N93-11962

A NEURO-COLLISION AVOIDANCE STRATEGY FOR ROBOT MANIPULATORS.

Joel P. Onema and Robert A. Maclaunchlan*
Department of Electrical Egineering and Computer Science
*Department of Mechanical Engineering and Industrial Engineering
Texas A & I University, Kingsville, Texas 78363

ABSTRACT

The area of collision avoidance and path planning in robotics has received much attention in the research community. Our study centers on a combination of an artificial neural network paradigm with a motion planning strategy that insures safe motion of the Articulated Two-Link Arm with Scissor Hand System relative to an object. Whenever an obstacle is encountered, the arm attempts to slide along the obstacle surface, thereby avoiding collision by means of the local tangent strategy and its artificial neural network implementation. This combination compensates the inverse kinematics of a robot manipulator. Simulation results indicate that a neuro-collision avoidance strategy can be achieved by means of a learning local tangent method.

I. INTRODUCTION

The problem of collision avoidance in robotics[8] Bradley, Hollerbach, Johnson and Lozano Perez can be described as follows: given a starting and a goal configuration of some object or linked group of objects in an environment cluttered with obstacles, find a path of connecting line segments from the starting to the goal configuration, such that the object to be moved follows this path without interfering with any obstacle of the environment.

Traditionally [1], designing a robot control system involves two steps first a set of kinematic equations which express the physical constraints of the robot are derived, and second, a computer program model is implemented generating arm configuration sequences that move the robot's end-effector from its current position to a target position. While simulation runs works well in a laboratory, they often suffer from serious limitations when applied in realistic environments. Wear and tear on mechanical parts changes the kinematics of the robot manipulators and sensor characteristics tend to wander with time. When such changes occur, the control program must be updated or the robot must be maintained. The response time slows down when the degree of difficulty increases by unpredictable obstacles with different shapes or change of position in the workspace. A high degree of autonomous adaptive learning is the ultimate approach to consider when solving these major impairments. This paper focuses on solving the inverse kinematics of an articulated two-link arm with a scissor hand system which can beconsidered as embody by a sensitive skin type sensors [1]. We study the following problem: given the position and orientation of the manipulator, calculate all possible sets of joint angles over time, which could be used to attain a given object position and orientation. The initial solution to the problem enables the manipulator to attain its goal without a planned strategy (no collision avoidance). The final strategy used combines motion planning and artificial neural networks. This strategy is to insure a collision-free motion of the robot manipulator. Simulation results are presented and a possible extension to this work is discussed. This paper is an extension to the work reported in [1, 4].

II. AN ADAPTIVE LEARNING APPROACH

The subject of neural network is hardly new[14], but there has been much more recent[11] progress in developing methods for training more complex configurations of these networks. The simplest net is the perceptron [13, 14], whose training procedure can be called an error-correcting procedure. The weights are revised whenever a mistake is made. Both the output and the correct answers are expressed as 0 or 1 else an error occurs. For the perceptron, once the weighted sum is computed, the activation of the output unit is determine by a threshold logic (0 or 1) depending whether the threshold value was exceeded. The thresholds activation function creates nonlinearity. When The Classes of behavior, etc. are linearly separable the perceptron will find a line or a plane that yields no error. It concentrates on reducing errors, but may perform poorly when classes of behavior etc, are not linearly separable.

In the LMS[12] (Least Mean Square) learning system, the correct answers are still express in terms of 0 and 1, however they are not restricted to a binary values but can have continuous real values. The goal of the LMS training procedure is to minimize the average (squared) distance from the true answer to the output. The LMS training procedure performs relatively well when classes are not linearly separable, because the best line of separation is found in terms of the minimum error distance. To compute error rates, decision criteria are needed to determine the class that is selected. For a single output this is the class that is closest to the output. The error distance to the correct answer can be small, while erroneous answers may have larger distance or be on the wrong side of the boundary. Eventually, LMS training should converge to the minimum distance. This iterative technique of minimization is also called gradient descent[12]. The weights are constantly adjusted in the direction of the greatest reduction of error, example we expect

that we are moving "down hill" in the direction of the minimum. The main problems for gradient descent methods is oscillation and not converging or poor convergence rate. The rate of convergence highly depends on the learning rate[11, 12]. The LMS learning system is completely linear. The weighted sum is directly used to activate the output unit; no additional mapping by the activation function is made. A linear function would not be useful, because the overall system would still be linear and would not effectively use the hidden units. While the threshold logic of the activation function perceptron is nonlinear, from the mathematical view point, the LMS needs a continuously differentiable function. This is not the case for threshold logic. Thus an alternative is activation function which is differentiable is used known as a logistic or sigmoidal function. For any real valued numbers, the activation function is a continuous value, this is a valid range in probability, these functions have been widely used in statistics.

Both the perceptron and the LMS learning System attempt to derive a linear separator from labeled data. Perceptron and LMS can be described as single layer neural networks, where a layer represents a set of output devices. The weights are termed feed forward, because they flow in the forward direction. Starting with the inputs node and weights, no weight cycles back to an input node or output node of previous layers.

Very few real-world applications are truly linearly separable this basic discussion is beyond the scope this paper. The multilayer neural network can be trained as by a generalized version of the LMS training procedure for a nonlinear logistic outputs known as Backpropagation [11]. We consider this paradigm to be the most suited for our needs. It is flexible and can be easily implemented. It is one the most popular and widely used neural networks today. Backpropagation has the ability to learn mappings by example, by a process of learning [1, 3, 5, 9, 10]. It uses a learning procedure that aims to minimize the mean squared error between the actual outputs of the network and some desired outputs. Because of convergence problems that found in the LMS and the learning rate constraints as previously mentioned we opted for modular[1] approach. It is much faster to train several smaller networks than one large one. We kept the number of neuron as small as possible, since removal of redundant neurons can make noticeable difference in the training time. The learning strategy that permits to solve the inverse kinematics problem [5, 7] and avoid obstacles is based upon a simple theoretical robot kinematics[1, 3] as presented in (fig. 1) and the value of the local tangent as presented in[2].

III. LOCAL TANGENT

The robot arm attempts to slide along the obstacle surface as reported in [2] (Fig. 2), whenever an obstacle is encountered. This sliding is accomplished by a coordinated move between joints J1 and J2, based on the value of the local tangent (Fig. 4). The local tangent is obtained from the following formulas depending on were the obstacle occurs in the work space. There are three categories (Fig. 3). Note: a complete derivation of the following equation can be found in [2]

Type I are those obstacles that obstruct link 1. Since link 2 is irrelevant in this case, then $d\theta_1=0$ and $d\theta_2 \neq 0$. The local tangent for

Type I is vertical [2].

Type II consists of the obstacles that obstruct link 2. Assume that link 2 is obstructed at point C by a type II obstacle, at the distance L_d from the joint J2 (Fig. 4). Then, the estimates [2] of $d\theta_1$, and $d\theta_2$ at C can be found as follows:

$$c_x=L_1\mathrm{COS}(\theta)+L_d\mathrm{COS}(\theta_1+\theta_2)$$

x coordinate of C (1)

$$c_y=L_1\mathrm{SIN}(\theta_1)+L_d\mathrm{SIN}(\theta_1+\theta_2)$$

y coordinate of C

The expression for the local tangent of type II at point C is given by the estimate [2]:

$$\frac{d\theta_2}{d\theta_1}=\left(\frac{L_1}{L_d}\mathrm{COS}\theta_2+1\right) \qquad (2)$$

Type III are obstacles that obstruct the wrist. Sliding of the wrist P along the obstacle corresponds to its moving along the line segment LM (Fig. 4). β_1 is the angle between the line perpendicular to link 2 and the line from P to the obstacle and β_2 is the angle between the line LM and the positive x axis. Then

$\beta_2 = \theta_1+ \theta_2+ \beta_1-\pi$. The expression for the local tangent estimate [2] of type III appears as:

$$\frac{d\theta_2}{d\theta_1}=-\left(\frac{\frac{L_1}{L_2}+\mathrm{COS}\theta_2+\mathrm{SIN}\theta_2 \tan(\theta_2+\beta_1)}{\mathrm{COS}\ \theta_2+\mathrm{SIN}\theta_2\tan(\theta_2+\beta_1)}\right) \qquad (3)$$

IV. LEARNING STRATEGY

The Neural Network is trained to map the Cartesian coordinate to the joint angle coordinate transformation for the three degree of freedom robot arm. The net learns the topology and unknown transformation from presentation of examples such that a solution to the inverse kinematics is found and an obstacle can be avoided. First, the network is presented with a set of examples for training, then tested to generate an output within an acceptable tolerance. Our study centers on the following mapping: given the Cartesian position (x, y), generate the accurate joint angles. Secondly, the net is trained and tested to recognize the obstacle types, (I, II, III). This permits the usage of the local tangent and the generation of a collision avoidance motion, in other words, the arm learns to slide along the obstacle surface with no contact at the tangential point. A combination of the modules described above can be an important tool that can carry out the extensive computation and planning needed to achieve an intelligent move, therefore avoiding an obstacle.

A possible real-time architecture [2] implementation of this study is presented in Fig. 10. Sensitive Skin sensor information and arm position are passed to the neural planner/ controller that generates a safe motion of the arm.

V. SIMULATION RESULTS

As already stated, we opted for a modular approach, since it is much faster to train several smaller networks than one large net. Fig. 5 shows the general network configuration followed by subnets configurations. The subnets configurations are three layer networks ranging from two to five nodes at the input layer and six to twelve nodes for the middle layer and raging from one to three nodes for the output layer. We have trained and tested the nets for a complete mapping of the kinematic transformation of the experimental arm/hand system 3 degree of freedom (dof). In our experiments [1] we focused on training the net by mapping the joint angles (shoulder, elbow and wrist) and their joint position. Our experiment has indicated an ease in training the wrist, the elbow and shoulder, because of the modularity and proper data normalization. Our first experiment focused on the kinematics transformation when there is no obstacle. Maximum and RMS (Root Mean Square) error values for a case study of 2 dof (fig. 6a, 6b) illustrate a downward trend, indicating the smoothness of the learning curves. Fig. 7a shows the expected joint position results graph from the examples presented to the network by training. Fig. 7b shows a graph depicting the network output when tested. As one might observe the network performance is within tolerable range when compared to the expeted results graph. The second experiment targets the type II obstacle using simulated sensor data. Fig. 8a shows the expected results graph from examples presented to the network via training and fig. 8b depicts the network output when tested. The network output is an adaptation of the joint angles as the robot manipulator moves from an initial to final configuration space by sliding along the obstacle surface. As one might observe the network performance is also within tolerable range when compared to the expected results graph.

VI. CONCLUSION AND FUTURE PROJECTION

In this paper we have shown that a neural network paradigm can compensate the inverse kinematic behavior of our model and promising results of the neuro-collision avoidance strategy have been obtained. Our experiments have indicated that a modular approach is the most appropriate. The network was able to learn and mimic a decision making strategy by means of the learning tangent approach, thereby allowing the manipulator to achieve a collision free motion. Real-time implementation of this experiment will require a parallel architecture that can be of great benefit to the enhancement of the overall system performance. Because of the sensor data complexity, in future work the Learning Tangent method could be compensated by a fuzzy logic system [15] that bases its decisions on inputs in the form of linguistic variables, for example smooth, slippery and rough. In our case it will be the obstacle types (I, II, III). Our ultimate goal is to enhance the capabilities of the experimental arm/hand system by means of applicable findings leading to its real-time implementation. Current applications could include the NASA EVA Retriever and related space operation. In manufacturing, the learning tangent can play an important (safety) role in discerning the sudden presence of an operator in the working space resulting from careless behavior.

REFERENCES

[1] Joel P. Onema, A Motion Planning Strategy and Neural its Network Implementation for an Articulated Two-Link Arm with Scissor Hand System, MS Thesis, EE&CS Department Texas A&I University, May 1991.

[2] Cheung, E and Lumelsky, V. '' Proximity sensing in Robot Manipulator Motion Planning System and Implementation'', IEEE Conference on Robotics and Automation, Vol. 5 No. 6. December 1989. pp. 740-751.

[3] Josin, G. , Charney, D. and White,'' A Neural Representation of an Unknown Inverse Kinematics, Transformation'', Neural Systems Inc. 1989, Vancouver, B. C. Canada.

[4] Joel P. Onema, Barbara S, Schreur, Robert A. Maclaunchlan, Muhammed Ashtijou, '' A Motion Planning Strategy and its Neural Network Implementation for an Articulated Two-Link Arm with Scissor Hand System", IJCNN-91. Seattle, Washington.

[5] Guez, A. and Ahmed, Z. ''Solution to the Inverse Kinematics Problem in Robotics by Neural Networks'', Proceeding of the Second InternationalConference on Neural Networks, San Diego, California, 1988, Vol. 2, pp. 14-20.

[6] Richard P. Paul,'' Robot Manipulators:Mathematics, Programming, and Control'', Cambridge , MA: MIT Press 1986.

[7] Guo, J. and Cherkassky, V. ''A Solution to the Inverse Kinematics Problem in Robotics Using Neural Network Processing. Proceeding of the IJNCNN-89, Washington DC, Vol 2.PP. 289-304.

[8] Brady, M. J , Hollerbach, T., Johnson, T. Lozano- Perez ''Robot Motion Planning'', MIT Press, MA. 1982.

[9] Judith E. Dayhoff,''Neural Network Architectures'', Van Nostrand Reinhold, NY,1990.

[10] Kung, S. Y. and Hwang, J. N. "Neural Network Architecture for robotics". IEEE Transaction on Robotics and Automation. 1989. Vol. 5. Nr. 5. PP. 641-657.

[11] Rumerlhart, D.,E. and McClelland, J. L., Parallel Distributed Processing: Explorations in Microstructure of Cognition, vol. I , MIT Press, 1986.

[12] Widrow, B., and Stearns, S. D. Adaptive Signal Processing , Prentice-Hall, Inc., Engelwood Cliff, NJ, 1985.

[13] R.P. Lipman: '' An Introduction To Computing With Neural Nets'', IEE ASSP Magazine, April 1987, pp.4-22. perceptron.

[14] Minsky M. L. S. Paper, Perceptrons: ''An introduction to computational Geometry". MIT Press, Cambridge, Mass 1969.

[15] Bart Kosko ,'' Neural Network and Fuzzy Systesms '', Prentice hall, Inc. Engelwood Cliff, NJ, 1991.

KINEMATICS EQUATIONS

Forward transformation:

$X= L_1COS\theta_1+L_2COS(\theta_1+ \theta_2)$ and

$Y= L_1SIN\theta_1+L_2SIN(\theta_1+\theta_2)$

Inverse transformation:

$COS\theta_2=(X^2+Y^2-L^2{}_1-L^2{}_2)/2L_1L_2$

$\theta_1=\arctan(X/Y)\arctan[L_2SIN\theta_2/(L_1+L_2COS\theta_2)]$

L_1 : length of first link

L_2 : length of second link

θ_1 : joint of first link

θ_2 : joint of second link

Fig. 1

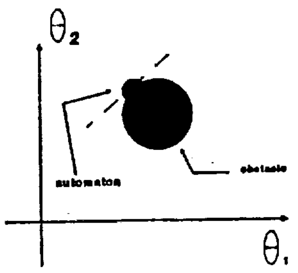


Fig. 2

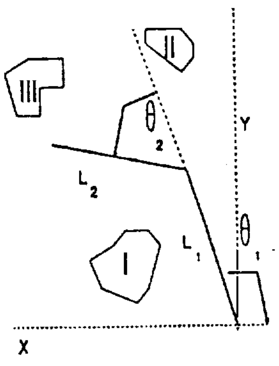


Fig. 3

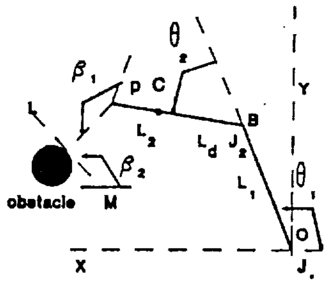
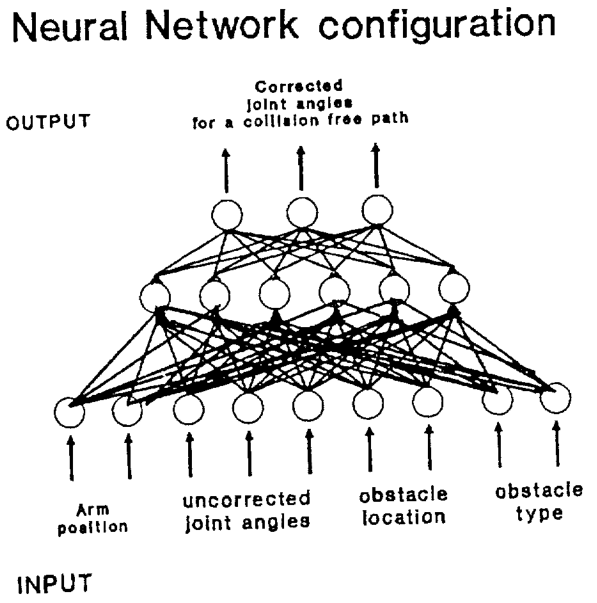


Fig. 4



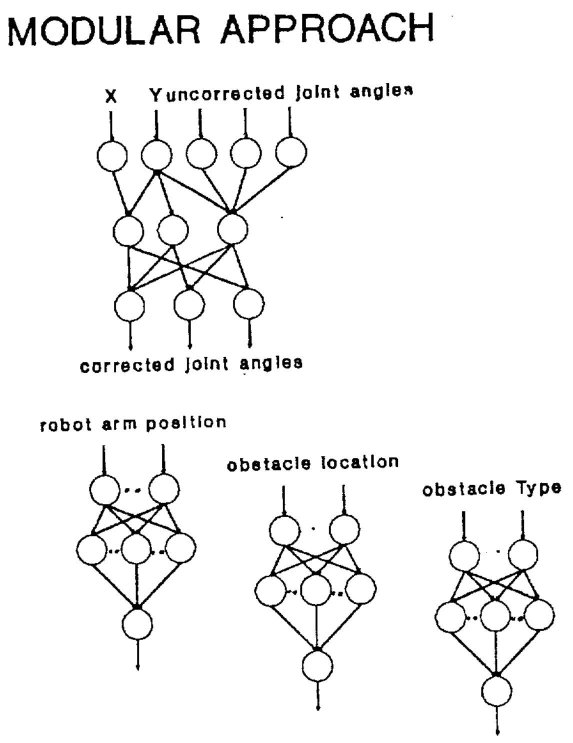


Fig. 5.

Learning the Kinematics Transformation

with no Obstacle

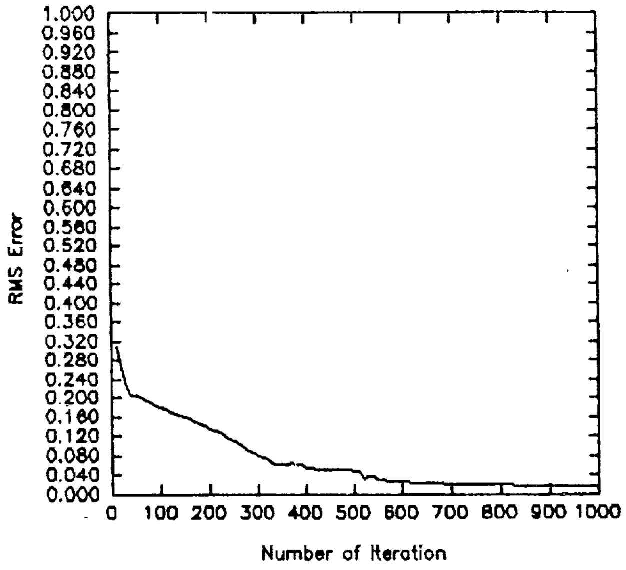


fig. 6a.

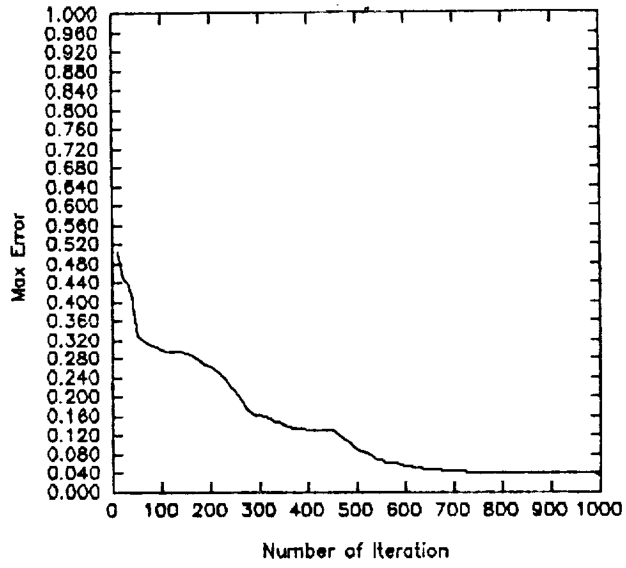


fig. 6b.

Expected Output

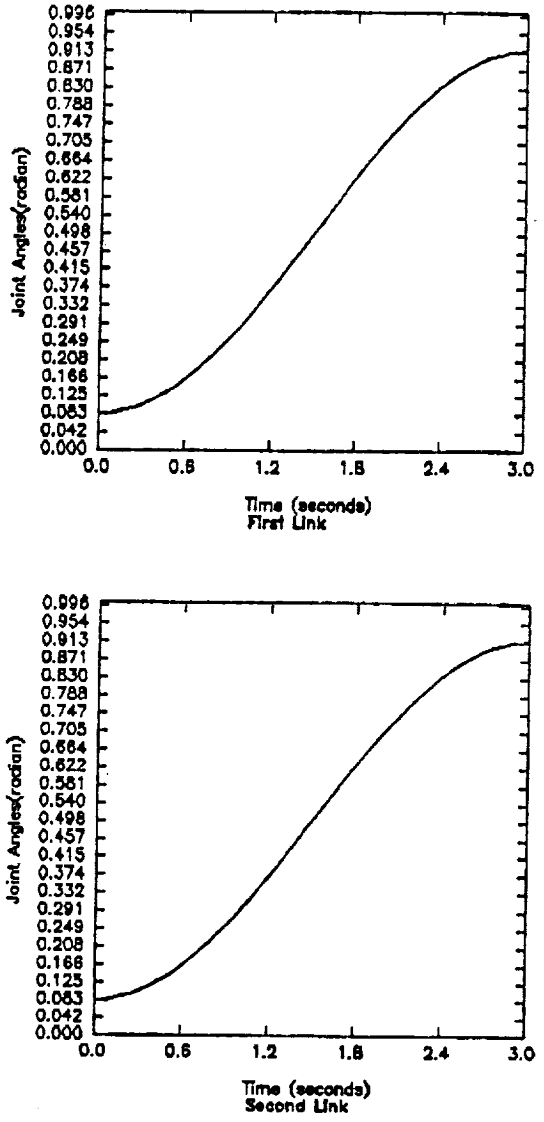


Fig. 7a.

Network Output

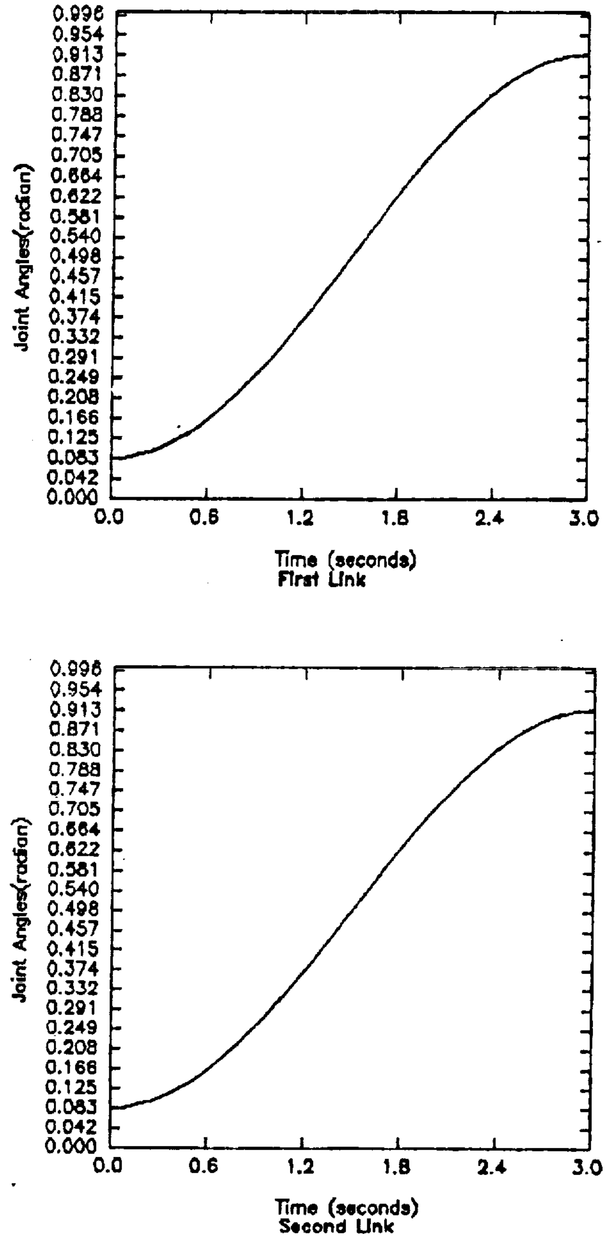


Fig. 7 b

Learning the kinematics transformation
for a Type II Obstacle

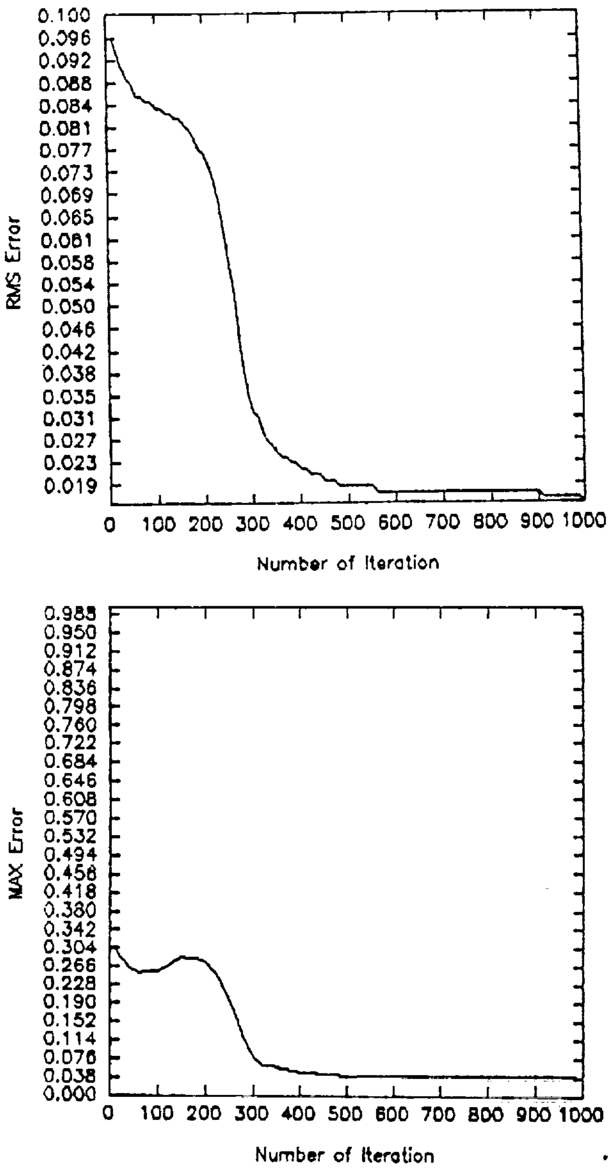


Fig. 8.

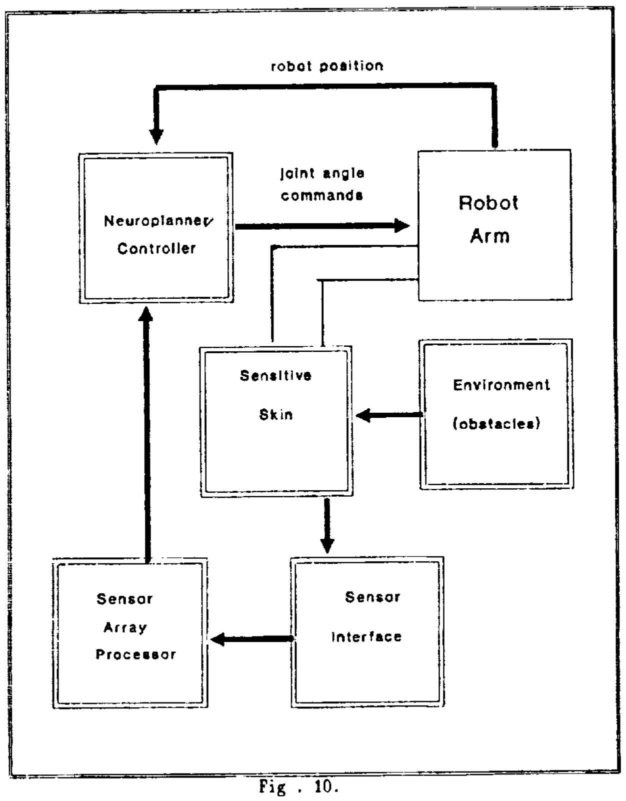


Fig . 10.

expected output

Network Output

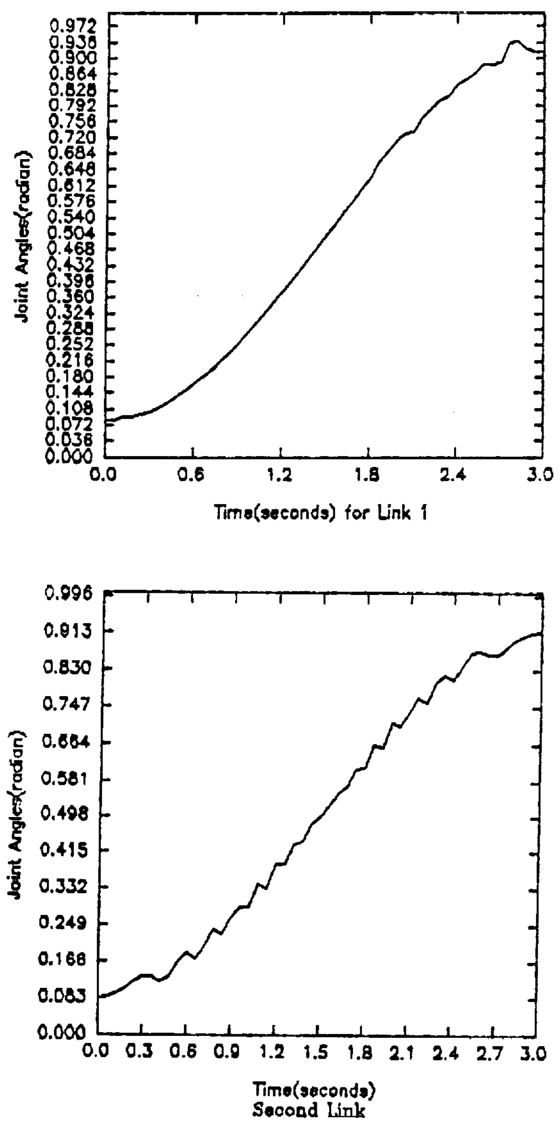


Fig. 8a.

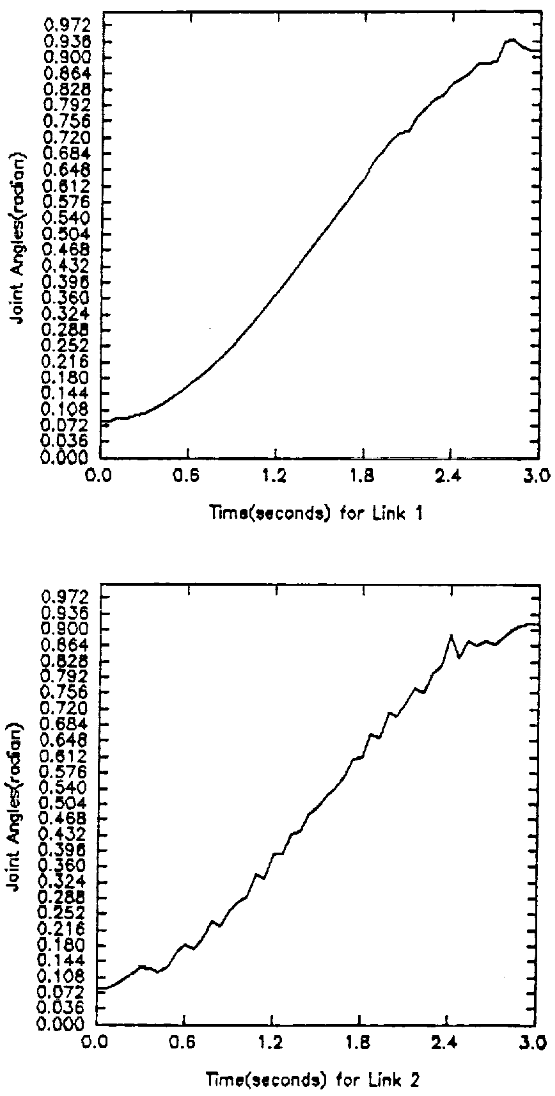


Fig. 8.b