ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 6, Issue 9, March 2017

Control of 4DoF Manipulator using Neural Network and Image Processing

[1]Tariq T. Darabseh*, [2]Nadeim Ussaleh
[1]Mechanical Engineering Department, United Arab Emirates University
[2]Mechanical Engineering D, Jordan University of Science and Technology

***Abstract*— Precise and fast control of any manipulator is the main challenge. In this paper, image processing is used to determine the 3D position of the objects and a predictive controller is used to estimate the best angle values to ensure that the manipulator will reach the target destination precisely, and to avoid any irrational move. This controller consists of a neural network which represents the plant and another neural network that replace the inverse kinematics (Controller). This system checks the controller output and enhances it before it reaches the real plant. This controller can be used for picking up constant objects like fruits, ICs, balls, and merchandise. This system is simple (no need for inverse kinematics), has fast response, and with accurate output (Mean Square Error of the whole system is around 0). To avoid over fitting and to enhance the generalization, around 0.5 million samples were collected using probability laws. Furthermore, this work can be used to control any manipulator, by some modification, so it can be a good assistance for the researchers.**

***Index Terms*—Image processing, inverse kinematics, manipulator, neural network.**

I. INTRODUCTION

The control of manipulators is currently a highly challenging research subject. The determination of the required angular orientation to let the manipulator achieves the new position correctly is not an easy task. The challenge in the manipulator control is to decrease the time response and the error.

The first method for determining a set of joint variables is the inverse kinematics solution. This method describes the manipulator motion by a series of matrices. Each matrix contains sixteen elements. Three of the elements represent the position in (P_x, P_y and P_z). Nine elements represent the orientation in (X, Y and Z). The last four elements are used to convert the matrix to a homogeneous one. This method is the basic method for a manipulator control. It gives a correct value of manipulator angles by using ordinary mathematic only. However, it has many problems; it is hard to derive the inverse equation and as the number of degrees of freedom of the manipulator increases the difficulty to derive the inverse equations increases. Therefore, solving such equations is very complex and the probability of errors is very high since each equation depends on the previous equation solution which accumulates errors.

The use of the inverse kinematics, in manipulator controlling, has been discussed in many papers as well as its uses. Manocha and Canny [1] presented two algorithms, implementations of efficient, and real time inverse kinematics for a general six revolute joints manipulator. Kung, Tseng, Chen, Sze and Wang [2] investigated the implementation of inverse kinematics and servo controller for robot manipulator using Field Programmer Gate Array (FPGA). Ali, Park and Lee [3] developed a consistent method for deriving a closed-form inverse kinematics joint solution of a general humanoid robot. Hang, Tung, Lin and Hsiao [4] used Denavit-Hartenberg (D-H) method for deriving the inverse kinematics, and they solved these equations by a geometric analysis. Wang et al. [5] controlled a six degree of freedom manipulator by deriving and solving the inverse kinematics by the closed form solution.

Traditional methods, such as algebraic are not suitable if the manipulator is very sophisticated [6]. For example; Wang et al. [7] supported a medical manipulator which can be adjusted by using the inverse kinematics method. Dahari and Tan [8] controlled welding process using the inverse kinematics, and they derived the forward and inverse kinematic equations by using D-H representation.

Modern scientists have devised new methods for controlling manipulator's motion. One of these methods is the Neural Network (NN), which is applied for modeling the inverse kinematics equations of a robot manipulator. NN is one of the most spread artificial intelligent methods, used for controlling manipulator position. This method is discussed in detail in section III. Thiang, Khoswanto and Pangaldus [9] described an application of NN for modeling the inverse kinematics equation of a robot manipulator. A large number of training data should be given in order to obtain better learning performance. NN can create the model of inverse kinematics automatically and of course, the model is not in a form of mathematic equation, but in the form of NN structure including weight and bias connection of the network [10]-[11]. Duguleana, Barbuceanu, Teirelbar and Mogan [12] analyzed the problem of Object Avoidance using Artificial Neural Network (ANN) to solve the inverse kinematics. Panwar, Kumar, Sukavanam and Borm [13] controlled multiple manipulator systems using NN, and they used this model to estimate the new value of position and orientation of each angle. Singh and Sukavanam [14] designed a neural-based network to decrease the outside disturbance of the robot manipulators. Daachi and Benallegue [15] estimated the NN parameters by using a closed-loop robot (Lyapunov approach). They increased the accuracy of a mobile robot in tracking object. Wei, Wang and Zuo [16] used a NN to

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 6, Issue 9, March 2017

decrease the external disturbance in the robot model.

In our work, we used the center point technique to estimate the 3D position using two cameras. The first one is fixed on the top of the roof to calculate the *X*, and *Y* position. The second camera is fixed in parallel with the manipulator and it moves with it as its eye. Then we derived the position model of the manipulator using a popular technique, which is D-H method. This model has been used to generate a NN which can replace the plant. The Plant NN has been used to test the efficiency of the NN controller before it reaches the real manipulator. This enhances the output, because there will be an online adapting of the output. Also, this will avoid any irrational output, so if the manipulator task is so danger, this will avoid catastrophic results.

The NN controller generated without the derivative of the inverse kinematics, which consumes too much time and effort. The NN controller generated by comprehensive and sufficient amount samples which almost cover most of the probability and avoid the singularity. These data were generated by using the D-H model. This model can be used for picking up fruit, balls, and ICs.

In Fig. 1 the manipulator task is to collect five balls in a net. The NN will determine the best angular values to achieve the task precisely.

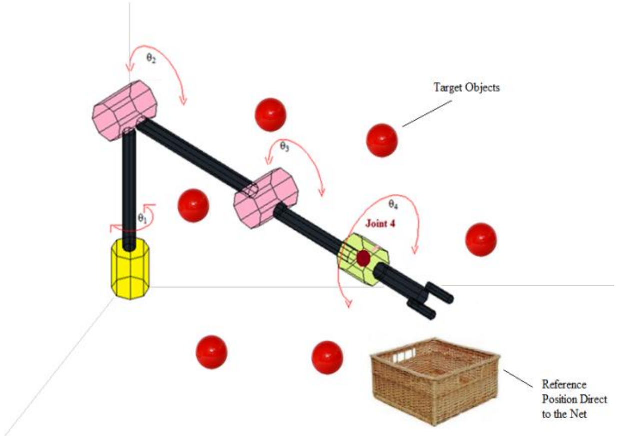

Fig. 1. 4-DoF Manipulator surrounded by Red balls.

II. IMAGE PROCESSING

Image processing is a technique, which focuses on processing image by analyzing it into pixels. Each pixel has a value, which depends on image type (black and white, gray or color). In this paper, the focus will be on color image processing. By using image processing; the manipulator can distinguish between many different objects. The task which will be discussed here is the object picking up or collecting in which the manipulator will pick up fixed objects like fruit and balls.

The method used here is color image processing, which depends on the value of RGB of each pixel in the image. Therefore, after taking the image, it will be analyzed to estimate the Red, Green, and Blue values of each pixel. All pixel values must be turned to zero except the pixels that represent the Red ball or the Red fruit. As we can see in Fig. 2 the pixels have different values, depends on which color it represents. We have also another three images represent the red and the green color of the image.

Then all pixel values must be turned to zero, except the one, which represents the target object (Red ball). This can be done by using a threshold value of RGB. The threshold value of the RGB estimated by view of the image using Matlab. Then place the mouse courser on any point in the image; the value of RGB will be displayed on the lowest left corner of the image. Consequently, the RGB of the ball can be achieved.

Then choose the minimum R value of the ball, and the maximum (G, B) value of the ball. By these values, we can convert the ball's pixels to 1, and the reset of image to zero. However, there will be some noise, but this noise can be removed by using different mask filters.

Then the position will be estimated by using the center point method. There are four center points: the first image's center point, the second image's center point, the red ball's center point from the first camera, and the center point of the red ball from the second camera.

For instance, Fig. 3 shows center of the ball and center of the camera, which calculates the *X*, and *Y* directions. By calculating the difference between the two centers and multiplying the values with real scale factor; the value of P_x and P_y will be estimated. The same thing applies for the camera, which calculates the *Z* direction.

31	32	32	33	35	36	33	32	34	37	37	32	43
35	36	37	36	31	27	11	14	21	29	30	0	0
9	15	25	25	12	0	7	3	0	0	0	16	16
0	4	12	15	9	2	0	0	0	0	0	17	13
9	1	0	0	0	0	0	0	0	9	15	20	19
0	0	0	0	0	2	35	25	17	19	25	0	0
36	25	11	5	9	16	0	0	1	0	0	10	5
0	0	0	0	0	0	0	0	0	0	0	11	5
13	30	42	50	56	59	68	34	1	0	18	40	37
46	35	17	0	0	0	0	0	0	0	0	33	29
9	0	0	0	0	17	18	14	20	39	51	0	0
0	0	0	16	16	9	17	16	22	33	36	19	26
18	26	29	29	27	26	26	23	22	17	5	0	0
60	59	51	45	46	51	58	51	46	42	32	0	5
0	6	36	55	51	38	29	21	23	35	45	29	17
0	0	0	0	0	0	0	0	0	1	14	0	0
6	0	0	0	0	0	1	0	0	3	3	0	10
0	0	5	7	3	0	0	0	0	0	0	42	28
2	10	21	26	23	17	5	0	0	0	10	0	0
19	12	3	0	0	0	15	0	0	0	0	0	0
32	15	0	0	0	0	9	0	0	0	0	24	21
0	0	0	0	0	0	0	5	13	19	22	6	0
0	0	0	4	8	8	5	15	26	30	26	15	7

Fig. 2. The blue value of the red ball and the background

III. ARTIFICIAL NEURAL NETWORK

In this section, the two NN implemented are summarized. These networks were tested using simulation programs.

A. Neural Network's Overview

NN is an intelligent system which can replace diverse plants and controllers around us. The main purpose of this program is to represent a system by sample data, and then the NN can build a mathematical function which can determine the accurate output value of a new input. The NN consists of several layers, neurons, functions, and number of optimization methods. Each system needs network, depends on the number of data, the type of data (static, periodic), and

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 6, Issue 9, March 2017

the application.

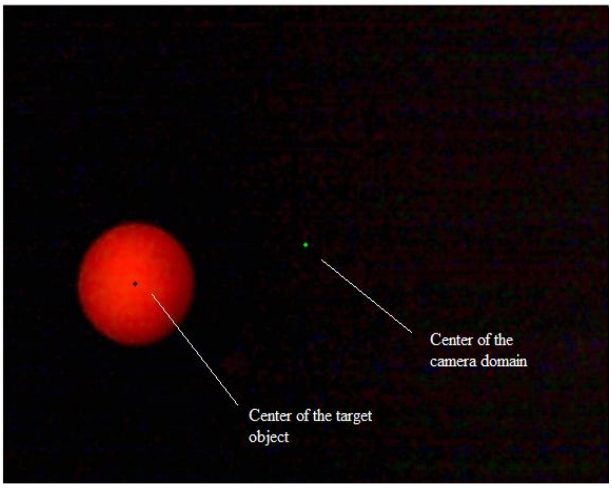

Fig. 3. The center of the ball and the camera domain.

Generally, back propagation algorithm is used in all the training function. It calculates the gradient and the Jacobian to enhance the performance of the system, by applying backward calculations. The performance is calculated using the MSE to check the difference between the target and the actual output. In Eqns. 1 and 2, the NN calculate the *MSE*:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 \quad (1)$$

$$e = (t - a) \quad (2)$$

where:
N: The total number of samples
i: The current sample number
e: Error
t: The target
a: The actual output

B. *Denavit- Hartenberg (D-H) method:*

In this application, two networks were designed. The sample data, which we need to train both networks, were collected using D-H method. In D-H method all angles rotate around Z axis. Each joint must move to the next joint, until the whole joints meet at the end effector. Any new rotation will be around the X axis, which is called α. Any move through the Z axis is called d, and any move along X axis is called a.

Our manipulator consists of four rotation angles, and four different links. By following D-H rules we estimated the following four matrices:

Eq. 3 represents the rotation about Z axis (θ_1) for the first joint, followed by rotation about X axis (α = 90°):

$$A_1 = \begin{bmatrix} C\theta_1 & 0 & S\theta_1 & 0 \\ S\theta_1 & 0 & -C\theta_1 & 0 \\ 0 & 1 & 0 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Eq. 4 represents the rotation about Z axis (θ_2) for the second joint, followed by translation along X axis (a_1):

$$A_2 = \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 & a_1 \times C\theta_2 \\ S\theta_2 & C\theta_2 & 0 & a_1 \times S\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

Eq. 5 represents the rotation about Z axis (θ_3), followed by translation along X axis (a_2), and followed by rotation about X axis (α = 90°):

$$A_3 = \begin{bmatrix} C\theta_3 & 0 & S\theta_3 & a_2 \times C\theta_3 \\ S\theta_2 & 0 & -C\theta_3 & a_2 \times S\theta_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

Eq. 6 represents the rotation about Z axis only (θ_4):

$$A_4 = \begin{bmatrix} C\theta_4 & -S\theta_4 & 0 & 0 \\ S\theta_4 & C\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

Eq. 7 represents the final value of the new position and the orientation, after a series of rotations:

$$^4A_0 = \begin{bmatrix} n_X & o_X & a_X & P_x \\ n_Y & o_Y & a_Y & P_y \\ n_Z & o_Z & a_Z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

Where:
C: Cosine
S: Sine
P_x: the position in the X direction
P_y: the position in the Y direction
P_z: the position in the Z direction
n_X represents $i_X.i_u$, o_X represents $i_X.j_v$, a_X represent $i_X.k_w$
n_Y represents $j_Y.i_u$, o_Y represents $j_Y.j_v$, a_Y represent $j_Y.k_w$
n_Z represents $k_Z.i_u$, o_Z represents $k_Z.j_v$, a_Z represent $k_Z.k_w$

Figure 4, shows four degree of freedom manipulator. All joints are rotated about Z axis; θ_1 (0° to 360°), θ_2 (-90° to 90°), θ_3 (0° to 125°), and θ_4 (0° to 360°). Table I illustrates the direction and the value of the basic movement of D-H method. Using these values, we can model the forward move of the manipulator, to estimate the training value of the NN.

C. *Predictable NN*

As mentioned earlier, two networks were used. The first one represents the plant, and the other one represents the controller. This done for two reasons, to predict the best values, before it reaches the real controller. This will save sequences of irrational values, and enhance the results. The second reason is to train the controller offline without the need of the real plant.

C.1. Plant NN

The Plant NN has four inputs (θ_1, θ_2, θ_3, and θ_4), and three outputs (P_x, P_y, and P_z). A table of four inputs and three outputs was generated by using probability rules. The data is

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 6, Issue 9, March 2017

gradual and comprehensive. The number of samples was about 0.5 million samples including inputs and outputs.

Table I. Denavit-Hartenberg parameters of the 4DoF manipulator

Joint	θ	α (°)	a (mm)	d (mm)
1	θ_1	90	0	300
2	θ_2	0	300	0
3	θ_3	90	250	0
4	θ_4	0	0	0

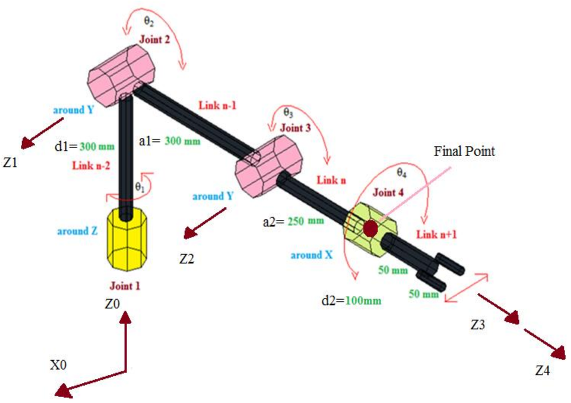

Fig. 4. The four degree of freedom manipulator

Figs. 5 and 6 illustrate the Plant Model. This model is generated using the real plant's values. The training signal is the difference between the real plant output and the NN plant output.

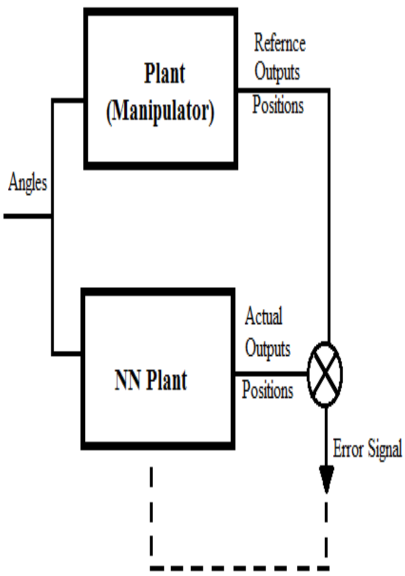

Fig. 5. NN Plant model

Diverse type of functions, optimization algorithm, training method, number of hidden layers, and number of neurons have been used to generate the best performance. Three hidden layers were used, with 20 neurons in the first, 50

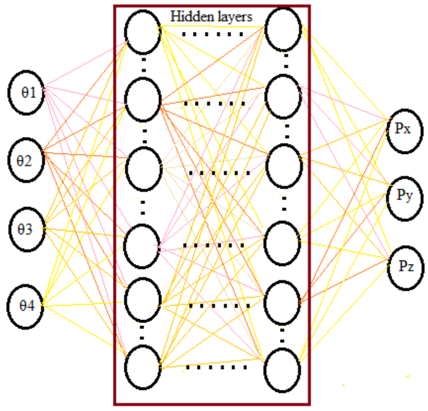

Fig. 6. NN Plant

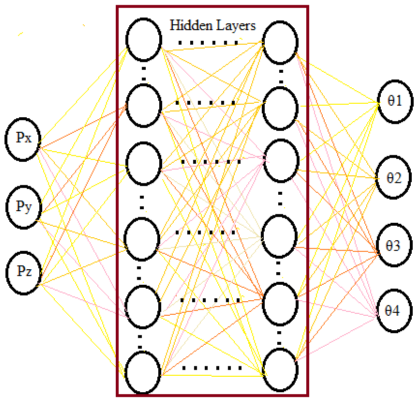

Fig. 7. NN controller

Neurons in the second and 20 neurons in the last. These values were estimated after many experiments. The MSE was 0.0003.

C.2. NN Controller

The NN controller generated without the derivative of the inverse kinematics, which consumes too much time and effort. The NN controller generated by comprehensive and sufficient amount samples which almost cover most of the probability and avoid the singularity.

The Plant NN has three inputs (P_x, P_y, and P_z), and four outputs (θ_1, θ_2, θ_3, and θ_4). This controller calculates the appropriate angle values to let the manipulator reaches the new position correctly.

Figs. 7 and 8 illustrate the Controller Model. This model is generated using the NN plant's values. The training signal is the difference between the NN plant output and the NN plant output.

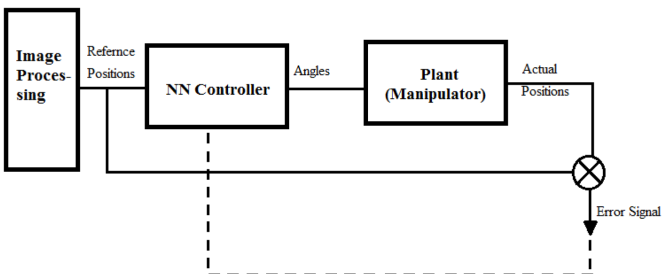

Fig. 8. NN controller model

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 6, Issue 9, March 2017

C.3. Using the NN

Now the two NN are ready to be used. The image processing determines the position values of the target object in 3D. These positions are directly being sent to the NN controller. The controller estimates the best angle values to reach the new position. These values will be sending to the NN Plant, which calculates the actual position. The difference between the target position with the actual output will generate an Error. This error reenters the NN controller as a new input, and then it will be added to the previous output of the NN controller. Finally, the sum of the outputs will be sent to the Plant (Manipulator). It ensures that the manipulator will get the best angle values, to reach the new position correctly. If the error is large, the manipulator won't move.

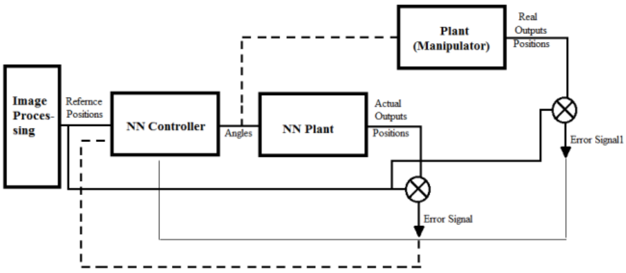

Fig. 9. The Predictive NN

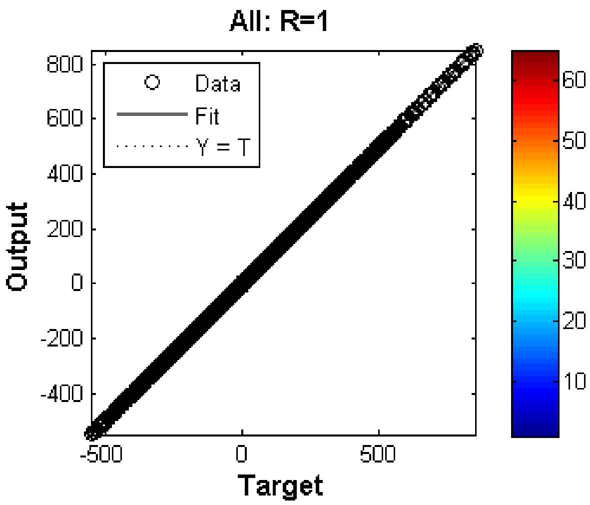

Fig. 10. Regression line

Fig. 9 illustrates how the Predictive NN works. The image processing controller estimates the Reference Position. Then it goes to the NN controller which determines the (Angles). These values are tested using the NN plant, which determines the (Actual position). The difference between the actual and reference position generates an error signal to decrease the error. Finally, the angles will be sent to the real plant which moves to the real position. The difference between the real and reference position will generate an error, which adapts the controller.

IV. SIMULATION AND RESULTS

After the neural networks, had been generated, they were tested with random values, which weren't among the training

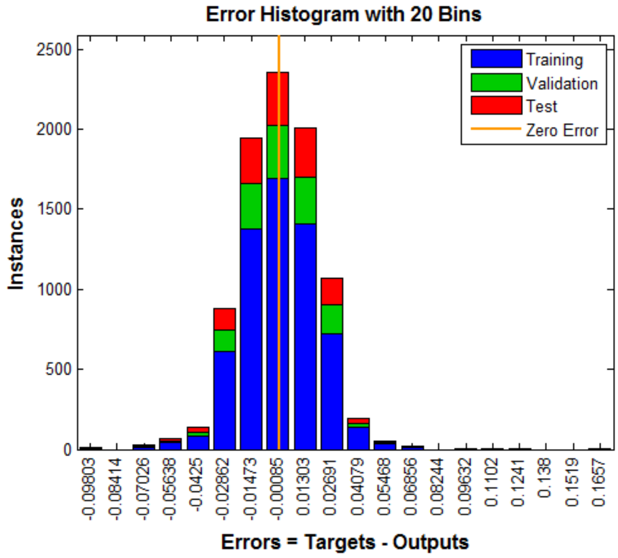

Fig. 11. Error Histogram

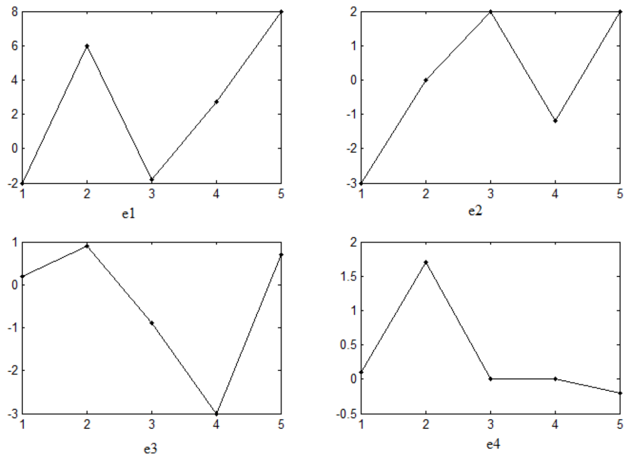

Fig. 12. NN controller error

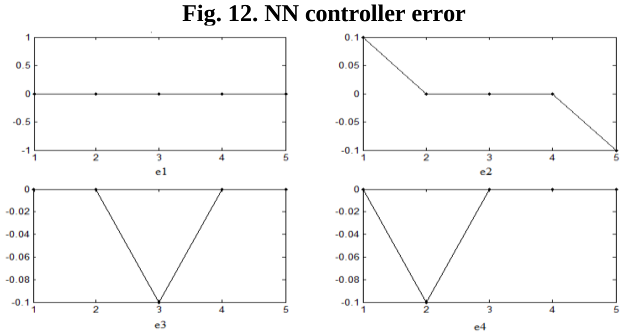

Fig. 13. Predictive NN error

Data. Each NN was tested separately. Then the whole system was tested together. Table II shows the response of NN Plant. Table III Shows the response of NN controller. Table IV presents the response of the whole system (Predictive NN). Fig. 10 illustrates the relation between the output and the target which is equal to 1. This means that the regression is perfect. Fig. 11 shows the distribution of error among variable instances. In Fig. 12 the NN controller's error is shown. Fig. 13 shows the error after applying the full predictive controller.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 6, Issue 9, March 2017

Table II. Random value to compare Real Plant output with NN Plant output

Angles				Real Plant			NN Plant			Error		
θ_1	θ_2	θ_3	θ_4	P_x	P_y	P_z	P_x	P_y	P_z	e_1	e_2	e_3
50	84	31	20	-47.4	-56.5	825.0	-47.4	-56.5	825.0	0.0	0.0	0.0
90	60	15	100	0.17	215.0	801.1	0.06	215.0	801.1	0.1	0.0	0.0
0	-40	90	30	390.6	0.0	298.6	390.7	0.0	298.6	-0.1	0.0	0.0
75	30	100	0	25.7	95.9	641.6	25.8	95.9	641.6	-0.1	0.0	0.0
250	80	30	40	11.3	30.9	830.4	11.3	30.9	830.4	0.0	0.0	0.0

Table III. Random value to compare Reference inputs with NN Controller output

Reference Inputs			Real Angles				NN Controller				Error			
P_x	P_y	P_z	θ_1	θ_2	θ_3	θ_4	θ_1	θ_2	θ_3	θ_4	e_1	e_2	e_3	e_4
-47.4	-56.5	825.0	50	84	31	20	52	87	30.8	19.9	-2.0	-3.0	0.2	0.1
0.17	215.0	801.1	90	60	15	100	84	60	14.1	98.3	6.0	0.0	0.9	1.7
390.6	0.0	298.6	0	-40	90	30	1.8	-42	90.9	30	-1.8	2.0	-0.9	0.0
25.7	95.9	641.6	75	30	100	0	72.3	31.2	103	0	2.7	-1.2	-3.0	0.0
11.3	30.9	830.4	250	80	30	40	242	78	29.3	40.2	8.0	2.0	0.7	-0.2

Table IV. Value to compare Reference inputs with the Predictive NN

Reference Inputs			Real Angles				NN Controller				Error			
P_x	P_y	P_z	θ_1	θ_2	θ_3	θ_4	θ_1	θ_2	θ_3	θ_4	e_1	e_2	e_3	e_4
-47.4	-56.5	825.0	50	84	31	20	50	84.1	31	20	0.0	0.1	0.0	0.0
0.17	215.0	801.1	90	60	15	100	90	60	15	100.1	0.0	0.0	0.0	-0.1
390.6	0.0	298.6	0	-40	90	30	0	-40	90.1	30	0.0	0.0	-0.1	0.0
25.7	95.9	641.6	75	30	100	0	75	30	100	0	0.0	0.0	0.0	0.0
11.3	30.9	830.4	250	80	30	40	250	80.1	30	40	0.0	-0.1	0.0	0.0

V. CONCLUSION

In this study, image processing is used to determine the 3D position of the objects and a predictive controller is used to estimate the best angle values to ensure that the manipulator will reach the target destination precisely, and to avoid any irrational move. This controller consists of a neural network which represents the plant and another neural network that replace the inverse kinematics (Controller). The NN controller generated without the derivative of the inverse kinematics, which consumes too much time and effort. The NN controller generated by comprehensive and sufficient amount samples which almost cover most of the probability and avoid the singularity.

Predictive NN is new concept of controlling, since you won't only control the current task, but you control future tasks. The image processing controller estimates the Reference Position. Then it goes to the NN controller which determines the (Angles). These values are tested using the NN plant, which determines the (Actual position). The difference between the actual and reference position generates an error signal to decrease the error. Finally, the angles will be sent to the real plant which moves to the real position. The difference between the real and reference position will generate an error, which adapts the controller.

For future study, a dynamic neural network will be used to catch moving objects, and to track difficult trajectory.

REFERENCES

[1] D. Manocha and J. F. Canny, "Real time inverse kinematics for general 6R manipulators," IEEE International Conference on Robotics and Automation, pp. 383-389, May 1992.

[2] Y. Kung, K. Tseng, C. Chen, H. Sze and J. Wang, "FPGA-Implementation of inverse kinematics and servo controller for robot manipulator," IEEE International Conference on Robotics and Biomimetics, pp. 1163-1168, December 2006.

[3] A. Ali, A. Park and G. Lee, "Closed-form inverse kinematic joint solution for humanoid robots," IEEE International Conference on Inelegant Robot and System, pp. 704-709, October 2010

[4] G. Hang, C. Tung, H. Lin and S. Hsiao, "Inverse kinematics analysis trajectory planning for a robot arm," IEEE International Conference on Control, pp. 965-970, May 2011.

[5] L. Wang, P. Fallavollita, R. Zou, X. Chen, S. Weidert and N. Navab, "Closed-form inverse kinematics for interventional C-arm X-ray imaging with six degrees of freedom: modeling and application," IEEE Trans. on Medical Imaging, Vol. 31, no. 5, pp. 1086-1099, May 2012.

[6] M. Aghajarian and K. Kiani, "Inverse kinematics solution of PUMA 560 robot arm using ANFIS," IEEE 8th International Conference on Ubiquitous Robots and Ambient Intelligence, pp. 574-578, November 2011.

[7] X. Wang, X. Duan, Q. Huang, H. Zhao, Y. Chen and H. Yu, "Kinematics and trajectory planning of a supporting medical manipulator for vascular interventional surgery," IEEE/ICME International Conference on Complex Medical Engineering, pp. 406-411, May 2011.

[8] M. Dahari and J. D. Tan, "Forward and inverse kinematics model for robotic welding process using KR-16KS KUKA robot," 4th International Conference on Modeling, Simulation and Applied Optimization, pp. 1-6, April 2011.

[9] T. Thiang, H. Khoswanto and R. Pangaldus, "Artificial neural network with steepest descent backpropagation training algorithm for modeling inverse kinematics of manipulator," World Academy of Science, Engineering and Technology, International Science Index 36, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering, Vol. 3, no. 12, pp. 2342-2345, December 2009.

[10] M. Al-Faiz, "Inverse kinematics solution for robot manipulator based on neural network," MASAUM Journal of Basic and Applied Sciences, Vol. 1, no. 2, pp. 147-154, September 2009.

[11] H. Patino, R. Carelli and B. Kuchen, "Neural networks for advanced control of robot manipulators," IEEE Trans. on Neural Networks, Vol. 13. no. 2, pp. 343-354, March 2002.

[12] M. Duguleana, F. G. Barbuceanu, A. Teirelbar and G. Mogan, "Obstacle avoidance of redundant manipulators using neural networks based reinforcement learning," Robotics and Computer-Integrated Manufacturing, Vol. 28, no. 2, pp. 132-146, April 2012.

[13] V. Panwar, N. Kumar, N. Sukavanam and J. H. Borm, "Adaptive neural controller for cooperative multiple robot manipulator system manipulating a single rigid object," Applied Soft Computing, Vol. 12, no. 1, pp. 216-227, January 2012.

[14] H. P. Singh and N. Sukavanam, "Neural network based control scheme for redundant robot manipulators subject to multiple self-motion criteria," Mathematical and Computer Modeling, Vol. 55, no. 3-4, pp. 1275-1300, February 2012.

[15] B. Daachi and A. Benallegue, "Stable neural network controller based observer for rigid robot manipulators," Journal of Robotics and Mechatronics, Vol. 15, no. 1, pp. 77-83, February 2002.

[16] S. Wei, Y. Wang and Y. Zuo, "Wavelet neural networks robust control of farm transmission line deicing robot manipulators," Computer Standards & Interfaces, Vol. 34, no. 3, pp. 327-333, March 2012.

AUTHOR BIOGRAPHY

Tariq T. Darabseh is currently an Associate Professor at Mechanical Engineering Department, United Arab Emirates University, UAEU. He obtained his Bachelor Degree and M.Sc. from Jordan University of Science and Technology in 1992, and in 1994, respectively. He received his Ph.D. in Mechanical Engineering from New Mexico State University in 2002, in USA. His current research interests are Robotics, Dynamic Stability, and Linear and Nonlinear Dynamics.

https://scholar.google.ae/citations?user=8fd0HhQAAAAJ&hl=en

Nadeim Ussaleh Received his M.Sc. in Mechanical Engineering from Jordan University of Science and Technology, Jordan, in 2012.