JKAU: Eng. Sci., **Vol. 19** No. 1, pp: 19-42 (2008 A.D. / 1429 A.H.)

Neural Networks for Multi-Finger Robot Hand Control

E.A. Al-Gallaf

Department of Electrical and Electronics Engineering,
College of Engineering, University of Bahrain
P.O. Box 13184, Kingdom of Bahrain
ebrgallaf@eng.uob.bh

Abstract. This paper investigates the employment of Artificial Neural Networks (ANN) for a multi-finger robot hand manipulation in which the object motion is defined in task-space with respect to six Cartesian based coordinates. The approach followed here is to let an ANN learn the nonlinear functional relating the entire hand joints positions and displacements to object displacement. This is done by considering the inverse hand Jacobian, in addition to the interaction between hand fingers and the object being grasped and manipulated. The developed network has been trained for several object training patterns and postures within a Cartesian based palm dimension. The paper demonstrates the proposed algorithm for a four fingered robot hand, where inverse hand Jacobian plays an important role in robot hand dynamic control.

Keywords: Robot Hand Manipulation, Artificial Neural Network, Inverse Hand Jacobian.

Nomenclature:

$\dot{u}$	absolute velocity of the grasped object $\in \Re^{6\times 1}(v_o \omega_o)^T$
M_h, N_h, C_h	hand inertia, centrifugal and gravity forces
Θ_h	hand joint space vector $\in \Re^{12\times 1}$
κ_h	concatenated hand Jacobian $\in \Re^{12\times 12}$
W_i	i^{th} finger Jacobian rotation as allied with object surface
θ_{ij}	i^{th} joint displacement of j^{th} finger

ϕ_{tip}	hand fingertip force vector $\in \Re^{12\times 1}$
M_i, N_i, C_i	i^{th} finger inertia, centrifugal and gravity forces
Θ_i	i^{th} finger joint space vector $\in \Re^{3\times 1}$
ϕ_{par}	particular solution of the fingertip force vector by solving $\phi_{tip} = (\phi_{par} + \lambda\eta)$
Z_i	control gain of grasp internal forces
ϕ_{cd}	commanded internal forces
w_{ij}	interconnection weight from node i^{th} *to* j^{th} node
$I_{n(xyz)}$	inertial system frame
$P^i{}_{(ijk)}$	frame coordinate at the i^{th} finger point contact
ϕ_{tipi}	i^{th} fingertip force vector $\phi_{tipi} = (\phi_x \quad \phi_y \quad \phi_z)^T$
F_b	object resultant force vector $\in \Re^{6\times 1}$ expressed with respect to the $I_{n(xyz)}$ coordinate
η	null space of G
G	hand grip transform
λ	adjusting vector of the null space solution
$\varphi 1_f, \varphi 2_f, \varphi 3_f, \varphi 4_f$	employed Neural Networks activation functions
$L_{1i}, \ldots\ldots L_{4i}$	lengths of the i^{th} finger links
X_i	hand constrained differential motion
E_{nxn}	an identity matrix of an appropriate dimension
W_i	i^{th} finger Jacobian rotation allied with the object surface
θ_{ij}	i^{th} joint displacement of j^{th} finger
ϕ_{tip}	hand fingertip forces $\phi^T_{tip} = (\phi_{tip1} \quad \phi_{tip2} \quad \phi_{tip3} \quad \phi_{tip4})$ expressed in body coordinate
$\Delta\Theta_k$	change in hand joint space vector at a k^{th} interval.
κ_h^{-1}	hand Jacobian inverse $\in \Re^{12\times 12}$
τ_h	hand joint torques
$\sigma_{i1}, \sigma_{i2}, \sigma_{i3}$ i^{th}	finger Jacobian singular values
E	neural network objective function for optimization
Δu_a^c	change in actual object Cartesian posture.

1. Introduction

1.1 Multi-Fingered Robot Hand and the Dynamics of Manipulation

Artificial Intelligence (AI) and heuristic techniques have been introduced by many researchers in the area of robot control and motion planning[1-3]. The main concern was to employ a mechanism for hand motion and dexterous finger maneuver which are totally complicated aspects in robot hands. Artificial Neural Network (ANN) are nonlinear, global approximation methods. Neural networks have been heavily employed in robotics technology such as robot arm visual control as introduced by Hashimoto, Kubota, Sato, and Harashima[4], Hashimoto, Kubota, Kuduo, and Harashima[5], inverse kinematics problem of six degree of freedom robot arm as done by Guez and Ahmed[6], the research introduced by Patrick and Krose[7] in which they employ a real-time learning neural robot controller for solving the inverse kinematics problem, and the research introduced by Schram, Linden, Krose, and Groen[8], in which they employ an artificial neural network for tracking and grasping a moving object observed by a six degree of freedom robotics arm system.

Recently a substantial amount of research has been carried out in the subject of dexterous manipulation and hand maneuvers, some including the dynamics of the hand[9]. Although a four fingered robot hand offers a number of positive issues regarding an object manipulation in an automated environment, finding a control strategy for forces application on an object even at singular task configuration is not an easy task. Furthermore, the degree of freedom of a four fingered robot hand represents a force redundancy with respect to the manipulated object. Hence the use of an optimization technique is one approach to solve the excess fingertip forces applied to an object.

The problem of Cartesian object manipulation based on the employment of hand Jacobian plays an essential role in the area of modern robot control. Model-based multi-finger robot hand control has been used extensively for accurate fingers positioning. However, the employment of hand dynamic model in Cartesian based control does require the employment also of the inverse hand Jacobian which is indeed a function of the entire hand positioning mechanism. There are a large number of proposed algorithms that take into account the problem

of computing the hand Jacobian, however, most of these approaches are time consuming and their applicability for real time control is not feasible.

Application of damped least-squares solutions to robot control can be easily formulated using Singular Value Decomposition (SVD) Theorem, and has been proposed as one of the most efficient ways to suppress high velocities as found in Ref. [10-12]. SVD has also been used in the specification of redundant robot dexterity measures (*the minimal singular value*)[13] and in describing and reshaping the manipulability ellipsoid for redundant robots[14-15]. Meanwhile, a broader application of SVD in real-time robot control has been prevented by its massive computational complexity when compared with that of the GAUSSIAN elimination (*approximately* $12\times n^3$ *with respect to* $2n^3/3$ *floating point operations for an* $n\times n$ *matrix*).

A method has been proposed in Maciejewski and Klein[12] to reduce the computational burden by applying the method based on GIVENS rotations, through which five to six time reduction in computational time is obtained. Two articles that have presented algorithms for the computation of multiple-robot dynamics are those by Lilly and Orin[16] and Rodriguez, Jain, and Kreutz-Delgado[17]. Both developed sequential algorithms that have a computational complexity of O(mN) where m is the number of manipulator chains in the system and N is the number of degrees of freedom per chain. Previous research contributions in task space measure can be found in Klein and Blaho[13], and Dubey and Luh[14], Lee[15], where static and dynamic manipulability ellipsoids have been introduced.

1.2 Artificial Neural Networks and Robot Hand Control

While artificial neural networks have been employed extensively in robotics and automation system, for robot arms, in particular, ANN have been used for approximating the mapping between object posture and the corresponding joint displacement. However, there has been a limited number of research articles concerning the employment of ANN in dexterous robot hand control and analysis. Example of such research papers has been the one discussed by Huan, Iberall, and Bekey[3], where they presented an architecture for multi-finger robot hand control via neural networks.

In addition to this, another employment of artificial neural networks has been the one presented by Wohlke[18], where the conception of the control system was based on the combination of a neural network approach for the adaptation of grasp parameters and a fuzzy logic approach for the correction of parameters values given to a conventional controller. Zsiros, Baranyi, and Korondi[19] did present a practical application of generalized neural networks for a dexterous hand moved by shape memory alloys, where the robot hand was controlled by a generalized neural network.

Furthermore, Li-Ren and Taipei[20] have discussed the use of digital signal processor (DSP) for fuzzy control of robot hands. In their approach, they presented fuzzy control of a multi-fingered robot hand using a (DSP), where the DSP has been employed to implement a fuzzy control methodology of the seventeen joints which are controlled simultaneously. On the other hand, Doersam, Ftikow, and Streit[21] have presented a fuzzy logic approach for on-line grasp-force-adaptation, which can be used for the control of fine manipulating with a robot hand, where a decision making logic that expresses a priori knowledge about the force behavior inside the friction cones has been used.

Caihua and Youlun[22] have discussed the employment of intelligent learning based techniques for a multifingered grasp force planning which has been based on Neural-Network. A technique through which an evolution strategy (*genetic optimization*) was employed for learning dexterous hand manipulation strategies has been presented by Fuentes and Nelson[23], through which they suggested a genetic-based technique for optimizing a robot hand grasp configurations to meet defined manipulation requirements.

Fischer, Rapela, and Woern[24] has studied controlling an object's pose and the forces between the object and its environment. They, advised an object-pose controller with feedback from an object pose sensor suits for multi-finger gripper control. Due to the non-linear dynamic system behavior in the joints, an effective, easy-adaptable joint controller was employed and was based on fuzzy and neural-network algorithms, where an exact analytical model for such a case was not utilized.

1.3 *Article Contributed Theme*

The research structure presented here is novel in the sense that, Jacobian hand inverse has been avoided to compute as compared to other techniques which used other numerical algorithms to compute the inverse via the Singularity Robust Inverse presented earlier by Maciejewski and Klein[12]. By mapping Jacobian hand inverse to a set of neural interconnection weights, this facilitates to reduce the computational execution time, in addition to the ability to add more training patterns that are useful specially once the hand passes through singularity. Finding an inverse to hand Jacobian is essential in multi-fingered robot hand systems, specifically once mapping some visual object displacement information to a set of hand joints displacement. Other presented techniques using hand Jacobian inverse have employed numerically extensive approaches which are not suitable for real-time control, whereas using this technique, mapping object motion to hand joints displacement was reduced to a set of neural computed nodes.

1.4 Article Structure

By analyzing and learning the relationship between object displacement within the hand palm and robot hand joints, manipulation aspects (joint displacement), this architecture provides a level of abstraction for generic use. Hence, the article has been divided into seven main sections. In Section (1) a review related to neural robot hand control is presented. Section (2) presents the hand-object dynamic equation formulation, whereas Section (3) presents the object to joint space mapping. Neural Network and hand differential motion are presented in Section (4) and in Section (5) the researcher presents the used back-propagation training algorithm. The results of hand simulation are presented in Section (6). Finally, Section (7) draws few conclusions.

2. Hand-Object System Dynamic Equation Formulation

2.1 Constrained Kinematics

Working under the contact kinematics assumption modeled by a frictional point of contact, with each finger we associate three forces

$(\phi_x \quad \phi_y \quad \phi_z)$ defined in terms of a normal force ϕ_z and a resultant force ϕ_r which are decomposed into two co-planar forces as expressed by :

$$\phi_{tipi}^T = (\phi_x \quad \phi_y \quad \phi_z) \tag{1}$$

where $-\mu\phi_z \le \sqrt{(\phi_x^2 + \phi_y^2)}$ and μ is the friction constant at point of contact. From Fig. 1-a, each finger maps its joints torque to the object via the entire hand grasp transform $G \in \Re^{6\times12}$ formulated by Equ. (2) :

$$G = [G_1 \quad G_2 \quad G_3 \quad G_4] \tag{2}$$

where grasp sub-matrices $G_i \in \Re^{6\times3}$ for $i = 1, ..., 4$ are defined in terms of contact location as :

$$G_i = \begin{bmatrix} I_{3\times3} \\ Y_i \end{bmatrix} \quad for \quad i = 1, ..., 4$$

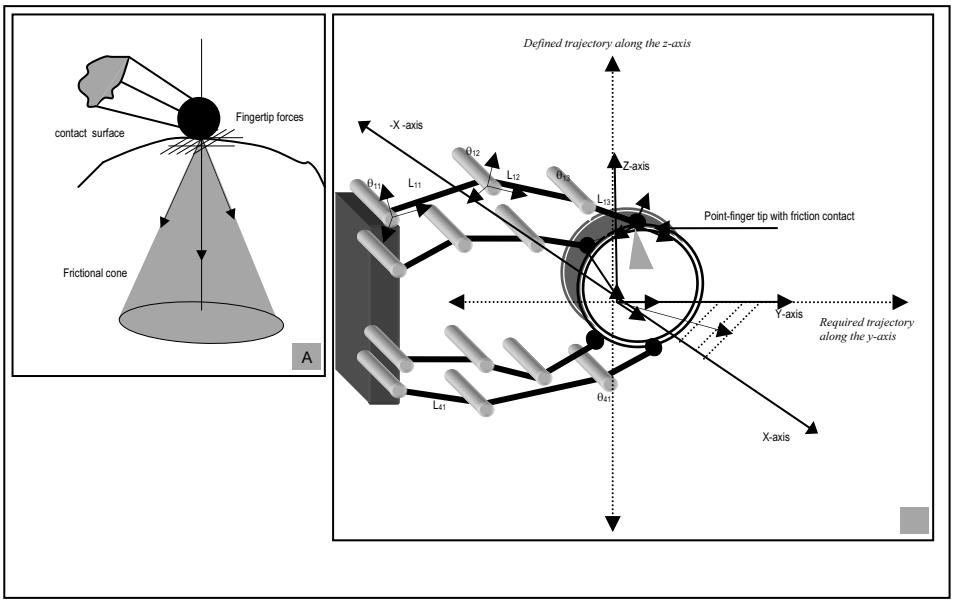

Fig. 1. Hand-object frames
a : Forces distribution.
b : Gasping coordinates.

Where Y_i are matrices performing the skew-matrix of position contact r_i on the object surface.

Letting the object motion velocity in the 3-dimensional space be designated as $\dot{u}$ in terms of linear v_o and angular ω_o velocities, where:

$$\dot{u}^T = \begin{pmatrix} v_o & \omega_o \end{pmatrix} \tag{3}$$

The object velocities at the points of contacts follow the corresponding velocities of the fingertips according to the following kinematics constraints :

$$G_h^T - k_h \dot{\Theta}_h \begin{bmatrix} v_o \\ \omega_o \end{bmatrix} = 0 \tag{4}$$

$$\ddot{\Theta}_h = \kappa_h^{-1} \dot{G}_h^T \begin{bmatrix} v_o \\ \omega_o \end{bmatrix} - \kappa_h^{-1} \dot{\kappa}_h \dot{\Theta}_h + G_h^T \kappa_h^{-1} \begin{bmatrix} \dot{v}_o \\ \dot{\omega}_o \end{bmatrix} \tag{5}$$

From Equ. (4), joint acceleration constraint is subsequently given by (*in case of non-singular hand configuration* $\det(\kappa_h) \neq 0$) Equ. (5), in which the constrained hand joints acceleration depend on the Hand Jacobian Inverse κ_h^{-1}.

2.2 Multi-Contact Articulated System Dynamics and Control

The dynamic equation of a four-fingered system is obtained by aggregating four equations, where each equation represents the finger dynamics :

$$M_h \ddot{\Theta}_h + N_h \dot{\Theta}_h + C_h = \tau_h + \kappa_h^T \phi_{tip} \tag{6}$$

whereas the balance equation of the object (*object dynamic*) :

$$G \phi_{tip} = -F_b \tag{7}$$

2.3 Hand-Object Complete System Dynamic Equation of Motion

Since fingertips do not slide over the grasped object, velocity of object frame at contact locations $\dot{v}_{cont} = \kappa_h \dot{\Theta}_h$ is derived in terms of joint

space relations using Equ. (7). From the fact that the reaction force equals the negative of the action force, we are left with the solution of Equ (7) :

$$\phi_{tip} = -\left(G^{+1} F_b + \lambda\eta\right) \tag{8}$$

where G^{+1} is the generalized inverse of the hand transform. Equating the grasped object dynamics with hand dynamics gives :

$$M_h \kappa_h^{-1}\left(\dot{G}^T \dot{u} - \kappa_h \dot{\Theta}_h\right) + N_h \dot{\Theta}_h + C_h = \tau_h - \kappa_h^T \dot{G}^{+1}\left(M_o \ddot{u} + N_o \dot{u} + C_o\right) - \kappa_h^T \lambda\eta \tag{9}$$

where $u_d^c = \begin{bmatrix} p_x & p_y & p_z & p_\theta & p_\phi & p_\psi \end{bmatrix}^T$ to be the position and orientation of the grasped object. Since there is no change at points of contacts, this results in $\left(\frac{\partial G_h}{\partial t}\right) = 0$ although there might be rotational change at each point of contact.

2.4 Cartesian Based PID Hand Control Law

The main control objective is to steer a grasped object to track a defined path (*which we shall designate as Task-Space Path*), Fig. 1-b. In addition to this, the load distribution and internal force control balance. Defining the Cartesian based posture error of the grasped object $e \in \Re^{6\times 1}$: as $e \cong u_d^c - u_a^c$ which is the difference between desired position-orientation vector of the object u_d^c and the actual position-orientation of the object u_a^c. If the Cartesian position and velocity error vectors are given by:

$$e \cong u_d^c - u_a^c \quad \text{and} \quad \dot{e} \cong \dot{u}_d^c - \dot{u}_a^c \tag{10}$$

An object-hand contact system of motion can be described in terms of the applied joint torques τ_h:

$$\tau_h = M_h\left(\kappa_h^{-1}G^{+1}\ddot{\Theta}_a - \kappa_h^{-1}\dot{\kappa}_h\dot{\Theta}_h\right) + N_h + C_h + \kappa_h^T G^{+}\left(M_o\ddot{\Theta}_a + N_h\dot{u} + C_h\right)$$
$$+\kappa_h^T\left(\phi_{cd} - Z_i\int(\phi_{cd} - \eta\lambda)\right) = M_h\kappa_h^{-1}\left(G^{+1}\ddot{\Theta}_a - \dot{\kappa}_h\dot{\Theta}_h\right) + T_{ex} \qquad (11)$$
$$= \left(M_h\kappa_h^{-1}X_h + T_{ex}\right)$$

where :

$$\ddot{\Theta}_a = \left(\left(\dot{v}_O + \dot{\omega}_O\right)^T + k_p e + k_d\dot{e} + k_i\int_0^\tau e d\tau\right) \quad \text{and}$$

$$T_{ex} = N_h + C_h + \kappa_h^T\left(\phi_{cd} - Z_i\int_0^\tau(\phi_{cd} - \eta\lambda)\right)$$

$$X_h = \left(G^{+1}\ddot{\Theta}_a - \dot{\kappa}_h\dot{\Theta}_h\right) \in \Re^{12\times 1}$$

In Equ. (11), τ_h is the computed torque at each joint in the hand, M_h, N_h, and C_h are the hand dynamics. $k_p \cong diag(k_{p1} \cdots k_{p6})$, $k_d \cong diag(k_{d1} \cdots k_{d6})$, and $k_i \cong diag(k_{i1} \cdots k_{i6})$ with k_{pj}, k_{dj} and $k_{ij} > 0$ for all j are the corresponding cartesian PID controller parameters. In Equ. (11) the computed joints torque are evaluated using the Inverse Jacobian matrix which by itself is a composite matrix of the four fingers Jacobian.

The control law defined by Equ. (11) is that it depends on the hand inverse kinematics function and the hand inverse Jacobian. Computation of hand kinematics is not an easy task, specially once talking about real-time hand control. Hence, this is where the potential of employing the Artificial Neural Networks in the hand control can be seen. It will approximate the nonlinear hand kinematics and provide an easy way of computing the hand joints, even without going in deep mathematical computation in real time.

In addition, the difficulty of inverting the matrix κ_h^T, (Hand Jacobian) is that it will be a function of the four fingers all together.

Furthermore, the control law specified by Equ. (11) realizes not only the desired object trajectory but also the desired internal grasping force, as expressed by $\kappa_h^T\left(\phi_{cd} - Z_i \int\left(\phi_{cd} - \eta\lambda\right)\right)$.

3. Object to Joint Space Mapping and Nonlinear Functional Approximation Problem

In Section (2), we have been analyzing the hand control law in the Cartesian space, and the associated problems of the hand inverse kinematics. In order to start analyzing the inverse kinematics problems associated with robot hands, let a single contact finger be investigated. The problem considered here is the solution of the linear equation presented by Equ. (11) which can be rewritten for a single finger as follows :

$$\tau_i = M_i\left(W_i\kappa_i\right)^{-1} X_i + T_{exi} \tag{12}$$

where $X_i \in \Re^{3\times 1}$ and it is the i^{th} decomposed vector of the constrained system acceleration of :

$$X_i = \left(G_i^{+1}\ddot{\Theta}_i - \dot{\kappa}_i\dot{\Theta}_i\right) \tag{13}$$

Finding a good mapping between the object Cartesian motion parameters $\left(u_a^c \quad \dot{u}_a^c \quad \ddot{u}_a^c\right)$ and the associated hand joints Θ_h is a crucial issue that has been investigated heavily in the literature. The mechanism which is being investigated here is that, there is a mapping which can be learned and used. Once this mapping is established, it can be utilized for the hand controller instead of computing the heavy hand Jacobian. This learning mechanism is established via a Neural Networks as will be discussed in Section (4).

4. Hand Differential Motion and Neural Network Mapping Design

Traditionally, once a multi-finger robot hand controller receives sensory information on which the hand motion has to be made, it

calculates a trajectory through the inverse kinematics (using hand Jacobian). An artificial neural network can be seen as a general parametric model that learns to represent a specific input-output relationship in terms of object displacement and joints motion. An artificial neural network can be used for approximating a function from a set of available examples called learning samples or training patterns. For the Hand-Object control system, the relation which will be used to train the network is defined in terms of some training patterns of object Cartesian posture u_a^c, rate of change of object posture Δu_a^c, and rate of change of hand joints space $\Delta\Theta_{k-1}$ as given by Equ. (14) :

$$\Delta\Theta_k = f_{neural}\left(NN, \Delta u_a^c, u_a^c, \Delta\Theta_{k-1}\right)$$

$$\Delta\Theta_k = f_{neural}\left(\varphi_{1f}, \varphi_{2f}, \varphi_{3f}, \varphi_{4f}, w_{1ij}, w_{2ij}, w_{3ij}, w_{4ij}, \ldots, \Delta u_a^c, u_a^c, \Delta\Theta_{k-1}\right) \quad (14)$$

In Equ. (14), the computed change in hand joints space $\Delta\Theta_k$ is made a function of the neural network structural parameters $\left(\varphi_{1f}, \varphi_{2f}, \varphi_{3f}, \varphi_{4f}, w_{1ij}, w_{2ij}, w_{3ij}, w_{4ij}, \ldots\right)$, in addition to the grasped object motion parameters (Δu_a^c, u_a^c, $\Delta\Theta_{k-1}$). Hence the main purpose of the neural network structure is to approximate the nonlinear mapping between changes in hand joints to changes in the object position. The neural network approximation of Equ. (14) is used by the hand Cartesian PID based controller, which indeed depends on the nonlinear mapping between changes in hand joints to changes in the object position.

Inputs and Output training data from the hand-object system have to be prepared and collected for training, as depicted in Fig. 2-a. Inputs to the neural network are: Desired Cartesian object positions u_a^c, changes in such Cartesian positions Δu_a^c, one step change in position of the entire joints in radians $\Delta\Theta_{k-1}$. The outputs are the required changes in joint angles for the entire hand $\Delta\Theta_k$. The desired object posture values are obtained in advance by moving the object to the required position. After the neural network learns this relation between input and output patterns sufficiently, it shows a nonlinear map between the position and orientation of the fingers and those of the object, which usually computed using hand Jacobian inverse.

4.1 Non-Linear Function Approximation Training

A layered feed-forward network consists of a certain number of layers, and each layer contains a certain number of units. There is an input layer, an output layer, and one or more hidden layers between the input and the output layer. Each unit receives its inputs directly from the previous layer (except for input units) and sends its output directly to units in the next layer. Unlike the Recurrent network, which contains feedback information, there are no connections from any of the units to the inputs of the previous layers nor to other units in the same layer, nor to units more than one layer ahead. Every unit only acts as an input to the immediate next layer. Obviously, this class of networks is easier to analyze theoretically than other general topologies because their outputs can be represented with explicit functions of the inputs and the weights. In this research we focused on the use of Back-Propagation algorithm learning method, where all associated mathematical formulae refer to Fig. 2-b. The figure depicts a multi-layer artificial neural net (a four layer) being connected to form the entire network which learns using the Back-propagation learning algorithm. To train the network and measure how well it performs, an objective function must be defined to provide an unambiguous numerical rating of system performance. Selection of the objective function is very important because the function represents the design goals and decides what training algorithm can be taken. For this research frame work, a few basic cost functions have been investigated, where the sum of squares error function was used as defined by Equ. (15):

$$E = \frac{1}{NP} \sum_{p=1}^{P} \sum_{i=1}^{N} (t_{pi} - y_{pi})^2 \tag{15}$$

where p indexes the patterns in the training set, i indexes the output nodes, and t_{pi} and y_{pi} are, respectively, the target hand joint space position and actual network output for the i^{th} output unit on the p^{th} pattern. An illustration of the layered network with the two hidden layers is shown in Fig. 2-b. In this network there are i inputs, m hidden units, and n output units. The output of the j^{th} hidden unit is obtained by first forming a weighted linear combination of the i input values, then adding a bias,

$$a_j = \sum_{i=1}^{l} w_{ji}^{(1)} x_i + w^{(1)}{}_{j0} \tag{16}$$

where $w_{ji}^{(1)}$ is the weight from input *i* to hidden unit *j* in the first layer and $w_{j0}^{(1)}$ is the bias for hidden unit *j*. If we are considering the bias term as being weights from an extra input $x_0 = 1$, Equ. (16) can be rewritten to the form of :

$$a_j = \sum_{i=0}^{l} w_{ji}^{(1)} x_i \tag{17}$$

The activation of hidden unit *j* then can be obtained by transforming the linear sum using a *nonlinear activation function* $g(x)$:

$$h_j = g(a_j) \tag{18}$$

The outputs of the network are obtained by transforming the activation of the hidden units using a second layer of processing units. For each output unit k, first we get the linear combination of the output of the hidden units,

$$a_k = \sum_{j=1}^{m} w_{kj}^{(2)} h_j + w_{k0}^{(2)} \tag{19}$$

absorbing the bias and rewrite the above equation to,

$$a_k = \sum_{j=0}^{m} w_{kj}^{(2)} h_j \tag{20}$$

Then applying the activation function $g2(x)$ to Equ. (20) we can get the k^{th} output :

$$y_k = g2(a_k) \tag{21}$$

Combining Equ. (17), Equ. (18), Equ. (20) and Equ. (21) we get the complete representation of the network as :

$$y_k = g2(\sum_{j=0}^{m} w_{kj}^{(2)} g(\sum_{i=0}^{l} w_{ji}^{(1)} x_i)) \tag{22}$$

The network of Fig. 2-b is a network with two hidden layers, which can be extended to have more hidden layers easily as long as we make the above transformation further. In this manner the error of the network is propagated backward recursively through the entire network and all of the weights are adjusted so as to minimize the overall network error. The network learns the relationship between the previous changes in the joint

angles $\Delta\Theta_{k-1}$, changes in the object posture Δu_a^c , and changes joint angles $\Delta\Theta_k$. This is done by executing some random displacements from the desired object position and orientation.

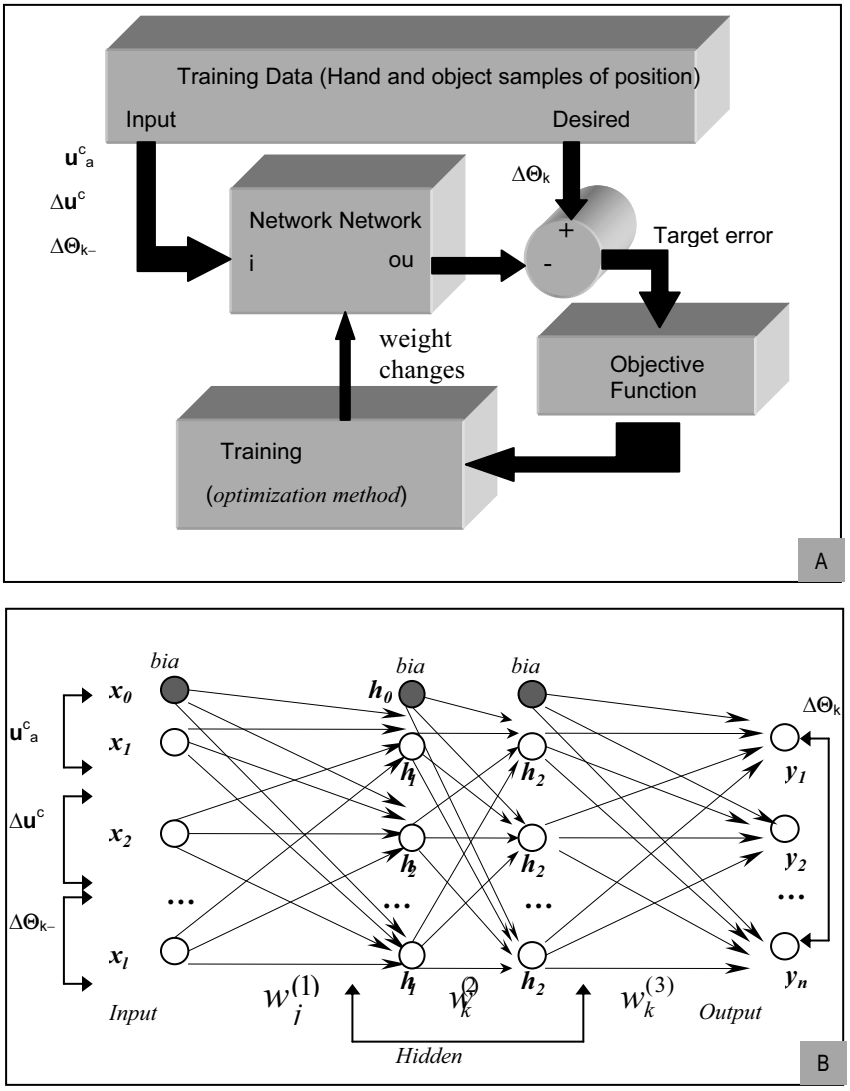

Fig. 2. Neural network structure and training :
a : Supervised learning model.
b : A four layer mapping ANN.

The hand fingers is set up in the desired position and orientation to the object. Different Cartesian based trajectories are then defined and the inverse Jacobian were used to compute the associated joints displacement $\Theta_h(k)$. Different object postures with joint positions and differential changes in joint positions are the input-output patterns for training the employed neural network. During the learning epoch, weights of connections of neurons and biases are changed so that errors decrease to a value close to zero, which resulted in the learning curve that minimizes the defined objective function. It should be mentioned at this stage that the training process has indeed consumed nearly up to three hours, this is due to the large mount of training patterns presented to the neural network.

5. Back-Propagation and Hand-Neural Weight Adjustment with Gradient Descent Method

For the neural network shown in Fig. 2-b, the learning process is based on a suitable error function, which is then minimized with respect to the weights and bias. Since the network has differential activation functions, the activations of the output units become differentiable functions of input variables, the weights and bias. Defining the differentiable error function of the network outputs as given by Equ. (15), then the error function itself is a differentiable function of the weights. Therefore, derivative of the error with respect to weights can be evaluated, and these derivatives are then used to find the weights that minimize the error function, by using the popular gradient descent optimization methods.

5.1 The Learning Process

For the considered feed-forward network shown in Fig. 2-b, with the chosen differentiable non-linear activation functions and the differential error function, each unit j is obtained by first forming a weighted sum of its inputs of the form,

$$a_j = \sum_i w_{ji} z_i \tag{23}$$

where in Equ. (23), z_i is the activation of an unit, or input. Apply the activation function of Equ. (18), this gives :

$$z_j = g(a_j) \tag{24}$$

One or more of the variables z_j in Equ. (23) could be a training input pattern, in which case we will denote it by x_i. Similarly, the unit j in Equ. (24) could be an output unit, which we will denote by y_k. The error function will be written as a sum, over all patterns in the training set, of an error defined for each pattern separately,

$$E = \sum_p E_p \,,\; E_p = E(Y;W) \tag{25}$$

where p indexes the patterns, Y is the vector of outputs, and W is the vector of all weights. E_p can be expressed as a differentiable function of the output variable y_k. Using Equ. (25), evaluating the derivatives of the error functions E with respect to the weights and bias, these derivatives as sums over the training set patterns of the derivatives for each pattern separately. During the forward pass, for one pattern at a time and with all the inputs, the activations of all hidden and output units in the network can be computed by successive application of Equ. (23) and Equ. (24). Now consider the evaluation of the derivative of the error function E_p with respect to some weight w_{ji} using the chain rule :

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} = \delta_j \frac{\partial a_j}{\partial w_{ji}} = \delta_j z_i \tag{26}$$

where we define

$$\delta_j = \frac{\partial E_p}{\partial a_j} \tag{27}$$

From equation Equ. (26), the derivative can be obtained by multiplying the value of δ for the unit at the output end of the weight by the value of z for the unit at the input. Thus the task becomes to find the δ_j for the two hidden and output units in the network. For the output unit, δ_k ;

$$\delta_k = \frac{\partial E_p}{\partial a_k} = \frac{\partial E_p}{\partial y_k} g'(a_k) \tag{28}$$

Hidden units can influence the error only through their effects on the unit k :

$$\delta_j = \frac{\partial E_p}{\partial a_j} = \sum_k \frac{\partial E_p}{\partial a_k}\frac{\partial a_k}{\partial a_j} \tag{29}$$

The first factor is just the δ_k of unit k so

$$\delta_j = \frac{\partial E_p}{\partial a_j} = \sum_k \delta_k \frac{\partial a_k}{\partial a_j} \tag{30}$$

For the second factor we know that if unit j connects directly to unit k then $\partial a_k / \partial a_j = g'(a_j) w_{kj}$, otherwise it is zero. So we can get the following back-propagation formula,

$$\delta_j = g'(a_j) \sum_k w_{kj} \delta_k \tag{31}$$

which means that the values of δ for a particular hidden unit is obtained by propagating the δ's backwards from units later in the network. Recursively applying the equation gets the δ's for all of the hidden units in a feed-forward network. With this algorithm, weights are updated in the direction in which E decreases along negative gradient, as in Equ. (32) :

$$\Delta w_{ji}^{(\tau+1)} = -\eta \frac{\partial E}{\partial w_{ji}} \tag{32}$$

where η is the learning rate, and α is the momentum term,

$$\Delta w_{ji}^{(\tau+1)} = -\eta \frac{\partial E}{\partial w_{ji}} + \alpha \Delta w_{ji}^{(\tau)} \tag{33}$$

The weight change is a combination of a step down the negative gradient, plus a fraction α of the previous weight change, where $\alpha = 0.75$. In this respect, the used trained ANN structure will rather achieve the similar computation done via the hand Jacobian inverse which uses numerical routines to achieve such inverse.

6. Hand Motion and System Simulation

Simulation results for the employed multi-finger robot hand with twelve degrees of freedom are presented in this section. The multi-layer neural network used in the simulation was four layer neural network architecture. The network consists of 18 inputs, 12 outputs and 50 hidden neurons. The neural net map the 18 inputs characterizing the object Cartesian position and hand joint positions into the 12 differential change in fingers positions. In order to assess the proposed control algorithm,

simulation of a constrained dynamics has been achieved using the kinematics and dynamic models of the CYBHAND[9]. An object sinusoidal motion and path was defined along different axes. At the beginning, the hand has been simulated with conventional inverse kinematics algorithms, where training patterns have been generated. Such training patterns have been based on object Cartesian motion and associated joints displacement. The hand has been run for large number of trials for producing as large as possible of training patterns.

6.1 Hand Displacement Training Patterns

Hand training patterns have been generated by letting the hand follow some pre-defined Cartesian trajectory, while holding a grasped object of known physical dimensions, as already shown in Fig. 1. The hand motion was defined in terms of moving the object center of gravity along the *y*-axis and the *z*-axis in a sinusoidal fashion. Typical training patterns are shown in Fig. 3-a, where the hand has been allowed to follow a pre-defined path over 5 sec manipulation time. In Fig. 3-a, it is revealed that: The object will reach a maximum displacement along the y-axis of 0.01 *m* and a maximum displacement along the *z*-axis of 0.005, then the hand is allowed to move also in different directions with different maximum displacement. In this sense, the associated patterns Δu_a^c, u_a^c, $\Delta\Theta_{k-1}$ are tabulated in the proper format to be suitable for the neural network training. The quantity of the training pattern was reaching a size of 500 for a single variable (*e.g.* Δu_a^c), as presented by the plot in Fig. 3-a. Hence to validate the neural network ability to model the hand inverse kinematics, the error between a typical neural output node (e.g. θ_{33}) with the actual one has been computed and analyzed. For instance, Fig. 3-b shows the error histogram of one neural net output node, which shows a great deal of slim spread around the zero reference, hence validating the ability of the network to reduce the mapping error. In addition, Fig. 3-c and Fig. 3-d depict the neural network mapping accuracy associated with θ_{11} and θ_{33}, where it is clearly shown the accuracy of the employed neural network system to reconstruct a finger joint displacement even from untrained patterns.

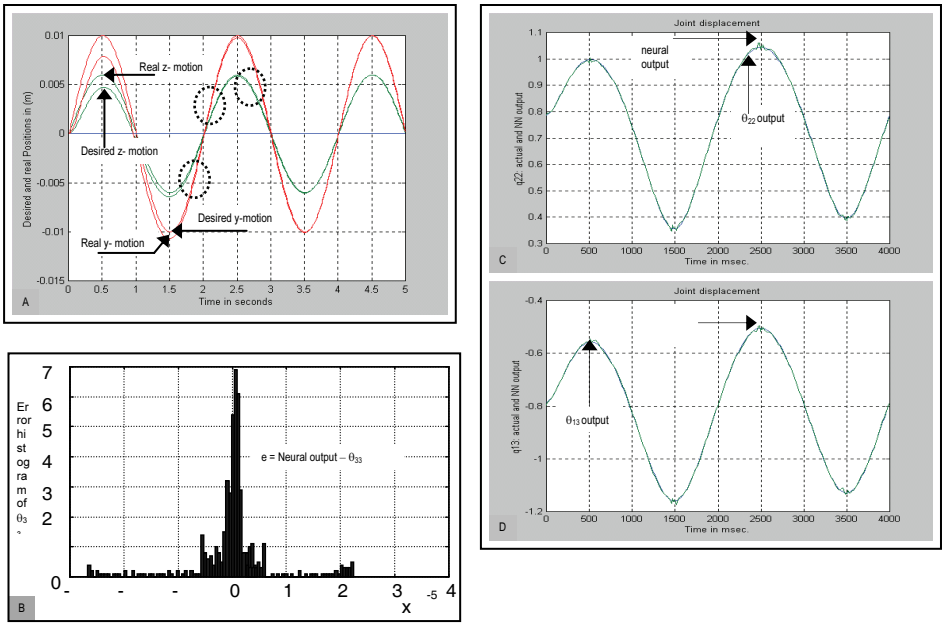

Fig. 3. Training patterns and errors :

a. Object motion (y and z axis), training patter generation.

b. Histogram of error between actual θ_{33} and neural net output.

c. Validating neural network mapping for θ_{22} .

d. Validating neural network mapping for θ_{13} .

6.2 Execution Process and Neural-Hand Controller Validation

Once the neural network has learned the required mapping from the presented examples, it is ready to be applied to the hand controller which depends heavily on the hand inverse kinematics. Hence the network is presented with some object motion directions, where it finds the associated hand joint-space used in the hand controller. The execution process starts first with employing the trained neural network in the hand dynamic controller (*which mainly depends on the hand Jacobian inverse*). Once the object position and orientation have been defined, the neural networks computes the associated hand joint positions by presenting the network with some patterns which were not included during the training process. Once the neural network presented with such pattern, it associates the input patterns with some trained joint displacement patterns. Such learned patterns at the neural output nodes are then employed in the hand controller.

Finally the ANN is employed in the hand controller for the calculation of joint displacement as required by the full controller already presented by Equ. (11). The ANN has shown it was able to reproduce a good mapping mechanism as compared to other full kinematics-based relations. The robot hand has been simulated dynamically by MATLAB-software, where such hand simulation is presented in Fig. 4. For instant, Fig. 4-a demonstrates the associated hand joint-space vector Θ_h required to move the object in a pre-defined trajectory, where as Fig. 4-b shows the error associated with the object displacement. Figure 4-c illustrates the required torque to move the hand joints, where it is apparent that the joint torques are working collectively, not just to move the fingers, but the generation of the suitable torques to grasp the object during the course of motion. Finally, the associated object displacement with the neural controller is shown in Fig. 4-d, where it is showing the ability of the hand to move the object along the required axis of motion in a smooth manner.

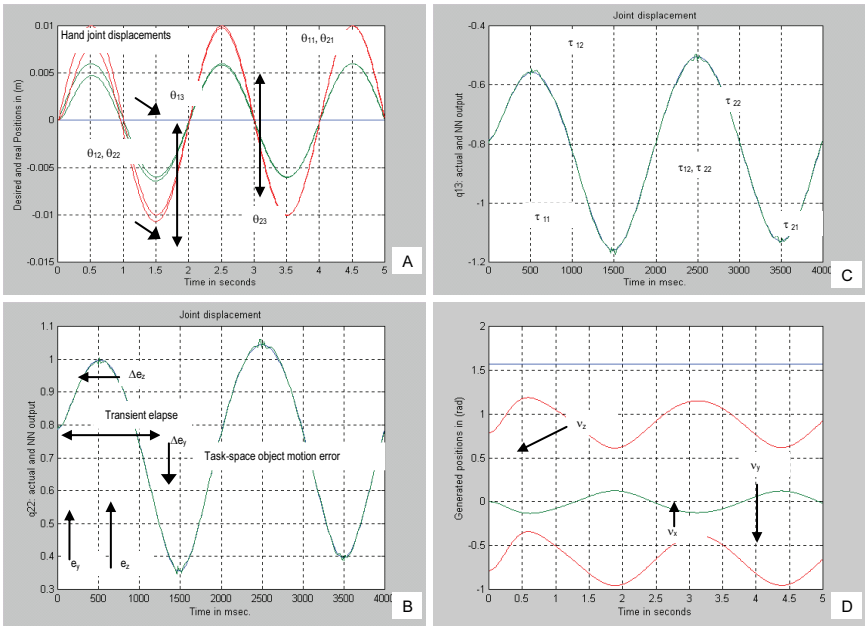

Fig. 4 . Validating the neural hand controller :

a. : Hand joint displacements as ANN used in hand controller.

b. : Object Cartesian error and change in error with ANN hand controller.

c. : Hand torques with neural controller.

d. : 3-D defined and real object velocities.

However, the implemented hand controller strategy does in fact need a large training data to fairly approximate the nonlinear mapping in such a way to cover most the non-singular hand working space. Larger training patterns could result in longer hand training time and the possibility of not getting a convergence neural network. In addition to this, there could be the case where passage of the object over singular hand posture at which the hand Jacobian inverse is not possible.

7. Conclusions

The issue of the inverse dynamics for multi-fingered robot hand has been studied where the object motion is defined in a Cartesian based system, hence the differential system Jacobian plays an important role. In this paper a scheme for the control of a robotic multi-finger has been presented. The nonlinear relation between the Cartesian object posture and the associated hand joint-space settings and control signals mapping was learned via a four layers artificial neural networks trained for most possible object displacement. The validity of this control scheme is confirmed by computer simulations, where a task-space object motion has been defined to move over the *y* and *z* axis Cartesian coordinates while grasping the object with a stable grasp. This approach is effective because it essentially decomposes complex geometric calculations into simple mapping of the network. The proposed controller strategy however, needs in fact a large training data to fairly approximate the nonlinear mapping, in addition to possible passage of the object over singular hand posture at which the hand Jacobian inverse is not possible.

References

[1] **Bekey, G., Tomovic, H.** and **Karplus**, **W.,** Knowledge-Based Control of Grasping in Robot Hands Using Heuristics from Human Motor Skills, *IEEE Transactions on Robotics and Automation*, **9**(6): 709-721 (1993).

[2] **Xi, N., Tran, J.** and **Bejczy, K**., Intelligent Planning and Control for Multi-robot Coordination: An Event-Based Approach*, IEEE Transactions on Robotics and Automation*, **12**(3): 439-445 (1996) .

[3] **Huan, L., Iberall, T.** and **Bekey, G.,** Neural Network Architecture for Robot Hand Control, *Proceedings of the IEEE International Conference on Neural Networks*, SAN DIEGO, pp:38-43 (1988).

[4] **Hashimoto, H., Kubota, T., Sato, M.** and **Harashima, F.,** Visual Control of Robotic Manipulator Based on Neural Networks, *IEEE Transactions on Industrial Electronics*, **39**(6): 490-495 (1992).

[5] **Hashimoto, H., Kubota, T., Kuduo, M.** and **Harashima, F.,** Self-organizing Visual Servo System Based on Neural Networks, *IEEE Control Systems Magazine*, pp: 31-36 (1992).

[6] **Guez, A.** and **Ahmed, Z.,** Solution to the Inverse Kinematics Problem in Robotics by Neural Networks, *Proceedings of the International Conference on Neural Networks*, USA, (1988).

[7] **van der Smagt**, **P. P.** and **Krose, J**., A Real-time Learning Neural Robot Controller, *Proceedings of the 1991 International Conference on Artificial Neural Networks*, ICANN-91, FINLAND, pp: 351-356 (1991).

[8] **Schram, G., Linden, F., Krose, B.** and **Groen, F.,** Visual Tracking of Moving Objects Using a Neural Network Controller, *International Journal of Robotics and Autonomous Systems*, **18**: 293-299 (1996).

[9] **Al-Gallaf, E.** and **Warwick, K.,** (CYBHAND) A Four Fingered Dexterous Hand, *Proceedings of the IEEE International Symposium in Signal Processing, Robotics, and Neural Networks*, France, pp: 657-777 (1996).

[10] **Maciejewski, A.** and **Klein, C.,** Numerical Filtering for the Operation of Robotics Manipulators through Kinematically Singular Configurations, *Journal of Robotics System*, **5** (6): 527-552 (1988) .

[11] **Nakamura, Y.** and **Hanafusa, H.,** Inverse Kinematics Solutions with Singularity Robustness for Robot Manipulator Control, *ASME Journal of Dynamic Systems Measurement and Control*, **108**:163-171 (1986).

[12] **Maciejewski, A.** and **Klein, C.,** The Singular Value Decomposition: Computation and Applications to Robotics, *International Journal of Robotics Research*, **8** (6): 63-79 (1989).

[13] **Klein, C.** and **Blaho, B.,** Dexterity Measures for the Design and Control of Kinematically Redundant Manipulators, *International Journal of Robotics Research*, **6** (2): 71-83 (1987).

[14] **Dubey, R.** and **Luh, J.**, Redundant Robot Control Using Task Based Performance Measures, *Journal of Robotics System*, **5** (5): 409-432 (1988) .

[15] **Lee, S.,** Dual Redundant Arm Configuration Optimization with Task-Oriented Dual Manipulability, *IEEE Transactions in Robotics and Automation*, **5** (1): 78-97(1989).

[16] **Lilly, K.** and **Orin, D.,** Efficient Dynamic Simulation for Multiple Chain Robotics Mechanisms, In: D. Bernard and G. Man (Ed.), *Proceedings of 3rd Annual Conference Aerospace Computational Control,* PASADENA, pp:73-87 (1989) .

[17] **Rodriguez, G., Jain, A.** and **Kreutz-Delgado, K.,** A Spatial Operator Algebra for Manipulator Modeling and Control, *International Journal of Robotics Research*, **10**: 371-381 (1991).

[18] **Wohlke, G.,** NEURO-FUZZY Based System Architecture for the Intelligent Control of Multi-Finger Robot hands, *Proceedings of the IEEE International Conference on Fuzzy Systems*, ORLANDO (1994).

[19] **Zsiros, P., Baranyi, P.** and **Korondi, P.,** A Generalized Neural Network for a Humanoid Hand, *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE' 2000)*, PUEBLA, pp: 523-528 (2000).

[20] **Li-Ren, L.** and **Taipei, H.,** DSP-Based Fuzzy Control of a Multi-fingered Robot Hand, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetic*s, VANCOUVER (1995).

[21] **Doersam, T., Ftikow, S.** and **Streit, I.,** Fuzzy-Based Grasp Force Adaptation for Multi-fingered Robot Hands, *Proceedings of the IEEE International Conference on Fuzzy Systems,* **3**, ORLANDO (1994).

[22] **Caihua, X.** and **Youlun, X.,** Neural-Network Based Force Planning for Multifingered Grasp, *International Journal of Robotics and Autonomous Systems,* **4**:365-375 (1997).

[23] **Fuentes, O.** and **Nelson, R.,** Learning Dexterous Manipulation Strategies for Multifingered Robot Hands Using the Evolution Strategy, *International Journal of Machine Learning*, **31**: 223-237(1998).

[24] **Fischer, T., Rapela, D.** and **Woern, H.,** Joint Controller for the Object-Pose Controlling on Multi-finger Grippers, *IEEE International Conference on Advanced Intelligent Mechatronics*, GEORGIA (1999).

استخدام الشبكات العصبية الاصطناعية للتحكم في حركة يد الروبوت الاصطناعية متعددة الأصابع

إبراهيم عبدالله الجلاف

قسم الهندسة الكهربائية والإلكترونية، كلية الهندسية، جامعة البحرين

ص ب: ١٣١٨٤ مملكة البحرين

المستخلص. يهدف هذا البحث إلى استخدام الشبكات العصبية الاصطناعية للتحكم في حركة يد الروبوت الاصطناعية متعددة الأصابع وذلك عبر تعليم النظام جميع الحركات المطلوبة. إذ كما هو معروف أن التحكم في حركة يد الروبوت متعددة الأصابع هي عملية جد معقدة وتحتاج إلى الكثير من عناصر التحكم الحديثة وذلك نظرًا لصعوبة تحريك الجسم المقبوض بين الأصابع وخاصة بسرعات متفاوتة. ومن هذا المنطلق يأتي استخدام الشبكات العصبية الاصطناعية لتسهيل معرفة وتعلم أوضاع مفاصل اليد وذلك بتدريب الشبكة العصبية الاصطناعية على عدة مواقع للجسم المقبوض ومن ثم تزويد نظام التحكم بهذه المعلومات والإشارات والتي تحتاج إلى عمليات حسابية معقدة لإيجادها ولاحتسابها.