INT J COMPUT COMMUN, ISSN 1841-9836
9(3):253-260, June, 2014.

Solving Continuous Trajectory and Forward Kinematics Simultaneously Based on ANN

C.K. Ang, S.H. Tang, S. Mashohor, M.K.A.M. Arrifin

Chun Kit Ang*, Sai Hong Tang,
Syamsiah Mashohor, Mohd Khairol Anuar Mohd Arrifin
Universiti Putra Malaysia
Department of Mechanical and Manufacturing Engineering,
Faculty of Engineering, 43400 UPM, Serdang, Selangor Darul Ehsan.
Email: ack_kit@hotmail.com, saihong@upm.edu.my,
syamsiah@upm.edu.my, khairol@upm.edu.my
*Corresponding author

Abstract: Robot movement can be predicted by incorporating Forward Kinematics (FK) and trajectory planning techniques. However, the calculations will become complicated and hard to be solved if the number of specific via points is increased. Thus, back-propagation artificial neural network is proposed in this paper to overcome this drawback due to its ability in learning pattern solutions. A virtual 4-degree of freedom manipulator is exploited as an example and the theoretical results are compared with the proposed method.
Keywords: Artificial Neural Network (ANN), back propagation neural network, forward kinematics, continuous trajectory.

1 Introduction

Trajectory planning can be considered as an important issue in robot industry. Without an appropriate trajectory planning, a robot's motion will become unpredictable and it may collide with obstacles or go through undesirable points. Thus, an appropriate trajectory planning enables us to determine the required workspace and provides us an opportunity to avoid obstacles.

Basically, there are two types of trajectory planning, point to point and continuous trajectory (with via points). However, continuous trajectory planning will be more practical compare to point to point trajectory because it allows robot to pass through various specific points and it is useful for real time robot.

Conventionally, there are a few methods that can be used for continuous trajectory planning such as Bezier function, Linear Segments with Parabolic Blends (LSPB), and high order polynomials. Unfortunately, those methods require extensive calculations when the number of specific points is increased and thus leading to the emergence of a large number of formulas ([1]- [5]). Hence, it is advisable to develop a method which is able to overcome this drawback.

2 Preliminaries

Forward kinematics (FK) can be used to determined the position and orientation of a robot's hand if all the configurations are known and the equations can be derived by using Denavit-Hartenberg (DH) method because it is the standard way of modelling the robot motion due to its simplicity of modelling robot links and joints that is applicable for any robot configuration s regardless of its complexity. The total transformation for a robot manipulator from base to hand will be

$$ {}^{R}T_{H} = {}^{R}T_{1}^{1}T_{2}^{2}T_{3}...^{n-1}T_{n} = A_1A_2A_3...A_n = \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1} $$

and the end effector can be defined as:

$$ r_e(t) = f_e(\theta(t)) \tag{2} $$

where $r_e \in R^m$ in Cartesian space is related to the joint space vector, m is dimensional position vector of end effector.

In fact, what we are actually interested in is the inverse kinematics (IK) because it allows us to determine the value of each joint in order to place the arm at a desired position and orientation. The inverse kinematic problem is to find the joint variable θ for any given$r_e(t)$. Unfortunately, it is impossible to find an analytic solution due to the nonlinearity of$f(\theta(t))$in reality. The inverse kinematics problem of manipulators is thus usually solved at the velocity level. Differentiating $r_e(t) = f_e(\theta(t))$ with respect to time yields a linear relation between the Cartesian velocity $\dot{r}_e$and the joint velocity$\dot{\theta}$:

$$ J_e(\theta)\dot{\theta} = \dot{r}_e \tag{3} $$

where $J_e(\theta) \in R^{m\times n}$ is the end effector Jacobian matrix which is defined as $J_e(\theta) = \frac{\partial f(\theta)}{\partial \theta}$ [6-13]. However, the derived IK equations are not necessarily available when the degeneracy and singularity problems ([6], [9], [12]- [14])] occurred and thus we need to use some new methods to solve it such as artificial neural network, genetic algorithm, neural fuzzy and etceteras ([7]- [15]).

Generally, high order polynomial is used to plan a trajectory with the intention to let the robot's hand to pass through all the specific points. However, solving the high order polynomial equations requires extensive calculations. Instead, it is advisable to use combinations of lower order polynomials for different segments of the trajectory and blend them together to satisfy all required boundary conditions ([1]- [5]).

$$ \theta\,(t) = c_0 + c_1t + c_2t^2 + ... + c_{n-1}t^{n-1} + c_nt^n \tag{4} $$

For an example, for a 4-3-4 trajectory, a fourth order polynomial is used to plan a trajectory between the initial point and the first via point, followed by a third order polynomial to plan a trajectory between two via points, lastly another fourth order polynomial is implemented to plan the trajectory between the last via point and final angle.

$$ \theta\,(t)_1 = a_0 + a_1t + a_2t^2 + a_3t^3 + a_4t^4 $$

$$ \theta\,(t)_2 = b_0 + b_1t + b_2t^2 + b_3t^3 $$

$$ \theta\,(t)_3 = c_0 + c_1t + c_2t^2 + c_3t^3 + c_4t^4 $$

In short, IK allows us to allocate the robot's hand at desired location and the motion of robot's hand can be determined by incorporating the FK and trajectory techniques. Sadly, the calculations will become tedious and hard to be solved when the number of specific points is increased.

3 Theoretical and Simulation Results

Experiments were performed on a simulated 4-degree of freedom (DOF) robot manipulator which shown in Figue 1, IK was used to determine the starting and ending configurations for the robot's hand while the high order polynomials and FK were used to track the motion of the robot's hand from beginning to the end in Cartesian space. It was specified that the robot need to pass through two specific points. At the same time, velocities and accelerations constraints need to be taken into consideration.

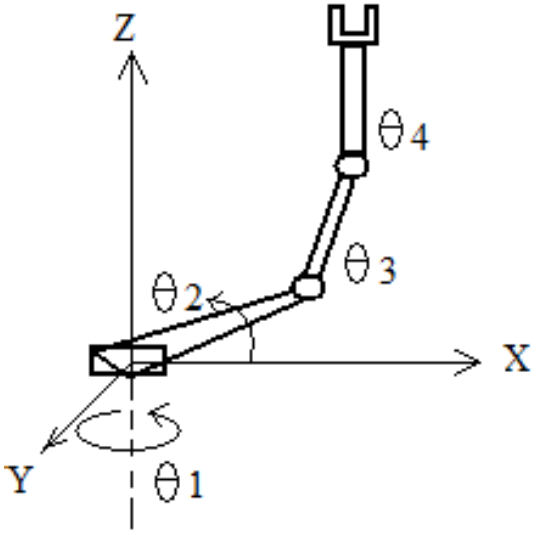


Figure 1: Simulated 4 DOF manipulator

Assuming that a manipulator needs to spend $t_f$to complete the moving path and a manipulator joint needs to reach $\theta_a$at interval time t_a then $\theta_b$at interval time t_b where $t_a < t_b$and $t_a, t_b \in [0, t_f]$. One may need to match the position, velocity, and accelerations of the two segments at each point to plan a continuous trajectory. Hence, higher order polynomials in the form of $\theta(t) = C_0 + C_1 t + C_2 t^2 + ... + C_{n-1} t^{n-1} + C_n t^n$ are required. However, it is an extensive calculation process to solve high order polynomial and it is advisable to use the combinations of lower order polynomials for different segment of the trajectory such as 4-3-4 trajectory or 3-5-3 trajectory.

The velocity and acceleration for initial and final point are zero. In order to fulfill the entire initial, final, and via points requirement, a matrix form can be generated where

$t_{ab} = t_b - t_a$and $t_c = t_f - t_a - t_b$.

$$\begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \ddot{\theta}_1 \\ \theta_2 \\ \theta_2 \\ 0 \\ 0 \\ \theta_3 \\ \theta_3 \\ 0 \\ 0 \\ \theta_4 \\ \dot{\theta}_4 \\ \ddot{\theta}_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & t_a & t_a^2 & t_a^3 & t_a^4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2t_a & 3t_a^2 & 4t_a^3 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 6t_a & 12t_a^2 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & t_{ab} & t_{ab}^2 & t_{ab}^3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2t_{ab} & 3t_{ab}^2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 6t_{ab} & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & t_c & t_c^2 & t_c^3 & t_c^4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2t_c & 3t_c^2 & 4t_c^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 6t_c & 12t_c^2 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \\ c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

Or

$$[\theta] = [M]\,[C] \tag{5}$$

By solving those constant values in matrix$[C]$, the joint turning angle at any interval time can be obtained where

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 \qquad 0 < t \le t_a$$

$$\theta(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 \qquad t \in [t_a, t_b]\ , t = t - t_a$$

$$\theta(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 \qquad t \in [t_b, t_f]\ , t = t - t_a - t_b$$

The equation which is used to represent the robot's hand at any interval time can be denoted as:

$$r_E(t_i) = \begin{pmatrix} \cos(\theta_1(t_i)) \left[l_1 \times \cos(\theta_2(t_i)) + l_2 \times \cos\left(\sum_{k=2}^{3} \theta_k(t_i) \right) + l_3 \times \cos\left(\sum_{k=2}^{4} \theta_k(t_i) \right) \right] \\ \sin(\theta_1(t_i)) \left[l_1 \times \cos(\theta_2(t_i)) + l_2 \times \cos\left(\sum_{k=2}^{3} \theta_k(t_i) \right) + l_3 \times \cos\left(\sum_{k=2}^{4} \theta_k(t_i) \right) \right] \\ l_1 \times \sin(\theta_2(t_i)) + l_2 \times \sin\left(\sum_{k=2}^{3} \theta_k(t_i) \right) + l_3 \times \sin\left(\sum_{k=2}^{4} \theta_k(t_i) \right) \end{pmatrix} \tag{6}$$

In fact, all these calculations can be solved simultaneously by back-propagation artificial neural network. The back-propagation neural network can be activated by presented a training set data with inputs $x_1(p), x_2(p) ... x_n(p)$and desired outputs$y_{d,1}(p), y_{d,2}(p), ..., y_{d,n}(p)$. Assume that the back-propagation network 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 $y_j(p) = sigmoid\left[\sum_{i=i}^{n} x_i(p) \times w_{ij}(p) - \theta_j\right]$ where n is the number of inputs for neuron j in the hidden layer, and sigmoid is the sigmoid activation function$Y^{sigmoid} = 1/(1+e^{-X})$. For the output layers, the actual outputs of the neurons will be $y_k(p) = sigmoid\left[\sum_{j=1}^{m} x_{jk}(p) \times w_{jk}(p) - \theta_k\right]$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 output layer and propagated backward for updating the weights.

For updating the weights at the output neurons with learning rate, α:

Error gradient,$\delta_k(p) = y_k(p) \times [1 - y_k(p)] \times e_k(p)$

$e_k(p) = y_{d,k}(p) - y_k(p)$

Weight correction, $\Delta w_{jk}(p) = \alpha \times y_j(p) \times \delta_k(p)$

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

Similar to output layer, for updating the weights at the hidden neurons with learning rate, α:

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

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.

A training data set which consists of inputs and outputs is presented to the networks for learning purpose. Inputs consists of the final angle of each joint $\theta_{f,n}$ and interval time variable t_i where $t_i \in [0, t_f]$ and $t_f$is the time for the manipulator to complete the moving path. The outputs will be the coordinates of robot's hand at any interval time with respect to t_i,$r_E(t_i) = [p_x\ p_y\ p_z]^T$.

There are two set of results in table 1. The theoretical results which obtained by using conventional FK and high order polynomial trajectory were compared to the artificial neural network (ANN) results. The robot started moving from its initial position ($\theta1$=$\theta2$=$\theta3$=$\theta4$=0) and passed through two via points before it reached its destinations. It shows that the robot still able to pass through those specific points even though the destination point was changed and the moving path could be tracked by using ANN. The theoretical and ANN results were almost the same and the errors were less than ±0.3unit (less than 1%). This can be used as an evidence to proof that artificial neural network is able to solve FK and continuous trajectory simultaneously due to its adaptive learning ability.

Since back-propagation neural network is able to track the motion of end effector, it can be used to track the position of each joint in Cartesian space as well. Referring to Figure 1, Joint 1 and joint 2 are located at the origin of reference frame, (0,0,0). For tracking the position of joint 3, the training data set inputs and outputs will be $[t_i\ \ \theta_1(t_i)\ \ \theta_2(t_i)]^T$and $r_{E,3}(t_i)$. Correspondingly, for joint 4, the training data set inputs and outputs will be $[t_i\ \ \theta_1(t_i)\ \ \theta_2(t_i)\ \ \theta_3(t_i)]^T$and $r_{E,4}(t_i)$. Once the joins location of all joints are computed, the manipulator link equations can be derived

by using parametric equations.

Assuming that the location and vector for joint m and joint n at any interval time t_i: $r_{E,n}(t_i) = [p_{x,n}\ p_{y,n}\ p_{z,n}]^T$, $\overrightarrow{OP_n} = [a_n\ b_n\ c_n]^T$ and $r_{E,m}(t_i) = [p_{x,m}\ p_{y,m}\ p_{z,m}]^T$, $\overrightarrow{OP_m} = [a_m\ b_m\ c_m]^T$.

The link equation that connecting joint n and m will be

$$\frac{x-p_{x,n}}{a_n-a_m} = \frac{y-p_{y,n}}{b_n-b_m} = \frac{z-p_{z,n}}{c_n-c_m} = \zeta$$

$$or$$

$$[x\ y\ z]^T = [p_{x,n}\ p_{y,n}\ p_{z,n}]^T + [a_n - a_m\ b_n - b_m\ c_n - c_m]^T\,\zeta$$

For the robot manipulator which is mentioned in this paper, the robot link equations can be denoted as:

Link 1: $\frac{x-p_{x,3}}{a_3-a_2} = \frac{y-p_{y,3}}{b_3-b_2} = \frac{z-p_{z,3}}{c_3-c_2} = \zeta_2 \qquad 0 \le z \le p_{z,3}$

Link 2: $\frac{x-p_{x,4}}{a_4-a_3} = \frac{y-p_{y,4}}{b_4-b_3} = \frac{z-p_{z,4}}{c_4-c_3} = \zeta_3 \quad p_{z,3} \le z \le p_{z,4}$

Link 3: $\frac{x-p_{x,5}}{a_5-a_4} = \frac{y-p_{y,5}}{b_5-b_4} = \frac{z-p_{z,5}}{c_5-c_4} = \zeta_4 \quad p_{z,4} \le z \le p_{z,5}$

Conclusively, the link equation can be derived as long as the coordinates of two successive joints are provided by using back-propagation neural network and the links parametric equations can be derived based on the vector of the two successive joints. Once the equation of each manipulator link is known, a person is able to predict or compute the entire motion of a robot manipulator.

Besides, obstacle collision can be detected if the link equations are known. Assuming that there is a rigid body obstacle enclosed in a sphere which possesses a r radius and center locates at (x_o, y_o, z_o), the sphere equation will be

$$(x - x_o)^2 + (y - y_o)^2 + (z - z_o)^2 = r^2$$

The intersection points between the link and obstacle can be obtained by solving the link and sphere surface equation. For an example, a line that pass through the point (e, f, g)possessing a same direction as vector$[a\ b\ c]^T$, the equation for the link will be:

$$\frac{x-e}{a} = \frac{y-f}{b} = \frac{z-g}{c} \quad = \quad \zeta \text{or} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{e} \\ \mathbf{f} \\ \mathbf{g} \end{pmatrix} + \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{pmatrix} \zeta$$

Substituting the link equation into sphere surface equation yields,

$$(a\zeta + e - x_o)^2 + (b\zeta + f - y_o)^2 + (c\zeta + g - z_o)^2 = r^2$$

simplified and yields,

$$A\zeta^2 + 2B\zeta + C = 0$$

where,

$$A = a^2 + b^2 + c^2$$

$$B = a(e - x_o) + b(f - y_o) + c(g - z_o)$$

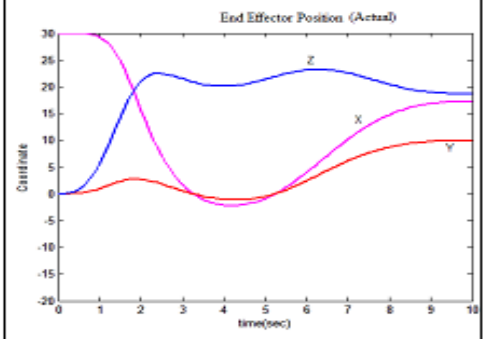
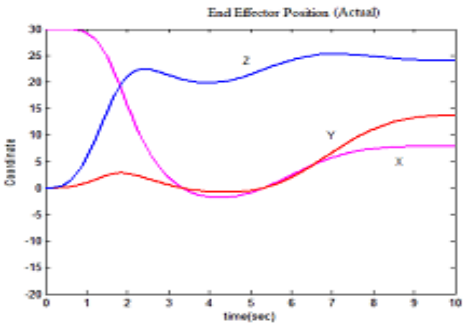
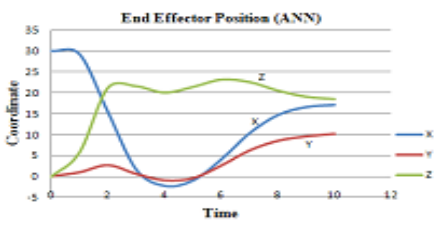
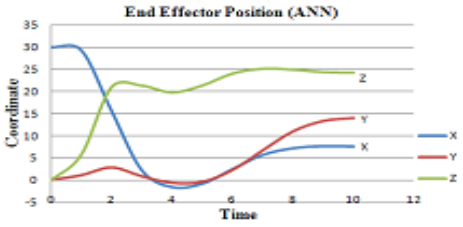
$$C = (e - x_o)^2 + (f - y_o)^2 + (g - z_o)^2 - r^2$$

By examining the discrimination roots, the intersection points between link and sphere surface exists when$\Delta \geq 0$,

$$[a(e-x_o)+b(f-y_o)+c(g-z_o)]^2 \geq 4\left[a^2+b^2+c^2\right]\left[(e-x_o)^2+(f-y_o)^2+(g-z_o)^2-r^2\right]$$

The intersection point between the manipulator links and obstacle can be determined by solving the constant ζ.

Table 1: Comparison of Conventional and ANN Techniques in Tracking Robot End Effector Moving Path

	Destination 1 Coordinate: (x=17.24; y=9.95; z=18.68) Joint values: (θ1=30; θ2=15; θ3=25; θ4=35)				Destination 2 Coordinate: (x=7.89; y=16.67; z=24.06) Joint values: (θ1=60; θ2=35; θ3=25; θ4=15)				
	Time	θ1	θ2	θ3	θ4	Time	θ1	θ2	θ3	θ4
Specific point 1 (x=15.58; y=2.75; z=21.08)	2	10	20	30	40	2	10	20	30	40
Specific point 2 (x=-0.87; y=-0.5; z=21.43)	5	30	40	50	60	5	30	40	50	60
Actual										
ANN										

4 Conclusions

This paper shows that back-propagation neural network is able to solve trajectory and kinematic problems simultaneously. The moving path of the robot end effector still able to be tracked even though the ending point is changed. Besides, this method can be used to track the position of robot joints which can be used to derive the robot link equations. In comparison to conventional continuous trajectory planning methods, ANN is much easier to be applied and provide high accuracy results.

Bibliography

[1] Saeed B. N. (2001). *Introduction to Robotics Analysis, System, Application.* Prentice Hall, 29-172.

[2] Ata, A., Myo, T. (2008). Optimal trajectory planning and obstacle avoidance for flexible manipulators using generalized pattern research. *World J. Modeling and Simulation*, 4:163–171.

[3] Guan, Y., Yokoi, K. , Stasse, O. , Kheddar, A. (2005). A. On robotic trajectory planning using polynomial interpolations. *Proc. IEEE Int. Conf. on Robotics and Biomimetics*, 111-116.

[4] Reichenbach, T, Kovacic, Z.(2005). Collision-Free Path Planning in Robot Cells Using Virtual 3D Collision Sensors. *Cutting Edge Robotics*, 683–697.

[5] Campos, J., Flores, J.A.R., Montufar, C.P.(2008). Robot Trajectory Planning for Multiple 3D Moving Objects Interception: A Polynomial Interpolation Approach. *Proc. IEEE Int. Conf. on Electronics, Robotics and Automotive Mechanics*, 478–483.

[6] Zarkandi S., Vafadar A., Esmaili M. R.(2011). PRRRRRP redundant planar parallel manipulator: Kinematics, workspace and singularity analysis. *IEEE 5th Int. Conf. on Robotics, Automation and Mechatronics (RAM)*, DOI: 10.1109/RAMECH.2011.6070457, 61–66.

[7] Al-Mashhadany, Y. I. (2010). Inverse Kinematics Problem (IKP) of 6-DOF Manipulator by LRNNs. *Int. Conf. on Management and Service Science (MASS)*, 1 – 5.

[8] Zhang, D., Lei, J.(2011). Kinematic analysis of a novel 3-DOF actuation redundant parallel manipulator using artificial intelligence approach. *Robotics and Computer-Intergrated Manufacturing*, 27: 157–163.

[9] Firmani, F., Podhoprodeski, R. P.(2009). Singularity analysis of planar parallel manipulatos based on forward kinematic solutions. *Mechanism and Machine Theory*, 44: 1386–1399.

[10] Vesselenyi, T., Dzitac, S., Dzitac, I., Manolescu, M.J. (2007). Fuzzy and Neural Controllers for a Pneumatic Actuator. *Int J Comput Commun*, ISSN 1841-9836, 2(4):375-387.

[11] Alvandar, S., Nigam, M. J. (2008). Neuro-Fuzzy based approach for inverse kinematics solution of industrial robot manipulators. *Int J Comput Commun*, ISSN 1841-9836, 3(3):224–234.

[12] Mahidzal, D., and Jian-Ding, T. (2011). Forward and Inverse Kinematics Model for Robotic Welding Process Using KR-16KS KUKA Robot. *Proc. IEEE Int. Conf. on Modeling, Simulation and Applied Optimization*, 1–6.

[13] Shah, J., Rattan S. S., Nakra, B.C.(2011). Kinematic Analysis of 2-DOF planer robot using artificial neural network. *World Academy of Science, Engineering and Technology*, 81: 282–285.

[14] Parikh, J.P., Lam, S.S. (2009). Solving the forward kinematics problem in parallel manipulators using an iterative artificial neural network strategy. *Int. J. Adv. Manuf. Technol.*, 40: 595–606.

[15] Her, M.-G., Chen, C.Y., Karkoub, M. (2002). Approximating a Robot Inverse Kinematics Solution Using Fuzzy Logic Tuned by Genetic Algorithms. *Int. J. Adv. Manuf. Technol.*, 20: 375–380.