

Article

Learning the Kinematics of a Manipulator Based on VQTAM

Luo Lan ^{1,*} , Hou Li ¹, Wu Yang ¹, Wei Yongqiao ² and Zhang Qi ³

¹ School of Mechanical Engineering, Sichuan University, Chengdu 610065, China; houlaoshishiyanshi@163.com (H.L.); walkerwuy@outlook.com (W.Y.)

² School of Mechanical & Electromechanical Engineering, Lanzhou University of Technology, Lanzhou 730050, China; scuwyq@163.com

³ School of Intelligent Manufacturing, Panzhihua University, Panzhihua 617000, China; pzhuzq@126.com

* Correspondence: studyluolan@163.com

Received: 24 February 2020; Accepted: 6 March 2020; Published: 2 April 2020



Abstract: The kinematics of a robotic manipulator is critical to the real-time performance and robustness of the robot control system. This paper proposes a surrogate model of inverse kinematics for the serial six-degree of freedom (6-DOF) robotic manipulator, based on its kinematics symmetry. Herein, the inverse kinematics model is derived via the training of the Vector-Quantified Temporal Associative Memory (VQTAM) network, which originates from Self-Organized Mapping (SOM). During the processes of training, testing, and estimating of this neural network, a priority K-means tree search algorithm is utilized, thus improving the training efficacy. Furthermore, Local Linear Regression (LLR), Local Weighted Linear Regression (LWR), and Local Linear Embedding (LLE) algorithms are, respectively, combined with VQTAM to obtain three improvement algorithms, all of which aim to further optimize the prediction accuracy of the networks for subsequent comparison and selection. To speed up the solving of the least squared equation, which is common among the three algorithms, Singular Value Decomposition (SVD) is introduced. Finally, data from forward kinematics, in the form of the exponential product of a motion screw, are obtained, and are used for the construction and validation of the VQTAM neural network. Our results show that the prediction effect of the LLE algorithm is better than others, and that the LLE algorithm is a potential surrogate model to estimate the output of inverse kinematics.

Keywords: robot kinematics; machine learning; VQTAM; priority K-means tree search algorithm; local linearization; SVD

1. Introduction

The robotic manipulator has been commonly used in various fields. Robot control is critical for high-speed and high-precision robot motion, and it is based on the kinematics of robots. Kinematics includes two aspects: forward kinematics and inverse kinematics. Inverse kinematics describes the mapping from the state of effector to the state of actuator [1]. The key to kinematics is establishing the mapping relationship of the manipulator. For the serial robot, the input is the rotation angle of each joint, and the output is the pose of the end effector.

Generally, the inverse kinematics analysis of serial robots requires highly complex calculations, which affect the response speed and work efficiency of the robot controller. The same problem also exists in the forward kinematics of parallel robots. Three types of methods are typically used for the inverse kinematics of serial robots: geometric [2–4], algebraic [5–7], and iterative algorithms [8–12]. However, algebraic methods cannot always derive a closed-form solution. In addition, the use of geometric methods is limited by the precondition, and the initial position influences the convergence speed of

iterative algorithms. Using these three methods, the inverse kinematics of the robot cannot be obtained quickly and effectively [1]. Therefore, much research is focused on how to simplify the solution of robot kinematics. Intelligent algorithms and machine learning are combined in solving the kinematics of the robot. Simpler mapping models of the kinematic input and output are established to replace the complex analytical solution. Robot kinematics has symmetry, meaning that the input and output of forward kinematics are the output and input of inverse kinematics, respectively. The manipulator's state of forward kinematics and inverse kinematics correspond one by one. According to this symmetry, the forward kinematics of the robotic manipulator can be used to generate sample data to train its inverse kinematics model. Hargis, B.E. et al. [13] trained an Artificial Neural Network (ANN) to obtain the inverse kinematics solution. Lou, Y. F. et al. [14] used a hybrid ANN to solve the inverse kinematics and realized the path control of a two-degree of freedom (2-DOF) manipulator. Akanshu Mahajan et al. [15] established an unsupervised learning algorithm based on a neural network, which approximately replaced the inverse kinematics model of the 2-DOF serial robot. Ben Kenwright [16] solved inverse kinematic problems by combining an ANN and a differential evolutionary algorithm. Koji Kinoshita [17] regarded the inverse kinematics of a 3-DOF manipulator as an optimization problem, and solved it using a Particle Swarm Optimization (PSO) algorithm. Rui Ting, Zhu et al. [18] proposed a method based on PSO to obtain an optimal solution using kinematic equations directly. Other agent models or intelligent optimization algorithms, such as evolutionary fuzzy extreme learning machine, support vector regression, RBF (Radial Basis Function) networks, Kohonen self-organizing map, modular neural network systems, sequential mutation genetic algorithms, etc., have been applied in robot kinematics [1,19–25]. When the DOF of the robotic manipulator is increased, the corresponding calculation amount increases in a geometric series. Therefore, the existing methods are not suitable for the inverse kinematics of a 6-DOF robotic manipulator. At the same time, the existing achievements are mostly based on the neural network model. The selection of hyperparameters of the neural network has a great impact on the prediction accuracy. Therefore, adjusting the parameters is time-consuming. Another problem of the neural network model is that it easily converges to the local optimum in the training process.

Robot kinematics can be regarded as a Nonlinear Autoregressive Model with Exogenous Input (NARX). In [26], the forward kinematics of parallel manipulators is solved by combining the wavelet network and NARX method. However, the validity of this method still relies on parameter initialization and the backpropagation learning process.

In [27], a self-organizing map structure was proposed, namely, the Vector-Quantified Temporal Associative Memory (VQTAM). Barreto, G. D. and Araujo, A. F. R. simulated a one-dimensional network, which was compared with the multilayer perceptron (MLP) and RBF networks. Furthermore, a VQTAM network with higher dimensions may behave better. Thus, in this paper, we train two-dimensional networks to research their performance. The VQTAM algorithm does not depend on the adjustment of hyperparameters, but on Self-Organized Mapping (SOM) to achieve topology preservation [27]. As an extension of SOM, it can be applied to the mapping between two spaces. The mathematical foundation of VQTAM is nonlinear manifold embedding. This method is suitable for time series modeling and the prediction of nonlinear systems. Depending on the self-organizing mapping structure, the discrete mapping relationship can effectively realize output estimation according to exogenous input [27–29]. Because spaces of VQTAM are discrete, a network with a large number of neurons is needed to achieve high-precision prediction. To improve the computational efficiency, it is necessary to reduce the number of VQTAM neurons. Based on VQTAM and Local Linear Embedding (LLE), the input is locally linearized in the neighborhood. Improved local linear algorithms of VQTAM are proposed to solve the inverse kinematics of a 6-DOF robot efficiently and accurately.

The paper discusses the complexity of the inverse kinematics solution when applied to serial manipulators. Meanwhile, the kinematics of parallel manipulators would be more complex, and it could also be effectively solved by the methods provided in this paper. This part of the work will be covered in the authors' future research.

2. Research Methodology

A robotic manipulator is a typical open-loop mechanism. Its forward kinematics can be achieved by coordinate transformation, D-H parameters, screw theory, and other mathematical tools. The result can be obtained quickly. However, the inverse solution of serial robots is very complex. The robotic manipulator is a nonlinear system. Generally, solving the inverse kinematics and dynamics accurately and conveniently is difficult.

Thus, the poses corresponding to different joint angles can be obtained using forward kinematics. The data obtained can be used to train the surrogate model for inverse kinematics based on its symmetry. In this paper, a VQTAM neural network is introduced to replace the complex calculation of inverse kinematics. The research content and structure of this paper are shown in Figure 1.

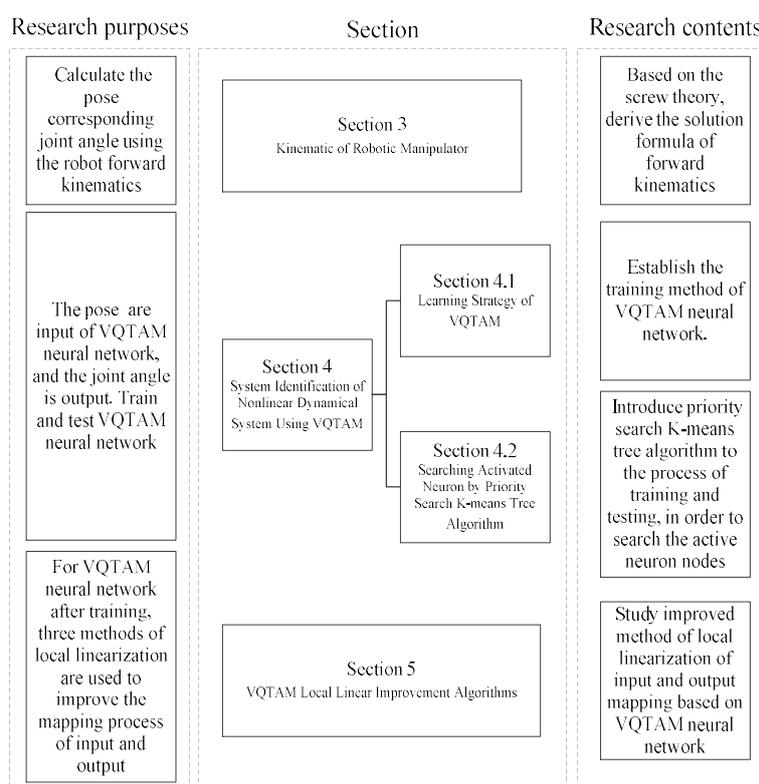


Figure 1. Research content and structure.

In Section 3, the forward kinematics is studied. Data for training are sampled from forward kinematics, meaning that each pose of the end effector corresponds to a group of joint angles.

In Section 4, the nonlinear system identification method based on VQTAM is studied, and the training method of the VQTAM neural network is established. In the process of VQTAM training and testing, it is critical to search activated neuron nodes. Moreover, the priority search K-means tree algorithm is introduced for this step.

In Section 5, the improved method of local linearization for the mapping process based on the VQTAM neural network is studied.

3. Kinematics of the Robotic Manipulator

In screw theory, any motion of a rigid body can be decomposed into rotational motion around a straight line and translational motion along the line. That is to say, the motion of a rigid body can be regarded as spiral motion.

Using screw theory has the following advantages in rigid body kinematics [5,30,31]:

- (1) The 6-DOF parameter is used to describe the pose relationship between two adjacent coordinate systems completely, thus avoiding the lack of completeness in the D-H model.
- (2) The global coordinate system is used to describe the motion state of a rigid body, which overcomes the singularity of the D-H model.
- (3) The motion characteristics of rigid bodies can be clearly described from a global perspective, thus simplifying the analysis of complex mechanisms and avoiding the abstraction of mathematical symbols.

To sum up, the advantages of screw include a clear geometric concept, clear physical meaning, simple expression, and efficient algebraic operation in the kinematics of the robotic manipulator.

According to Chasles' theorem, the spiral motion of a rigid body can be expressed in the form of exponential coordinates of the screw. For the 6-DOF robotic manipulator (RRRRRR type) shown in Figure 2, the relative inertial coordinate system of each joint is established according to screw theory, as shown in Equation (1).

$$\begin{aligned} \omega_1 &= \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & \omega_2 &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & \omega_3 &= \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \\ \omega_4 &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & \omega_5 &= \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} & \omega_6 &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \end{aligned} \quad (1)$$

The coordinates of the points on the axes of each joint are shown in Equation (2).

$$\begin{aligned} r_1 &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} & r_2 &= \begin{bmatrix} x_2 \\ 0 \\ z_2 \end{bmatrix} & r_3 &= \begin{bmatrix} x_3 \\ 0 \\ z_3 \end{bmatrix} \\ r_4 &= \begin{bmatrix} x_4 \\ 0 \\ z_4 \end{bmatrix} & r_5 &= \begin{bmatrix} x_5 \\ 0 \\ z_5 \end{bmatrix} & r_6 &= \begin{bmatrix} x_6 \\ 0 \\ z_6 \end{bmatrix} \end{aligned} \quad (2)$$

Then, the unit screw of each joint can be obtained as shown in Equation (3).

$$\begin{aligned} \xi_1 &= \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} & \xi_2 &= \begin{bmatrix} 1 \\ 0 \\ 0 \\ -z_2 \\ 0 \end{bmatrix} & \xi_3 &= \begin{bmatrix} 0 \\ 1 \\ 0 \\ z_3 \\ 0 \\ -x_3 \end{bmatrix} \\ \xi_4 &= \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ -z_4 \\ 0 \end{bmatrix} & \xi_5 &= \begin{bmatrix} 0 \\ 1 \\ 0 \\ z_5 \\ 0 \\ -x_5 \end{bmatrix} & \xi_6 &= \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ -z_6 \\ 0 \end{bmatrix} \end{aligned} \quad (3)$$

ω_i can be mapped to the rigid body rotation transformation matrix to describe the rigid body rotation transformation in three-dimensional space, as shown in Equation (4). v_i is the vector comprising the third, fourth, and fifth components of the screw, ξ_i .

$$\omega_i = \begin{bmatrix} \omega_i^1 \\ \omega_i^2 \\ \omega_i^3 \end{bmatrix} \mapsto \hat{\omega}_i = \begin{bmatrix} 0 & -\omega_i^3 & \omega_i^2 \\ \omega_i^3 & 0 & -\omega_i^1 \\ -\omega_i^2 & \omega_i^1 & 0 \end{bmatrix} \quad (4)$$

ξ_i can be mapped to a rigid body transformation matrix, describing the rigid body transformation in three-dimensional space, as shown in Equation (5).

$$\xi_i = [\omega_i^T; v_i^T]^T \mapsto \hat{\xi}_i = \begin{bmatrix} \hat{\omega}_i & v_i \\ \mathbf{0} & 0 \end{bmatrix} \quad (5)$$

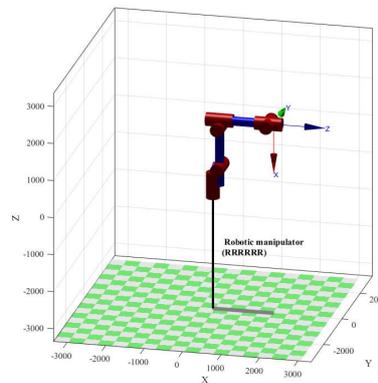


Figure 2. Six-degree of freedom (6-DOF) robotic manipulator (RRRRRR type).

As a special Euclidean group $SE(3)$, the rigid body transformation matrix in three-dimensional space can be regarded as a six-dimensional manifold, and the points on the manifold correspond to the rigid body transformation. $SE(3)$ comprises all possible rigid body motion mappings. Then, $\hat{\xi}_i$ is the Lie algebra $se(3)$ of $SE(3)$. According to the isomorphism relation shown in Equation (6), the mapping relation between the Plucker coordinate form and matrix $\hat{\xi}_i$ can be defined and expressed using operators \vee and \wedge , as shown in Equations (7) and (8).

$$\hat{\xi}_i = \begin{bmatrix} \hat{\omega}_i & v_i \\ \mathbf{0} & 0 \end{bmatrix} \in se(3) \mapsto [\omega_i^T; v_i^T]^T \in \mathbb{R}^6 \quad (6)$$

$$\begin{bmatrix} \hat{\omega}_i & v_i \\ \mathbf{0} & 0 \end{bmatrix}^{\vee} = \begin{pmatrix} \omega_i \\ v_i \end{pmatrix} \quad (7)$$

$$\begin{pmatrix} \omega_i \\ v_i \end{pmatrix}^{\wedge} = \begin{bmatrix} \hat{\omega}_i & v_i \\ \mathbf{0} & 0 \end{bmatrix} \quad (8)$$

The helical motion of a rigid body $g \in se(3)$ is expressed in the form of exponential coordinates of the motion screw, as shown in Equation (9) [30,31].

$$g_i = e^{\hat{\xi}_i \theta_i} = \begin{cases} \begin{bmatrix} e^{\hat{\omega}_i \theta_i} & (\mathbf{I} - e^{\hat{\omega}_i \theta_i})(\omega_i \times v_i) + \theta_i \omega_i \omega_i^T v_i \\ \mathbf{0} & 1 \end{bmatrix}, \omega_i \neq 0 \\ \begin{bmatrix} \mathbf{I} & \theta_i v_i \\ \mathbf{0} & 1 \end{bmatrix}, \omega_i = 0 \end{cases} \quad (9)$$

Combining the motion of each joint, the exponential product equation of the forward kinematics of the robot is obtained, as shown in Equation (10) [30,31].

$$g_{st}(\theta) = e^{e^{\hat{\xi}_1 \theta_1}} e^{e^{\hat{\xi}_2 \theta_2}} \dots e^{e^{\hat{\xi}_6 \theta_6}} g_{st}(\mathbf{0}) \quad (10)$$

$$\theta = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]$$

$g_{st}(\mathbf{0})$ is the pose in the initial state, and $g_{st}(\theta)$ is the pose of the manipulator, where θ is the angle vector of each joint relative to the initial state. Given the joint screw and initial pose, the forward kinematics solution of the robotic manipulator can be obtained.

The joint screw parameters of the robotic manipulator are given in Table 1. The pose shown in Figure 1 is determined to be the initial pose. Poses corresponding to different joint rotation angles can be obtained by Equation (10). Some data for computation validation are shown in Table 2 and Figure 3, which were drawn using the robotics MATLAB toolbox by Peter Corke. In Table 2, (x,y,z) represents the position coordinate of the end actuator, and (u,v,w) is the attitude in Euler angle in radians.

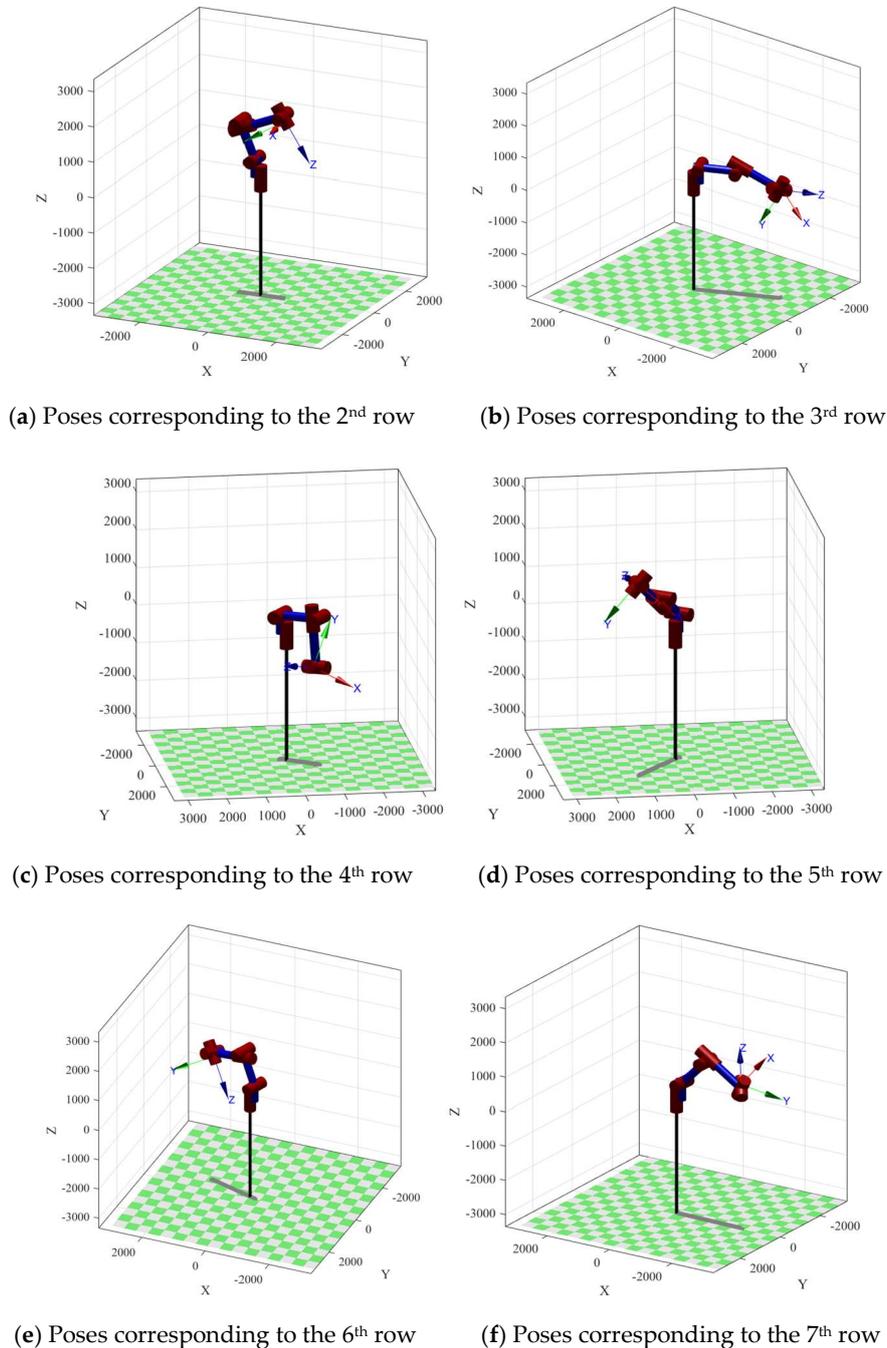


Figure 3. Poses corresponding to Table 2.

Table 1. Joint screw parameters.

parameter value	x_2 (mm) 175	x_3 (mm) 175	x_4 (mm) 175	x_5 (mm) 1445	x_6 (mm) 1445
parameter value	z_2 (mm) 495	z_3 (mm) 1590	z_4 (mm) 1765	z_5 (mm) 1765	z_6 (mm) 1765

Table 2. Poses corresponding to different joint rotation angles.

θ_1	θ_2	θ				x (mm)	y (mm)	$g_{st}(\theta)$ z (mm)	u	v	w
		θ_3	θ_4	θ_5	θ_6						
0	0	0	0	0	0	1580	0	1765	-2.81	-1.57	2.81
-3.14	0.39	-3.14	0.03	-1.41	-4.81	717.58	9.07	1715.51	-1.44	-0.54	30.09
-2.71	1.58	-1.11	7.78	0.81	3.25	-2292.10	-1151.06	23.08	2.63	0.24	-1.95
-0.63	-1.53	-3.1	-1.46	-1.82	-5.04	-597.84	594.06	-706.84	-2.67	-0.73	1.24
-2.24	-0.95	-1.55	7.43	2.22	6.30	1152.39	1311.719	1743.21	2.40	-0.15	-1.67
2.81	-0.42	-2.98	-2.90	2.01	6.95	1304.39	-420.67	912.84	2.02	0.54	2.64
3.12	0.79	-0.24	-7.64	-2.34	-7.61	-2057.54	-44.10	822.60	-2.44	-0.52	-0.97

4. System Identification of a Nonlinear Dynamical System Using VQTAM

The inverse kinematics of the robotic manipulator aims at solving the corresponding joint rotation once given the pose and initial pose of the manipulator. The joint rotation angle is regarded as output Q and the pose as input p . The inverse kinematics problem of the robot can be transformed into the identification problem of a nonlinear dynamic system. The time-discrete difference equation is established as shown in Equation (11) [27,29]. $Q(t) = \theta(t)$ is the joint rotation angle variable of the manipulator. $p(t) = [x(t), y(t), z(t); u(t), v(t), w(t)]$ is the pose of the manipulator at t time. It is expressed by the position coordinates combined with the Euler angle. $f(\cdot)$ represents a nonlinear function reflecting the characteristics of the system. The output Q at $t + 1$ is determined by the previous n_q outputs and n_p inputs.

The purpose of inverse kinematics is to find the mapping function $f(\cdot)$ between the input and output. As a complex nonlinear problem, on the premise of ensuring the accuracy, simplifying the model can improve efficiency and ensure real-time performance in engineering applications.

$$Q(t+1) = f[Q(t), \dots, Q(t-n_q+1); p(t), \dots, p(t-n_u+1)] \quad (11)$$

$$X^{in} = [Q(t), \dots, Q(t-n_q+1); p(t), \dots, p(t-n_u+1)] \quad (12)$$

$$X^{out} = [Q(t+1)] \quad (13)$$

VQTAM is a variant of SOM, which can be used to realize nonlinear mapping. The concept of nonlinear manifold embedding can be used to establish the nonlinear mapping. SOM is realized through competition and cooperation among feature neurons. In the mapping process, different neurons are activated for different inputs, and the activated neurons affect the output together with the neurons in their neighborhood. Therefore, time series data can be constructed as training samples of VQTAM.

VQTAM adopts training samples to build a topological structure embedded in multidimensional data space, which can construct the mapping relationship between multidimensional input vectors and multidimensional output vectors. VQTAM consists of three layers: input space ω^{in} , output space ω^{out} , and lattice space. The establishment of the mapping process is shown in Figure 4. The definitions of X^{in} and X^{out} are shown in Equations (12) and (13), respectively.

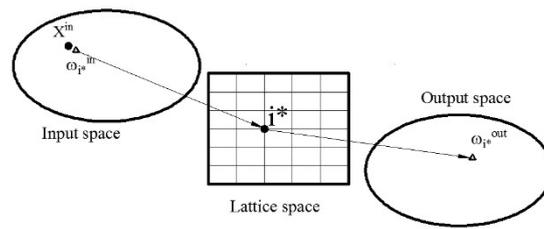


Figure 4. Vector-Quantized Temporal Associative Memory (VQTAM) spatial mapping.

The input space and output space are composed of the weight vectors ω_i^{in} and ω_i^{out} , respectively, where i is the index of the neuron in lattice space that reflects the location of the neuron in the lattice topology structure. Weight vectors ω_i^{in} and ω_i^{out} have mapping relationships with neurons in lattice space. ω_i^{in} and ω_i^{out} have the same dimensions as X^{in} and X^{out} , respectively. In the mapping process of VQTAM, the nearest weight vector $\omega_{i^*}^{in}$ to X^{in} is found in the input space, which corresponds to the activated neuron in the lattice space. In this process, the Euclidean distance is used to measure the distance between two vectors. The definition of i^* is shown in Equation (14), where A is the collection of all neuron indexes in lattice space [27].

$$i^* = \operatorname{argmin}_{i \in A} \{\|X^{in} - \omega_i^{in}\|\} \quad (14)$$

$\omega_{i^*}^{out}$ is obtained according to the neuron index i^* , and the output X^{out} is estimated, as shown in Equation (15).

$$\hat{X}^{out} = \omega_{i^*}^{out} \quad (15)$$

4.1. Learning Strategy of VQTAM

The main parameters of VQTAM are neuron weight vectors ω_i^{in} and ω_i^{out} , and lattice topology. The initial lattice topology structure can be determined as a hyperparameter before training, so the learning strategy mainly aims at updating the weight vectors ω_i^{in} and ω_i^{out} in the training process.

There is competition and cooperation among neurons in SOM. Different neurons and their neighborhoods are activated by inputting to participate in the learning process, and the weight vectors ω_i^{in} and ω_i^{out} are updated. The index search of the active neurons in the lattice space is consistent with Equation (14).

To improve the convergence rate in the learning process, the influence range parameters $\sigma(t)$ of the activation neuron and the learning speed $\alpha(t)$ decrease exponentially with the learning epoch, as shown in Equations (16) and (17).

$$\alpha(t) = \alpha_0 (\alpha_T / \alpha_0)^{t/T} \quad (16)$$

$$\sigma(t) = \sigma_0 (\sigma_T / \sigma_0)^{t/T} \quad (17)$$

where t is the current learning epoch, T is the total learning epoch. α_0, σ_0 are the initial values, and α_T, σ_T are the parameter values of the T training epoch.

The Gauss neighborhood function is used to determine the effect of input on the neighbors of the activated neuron, as shown in Equation (18) [27–29].

$$h(i^*, i; t) = \exp\left(\frac{-\|i(t) - i^*(t)\|^2}{\sigma^2(t)}\right) \quad (18)$$

Complete the updates of ω_i^{in} and ω_i^{out} , respectively, as shown in Equations (19) and (20).

$$\omega_i^{in} \leftarrow \alpha(t) h(i^*, i; t) [X^{in} - \omega_i^{in}] \quad (19)$$

$$\omega_i^{out} \leftarrow \alpha(t) h(i^*, i; t) [X^{out} - \omega_i^{out}] \quad (20)$$

To sum up, the VQTAM algorithm flow is as follows.

Algorithm 1: VQTAM Algorithm:

Begin
 (training part)
 1 Input: X^{in} and X^{out} in training set
 2 Search activating neuron according to X^{in}
 $i^* = \operatorname{argmin}_{i \in A} \{\|X^{in} - \omega_i^{in}\|\}$
 3 Update the weight vector of the neuron
 $\omega_i^{in} \leftarrow \alpha(t)h(i^*, i; t)[X^{in} - \omega_i^{in}]$
 $\omega_i^{out} \leftarrow \alpha(t)h(i^*, i; t)[X^{out} - \omega_i^{out}]$
 4 Continue execution until termination conditions are met
 (testing part)
 5 Input: X_{test}^{in} in testing set
 6 Search activating neuron according to X_{test}^{in}
 $i^* = \operatorname{argmin}_{i \in A} \{\|X_{test}^{in} - \omega_i^{in}\|\}$
 7 Output:
 $\hat{X}_{test}^{out} = \omega_{i^*}^{out}$

4.2. Searching the Activated Neuron by the Priority Search K-Means Tree Algorithm

During the training, testing, and estimation of VQTAM, the weight vector $\omega_{i^*}^{in}$ to which X^{in} is nearest in the input space is searched, as shown in Equation (14). The search algorithm applied has a great impact on the efficiency of VQTAM. The global traversal method needs to find all the Euclidean distance from the weight vector ω_i^{in} to the input X^{in} , and search the minimum value through traversal, which is inefficient. The K-Dtree algorithm can reduce the time complexity of searching, but for d-dimensional data, the time complexity of the K-Dtree data structure search is $O(n^{d/(d-1)})$. For a high-dimensional data search, especially when the data dimension $d > 20$, the K-Dtree data structure is inefficient. ω_i^{in} is 30-dimensional. The searching efficiency is severely restricted. The search time complexity of the priority search k-means tree algorithm (PSKMT) is $O(Ld \log n / \log K)$, where n is the amount of data in the data set, K is the number of nearest neighbors to be searched, and L is the maximum number of data retrieved in the search process. The priority search k-means tree algorithm is suitable for searching high-dimensional data.

The PSKMT algorithm divides the space into B different regions using the k-means clustering algorithm. Then, the points in each region are partitioned by the same operation until the number of data points in the region is no greater than K . In the process of searching, the idea of “divide and conquer” is used to find the nearest point in a smaller area and reduce the amount of data to be searched, so as to improve the efficiency.

The algorithm used for establishing a priority search K-means tree data structure is as follows.

Algorithm 2: K-means tree data structure building Algorithm [32]:

Begin
 1 Input: weight vector ω_i^{in} as search data set D , branch parameter B , maximum iteration number I_{max} , center selection algorithm using C_{alg}
 2 Compare size $|D|$ of data set D with branch parameter B
 3 If $|D| < B$: Create leaf nodes from data sets
 4 else, $P \leftarrow$ uses C_{alg} algorithm to select B points from data set D
 5 start loop:
 6 $C \leftarrow$ Clustering Data in D Centered on P
 7 $P_{new} \leftarrow$ Finding the Mean Value of Group C Data after Clustering
 8 if $P = P_{new}$, P_{new} is the non-leaf node, and terminate the loop
 9 These processes are executed on the sub-regions C until all leaf nodes are created
 10 Output: the entire K-means tree data structure

In the establishment of the K-means tree, the algorithm C_{alg} is used to select B points from data set D that can be chosen from the random selection, Gonzalez, and K-Means++ algorithms. The specific algorithm has little effect on data structure establishment. Generally, the random algorithm is recommended [16].

In the built K-means tree, leaf nodes are the data points in the original data set, and non-leaf nodes are the central points of the regions after segmentation. The search process can only be executed in a few areas, thus avoiding the global search of data sets. The K-means tree data structure is searched from the root node. Sub-nodes are sorted according to the distance between the clustering center and the query data points. The nearest sub-nodes are searched in advance. The priority K-means tree search algorithm is as follows.

Algorithm 3: PSKMT Algorithm [32]:

1. Input: K-means tree, query data $Q (X^{in})$, the nearest neighbor number K , the maximum number of search data L
 2. Initialize a stack and place the root node in the stack
 3. Starts loop. The condition for termination is that the stack is not empty and the amount of retrieved data $|P|$ is less than L :
 4. If the top node of the stack is a leaf node, add it to the retrieved data array P
 5. Otherwise, the top node goes out of the stack, reads the sub-nodes, sorts them according to the distance between the clustering center and the query data points, and pushes them into the stack.
 6. Loop terminated
 7. Output: K data nearest to query data Q in retrieval data array P .
-

5. VQTAM Local Linear Improvement Algorithms

The VQTAM algorithm is used to quantify the input space ω^{in} and output space ω^{out} . The neurons in ω^{in} and ω^{out} correspond to each other through mapping relations. The input X^{in} is approximated to the nearest neuron $\omega_{i^*}^{in}$ in estimation. The accuracy of the estimation results can be guaranteed when the number of neurons is large enough. However, the increase in number is followed by an increase in network size and a decrease in computing efficiency. Thus, local linearization of the activated node is used, which can balance the number of neurons and prediction accuracy. Based on local linearization, three improved algorithms for VQTAM are proposed: Local Linear Regression (LLR), Local Weighted Linear Regression (LWR), and LLE. The inverse kinematics can be further optimized, ensuring the computational efficiency.

Local linearization is performed in VQTAM networks. The K-means tree algorithm is used to search the nearest n data points $\omega_{i^*n}^{in}$ and the corresponding output data $\omega_{i^*n}^{out}$ are mapped, as shown in Figure 5.

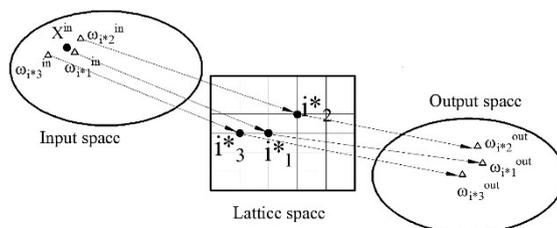


Figure 5. Local linearization of the VQTAM network.

5.1. Improvement Algorithm of VQTAM with LLR

It is assumed that the relationship between the local input and output can be expressed linearly, so LLR can be performed as shown in Equation (21).

$$\mathbf{Y}\mathbf{B} = \mathbf{Z} \quad (21)$$

\mathbf{Y} is the nearest neighborhood $\omega_{i^*n}^{in}$ of X^{in} , and \mathbf{Z} is the corresponding data point $\omega_{i^*n}^{out}$, as shown in Equations (22) and (23).

$$\mathbf{Y} = \begin{bmatrix} \omega_{i^*1}^{in} \\ \vdots \\ \omega_{i^*n}^{in} \end{bmatrix} \in \mathbb{R}^{n \times m} \quad (22)$$

$$\mathbf{Z} = \begin{bmatrix} \omega_{i^*1}^{out} \\ \vdots \\ \omega_{i^*n}^{out} \end{bmatrix} \in \mathbb{R}^{n \times l} \quad (23)$$

The linear relationship between the local input and output can be obtained by solving \mathbf{B} . Equation (21) is an overdetermined equation, which is generally treated as a least squares problem and solved by a generalized inverse matrix, as shown in Equation (24).

$$\mathbf{B} = (\mathbf{Y}^T \mathbf{Y})^{-1} \mathbf{Y}^T \mathbf{Z} \quad (24)$$

$\mathbf{Y}^T \mathbf{Y}$ is an n -dimensional square matrix, and the calculation of the inverse operation is huge when the dimension increases. Meanwhile, $\mathbf{Y}^T \mathbf{Y}$ may be a singular matrix or ill-conditioned matrix, so finding the inverse matrix directly is difficult. Singular Value Decomposition (SVD) can be used to solve the least squares problem. \mathbf{Y} can be decomposed as shown in Equation (25).

$$\text{SVD}(\mathbf{Y}) = [\mathbf{U}][\mathbf{S}][\mathbf{V}^T] \quad (25)$$

where $\mathbf{Y} \in \mathbb{R}^{m \times m}$, $\mathbf{U} \in \mathbb{R}^{m \times l}$, $\mathbf{V} \in \mathbb{R}^{l \times l}$

The column vectors of \mathbf{U} are the eigenvectors of $\mathbf{Y}\mathbf{Y}^T$; the column vectors of \mathbf{V}^T are the eigenvectors of $\mathbf{Y}^T \mathbf{Y}$.

$$\mathbf{S} = \begin{bmatrix} \Sigma \\ \mathbf{0} \end{bmatrix} \quad (26)$$

Σ is a diagonal matrix whose value is the singular value v of matrix \mathbf{Y} , i.e., the eigenvalue of $\mathbf{Y}\mathbf{Y}^T$ is $\lambda = v^2$, and \mathbf{U} and \mathbf{V}^T are the orthogonal matrix.

\mathbf{U} can be disassembled as $[\mathbf{U}_n, \mathbf{U}_{m-n}]$. For the least squares problem as shown in Equation (21), the solution obtained by SVD is shown in Equation (27).

$$\mathbf{B} = \mathbf{V}\Sigma^{-1}\mathbf{U}_n^T \mathbf{Z} \quad (27)$$

The input X^{in} is used to estimate the output, as shown in Equation (28).

$$\hat{X}^{out} = X^{in} \mathbf{B} \quad (28)$$

5.2. Improvement Algorithm of VQTAM with LWR

Using the LLR algorithm, the impact of all neighbor points on the prediction results is consistent. Considering the different influences of neighbor points, a distance weight is added in the regression process. A local weight linear regression (LWR) is adopted. The cost function is defined as shown in Equation (29).

The weight w_k in Equation (29) is determined by the Gauss neighborhood function, as shown in Equation (30), where τ is the bandwidth parameter.

$$J(\boldsymbol{\varphi}) = \sum_{k=1}^n w_k \|\boldsymbol{\omega}_{i^*k}^{out} - \boldsymbol{\varphi}_k \boldsymbol{\omega}_{i^*k}^{in}\|^2 \quad (29)$$

$$w_k = \exp\left(-\frac{\|\mathbf{X}^{in} - \boldsymbol{\omega}_{i^*k}^{in}\|^2}{\tau^2}\right) \quad (30)$$

The analytical solution of the cost function minimization is equivalent to the solution of the overdetermined equation shown in Equation (31).

$$\mathbf{WY}\boldsymbol{\varphi} = \mathbf{WZ} \quad (31)$$

where \mathbf{W} is an n -dimensional diagonal matrix, and the diagonal element is w_k .

The SVD for matrix \mathbf{WY} is also used to solve the overdetermined equation.

$$SVD(\mathbf{WY}) = [\mathbf{UY}][\mathbf{S}][\mathbf{V}^T] \quad (32)$$

$$\boldsymbol{\varphi} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}_n^T\mathbf{WZ} \quad (33)$$

5.3. Improvement Algorithm of VQTAM with LLE

It is assumed that an input datum can be represented by a linear combination of several samples in its neighborhood; that is, the input X^{in} to be predicted is represented by a linear combination of n data points in its neighborhood in the input space $\boldsymbol{\omega}_{i^*n}^{in}$.

$$\mathbf{X}^{in} = \sum_{k=1}^n c_k \boldsymbol{\omega}_{i^*k}^{in} \quad (34)$$

The output has the same linear combination.

$$\hat{\mathbf{X}}^{out} = \sum_{k=1}^n c_k \boldsymbol{\omega}_{i^*k}^{out} \quad (35)$$

c_k is the coefficient of the linear combination, and the output can be estimated by Equation (35) after solving c_k . Define the cost function, as shown in Equation (36), and rewrite it as a matrix:

$$\begin{aligned} J(c) &= \|\mathbf{X}^{in} - \sum_{k=1}^n c_k \boldsymbol{\omega}_{i^*k}^{in}\|^2 \\ &= \|\mathbf{X}^{in} - \mathbf{Y}\mathbf{c}\|^2 \end{aligned} \quad (36)$$

The least squares problem, as shown in Equation (35), can also be solved by SVD.

$$\mathbf{c} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}_n^T\mathbf{X}^{in} \quad (37)$$

6. Simulation Results and Discussion

6.1. Standard VQTAM Network Test Results

According to the parameters of the robot shown in Table 1, the joint rotation angles for the sample set are generated as shown in Equations (38)–(43) [26,28]. The generated data can effectively cover the whole workspace of the manipulator.

$$\theta_1(t) = \pi(1 - e^{-\pi t}) \cos 1.88\pi t \quad (38)$$

$$\theta_2(t) = \frac{3}{2}\pi(1 - e^{-\pi t}) \sin 1.88\pi t + \pi/6 \quad (39)$$

$$\theta_3(t) = \frac{3}{4}\pi \cos t - \pi/4 \quad (40)$$

$$\theta_4(t) = \frac{5}{2}\pi \sin t \quad (41)$$

$$\theta_5(t) = \frac{4}{5}\pi(1 - e^{-\pi t}) \sin 0.86\pi t \quad (42)$$

$$\theta_6(t) = \frac{5}{2}\pi(1 - e^{-\pi t}) \sin 0.74\pi t \quad (43)$$

Taking time $t \in [0,1000]$ and step $t = 0.1$, 10,001 joint angles $\mathbf{Q}(t) = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]$ are calculated as the output of the sample set. The poses $\mathbf{p}(t) = [x, y, z; u, v, w]$ of the manipulator are calculated using the exponential coordinates of the motion screw, which are used as the input of the sample set. The VQTAM network is trained with the sample set X^{in} and X^{out} . In each epoch, 1000 pieces of data are selected randomly from the sample set, 80% of which are used as the training set and 20% as the test set. Meanwhile, to test the robustness of the algorithm, random errors are added to the test set data according to $N(0,1)$ distribution.

The hyperparameter setting of the VQTAM network training is shown in Table 3. A VQTAM network with dimensions 60×60 ($M_x \times M_y$) is trained based on the setting of hyperparameters, and the training effect is tested by the test set.

Table 3. Hyperparameter setting of VQTAM.

n_q	n_p	<i>epoch</i>	α_0	α_M	σ_0	σ_M
3	3	5000	0.8	0.001	15	0.001

Root Mean Squared Error (RMSE), R-squared (R^2), and Relative Maximum Absolute Error (RMAE) are used to evaluate the prediction accuracy of VQTAM for inverse kinematics, as shown in Table 4.

Table 4. Root Mean Squared Error (RMSE), R-squared (R^2), and Relative Maximum Absolute Error (RMAE) of the standard VQTAM network.

	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
RMSE	0.1871	0.1663	0.0895	0.2189	0.2459	0.5708
R^2	0.9931	0.9808	0.9972	0.9984	0.9808	0.9894
RMAE	0.3914	0.6660	0.4840	0.1734	0.9828	0.5477

The value of R^2 ranges from 0 to 1. The closer it is to 1, the better the effect of regression fitting is, which means that the learning VQTAM network has higher accuracy as an approximate model. Generally, $R^2 > 0.95$ can be applied in engineering. For VQTAM networks with dimensions 60×60 , the R^2 values after learning are all above 0.98. This shows that this network has value for engineering applications. RMSE and RMAE reflect the prediction error of the VQTAM network, and their small

values also indicate the prediction accuracy of the network. The convergence result in the neural network training process is shown in Figure 6.

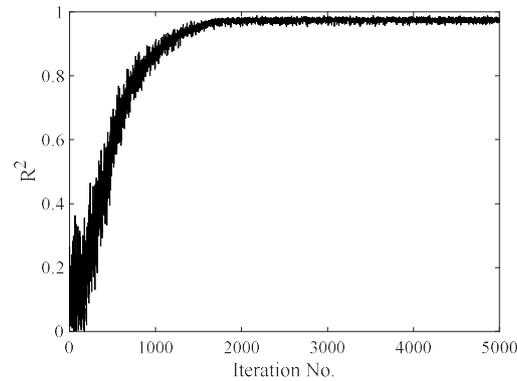


Figure 6. Convergence result.

It can be seen from Figure 6 that the mean square R^2 of the VQTAM neural network tested by the test set in the training process gradually increases and converges near 1. This proves the effectiveness of the VQTAM neural network training algorithm.

The consistency between the actual data and the estimated data can be displayed more intuitively through box diagrams, as shown in Figure 7. It can be seen that the box diagrams of actual values and predicted values are very similar, which verifies the excellent prediction effect of the VQTAM network with dimensions 60×60 .

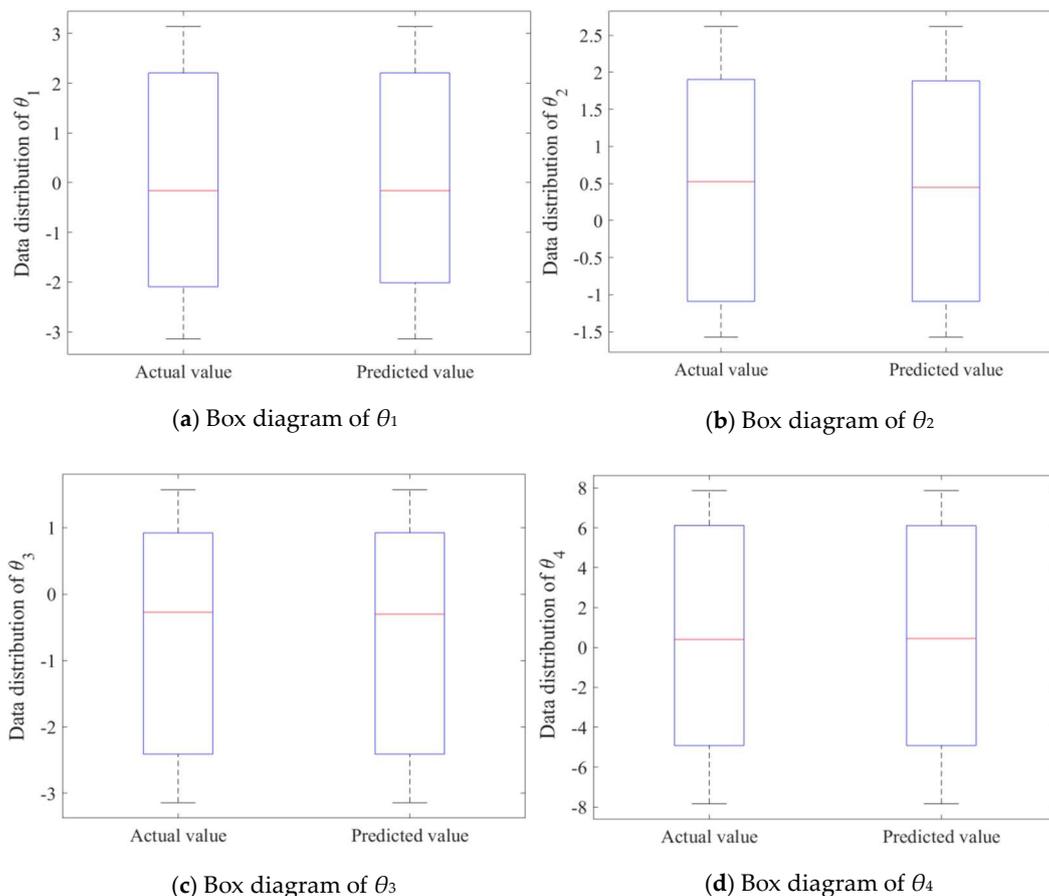


Figure 7. Cont.

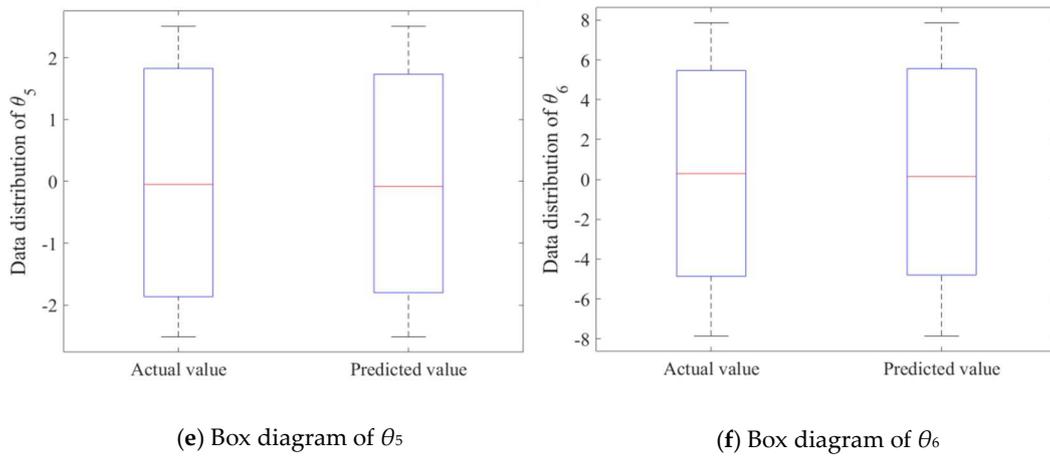


Figure 7. Box diagrams of the VQTAM network prediction effect.

6.2. VQTAM Local Linear Improvement Algorithms

The output of the inverse kinematics problem is estimated using the VQTAM local linear improvement algorithms. The priority K-means tree search algorithm is used to search the neighbor data of the input. The influence of parameter k on the prediction accuracy is analyzed, taking a VQTAM network with dimensions 35×35 as an example, as shown in Figure 8.

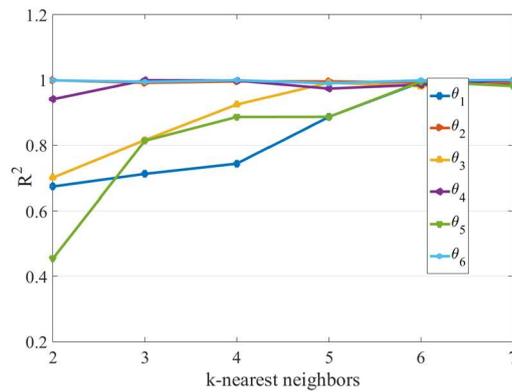


Figure 8. The influence of k-nearest neighbor number in the improvement algorithm of VQTAM with Local Linear Regression (LLR).

As can be seen from Figure 8, with the increase in k, R^2 increases continuously and approaches 1, which shows an increase in prediction accuracy. When $k = 6$, the R^2 for all variables is close to 1.

As can be seen from Figure 9, when $k = 5$, the R^2 for all variables is more than 0.95. When $k = 6$, R^2 is close to 1. Comparing Figures 8 and 9, the overall prediction accuracy of the LWR algorithm is higher than that of the LLR algorithm.

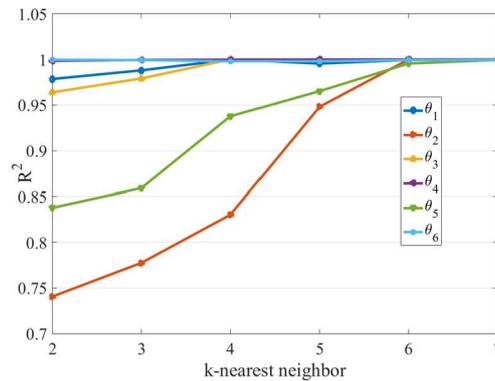


Figure 9. The influence of k-nearest neighbor number in the improvement algorithm of VQTAM with Local Weighted Linear Regression (LWR).

As can be seen from Figure 8, when $k = 5$, the R^2 for all variables is greater than 0.99. Comparing Figures 8–10, the overall prediction accuracy of the improvement algorithm of VQTAM with LLE is the highest.

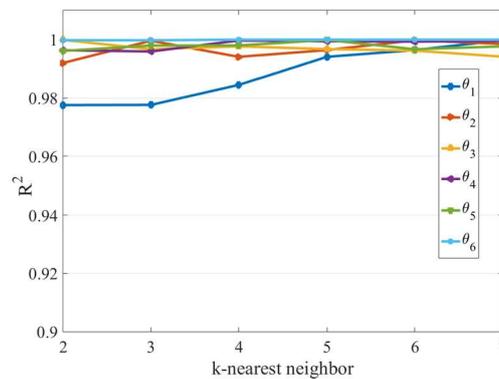


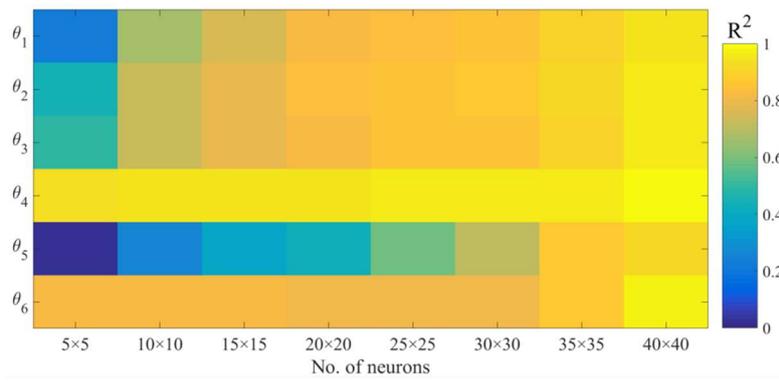
Figure 10. The influence of k-nearest neighbor number in the improvement algorithm of VQTAM with Local Linear Embedding (LLE).

In conclusion, when the number of neurons is small, the prediction accuracy of the network can be improved by combining local linear algorithms.

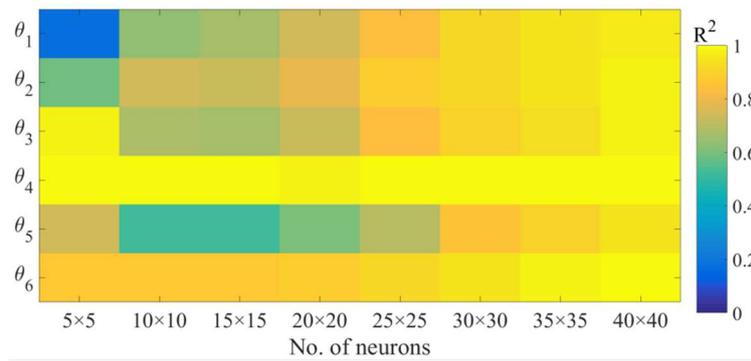
6.3. Overall Examples and Test Results

According to the VQTAM network hyperparameters shown in Table 3, the prediction accuracy of the four algorithms under different network dimensions (5×5 , 10×10 , 15×15 , 20×20 , 25×25 , 30×30 , 35×35 , 40×40) is investigated, as shown in Figure 9. The number of the nearest nodes of the local linear algorithm is set to $k = 6$.

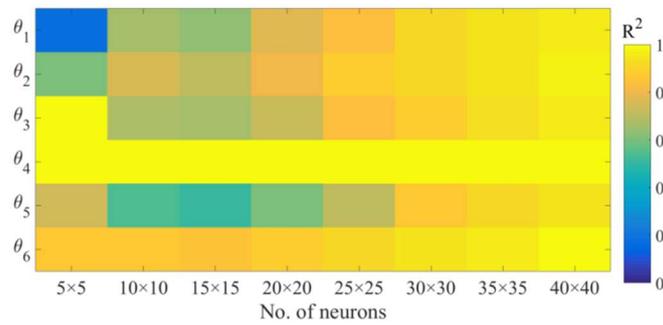
As can be seen from Figure 11, with the increase in the number of neurons, the prediction accuracy of the four algorithms significantly increases. The overall prediction accuracy of the improvement algorithm of VQTAM with LLE is the highest. The local linearization of VQTAM achieved remarkable results. Using the improved VQTAM algorithm, the estimation accuracy level of a network with dimensions of 35×35 is close to that of the standard VQTAM with dimensions of 60×60 . Taking the improvement algorithm of VQTAM with LLE as an example, the output of the inverse kinematics, i.e., the rotation angles of six joints, is estimated, as shown in Figure 12. The estimated rotation angles of joints are compared with actual angles. It is shown that the VQTAM neural network can effectively estimate the output.



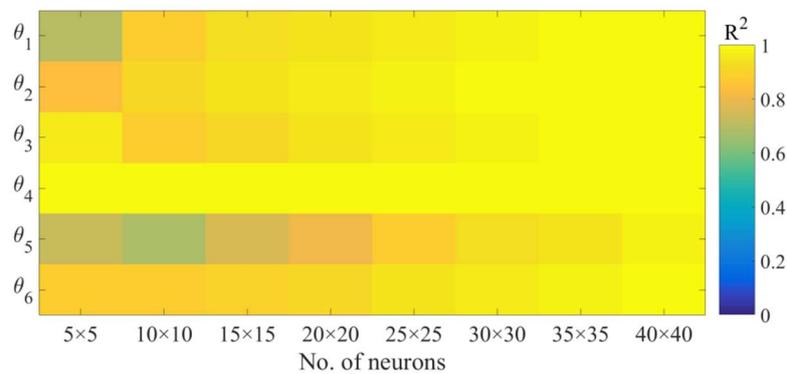
(a) Prediction accuracy of the standard VQTAM network.



(b) Prediction accuracy of the improvement algorithm of VQTAM with LLR.



(c) Prediction accuracy of the improvement algorithm of VQTAM with LWR.



(d) Prediction accuracy of the improvement algorithm of VQTAM with LLE.

Figure 11. Prediction accuracy of 4 algorithms.

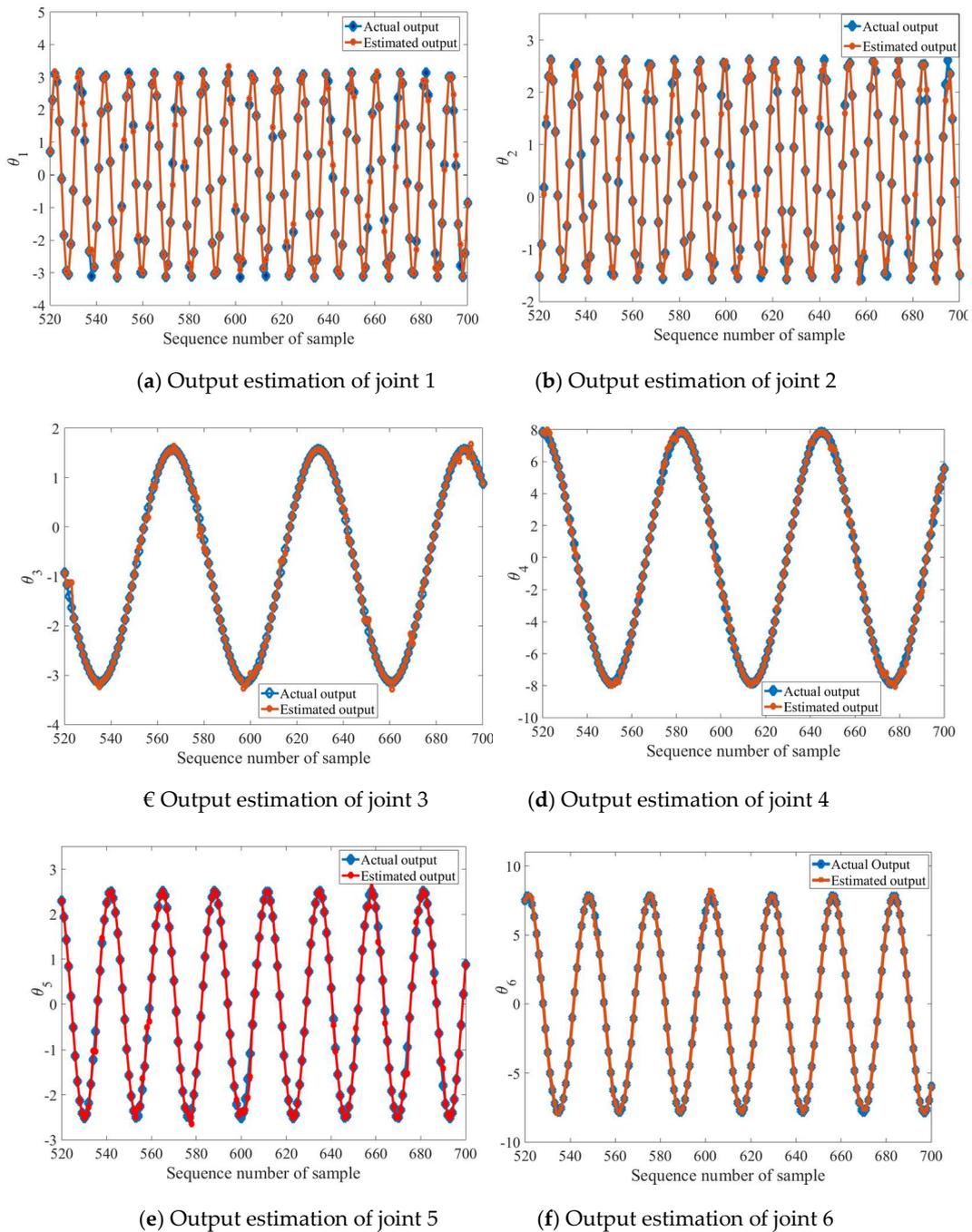


Figure 12. Inverse kinematics output estimation.

Substituting the joint rotation angle into the forward kinematics equation for calculation, the error of VQTAM can be obtained by comparing the result with the actual pose of the robot. The maximum error in the tool space is 3.9686×10^{-4} mm, and the average error is 4.6857×10^{-5} mm. This precision has met the control requirements of general robots in industrial applications.

The multilayer perceptron (MLP) can also provide the estimated results for the inverse kinematics. Thus, an MLP NARX network is trained for the comparison of prediction accuracy with VQTAM and its improvement algorithms. The MLP network has a hidden layer, and the number of hidden neurons is 3600. The sample set $p(t)$, $Q(t)$ for training MLP is the same as that for VQTAM. The Levenberg–Marquardt algorithm is chosen as the training algorithm. The comparison of prediction accuracy is shown in Table 5.

Table 5. Comparison of prediction accuracy.

	MLP	VQTAM (60 × 60)	VQTAM (50 × 50)
RMSE	1.5935	0.2901	0.7823
	VQTAM with LLR (50 × 50)	VQTAM with LWR (50 × 50)	VQTAM with LLE (50 × 50)
RMSE	0.5163	0.6792	0.3018

The prediction accuracy of the standard VQTAM network is significantly higher than that of the MLP neuron network. Meanwhile, the prediction accuracy of the improvement algorithm of VQTAM also has better performance. The VQTAM is less sensitive to weight initialization than the MLP. Training multiple times will generate different results due to different initial conditions. The variation in the range of results of VQTAM is smaller. Compared with the results of a one-dimensional VQTAM in a previous study [27], the increase in dimensions improves the performance of the neural network.

7. Conclusions

The kinematics of the robotic manipulator is studied, and the forward kinematics solution method in the form of the exponential product of the screw is derived. Given the input angle of each joint of the robot, this method can be used to calculate the pose of the manipulator conveniently and quickly. It can also be used as the computational basis of a VQTAM network to establish sample sets.

A fast inverse kinematics mapping method for a robotic manipulator is proposed by using VQTAM for identification of nonlinear dynamic systems. The VQTAM algorithm is constructed to train and test the network. Based on the priority K-means tree search algorithm, the training, testing, and estimating processes of the VQTAM network are improved to enhance the search efficiency of nearest neighbors.

According to local neuron activation, the VQTAM network is improved based on local linearization. To further optimize the prediction accuracy of the network and reduce the dimensions of the network in application, VQTAM local linear improvement algorithms are proposed.

In this study, a sample data set for VQTAM training is constructed and the VQTAM algorithm and its improved algorithms are tested. The test results show that the increase in network dimensions can improve the prediction accuracy of VQTAM, but the computational efficiency is affected. By using VQTAM local linearization improvement algorithms, the estimation accuracy can be optimized for a low-dimensional network. When the search range is $k = 6$, the above algorithm can show good prediction accuracy, and the prediction effect of the LLE algorithm is the best among the three algorithms.

Author Contributions: Data curation, L.L.; Formal analysis, L.L.; Investigation, Z.Q.; Methodology, L.L.; Project administration, H.L.; Supervision, H.L.; Validation, L.L., W.Y., and Z.Q.; Visualization, W.Y.; Writing—original draft, L.L.; Writing—review and editing, L.L. and W.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This project is supported by the National Natural Science Foundation of China (Grant No. 5187052280) and by the Economic and Information Commission of Sichuan Province (Development of Normal-position Intelligent Manufacturing Technology and Device for Major Equipment).

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Mustafa, A.; Tyagi, C.; Verma, N.K. Inverse Kinematics Evaluation for Robotic Manipulator using Support Vector Regression and Kohonen Self Organizing Map. In Proceedings of the International Conference on Industrial and Information Systems, Roorkee, India, 3–4 December 2016; pp. 375–380.
2. Amouri, A.; Mahfoudi, C.; Zaatri, A.; Lakhal, O.; Merzouki, R. A Meta-heuristic Approach to Solve Inverse Kinematics of Continuum Manipulators. Available online: <https://journals.sagepub.com/doi/abs/10.1177/0959651817700779> (accessed on 24 February 2020).

3. Jiang, L.; Huo, X.; Liu, Y.; Liu, H. An Analytical Inverse Kinematic Solution with the Reverse Coordinates for 6-DOF Manipulators. In Proceedings of the 2013 IEEE International Conference on Mechatronics and Automation, Takamatsu, Japan, 4–7 August 2013; pp. 1552–1558.
4. Sariyildiz, E.; Temeltas, H. Solution of Inverse Kinematic Problem for Serial Robot Using Quaternions. In Proceedings of the 2009 IEEE International Conference on Mechatronics and Automation, Singapore, 14–17 July 2009; pp. 26–31.
5. Sariyildiz, E.; Temeltas, H. A Comparison Study of Three Screw Theory Based Kinematic Solution Methods for the Industrial Robot Manipulators. In Proceedings of the 2011 IEEE International Conference on Mechatronics and Automation, Beijing, China, 7–10 August 2011; pp. 52–57.
6. Zhang, L.; Zuo, J.; Zhang, X.; Yao, X. A New Approach to Inverse Kinematic Solution for a Partially Decoupled Robot. In Proceedings of the 2015 International Conference on Control, Automation and Robotics, Singapore, 20–22 May 2015; pp. 55–59.
7. Sun, J.-D.; Cao, G.-Z.; Li, W.-B.; Liang, Y.-X.; Huang, S.-D. Analytical Inverse Kinematic Solution Using the D-H Method for a 6-DOF Robot. In Proceedings of the 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence, Jeju, South Korea, 28 June–1 July 2017; pp. 714–716.
8. Xin, S.Z.; Feng, L.Y.; Bing, H.L.; Li, Y.T. A Simple Method for Inverse Kinematic Analysis of the General 6R Serial Robot. In Proceedings of the ASME 2006 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Philadelphia, PA, USA, 10–13 September 2006; pp. 1–8.
9. Joseph, C.; Musto, A. Two-Level Strategy for Optimizing the Reliability of Redundant Inverse Kinematic Solutions. *J. Intell. Robot. Syst.* **2002**, *33*, 73–84.
10. Wang, Y.; Sun, L.; Yan, W.; Liu, J. An Analytic and Optimal Inverse Kinematic Solution for a 7-DOF Space Manipulator. *Robot* **2014**, *36*, 592–599.
11. Chiaverini, S.; Ecgeland, O. An efficient pseudo-inverse solution to the inverse kinematic problem for six-joint manipulators. *Model. Identif. Control* **1990**, *11*, 201–222. [[CrossRef](#)]
12. Wang, W.; Suga, Y. Hiroyasu Iwata and Shigeki Sugano, Solve Inverse Kinematics Through A New Quadratic Minimization Technique. In Proceedings of the 2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Kachsiung, Taiwan, 11–14 July 2012; pp. 313–360.
13. Hargis, B.E.; Demirjian, W.A.; Powelson, M.W.; Canfield, S.L. Investigation of Neural-Network-Based Inverse Kinematics for A 6-DOF Serial Manipulator with Non-Spherical Wrist. In Proceedings of the ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Quebec City, QC, Canada, 26–29 August 2018; pp. 1–14.
14. Lou, Y.F.; Brunn, P. A hybrid artificial neural network inverse kinematic solution for accurate robot path control. *Proc. Inst. Mech. Eng. Part I J. Syst. Control Eng.* **1999**, *213*, 23–32. [[CrossRef](#)]
15. Mahajan, A.; Singh, H.P.; Sukavanam, N. An unsupervised learning based neural network approach for a robotic manipulator. *Int. J. Inf. Technol.* **2017**, *9*, 1–6. [[CrossRef](#)]
16. Kenwright, B. Neural Network in Combination with a Differential Evolutionary Training Algorithm for Addressing Ambiguous Articulated Inverse Kinematic Problems. In Proceedings of the 11th ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia, Tokyo, JAPAN, 4–7 December 2018.
17. Kinoshita, K. Estimation of Inverse Model by PSO and Simultaneous Perturbation Method. In Proceedings of the 2015 Seventh International Conference of Soft Computing and Pattern Recognition, Fukuoka, Japan, 13–15 November 2015; pp. 48–53.
18. Rui, T.; Zhu, J.-W.; Zhou, Y.; Ma, G.-Y.; Yao, T. Inverse Kinematics Analysis of Multi-DOFs Serial Manipulators Based on PSO. *J. Syst. Simul.* **2009**, *21*, 2930–2932.
19. Shihabudheen, K.V.; Pillai, G.N. Evolutionary Fuzzy Extreme Learning Machine for Inverse Kinematic Modeling of Robotic Arms. In Proceedings of the 2015 39th National Systems Conference, Noida, India, 14–16 December 2015.
20. Oyama, E.; Maeda, T.; Gan, J.Q.; Rosales, E.M.; MacDorman, K.F.; Tachi, S.; Agah, A. Inverse kinematics learning for robotic arms with fewer degrees of freedom by modular neural network systems. In Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, AB, Canada, 2–6 August 2005; pp. 1791–1798.

21. Wu, X.; Xie, Z. Forward kinematics analysis of a novel 3-DOF parallel manipulator. *Sci. Iran.* **2019**, *26*, 346–357. [[CrossRef](#)]
22. Pham, D.T.; Castellani, M.; Fahmy, A.A. Learning the inverse kinematics of a robot manipulator using the Bees algorithm. In Proceedings of the 2008 6th IEEE International Conference on Industrial Informatics, Daejeon, South Korea, 13–16 July 2008; pp. 493–498.
23. Giorelli, M.; Renda, F.; Calisti, M.; Arienti, A.; Ferri, G.; Laschi, C. Learning the inverse kinetics of an octopus-like manipulator in three-dimensional space. *Bioinspir. Biomim.* **2015**, *10*, 035006. [[CrossRef](#)] [[PubMed](#)]
24. Karkalos, N.E.; Markopoulos, A.P.; Dossis, M.F. Optimal Model Parameters of Inverse Kinematics Solution of a 3R Robotic Manipulator Using Ann Models. *Int. J. Manuf. Mater. Mech. Eng.* **2017**, *7*, 20–40. [[CrossRef](#)]
25. Rong, P.-X.; Yang, Y.-J.; Hu, L.-G.; Ma, G.-F. Inverse kinematic in SCARA manipulator based on RBF networks. *Electr. Mach.* **2007**, *11*, 303–305.
26. Kumar, P.R.; Bandyopadhyay, B. The Forward Kinematic Modeling of a Stewart Platform using NLARX Model with Wavelet Network. In Proceedings of the IEEE International Conference on Industrial Informatics, Bochum, Germany, 29–31 July 2013; pp. 343–348.
27. Barreto, G.D.; Araujo, A.F.R. Temporal associative memory and function approximation with the self-organizing map. In Proceedings of the Neural Networks For Signal Processing Xii, Martigny, Switzerland, 6 September 2002; pp. 109–118.
28. Limtrakul, S.; Arnonkijpanich, B. Supervised learning based on the self-organizing maps for forward kinematic modeling of Stewart platform. *Neural Comput. Appl.* **2019**, *31*, 619–635. [[CrossRef](#)]
29. Heskes, T. Self-organizing maps, vector quantization, and mixture modeling. *IEEE Trans. Neural Netw.* **2001**, *12*, 1299–1305. [[CrossRef](#)] [[PubMed](#)]
30. Zhu, S.; Chen, Q.; Wang, X.; Liu, S. Dynamic modeling using screw theory and nonlinear sliding mode control of series robot. *Int. J. Robot. Autom.* **2016**, *31*, 63–75.
31. Chen, Q.; Zhu, S.; Zhang, X. An Improved Inverse Kinematic Algorithm for 6 DOF Serial Robot Using Screw Theory. *Int. J. Adv. Robot. Syst.* **2015**, *12*, 140. [[CrossRef](#)]
32. Muja, M.; Lowe, D.G. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *36*, 2227–2240. [[CrossRef](#)] [[PubMed](#)]



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).