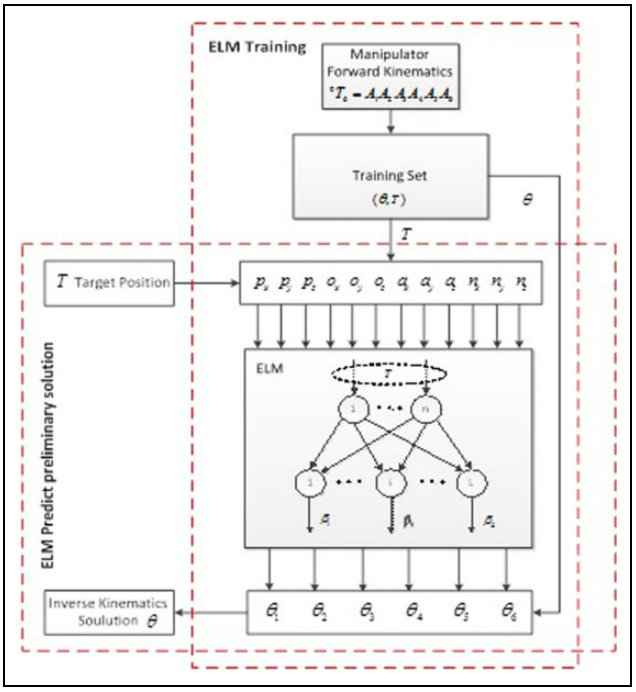

Figure 3. Details of the training and prediction processes in the ELM algorithm. ELM: extreme learning machine.

Algorithm 2. Steps involved in the ELM algorithm to compute the preliminary inverse kinematics solution of the robotic manipulator.

Training process:
Use forward kinematics to compute the training set $\{(\theta^i, T^i)|\theta^i \in R^n, T^i \in R^m, i = 1, 2, \cdots, N\}$.
Choose the activation function $g(x)$ and hidden layer size L.
Randomly initialize weights W_i of the input layer and biases b_i of the hidden layer.
Compute the output matrix of the hidden layer H.
Compute the output weight matrix $\beta : \hat{\beta} = H^{\dagger}T$.
Trained model: $ELM\{\beta, g(x), L\}$.
Prediction process:
Input:$T' = (p_x, p_y, p_z, o_x, o_y, o_z, a_x, a_y, a_z, n_x, n_y, n_z)$
Prediction model: $ELM\{\beta, g(x), L\}$
Output: Preliminary inverse solution $\theta' = (\theta_1, \theta_2, \cdots, \theta_6)$

Optimization of the inverse kinematics solution using SGA

The second stage of the proposed ELM-SGA algorithm involves optimizing the preliminary inverse kinematics solution. Even though GA is widely used for optimization problems, the crossover and mutation operations of GA process randomly, resulting in poor local search capability. Thus, conventional GA is not suitable to optimize highly precise preliminary inverse kinematics solutions, which only require micro adjustments during the optimization process. Hence, in this work, SGA is used to enhance the local search capability of GA, especially at the end of the evolution. In the SGA algorithm, the mutation operation is performed bit by bit from high to low. In each generation, the value of a specific bit is mutated and determined by the fitness function. The weight of the high bit is larger than that of the low bit and therefore, it has a greater contribution to the final solution. Thus, the local search capability

of GA can be enhanced significantly with the SGA algorithm.

The details of the optimization process using SGA are as follows. First, the preliminary inverse kinematics solution $\theta_{\text{preliminary}}(\theta_1, \theta_2, \cdots, \theta_6)$ obtained from the ELM model is encoded into binary format. Each binary format chromosome represents a joint angle of the robotic manipulator. Assuming that the preliminary inverse kinematics solution $\theta_{\text{preliminary}}$ has already achieved high accuracy, only the decimal part of the solution needs to be encoded and optimized. When the joint angle approaches zero, even a small adjustment in precision will change the sign. Hence, a sign bit is assigned at the head of the chromosome to rectify this issue. Second, the original population of the SGA is randomly initialized except for the ones obtained from the ELM model. Third, the fitness value of each individual is calculated according to equations (21) and (22)

$$\begin{aligned} &F_{\text{forward kinematics}}(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) \\ &\quad = (p_x, p_y, p_z, o_x, o_y, o_z, a_x, a_y, a_z, n_x, n_y, n_z) \end{aligned} \tag{21}$$

$$\begin{aligned} F_{\text{fitness}} &= TP_{\text{error}} = \sqrt{(TP' - TP)^2} \\ &= \sqrt{(p'_x - p_x)^2 + (p'_y - p_y)^2 + (p'_z - p_z)^2} \end{aligned} \tag{22}$$

where $TP = (p_x, p_y, p_z)$ represents the coordinate values of the attitude matrix computed by the forward kinematics model and $TP' = (p'_x, p'_y, p'_z)$ represents the coordinate values of the initialization attitude matrix.

The artificial potential field method is chosen as the obstacle avoidance method. The environment in which the robot is located is defined by the potential field. The position information is used to control the robot arm in order to avoid obstacles. If the computed results indicate the possibility of collisions, the experimental results will be discarded. The computed results are retained if there are no collisions.

Next, the convergence criteria are set, which serve as the basis for termination of the SGA algorithm. In this work, the convergence criteria are determined by the maximum number of generations and a predefined tolerance value. When the maximum number of generations or the predefined convergence error is reached, the algorithm terminates the iterative loop. Fourth, in order to ensure the diversity of the population, the best 10 individuals are selected as the candidates. These individuals are then mutated in the order from high to low. For six joints and two mutation results (0 and 1), there are 2^6 different candidate individuals. In each generation, the SGA determines the value of the corresponding bit. The SGA then iterates from step 4 to 3 until the solution converges. Algorithm 3 shows the steps involved in the SGA algorithm used to optimize the inverse kinematics solution of the robotic manipulator. Figure 4 shows the flowchart of the SGA algorithm. Figure 5 shows the details of the mutation between the ith and $(i+1)$th generations for one individual.

Algorithm 3. Steps involved in the SGA algorithm used to optimize the inverse kinematics solution of the robotic manipulator.

Input: Preliminary inverse kinematics solution obtained from the ELM model $\theta = (\theta_1, \theta_2, \cdots, \theta_6)$
Coding: Decimal θ to binary code θ_{binary}. A sign bit, some float bits.
Random initial population except for the input obtained from the ELM model.
While $i++ <$ Length$(\theta_{\text{binary}}) || T_{\text{error}} < \varepsilon$ **do**
Compute the fitness value.
Select the best 10 individual genes for the next generation.
Mutate ith value of each θ_{binary} to generate 64 new genes for every individual.
End
Output: Final inverse kinematics solution obtained by SGA $\theta_{\text{final}} = (\theta_1, \theta_2, \cdots, \theta_6)$.

Results and discussion

The performance of the proposed ELM-SGA algorithm in solving the inverse kinematics problem of the robotic manipulator is validated by performing simulations and experiments, which will be described in this section. The simulations are divided into three parts in order to determine the time efficiency during the training process, optimization process, and computational process. In part 1, the ELM and ANN models with different hidden layer nodes are compared in order to determine the performance of these models in computing the preliminary inverse kinematics solution of the robotic manipulator. The performance of the ELM and ANN models are assessed in terms of the training time and output MSE. In part 2, the proposed SGA and Hybrid[11] algorithms are compared in order to determine the performance of these algorithms in optimizing the preliminary inverse kinematics solution determined using the ELM and ANN models, respectively. The convergence process is assessed based on the end effector error. In part 3, the performance of the ELM, Hybrid, dual redundant camera robot based on genetic algorithm (DRCRB-GA),[12] and proposed ELM-SGA algorithms will be compared in detail. The ELM-SGA algorithm is then applied to the 6-DOF Stanford MT-ARM robotic manipulator in order to validate the performance of the algorithm. The final experiments consist of six experiments to validate the accuracy of different end effector poses as well as a fixed-point grasping experiment. The algorithms are coded using Visual Studio C++.NET 2003 on a personal computer installed with an Intel Core i7-4720HQ @2.60 GHz processor and the application framework uses the Microsoft Basic Class Library version 7.0 (i.e. the MFC library). The time points are set before and after the inverse kinematics solution is computed and the time difference between the two time points is used as an index to evaluate the time efficiency of the algorithms. Experiments are then performed on the 6-DOF Stanford MT-ARM modular robotic manipulator.

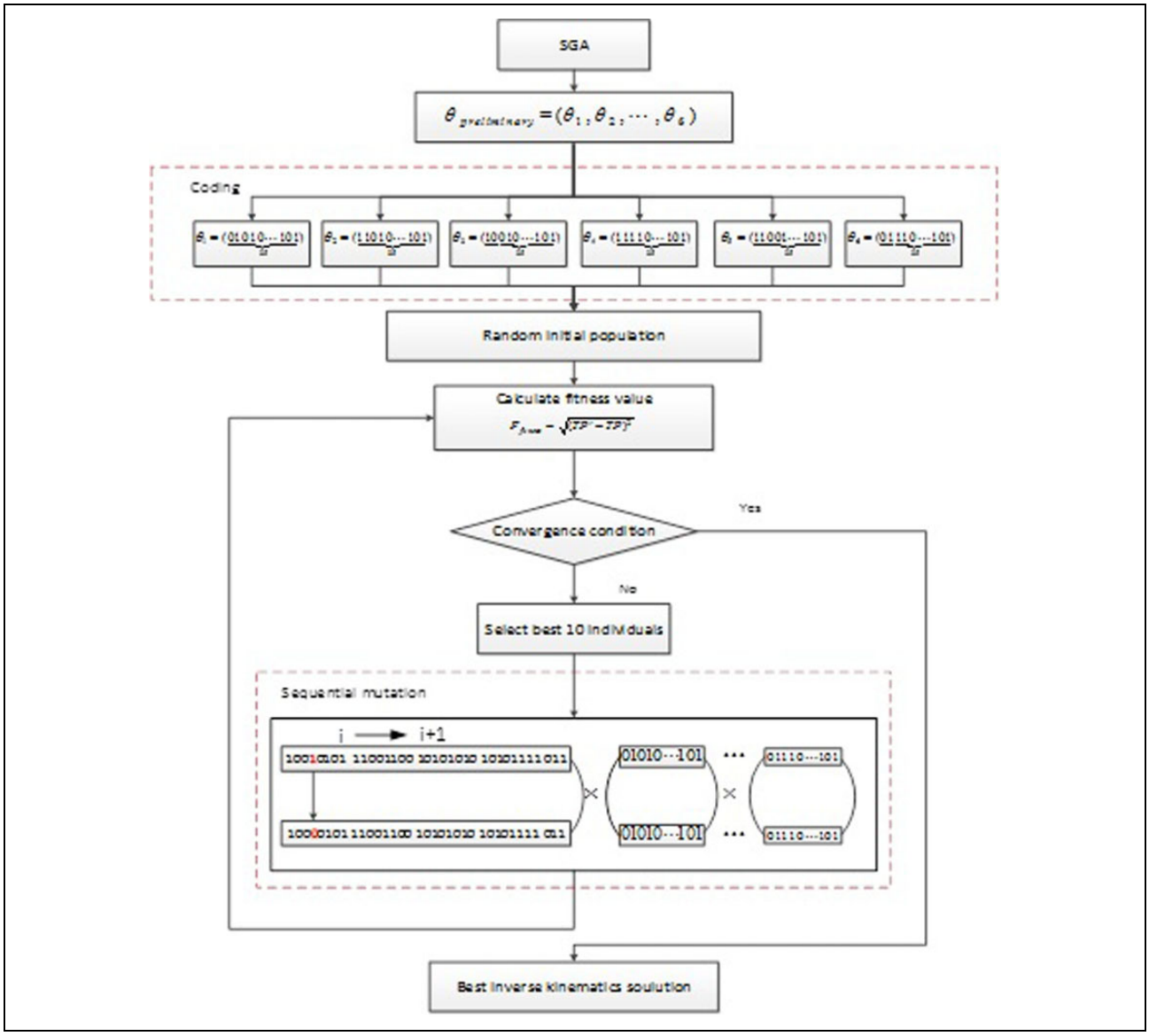

Figure 4. Flowchart of the SGA algorithm. SGA: sequential mutation genetic algorithm.

Comparison between ELM and ANN in computing the preliminary inverse kinematics solution of robotic manipulator

The performance between the ELM and ANN models in computing the preliminary inverse kinematics solution of a 6-DOF Stanford MT-ARM robotic manipulator is presented in this section. Figure 3 shows the processes involved in solving the inverse kinematics problem using the ELM model. In order to obtain the training set, the joint vector $\theta(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ is randomly initialized and then the forward kinematics model is used to generate the mapping from the joint space of the manipulator to the pose of the end effector $T(p_x, p_y, p_z, o_x, o_y, o_z, a_x, a_y, a_z, n_x, n_y, n_z)$ in the Cartesian space. The number of data sets in the simulations is 10,000. The total number of dimensions for each data set is 18 (12 for pose and 6 for joint angles). A total of 8000 data are used to train the ELM model, whereas 2000 data are used for testing. In order to determine the best ELM model, the number of hidden layer nodes of the ELM and ANN models is varied during the training process. The MSE and training time are recorded for each model. The main objectives of the simulations are to determine the best ELM model to determine the preliminary inverse kinematics solution and compare the performance of the ELM and ANN models. The Levenberg–Marquardt algorithm with the sigmoid activation function is used to adjust the parameters in the output layer.

To compare the performance of the ELM and conventional ANN models, seven data sets are recorded, including the training time and MSE. The initial number of nodes is 25 and the number of nodes is increased in steps of 50, as shown in Table 2. When the ELM model is used to compute

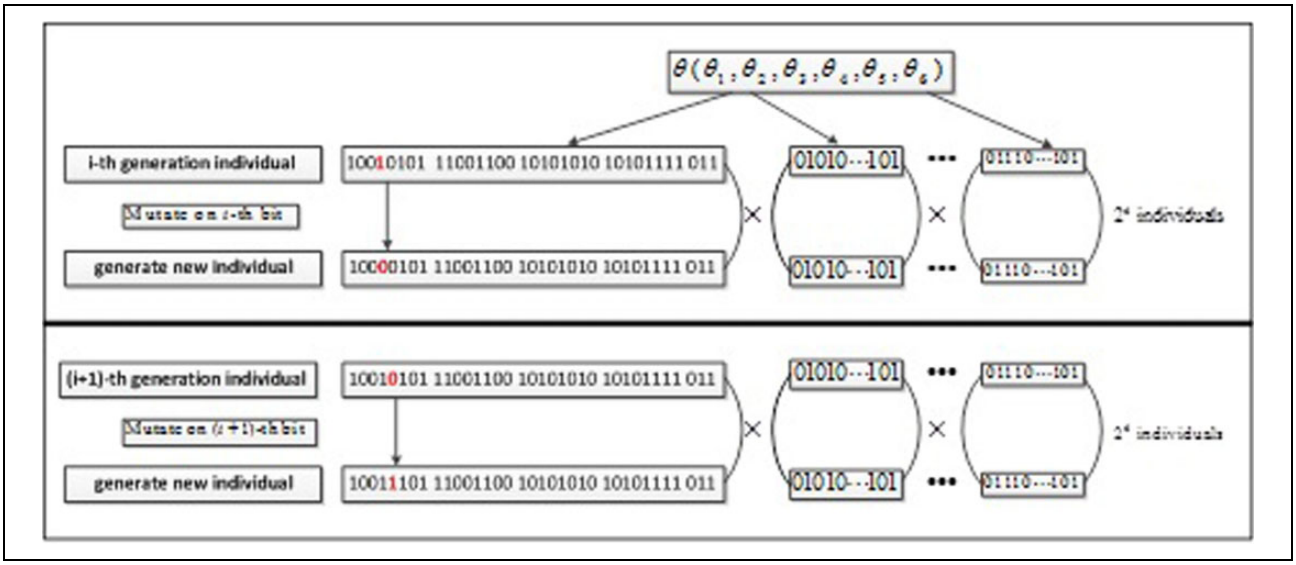

Figure 5. Details of the mutation between the *i*th and $(i + 1)$th generations.

Table 2. Comparison of the MSE and training time between the ANN and ELM models with different number of hidden layer nodes.

No. of nodes		25	75	125	175	225	275	325
ELM	MSE	2.3004	2.1505	2.0559	2.0254	1.9930	**1.9802**	1.9818
	Training time (s)	0.0254	0.0571	0.1035	0.1662	0.2701	**0.3814**	0.4518
ANN	MSE	1.9212	1.7358	1.6932	**1.6513**	1.66364	1.6672	1.6541
	Training time (s)	0.7105	3.5217	8.4615	**13.9127**	22.3125	33.2743	54.9453

MSE: mean square error; ELM: extreme learning machine; ANN: artificial neural network.
Note: The bold values represent the minimum mean square error and the minimum training time in different algorithms.

the inverse kinematics solution of the MT-ARM robotic manipulator, it is found that the MSE of the output joint angle decreases with an increase in the number of hidden layer nodes. The MSE is 1.98028593 when the number of hidden layer nodes is 275. The corresponding training time is 0.3814 s. The simulations are repeated using the ANN model. Likewise, the MSE of the output joint angle decreases with an increase in the number of hidden layer nodes. The lowest MSE (1.65137561) is attained when the number of hidden layer nodes is 175, while the corresponding training time is 13.9127 s. Comparing the results obtained for the ELM model (275 hidden layer nodes) and ANN model (175 hidden layer nodes), it can be seen that the ANN model has a slightly higher precision than the ELM model. However, the training time for the ANN model is 4.36 times the training time for the ELM model. Figure 6(a) and (b) shows the variations of the MSE and training time for the ELM and ANN models. It is evident that the ANN model has a slightly higher accuracy than the ELM model. However, this is negated by the fact that the ANN model is more time-consuming compared with the ELM model, as indicated by the increasing trend in the training time (Figure 6(b)). The simulation results indicate that ELM model can significantly reduce the training time without compromising the accuracy of the solution since the precision of the output is comparable to that for the ANN model.

Optimization of inverse kinematics solution of the robotic manipulator using SGA

As described in the previous section, ELM is used to compute the preliminary inverse kinematics solution of the robotic manipulator. The main purpose of the ELM model is to reduce the training time. However, the end effector error obtained from the ELM model is on the order of a few centimeters, which is undesirable. Hence, SGA is used to optimize the preliminary inverse kinematics solution and the results will be presented and discussed in this section. Even though GA has been used extensively to solve optimization problems, the poor local search capability of GA reduces its convergence speed, especially at the end of the evolution. Hence, the SGA algorithm is used in this work to compensate the disadvantages of classic GA and promote its convergence speed. The SGA algorithm is used to optimize the preliminary inverse kinematics solution obtained from the ELM model. The coding, details of the sequential mutation, and end effector errors are recorded for each generation. The performance of the ELM-SGA and Hybrid[11] algorithms are compared in order to determine their convergence capability.

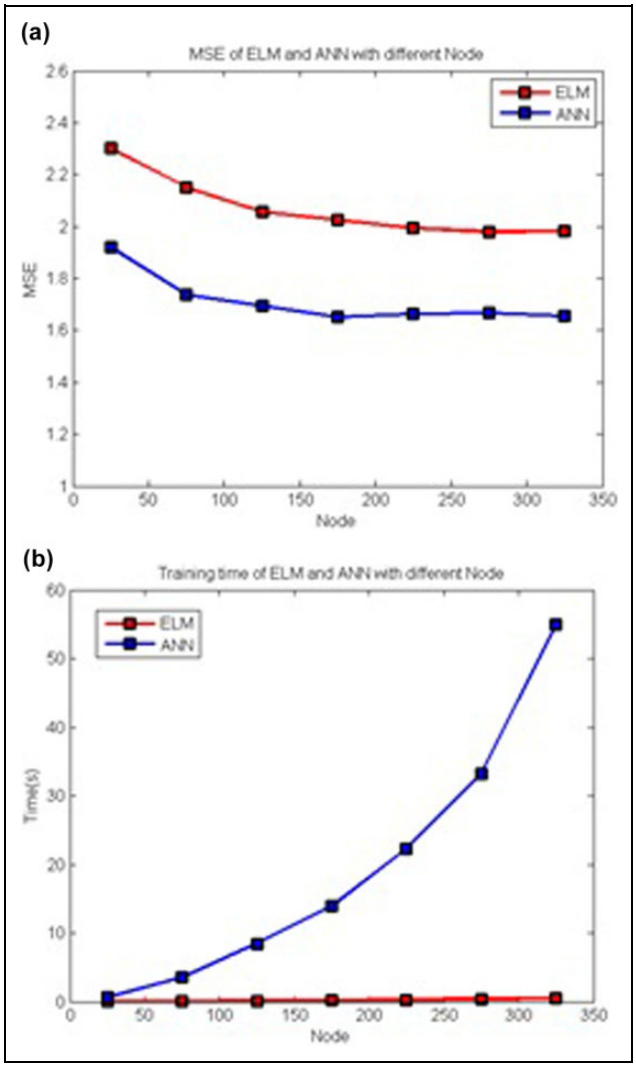

Figure 6. Variations of the (a) MSE and (b) training time of the ANN and ELM models with different number of hidden layer nodes. MSE: mean square error; ANN: artificial neural network; ELM: extreme learning machine.

Table 3 shows the preliminary inverse kinematics solutions obtained by the ELM model and their corresponding binary formats based on the rules described in the "Optimization of inverse kinematics solution of the robotic manipulator using SGA" section. The binary format of each joint angle consists of a sign bit and 34 binary bits converted from the floating parts of each joint angle. After coding, the population is initialized, the fitness value is computed, the convergence criteria are evaluated, and the selection operations are run. The principle of SGA is the same as classic GA and therefore, the procedure will not be elaborated in detail in this section. The inverse kinematics solutions presented in Table 3 are taken as the example in order to demonstrate the sequential mutation process of the SGA, as shown in Figure 7. The evolutions between the fourth and fifth generations are recorded. In the fourth generation, the fourth bit (highlighted in red) mutates and generates 2^6 new individuals. The fitness value is then computed in order to identify which individuals are inherent to the next generation. To ensure diversity of the population, the algorithm does not select the best mutated individual as its offspring for each individual, rather the algorithm chooses a certain number of best fit individuals as the offspring.

A hybrid intelligent algorithm based on ANN, GA, and SA is used to compare with the convergence ability of the ELM-SGA algorithm developed in this work during the optimization process. Köker and Cakar[11] proposed this hybrid intelligent algorithm to solve the inverse kinematics problem of a robotic manipulator in 2016. In the hybrid intelligent algorithm, ANN is used to compute the preliminary inverse kinematics solution. Following this, an improved GA with SA is used to optimize the inverse kinematics solution. In one study,[12] the DRCRB-GA algorithm is proposed by setting two variable parameters, which constrains two redundant DOFs. The structure of the optimized objective function is more flexible than the basic GA and the algorithm is shown to be more efficient. The ELM-SGA, DRCRB-GA, and Hybrid algorithms are used to compute the inverse kinematics solution for the same pose. The end effector errors are recorded for each of these algorithms in each generation.

Figure 8 shows the comparison of the convergence process between the SGA and Hybrid algorithms. The abscissa represents the algebra of Gas, whereas the ordinate represents the end effector error. The blue data markers represent the results of the Hybrid algorithm proposed by Köker and Cakar,[11] whereas the red data markers represent the SGA algorithm proposed in this study. Since the SGA uses the preliminary inverse kinematics solution determined by the ELM model, the accuracy of the solution is slightly lower compared with that for the ANN model. Therefore, the end effector error is larger for the SGA algorithm compared with that for the Hybrid algorithm at the beginning of the optimization process. However, the SGA algorithm has faster convergence speed than the Hybrid algorithm and the end effector error of the SGA algorithm begins to approach zero in the 29th generation. In contrast, the Hybrid algorithm achieves convergence in the 36th generation. This indicates that the proposed ELM-SGA algorithm improves the optimization of the inverse kinematics solution of a robotic manipulator.

Comparison of the inverse kinematics solutions between the ELM, Hybrid, DRCRB-GA, and ELM-SGA algorithms for 10 different random poses

In order to prevent randomness of a single simulation, the ELM, Hybrid, DRCRB-GA, and proposed ELM-SGA algorithms are used to compute the inverse kinematics solutions for 10 random poses. The simulation results are presented in Table 4, including the average end effector

Table 3. Binary formats of the preliminary inverse kinematics solutions obtained from the ELM model.

Joint no.	Inverse kinematics solution	Sign	Floating part	Binary format
1	−40.2685051658	−	2685051658	**1** 0010100000 0000101010 1000110000 1010
2	−83.2850305087	−	2850305087	**1** 0010101001 1110010000 1101000011 1111
3	41.4126445869	+	4126445869	**0** 0011110101 1111010010 0100010010 1101
4	60.0713016643	+	0713016643	**0** 0000101010 0111111111 0001010100 0011
5	74.2925152099	+	2925152099	**0** 0010101110 0101101001 0001110110 0011
6	−56.7779402578	−	7779402578	**1** 0111001111 1011000001 0000110101 0010

ELM: extreme learning machine.
Note: The bold values indicate positive and negative signs, 0 for positive and 1 for negative.

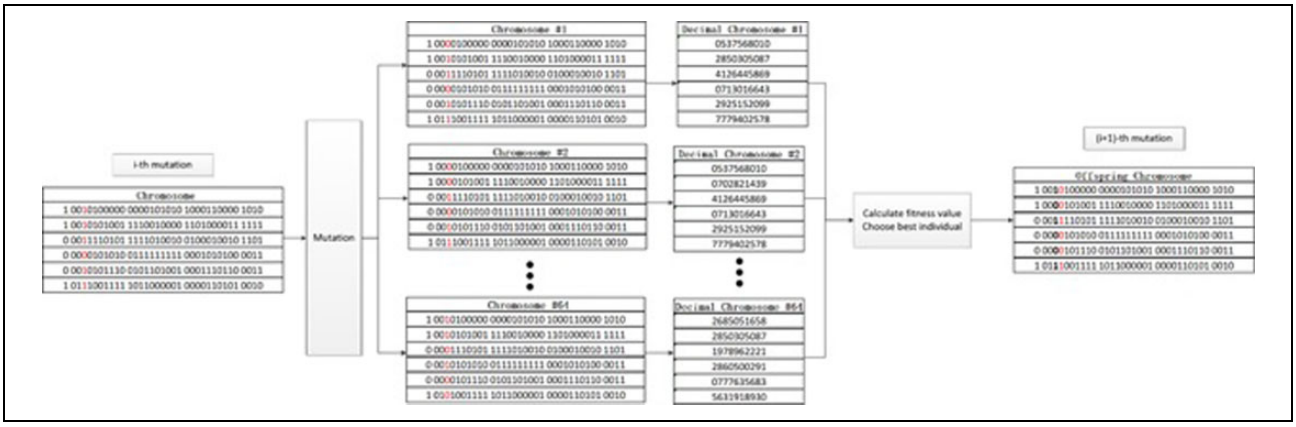

Figure 7. Sequential mutation from the *i*th generation to $(i + 1)$th generation. MSE: mean square error; ANN: artificial neural network; ELM: extreme learning machine.

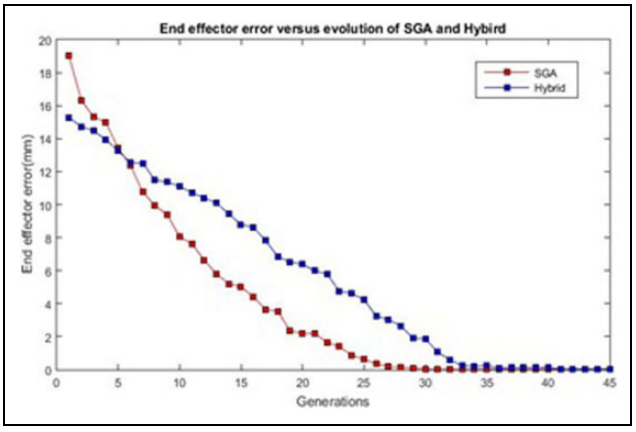

Figure 8. Comparison of the convergence process between the SGA and Hybrid algorithms. SGA: sequential mutation genetic algorithm.

Table 4. Average end effector error and average computational time for the Hybrid, DRCRB-GA, ELM, and ELM-SGA algorithms.

	Average end effector error (mm)	Standard deviation of end effector error	Average computational time (ms)	Standard deviation of computational time
Hybrid	0.0255	0.0084	5.84	0.0696
DRCRB-GA	0.0265	0.0125	5.73	0.1157
ELM	11.9852	4.5668	0.91	0.1118
ELM-SGA	0.0225	0.0081	5.52	0.1297

ELM: extreme learning machine; SGA: sequential mutation genetic algorithm.

error, standard deviation of the end effector error, average computational time, and standard deviation of the computational time. It can be seen from Table 4 that the accuracy of the inverse kinematics solution is significantly higher for the ELM-SGA algorithm compared with other algorithms, while the computational time is less than that those for the Hybrid and DRCRB-GA algorithms. Table 5 shows the results of the Wilcoxon signed test for the Hybrid, DRCRB-GA, ELM, and ELM-SGA algorithms. The significance level is 0.05. If the calculated significance level is less than 0.05, this indicates that the difference in the result between two algorithms is significant. In contrast, if the calculated significance level is greater than 0.05, this indicates that there is no significant difference in the result between two algorithms. Indeed, there is a significant difference in the time efficiency between the proposed ELM-SGA algorithm and other algorithms. In addition, there is a significant difference in error between the ELM-SGA and ELM algorithms. Based on the results, it can be deduced that the ELM-SGA algorithm can achieve the same accuracy as the Hybrid and DRCRB-GA algorithms, but with higher time efficiency compared with other algorithms.

Table 5. Results of the Wilcoxon signed test for the Hybrid, DRCRB-GA, ELM, and ELM-SGA algorithms.

Algorithms	Significance level for end effector error	Significance level for computational time
Hybrid and ELM-SGA	0.5708	**1.7963e-04**
DRCRB-GA and ELM-SGA	0.2730	**0.0073**
ELM and ELM-SGA	**1.8267e-04**	**1.8165e-04**

ELM: extreme learning machine; SGA: sequential mutation genetic algorithm.
Note: The bold values indicate whether there is a significant difference, the bold indicates that there is a significant difference between the two algorithms, and the non-bold indicates that the two algorithms have no significant difference.

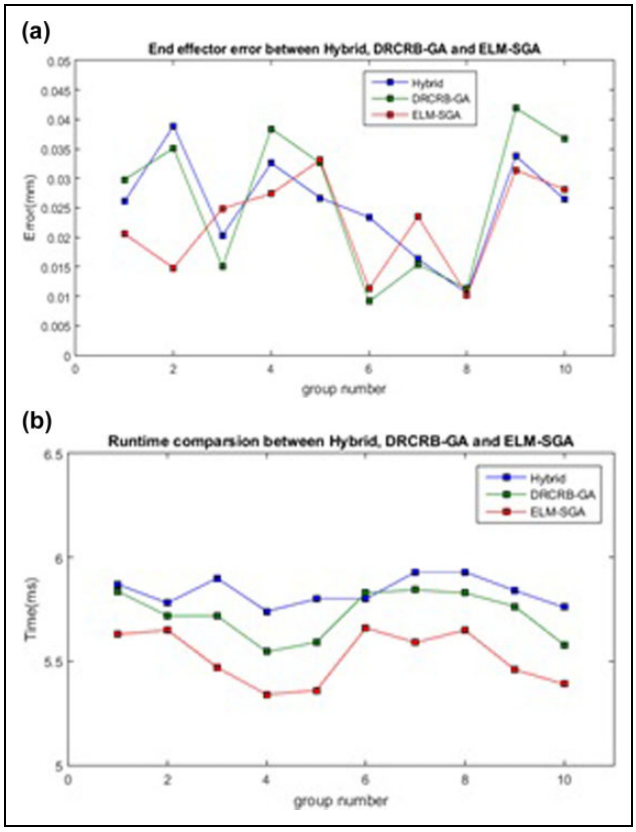

Figure 9. Comparison of the end effector errors and computational time between the Hybrid, DRCRB-GA, and ELM-SGA algorithms. ELM: extreme learning machine; SGA: sequential mutation genetic algorithm.

In order to compare the capability of the algorithms in computing the inverse kinematics solution, the end effector errors and computational time are plotted for the Hybrid, DRCRB-GA, and ELM-SGA algorithms, as shown in Figure 9. The red, blue, and green polylines in Figure 9(a) indicate the end effector errors for the ELM-SGA, Hybrid, and DRCRB-GA algorithms, respectively. The errors of the three algorithms fall within a range of 0.005–0.045 mm. The trajectory of the three polylines is very close and therefore, it can be deduced that the algorithms achieve the same accuracy. Even though the ELM-SGA algorithm can obtain a more precise inverse kinematics solution, the main concern here is the convergence speed of the algorithm. Thus, the convergence criterion is set such that the end effector error is less than 0.05 mm. The red, blue, and green polylines in Figure 9(b) indicate the computational time of the ELM-SGA, Hybrid, and DRCRB-GA algorithms, respectively. It can be seen that the ELM-SGA algorithm is less time-consuming compared with the Hybrid and DRCRB-GA algorithms, indicating that the proposed algorithm achieves faster convergence than other algorithms while attaining the same precision. Based on Table 4, the average computational time of the ELM-SGA algorithm is lower than those for the Hybrid and DRCRB-GA algorithms by 5.5% and 3.7%, respectively.

Experiments using the 6-DOF Stanford MT-ARM robotic manipulator

Two groups of validation experiments are conducted and the proposed ELM-SGA algorithm is used to determine the inverse kinematics solution of the Stanford MT-ARM robotic manipulator, which will be presented in this section. The robotic manipulator has been described in detail in the "Structure and Denavit–Hartenberg parameters of the 6-DOF Stanford MT-ARM robotic manipulator" section. In the first group of experiments, the forward kinematics of the robotic manipulator is used to verify the accuracy of the ELM-SGA algorithm in computing the inverse kinematics solution. In the second group of experiments, the ELM-SGA algorithm is used to compute the inverse kinematics solution in order to realize fixed-point grasping of the robotic manipulator.

End effector errors for six poses of the robotic manipulator obtained from ELM-SGA algorithm. In this section, the proposed ELM-SGA algorithm is applied to the 6-DOF Stanford MT-ARM robotic manipulator to verify the accuracy of the inverse kinematics solution. Each joint angle of the robotic manipulator is set equidistantly from 15° to 90° with a pitch of 15° and the pose data T returned by the robotic manipulator is recorded. The ELM-SGA algorithm is then used to compute the inverse kinematics solution θ for each pose T. The inverse kinematics solutions obtained are set as the joint parameters of the robotic manipulator. The error between the observed pose T' and expected pose T of the end effector (i.e. end effector error) is calculated using equation (22) in order to verify the accuracy of the ELM-SGA algorithm.

The results shown in Table 6 indicate that the ELM-SGA algorithm yields highly precise inverse kinematics solutions for the 6-DOF Stanford MT-ARM robotic manipulator. Figure 10 shows the six poses of the robotic manipulator corresponding to the inverse kinematics solutions tabulated in Table 6. In general, all of the end effector errors are less than 5 mm. The end effector errors are contributed by the following factors: (1) error between the

Table 6. Validation of the inverse kinematics solutions obtained from the ELM-SGA algorithm for six poses.

No.	1	2	3	4	5	6
Angle	15	30	45	60	75	90
Inverse kinematics solution	32.273	61.2974	84.6442	100.3162	104.8313	90.0085
	−15.0027	−30.0001	−45.0064	−59.9931	−74.9983	90.0086
	−15.0023	−29.9999	−45.0064	−59.9929	−74.9981	89.9912
	32.1415	67.3406	−72.4333	−24.4772	32.746	90.0085
	25.8107	44.2927	−51.5755	−51.4838	−58.6461	90.0086
	27.3888	52.8848	−101.351	−88.6134	−92.1772	89.9914
Target position	−424.235	−622.713	−553.477	−365.03	−239.187	−242
	744.566	448.001	202.752	137.759	209.583	278
	444.082	629.244	812.49	896	851.382	722.5
Real position	−421.364	−621.12.0	−552.189	−366.183	−239.539	−242.083
	745.96	452.737	198.785	142.073	209.956	279.884
	444.011	628	813.166	895.988	849.692	722.658
Error	3.19	4.78	4.23	4.47	1.76	1.89

ELM: extreme learning machine; SGA: sequential mutation genetic algorithm

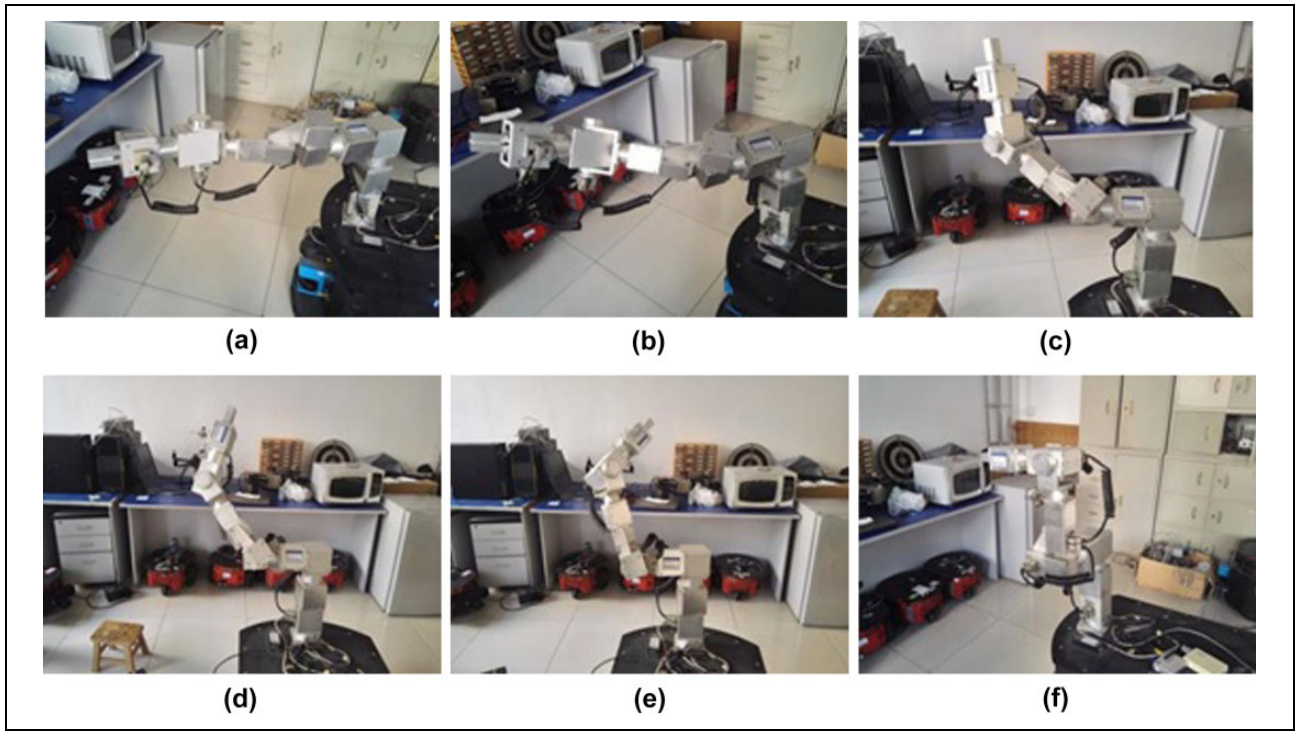

Figure 10. Validation of the inverse kinematics solutions obtained from the ELM-SGA algorithm for six poses using the 6-DOF Stanford MT-ARM robotic manipulator. DOF: degree of freedom; ELM: extreme learning machine; SGA: sequential mutation genetic algorithm.

inverse kinematics solution obtained by the ELM-SGA algorithm and ideal inverse kinematics solution whose end effector error relative to the target position equals zero and (2) accuracy of the arm-drive motor as well as drawbacks of the software control system. The input parameters of the MT-ARM robotic manipulator must be rounded to the nearest integer, which is the major contributor of the errors listed in Table 6. In summary, the error contributed by the ELM-SGA algorithm is less than 5 mm and therefore, it can be deduced that the proposed algorithm is capable of giving highly precise inverse kinematics solutions.

Fixed-point grasping experiments based on the ELM-SGA algorithm. Fixed-point grasping experiments are conducted to verify the accuracy of the inverse kinematics solutions obtained from the ELM-SGA algorithm. The experimental procedure is described as follows. First, a set of joint angles $\theta_0 = (45, -110, 40, 0, 30, 180)$ is randomly selected as the joint parameters of the MT-ARM robotic manipulator and the position $T_0 = (-312.698,\ 550.647,\ -95, 778)$ of the end effector is recorded. Following this, a black rubber block is placed at T_0, as shown in Figure 11(a). The ELM-SGA computes the inverse kinematics solution

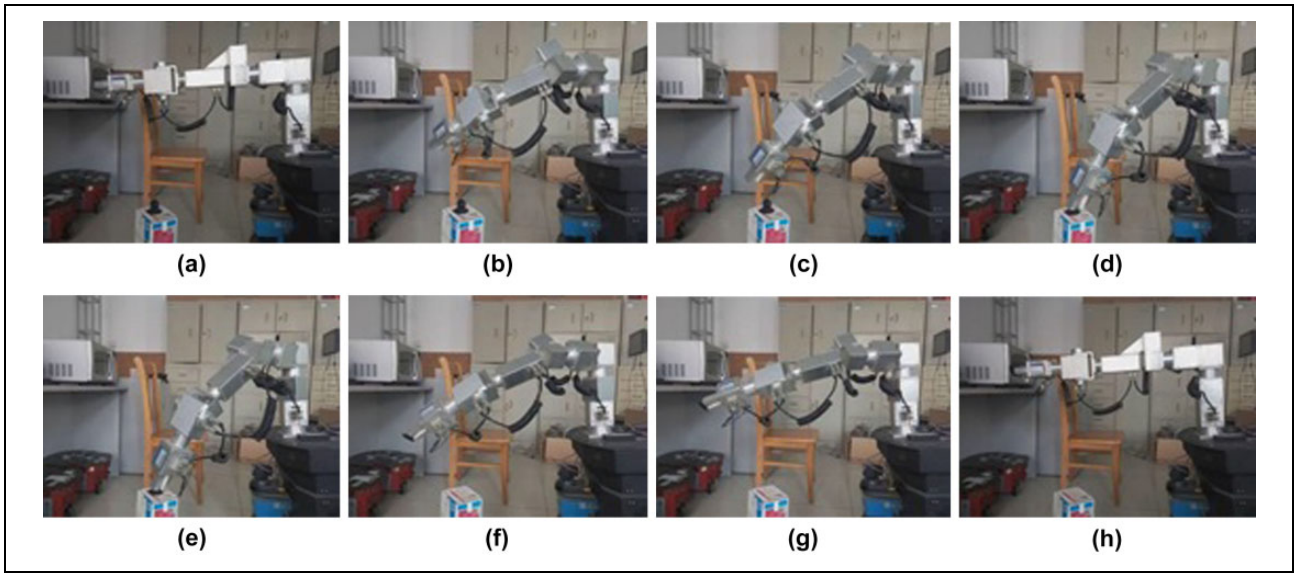

Figure 11. Fixed-point grasping experiments using the 6-DOF Stanford MT-ARM robotic manipulator. DOF: Degree of freedom.

Table 7. Parameters of the fixed-point grasping experiments.

		Target	Inverse kinematics solution	Input
Joint angle	1	45	45.0225	45
	2	−110	69.9285	70
	3	40	−40.0233	−40
	4	0	0.1585	0
	5	30	−29.9699	−30
	6	180	−0.0842	0
Position	p_x	−312.698	−312.696	−312.076
	p_y	550.647	550.6554	550.648
	p_z	−95.778	−95.7941	−95.929
Error	(mm)	—	0.0177	0.64

$\theta_{\mathrm{ELM-SGA}}$ of the target position T_0. Following this, $\theta_{\mathrm{ELM-SGA}}$ is set as the input parameter of the MT-ARM robotic manipulator to perform the grasping operation. Table 7 shows the details of the data. The theoretical error is 0.0177, whereas the actual error is 0.6400 mm. The error is within the allowable range of the grasping operation. Figure 11(a) to (h) shows the fixed-point grasping experiments using the 6-DOF Stanford MT-ARM robotic manipulator.

Conclusions

An intelligent ELM-SGA algorithm has been proposed in this article in order to obtain the inverse kinematics solutions of a 6-DOF robotic manipulator. This is the first time ELM is used to compute the inverse kinematics solution of a 6-DOF robotic manipulator. The proposed ELM-SGA algorithm computes the inverse kinematics solution of the robotic manipulator in two stages. In the first stage, ELM is used to compute the preliminary inverse kinematics solution. In the second stage, SGA is used to optimize the preliminary inverse kinematics solution obtained from the ELM model. ELM randomly initializes the weights of the input layer and biases of the hidden layer, which greatly improves the training speed. Sequential mutation significantly improves the local searching capability of the GA. The ELM-SGA algorithm sacrifices the accuracy of preliminary inverse kinematics solution to a small extent, which significantly reduces the training time. The simulation and experimental results indicate that the ELM-SGA algorithm significantly improves the computational time without compromising the accuracy of the inverse kinematics solution of the robotic manipulator.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work is supported by Zhejiang Provincial Natural Science Foundation of China (no. LY18F030018, LZ15F020004), Natural Science Foundation of China (no. 51376055, 61272311), 521 Plan of Zhejiang Sci-Tech University, and Science and Technology Plan of Zhejiang Province (no. 2017C31017).

ORCID iD

Zhiyu Zhou http://orcid.org/0000-0003-4487-8192

References

1. Xu W, Mu Z, Liu T, et al. A modified modal method for solving the mission-oriented inverse kinematics of hyper-redundant space manipulators for on-orbit servicing. *ACTA Astronaut* 2017; 139: 54–66.

2. Gu J, Wang H, Pan Y, et al. Neural network based visual servo control for CNC load/unload manipulator. *OPTIK* 2015; 126(23): 4489–4492.
3. Carlos LF, Jesus HB, Alanis AY, et al. Inverse kinematics of mobile manipulators based on differential evolution. *Int J Adv Robot Syst* 2018; 15(1): 1729881417752738.
4. Hasan AT, Ismail N, Hamouda AMS, et al. Artificial neural network-based kinematics Jacobean solution for serial manipulator passing through singular configurations. *Adv Eng Softw* 2010; 41: 359–367.
5. Martin JA, Lope JD, and Santos M. A method to learn the inverse kinematics of multi-link robots by evolving neuro-controllers. *Neurocomputing* 2009; 72: 2806–2814.
6. Duguleana M, Barbuceanu FG, Teirelbar A, et al. Obstacle avoidance of redundant manipulators using neural networks based reinforcement learning. *Robot Comput Integr Manuf* 2012; 28: 132–146.
7. Tejomurtula S and Kak S. Inverse kinematics in robotics using neural networks. *Inform Sci* 1999; 116: 147–164.
8. Karlik B and Aydin S. An improved approach to the solution of inverse kinematics problems for robot manipulators. *Eng Appl Artif Intel* 2000; 13: 159–164.
9. Kalra P, Mahapatra PB, and Aggarwal DK. An evolutionary approach for solving the multimodal inverse kinematics problem of industrial robots. *Mech Mach Theory* 2006; 41: 1213–1229.
10. Bingul Z, Ertunc HM, and Oysu C. Applying neural network to inverse kinematic problem for 6R robot manipulator with offset wrist. In: *Adaptive and natural computing algorithms*, Istanbul, Turkey, 15–17 December 2005, pp. 112–115. Vienna: Springer.
11. Köker R and Cakar T. A neuro-genetic-simulated annealing approach to the inverse kinematics solution of robots: a simulation based study. *Eng Comput* 2016; 32: 553–565.
12. Zhang LB, He JJ, and Wang S. Inverse kinematic solutions of dual redundant camera robot based on genetic algorithm. *Math Probl Eng* 2017; 2017: 1–10.
13. Köker R. A neuro-genetic approach to the inverse kinematics solution of robotic manipulators. *Sci Res Essays* 2011; 6(13): 2784–2794.
14. Köker R. A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization. *Inform Sci* 2013; 222: 528–543.
15. Köker R. A neuro-simulated annealing approach to the inverse kinematics solution of redundant robotic manipulators. *Eng Comput* 2013; 29(4): 507–515.
16. Tabendeh S, Clark C, and Melek W. A genetic algorithm approach to solve for multiple solutions of inverse kinematics using adaptive niching and clustering. In: IEEE congress on evolutionary computation, Vancouver, Canada, 2006, pp. 1815–1822.
17. Pham DT, Castellan M, and Fahmy AA. Learning the inverse kinematics of a robot manipulator using the bees algorithm. In: *Proceedings of the IEEE international conference on industrial informatics*, Daejeon, Korea, 13–16 July 2008, pp. 493–498. Institute of Electrical and Electronics Engineers Inc.
18. Jiang H, Liu S, and Zhang B. Inverse kinematics analysis for 6 degree-of-freedom modular manipulator. *J Zhejiang Univ Eng Sci* 2010; 44(7): 1348–1354.
19. Zhang D and Lei J. Kinematic analysis of a novel 3-DOF actuation redundant parallel manipulator using artificial intelligence approach. *Robot Comput Integr Manuf* 2011; 27: 157–163.
20. Lian RJ. Grey-prediction self-organizing fuzzy controller for robotic motion control. *Inform Sci* 2012; 202: 73–89.
21. Huang GB, Zhu QY, and Siew CK. Extreme learning machine: theory and applications. *Neurocomputing* 2006; 70: 489–501.
22. Nian R, He B, and Amaury L. 3D object recognition based on a geometrical topology model and extreme learning machine. *Neural Comput Appl* 2013; 22(3-4): 427–433.
23. Lu S, Qiu X, Shi J, et al. A pathological brain detection system based on extreme learning machine optimized by bat algorithm. *CNS Neurol Disord Drug Targets* 2017; 16(1): 23–29.
24. Huang G, Huang GB, and Song S. Trends in extreme learning machines: a review. *Neural Netw* 2015; 61: 32–48.
25. Zhang YD, Zhao G, Sun J, et al. Smart pathological brain detection by synthetic minority oversampling technique, extreme learning machine, and Jaya algorithm. *Multimed Tools Appl* 2017: 1–20. DOI: 10.1007/s11042-017-5023-0.
26. Holland JH. *Adaptation in natural and artificial systems*. Cambridge: MIT Press, 1975.
27. Kuo RY, Syu YJ, Chen ZY, et al. Integration of particle swarm optimization and genetic algorithm for dynamic clustering. *Inform Sci* 2012; 195: 124–140.
28. Garg H. A hybrid PSO-GA algorithm for constrained optimization problems. *Appl Math Comput* 2016; 274: 292–305.