Intelligent Vision-Based Part-Feeding on Dynamic Pursuit of Moving Objects

Kok-Meng Lee
The George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology,
Atlanta, GA 30332-0405
kokmeng.lee@me.gatech.edu

Yifei Qian
Lucent Technologies, Inc.,
Bell Laboratories,
Preform Technology,
2000 Northeast Expressway,
Room G010/GAE870,
Norcross, GA 30071
yfqian@lucent.com

The paper addresses the problem of picking up moving objects from a vibratory feeder with robotic hand-eye coordination. Since the dynamics of moving targets on the vibratory feeder are highly nonlinear and often impractical to model accurately, the problem has been formulated in the context of Prey Capture with the robot as a "pursuer" and a moving object as a passive "prey." A vision-based intelligent controller has been developed and implemented in the Factory-of-the-Future Kitting Cell at Georgia Tech. The controller consists of two parts: The first part, based on the principle of fuzzy logic, guides the robot to search for an object of interest and then pursue it. The second part, an open-loop estimator built upon back-propagation neural network, predicts the target's position at which the robot executes the pick-up task. The feasibility of the concept and the control strategies were verified by two experiments. The first experiment evaluated the performance of the fuzzy logic controller for following the highly nonlinear motion of a moving object. The second experiment demonstrated that the neural network provides a fairly accurate location estimation for part pick up once the target is within the vicinity of the gripper.

1 Introduction

Dedicated fixtures are often used with vibratory feeders to locate and orient small parts for subsequent robot handling. For small volume production, the success of a specially designed feeder is often achieved at the expense of operational cost and flexibility [1]. An alternative method is to eliminate the object-dependent fixture by having parts circulate continuously on the vibratory feeder and to use a vision system to guide the robot to pick up moving parts on the vibratory feeder. The approach has potential applications for flexible manufacturing and/or demanufacturing of short production runs where a large variety of product sizes and component types are encountered. We present here a method to kit moving parts from vibratory feeders into a loosely palletized waffle pack by using hand-eye coordination. The problem of picking up moving objects from the vibratory feeder is formulated in the context of Prey Capture with the robot as a "pursuer" and a moving object as a passive "prey."

To maintain sufficient position accuracy in transport would often require excessive packaging costs and result in a lack of flexibility. It is desired to have parts transported in separate, regularly-spaced locations in totes, pallets, or kits without maintaining sufficient dimensional accuracy to permit loading or assembly by a totally pre-programmed robot. Lee [2] developed a low cost vision-based method to locate parts in regularly-spaced totes for machine loading and assembly, which eliminates the need to maintain dimensional accuracy in transporting parts between work cells. In this case, parts to be assembled or loaded on machines are only required to be kitted in separate, regularly spaced pallets. His efforts have further led to the development of a *f*lexible *i*ntegrated *v*ision *s*ystem (FIVS) at Georgia Tech [3], which provides a means to process images without the limitation of the TV video standard established in the 1950's. Since FIVS provides a means to process a partial image without having to pre-store the complete image, it allows high-speed image processing in real-time. The creation of FIVS has motivated us to examine the feasibility of developing an intelligent vision-guided controlled system to pick up moving parts on a vibratory feeder in real-time. In this paper, we discuss the use of FIVS as a real-time visual feedback element to kit moving parts from vibratory feeders by an industrial robot.

The concept of prey capture has only been explored in robotics in recent years. Sharma and Aloimonos [4] investigated the problem of a mobile robot tracking a moving object. They modeled the motion control as a differential game [5] 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 [6] 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 investigate the use of linguistic rules based on "rules-of-thumb" experiences and engineering judgments to specify control laws, and apply experimental or heuristic knowledge as a basis for logical inference. Such a formulation mimics the function and capability of a natural being to pursue its prey. This approach does not require an accurate description of the dynamics of the pursuit process and the motion of moving objects, which often are highly nonlinear and impractical to model analytically and reasonably accurately. In addition, it does not require the goals and constraints of the system to be quantified by a single numerical value, provided that the "pursuer" can approach the object within the reach of the robot gripper.

The remainder of this article is organized as follows: An overview on the system design is presented in Section 2. The problem of dynamic pursuit of moving objects is then defined and an alternative approach, based on fuzzy logic theory and neural network technique, is developed in Sections 3 and 4. Next, the experimental results are presented in Section 5. Finally, conclusions are drawn, and the advantages of the design are summarized.

2 Overview

The vision-based part-pickup system that does not require object-dependent tooling consists of three major subsystems;

Contributed by the Manufacturing Engineering Division for publication in the JOURNAL OF MANUFACTURING SCIENCE AND ENGINEERING. Manuscript received March 1996; revised June 1997. Associate Technical Editor: C. H. Meng.

Copyright © 1998 by ASME

Downloaded 16 Mar 2010 to 130.207.50.192. Redistribution subject to ASME license or copyright; see http://www.asme.org/terms/Terms_Use.cfm

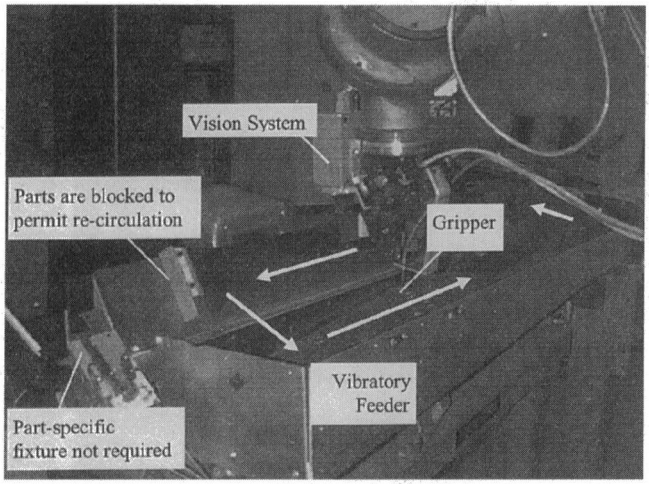

Fig. 1 Schematic illustrating the intelligent vision-based feeder

namely, a robot, a vision system, and a vibratory feeder that separates objects circulating on its surface. The vision system may either be fixed in space or attached onto the robot gripper. The latter setup provides a higher flexibility and is used in this design. As shown in Fig. 1, parts to be picked up circulate continuously on the vibratory surface. The approach eliminates the need of a part-specific fixture which traps a part in order to allow pick up by a pre-programmed robot. A typical cycle of the dynamic pursuit is as follows: The vision system is positioned at a prespecified location above the vibratory feeder such that the optical axis of the camera is perpendicular to the vibratory surface. The vision system would locate a moving object on a vibratory feeder once it appears in its field of view (FOV). The robot is then commanded to pick up the moving object. Alternatively, the robot is commanded to "pursue" the object of interest with an attempt to approach its vicinity as quickly as possible. As soon as the object is within the reach of the gripper, the robot will pick up the object before it "escapes."

Mathematically, the problem can be expressed as follows:

$$\|\boldsymbol{x}_o - \boldsymbol{x}_r\| \leq \|\boldsymbol{x}_f\|, \tag{1}$$

where $\boldsymbol{x}_o$ and $\boldsymbol{x}_r$ are the state vectors of the moving object and the robot respectively; $\boldsymbol{x}_f$ is the desired final state difference between the robot and the moving object; and the notation $\|*\|$ denotes the norm of vectors:

$$\|x\| = (x_1^2 + x_2^2 + \ldots + x_n^2)^{1/2}.$$

In Eq. (1), the state vector is defined as follows:

$$\boldsymbol{x}_{(\cdot)} = [\mathbf{p}_{(\cdot)}\ \mathbf{v}_{(\cdot)}\ \theta_{(\cdot)}]^T$$

where $\mathbf{p}_{(\cdot)}$, $\mathbf{v}_{(\cdot)}$, and $\theta_{(\cdot)}$ are the position, velocity and orientation vectors of o and r respectively, and where $\boldsymbol{x} \subset R^n$, and $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$. Equation (1) represents the condition the pursuit process must satisfy in order to pick up a moving object of interest. The general constraints imposed by the robot, the object, and the environment are as follows:

(1) The speed and acceleration of the robot are limited:

$$\|\mathbf{v}_r\| \leq v_{\max}, \tag{2a}$$

$$\|\boldsymbol{a}_r\| \leq a_{\max}, \tag{2b}$$

where $v_{\max}$ and $a_{\max}$ denote the maximum values of the speed and acceleration of the robot respectively.

(2) The robot can only move without hitting any obstacle within a certain workspace, whereas moving objects will not venture into a certain area. This can be expressed as

$$\boldsymbol{p}_r \subseteq \boldsymbol{p}_{r0}, \tag{3a}$$

$$\boldsymbol{p}_o \subseteq \boldsymbol{p}_{o0}, \tag{3b}$$

where $\boldsymbol{p}_{r0}$ denotes the range of $\boldsymbol{p}_r$ (or the difference between the robot workspace and the solid obstacles within the workspace), and $\boldsymbol{p}_{o0}$ the range of $\boldsymbol{p}_o$ (or the area where objects can move around).

(3) The robot must pick up the object within a specified time or within a specified range of pursuit in certain area, namely:

$$\sum_{i=1}^{n} t_i \leq T, \tag{4}$$

$$\|\boldsymbol{p}_o\| \leq p_{o0}, \tag{5}$$

where n denotes the number of pursuing steps, t_i the time the pursuing step i takes, and T the upper bound on the time spent on pursuit process.

The challenges in vision-guided dynamic pursuit of moving objects on vibratory feeders are as follows: (1) As will be shown experimentally in Section 5, the objects' dynamics on the vibratory feeder are position and object dependent and often impractical to model. (2) Often, the view is obscured or blocked by the robot gripper, particularly at the instant prior to pick up. (3) Since the system requires a finite amount of processing time, there is a significant delay between sensing and pick up.

Conceptually, the pursuit process can be broadly divided into two sub-control problems once a moving object of interest is identified. The first is to pursuit the moving object found. Linguistic rule-based control strategies based on the fuzzy logic theory [7] are developed to search for a moving object of interest and then guide the robot to pursue it. The second is to locate the object for the robot to pick up. Here, a control strategy based on the neural network technique is developed to estimate the object position such that the robot will catch it before it "escapes." In general, a typical industrial robot has its own controller. In this context, the neural network estimator and the robot controller are essentially working at the execution level while the fuzzy logic controller works at a higher level or the process controller level which mimics the mind of a human operator. It is expected that in some instances, the neural network itself may perform alone to accomplish the task.

3 Fuzzy Logic Control for Object Pursuit

A fuzzy rule-based controller is designed for the phase of pursuing an object. The purpose of the controller, as shown in Fig. 2(*a*), is to apply the appropriate control action Δv_r to the robot and direct it to the vicinity of the object based on the feedback of the vision system. A fuzzy logic controller, as shown in Fig. 2(*b*), consists of a fuzzifier, an inference engine made up of a data base and a rule base, and a defuzzifier. Since the underlying principle of fuzzy logic control can be found in other literature [8], only those issues related to the fuzzy controller design for dynamic pursuit are discussed in the following.

(1) Input and Output Variables. A "pursuer" determines its action based on how far a "prey" is away from it, how fast the "prey" is fleeing, and in which direction the "prey" is going to flee. Therefore, the displacement and velocity differences are chosen as the primary motion parameters in describing the relationship between the robot and the object, and are used as the input variables to the controller. Likewise the change to the velocity of the "pursuer" is to adjust the motion of the "pursuer," which determines if the "pursuer" should accelerate or decelerate in a certain direction, so that it is chosen as the output variable.

(2) Description and Transformation of Variables. The three input and output fuzzy variables are defined over the fixed universes of discourse. The library of fuzzy sets to describe the

Downloaded 16 Mar 2010 to 130.207.50.192. Redistribution subject to ASME license or copyright; see http://www.asme.org/terms/Terms_Use.cfm

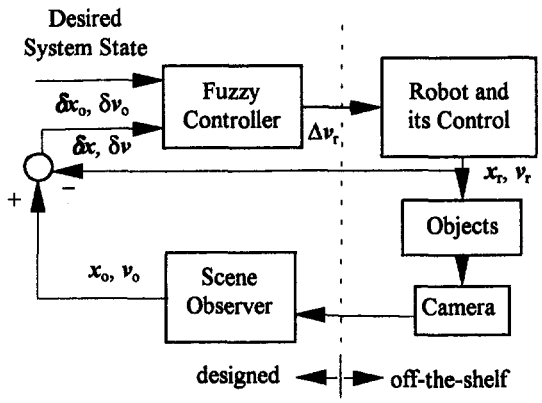

Fig. 2(*a*) Function block of object pursuit operation

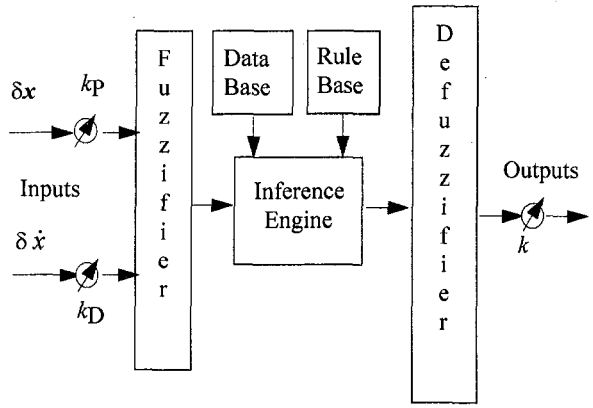

Fig. 2(*b*) Schematic of a fuzzy logic controller

fuzzy states of the three variables is defined as (NL, NM, NS, ZE, PS, PM, PL), where N means negative, P positive, L large, M medium, S small, and ZE zero. The characteristics of these fuzzy sets are described by their membership functions over the universe of discourse of each variable. The membership functions are defined in Fig. 3. The lengths of the upper and lower bases provide design parameters to be tuned for satisfactory performance of the controller. All the information is stored in the data base (Fig. 2) which is a part of the Knowledge Base.

The actual inputs to the controller are real numbers defined over their physical variation range. They are scaled and normalized first to the fixed universe of discourse of the corresponding fuzzy variable. They are then transformed into fuzzy sets in order to describe the state of the physical input variables in linguistic terms. The reverse process is performed by the defuzzifier to generate the physical output from the results of the inference engine, a fuzzy set. The centroid-of-area method is used.

(3) Fuzzy Inference Engine for Control Action. The control rules which relate the output variables to the input variables are derived intuitively based on the understanding of the process behavior and on human experience. The basic idea here is to find the action of the "pursuer" in order to reduce the position and speed differences between the robot gripper and the moving object to a level that is desired or specified. The intuitive rules are as follows: When the position difference is large, the emphasis is to reduce the position difference, so that the strongest control action is used by a "pursuer" to effectively get itself close to its "prey" quickly. However, when the "pursuer" approaches the final desired state, the emphasis is to maintain the speed as close as possible to that of its "prey" while keeping the target within the specified vicinity of the robot gripper.

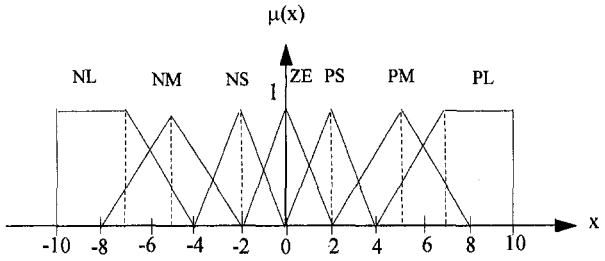

Fig. 3 Definition of the membership functions

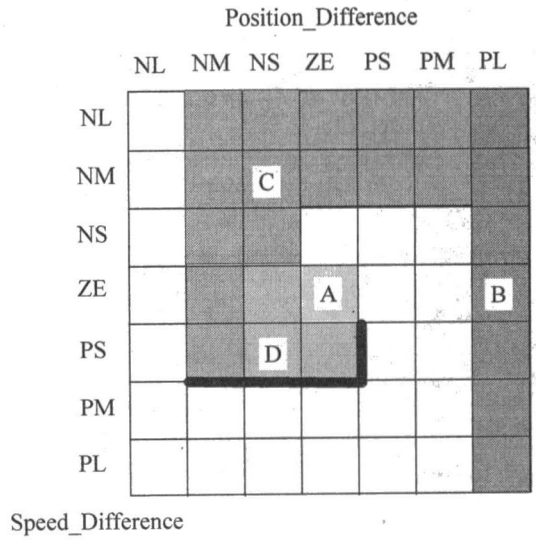

Fig. 4(*a*) Desired control action

Position_Difference

Speed_Difference	NL	NM	NS	ZE	PS	PM	PL
NL	NL	NL	NL	NL	NL	NM	ZE
NM	NL	NL	NL	NL	NL	ZE	PM
NS	NL	NL	NL	NS	ZE	PM	PL
ZE	NL	NL	NM	ZE	PM	PL	PL
PS	NL	NM	ZE	PS	PL	PL	PL
PM	NM	ZE	PL	PL	PL	PL	PL
PL	ZE	PM	PL	PL	PL	PL	PL

Fig. 4(b) Control rules base

For a two-input system, position and velocity differences, each of which has seven fuzzy values, there are $7 \times 7 = 49$ possible input combinations. Therefore, if the rule base is to be fully populated, there will be 49 rules. Figure 4(*a*) shows the control strategy for the pursuit operation, where the unshaded region is a mirror image of the shaded cells. The completed rulebase is shown in Fig. 4(*a*), where the central cell is the desired system state. The output from the controller can be obtained from the inputs by using the compositional rule of inference [8].

The typical control action for the dynamic pursuit is illustrated as follows: Consider if there is a large positive difference

Downloaded 16 Mar 2010 to 130.207.50.192. Redistribution subject to ASME license or copyright; see http://www.asme.org/terms/Terms_Use.cfm

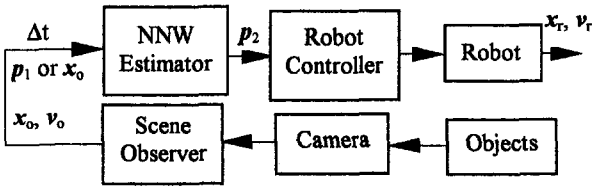

Fig. 5 Schematic illustrating location estimation for picking up moving object

in position (PL), as indicated in shaded region B in Fig. 4(*a*). The robot speed is commanded to increase using a linguistic rule such as:

IF *Position_Difference* is PL AND *Speed_Difference* is ZE,

THEN *Robot_Speed_Change* is PL.

The corresponding speed difference will then become more and more negative. Finally the speed difference reaches NL and the control action will gradually be lowered to ZE as shown on the upper right cell of the rulebase matrix in Fig. 4(*b*). When the position difference reduces, the control action reverses to NL reducing the speed of the robot as the system enters region C. With the reduction of the robot speed, the speed difference is increasing. The system then moves through region C and towards region D of Fig. 4(*a*). The rules for the three cells in region D are

IF *Position_Difference* is NS AND *Speed_Difference* is ZE

THEN *Robot_Speed_Change* is NM;

IF *Position_Difference* is NS AND *Speed_Difference* is PS

THEN *Robot_Speed_Change* is ZE;

IF *Position_Difference* is ZE AND *Speed_Difference* is PS

THEN *Robot_Speed_Change* is PS.

Finally, the system is further dampened in toward cell A which is the desired system state. Note that the heavy black line below region D is a boundary the response cannot pass, because the unshaded cells in Fig. 4(*a*) must be the mirror image of the shaded cells to respond to a negative large position difference; that is, they must have identical amplitude but opposite polarity actions. If the system response crosses the black line during the last stages of the operation, it will act as if it were in the middle stages of a negative position difference response. The completed rulebase is shown in Fig. 4(*b*).

4 Neural Network Estimation for Pick Up Operation

As the robot moves closer towards the vicinity of the target, the field-of-view of the vision system becomes smaller, and often the visibility of the vision system must give way as the gripper adapts itself to perform a pick up task. An alternative technique is presented here to estimate the location at which the object must be picked up once the target is within the vicinity of the robot gripper.

Consider that the moving object is at location $\boldsymbol{p}_1$ at time t_1 and the robot is commanded to pick the object up. The robot requires, however, a finite reaction time to compute and execute the pick up command in addition to the time needed for communication among the subsystems. The location $\boldsymbol{p}_2$ at which the object must be picked up at time $t_2 = t_1 + \Delta t$ is estimated using a trained neural network. Unlike the closed-loop fuzzy logic controller for the phase of pursuit, the neural network is used in an open-loop sense, as shown in Fig. 5, to estimate the position of moving object for the robot to pick up. That is,

$$\boldsymbol{p}_2 = \boldsymbol{f}(\boldsymbol{p}_1, \Delta t) \tag{6}$$

where Δt is specified as a constant value larger than the sum of all the incremental times needed between the computation of the last object location and the execution of the pick up task. The goal is to adapt the parameters of the network so that for any input $(\boldsymbol{p}_1, \Delta t)$ in real time, it will perform a sufficiently accurate on-line estimation of the object position $\boldsymbol{p}_2$.

The object motion, though nonlinear, exhibits certain complex patterns over a certain area. This makes it suitable to apply an adaptive neural network to recognize the complex motion patterns and perform the nontrivial nonlinear mapping between $(\boldsymbol{p}_1, t_2 - t_1)$ and $\boldsymbol{p}_2$. The basic learning rule of the adaptive network is the well-known steepest descent method, in which the gradient vector is derived by successive invocations of the chain rule. A back-propagation neural network is used to find a mapping network which compute a functional relationship between their inputs and outputs from a representative collection of sample training data. Back-propagation is chosen since this most popular network has been proven to work well in many applications, is easy to learn and is powerful. In addition, it requires a relatively fewer operations to calculate gradient than do other methods and it can be relatively easily implemented by parallel digital hardware [9–10].

Here, the objective is to train the network so that it will learn an approximation $\boldsymbol{p}_2 = \boldsymbol{f}(\boldsymbol{p}_1, \Delta t)$ using sample trajectories of moving targets pre-determined off-line using a vision system. The basic concept is as follows: Starting with an initial set of weights and bias set to small random numbers, an input vector $(\boldsymbol{p}_1, t_2 - t_1)$ is then presented to the network. It sweeps forward through the network to generate an estimated output response vector $\boldsymbol{p}_2'$. The algorithm computes the errors at each output. Next, the effects of the errors are propagated backward through the network to associate a "square-error derivative" δ with each node and compute the weights of each node based on the corresponding gradient. A new pattern is then presented and the process is repeated. The process is repeated until the error is within a specified tolerance.

Figure 6 shows a three-layer network with N inputs, M outputs and two layers of hidden nodes. The sizes of the input and output layers which deal with numeric input data are dictated by the applications. The preprocessing of the images is one of the keys to the network's ability to handle many different tasks. The system must be simple since with a neural network a person on the factory shop-floor doesn't have to be a programmer to train the machine. Thus, in the case of a vision-based part-feeder where the vibratory surface can be pre-determined, we choose the inputs to be the x and y coordinates of the object position derived from vision images and the reaction time $(t_2 - t_1)$ of the system. The outputs are the estimated x and y coordinates of the object.

The linear output $s_j^{(i)}$ of the jth node in layer i (see Fig. 6) is given by

$$s_j^{(i)} = \sum_{l=1}^{n(i-1)} W_{jl}^{(i)} x_l^{(i)} + b_j, \tag{7}$$

where $W_{jl}^{(i)}$ is the weight between lth node in layer $i-1$ and jth node in layer i; $x_l^{(i)}$ is the lth input of the node j in layer i; $n(i-1)$ is the number of nodes in layer $i-1$; and b_j is the bias term for node j in layer i. In general, given a set of vector-pairs, $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_p, \mathbf{y}_p)$ characterized by a functional mapping $\boldsymbol{y} = \boldsymbol{\phi}(\boldsymbol{x})$: $\boldsymbol{x} \in R^N, \boldsymbol{y} \in R^M$, the back-propagation training is to find an approximation $\boldsymbol{y}' = \boldsymbol{\phi}'(\boldsymbol{x})$ by using an iterative gradient algorithm. This algorithm is designed to minimize the sum of the squared error between the actual output of a multi-layer feed-forward network and the desired output:

$$\epsilon^2 = \sum_{l=1}^{M} \epsilon_l^2 = \sum_{l=1}^{M} (y_l - y_l')^2, \tag{8}$$

where y_l and y_l' are the desired and the actual output of the

Downloaded 16 Mar 2010 to 130.207.50.192. Redistribution subject to ASME license or copyright; see http://www.asme.org/terms/Terms_Use.cfm

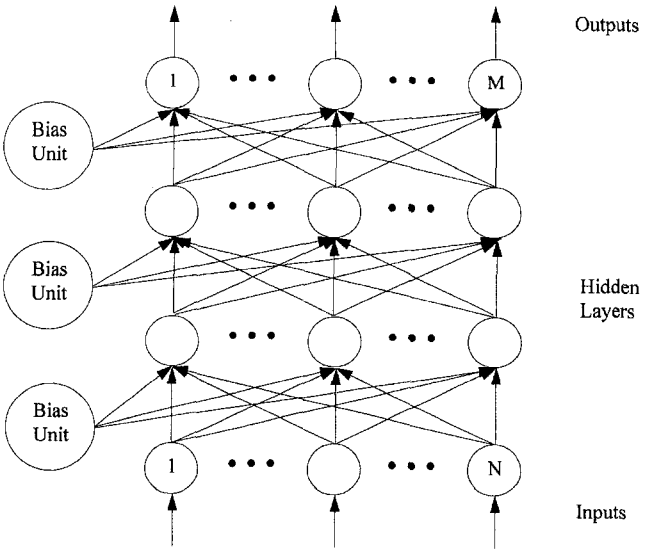

Fig. 6 A three layer neural network

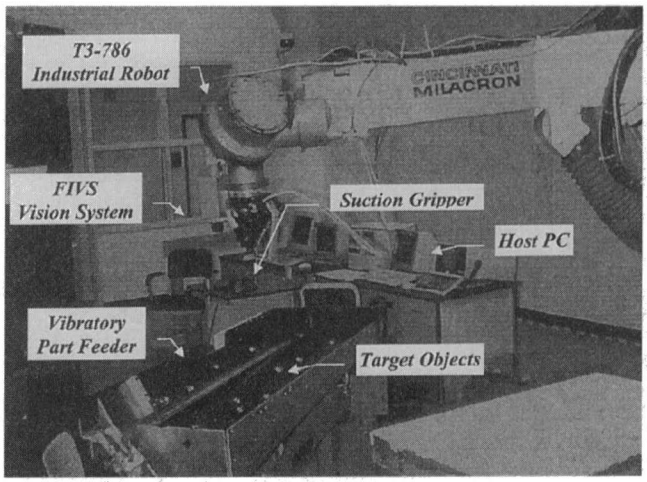

Fig. 7 Experimental setup for flexible robotic part-feeding

neural network, respectively; M is the number of the neural network output nodes.

The goal is ultimately to minimize the error between the desired output and the current output sample by continuously modifying the weights using supervised learning. With supervised learning, it is necessary to "train" the neural network before it becomes operational. Training consists of presenting input and output data, often referred to as the training set, to the network. That is, for each input presented, the corresponding desired output is presented as well. In many applications, actual data are used. The training procedure can be organized as follows:

Step 1: All the weights and node biases are set to small random values.

Step 2: The input vector, $\boldsymbol{x}_p = (x_{p1}, x_{p2}, \ldots, x_{pN})^T$ and the desired outputs, $\boldsymbol{y}_p = (y_{p1}, y_{p2}, \ldots, y_{pM})^T$ are specified.

Step 3. Starting from the first layer, compute the output of each node through the system to generate the actual system outputs, $\boldsymbol{y}'_p = (y'_{p1}, y'_{p2}, \ldots, y'_{pM})^T$. For example, we can compute the output of node j in layer i as follows:

$$x_j^{(i)} = f(s_j^{(i)}), \tag{9}$$

where f is the transfer function of the node, and $s_j^{(i)}$ is the linear output of the node.

Step 4. Use a recursive algorithm starting at the output nodes to propagate back to the first hidden layer. Adjust weights by using the following equation,

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \Delta\boldsymbol{w}(t) \tag{10a}$$

$$\Delta\boldsymbol{w}(t) = -\eta\delta(t)\boldsymbol{x}(t) + \rho\Delta\boldsymbol{w}(t-1) \tag{10b}$$

where η is a constant or learning rate and ρ is a momentum constant. In Equation (10b), the square-error derivatives are computed by

$$\delta = \epsilon f'(s), \tag{11}$$

where $\epsilon = y_l - y'_l$ is the output error associated with the node, f' is the derivative of the transfer function of the node. The node output is computed by Eq. (7). The learning rate has an effect on the convergence speed and stability of the weights during learning. A slower learning rate means a lot more than time is spent in accomplishing the off-line learning to produce a trained system. A typical learning rate η is between 0 and 1. If the learning rate is greater than 1, it is easy for the learning algorithm to "overshoot" in correcting the weights, and the network may oscillate. The addition of the momentum term smoothes weight updating and tends to resist erratic weight changes due to gradient noise. However, use of the momentum term does not always seem to speed up training; it is typically application dependent. A practical guide to select the learning parameters can be found in [11].

Step 5. Compute the error term with Eq. (8). The quantity is the measure of how the network is learning. When the error is acceptably small for each of the training-vector pairs, training can be discontinued. Otherwise, go back to Step 2.

5 Experimental Investigation

The vision-based controller was experimentally implemented. Specifically, two experiments were designed to examine the control strategies. The first experiment was to tune the fuzzy logic controller for the process of target pursuit. The second experiment was to evaluate the target position for the purpose of part pick up.

5.1 System Setup. The experiments were performed in the Factory-of-the-Future Kitting Cell at Georgia Tech, where multiple vibratory feeders are used to handle kits of all sizes and types and to prepare kits for transporting between assembly systems. In a typical industrial layout, the feeder system receives parts in bulk on a vibratory resilient surface where parts are separated and directed to a nest position (part-specific fixture) at the end of the feeder for pick up by a pre-programmed robot. The robot then places the parts on a generic (or egg-crate style) tray. As shown in Fig. 7, the specific system used in the experiments consists of a six degree-of-freedom (DOF) Cincinnati Milacron T3 industrial robot, a re-circulating vibratory feeder as shown in Fig. 1, a flexible integrated vision system (FIVS), and a vision-guided controller which has been implemented on a Intel 486-25 MHz personal computer. The functional relationship of the vision-guided robot system is shown in Fig. 8 where the notation ${}^a_bT$ defines the transformation of the coordinate frame of "b" with respect to that of "a." In Fig. 8, the intrinsic parameters of the camera (focal length f and lens distortion coefficient k) and the kinematic relationship among the vision system (c), the gripper (g), and the vibratory feeder (v) are determined off-line using a calibration algorithm based on the theoretical framework originally established by Tsai [12] and Tsai and Lenz [13].

In the system discussed here the robot is treated as a black box. In other words, the forward and inverse kinematics, trajectory generation, and motion control of the robot are handled by the robot controller. The robot is driven by sensory information

Downloaded 16 Mar 2010 to 130.207.50.192. Redistribution subject to ASME license or copyright; see http://www.asme.org/terms/Terms_Use.cfm

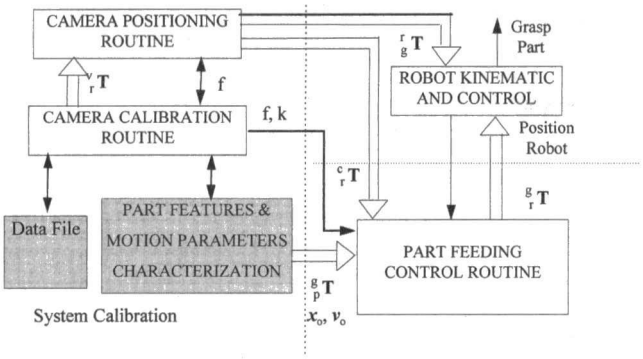

Fig. 8 Machine vision for intelligent part feeding

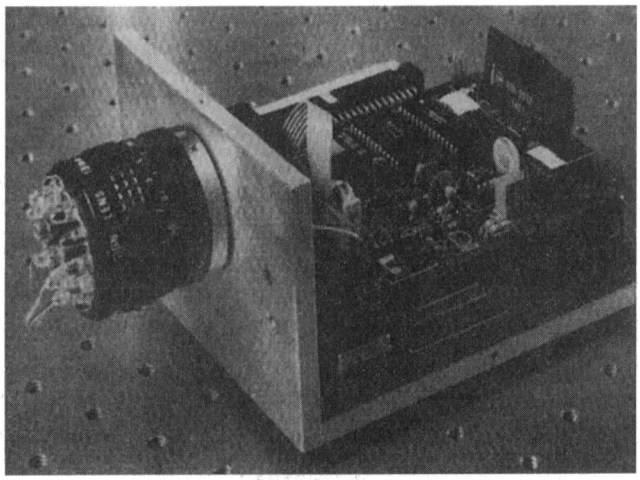

Fig. 9 Flexible Integrated Vision System (FIVS)

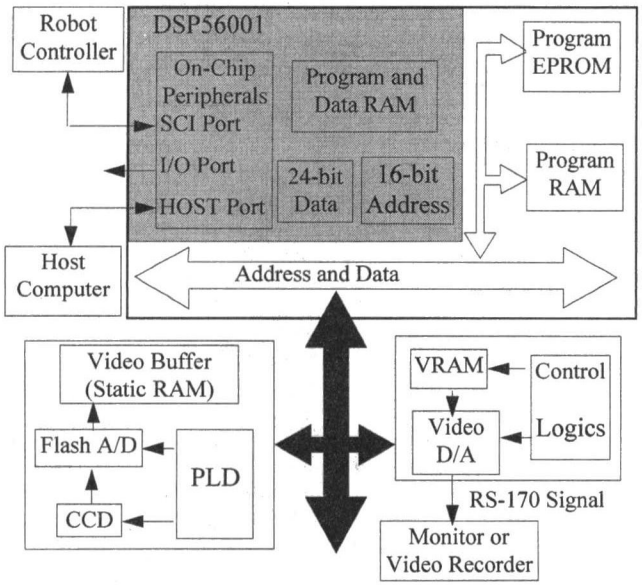

Fig. 10 Schematic illustrating FIVS

from the vision-guided controller as well as inputs from the off-line calibration. The function of the vision-guided controller is to serve as the host and thus control all system activities.

A photo of the *f*lexible *i*ntegrated *v*ision *s*ystem (FIVS) and its schematic are given in Figs. 9 and 10 respectively. As shown in Fig. 10, FIVS has five basic functional modules: (1) an

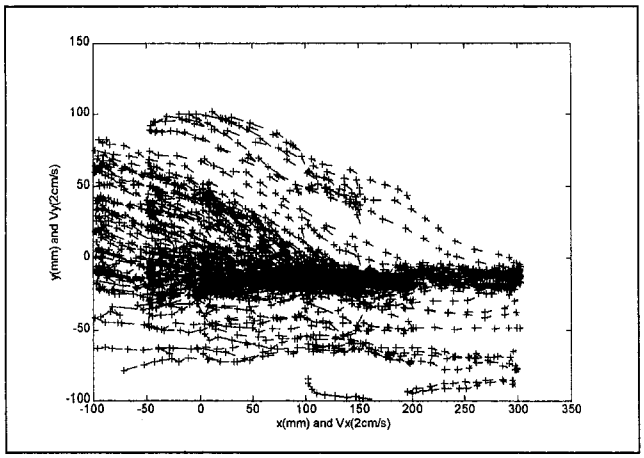

Fig. 11 Typical velocity vectors of moving parts

on-board computer which consists of a microprocessor and its associated *e*lectrically *e*rasable *p*rogrammable *r*ead-*o*nly-*m*emeory (EEPROM), scratch *r*andom *a*ccess *m*emory (RAM), and communication hardware, (2) a video head which includes the imaging sensor - a charge coupled device (CCD), a high-bandwidth signal-conditioning-amplifier, an analog-to-digital (A/D) converter, and video RAM, (3) an optic system to house the lens (or simply a pin-hole) and its associated illumination, (4) an off-line host-interface for a user to carry out off-line calibration, perform image analysis, and implement application-specific software through a host computer, and (5) a real-time video record/playback to analyze failure-modes off-line. The main kernel of FIVS provides a user interface whereby a user can reprogram the EEPROM of the processor board. This allows the user to customize the image processing for a particular task, from a library of algorithms. Based on the hardware design, the software is able to control the CCD array scanning and integration time, and the intensity of the collocated illumination. With the CCD under software control, partial frames can be "captured" instead of the customary full frame, reducing the cycle time required to capture and process an image. The ability to shift out partial frames is ideal for high speed tracking applications where the approximate location is known from a prior image. By reducing the time to capture an image, the effective frame rate is increased. For example, shifting out $\frac{1}{4}$ of an image can increase the frame rate up to 480 fps, not including the time required for illumination and image processing.

5.2 Experimental Results and Discussions. A component part used in car steering mechanisms was chosen as a target to demonstrate the concept. The part has a flat inner surface of 24 mm diameter. The suction gripper with a vacuum pad of 10 mm was used for the pick up operation. Provided that the difference between the centers of the suction gripper and the moving object is no more than 5 mm on the vibratory surface of known height, the object can be picked up. Thus, the final desired position difference between the robot gripper and the moving target was selected to be ± 5 mm in both x and y direction. The speed difference was determined experimentally to be 5 mm/s. Figure 11 shows the typical motion of the objects moving on the vibratory feeder, which were captured at a rate of approximately 25 frames per second. The data were obtained with the image plane of the FIVS vision system positioned in parallel to and at 1 m from the surface of the vibratory feeder.

Figure 12(*a*) shows typical position tracking results of the fuzzy logic controller. Figure 12(*b*) displays the corresponding velocity of the target (indicated as "*o*") and the robot (indicated as "+") in x and y directions. The gains of FLC were tuned experimentally and are given in Table 1. As shown in

Downloaded 16 Mar 2010 to 130.207.50.192. Redistribution subject to ASME license or copyright; see http://www.asme.org/terms/Terms_Use.cfm

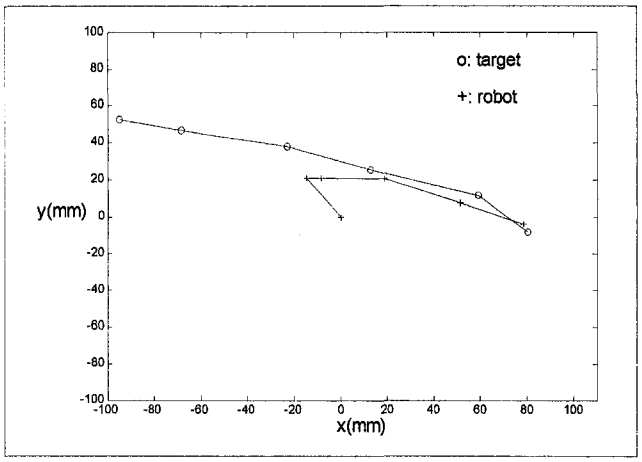

Fig. 12(*a*) Position tracking results of FLC-based pursuit operation

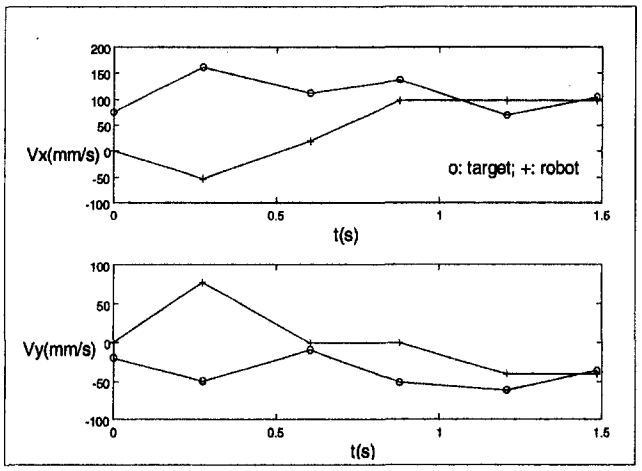

Fig. 12(*b*) Robot and target velocity data of FLC-based pursuit operation

Table 1 Parameters of fuzzy logic controller

	K_p	K_D	K_U
x-direction	20	40	24
y-direction	40	20	12

Fig. 12(*a*), the FLC successfully guides the robot to "pursue" the object to within its vicinity defined as ±5 mm in both x and y directions in about 0.6 second. However, since there was a significant difference in velocity between the target and the robot, the FLC continues to command the robot to pursue the target until the final velocity difference was within 5 mm/s. The complete tracking operation took about 1.5 seconds.

In the case of grasping, there will come to a point in which the view of the object being tracked will become obscured by the actuating mechanism as the gripper approaches the object in the z-direction in an attempt to pick up the object. Thus, it is necessary to predict the future state of an object that is desired to be grasped since the system essentially becomes an open-loop system. Unlike the learning algorithm proposed by Miller [14], which commands the robot through joint velocity servo loops to track an object on a non-vibratory moving conveyor at a constant velocity, the method proposed here has been based on end-point position command in Cartesian space. In other words, the robot and its controller has been treated as a "black box."

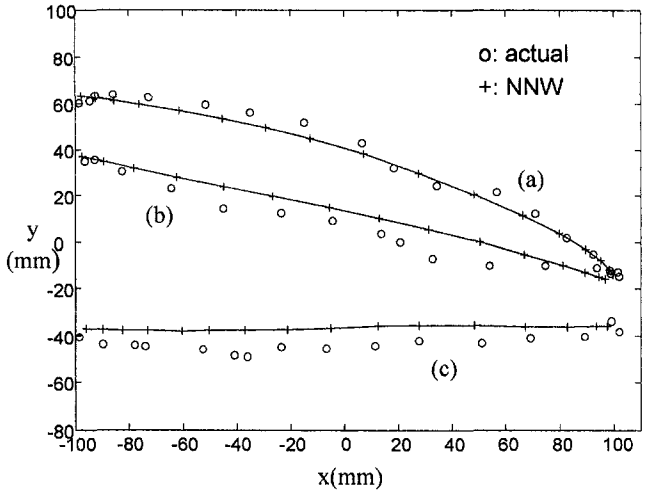

Fig. 13 Actual and estimated part trajectories

A three layer back-propagation neural network structure is adopted with the tangent sigmoid transfer function used for neurons in the first two layers and the linear transfer function for neurons in the final output layer. The inputs to the back-propagation network are the position of an object $\boldsymbol{p}_1(x_1, y_1)$ at time t_1 and the time interval Δt, the outputs are the position $\boldsymbol{p}_2(x_2, y_2)$ of the object at time $t_1 + \Delta t$. The trajectories of the moving objects, which were used to train the network, were monitored and recorded by the FIVS vision system. Each of the iterative processes include capturing and processing an image, computing the location of the object, and sending the computed location through a serial communication to the host computer which stores the data in a data file. The data acquisition was performed off-line between the FIVS and the host computer with a sample object moving on the vibratory feeder. With a 486-25 MHz PC, the typical cycle time is in the order of 0.25 second. A total of 584 input data (x_1, y_1, Δt) and the same number of desired output data (x_2, y_2) were constructed with a range of Δt in the training. The training data are then used to find a mapping function using the back-propagation network. Once trained, a neural network can operate in software, hardware (special chips), or a combination of the two.

The data not used in the training were used as a basis for the performance measurement of the trained network. The positions of the object estimated by the network were compared with its actual positions at different times, which were graphed in Fig. 13. The successful estimation is defined as within ±5 mm of the desired position in both x and y directions and the corresponding success rate for each test trajectory is shown in Table 2.

It has been found that the back-propagation neural network can estimate the motion of a moving target in most cases. Based on the data sets which were not used in the training, the success rate was found to vary from 86.67 percent to 93.75 percent. In general, the trajectories of moving targets over the entire region need to be collected as the training data. With the FIVS vision system and an appropriate data collection and preparation algorithm, this is not a major problem. Therefore, the accuracy of the estimation can be further improved. More detailed discussion and experimental data can be found in Reference [15].

Table 2 Success rate for part position estimation

Trajectory	a	b	c
Success Rate	88.24	86.67	93.75

Downloaded 16 Mar 2010 to 130.207.50.192. Redistribution subject to ASME license or copyright; see http://www.asme.org/terms/Terms_Use.cfm

The eye-on-hand vision-guided part-feeder effectively eliminates a part-specific nest which is otherwise needed in the case of hard automation. Since the vision system is attached on the end-effector of the robot, the system must execute the estimation and grasping tasks sequentially. It is expected that a finite amount of vision processing time can be effectively eliminated if the vision system can be fixed in space instead of attached onto the robot end-effector, which would allow the vision and robot systems to function in parallel.

6 Conclusions

The concept of applying the principle of dynamic pursuit to locate moving objects for flexible part kitting applications has been discussed. Since the dynamics of moving parts on the vibratory feeder are highly nonlinear and are impractical to model accurately, the problem has been formulated in the context of Prey capture with the robot as a "pursuer" and a moving object as a passive "prey" or "evader." By incorporating a real-time machine vision system to guide the robot, the system gains the ability to locate parts and feed them intelligently to subsequent processes. Such a formulation mimics the function and capability of a natural being to pursue its prey, which opens the opportunity to apply experimental or heuristic knowledge as basis to do logical inference and linguistic rules based on "rules-of-thumb" experiences and engineering judgments to specify control laws.

The vision-based intelligent controller consists of two parts: a tracking controller and an open-loop estimator. The controller has been developed and examined in the Factory-of-the-Future Kitting Cell at Georgia Tech. The tracking controller is based on the principle of fuzzy logic to guide the robot search for an object of interest and then to pursue it. The open-loop estimator is based on a back-propagation neural network to predict its position at which the robot gripper picks up the object. The feasibility of the concept and the control strategies were verified by two experiments. The first experiment showed that the fuzzy logic controller could command the robot to successfully follow the highly nonlinear motion of a moving object and approach its vicinity. The second experiment demonstrated that the neural network could estimate its position fairly accurately in a finite period of time after the command of pick up operation was issued.

As compared to traditional control in part feeding, the system structure and control strategies developed have the following advantages: (1) By providing a means to estimate the location of the moving objects, the need of an object-dependent fixture for each part family can be eliminated. (2) The multi-phase intelligent controller can be separately designed for different phases. (3) It does not require the specific analytical model of a system and can successfully deal with highly non-linear problems. (4) The control parameters of the system can be easily adjusted to different applications or different objects without having to change the system hardware. Thus, the technique can not only be readily applied to integrating off-the-shelf components, but it can also make the resulting system less structured and more flexible to deal with different parts. It is expected that the system has a significant potential to improve the flexibility for quick model changeover and to reduce the cost of implementation.

Acknowledgments

This research was partly supported by the National Science Foundation (NSF) through the NSF Presidential Young Investigator Grant No. 8958383 and the Woodruff Faculty Fellow. The donation of the Small Parts Kitting Cell by General Motor Company in supporting this work is greatly appreciated.

References

1 Lotter, B., "Planning and Implementation of Flexible Assembly Cells," *Proceedings of the 7th International Conference on Assembly Automation,* Zurich, Switzerland, Feb. 2–4, 1986, pp. 1–9.

2 Lee, K.-M., "Design Concept Of An Integrated Vision System For Cost-Effective Flexible Part-Feeding Applications," ASME JOURNAL OF ENGINEERING FOR INDUSTRY, Vol. 116, pp. 421–428, November 1994.

3 Lee, K.-M., and Blenis, R., "Design Concept and Prototype Development of a Flexible Integrated Vision System," *Journal of Robotic Systems,* Vol. 11, No. 5, pp. 387–398, 1994.

4 Sharma, R., and Aloimonos, J., "Target Pursuit or Prey Catching Using Qualitative Visual Data," *Proceedings of AAAI-90 Workshop on Qualitative Vision,* pp. 195–198, July 1990.

5 Issacs, R., *Differential Games,* John Wiley & Sons, Inc., New York, 1965.

6 Kuc, R., and Barshan, B., "Bat-like Sonar for Guiding Mobile Robots," *IEEE Control Systems Magazine,* pp. 4–12, August 1992.

7 Zadeh, L. A., "Fuzzy Sets," *Information Control,* Vol. 8, No. 3, pp. 338–353, June 1965.

8 Berenji, H. R., "Fuzzy Logic Controllers," *An Introduction to Fuzzy Logic Applications in Intelligent Systems,* Ronald R. Yager and Lotfi A. Zadeh, eds., Kluwer Academic Publishers, 1993.

9 Lippman, R. P., "An Introduction of Computing with Neural Nets," *IEEE Acoustics, Speech, and Signal Processing,* April 1987, pp. 4–22.

10 Widrow, B., and Lehr, M. A., "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Back-propagation," *Proceedings of IEEE,* Vol. 78, No. 9, September 1990, pp. 1415–1442.

11 Nelson, M. M., and Illingworth, W. T., *A Practical Guide to Neural Nets,* Addison Wesley, 1991.

12 Tsai, R. Y., "A Versatile Camera Calibration Technique for High Accuracy 3D Machine Vision Metrology using Off-the-shelf TV Camera and Lenses," *IEEE Robotics and Automation,* Vol. RA-3, No 4, August 1987, pp. 323–344.

13 Tsai, R. Y., and Lenz, R. K., "A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration," *IEEE Transactions on Robotics and Automation,* Vol. 5, No. 3, June 1989, pp. 345–358.

14 Miller, M., "Sensor-based Control of Robotic Manipulators Using a General Learning Algorithm," *IEEE Transactions on Robotics and Automation,* Vol. 3, No. 2, June 1987, pp. 157–165.

15 Qian, Y., "A Study on Dynamic Pursuit of Moving Objects with Hand-Eye Coordination," Ph.D. thesis, the George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, Georgia, May 1995.

Downloaded 16 Mar 2010 to 130.207.50.192. Redistribution subject to ASME license or copyright; see http://www.asme.org/terms/Terms_Use.cfm