Presented at the IEEE/ASME AIM'97, Tokyo, Japan
June 16-20, 1997

VISION-GUIDED DYNAMIC PART PICKUP LEARNING ALGORITHM

Kok-Meng Lee and James Downs
The George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0405

ABSTRACT
This paper addresses the problem of picking up moving objects in pseudo-random motion. Specifically, we present a recursive learning technique which utilizes two neural networks to model complex motion of such objects as well as the robot's response time. Using the trained neural networks, a dynamic part pickup control strategy has been developed and implemented on a six degrees-of-freedom (DOF) industrial robot. The performance of this hand-eye coordination control strategy has been evaluated experimentally in real time for picking up moving objects on vibratory feeder.

1. INTRODUCTION

There are many industrial tasks in which visual servoing is required to provide sophisticated guidance information for either the tracking or grasping of objects in motion. In the case of grasping, there will come a point at which the view of the object being tracked will become obscured by the actuating mechanism itself, whether the vision system is mounted on or off the actuation mechanism. Thus, it becomes necessary to predict the future state of an object that is desired to be grasped.

Some types of motion such as moving parts on a vibratory feeder where motion is pseudo-random, the problem of predicting the future state of an object based on its initial state is rather challenging. Vibratory feeders are commonly used to separate industrial parts prior to robot handling. Lee and Qian [1] formulated the problem in the context of Prey Capture with the robot as a "pursuer" and a moving object as a passive "prey." They demonstrated the use of neural network to estimate a moving part's position. The concept of prey capture has only been explored in robotics in recent years. Sharma and Aloimonos [2] investigated the problem of a mobile robot tracking a moving object. They modeled the motion control as a differential game [3] of pursuit and evasion, and used a camera on a mobile robot to obtain the information about a moving target from a sequence of its images. However, their emphasis was on the use of qualitative information for motion control. Detailed control strategy and implementation problems were not discussed. The problem of docking mobile robots using a bat-like sonar system was considered by Kuc and Barshan [4] in the context of prey capture in two dimensions. By constraining the prey motion to be linear, the lower bound for the capture time was determined from game theory. However, complete information about the prey was assumed.

We present here a recursive learning algorithm to guide an industrial robot to pick up moving objects from the surface of a vibratory feeder. The contributions of the paper are briefly summarized as follows: (1) The algorithm requires only an initial location and orientation of the moving object to predict the state of the object at the point of pickup. This overcomes a common vision problem; that is, the view of the object at the point of pickup often becomes obscured by the gripper itself. (2) For a given initial state of the object, only the robot response time is needed to command the robot to execute the pickup task, which can be determined off-line by training for a specified velocity. Thus, the technique introduced in this article can be readily implemented on an off-the-shelf industrial robot without special modification of its controller which is treated as a "black box". (3) The concept feasibility of the dynamic part pickup system has been experimentally demonstrated and evaluated on an industrial robot and a vibratory feeder in real time. The results provide significant insights to the other similar applications such as catching and hanging live birds on shackle line for poultry processing.

The remainder of this article is organized as follows: An overview of the dynamic part pickup controlled system is presented in Section 2, followed by a description of the recursive learning algorithm in Section 3. The experimental results and discussions are detailed in Section 4. Finally, conclusions are summarized in Section 5.

2. SYSTEM SETUP AND OVERVIEW

Figure 1 shows a schematic of a dynamic part pickup system which consists of a vibratory feeder, an industrial robot, and a vision system mounted on the robot's end-effector. Parts to be picked up circulate continuously on the vibratory surface of a Dyna-Slide vibratory feeder which generates pseudo-random type motion to singulate the moving objects. The robot used in this investigation is a Cincinnati Milacron T3-786 industrial robot. This is a six DOF electrically driven, computer-controlled, articulated robot. Each of the six axes is servo controlled by the Cincinnati ACRAMATIC Version 4.0 Control. A user-programmable flexible integrated vision

system (FIVS) developed at Georgia Tech [5] is used to determine the initial object's pose. Customized machine vision algorithms can be written, compiled, and downloaded into the FIVS' EEPROM for real-time execution. The camera and the kinematic relationship between the imaging sensor and the gripper were calibrated using Tsai and Lens calibration algorithms [6] [7] respectively. The system is controlled by a dynamic part pickup controller (DPPC) implemented on an Intel 486-33MHz computer which communicates with the FIVS and the robot controller through serial communication.

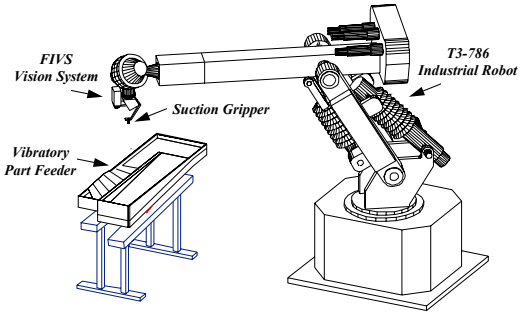


Fig. 1 Schematic of a typical robotic pick-up system

A typical cycle of the part pick-up operation is as follows: The FIVS is positioned at a pre-specified location above the vibratory feeder such that the optical axis of the camera is perpendicular to the vibratory surface. Once the FIVS detects an object in its field-of-view (FOV), it will compute the initial pose of the object, namely the location and orientation, and feedback this visual information to the DPPC. The DPPC, in turn, determines a rendezvous point in time and space, and computes the pose of the end-effector for the pickup operation to occur.

Figure 2 shows the relationship between a moving object and the suction gripper. Consider that the moving object is detected at time t_i and the robot is commanded to pick the object up at time $t_{i+1} = t_i + \Delta t$ where Δt is the time interval between sensing and pickup. In order to pickup the moving object, the robot controller must compute the joint rotations to move its end-effector from its initial viewing position to the grasping position. Clearly, there is a finite time interval required to complete the pick up task in response to the command from the DPPC. This time interval, denoted here as the robot's response time t_r, is a function of joint velocities.

To ensure a successful pick up, the following control objective must be matched:

$$\boldsymbol{p}_{i+1} - \boldsymbol{r}_{i+1} \le \boldsymbol{\delta} \qquad (1)$$

where

$$\boldsymbol{p}_{i+1} = \boldsymbol{f}_1(\boldsymbol{p}_i, \Delta t); \qquad (2)$$

$$\boldsymbol{r}_{i+1} = \boldsymbol{f}_2(\boldsymbol{r}_i, t_r); \qquad (3)$$

and where $\boldsymbol{\delta}$ is the specified tolerance vector of the suction gripper for the pick up operation; $\boldsymbol{r}_i$ and $\boldsymbol{r}_{i+1}$ are the initial and the subsequent poses of the robot's end-effector; and $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$ are the position and orientation of the object with respect to the world coordinate system at time t_i and t_{i+1} respectively. All the objects are considered as rigid bodies in three dimensional space, represented by a dual vector notation (translation and orientation vectors). The part pick-up system is subjected to the following constraint:

$$\Delta t \ge t_r + t_v \qquad (4)$$

where t_v is the time required for the vision system to process the image, determine the location and orientation of the moving object at time t_{i+1}, and execute the pick up task. If Equation (4) is not satisfied, the robot will be unable to catch up with the moving object.

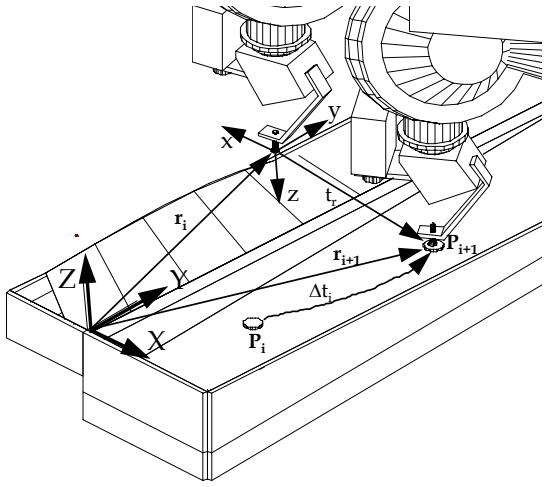


Fig. 2 Relationship between moving object and gripper

3. PART PICKUP SYSTEM MODELING

The dynamic part pickup system is modeled by means of a recursive learning algorithm as shown in Fig. 3.

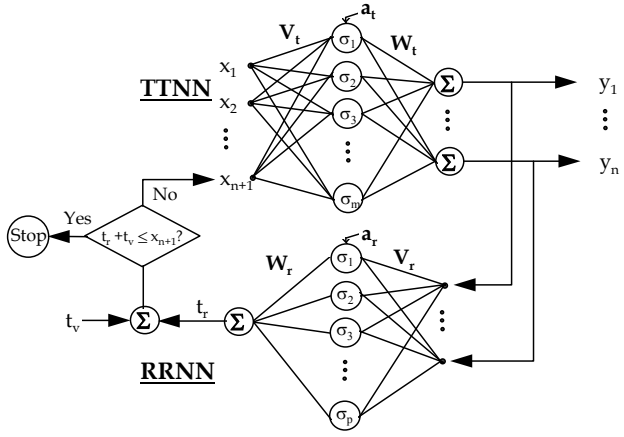


Fig. 3 Recursive part pickup learning algorithm

This algorithm consists of two networks; namely, the target trajectory neural network (TTNN) and the robot response neural network (RRNN). The two networks are trained off-line separately using experimentally collected input-output data pairs (or the training data). The parameters of each of the networks are adapted using the back-propagation network training algorithm outlined in the Appendix.

Robot Response Modeling

As the time interval Δt in Equation (2) is a function of the current state of the target itself, it is therefore not known in advance. In order to estimate the robot response time t_r, the following inverse mapping relationship *for a given robot velocity* is defined in Equation (5).

$$t_r = f_2^{-1}(\boldsymbol{y}; \mathbf{W_r}, \mathbf{V_r}, \bar{\theta}_r) \tag{5}$$

where $\mathbf{W_r}$, $\mathbf{V_r}$, and $\bar{\theta}_r$ are the weight matrices and vector of RRNN to be determined using the back-propagation learning algorithm (see Appendix) and $\boldsymbol{y} = \boldsymbol{r}_{i+1}$. Since initial viewing position is fixed, only the final pose is required as an input to the RRNN. To obtain the training data experimentally, the robot is taught to execute successive pick up movements over a fine grid on the plane of the vibratory feeder and the time taken to execute each of the pickup tasks at a specified robot velocity is stored. The training data are then used to determine the mapping relationship for Equation (5).

Trajectory Learning

The mapping relationship for Equations (6) is determined by means of the TTNN as shown in Fig. 3, where the first n elements of the vector $\boldsymbol{x}$ represent the components of the object pose vector $\boldsymbol{p}_i$ and the element x_{n+1} refers to the time interval Δt. The output vector $\boldsymbol{y}$ from the TTNN corresponds to the pose vector $\boldsymbol{p}_{i+1}$.

$$\boldsymbol{y} = \boldsymbol{f}_1(\boldsymbol{x}, \mathbf{W_t}, \mathbf{V_t}, \bar{\theta}_t) \tag{6}$$

where $\mathbf{W_t}$, $\mathbf{V_t}$ and $\bar{\theta}_t$, are the weight matrices and vector of TTNN. To obtain the training data for adapting the TTNN networks' parameters, a sample part of interest is allowed to circulate on the vibratory feeder. Trajectories of the part over a representative region on which the part is to be picked up are obtained using the FIVS. The input and output training data pairs, $\boldsymbol{A}_{tk}\{x_{k1}, x_{k2}, \ldots, x_{k(n+1)}\}$ and $\boldsymbol{B}_{tk}\{y_{k1}, y_{k2}, \ldots, y_{kn}\}$ respectively, are then used to determine the network parameters, $\mathbf{W_t}$, $\mathbf{V_t}$, and $\bar{\theta}_t$.

Part Pickup Learning Algorithm

Using the trained RRNN and TTNN, the recursive learning begins with an initial guessed $\Delta t_{i=0}$ with the detected target's $\boldsymbol{p}_i$. The predicted end-effector's pose $\boldsymbol{p}_{i+1}$ from the TTNN is then fed into the RRNN which, in turn, computes the approximate time required for the robot to execute the pickup task. If Equation (4) is not satisfied, a new pose will be re-computed using the original detected target's pose and the new time interval or

$$x_{n+1}^{new} = t_r + t_v . \tag{7}$$

The network is performed recursively until the following condition is matched:

$$t_r + t_v \le x_{n+1}^{current} \tag{8}$$

Equation (8) represents the condition upon which the robot is commanded by the DPPC to pickup the moving object.

4. EXPERIMENTAL INVESTIGATION

Two different types of industrial parts; a circular disk with a diameter of 34mm and a 3/8-inch machine screw of 2 inches long, were used to exemplify the dynamic part pick up. The significant difference between the two shapes are in the number of states required to approximate each for the pickup. For the circular disk, two state variables, the (x, y) Cartesian coordinates, are required for grasping. In the case of an elongated shape, the orientation angle ϕ in addition to its (x, y) Cartesian coordinates must be approximated.

4.1 Network Training Considerations

For the purpose of performing the required mappings, a single hidden-layer feedforward sigmoidal architecture was designed (Hornik and White, 1989). Attempts were made to use the minimal number of processing elements (neurons) necessary to represent the respective systems. Table 1 summarizes the number of neurons used, the input and output state variables in each network training.

Table 1 Summary of training configurations

	Neurons	Input	Output	Data
RRNN	5	x, y	t_r	387
Circular disk	30	x, y Δt	x, y	500
Machine screw	65	x, y, ϕ, Δt	x, y, ϕ	500

Noisy or undesirable data sets, termed outliers, can not only slow the convergence of a network, but can also result in the network most closely approximating this noisy data as opposed to that which is desired [8]. For these reasons, target trajectory data which contrasted highly with that of the norm was filtered out prior to training.

Network parameters are originally set at random values. The learning rate η was allowed to adapt during the training procedure based on the change of sum squared error over one complete run through the data sets (one epoch). That is,

$$\eta_{k+1} = \begin{cases} \eta_k + \varepsilon & if\ \Delta E \le 0 \\ \eta_k - \varepsilon & if\ \Delta E > 0 \end{cases} \tag{10}$$

where ε is a small positive increment and ΔE is the difference between the current and previous cost function E defined in Equation (A.1).

As oulined in the Appendix, the back-propagation training algorithm performs the weight adjustments by moving along

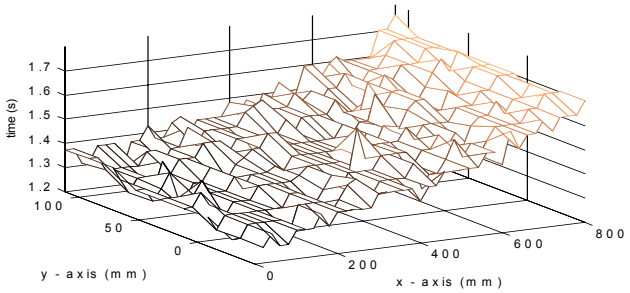
the cost function in the opposite direction of the gradient to a "local" minimum in the network's parameter space. In an attempt to locate the global error minimum, the adjustment of the network's parameters was performed iteratively using numerous random sets of initial conditions.

4.2 Experimental Results Of Network Training

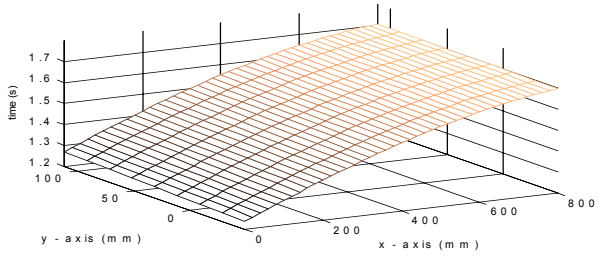
Three separate network training were performed; namely, robot response time modeling, trajectory of a circular disk-like object; and trajectory of a machine screw.

Robot Response

Figures 4(a) and 4(b) show the actual and RRNN trained response time data of the robot to reach a fine grid of positional points on the vibratory surface form its viewing position (700mm above the surface). The robot response time corresponds to the time increment between the command issued to the robot controller and the acknowledgment received from the robot controller. The maximum percent error in the robot response time model across the entire trained region is 11.1%.



(a) Robot Response Time to pick up object at (*x,y*)



(b) RRNN approximation of Robot Response Time

Fig. 4 Experimental and Simulated Robot Response Time

As shown in Fig. 4, the robot response time is a linear function of x and y:

$$x = v_x t + x_0 \quad (11a)$$

$$y = v_y t + y_0 \quad (11b)$$

where v_x = 1.555 m/s; v_y = 0.325 m/s; x_0 = -1.435 m; and y_0 = -0.429 m. The nominal robot velocity was 0.635m/s.

It was observed that the wrist typically reaches its desired orientation by the time the robot moves the wrist into the desired position. Thus, the same network parameters were used for the picking up the two different objects since the limiting factor contributing to the robot's response time has been the Cartesian translation undergone by the robot.

Trajectory of Circular Disk

The TTNN approximated trajectories are shown in Figure 5. Each vector in this plot represents a 0.2 second trajectory followed by the target. It is interesting to note that the neural network actually learned to model somewhat of a stationary point where these disk-like parts were observed to frequently slow down, and sometimes get temporarily 'stuck' on the feeder.

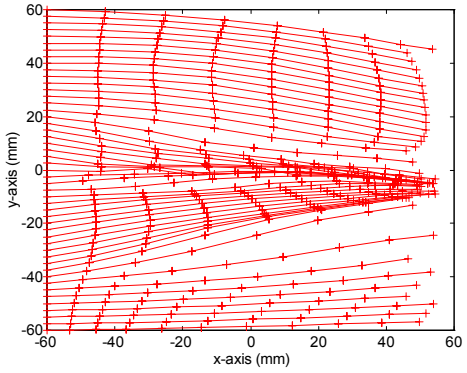


Fig. 5 TTNN approximated trajectories (Circular disk)

Figure 6 compares the TTNN approximated trajectories against the real data not included in the training set over a broad range of initial starting conditions. As shown in the Figure 6, the network model fairly accurately predicts these complex paths for short time intervals.

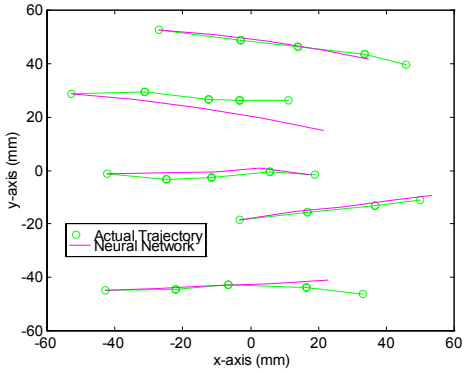


Fig. 6 Comparison between predicted and actual trajectories

Table 2 shows the TTNN's success rate of predicting the final state of 100 moving parts after various time intervals. A successful estimation was taken to be one in which the final position difference in modeled and actual trajectories was less than ±8 mm in both *x* and *y* directions. As shown in Table 2, it is desirable in implementation to minimize Δt, the

time interval between sensing and pickup since the error propagates as the time interval increases.

Table 2 Success rate of TTNN's on trajectory prediction

Time Interval (s)	Success Rate
0.3	90.2 %
0.6	86.1 %
0.9	79.8 %
1.2	75.4 %
1.5	68.1 %

Trajectory of Machine Screw

Figure 7 show approximated trajectories of the moving screw for three initial orientation angles; 90°, 0°, 45°and -45°. In each plot, the increments are of constant time intervals of 0.3 seconds each showing the evolution of three state variables over time. It is interesting to note the inter-relation between initial orientation angle and final trajectory of the part. There is also a noticable tendency for the screw's orientation to move towards 0 deg as time evolves, due to the screw's head dragging behind the rest of the screw.

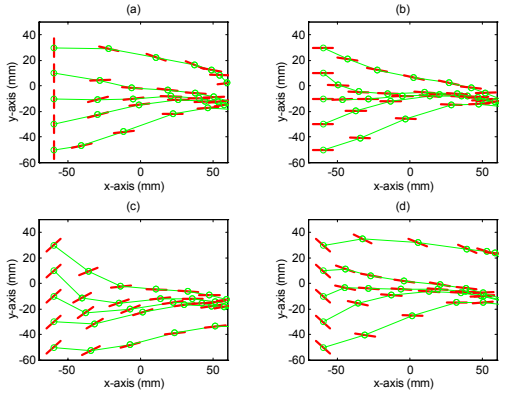


Fig. 7 TTNN approximated trajectories (Machine screw)

4.3 Experimental Results Of Dynamic Part Pickup

In an attempt to improve the success rate for picking up the circular disks, the target position over elapsed time and the robot response time to reach each position are graphed in Figure 8 and the time distribution among the activities in each control loop cycle is summarized in Table 3. As shown in Figure 8, a representative target of interest to be picked up is located at initial location (-60, -40) in mm. The rendezvous point for the target is calculated to occur at the location (-18.7, -30.3) after 1.23 seconds. Convergence was typically achieved after 6 iterations through the DPPC.

Table 4 summarizes the success rate of the dynamic part pickup strategy in real-time using three different viewing configurations. The first and second configurations differ in initial viewing heights; namely, at the vertical distance of 700mm and 500mm respectively. A third control strategy was implemented in which two network approximations are made in succession. The viewing height begins at 700mm, moves to 500mm while estimating part trajectory, and finally grasps from 500mm. The results summarized in Table 3 are based on 100 pickup attempts, indicating viewing configuration used, the corresponding FOV, average time interval for pick up, and success rate associated with each method. As shown in Figure 8 and Table 3, since a dominant time-delay of 1.2 seconds is needed for the robot to execute a single part, the time interval between sensing and pickup, Δt, becomes relatively insensitive to change in viewing height.

Table 3 Control loop cycle-time distribution

Process	**Time (ms)**	**Time (s)**
Integration	60	t_v=0.227
Image Processing	38	
Serial Communication	74	
486 Processing	55	
Robot Response	1200	t_r=1.2
Total Average Cycle Time	**1427**	***Δt=1.427***

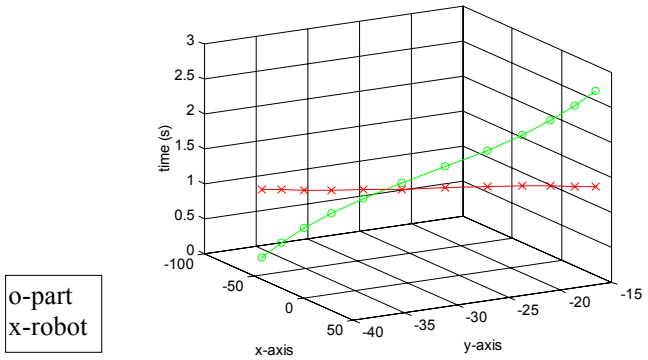


Fig. 8 Part trajectory and Robot Response Time

Table 4 Success rate for three different viewing methods

Viewing method	FOV(mm)	Average δt	Success
700mm	(135x135)	1.43s	61%
500mm	(96x96)	1.36s	68%
Successive	$135^2 \Rightarrow 96^2$	2.72s	66%

As an additional measure of performance, the average cycle time to pick up a single part was compared with one using a part-specific fixture. In the case of part specific setup, the system must only verify the part's presence visually, and then pick it up at a known position (chosen to be 500mm away). This average cycle time was approximately 4.52s. The DPPC's pickup strategy had an cycle time of 4.88s based on a viewing method of 500mm, about 7.4% longer per cycle than the approach with a part-specific fixture.

6. CONCLUSIONS

A technique to pickup objects moving in pseudo-random motion on vibratory feeder has been presented. Specifically, a dynamic part pickup controlled system has been developed and implemented on an industrial robot. The system uses a position-based vision system to determine the initial state of

the object and a recursive learning algorithm to predict the pickup state. The latter utilizes two neural networks to model the trajectories associated with individual parts on the feeder and the robot response time at a specific velocity. The performance of this hand/eye coordination control strategy has been evaluated experimentally on picking up objects moving on an industrial vibratory feeder in real-time. As demonstrated experimentally, the vision-guided dynamic part-pickup system can provide an effective means to predict the future state of the moving object when the view of this object becomes temporarily blocked or obscured by the actuating mechanism itself.

REFERENCES

1. Lee, K.-M. and Qian, Y., "Intelligent Vision-Based Part Feeding on Dynamic Pursuit of Moving Objects," SPIE Photonics East, 23-26 October 1995, Philadelphia, PA, pp 172-183.
2. Sharma, R. and Aloimonos, J., "Target Pursuit or Prey Catching Using Qualitative Visual Data," *Pro. of AAAI-90 Workshop on Qualitative Vision*, pp. 195-198, July 1990.
3. Issacs, R., "*Differential Games*," John Wiley & Sons, Inc., New York, 1965.
4. Kuc R. and Barshan, B., "Bat-like Sonar for Guiding Mobile Robots," *IEEE Control Systems Mag.*, pp. 4-12, August 1992.
5. Lee, K.-M. and Blenis, R.,"Design Concept and Prototype Development of a Flexible Integrated Vision System," *J. of Robotic Systems*, Vol. 11, No. 5, pp. 387-398, 1994
6. Tsai, R. Y., "A Versatile Camera Calibration Technique for High Accuracy 3D Machine Vision Metrology using Off-the-shelf TV Camera and Lenses," *IEEE Trans. on Robotics and Automation*, Vol. RA-3, No. 4, pp. 323-344, August, 1987.
7. Tsai, R.Y., and Lenz, R. K., "A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration," *IEEE Trans. on Robotics and Automation*, Vol. 5, No. 3, June 1989, pp. 345-358.
8. Dumuth, H. and Beale, M., *Neural Network Toolbox for Use with MATLAB*, The Math Works, Inc., 1993.

ACKNOWLEDGMENTS

This research was partially supported by the National Science Foundation EEC-9420492 and by the Woodruff Faculty Fellow.

APPENDIX

Let the cost function of a typical neural network (as shown in Figure A.1) be

$$E = \frac{1}{2}\sum_{j=1}^{n}(b_{kj} - z_j)^2 \qquad \text{(A.1)}$$

where b_{kj}, is the j[th] element of the k^{th} output data set $\boldsymbol{B_k}\{b_{k1}, b_{k2}, ..., b_{kn}\}$ corresponding to the input data set $\boldsymbol{A_k}\{a_{k1}, a_{k2}, ..., a_{k(n+1)}\}$ in training data pairs; and z_j is the j^{th} output given by

$$z_j = \sum_{i=1}^{n} w_{ij}\sigma_i \qquad \text{(A.2)}$$

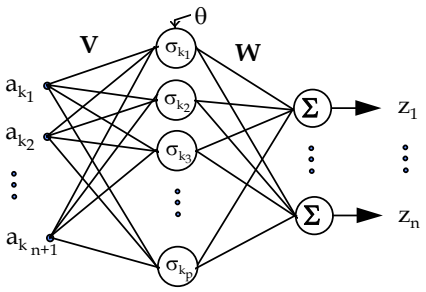


Figure A.1. Target Trajectory Neural Network

The hidden processing elements where $\boldsymbol{\sigma_k}\{\sigma_{k1}, \sigma_{k2}, ..., \sigma_{kp}\}$ is computed from Equation (A.3):

$$\sigma_i = f(\sum_{h=1}^{n} a_{kh} v_{hi} + \theta_i) \qquad \text{(A.3)}$$

where the hidden-layer processing element function is

$$f(\gamma) = \frac{1}{1+e^{-\gamma}} \quad . \qquad \text{(A.4)}$$

The cost function is minimized in the opposite direction of the gradient:

$$w_{ij}^{new} = w_{ij}^{current} - \eta_1 \frac{\partial E}{\partial w_{ij}} \qquad \text{(A.5a)}$$

$$v_{ij}^{new} = v_{ij}^{current} - \eta_2 \frac{\partial E}{\partial v_{ij}} \qquad \text{(A.5b)}$$

$$\theta_i^{new} = \theta_i^{current} - \eta_3 \frac{\partial E}{\partial \theta_i}. \qquad \text{(A.5c)}$$

where $\eta_{i\ (i=1,2,3)}$ are positive-valued constants (learning rate) that regulate the amount of adjustments made with each gradient move,

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial z_i}\frac{\partial z_i}{\partial w_{ij}} = -e_k \sigma_j \qquad \text{(A.6a)}$$

$$\frac{\partial E}{\partial v_{ij}} = \frac{\partial E}{\partial z_i}\frac{\partial z_i}{\partial \sigma_i}\frac{\partial \sigma_i}{\partial v_{ij}} = -\sum_{k=1}^{n} e_k w_{ki}\frac{\partial \sigma_i}{\partial v_{ij}} \qquad \text{(A.6b)}$$

$$\frac{\partial E}{\partial \theta_i} = \frac{\partial E}{\partial z_i}\frac{\partial z_i}{\partial \sigma_i}\frac{\partial \sigma_i}{\partial \theta_i} = -\sum_{k=1}^{n} e_k w_{ki}\frac{\partial \sigma_i}{\partial \theta_i} \qquad \text{(A.6c)}$$

where $e_k = b_{kj} - z_j$.