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 \((P_x, P_y, \text{and} P_z)\). Nine elements represent the orientation in \((X, Y, \text{and} Z)\). 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.


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
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.

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.
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:

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

$$e = (t - a)$$

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 $a$. 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 ($\theta_1$) for the first joint, followed by rotation about X axis ($\alpha = 90^\circ$):

$$A_1 = \begin{bmatrix} C\theta_1 & 0 & S\theta_1 & 0 \\ S\theta_1 & C\theta_1 & 0 & 0 \\ 0 & 1 & 0 & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Eq. 4 represents the rotation about Z axis ($\theta_2$) for the second joint, followed by translation along X axis ($a_2$):

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

Eq. 5 represents the rotation about Z axis ($\theta_3$), followed by translation along X axis ($a_2$), and followed by rotation about X axis ($\alpha = 90^\circ$):

$$A_3 = \begin{bmatrix} C\theta_3 & 0 & S\theta_3 & a_2 \times C\theta_3 \\ S\theta_3 & C\theta_3 & 0 & a_2 \times S\theta_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Eq. 6 represents the rotation about Z axis only ($\theta_4$):

$$A_4 = \begin{bmatrix} C\theta_4 & 0 & S\theta_4 & 0 \\ S\theta_4 & C\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

$$4A_0 = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

Figure 4, shows four degree of freedom manipulator. All joints are rotated about Z axis; $\theta_1$ ($0^\circ$ to $360^\circ$), $\theta_2$ (-90$^\circ$ to 90$^\circ$), $\theta_3$ ($0^\circ$ to 125$^\circ$), and $\theta_4$ ($0^\circ$ to $360^\circ$). 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 ($\theta_1$, $\theta_2$, $\theta_3$, and $\theta_4$), 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
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

<table>
<thead>
<tr>
<th>Joint</th>
<th>$\theta$</th>
<th>$\alpha$ (°)</th>
<th>$a$ (mm)</th>
<th>$d$ (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\theta_1$</td>
<td>90</td>
<td>0</td>
<td>300</td>
</tr>
<tr>
<td>2</td>
<td>$\theta_2$</td>
<td>0</td>
<td>300</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>$\theta_3$</td>
<td>90</td>
<td>250</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>$\theta_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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.

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 ($\theta_1$, $\theta_2$, $\theta_3$, and $\theta_4$). This controller calculates the appropriate angle values to let the manipulator reaches the new position correctly.

Fig. 5 and 6 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.

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

All: \( R=1 \)

![Fig. 10. Regression line](image)

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 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.


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


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.