World's largest Science, Technology & Medicine Open Access book publisher

Chapter from the book
Downloaded from: http://www.intechopen.com/books/

Interested in publishing with InTechOpen?
Contact us at book.department@intechopen.com
Predicting the Motion of a Robot Manipulator with Unknown Trajectories Based on an Artificial Neural Network

Regular Paper

Sai Hong Tang¹, Chun Kit Ang¹*, Mohd Khairul Anuar Bin Mohd Ariffin¹ and Syamsiah Binti Mashohor²

¹ Department of Mechanical and Manufacturing Engineering, Faculty of Engineering, Universiti Putra Malaysia, Serdang, Selangor Darul Ehsan, Malaysia
² Department of Computer and Communications Systems Engineering, Faculty of Engineering, Universiti Putra Malaysia, Serdang, Selangor Darul Ehsan, Malaysia
* Corresponding author E-mail: ack_kit@hotmail.com

Received 11 Feb 2014; Accepted 17 Sep 2014

DOI: 10.5772/59278

© 2014 The Author(s). Licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract Mathematically, the motion of a robot manipulator can be computed through the integration of kinematics, dynamics, and trajectories calculations. However, the calculations are complex and only can be applied if the configuration of the robot and the characteristics of the joint trajectories are known. This paper introduces the use of artificial neural networks (ANN) to overcome these shortcomings by solving nonlinear functions and adapting the characteristics of unknown trajectories. A virtual six-degree-of-freedom (DOF) robot manipulator is exploited as an example to show the robustness of the developed ANN topology.

Keywords Artificial Neural Networks, Robotic Manipulator, Unknown Trajectory

1. Introduction

Robot manipulators have been widely used in recent decades, for example in heavy industries, on production sites or in the medical sector, due to their ability and efficiency in monotonous work. However, serious accidents have occurred due to lack of awareness about robot machinery or improper manipulation issues. The study of robot manipulator mechanisms and motion has therefore become indispensable.

Mathematically, the motion of a robot manipulator can be computed through the integration of kinematics (forward and inverse kinematics), dynamics and trajectory-planning calculations [1-22]. However, it is difficult to apply these mathematical solutions. Problems include failure to derive the kinematic equations and inability to find analytical solutions [23-29] or to solve the variety of complex mathematical equations. Furthermore, the mathematical method can only be applied if the characteristics of the trajectories are known [7,8,10,12,14,18,22]. It is a complex and tedious process to compute the motion of a robot manipulator by using the conventional approach.
Previous researchers have developed a lot of primitive trajectories, such as linear trajectory [11], linear segment parabolic blends (LSPB) [7,9,18], bang-bang trajectory [7,9,18,20], quintic polynomial trajectory [4,7,9,12-14,20], high-order polynomial trajectory [4,7,9,12-14,20,22], acceleration sine function profile trajectory [7,20], etc., which possess different characteristics. Linear trajectory is the most primitive trajectory, whereby the robot movement is represented by piece-wise linear functions [7,9,11]. LSPB has a trapezoidal velocity profile, and a constant velocity is required along a portion of the path. The velocity is initially ‘ramped up’ from zero to its desired value and then ‘ramped down’ when it approaches the goal position [7,9,18]. Bang-bang trajectory is similar to LSPB but produces the fastest trajectory through accelerating the velocity to maximum in a short time [7,9,18,20]. Cubic polynomial can be used if the position and velocity at initial and end point are known, but produces discontinuities in acceleration. Quintic polynomial is able to produce continuity in acceleration but all the constraints (position, velocity, acceleration) of a robot at initial and end point must be sought out [4,7,9,12-14,22]. High-order polynomial can be applied for robots which have to pass through a few via points. It requires extensive calculation [4,7,9,12-14,20,22].

There are also a lot of complex trajectory methods that have been developed to meet different manipulation requirements. For examples, Pan et al. proposed a sample-based planning method to produce smooth trajectory, which consisted of the fundamentals of cubic spline [4]. Macfarlance and Croft proposed a method to reduce the jerk for manipulators in real-time manipulation by integrating the techniques of quintic polynomial and sine wave acceleration function [7]. Hauser and Ng integrated the techniques of straight-line trajectory and parabolic trajectory to produce a fast-smoothing trajectory [11]. Rossi and Savino proposed the ‘envelop of tangent method’ for point-to-point trajectory [18]. Briot et al. proposed a method to reduce the shaking force for high-speed mobile manipulators by implementing the fundamentals of bang-bang and trapezoidal velocity profile trajectories [20], etc. These trajectories possess different characteristics and their trajectory equations are totally different.

In order to compute the motion of a robot, it is crucial to derive the trajectory equations. Sometimes the trajectory equations cannot be derived if the characteristics of a trajectory are unknown. Without the trajectory equations, one might not be able to compute the motion of a robot. The main motivation of this paper is to propose a method which is able to adapt the characteristics of different trajectories, even those trajectories with unknown characteristics. Artificial neural network (ANN) is strongly recommended for achieving this objective based on its ability in solving nonlinear functions and its cognitive learning ability [10,16,23,25-31,33-34,36]. Users are able to compute the position and motion of a robot without deriving any trajectory equations using the current proposed method.

There are various types of neural network, such as the Hopfield (recurrent) network [30,31,36], and Bidirectional Associative Memory (BAM) [37], unsupervised learning networks [38] and self-organizing neural networks [39-41]. Among these network models, back-propagation neural network (BPNN) is chosen as a fundamental topology for the current research because it is widely used for pattern recognition problems and is easier to implement with high accuracy [10,16,23,25-31,33,34,36,42]. Some researchers have applied ANN to solve trajectory issues such as recurrent neural network [30,31,34,36] and BPNN (single multilayer) [43-45]. However, there are certain limitations in such studies. Recurrent neural network allows a robot to follow a specified path (with known trajectory) after training [30,31,34,36] but does not allow prediction of the motion of a robot with unknown trajectory. BPNN possesses the ability to adapt and learn from experiences and analogy, but a single multilayer back-propagation neural network is unable to produce good results in predicting the motion of a robot, especially a robot with a high degree of freedom (≥ 6DOF) [43-45]. It is hard for a single multilayer BPNN to adapt multi-nonlinear functions. Therefore, the novelty of this paper is to introduce a neural network topology which consists of multi-BPNNs and can be used for predicting the motion of a high-DOF robot with unknown trajectory.

2. Preliminaries

2.1 Kinematics

Robot manipulators are governed by two basic kinematic theories, namely forward kinematics (FK) and inverse kinematics (IK). FK theory calculates the position and orientation of the hand or wrist (end effector) base on the robot’s configuration, and the robot’s configuration is assumed to be known in terms of all the joint variable angles and the length of links [10,16,17,21,23,34]. Conversely, given the desired position and the orientation of the end effector \( \mathbf{r}(t) \), IK theory enables the robot to calculate all the joint values \( \mathbf{\theta} \), to place itself at the desired location and orientation [10,16,17,21,23,26-27,29,35].

Unfortunately, it is usually impossible to find an analytic solution for IK due to the nonlinearity of the equations \( \mathbf{f}(\mathbf{\theta}(i)) \), especially in singularity condition [23-29,34-35] such as degeneracy, which causes the robot to lose a degree of freedom (DOF). IK problems are usually solved, therefore, by using a pseudo-inverse method with Jacobin matrix in robotic dynamic system to overcome this...
shortcoming. Jacobin is a function of $\theta$ values, defined as
$$J_e(\theta) = \frac{\partial}{\partial \theta} f_e(\theta(t))$$
with respect to time yields a linear relationship between the Cartesian velocity $\dot{r}_e$ and the joint velocity $\dot{\theta}$:
$$J_e(\theta) \dot{\theta} = \dot{r}_e$$
where $J_e(\theta) \in R^{m \times n}$ is the end-effector Jacobin matrix [23-36].

2.2 Dynamics

Other than kinematics, a person has to know the dynamics system of a robot manipulator for computing the robot motion. As mentioned above, inverse kinematics are usually solved at the velocity level. For an $n$-DOF manipulator, the joint position variable can be denoted by $q(t) = [q_1(t), q_2(t), \ldots, q_n(t)]^T$ where $t \in [0, T]$, $J_q$ can be denoted as the time which is required for the entire motion of a manipulator. Assuming that a class of tasks (location of the robot end-effector) is given by $\mathbf{m} \in R^{m \times n}$, the relation between $q$ and $x$ is given by
$$x(t) = f(q(t))$$
where $f$ is a function representing the forward kinematics. Differentiating $x(t)$ with respect to time yields
$$\dot{x}(t) = J(q(t)) \dot{q}(t)$$
$$\ddot{x}(t) = \dot{J}(q(t)) \dot{q}(t) + J(q(t)) \ddot{q}(t)$$

The pseudo-inverse of $J(q(t))$ denoted by $J^+(q(t))$ is defined as follows:
$$J^+ = J^T (JJ^T)^{-1}$$
$$J_n^+ = I_n, I_n \in R^{n \times n}$$

$I_n$ represents the identity matrix, while the pseudo-inverse, $J^+(q)$ as defined above, satisfies the following conditions:
$$JJ^+ J = J, J^+ J^+ J = J^+, (J^+)^T = J^+$$
$$(I_n - J^+) (I_n - J^+) = (I_n - J^+ J)$$
$$J(I_n - J^+ J) = 0$$
$$(I_n - J^+) (I_n - J^+)^T = (I_n - J^+ J)$$
$$(I_n - J^+) J^+ = 0$$

The constraints for the dynamic model are given by
$$M(q(t)) \ddot{q}(t) + V_n(q(t), \dot{q}(t)) + G(q(t)) + F(q(t)) = \tau(t)$$

where $M(q(t)) \in R^{n \times n}$ represents the inertia matrix, $V_n(q(t), \dot{q}(t)) \in R^n$ represents the centripetal-Coriolis matrix, $G(q(t)) \in R^n$ represents the gravity effects, $F(q(t))$ represents the friction effects and $\tau(t) \in R^n$ represents the torque input vector.

The inertia matrix is symmetric, positive-definite, and satisfies the following criteria:
$$m_1 \|\xi\|^2 \leq \xi^T M(q) \xi \leq m_2 \|\xi\|^2 \forall \xi \in R^n$$

The inertia and centripetal-Coriolis matrices satisfy the following skew symmetric relationship:
$$\xi^T \left[\frac{1}{2} M(q) - V_n(q, \dot{q})\right] \xi = 0 \forall \xi \in R^n$$

The task-space tracking error $e(t) \in R^n$ can be defined as:
$$e = x_\circ - x$$
where $x_\circ \in R^n$ represents the desired Cartesian task space. The subtask tracking error $e_s(t) \in R^n$ of the system can be defined as:
$$e_s = (I_n - J^+ J)(g - q)$$

where $g(.)$ is a vector function which is constructed according to the subtask control objectives such as obstacles avoidance, singularity avoidance, etc. [23-36]. The dynamics model of a robot might affect the rotation time of each joint and thus it can be considered as a crucial part in computing robot motion.

2.3 Motion Planning

Basically, there are two types of motion planning for a robot, namely path planning [14-17] and trajectory planning [7-9,12-14,18-22]. However, trajectory planning will be more practical compared to path planning as it includes the motion timing, velocities and accelerations. In terms of trajectory planning, trajectories may be planned in joint space or in Cartesian space. In comparison to joint space trajectories, Cartesian space trajectories can be visualized easily but they are more difficult to calculate and plan. On the other hand, if the robot is not required to follow a specific path, joint space trajectories can be applied, as the trajectories are easier to calculate and realistic motions can be generated.

Mathematically, if kinematics, dynamics, and the trajectory equations are known, the robot motion can be computed. The block diagram for computing the motion of a robot manipulator is illustrated in Figure 1. In Figure 1,
and $\alpha \in R^n$ represent a diagonal, positive definite gain matrix. The output of the controller will be in the form of a nonlinear function and it is used to eliminate or reduce the dynamic tracking errors.

The main disadvantage of using the mathematical model is that it requires extensive computation. Besides, it is crucial to have kinematics and trajectory equations, or else the robot motion might fail to be computed.

Figure 1. Block diagram of computing the motion of a robot manipulator based on the mathematical solution

2.4 Artificial Neural Networks

Artificial intelligence (AI) approaches such as genetic algorithm (GA) [42], neuro-fuzzy [11,21,33] and ANN [10,23,25,28-31,34] have been extensively used for solving robotic problems in various aspects. As stated in the introduction, the computation of robot motion involves solving multi-nonlinear functions and adapting the characteristics of the trajectory. Back-propagation neural network (BPNN) seems to be the most suitable AI approach for solving these issues due to its ability in solving nonlinear functions and its cognitive learning ability, which have been proven by other researchers [10,16,23-31,33-34].

Figure 2. Topology of back-propagation neural network

Basically, there are four steps to develop the BPNN: initialization, activation, weight training and iteration. Initialization is the procedure to set the weights and threshold levels of the network randomly. The BPNN can be activated by presenting a training set of data with inputs $x_1(p), x_2(p),..., x_n(p)$ and desired outputs $y_{d1}(p), y_{d2}(p),..., y_{dn}(p)$. Assume that the BPNN consists of three layers, which are input layer, hidden layer, and output layer. The actual outputs of the neurons in the hidden layer will be

$$y_j(p) = \text{sigmoid} \left[ \sum_{i=1}^{n} x_i(p) \times w_{ij}(p) - \theta_j \right]$$

where $n$ is the number of inputs for neuron $j$ in the hidden layer, and sigmoid is the sigmoid activation function $\text{sigmoid} = 1/(1+e^{-x})$. For the output layers, the actual outputs of the neurons will be

$$y_k(p) = \text{sigmoid} \left[ \sum_{j=1}^{m} x_j(p) \times w_{kj}(p) - \theta_k \right]$$

where $m$ is the number of inputs for neuron $k$ in the output layer. An error signal $e_k(p) = y_{d,k}(p) - y_k(p)$ will be generated from neuron $k$ at the output layer and propagated backward to update the weights.

For updating the weights at the output neurons with learning rate $\alpha$:

Error gradient:

$$\delta_j(p) = y_j(p) \times [1 - y_j(p)] \times e_k(p)$$

Weight correction:

$$\Delta w_{jk}(p) = \alpha \times y_j(p) \times \delta_j(p)$$

Update weight:

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p)$$

Similarly to the output layer, for updating the weights at the hidden neurons with learning rate $\alpha$:

Error gradient:

$$\delta_j(p) = y_j(p) \times [1 - y_j(p)] \times \sum_{k=1}^{m} \delta_k(p) \times w_{jk}(p)$$

Weight correction:

$$\Delta w_{ij}(p) = \alpha \times x_i(p) \times \delta_j(p)$$

Update weight:

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$$

The same procedures can be repeated by increasing the iteration $p$ until the output error criterion is satisfied [42].

3. Methodology

Mathematically, the motion of a robot end effector can be computed based on the block diagram in Figure 1. However, it requires complex calculations to derive the
mathematical equations. Hence, one of the objectives of the current paper is to use multi-BPNNs to solve the kinematics, dynamics, and trajectory issues instead of deriving the mathematical equations. The method in Figure 3 is used to ease the steps of computing the motion of robot end effector, to help individuals with a lack of knowledge about robotic kinematics, dynamics and trajectories. Trajectory equations are not necessary for the block diagram in Figure 3 and thus the user does not really have to spend time finding out the characteristics of the trajectory.

Although BPNN is able to solve the nonlinear functions, it is still difficult for a single multi-layer BPNN to solve the kinematic, dynamic, and trajectory problems simultaneously. Failure to solve these simultaneously might reduce the training speed and accuracy and the precision of the results. Therefore, the novelty of the proposed method is to develop a multi-BPNNs topology by combining multiple BPNNs which handle different tasks at the same time.

![Figure 3. Block diagram of computing the motion of a robot end effector based on multi-BPNNs](image)

**3.1 Topology of Neural Network Used in Current Research**

The multi-BPNNs topology proposed here can be used for predicting the motion of the entire robot manipulator. For a virtual 6-DOF robot manipulator as shown in Figure 4, the motion of the robot manipulator can be computed by using the proposed multi-BPNNs topology in Figure 5. This architecture consists of eight primitive BPNNs (NNi, NN2, NN3, NNI, NNii, NNNi, NNii, and NNNjoint). There are three main parts in the architecture. Firstly, NNI, NNi, and NNii are used to predict the required motion time for each joint. Secondly, NNii, NNNi, and NNNi are used to predict the joint angle of each joint at certain interval time. Lastly, NNi and NNNjoint are used to solve FK. The inputs for BPNNi (NNi) are the initial (θi(t0)) and final position (θi(tf)) of joint 1 in joint space. Y1 is the output of NNi which is the required time for the rotation of joint 1. H1 is the outcome of function h(ti,Y1) where h is an arbitrary interval time and ti ∈ [t0,tf]. The main function of h(ti,Yl) is to control the input of NNi.

$$H1 = h(t_i, Y_i) = \begin{cases} \frac{t_i}{Y_i} & t_i \leq Y_i \\ Y_i & t_i > Y_i \end{cases}$$

Function h(ti,Yl) is used in conjunction with several BPNNs (Figure 5) to overcome the weakness of BPNN in adapting two different functions simultaneously. The occurrence of two different functions arises when the required rotation times between joints 1 and 2 are different. As shown in Figure 6, joint 1 is rotating from θ1(t0) to θ1(tf) and the required rotation time is t1. Meanwhile, joint 2 is rotating from θ2(t0) to θ2(tf) and the required rotation time is t2. The variation of angle for joint 1 from time 0 to t1 is a completely nonlinear function and thus the characteristic of the function can be adapted by using BPNN and the position (joint space) of joint 1 at any interval time can be predicted. Conversely, there are two portions for joint 2, portion A (nonlinear function) and portion B (linear function). It is a linear function in portion B because joint 2 remains in static mode after completing its rotation. For joint 2, BPNN is unable to predict the position of joint 2 at any interval time correctly because it consists of two different functions. In order to increase the accuracy of the results, the position of joint 2 in portion A will be predicted by using ANNI and the position of joint 2 in portion B will be computed by using function h(t2,Y2). The use of function h(t2,Y2) is to let the model to understand that there is a linear function in portion B, where t2<t11. The function of h(t2,Y2) only can be neglected if the rotation time for each joint is the same. The interconnection between NNI and NNI is illustrated in Figure 7. The output of ANNi will be the angle of joint 1 at interval time ti and ANNi is used to calculate the position (FK) of the end effector in Cartesian space at interval time ti. In comparison to the conventional single multilayer BPNN in Figure 2, this multi-BPNNs topology produces better results because this topology combines multiple BPNNs to handle different tasks. NNI, NNi, and NNii are able to predict the rotation time, which is normally handled by a complex dynamic model, while NNI, NNI, and NNNi are able to solve the trajectory issue without deriving the trajectory equations.

Similarly, this architecture can be modified and changed to the block diagram in Figure 8 for computing the orientation of the robot’s end effector. Eventually, by using parametric equations the robot’s axial equation for
each link can be derived as long as the positions of the connected joints are known. This is important for knowing the motion of the robot’s links. The position and vector for joint \( m \) and joint \( n \) at an arbitrary interval time \( t \) are

\[
\begin{align*}
    \mathbf{r}_{E,m}(t) &= \left[ p_{x,m} p_{y,m} p_{z,m} \right]^T, \\
    \mathbf{r}_{E,n}(t) &= \left[ p_{x,n} p_{y,n} p_{z,n} \right]^T, \\
    \mathbf{OP}_m &= [a_m b_m c_m]^T \\
    \mathbf{OP}_n &= [a_n b_n c_n]^T
\end{align*}
\]

The link equation that connects joints \( n \) and \( m \) would be

\[
\begin{align*}
    \frac{x-p_{x,n}}{a_n-a_m} &= \frac{y-p_{y,n}}{b_n-b_m} = \frac{z-p_{z,n}}{c_n-c_m} = k
\end{align*}
\]

Figure 4. Configuration and DH table for virtual robot manipulator

Figure 5. Topology of multi-BPNNs for computing robot motion

Figure 6. Joint position in joint space

4. Results and Discussion

The only limitation of this method is that a person has to collect around 150-200 samples for training each of the BPNNs, which may be considered a time-consuming process in the early stage. The activation function for all the neurons is the sigmoid function; the preliminary experiments showed that for sigmoid activation function, a constant learning rate of 0.5 and a momentum term of 0.9 worked well for this architecture. The mean square error (MSE) of the trained results is close to 0 (approximately \( 1.5 \times 10^{-4} \)). After the BPNNs are trained, the location for each actuator in Cartesian space can be predicted through the block diagram in Figure 5, and the robot link equations can be derived by using equation (9). The general flow of this method is shown in Figure 9.

For an example, the actuators for the virtual robot manipulator in Figure 4 were subjected to the control of unknown trajectories in the aspects of positions, velocities or accelerations. The parameters of an unknown trajectory such as velocity, acceleration, torque, etc., were set randomly by another user. It was assumed that the only information that could be obtained was the positions of the initial point and end point in Cartesian space. Fifty samples with different initial and end points for the end effector were used to test the reliability of the developed ANN topology in predicting the motion of the robot. The results (10 out of 50) are shown in Figure 10.
In comparison to the actual results, the results in Figure 10 clearly show that for all the samples, the maximum deviations of the motion for the virtual robot in Cartesian space were less than 0.5 units, while the maximum deviations of orientation were less than 0.1 units. Besides, the root mean square errors (RMSE) were small (maximum=0.2), proving that this topology possessed high accuracy and repeatability in tracking the position and orientation of the robot.

For sample 10, the end effector of the robot was required to move from point A (-3.41, -0.6, 19.62) to point B (13.91, 3.73, 5.24). The motion of the robot manipulator was predicted and shown in Figure 11. The robot manipulator started from rest and stopped at its destination, point B, and took 12 seconds to complete its motion. In other words, this developed ANN is not only able to predict and track the motion of the virtual robot manipulator but also to predict the time taken for the motion. Figures 12 and 13 show the comparison between multi-BPNNs and the actual results for the variation of the robot end effector’s position and orientation in Cartesian space. The outputs from developed multi-BPNNs were almost the same as the actual hardware outputs, where the multi-BPNNs output lines almost overlapped with the actual output lines and the absolute value of maximum deviation was less than 0.5 units.

In comparison to the conventional mathematical method, this method can be applied easily without deriving any complex mathematical equations, such as FK equations and trajectory equations. Besides, the main advantage of this method is that it can be used to predict the motion of a robot manipulator even though the characteristics of the joint trajectories are unknown. Other than that, this multi-BPNNs topology is able to solve dynamics, kinematics, and trajectory simultaneously and track the motion of the entire robot manipulator instead of only the end effector.
5. Conclusions

Motion planning is an important task to prevent the occurrence of accidents. However, by using mathematical solutions, the motion of a robot manipulator might become unpredictable or untraceable if the characteristics of the trajectories are unknown. Complexity of mathematical equations also means they are prone to error. Hence, this paper recommends multi-BPNNS for overcoming these limitations. The developed ANN topology is not only able to adapt and predict the motion of a robot manipulator with unknown trajectories but also for robot orientation. The results showed that the developed multi-BPNNS topology produced accurate (maximum deviation ≤ 0.45 units) and precise (RMSE ≤ 0.2) results in adapting and predicting the motion of a robot manipulator with unknown trajectory.

6. References

[1] Solea R, Cernega DC (2013) Mobile Manipulators Motion Planning Based on Trajectory Tracking Control, Informatics in Control, Automation and Robotics. New York: Springer. pp. 77-88.

[2] Xie HB, Duan XM, Yang HY, Liu ZB (2012) Automatic Trajectory Tracking Control of Shield Tunneling Machine under Complex Stratum Working Condition. Tunn.undergr.space technol.32: 87-97.

[3] Carbone G, Gómez-Bravo F, Selvi O (2012) An Experimental Validation of Collision-Free Trajectories for Parallel Manipulators. Mech. based des. struc.40: 414-433.

[4] Pan J, Zhang LJ, Manocha D (2012) Collision-free and Smooth Trajectory Computation in Cluttered Environments. Int. j. robot res.31: 1155–1175.

[5] Capisani LM, Ferrara A (2012) Trajectory Planning and Second-Order Sliding Mode Motion/Interaction Control for Robot Manipulators in Unknown Environments. IEEE t. ind. electron.59: 3189-3198.

[6] Korayem MH, Rahimi HN, Nikoobin A (2012) Mathematical Modeling and Trajectory Planning of Mobile Manipulators with Flexible Links and Joints. Appl. math. model.36: 3229-3244.

[7] Macfarlane S, Croft EA (2003) Jerk-bounded Manipulator Trajectory Planning: Design for Real-Time Applications. IEEE trans. robotics autom.19: 42-52.

[8] Sariyildiz E, Temeltas H (2011) Performance Analysis of Numerical Integration Methods in the Trajectory Tracking Application of Redundant Robot Manipulators. Int. j. adv. robot. syst.8: 25-38.

[9] Liu K, Kujath MR (1996) Trajectory Optimization for A Two-Link Flexible Manipulator. Int. j. robot automat., 11: 56-61.

[10] Kumar N, Panwar V, Sukanam N, Sharma SP and Born JH (2011) Neural Network-based Nonlinear Tracking Control of Kinematically Redundant Robot Manipulators. Math. comput. model.53: 1889-1901.

[11] Hauser K, Victor Ng TH (2010) Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. Proceedings of IEEE International Conference on Robotics and Automation (ICRA). 2010 May 3-7; Anchorage, Alaska. IEEE: U.S.App.2493-8.

[12] Korayem MH, Nikoobin A, Azimirad V (2009) Trajectory Optimization of Flexible Link Manipulators in Point-To-Point Motion. Robotica27: 825-840.

[13] Saravanan R, Ramabalan S, Balamurugan, C (2008) Evolutionary Collision-Free Optimal Trajectory Planning for Intelligent Robots. Int. j. adv. manuf. tech.36: 1234-1251.

[14] Lo Bianco CG, Piazzzi A (2002) Minimum-Time Trajectory Planning of Mechanical Manipulators under Dynamic Constraints. Int. j. control.75: 967-980.

[15] Kohrt C, Stamp R, Pipe AG, Kiely J, Schiedermeier G (2013) An Online Robot Trajectory Planning and Programming Support System for Industrial Use. Robot. cim-int. manuf.29: 71–79.

[16] Zhang YN, Yin JP, Cai BH (2009) Infinity-Norm Acceleration Minimization of Robotic Redundant Manipulators Using the LVI-based Primal-Dual Neural Network. Robot. cim-int. manuf.25: 358–365.

[17] Boudreau R, Nokleby S (2012) Force Optimization of Kinematically-Redundant Planar Parallel Manipulators Following A Desired Trajectory. Mech. mach. theory56: 138-155.

[18] Rossi C, Savino S (2012) Robot Trajectory Planning by Assigning Positions and Tangential Velocities. Robot. cim-int. manuf. 29: 139-156.

[19] HsuYL, Huang MS, Fung RF (2012) Minimum-absolute-energy trajectory planning for a toggle mechanism driven by a PMSM. Proceeding of 2012 International Conference on System Science and Engineering (ICSSE). 2012 Jun 30-Jul 2; Dalian, China.IEEE: U.S.A, pp.193-8.

[20] Briot S, Arakelian V, Le Baron JP (2012) Shaking Force Minimization of High-Speed Robots via Centre of Mass Acceleration Control. Mech. mach. theory57: 1-12.

[21] Agarwal V (2012) Trajectory Planning of Redundant Manipulator Using Fuzzy Clustering Method. Int. j. adv. manuf. tech.61: 727-744.

[22] Boryga M, Graboś A (2009) Planning of Manipulator Motion Trajectory with Higher-Degree Polynomials Use. Mech. mach. theory44: 1400-1419.

[23] Shah J, Rattan S S, Nakra BC (2011) Kinematic Analysis of 2-DOF Planar Robot Using Artificial Neural Network. World acad. sci. eng. technol.81: 282-285.
[24] Zarkandi S, Vafadar A, Esmaili MR (2011) PRRRRRP redundant planar parallel manipulator: kinematics, workspace and singularity analysis. Proceedings of 2011 IEEE 5th International Conference on Robotics, Automation and Mechatronics (RAM). 2011Sep 17-19; Qingdao, China. IEEE: U.S.A, pp. 61-6.

[25] Zhang D, Lei J (2011) Kinematic Analysis of A Novel 3-DOF Actuation Redundant Parallel Manipulator Using Artificial Intelligence Approach. Robot. cim.-int. manuf.27: 157-163.

[26] Hassan AT, Hamouda AMS, Ismail N, Al-Assadi HMAA (2006) An Adaptive-Learning Algorithm to Solve the Inverse Kinematics Problem of A 6 D.O.F Serial Robot Manipulator. Adv. eng. softw.37 (7): 432-438.

[27] Van Hentan EJ, Schenk EJ, Willigenburg LG, Meuleman J, Barreiro P (2010) Collision-free Inverse Kinematics of the Redundant Seven-Link Manipulator Used in A Cucumber Picking Robot. Biosyst. eng.106: 112-124.

[28] Al-Faiz MZ (2007) Inverse Kinematics Solution for Robot Manipulators Based on Neural Network. MASAUM j. basic appl. sci.1 (2): 147-154.

[29] Kim B (2014) H. An Adaptive Neural Network Learning-Based Solution for the Inverse Kinematics of Humanoid Fingers. Int. j. adv. robot. syst.11: 1-9.

[30] Park H, Lee S, Chu B, Hong D (2008) Obstacle avoidance for robotic excavators using a recurrent neural network. Proceedings of the IEEE International Conference on Smart Manufacturing Application. 2008Apr 9-11; Gyeonggi-do, South Korea. IEEE: U.S.A, pp.585-90.

[31] Wang J (2008) Obstacle avoidance for kinematically redundant manipulators based on recurrent neural networks. In: Xiong CH, Huang YA, Xiong YL, Liu HH, editors. Intelligent Robotics and Applications. First International Conference on Intelligent Robotics and Applications. 2008 Oct 15-17; Wuhan, China. Springer: Berlin, pp.10-3.

[32] Zhang YB, Liu HZ, Wu X (2009) Kinematics Analysis of A Novel Parallel Manipulator. Mech. mach. theory44: 1648-1657.

[33] Mayorga RV, Chandana S (2006) A neuro-fuzzy approach for the motion planning of redundant manipulators. Proceedings of International Joint Conference on Neural Networks. 2006 Jul 16-21; Vancouver, Canada. IEEE: U.S.A, pp. 2873-8.

[34] Liu S, Wang J (2005) Obstacle avoidance for kinematically redundant manipulators using the deterministic annealing neural network. In: Wang J, Liao, XF, Yi Z, editors. Advances in Neural Networks-ISNN 2005. Second International Symposium on Neural Networks. 2005 May 30-Jun 1; Chongqing, China. Springer: Berlin, pp. 240-6.

[35] Wang, J.G., Li, Y.M., and Zhao, X.H. Inverse kinematics and control of a 7-DOF redundant manipulator based on the closed-loop algorithm. Int. J. Adv. Robot. Syst., 2010, 7, 1-9.

[36] Zhang Y, Wang J (2004) Obstacle Avoidance for Kinematically Redundant Manipulators Using A Dual Neural Network. IEEE trans. sys. man cybern. part b: cybernetics34 (1): 752-759.

[37] Kosko B (1987) Adaptive Bidirectional Associative Memories. Appl.Optics 26: 4947-4960.

[38] Hebb DO (1949) The Organization of Behavior: A Neuropsychological Theory. New York: John Wiley.

[39] Fukushima K (1975) Cognition: A Self-Organizing Multilayered Neural Network. Biol. cybern. 20: 121-136.

[40] Von der Malsburg C (1973) Self-organisation of Orientation Sensitive Cells in the Striate Cortex. Kybernetik 14: 85-100.

[41] Grossberg S (1972) Neural Expectation: Cerebellar and Retinal Analogs of Cells Fired by Learnable or Unlearned Pattern Classes.Kybernetik 10: 49-57.

[42] Negnevitsky M (2005) Artificial Intelligence: A Guide to Intelligent Systems, 2nd Edition. London: Addison Wesley Publishing Company Incoroporated.

[43] Payeur P, Hoang LH, Gosselin CM (1995) Trajectory Prediction for Moving Objects Using Artificial Neural Networks. IEEE trans. ind. electron. 42 (2): 147-158.

[44] Seyr M, Jakubek S, NovakG (2005) Neural network predictive trajectory tracking of an autonomous two-wheeled mobile robot. In: Zitek, P. Proceedings of the 16th IFAC World Congress. 2005 Jul 4-8; Czech Republic. International Federation of Automatic Control: Czech Republic, pp. 1333-8.

[45] Le Fablec Y, Alloit JM (1999) Using neural networks to predict aircraft trajectories. Proceedings of the International Conference on Artificial Intelligence (ICAI’99). 1999 Jun 28-Jul 1;Las Vegas, U.S.A. CSREA Press.