Available online at www.sciencedirect.com
ScienceDirect
Procedia Technology 12 (2014) 20 – 27
Procedia Technology

The 7th International Conference Interdisciplinarity in Engineering (INTER-ENG 2013)

Neural network based inverse kinematics solution for trajectory tracking of a robotic arm

Adrian-Vasile Duka*

"Petru Maior" University of Tg. Mureş, No. 1 N.Iorga St., Tg.Mureş, 540088, Romania

Abstract

Planar two and three-link manipulators are often used in Robotics as testbeds for various algorithms or theories. In this paper, the case of a three-link planar manipulator is considered. For this type of robot a solution to the inverse kinematics problem, needed for generating desired trajectories in the Cartesian space (2D) is found by using a feed-forward neural network.

Keywords: robotic arm; planar manipulator; inverse kinematics; trajectory; neural networks

1. Introduction

In the forward kinematics problem the end-effecter's location in the Cartesian space (or work space), that is its position and orientation, is determined based on the joint variables. The joint variables are the angles between the links, in the case of rotational joints, or the link extension, in the case of prismatic joints. Conversely, given a desired end-effecter position and orientation, the inverse kinematics problem refers to finding the values of the joint variables that allow the manipulator to reach the given location.

The relationship between forward and inverse kinematics, as well as the relationship between joint space and Cartesian space is shown in Fig. 1.

* Corresponding author. Tel.: +40-0265-233212
*E-mail address:*adrian.duka@ing.upm.ro

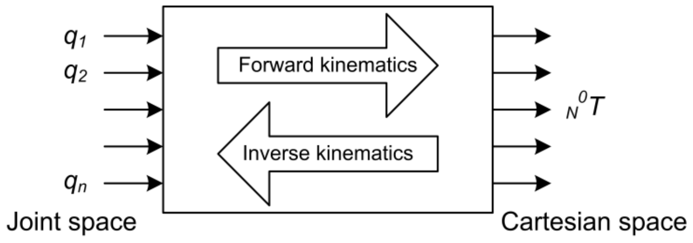


Fig. 1. Forward and inverse kinematics.[6,16]

Solving the inverse kinematics problem for robotic manipulators is a difficult and also quite challenging task. The complexity of this problem is given by the robot's geometry and the nonlinear trigonometric equations that describe the mapping between the Cartesian space and the joint space [6,12,18,21].

Although a closed form solution to this problem is preferable in many applications, most of the time this is impossible to find. Therefore, various other ways to determine the solution to the inverse kinematics problem were proposed. These include, among others, listed in [3], geometrical solutions (where possible), numerical algorithms based on optimization techniques [7,20,24], evolutionary computing [14,15,19,21,23] or neural networks [5,10,13]

Neural networks have long been recognized as being able to represent non-linear relationships that occur between input and output data. Their ability to learn by example make them a good candidate to provide the mapping between the Cartesian space and the joint space required by the inverse kinematics problem.

In [22] several neural network structures used for solving the inverse kinematics problem are analyzed. These include backpropagation trained feed-forward neural networks or neural networks whose weights are defined in terms of *sin* and *cos* to fit the forward kinematics representation of the robot.

This paper investigates the use of a neural network to produce the solution to the inverse kinematics problem for a three-link robotic manipulator. The neural network is trained using the data provided by the forward kinematics to learn the inverse forward mapping of the configuration space. This means the end effecter's position and orientation are given as inputs and the neural network identifies which joint configuration corresponds to the given localization of the end effecter.

2. Problem formulation

A hypothetical robotic arm is considered as testbed for the proposed algorithm. It is a three-link planar manipulator with rotational joints whose links have all the same size:

$$l_1 = l_2 = l_3 = 2 \tag{1}$$

and whose joint movements are limited within the following ranges:

$$\begin{aligned} q_1 &\in [0, \pi] \\ q_2 &\in [-\pi, 0] \\ q_3 &\in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \end{aligned} \tag{2}$$

Fig. 2 shows a representation of the planar robotic arm. The arm consists of three movable links, $l_1 = \overline{OA}$, $l_2 = \overline{AB}$ and $l_3 = \overline{BE}$ that move within a plane. These links are connected by rotational joints whose axes are all perpendicular to the plane of the links.

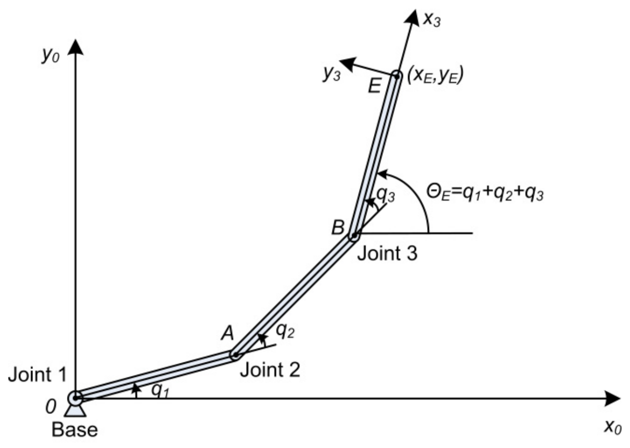


Fig. 2. A three link planar manipulator

The forward kinematic equations of the robotic arm are given in (3).

$$\begin{aligned} x_E &= l_1 \cos q_1 + l_2 \cos(q_1+q_2) + l_3 \cos(q_1+q_2+q_3) \\ y_E &= l_1 \sin q_1 + l_2 \sin(q_1+q_2) + l_3 \sin(q_1+q_2+q_3) \end{aligned} \tag{3}$$

Equations (3) define the coordinates of the end-effecter in the frame attached to the base of the robot *(Ox_0y_0)*.

To completely locate the end-effecter within the plane its desired position and orientation are needed. While position is defined by equations (3), the orientation of the end-effecter can be described as the angle of rotation of the frame attached to the end-effecter relative to the fixed frame attached to the base of the robot. The end-effecter's orientation Θ_E is related to the actual joint displacements as:

$$\Theta_E = q_1 + q_2 + q_3 \tag{4}$$

Equations (3) and (4) describe the position and orientation of the robot end-effecter viewed from the fixed coordinate system attached to the base of the robot in relation to the joint variables q_1, q_2, q_3 [4]

The robotic arm is required to perform a circular trajectory in its workspace. The points on this trajectory are defined using the parametric form of the equation of the circle as:

$$\begin{aligned} x_p &= x_C + r\cos\varphi \\ y_p &= y_C + r\sin\varphi \\ \varphi &= [0:2\pi] \end{aligned} \tag{5}$$

where r is the radius of the circle, (x_c,y_c) are the coordinates of the circle's centre and φ is the parametric variable in the range 0 to 2π.

Equations (5) correspond to the desired position of the end-effecter. The desired orientation of the end-effecter is expressed by the angle between the positive x coordinate axis and the line that connects the origin of the frame attached to the base of the robot and a point on the circle trajectory. This is equivalent to:

$$\Theta_d = atan\left(\frac{y_p}{x_p}\right) \tag{6}$$

To produce the above specified trajectory the computation of the inverse kinematics is necessary for each considered point on the circle.

3. The neural network

To compute the solution to the inverse kinematics problem a feed-forward neural network is proposed. In order to produce the training data for the neural network, random joint angle values that uniformly cover the ranges specified in equation (2) are generated. To each set $(q_1, q_2, q_3)^T$ of joint angle values, by using the forward kinematics equations (3) and (4), the corresponding localization of the end-effecter is computed. The resulting values are stored in a vector of the form $(x_E, y_E, \Theta_E)^T$

Fig. 3 shows the resulting position location of the end-effecter (*: position (x_E, y_E),—: orientation Θ_E) for 1000 random generated sets of joint angle values. As seen, by accumulating the resulting values, the actual workspace of the robot is shaping up. Three random poses of the robotic arm are also shown in Fig. 3

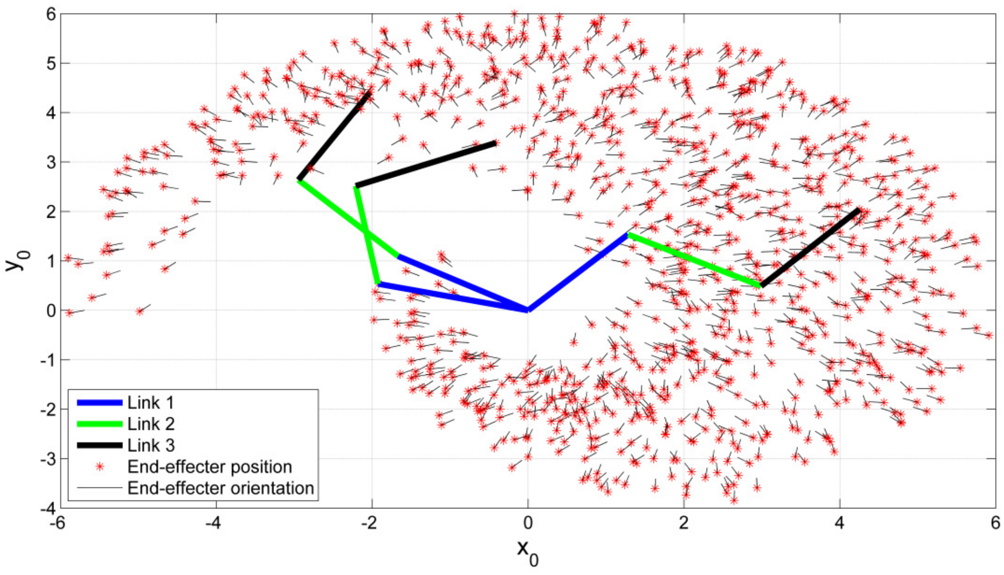


Fig. 3.The resulting localization of the end-effecter for random generated joint parameters

The process of finding a solution to the inverse kinematics problem, in this case, can be seen as an inversion problem for neural networks. This means, finding the inputs (joint parameters) that yield a desired output (localization of the end-effecter)[17,22]. For this purpose, the neural network is trained using the training data obtained according to the procedure mentioned above. For the training stage, the input data is the localization vector $(x_E, y_E, \Theta_E)^T$ resulted from the forward kinematics and the target data is its corresponding joint parameters set $(q_1, q_2, q_3)^T$ of randomly generated values.

Since, for the same localization in space of the end-effecter, the inverse kinematics problem has multiple solutions, duplicates in the neural network input data are identified and the corresponding training set (input data and target data) is removed. This way each configuration the neural network is learning refers to a unique mapping from Cartesian space to joint space.

The training set is subjected to a final processing stage, before being used for the actual training, that scales all the available values to the [-1,1] interval.

The structure of the neural network used to learn the inverse kinematics of the robot is shown in Fig. 4.

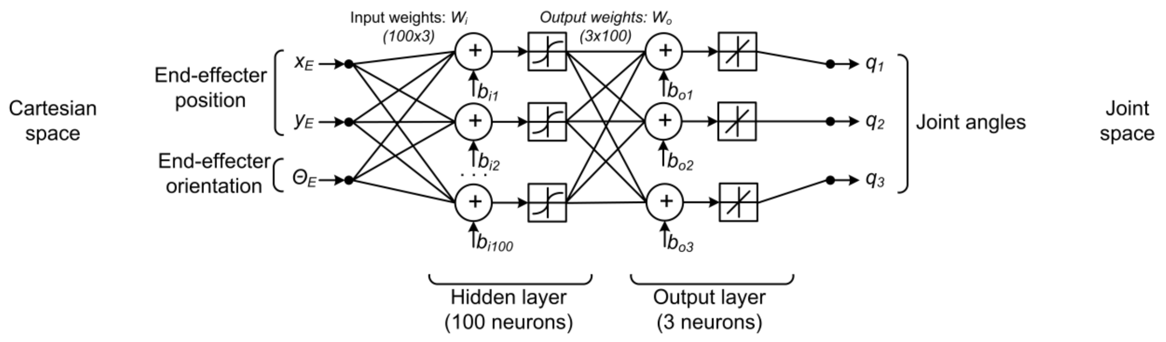


Fig. 4. The structure of the neural network

This feed-forward neural network consists of 3 inputs, 100 neurons in the hidden layer and 3 neurons in the output layer. The transfer function for the neurons in the hidden layer is the hyperbolic tangent sigmoid, shown in equation (7) and for the output neurons is the linear function, shown in equation (8).

$$y = \tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{7}$$

$$y = x \tag{8}$$

The output of the neural network represents a 3 by 1 vector of joint angles corresponding to the three joints of the robotic arm. It is determined based on the desired position and orientation of the end-effecter, according to equation (9).

$$\begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} = \underline{W_o} \cdot \tanh\left(\underline{W_i} \cdot \begin{pmatrix} x_E \\ y_E \\ \Theta_E \end{pmatrix} + \underline{B_i} \right) + \begin{pmatrix} b_{o1} \\ b_{o2} \\ b_{o3} \end{pmatrix} \tag{9}$$

where: $\underline{W_i}$ is100 by 3 matrix containing the weights in the hidden layer, $\underline{B_i} = (b_{i1}, b_{i2}, \ldots, b_{i100})^T$ is the vector of biases for the neurons in the hidden layer; $\underline{W_o}$ is 3 by 100 matrix of the weights in the output layer and $(b_{o1}, b_{o2}, b_{o3})^T$ are the biases for output neurons.

The Levenberg-Marquardt training algorithm was used, which is known to assure fast convergence of the training error [1, 2, 8 ,9, 11, 25], and is also a very popular curve-fitting algorithm [8].

From the set of 1000 collected samples 15% are used for validation and 15% for testing the neural network. The other 70% of the samples are used for the actual training of the neural network.

The performance of the neural network was determined based on the mean squared error between the neural network's actual output and the desired output. It's evolution is shown in Fig. 5 for the three categories of samples mentioned before.

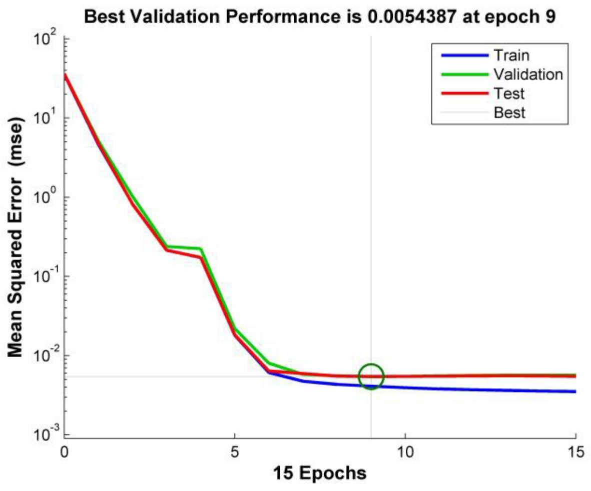


Fig. 5. The neural network's performance.

4. Results

Once the training is done, the coordinates and orientation of the points on the desired trajectory in the Cartesian space can be applied as inputs to the neural network, which will produce the joint parameters that place the end-effecter on the required trajectory.

Fig. 6 shows the resulting joint space trajectories corresponding to the circular trajectory defined in the Cartesian space, which was specified by equation (5).

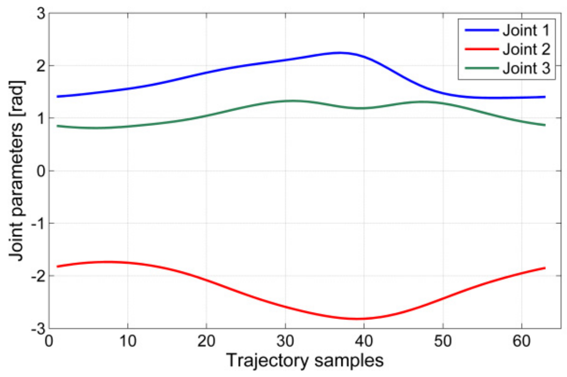
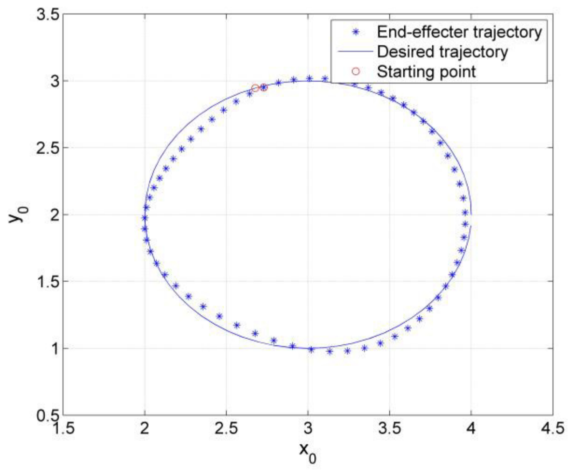


Fig. 6. Joint space trajectories.

The points on the joint space trajectories can be easily interpolated to 5th order polynomials and the trajectory repeated over and over, without having to appeal each time to the computational requirements of the neural network

Fig. 7 shows the tracking capabilities of the robot whose joint parameters are outputted by the neural network based on the desired trajectory in the Cartesian space. Fig 7.a shows how the end-effecter tracks the desired position and Fig. 7.b shows the end-effecter's tracking capabilities for the desired orientation.



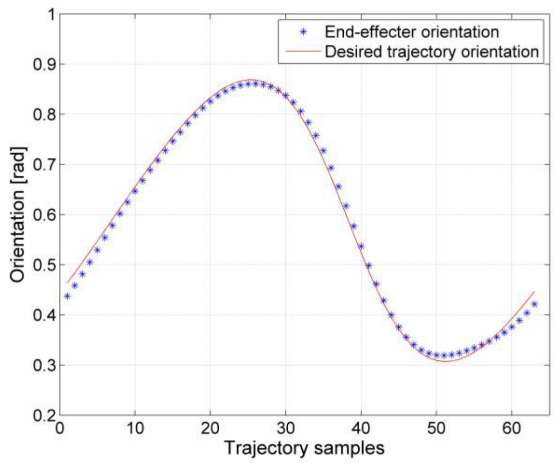


Fig. 7. (a) End-effecter position; (b) End-effecter orientation.

Fig. 8 provides an overall image of the system, showing a representation of the robot tracking the desired trajectory.

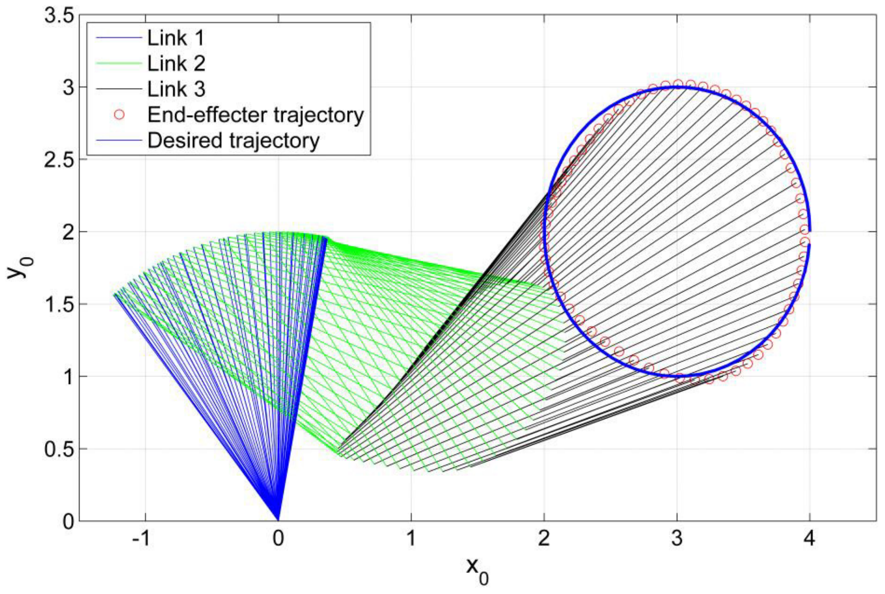


Fig. 8. Three-link planar manipulator trajectory tracking

5. Conclusion and future work

All results shown in the previous section were obtained using Matlab modeling of the proposed system.

The inverse kinematics problem for a three-link robotic manipulator was transformed in a fitting problem, in which a neural network was used to map between a data set of numeric inputs and a set of numeric targets. The data sets, used for training the neural network, were determined based on the forward kinematics equations of the robot and the neural network was trained to solve an inversion problem, producing the inputs (joint parameters) that match the outputs (localization of the end-effecter).

As seen, the two-layer feed-forward network with sigmoid hidden neurons and linear output neurons can fit multi-dimensional mapping problems arbitrarily well, given consistent data and enough neurons in its hidden layer.

Although the results show promising trajectory tracking capabilities, the problem of using a neural network for the inverse kinematics should be further addressed. There are several directions that require further attention, such as optimizing the size / structure of the neural network, improving the performance of the neural network, and, of course, applying the algorithm to real robotic manipulators.

References

[1] Al-Alaoui M.A., et al. A Cloning Approach to Classsifier Training, IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans, 2002; 32(6).
[2] Alba E, Chicano JF. Training Neural Networks with GA Hybrid Algorithms, Lecture Notes in Computer Science Vol. 3102 / 2004, Springer Berlin / Heidelberg, 2004: 852-863.
[3] Aristidou A, Lasenby J. Inverse Kinematics: a review of existing techniques and introduction of a new iterative fast solver. Cambridge University Engineering Department, Technical Report, 2009.
[4] Asada HH. Introduction to Robotics. Department of Mechanical Engineering, Massachusetts Institute of Technology.
[5] Bassam Daya, Shadi Khawandi, Mohamed Akoum. Applying Neural Network Architecture for Inverse Kinematics Problem in Robotics. J. Software Engineering & Applications, 2010; (3): 230-239
[6] Bingul Z, Ertunc HM, Oysu C. Applying Neural Network to Inverse Kinematic Problem for 6R Robot Manipulator with Offset Wrist. Adaptive and Natural Computing Algorithms, 2005: 112-115.
[7] Courty N, Arnaud E. Inverse Kinematics Using Sequential Monte Carlo Methods. Proceedings of the 5th international conference on Articulated Motion and Deformable Objects, 2008: 1-10
[8] Demuth H, Beale M, Hagan M. Neural Network Toolbox 6. User's Guide", The Mathworks, Inc., 2008.
[9] Finschi L. An Implementation of the Levenberg-Marquardt Algorithm", Eidgenossische Technische Hochschule Zurich, 1996.
[10] Gardner JF, Brandt A, Luecke G. Applications of Neural Networks for Coordinate Transformations in Robotics, Journal of Intelligent and Robotic Systems 8, Kluwer Academic Publishers, 1993: 361-373.
[11] Hagan MT, Menhaj MB. Training Feedforward Networks with the Marquardt Algorithm. IEEE Transactions on Neural Networks, 1994; 5(6).
[12] Ali T Hasan, Hayder MAA Al-Assadi, Ahmad Azlan Mat Isa, Neural Networks' Based Inverse Kinematics Solution for Serial Robot Manipulators Passing Through Singularities", Artificial Neural Networks. Industrial and Control Engineering Applications, Prof. Kenji Suzuki (Ed.), InTech, 2011.
[13] Jack H, Lee DMA, Buchal RO. Elmaraghy WH. Neural networks and the inverse kinematics problem. Journal of Intelligent Manufacturing 4, 1993: 43-66.
[14] Kalra P, Mahapatra PB, Aggarwal DK. An evolutionary approach for solving the multimodal inverse kinematics problem of industrial robots, Mechanism and Machine Theory 41, Elsevier, 2006:1213-1229
[15] Raşit Köker. A neuro-genetic approach to the inverse kinematics solution of robotic manipulators. Scientific Research and Essays Vol. 6 (13), 2011: 2784-2794
[16] Kucuk S, Bingul Z, Robot Kinematics: Forward and Inverse Kinematics. Industrial Robotics: Theory, Modelling and Control, Sam Cubero (Ed.), InTech., 2006
[17] Linden A, Kindermann J. Inverting of multilayer nets. IJCNN, 2, 1989.
[18] Oltean S, Şoaita D, Dulau M, Jovrea T, Aspects of Interdisciplinarity in the MRIP 02 System Design, International Conference "Optimization of the Robots and Manipulators" OPTIROB 2007, Bren Publishing House, 2007: 121-125
[19] J Ramirez A, A Rubiano F. Optimization of Inverse Kinematics of a 3R Robotic Manipulator using Genetic Algorithms, World Academy of Science, Engineering and Technology 59, 2011: 1425- 1430
[20] Secară C, Dumitriu D. Direct Search Based Strategy for Obstacle Avoidance of a Redundant Manipulator. Analele Universităţii "Eftimie Murgu" Reşiţa, Anul XVII, Nr. 1, 2010: 11-20
[21] Secară C, Vlădăreanu L, Iterative genetic algorithm based strategy for obstacles avoidance of a redundant manipulator. Proceedings of the 2010 American conference on Applied mathematics, 2010: 361-366
[22] Sreenivas Tejomurtula, Subhash Kak, Inverse kinematics in robotics using neural Networks. Information Sciences 116, 1999: 147-164
[23] Tabandeh S, Clark C, Melek W. A Genetic Algorithm Approach to solve for Multiple Solutions of Inverse Kinematics using Adaptive Niching and Clustering. Evolutionary Computation, 2006. CEC 2006. IEEE Congress on , 16-21 July 2006: 1815-1822.
[24] Wang L-CT, Chen CC. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. Robotics and Automation, IEEE Transactions on, 1991; 7(4): 489-499.
[25] Wilamowski BM, Iplikci S. An Algorithm for Fast Convergence in Training Neural Networks, Proceedings of the International Joint Conference on Neural Networks (IJCNN '01), 2001: 1778-1782.