ARPA Image Understanding Workshop

Nov, 1994

General Learning Algorithm for Robot Vision

Shree K. Nayar, Hiroshi Murase, Sameer A. Nene*
Department of Computer Science
Columbia University, New York, N.Y. 10027, U.S.A.

Abstract

The problem of vision-based robot positioning and tracking is addressed. A general learning algorithm is presented for determining the mapping between robot position and object appearance. The robot is first moved through several displacements with respect to its desired position, and a large set of object images is acquired. This image set is compressed using principal component analysis to obtain a low-dimensional subspace. Variations in object images due to robot displacement are represented as a compact parametrized manifold in the subspace. While positioning or tracking, errors in end-effector coordinates are efficiently computed from a single brightness image using the parametric manifold representation. The learning component enables accurate visual control without any prior hand-eye calibration. To demonstrate its generality, the learning algorithm is also used to develop a real-time object recognition system. Several experiments are reported that involve tasks pertinent to industrial applications.

1 Introduction

For a robot to be able to interact in a precise and intelligent manner with its environment, it must rely on sensory feedback. Vision serves as a powerful component of such a feedback system. It provides a richness of information that can enable a manipulator to handle uncertainties inherent to a task, react to a varying environment, and gracefully recover from failures. A problem of substantial interest and relevance to robotics is visual positioning/tracking; the ability of a robot to either automatically position itself at a desired location with respect to an object, or accurately follow an object as it moves along an unknown trajectory.

This paper presents a new approach to visual positioning and tracking. Before we proceed to describe our approach, a brief review of previous work is in order. Previous tracking methods can be broadly classified into two categories; (a) feature/model based and (b) learning based. The first category uses image features to estimate the robot's displacement with respect to the object. The goal is to find the rotation and translation that must be applied to the end-effector to bring the features back to their desired positions in the image. Image features used vary from geometric primitives such as edges, lines, vertices, and circles [Weiss et al. 87] [Feddema et al. 91], [Koivo and Houshangi 91] [Hashimoto et al. 91] to optical flow estimates [Papanikolopoulos et al. 91] [Luo et al. 88] and object location estimates obtained using stereo [Allen et al. 92]. The control schemes used to drive the robot to its desired position vary from simple prediction algorithms employed to achieve computational efficiency, to more sophisticated adaptive self-tuning controllers that account for the dynamics of the manipulator. Many of the above methods require prior calibration of the vision sensor's intrinsic parameters (e.g. focal length) as well as its extrinsic parameters (rotation and translation with respect to the manipulator).

The second category of positioning/tracking methods includes a learning component. In the learning stage, the mapping between image feature locations and robot coordinates is generated prior to positioning/tracking (off-line). This mapping is then used to determine, in real-time, errors in robot position/velocity from image feature coordinates. This is generally accomplished without any explicit knowledge of the object's geometry or the robot's kinematic parameters. In addition, calibration of the vision sensor is not required as long as the sensor-robot configuration remains unaltered between learning and tracking. Methods in this category differ from each other primarily in the type of learning algorithm used. The learning strategies vary from neural-like networks [Kuperstien 87] [Mel 87] [Miller 89] [Walter et al. 90] to table lookup mechanisms such as the cerebellar model articulation controller (CMAC) [Albus 75] [Miller 87].

*This work was supported in part by ARPA contract DACA-76-92-C-007 and in part by the David and Lucile Packard Foundation. A part of the research reported in this paper was presented at the IEEE International Conference on Robotics and Automation, San Diego, May 1994.

Here, we propose a new framework for learning-based visual servoing. Our approach differs from previous ones in two significant ways; (a) it uses raw brightness images directly without the computation of image features, and (b) the learning algorithm introduced is based on *principal component analysis*[1] (see [Oja 83]) rather than a large input/output mapping network. During the learning stage, a sizable image window is selected that represents the appearance of the object when the robot is in the desired position. A large set of object images is then obtained by incrementally perturbing the robot's end-effector (hand-eye system). The image set is compressed using principal component analysis to obtain a low-dimensional subspace, called eigenspace. Variations in object images due to robot displacements are represented in the form of a parametrized manifold in eigenspace. The advantages of using this representation are discussed in the paper.

In the positioning or tracking application, each new image is projected to the eigenspace and the location of the projection on the parametrized manifold determines the robot displacement (error) with respect to the desired position. Positioning and tracking are achieved without prior knowledge of the object's geometry or reflectance, the robot's kinematic parameters, and the vision sensor's intrinsic and extrinsic parameters. We conclude with experiments conducted using an Adept robot. The accuracy and efficiency of the proposed method are demonstrated using three sample applications; (a) the assembly of electronic devices on a circuit board, (b) the insertion of a peg in a hole; and (c) the tracking of a manufactured part moving on a turntable.

Though we have focused primarily on the problem of visual servoing, the learning and recognition framework presented here can be applied to a variety of vision problems. As an example, we present a real-time object recognition system with 20 complex objects in its database. A complete recognition and pose estimation cycle takes less than 1 second on a Sun IPX workstation without the use of any customized hardware.

2 The Approach

Figure 1 shows the *hand-eye* system we have used to demonstrate visual positioning and tracking. The manipulator used is a 5 degree-of-freedom Adept robot that is interfaced with a Sun IPX workstation. A CCD camera is mounted adjacent to the robot gripper and provides images of the tracked object. The end-effector's position (translation and rotation) can be described in any frame of reference attached to the robot. Without loss of generality, we denote the end-effector position by the generalized coordinates:

$$\mathbf{q} = [q_1, q_2, ..., q_m]^T \tag{1}$$

where m represents the end-effector degrees of freedom (DOF) used in the positioning or tracking application. The imaging optics is selected such that the tracked object occupies a large section of the image. The image area used as visual input is a fixed window, within the complete image, which includes sufficient object detail (for example, see Figure 2) [2]. This image window is written as a vector $\mathbf{i}$ by reading brightness values from it in a raster scan fashion:

$$\mathbf{i} = [i_1, i_2, ..., i_N]^T \tag{2}$$

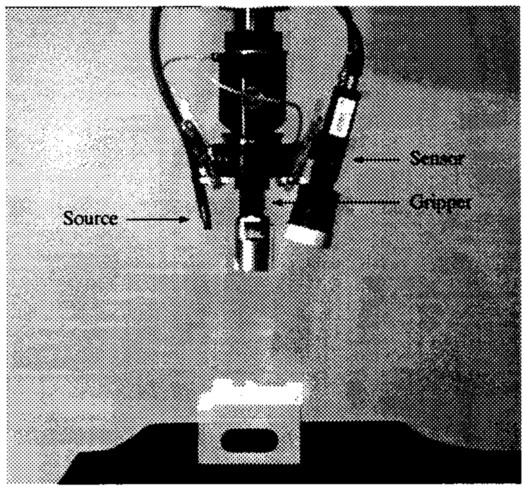


Figure 1: The hand-eye system used for visual positioning and tracking. The end-effector includes a gripper, an image sensor, and a light source.

Our objective is to compute off-line the mapping between the robot end-effector coordinates and object images. The brightness image $\mathbf{i}$ for any given robot position $\mathbf{q}$ depends on the three-dimensional shape of the object, its reflectance properties, the illumination conditions, and the end-effector coordinates with respect to the object. Shape and reflectance are intrinsic properties of a rigid object that do not change during positioning or tracking. In order to overcome the effects of possible illumination variations, we have

[1] In machine vision, principal component analysis has been applied to problems such as edge detection [Hummel 79] and face recognition [Turk and Pentland 91].

[2] Alternatively, the contents of several windows of fixed sizes and shapes, scattered in the image, can be concatenated and treated as a single window.

used a light source that is also mounted on the end-effector. In our setup (see Figure 1), the source is one end of a fiber-optic cable connected to a strong light source at the other end. This *hand-source* is the dominant source of object illumination [3], and minimizes the effects of ambient illumination.

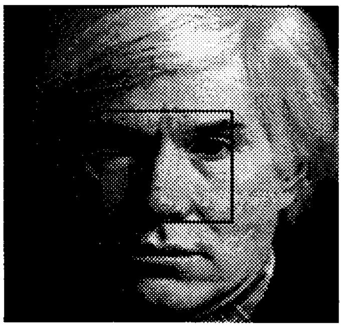


Figure 2: Each image vector $\mathbf{i}$ is obtained by reading pixel brightness values from an image window (black box) of fixed size and position.

This leaves us with the position and orientation of the end-effector with respect to the object. We define the nominal or *desired image* $\tilde{\mathbf{i}}$ as the one produced when the robot is in the desired position and orientation $\tilde{\mathbf{q}}$ with respect to the object. For convenience, we define all robot displacements $\mathbf{q}$ with respect to $\tilde{\mathbf{q}}$, i.e. a coordinate system with $\tilde{\mathbf{q}} = 0$. For any given positioning/tracking application, one can define a *task workspace* as the range of permissible end-effector displacements $\mathbf{q}$ with respect to $\tilde{\mathbf{q}}$. This in turn corresponds to a range of visual appearances that we refer to as the *visual workspace*, $\mathbf{i}(\mathbf{q})$. It is this mapping, $\mathbf{i}(\mathbf{q})$, that we wish to learn. Once it has been learned, the robot displacement $\mathbf{q}$ with respect to the desired position can be determined from any image by *inverse mapping*. As we will see shortly, this inverse mapping is continuous and can be achieved using raw images without the computation of image features such as edges, lines, corners, circles, or optical flow estimates.

Several advantages result from the proposed approach. (a) The three-dimensional shape and reflectance properties of the object need not be known or computed. The effects of shape and reflectance are embedded in the raw brightness images. (b) Robot displacements are computed using images rather than image features. This not only saves computations but also avoids detection and localization errors introduced by feature extraction algorithms. (c) The extrinsic and intrinsic parameters of the camera are not used. Therefore, it is not necessary to calibrate the camera with respect to the hand or any other coordinate system; a process that is known to be cumbersome. All that is required is that the camera be positioned on the hand such that a descriptive image of the object is available.

[3]This illumination method is effective not only for the approach described in this paper but also for visual servoing methods that rely on robust computation of image features. Placing the source close to the sensor also minimizes shadows in the image.

3 Learning the Visual Workspace

We now present the learning approach used to determine a direct mapping between the generalized robot coordinates and object images. Later, we show how this mapping can be effectively used for positioning and tracking purposes.

3.1 Acquiring Learning Images

All images of the object taken by varying the robot position are of the same size, the size of the image window used (Figure 2). The window is selected such that it remains on the object even while robot coordinates are varied during the learning process. This ensures that the positioning and tracking stages are unaffected by the background of the object, avoiding the use of a segmentation algorithm. It is reasonable to assume that in a positioning or tracking application errors in robot position are relatively small. Hence, the range of discrete manipulator positions used to obtain the image set may be confined to a small area around the desired position $\tilde{\mathbf{q}}$. We denote the image corresponding to the discrete position $\mathbf{q}_j$, as $\mathbf{i}_j$.

Prior to learning, the imaging sensor is linearized using a simple calibration procedure. This ensures that image brightness is proportional to scene radiance. We would like our learning process to be unaffected by variations in the intensity of illumination or the aperture of the imaging system. This can be achieved by normalizing each acquired image such that the total energy contained in the image is unity: $\hat{\mathbf{i}}_j = \mathbf{i}_j / \parallel \mathbf{i}_j \parallel$. Let the number of discrete samples obtained for each DOF l be R_l. Then the total number of images is $M = \Pi_{l=1}^{m} R_l$. The complete image set:

$$\{ \hat{\mathbf{i}}_1, ..., \hat{\mathbf{i}}_2, ..., \hat{\mathbf{i}}_M \} \qquad (3)$$

is a sampling of the visual workspace. The image $\tilde{\mathbf{i}}$ corresponding to the desired robot position $\tilde{\mathbf{q}} = 0$ is also included in this set.

3.2 Computing Eigenspaces

Images in the above set tend to be correlated to a large degree since end-effector displacements between consecutive images are small. Our first step is to take advantage of this correlation and compress the large set to a low-dimensional representation that captures the key appearance characteristics of the visual workspace. A suitable compression technique is based on principal component analysis [Oja 83], where the eigenvectors of the image set are computed and used as orthogonal bases for representing individual images. Though, in general, all the eigenvectors of an image set are required for perfect reconstruction of any particular image, only a few are sufficient for positioning or tracking applications. These eigenvectors constitute the dimensions of the *eigenspace*, or image subspace, in which the visual workspace is compactly represented.

First, the average $\mathbf{c}$ of all images in the set is subtracted from each image. This ensures that the eigenvector with the largest eigenvalue represents the subspace dimension in which the variance of images is maximum in the correlation sense. In other words, it is the most important dimension of the eigenspace. An image matrix is constructed by subtracting $\mathbf{c}$ from each image and stacking the resulting vectors columnwise:

$$\mathbf{P} \triangleq \left\{ \hat{\mathbf{i}}_1 - \mathbf{c},\ \hat{\mathbf{i}}_2 - \mathbf{c},\ \ldots\ldots,\ \hat{\mathbf{i}}_M - \mathbf{c} \right\} \qquad (4)$$

$\mathbf{P}$ is $N \times M$, where N is the number of pixels in each image and M is the total number of images in the set. To compute eigenvectors of the image set we define the *covariance matrix*:

$$\mathbf{Q} \triangleq \mathbf{P}\,\mathbf{P}^{\mathrm{T}} \qquad (5)$$

$\mathbf{Q}$ is $N \times N$, clearly a very large matrix since a large number of pixels constitute an image. The eigenvectors $\mathbf{e}_k$ and the corresponding eigenvalues λ_k of $\mathbf{Q}$ are to be determined by solving the well-known eigenstructure decomposition problem:

$$\lambda_k\, \mathbf{e}_k \;=\; \mathbf{Q}\, \mathbf{e}_k \qquad (6)$$

The calculation of the eigenvectors of a matrix as large as $\mathbf{Q}$ is computationally intensive. Fast algorithms for solving this problem have been a topic of active research in the area of image coding/compression and pattern recognition (see [Oja 83]). We have used a fast implementation [Murase and Nayar 92] of the algorithm proposed by Murakami and Kumar [Murakami and Kumar 82]. On a Sun IPX workstation this implementation enables us to compute, for example, 20 eigenvectors of a set of 100 images (each 128x128 in size) in about 3 minutes, and 20 eigenvectors of a set with 1000 images in less than 4 hours.

The result of eigenstructure decomposition is a set of eigenvalues $\{\,\lambda_k \mid k = 1, 2, \ldots, K\,\}$ where $\{\,\lambda_1 \geq \lambda_2 \geq \ldots\ldots \geq \lambda_K\,\}$, and a corresponding set of orthonormal eigenvectors $\{\,\mathbf{e}_k \mid k = 1, 2, \ldots, K\,\}$. Note that each eigenvector is of size N, i.e. the size of an image. These K eigenvectors constitute our eigenspace; it is an approximation to the complete eigenspace with N dimensions. In our experiments we have used eigenspaces of less that 20 dimensions.

3.3 Parametric Eigenspace Representation

We now represent the visual workspace as a function of the robot coordinates $\mathbf{q}$. The result is a manifold in the K-dimensional eigenspace computed above. This representation is called the *parametric eigenspace*[4].

Each learning sample $\hat{\mathbf{i}}_j$ in the image set is projected to the eigenspace by first subtracting the average image $\mathbf{c}$ from it and finding the inner product of the result with each of the eigenvectors (dimensions) of the eigenspace. The result is a point $\mathbf{f}_j$:

$$\mathbf{f}_j \;=\; \left[\mathbf{e}_1, \mathbf{e}_2, \ldots\ldots, \mathbf{e}_K\right]^{\mathrm{T}} \left(\hat{\mathbf{i}}_j \;-\; \mathbf{c}\right) \qquad (7)$$

By projecting all the learning samples in this manner, a set of discrete points are obtained in eigenspace. Since consecutive object images are strongly correlated, their projections in eigenspace are close to one another. Hence, the discrete points obtained by projecting all the learning samples can be assumed to lie on a manifold that represents all possible object appearances for all possible manipulator coordinates. The discrete points are interpolated to obtain this manifold. In our implementation, we have used a standard quadratic B-spline interpolation algorithm [Rogers 90]. The resulting manifold can be expressed as:

$$\mathbf{f}\,(\mathbf{q}) \;=\; \mathbf{f}\,(\,q_1, q_2, \ldots\ldots, q_m\,) \qquad (8)$$

This manifold is in a low-dimensional space and therefore is a compact *continuous* representation of object appearance as a function of manipulator coordinates $\mathbf{q}$. In practice, the number of end-effector DOFs used for positioning and tracking can vary.

The above representation has an important property. Consider two images $\hat{\mathbf{i}}_r$ and $\hat{\mathbf{i}}_s$ that belong to the image set used to compute an eigenspace. Let the points $\mathbf{f}_r$ and $\mathbf{f}_s$ be the projections of the two images in eigenspace. It is well-known in pattern recognition theory [Oja 83] [Murase and Nayar 92] that the distance between the two points in eigenspace is an approximation to the correlation between the two

[4] The parametric eigenspace representation was introduced in [Murase and Nayar 93] for object recognition and pose estimation.

brightness images:

$$\| \hat{\mathbf{i}}_r - \hat{\mathbf{i}}_s \|^2 \approx \| \mathbf{f}_r - \mathbf{f}_s \|^2 \qquad (9)$$

The closer the projections are in eigenspace, the more similar are the images in l^2. Therefore, the eigenspace is optimal for computing the correlation between images. It is this property that motivates us to develop a learning methodology based on principal component analysis.

4 Visual Positioning

We now discuss the automatic positioning of a manipulator at its desired coordinates $\tilde{\mathbf{q}}$ with respect to the viewed object. A brute force solution would be to compare an unknown input image with all images corresponding to different discrete learning coordinates. Such an approach is equivalent to exhaustive template matching. Clearly, this is impractical from a computational perspective given the large number of learning images obtained. Further, the input image may not correspond exactly to any one of the learning images, i.e. the current displacement may lie in between the discrete ones used for learning.

The parametric eigenspace representation enables us to accomplish image-displacement mapping in a very efficient manner. Since the eigenspace is optimal for computing the correlation between images, we can project the current image to the eigenspace and simply look for closest point on the manifold. Also, since the manifold is continuous, displacements that are not exactly the ones used for learning can be estimated.

Let the robot's current position be $\mathbf{q}_c$ and the corresponding normalized image be $\hat{\mathbf{i}}_c$. The average $\mathbf{c}$ of the learning set is subtracted from $\hat{\mathbf{i}}_c$ and the resulting vector is projected to eigenspace to obtain the point:

$$\mathbf{f}_c = [\mathbf{e}_1, \mathbf{e}_2, \ldots.., \mathbf{e}_K]^{\mathrm{T}} (\hat{\mathbf{i}}_c - \mathbf{c}) \qquad (10)$$

The positioning problem then is to find the minimum distance d between $\mathbf{f}_c$ and the manifold $\mathbf{f}(\mathbf{q})$:

$$d = \overset{\min}{\mathbf{q}} \| \mathbf{f}_c - \mathbf{f}(\mathbf{q}) \| \qquad (11)$$

If d is within some pre-determined threshold value (selected based on the noise characteristics of the image sensor), we conclude that the manipulator lies within the range of coordinates used for learning. Then, positioning is reduced to finding the coordinate $\mathbf{q}_c$ on the manifold corresponding to the minimum distance d. In practice, the manifold is stored in memory as a list of K-dimensional points obtained by densely re-sampling $\mathbf{f}(\mathbf{q})$. The closest point to $\mathbf{f}_c$ on $\mathbf{f}(\mathbf{q})$ can be determined either by exhaustive search (if the list of manifold points is small), binary search, or indexing. In [Nene and Nayar 93] we have developed an algorithm that results in near-constant search time of approximately 20 msec on a Sun IPX workstation. Alternatively, $\mathbf{q}_c$ can be determined from $\mathbf{f}_c$ by training a regularization network of the type described in [Poggio and Girosi 90].

5 Positioning Experiments

We have conducted several positioning experiments. For lack of space, we present only two of these results. All experiments were conducted using the Adept robot and hand-eye system shown in Figure 1. Figure 4(a) shows a printed circuit board. The box shown is the image area (128x128 pixels) used for learning and positioning. Note that the image is rather complex and includes a variety of subtle features. Images were acquired using an Analogics digitizer board. In this experiment, robot displacements were restricted to two dimensions (x and y). A total of 256 images were obtained by moving the robot to 16x16 equally spaced discrete points within a 2cm x 2cm region around the desired position. A 15-dimensional eigenspace was computed using the 256 images. Each learning image was then projected to eigenspace and the 256 resulting points were interpolated to obtain a manifold with two parameters, namely, x and y. Since we are unable to display the manifold in 15-D space, we have shown it (see Figure 4(b)) in a 3-D space where the dimensions are the three most prominent eigenvectors of the eigenspace. The complete learning process including image acquisition, eigenspace computation, and manifold interpolation took approximately 11 minutes on a Sun IPX workstation. The parametric eigenspace is stored in memory as a set of 251x251 = 63001 points obtained by resampling the continuous manifold. A robot displacement (x,y) is stored with each manifold point.

Next, the accuracy of the positioning algorithm was tested. In these experiments, the robot was displaced by a random distance from its desired position. The random positions were uniformly distributed within the 2cm x 2cm region used for learning. Note that the random positions are generally not the same as any of the positions used while learning. The positioning algorithm was then used to estimate the robot's displacement from its desired position. This process was repeated 1000 times, each time computing the euclidean distance (error) between the robot location after positioning and the desired location. A histogram of positioning errors is shown in Figure 4(c). The average of the absolute positioning error is 0.676 mm and standard deviation is 0.693 mm. The positioning accuracy was dramatically improved by simply using a larger number of learning images. Figure 4(d) shows the error histogram for 21x21 = 441 learning images obtained within the same 2cm x 2cm displacement region. In this case, the learning process was completed in approximately 30 minutes. The average absolute error was found to be 0.151 mm and standard deviation 0.107 mm. This reflects very high positioning accuracy, sufficient for reliable insertion of a circuit chip into its holder. This task was in fact accomplished with high repeatability using the gripper of the hand-eye system.

Similar experiments were conducted for the object shown in Figure 4(e). In this case, however, three displacement parameters were used, namely, x, y, and θ (rotation in the x-y plane). During learning the x and y parameters were each varied within a $\pm$1cm range, and θ within a $\pm$10deg range for each (x,y) displacement. A total of 11x11x11 = 1331 learning images were obtained and a 5-D eigenspace computed. The parametric eigenspace representation in this case is a three-parameter manifold in 5-D space. In Figure 4(f) a projection of this manifold is shown as a surface (x and y are the parameters, while $\theta = 0$) in 3-D eigenspace. Again, this reduced representation is used only for the purpose of display. The actual manifold is stored in memory as a set of 65x65x65=274625 points. In this case, the entire learning process took approximately 5 hours.

Once again, 1000 random displacements were used in the positioning experiments. The absolute euclidean positioning errors in x-y space are illustrated by the histogram in Figure 4(g). An average absolute error of 0.291 mm and standard deviation of 0.119 mm were computed. The absolute errors for θ were computed separately and found to have a mean value of 0.56 deg and deviation of 0.45 deg. These results again indicate high positioning accuracy. Figure 4(h) shows that positioning accuracy is only marginally improved for this particular object by doubling the eigenspace dimensionality. Here, 10 eigenvectors were computed to obtain a more descriptive representation of object appearance at the cost of additional memory usage. The positioning errors in this case have a mean of 0.271 mm and deviation of 0.116 mm, and the angular errors a mean of 0.44 deg and deviation of 0.33 deg. This accuracy was verified by successful insertions of a peg in the hole of the object.

6 Visual Tracking

The visual processing aspects of tracking are identical to that of positioning. The primary difference is in the selection of learning parameters. In tracking applications successive images may be assumed to be close to one another since the manipulator is in the process of tracking the object and hence always close to the desired position. This implies that fewer learning samples are generally needed. For any new image acquired the positioning algorithm is used to determine the error $\mathbf{q}_c$ in robot coordinates. This error may be used as input to a position control system as shown in Figure 3. The control law may vary from a simple PID controller to more sophisticated adaptive controllers that incorporate the dynamics of the manipulator as well as delays introduced by the visual processing. The position controller generates a reference point $\mathbf{q}_r$ for the low-level robot actuator controller.

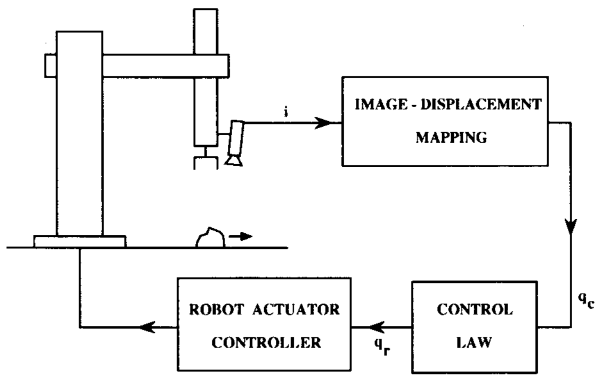


Figure 3: Schematic diagram of the visual tracking system.

7 Tracking Experiments

Figure 5(a) shows an object we have used to test the tracking algorithm. The box illustrates the 96x96 pixel image region used for learning and tracking. As in the previous experiment, robot displacements were confined to three dimensions (x,y,θ). A total of 13x13x13 = 2197 images were acquired during the learning stage by using robot displacements within $x = \pm$ 1cm, $y = \pm$ 1cm, $\theta = \pm$ 10deg. A 10-D eigenspace was used to represent the three-parameter manifold. A projection of the manifold (using θ= 0) in 3-D is shown in Figure 5(b).

Each cycle of the tracking algorithm involves the digitization of an input image, transfer of image data from the digitizer to the workstation, projection of the input image into eigenspace, search for the closest manifold point, computation of reference coordinates using a control law, and communication of the reference coordinates to the robot controller. In the present implementation, all computations are done on the Sun IPX workstation without the use of any customized image processing hardware. The total cycle time at present is approximately 250msec yielding a control rate of 4Hz. This restricts our present experiments to objects moving at relatively slow speeds (approximately 0.5cm/sec). It may be noted that this is merely a limitation of the current implementation. All computations involved in the image-displacement mapping are simple and can be easily done at frame-rate (30Hz) with a single frame-time delay using a more powerful workstation such as a DEC Alpha machine, or inexpensive image processing hardware such as a standard i860 board.

The present control law is based on a simple interpolation/prediction scheme to facilitate smooth manipulator motion. The tracking accuracy was determined by moving the object at known velocity along a circle using a motorized turntable (Figure5(a)). The turntable was rotated through 90 deg, moving the object through a total distance of 19 cm. In Figures5(c)-(e) the desired and actual coordinates of the robot are plotted as a function of time. The deviations and lags that result while tracking are attributed mostly to delays introduced by the vision computations and the simple control scheme used. Our current work is geared towards overcoming these limitations to achieve higher tracking speeds. Also, the experiments reported here were confined to three end-effector parameters (x,y,θ). We are currently exploring extensions to positioning/tracking problems that involve more than three degrees of freedom.

8 Real-Time Object Recognition

The learning and recognition framework presented in this paper has wide-spread applications in computer vision. As an example, we implemented a recognition system with 20 objects in its database (see Fig.6). These objects vary from smoothly curved shapes with uniform reflectance, to fairly complex shapes with intricate textures and specularities. Developing CAD models of such objects could prove extremely cumbersome and time-consuming. Both learning and recognition are done in a laboratory environment where illumination remains more or less unchanged. In this case, learning involves acquiring an image set of each object by varying pose [Murase and Nayar 93]. Each object image set includes 72 learning images (5 degree rotations about a stable configuration of the object), resulting in a complete image set of 1440 images. Each object is represented by a curve in a 20-D eigenspace parametrized by pose [Murase and Nayar 92]. The entire learning process, including, image acquisition, computation of eigenvectors, and construction of appearance curves was completed in less than 12 hours using a Sun SPARC workstation.

The recognition system automatically detects significant changes in the scene, waits for the scene to stabilize, and then digitizes an image. In the present implementation, objects are presented to the system one at a time and a dark background is used to alleviate object segmentation. The complete recognition process, including, segmentation, scale and brightness normalization of object regions, image projection in universal eigenspace, and search for the closest object and pose is accomplished in less than 1 second on the Sun workstation. The robustness of this system was tested using 320 test images of the 20 objects taken at randomly selected but known poses of the objects. All test images were correctly identified by the system. A histogram of the absolute pose error is shown in Fig.6(c); the average and standard deviation of the absolute pose error were found to be 1.59 degrees and 1.53 degrees, respectively. In related work [Murase and Nayar 94], we have also used the parametric eigenspace representation to determine illumination conditions in a structured environment that would optimize the performance of a recognition system such as the one described above.

9 SLAM: A Software Library for Appearance Matching

As is evident from the above results, the parametric eigenspace representation can serve as the basis for solving a variety of real-world vision problems. In view of this, we have developed the software package SLAM [Nene et al. 94] as a general tool for appearance learning and recognition problems in computational vision. The package is coded in C++ and uses advanced object-oriented programming techniques to achieve high space/time efficiency. The package has four primary modules: image manipulation, subspace computation, manifold generation, and recognition. Image manipulation includes image segmentation, scale and brightness normalization, image-vector conversions, and provides tools for maintaining large image databases. Subspace computation, the second module, computes eigenvectors and eigenvalues of large image sets using the method outlined in [Murakami and Kumar 82]. The manifold generation module can be used for projecting image (or feature) sets to subspaces, B-spline interpolation [Rogers 90] of subspace projections to produce multivariate manifolds, dense resampling of manifolds, and orthogonalization of multiple subspaces. Finally, the recognition module includes efficient search implementations [Nene and Nayar 93] that find manifold points which lie closest to novel input projections. All four modules can be accessed via an intuitive graphical interface that has been built on X/Motif. We are currently in the process of licensing SLAM to several academic and industrial research institutions.

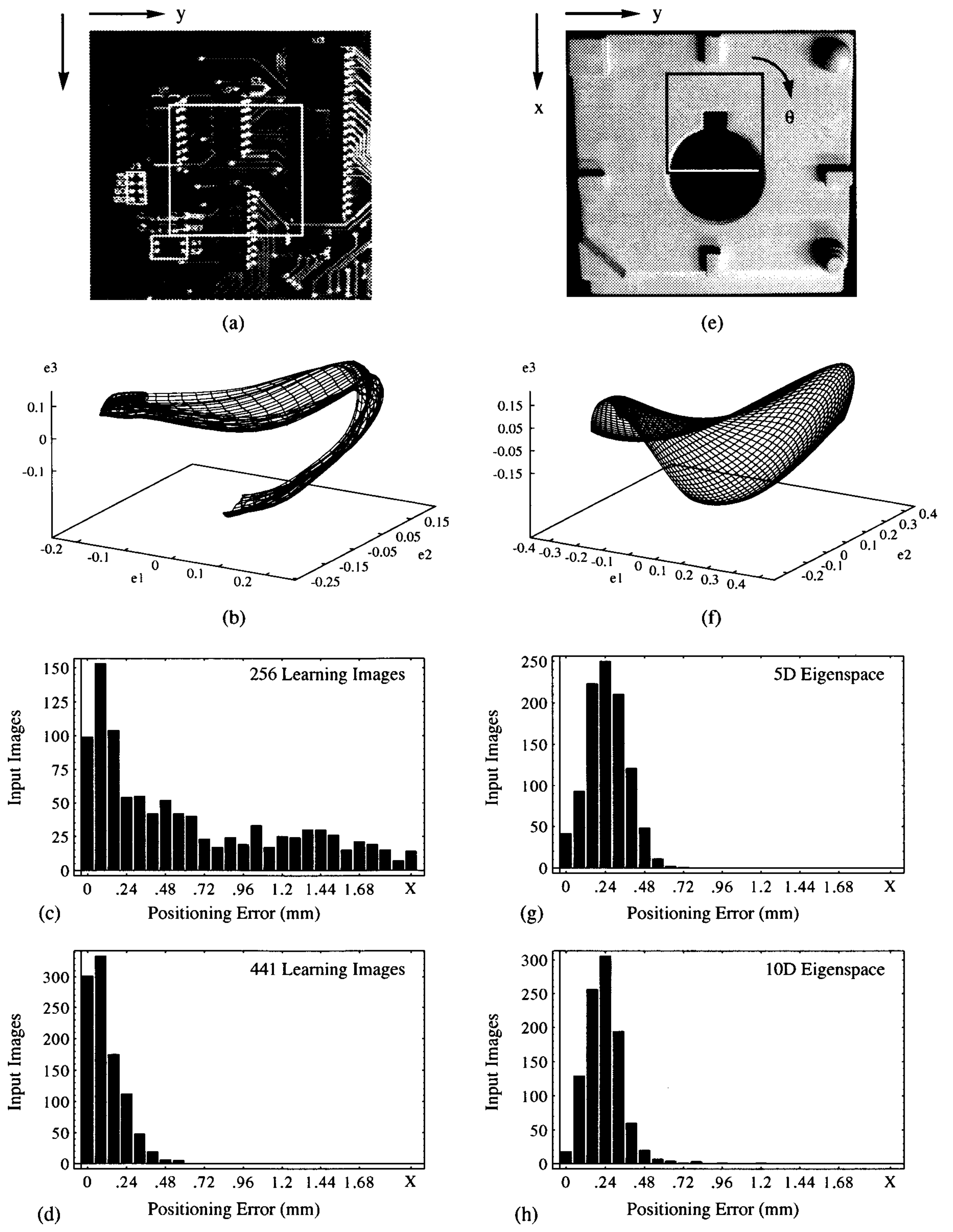
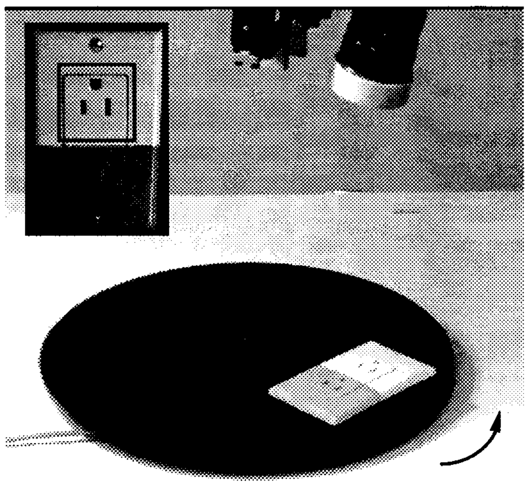
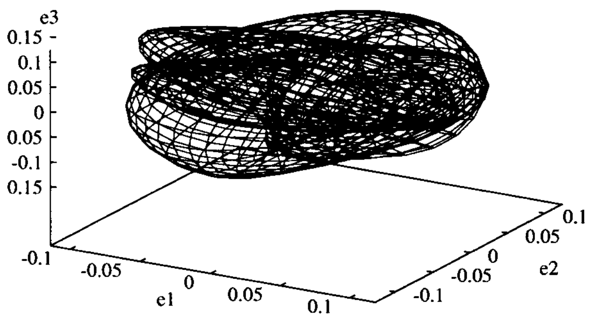
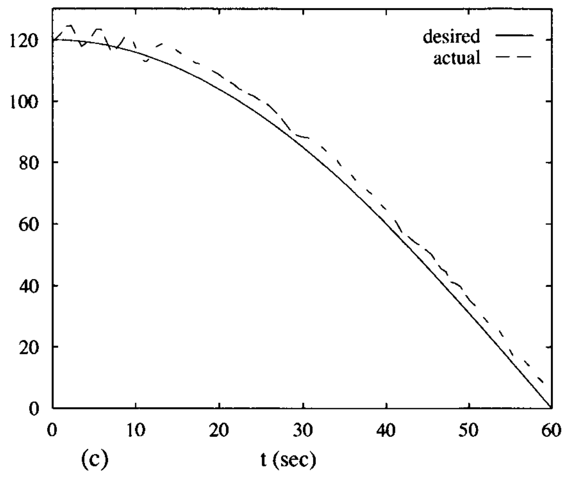


Figure 4: **Visual positioning experiment:** printed circuit board. (a) Image window (white box) used for learning and positioning. (b) Parametric eigenspace representation of the visual workspace displayed in 3-D. Robot displacements are in two dimensions (x and y). Histograms of absolute positioning error (in mm) for (c) 256 learning images and (d) 441 learning images. (e) Object with hole and slot. (f) Parametric eigenspace representation displayed in 3-D. Displacements are in three dimensions (x, y, θ). Histograms of absolute positioning error (in mm) for (g) 5-D eigenspace and (h) 10-D eigenspace.

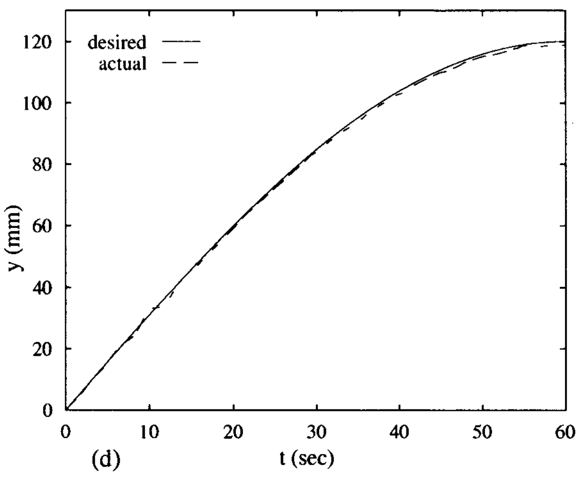

(a)



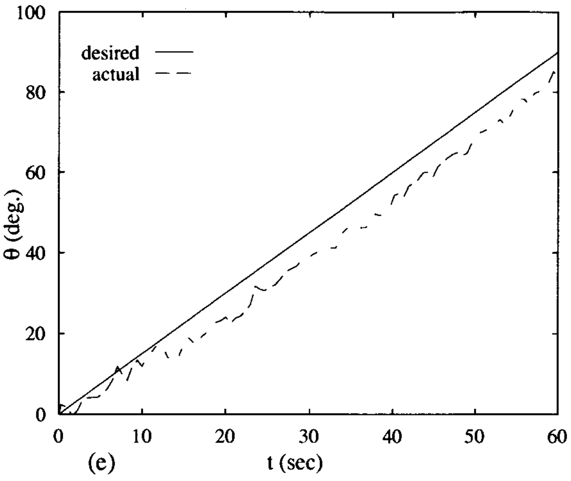
(b)



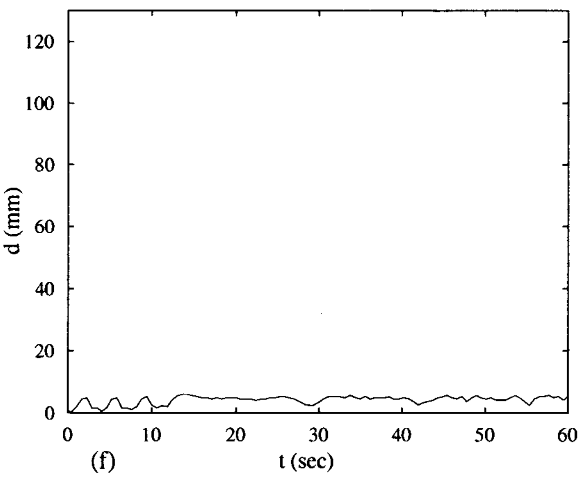
(c)



(d)

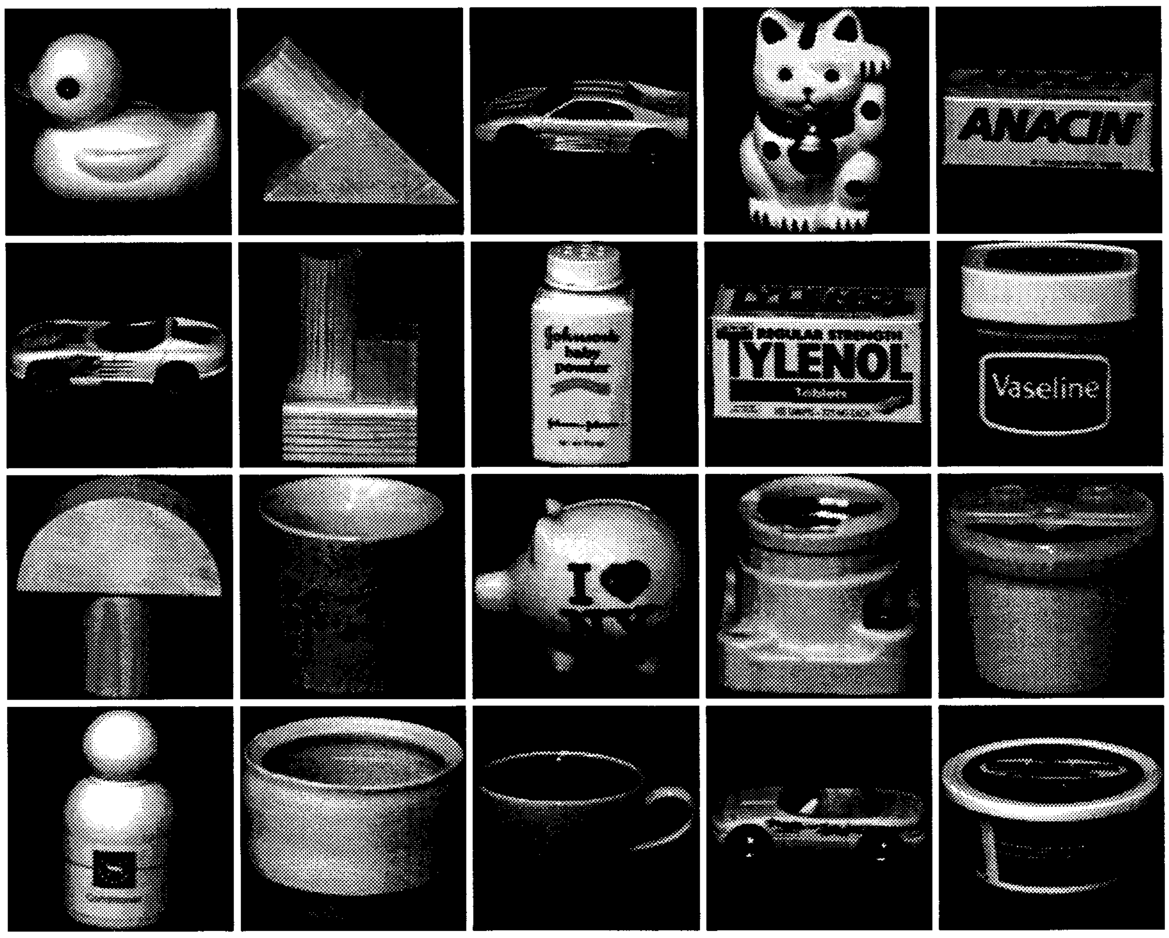


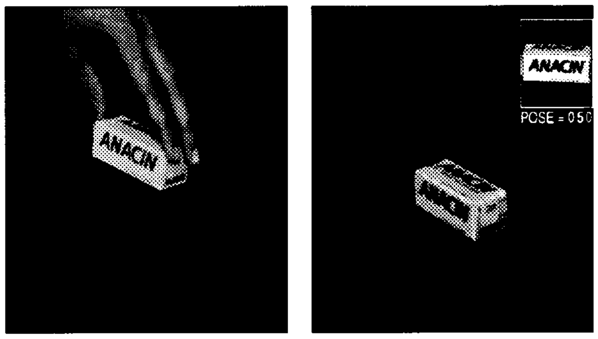
(e)



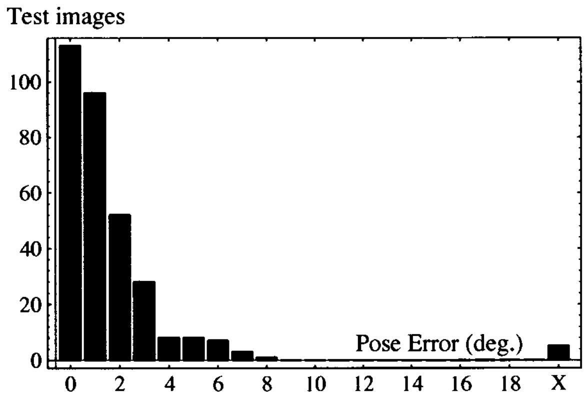
(f)

Figure 5: **Visual tracking experiment:** moving electric socket. (a) Image window (black box shown in inset). (b) Parametric eigenspace representation. Desired and actual coordinates: (c) $x(t)$; (d) $y(t)$; and (e) $\theta(t)$. (f) Tracking distance error $d(t)$.


(a) Object set



(b) Real-time recognition



(c) Pose estimation accuracy

Figure 6: **Real-time recognition system:** The system has 20 objects in its database, each represented as a manifold parametrized by pose for a single stable configuration. A complete recognition and pose estimation cycle takes less than 1 second on a Sun IPX workstation without the use of customized hardware.

References

[Albus 75] J. S. Albus, *A new approach to manipulator control: The cerebellar model,* Journal of Dynamic Systems Measurement and Control, Vol. 97, pp. 220-227, Sept. 1975.

[Allen et al. 92] P. K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman, *Trajectory Filtering and Prediction for Automated Tracking and Grasping of a Moving Object,* Proc. of IEEE Intl. Conf. on Robotics and Automation, Nice, 1992, pp. 1850-1856.

[Feddema et al. 91] J. Feddema, C.S.G. Lee, and O. Mitchell, *Weighted selection of image features for resolved rate visual feedback control,* IEEE Trans. on Robotics and Automation, Vol. 7, No. 1, pp. 31-47, Feb. 1991.

[Hashimoto et al. 91] K. Hashimoto, T. Kimoto, and H. Kimura, *Manipulator Control with Image-Based Visual Servo,* Proc. of IEEE Intl. Conf. on Robotics and Automation, 1991, pp. 2267-2271.

[Hummel 79] R. A. Hummel, "Feature Detection Using Basis Functions," *Computer Graphics and Image Processing,* Vol. 9, pp. 40-55, 1979.

[Koivo and Houshangi 91] A. Koivo and N. Houshangi, *Real-time vision feedback for servoing robotics manipulator with self-tuning controller,* IEEE Trans. on Systems, Man, and Cybernetics, Vol. 21, No. 1, pp. 134-142, Feb. 1991.

[Kuperstien 87] M. Kuperstien, *Adaptive visual-motor coordination in multijoint robots using parallel architecture,* Proc. of IEEE Intl. Conf. on Robotics and Automation, Raleigh, N.C., 1987, pp. 1595-1602.

[Luo et al. 88] R. Luo, R. Mullen, and D. Wessel, *An Adaptive Robotic Tracking System using Optical Flow,* Proc. of IEEE Intl. Conf. on Robotics and Automation, 1988, pp. 568-573.

[Mel 87] B. W. Mel, *MURPHY: A robot that learns by doing,* AIP Proc. of Neural Information Processing System Conference, Denver, CO, 1987.

[Miller 89] W. T. Miller, *Real-time application of neural networks for sensor-based control of robots with vision,* IEEE Trans. on Systems, Man, and Cybernetics, Vol. 19, No. 4, pp. 825-831, July/August, 1989.

[Miller 87] W. T. Miller, *Sensor-based control of robotic manipulators using a general learning algorithm,* IEEE Journal of Robotics and Automation, Vol. RA-3, No. 2, pp. 157-165, April, 1987.

[Murakami and Kumar 82] H. Murakami and V. Kumar, "Efficient Calculation of Primary Images from a Set of Images," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. 4, No. 5, pp. 511-515, Sept. 1982.

[Murase and Nayar 93] H. Murase and S. K. Nayar, "Learning Object Models from Appearance," *Proc. of AAAI,* Recognition," Washington D. C., July 1993.

[Murase and Nayar 92] H. Murase and S. K. Nayar, "Visual Learning and Recognition of 3D Objects from Appearance," *International Journal of Computer Vision,* in press. Also Tech. Rep. CUCS-054-92, Dept. of Computer Science, Columbia Univ.

[Murase and Nayar 94] H. Murase and S. K. Nayar, "Illumination Planning for Object Recognition in Structured Environments," *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition,* Seattle, June 1994.

[Nayar et al. 94] S. K. Nayar, H. Murase, and S. A. Nene, "Learning, Positioning, and Tracking Visual Appearance," *Proc. of IEEE Intl. Conf. on Robotics and Automation,* San Diego, May 1994.

[Nene and Nayar 93] S. A. Nene and S. K. Nayar, "Binary Search Through Multiple Dimensions," Technical Report CUCS-26-93, Dept. of Computer Science, Columbia Univ., August, 1993.

[Nene et al. 94] S. A. Nene, S. K. Nayar, H. Murase, "SLAM: A Software Library for Appearance Matching,," *Proc. of ARPA Image Understanding Workshop,* Monterey, Nov 1994.

[Oja 83] E. Oja, *Subspace methods of Pattern Recognition,* Res. Studies Press, Hertfordshire, 1983.

[Papanikolopoulos et al. 91] N. Papanikolopoulos, P. Khosla, and T. Kanade, *Adaptive robotic visual tracking,* Proc. of Automatic Control Conf., 1991.

[Poggio and Girosi 90] T. Poggio and F. Girosi, "Networks for Approximation and Learning," *Proc. of the IEEE,* Vol. 78, No. 9, pp. 1481-1497, September 1990.

[Rogers 90] D. F. Rogers, *Mathematical Elements for Computer Graphics,* 2nd ed., McGraw-Hill, New York, 1990.

[Turk and Pentland 91] M. A. Turk and A. P. Pentland, "Face Recognition Using Eigenfaces," *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition,* pp. 586-591, June 1991.

[Walter et al. 90] J. Walter, T. Martinez, and K. Schulten, *Industrial robot learns visuo-motor coordination by means of neural-gas network,* Proc. of Intl. Joint Conf. on Neural Networks, June, 1990.

[Weiss et al. 87] L. Weiss, A. Sanderson, and C. Neuman, *Dynamic sensor-based control of robots with visual feedback,* IEEE Journal of Robotics and Automation, Vol. RA-3, No. 5, pp. 404-417, Oct. 1987.