[9] G. Motyl, F. Chaumette, and J. Gallice, "A camera and laser stripe in sensor based control," *Second Int. Symp. on Measurement and Control in Robotics*, AIST Tsukuba Research Center, Japan, Nov. 1992, pp. 685–692.

[10] G. Motyl, P. Martinet, J. Gallice, "Visual servoing with respect to a target sphere using a camera/laser-stripe sensor," in *1993 Int. Conf. on Advanced Robotics, ICAR'93*, Tokyo, Japan, Nov. 1993, pp. 591–596.

[11] C. Samson, M. Le Borgne, and B. Espiau, *Robot Control: The Task Function Approach.* Cambridge, U.K.: Oxford Univ. Press, 1991.

[12] J. P. Urban, G. Motyl, and J. Gallice, "Real-time visual servoing using controlled illumination," *Int. J. Robot. Res.*, vol 13, no. 1, pp. 93–100, Feb. 1994.

[13] S. Venkatesan and C. Archibald, "Real-time tracking in five degrees of freedom using two wrist-mounted laser range finders," in *IEEE Int. Conf. on Robotics and Automation*, pp. 2004–2010, 1990.

[14] M. R. Ward, L. Rossol, S. W. Holland, and R. Dewar, "CONSIGHT: A practical vision based robot guidance system," in *Proc 9th Int. Symp. on Industrial Robotics*, Washington, D.C., Mar. 1979, pp. 195–211.

[15] L. E. Weiss, A. C. Sanderson, and C. P. Neuman, "Dynamic sensor-based control of robots with visual feedback," *IEEE J. Robot. Automat.*, vol. RA-3, no. 5, pp. 404–417, Oct. 1987.

Subspace Methods for Robot Vision

Shree K. Nayar, Sameer A. Nene, and Hiroshi Murase

Abstract—In contrast to the traditional approach, visual recognition is formulated as one of matching appearance rather than shape. For any given robot vision task, all possible appearance variations define its visual workspace. A set of images is obtained by coarsely sampling the workspace. The image set is compressed to obtain a low-dimensional subspace, called the eigenspace, in which the visual workspace is represented as a continuous appearance manifold. Given an unknown input image, the recognition system first projects the image to eigenspace. The parameters of the vision task are recognized based on the exact location of the projection on the appearance manifold. An efficient algorithm for finding the closest manifold point is described. The proposed appearance representation has several applications in robot vision. As examples, a precise visual positioning system, a real-time visual tracking system, and a real-time temporal inspection system are described.

Index Terms—Visual workspace, parametric eigenspace representation, learning appearance manifolds, image recognition, nearest neighbor, visual positioning, real-time tracking, temporal inspection.

I. 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

S. K. Nayar and S. A. Nene are with the Department of Computer Science, Columbia University, New York, NY 10027 USA.

H. Murase is with the NTT Basic Research Laboratory, Kanagawa 243-01, Japan.

uncertainties inherent to a task, react to a varying environment, and gracefully recover from failures. In order for the robot to interact with objects in its workspace, it requires a-priori models of the objects. Traditionally, robot vision systems have heavily relied on shape (CAD) models [4].

Will shape representation suffice? After all, most vision applications deal with brightness images that are functions not only of shape but also other intrinsic scene properties such as reflectance and perpetually varying factors such as illumination. This observation has motivated us to take an extreme approach to visual representation. What we seek is not a representation of geometry but rather *appearance* [20], encoded in which are brightness variations caused by three-dimensional shape, surface reflectance properties, illumination conditions, and the parameters of the robot task. Given the number of factors at work, it is immediate that an appearance representation that captures all possible variations is simply impractical. Fortunately, there exist a wide collection of robot vision applications where pertinent variables are few and hence compact appearance representation in a low-dimensional *subspace* is indeed practical.

A problem of substantial relevance to robotics is *visual servoing;* 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 through an unknown trajectory. We use the visual servoing problem to describe our appearance based approach. To place our approach in perspective, we review existing methods for servoing. All of these 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 objective 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 [33], [5], [10], [7] to optical flow estimates [26], [13], [3] and object location estimates obtained using stereo [2]. 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 (e.g., rotation and translation with respect to the manipulator).

The second category of servoing methods includes a learning component. In the learning stage, the mapping between image features and robot coordinates is generated prior (off-line) to positioning or tracking. 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 servoing. These methods differ from each other primarily in the type of learning algorithm used. The learning strategies vary from neural-like networks [11], [14], [17], [32] to table lookup mechanisms such as the cerebellar model articulation controller (CMAC) [1], [16].

Our appearance based approach to robot vision offers a solution to servoing that differs from previous work in two significant ways; (a) the method uses raw brightness images directly without the computation of image features, and (b) the learning algorithm is based on principal component analysis [25], [6] 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 displacing the robot's end-effector (hand-eye system). Since all images in the set are of the same object, consecutive images tend to be strongly correlated. This allows us to compress the image set using principal component analysis to obtain a low-dimensional subspace, called the *eigenspace*. Variations in object images due to robot displacements are represented in the form of a parametrized manifold in eigenspace. The manifold is a continuous representation of what we refer to as the *visual workspace* of the task (servoing, in this case).

During visual positioning or tracking, 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. An efficient algorithm for finding the closest manifold point in eigenspace is described. It is worth emphasizing that 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 parameters. Several servoing experiments are conducted using a hand-eye system mounted on an Adept robot. The results demonstrate high performance in both accuracy and speed. To monstrate the scope of the subspace approach, we introduce a new technique called temporal visual inspection. The hand-eye system is swept over a complex manufactured part to acquire an appearance manifold (model) of the part that is parametrized by travel time. During inspection, images of a novel part are projected to eigenspace and compared in real-time with the model manifold.

Our experimental results demonstrate that the techniques underlying appearance modeling and matching are general. This has led to the development of a comprehensive software package [23] for appearance matching called SLAM[1]. Several extensions to the methods presented in this paper are discussed in [22].

II. APPEARANCE BASED APPROACH

Fig. 1 shows the *hand-eye* system we have used to implement positioning, tracking and inspection. The manipulator used is a 5 degree-of-freedom Adept robot that is interfaced with a workstation. A CCD camera is mounted adjacent to the robot gripper and provides images of the object. We define the variables of a vision task as the *visual degrees of freedom* (DOF), $q = [q_1, q_2, \ldots, q_m]^T$, where, m represents the manipulator's degrees of freedom (DOF) used in a task at hand. To describe the basic principles underlying the subspace based approach, we shall use visual servoing as the task of interest. 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 large fixed window, within the complete image, which includes sufficient object detail as shown in Fig. 2(a). Alternatively, as illustrated in Fig. 2(b), the contents of several windows of fixed sizes and shapes that are scattered in the image can be concatenated and treated as a single visual input. The positions, shapes, and sizes of the these windows are selected to ensure that the visual input is descriptive and is sensitive to variations in the DOF q. This visual input, also referred to as appearance, is a vector $i = [i_1, i_2, \ldots, i_N]^T$ obtained by reading brightness values from the selected window (or set of windows) in a raster scan fashion.

The visual appearance i for any given robot position q depends on the three-dimensional shape of the object, its reflectance properties, the illumination conditions, and the robot coordinates with respect to the object. Shape and reflectance are intrinsic properties of a rigid
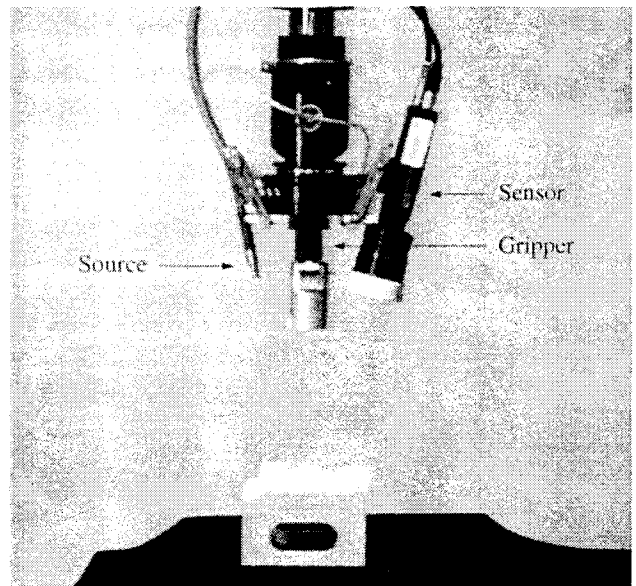

Fig. 1. The hand-eye system used for visual positioning, tracking and inspection. The end-effector includes a gripper and an image sensor. In applications where ambient lighting is not diffuse, a light source may be mounted on the end-effector to achieve illumination invariance.

object that do not change during a task. In order to avoid possible variations in illumination, we have used a light source that is also mounted on the end-effector. In our setup (see Fig. 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. Further, since the source and sensor directions are fixed with respect to one another, the appearance of an object depends only on its position and orientation with respect to the end-effector and not its position and orientation in the manipulator's workspace. Placing the source close to the sensor also minimizes shadows in the image.

Industrial tasks often involve smooth objects that produce strong specular reflections in the image. Specular highlights can be used to our advantage as they cause the visual input vector i to be sensitive to object pose. However, they are often a curse as they produce undesirable effects such as image saturation. In such cases, two cross-polarized filters can be used, one in front of the source and the other the sensor. This causes the illumination of the scene to be linearly polarized. Since specular reflections tend to preserve the polarization characteristics of the incident light, they are blocked by the cross-polarized filter appended to the sensor [15]. Diffuse reflections, on the other hand, tends to be unpolarized even under polarized illumination and hence are allowed to pass through to the sensor. The result is an image that is more or less devoid of specularities.

Simple normalizations can be applied to the input vector i to enhance the robustness of visual processing.[2] It is desirable that object appearance be unaffected by variations in the intensity of the hand-source or the aperture of the imaging system. This can be achieved by normalizing each acquired image such that the total energy contained within is unity

$$\hat{i}_j = i_j / \|i_j\|.$$

Having taken care of illumination variations, we are left with the coordinates q of the end-effector with respect to the object. Judicious selection of the image windows can ensure that the each task

---

[1] Information on the SLAM software package can be obtained by sending a query to **slam@cs.columbia.edu**.

[2] In the case of object recognition, each object region is segmented from the scene and scale normalized [20] to fit a predetermined image size. This ensures that the recognition system is invariant to magnification, i.e., the distance of the object from the image sensor.
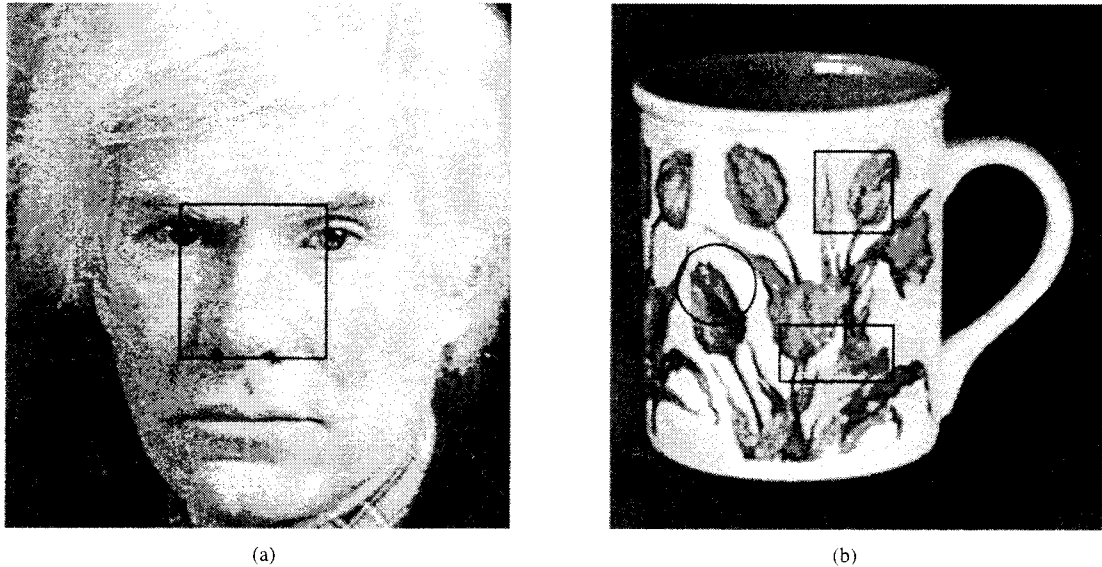
(a)   (b)

Fig. 2. (a) Each image vector $\mathbf{i}$ is obtained by reading pixel brightness values from a large image window (box) of fixed size and position. (b) Alternatively, the contents of several windows of fixed sizes and shapes, scattered in the image, can be concatenated and treated as a single image vector.

coordinate $\mathbf{q}$ produces a unique visual appearance $\hat{\mathbf{i}}$. In a given application, $\mathbf{q}$ has lower and upper bounds and its continuous set of values within these bounds map to a continuous domain of images $\hat{\mathbf{i}}(\mathbf{q})$. This range of appearances is what we refer to as the *visual workspace* of the task. Our approach is to acquire an image set by coarsely sampling the visual workspace and then produce a compact subspace representation of the image set that can be used not only to recognize the discrete appearances in the image set but also those that lie in between the ones in the set, i.e., a continuous representation of the entire visual workspace. We show that once such a representation is computed, the task coordinates $\mathbf{q}$ can be efficiently determined for any visual input $\hat{\mathbf{i}}$.

Several advantages result from the above approach. (a) In contrast to popular CAD model based techniques, 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 $\hat{\mathbf{i}}$. (b) The task coordinates $\mathbf{q}$ 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. It is only required that the camera parameters remain unchanged between the stages of learning and using the visual workspace. 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.

III. LEARNING THE VISUAL WORKSPACE

For any given application, the visual workspace is coarsely sampled by varying the task DOF $\mathbf{q}$ in increments. This sampling may be uniform or nonuniform. To ensure high accuracy, one may choose a sampling frequency that increases as $\mathbf{q}$ approaches the desired coordinate. Let the number of discrete samples obtained for each degree of freedom q_l be R_l. Then the total number of images acquired is $M = \prod_{l=1}^{m} R_l$. The complete image set is

$$\{\hat{\mathbf{i}}_1, \dots, \hat{\mathbf{i}}_2, \dots, \hat{\mathbf{i}}_M\}. \tag{1}$$

Note that the input vectors $\hat{\mathbf{i}}_j$ represent unprocessed (barring possible scale and brightness normalizations) brightness images. Alternatively, processed images such as smoothed images, image derivatives, or even the power spectrum of each image may be used. In applications

that employ depth sensors, the inputs could be range maps. Here, for the purpose of description we use raw brightness images, bearing in mind that visual workspaces can in principle be constructed for any image type.

A. Computing Subspaces

Images in the set tend to be correlated to a large degree since visual displacements between consecutive images are small. The obvious step is to take advantage of this redundancy 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 [25], [6] where the eigenvectors of the image set are computed and used as orthogonal bases for representing individual images. Principal component analysis has been previously used in computer vision for deriving basis functions for feature detection [9], [12] representing human face images [30] and recognizing face images [31], [27]. Though, in general, all the eigenvectors of an image set are needed for perfect reconstruction of any particular image, only a few are sufficient for visual recognition. These eigenvectors constitute the dimensions of an image subspace, called the *eigenspace*, 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 column-wise

$$\mathbf{P} \triangleq \{\hat{\mathbf{i}}_1 - \mathbf{c}, \hat{\mathbf{i}}_2 - \mathbf{c}, \dots, \hat{\mathbf{i}}_M - \mathbf{c}\}. \tag{2}$$

$\mathbf{P}$ 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*

$$\mathbf{Q} \triangleq \mathbf{P}\mathbf{P}^T. \tag{3}$$

$\mathbf{Q}$ 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 determined by solving the well-known

eigenstructure decomposition problem

$$\lambda_k e_k = Q e_k. \tag{4}$$

Calculation of the eigenvectors of a matrix as large as 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. We have summarized a few of the representative algorithms in [20]. In our experiments, we have used a fast implementation [23] of the algorithm proposed by Murakami and Kumar [18]. On a Sun IPX workstation, for instance, 20 eigenvectors of a set of 100 images (each 128×128 in size) can be computed in about 3 minutes, and 20 eigenvectors of a 1000 image set in less than 4 hours. Workstations are fast gaining in performance and these numbers are expected to diminish with time.

The result of eigenstructure decomposition is a set of eigenvalues $\{\lambda_k \mid k = 1, 2, \ldots, K\}$ where $\{\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_K\}$, and a corresponding set of orthonormal eigenvectors $\{e_k \mid k = 1, 2, \ldots, K\}$. 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 a complete Hilbert space with N dimensions. Pattern recognition theory suggests several criteria for selecting K given the covariance matrix Q [25]. In all of our applications, we have found eigenspaces of 20 or less dimensions to be more than adequate.

B. Parametric Eigenspace Representation

Each workspace sample $\hat{i}_j$ in the image set is projected to eigenspace by first subtracting the average image c from it and finding the inner product of the result with each of the K eigenvectors. The result is a point f_j in eigenspace, where $f_j = [e_1, e_2, \ldots, e_K]^T (\hat{i}_j - c)$. By projecting all images in this manner, a set of discrete points is obtained. Since consecutive images are strongly correlated, their projections are close to one another. Hence, the discrete points obtained by projecting all the samples of the workspace can be assumed to lie on a manifold that represents a *continuous* appearance function. The discrete points are interpolated to obtain this manifold. In our implementation, we have used a standard quadratic B-spline interpolation algorithm [29]. The resulting manifold can be expressed as $f(q) = f(q_1, q_2, \ldots, q_m)$. It resides in a low-dimensional space and therefore is a compact representation of the visual workspace, i.e. appearance as a function of the task DOF q. The exact number of task DOF is of course application dependent. The above representation is called the *parametric eigenspace*. It was initially introduced in [19], [20] for real-time recognition and pose estimation of 3D objects.

It is worth pointing out that multiple visual workspaces can be represented in the same eigenspace as set of manifolds $F = \{f^1, f^2, \ldots, f^P\}$. In this case, the eigenspace is computed using image sets of all the visual workspaces. Multiple workspaces are used for example when a tracking application involves more than one object and requires the robot to switch and track an object whose appearance most closely matches the visual input.

It is well-known in pattern recognition theory [25], [20] that the distance between the two points in eigenspace is an approximation to the correlation between the two images. The closer the projections, the more similar are the images in l^2. The eigenspace is the optimal subspace for computing correlation between images. It is this property that motivates us to use principal component analysis to represent the visual workspace.

IV. IMAGE RECOGNITION

Our goal here is to develop an efficient method for recognizing an unknown input image $\hat{i}_c$ and find the corresponding task parameters

q_c. Since the eigenspace is the optimal subspace for computing the correlation between images, we can project the current image to eigenspace and simply look for the closest point on the workspace manifold. Image recognition proceeds as follows. We will assume that $\hat{i}_c$ has already been normalized in scale and brightness to suit the invariance requirements of the application. The average c of the visual workspace is subtracted from $\hat{i}_c$ and the resulting vector is projected to eigenspace to obtain the point, $f_c = [e_1, e_2, \ldots, e_K]^T (\hat{i}_c - c)$. The matching problem then is to find the minimum distance d between f_c and the manifold $f(q)$

$$d = \min_q \| f_c - f(q) \|. \tag{5}$$

If d is within some pre-determined threshold value (selected based on the noise characteristics of the image sensor), we conclude that $\hat{i}_c$ does belong to the manifold f, i.e., the input image is within the visual workspace. Then, parameter estimation is reduced to finding the coordinate 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 $f(q)$. Therefore, finding the closest point to f_c on $f(q)$ (or even a set of manifolds, F) is reduced to the classical nearest-neighbor problem.

V. FINDING THE CLOSEST MANIFOLD POINT

Mapping a novel input image to eigenspace is computationally simple. As mentioned earlier, the eigenspace is typically less than 20 in dimensions. The projection of an input image to a 20D space requires 20 dot products of the image with the orthogonal eigenvectors that constitute the space. This procedure can easily be done in real-time (frame rate of a typical image digitizer) on almost any general purpose workstation. What remains to be addressed is an efficient way of finding the closest manifold point.

We have implemented an efficient technique for binary search in multiple dimensions [24]. This algorithm uses a carefully designed data structure to facilitate quick search through the multi-dimensional eigenspace in $O(k \log_2 n)$, where n is the number of manifold points and K is the dimensionality of the eigenspace. In [24], code for the above algorithm is outlined and its performance is compared with those of other popular algorithms.

VI. VISUAL POSITIONING

Consider the problem of chip insertion on a circuit board (see Fig. 3). In most assembly lines, one can expect uncertainties in the positions of the manipulator, circuit board, and the chip holder. Visual positioning can play the critical role of forcing the manipulator to a desired position before the task of chip insertion is executed. One approach would be to compute geometric features such as corners and lines and match these features with a pre-stored geometric model of the circuit board. Then, the pose of the chip holder with respect to the manipulator needs to be computed before positioning can be executed. This is known to be difficult since precise feature detection and localization remain unresolved problem.

Our appearance based approach allows us to circumvent the need for precise feature detection and matching. In fact, it enables us to use substantially more information than just image discontinuities caused by features. It also captures pertinent visual cues such as shading. The visual workspace is defined as a range of possible appearances that result from deviations of the end-effector from the desired position with respect to the object. This workspace is learned off-line. During execution, a visual input corresponding to the present location of the end-effector is simply projected to eigenspace and its position on the workspace manifold reveals the displacement of the end-effector from its desired location. This information is fed to the robot controller to
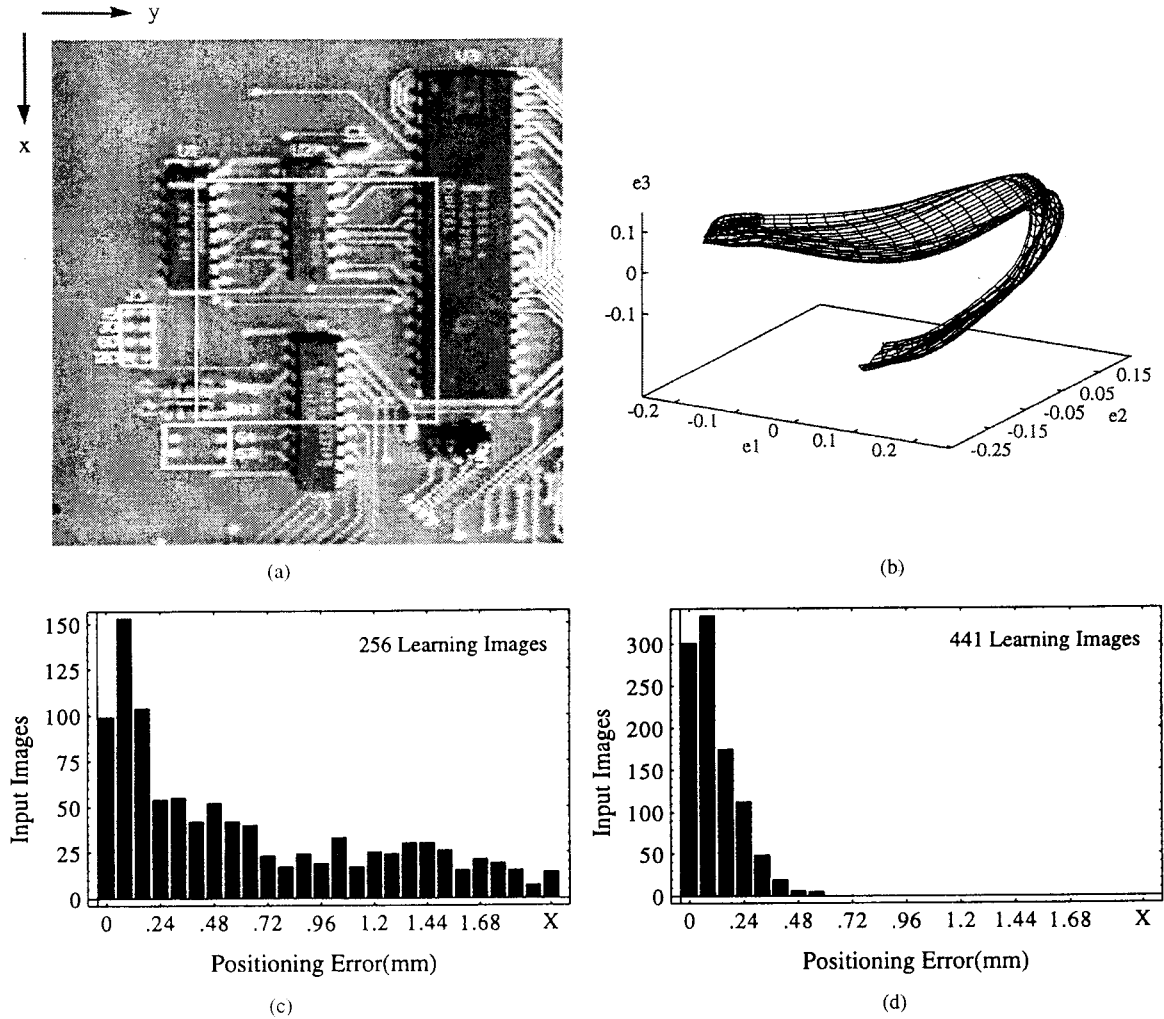
Fig. 3. Visual positioning experiment: printed circuit board. (a) Image window used for learning and positioning; (b) Parametric eigenspace representation of visual workspace displayed in 3D. Displacements are in two dimensions (x and y). Histograms of absolute positioning error (in mm) for (c) 256 learning images and (d) 441 learning images.

achieve accurate positioning. Then, the robot task is executed, in this case, insertion of the chip.

All our experiments were conducted using the Adept robot and hand-eye system shown in Fig. 1. The box shown in Fig. 3(a) is the image area (128×128 pixels) used for learning and positioning. Note that the image is rather complex and includes a variety of subtle features. In this experiment, robot displacements were restricted to two dimensions (x and y). A total of 256 images were taken by moving the robot to 16×16 equally spaced discrete points within a 2 cm $\times$ 2 cm region around the desired position. A 15D ($K = 15$) eigenspace was computed using the 256 images. Each workspace sample 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 15D eigenspace, we have shown it (see Fig. 3(b)) in a 3D 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 workspace manifold is stored in memory as a set of $251 \times 251 = 63\,001$ 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 2 cm. $\times$ 2 cm. region used for learning. Note that these positions are generally not the same as any of the workspace samples used for 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 Fig. 3(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. Fig. 3(d) shows the error histogram for $21 \times 21 = 441$ learning images obtained within the same 2 cm $\times$ 2 cm displacement region. In this case, the learning process was completed in approximately 30 min. 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 Fig. 4(a). In this case, 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 ± 1 cm range while θ within a $\pm 10°$ range for each (x, y) displacement. A total
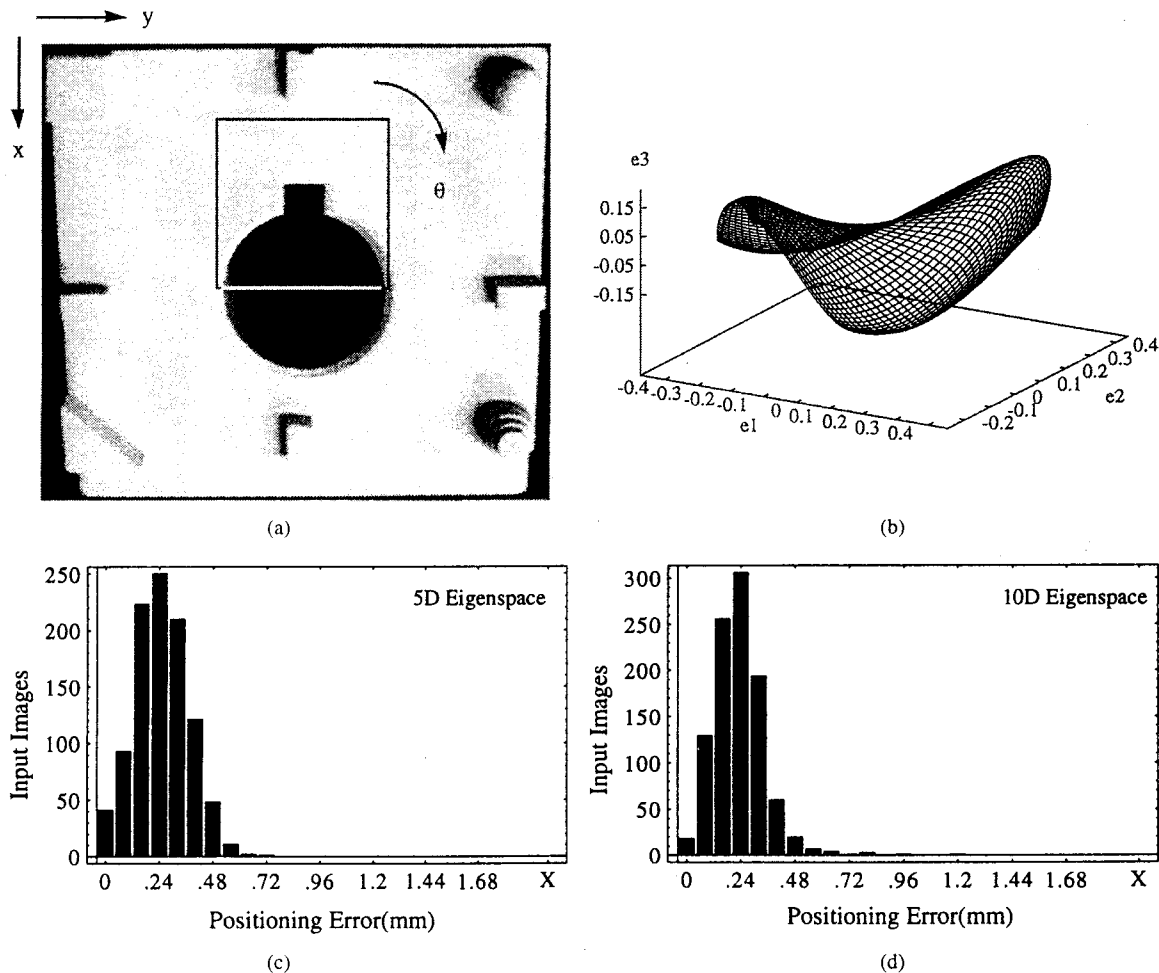
(a)

(b)

(c)

(d)

Fig. 4.   Visual positioning experiment: object with hole and slot. (a) Image window. (b) The workspace manifold in eigenspace displayed in 3D. Displacements are in three dimensions (x, y, θ). Histograms of absolute positioning error (in mm) for (c) 5D eigenspace and (d) 10D eigenspace.

of $11 \times 11 \times 11 = 1331$ learning images were obtained and a 5D eigenspace computed. The visual workspace representation in this case is a three-parameter manifold in 5D space. In Fig. 4(b) a projection of this manifold is shown as a surface (x and y are the parameters, while $\theta = 0$) in 3D. The actual manifold is stored in memory as a set of $65 \times 65 \times 65 = 274\,625$ points. In this case, the ire learning process took approximately 5 hours. Once again, 1000 ...dom displacements were used in the positioning experiments. The absolute Euclidean positioning errors in x-y space are illustrated by the histogram in Fig. 4(c). 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 $0.56°$ and deviation of $0.45°$. These results again indicate high positioning accuracy. Fig. 4(d) shows that positioning accuracy is only marginally improved for this particular object by doubling the eigenspace dimensions. 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 $0.44°$ and deviation of $0.33°$. This accuracy was verified by successful insertions of a peg in the hole of the object.

VII. REAL-TIME TRACKING

The visual processing aspects of tracking are identical to that of positioning. In tracking applications successive images may be assumed to be close to one another as 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 q_c in robot coordinates. This error may be used as input to a position control system. 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 control law we have used is based on a simple interpolation/prediction scheme to facilitate smooth manipulator motion. The controller generates a reference point q_r for the low-level robot actuators.

Fig. 5(a) shows an object we have used to test the tracking algorithm. The box illustrates the 96×96 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 $13 \times 13 \times 13 = 2197$ images were acquired during the learning stage by using robot displacements within $x = \pm 1$ cm, $y = \pm 1$ cm, $\theta = \pm 10°$. A 10D eigenspace was used to represent the three-parameter manifold. A projection of the manifold (using $\theta = 0$) is shown in 3D in Fig. 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 to 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.

The tracking accuracy was determined by moving the object at known velocity along a circle using a motorized turntable (Fig. 5(a)). The turntable was rotated through $90°$, moving the object through
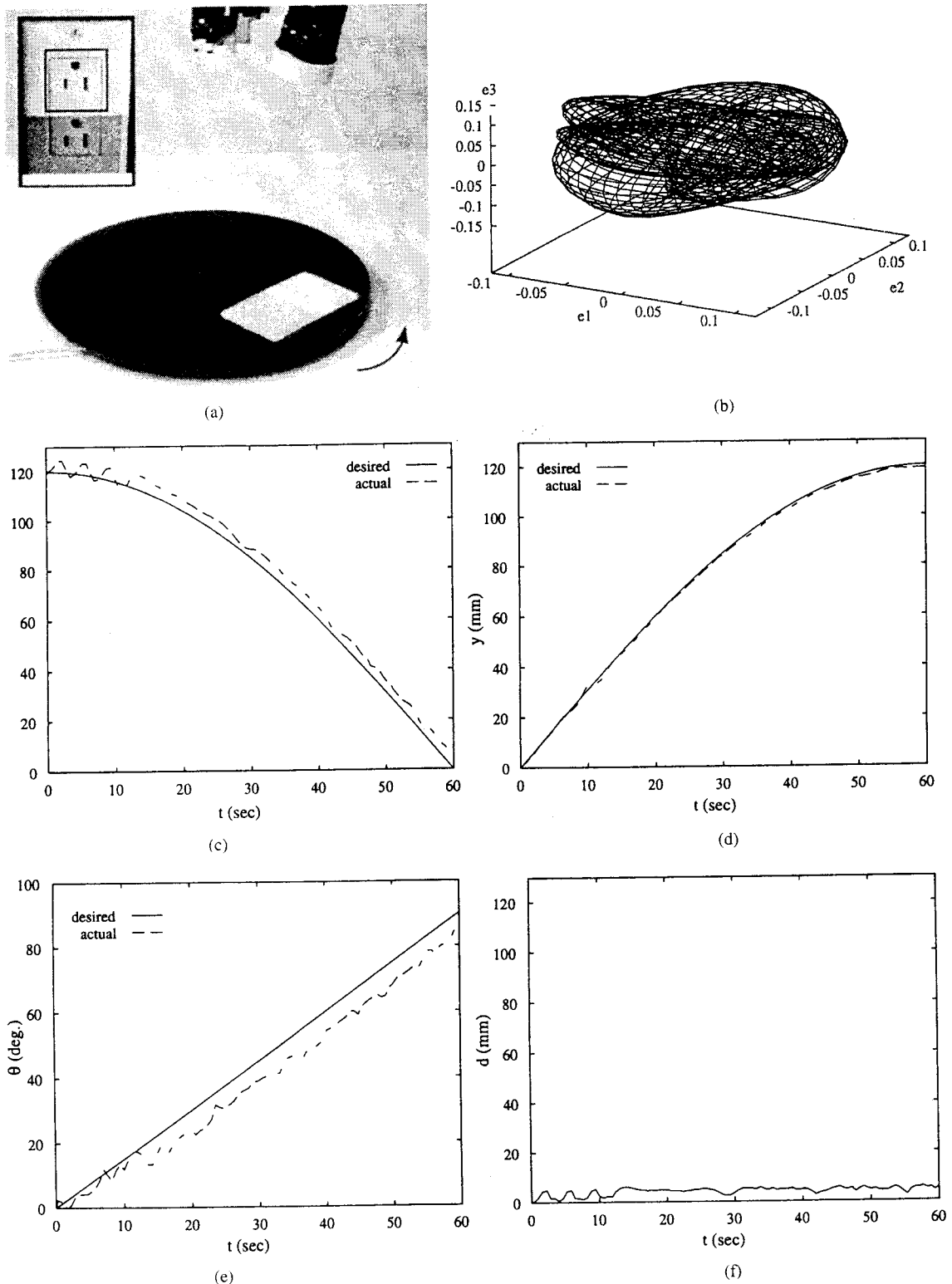
Fig. 5. Visual tracking experiment: moving electric socket. (a) The black box in the inset is the image window used. (b) Workspace manifold in eigenspace. Desired and actual coordinates: (c) $x(t)$, (d) $y(t)$, and (e) $\theta(t)$. (f) Tracking distance error $d(t)$.

a total distance of 19 cm. In Figs. 5(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.

In this experiment, all computations were done on a Sun IPX workstation without the use of any customized image processing hardware. The total cycle time was approximately 250 msec yielding a control rate of 4 Hz. This restricted us to objects moving at relatively slow speeds (approximately 0.5 cm/sec). It may be noted that this is merely a limitation of the implementation. All computations involved in the visual processing are simple and can be easily done at frame-rate (30 Hz) with a single frame-time delay using inexpensive image processing hardware (such as a standard i860 board). In other
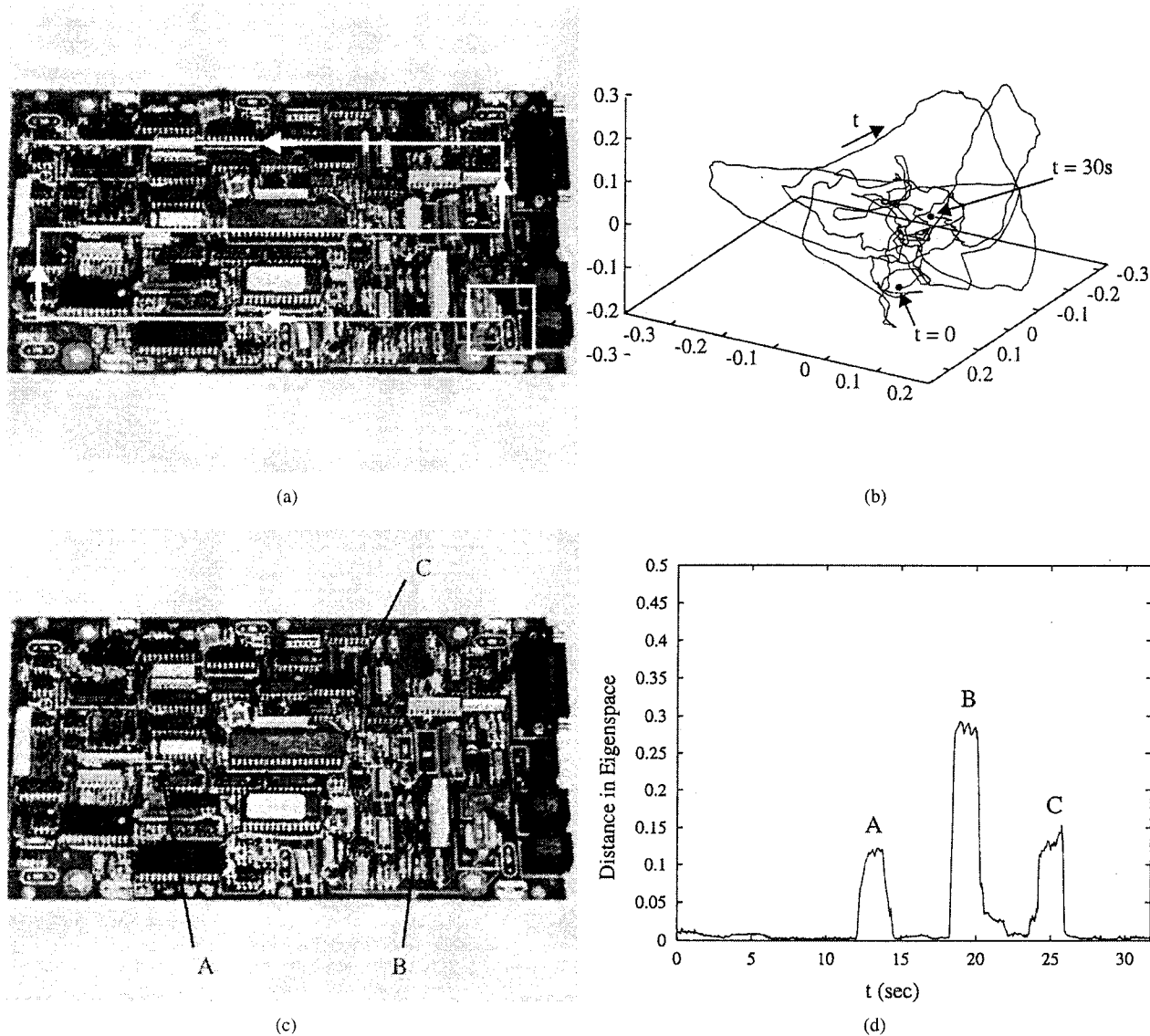
(a)

(b)

(c)

(d)

Fig. 6.   Temporal inspection experiment: modem circuit board. (a) The white box indicates the image window used and the white curve represents the visual trajectory traversed by the hand-eye system. (b) The workspace manifold in this case is the temporal appearance of the model circuit board parametrized by time t. (c) A modem circuit board with three missing components (visual defects indicated as A, B, and C). (d) Distance between eigenspace projections of the model and defective boards plotted as a function of time.

experiments reported in [22], a significant improvement in tracking speed was achieved using a DEC Alpha workstation.

VIII. TEMPORAL INSPECTION

We conclude with one last application of the appearance based approach, namely, inspection of manufactured parts. Consider the case of a fairly complex product such as the chassis of an automobile engine. Our objective is to visually scan the product (object) with the hand-eye system and in real-time determine whether all its components are in place. It is assumed that the reference coordinates of the object are either known a-priori or determined using the positioning algorithm described in Section VI. The parameters of the imaging optics (magnification, aperture, etc.) of the hand-eye system are selected by the user so as to ensure that pertinent visual characteristics, such as defects, of the object are clearly imaged. The hand-eye system is then swept along a fixed trajectory that encertains that the visual input window passes over all the relevant parts of the object. This trajectory again is chosen by the user and can involve the variation of any number of end-effector coordinates in any sequence.

Using a motorized imaging lens, the optical parameters can also be varied over the trajectory enabling the sensor to capture visual features at varying resolutions.

The end result of this scanning process is a sequence of images that are taken sequentially with *time*. The parametric eigenspace representation in this case is a single continuous *curve* that is parametrized by time rather than end-effector coordinates. This curve is a temporal appearance representation that is the model against which all manufactured products are to be compared. Given a new instance of the product, the hand-eye system is run through the same trajectory and the visual inputs are in real-time projected to eigenspace and compared to their corresponding points on the model curve. Note that the hand-eye travel time t is always known and it determines the corresponding point on the model curve. Therefore, in this particular application, the problem of finding the closest manifold point is obviated. The distance d between a projected point and its temporally corresponding model point determines whether the projected image has a defect. The time a defect is detected reveals its location on the fixed trajectory. Note that this approach detects only

defects in the correlation sense. However, once the defect images have been identified and separated from the entire sequence, these images can be further analyzed, perhaps using traditional feature based approaches, to determined the exact nature of the defect.

We have tested this idea of temporal inspection on a variety of complex parts. Fig. 6 demonstrates the reliability and real-time performance of our temporal inspection algorithm when applied to the circuit board of a modem. As is evident from Fig. 6(a), the board is very complex in appearance, including a variety of electronic devices. The image window and trajectory used are also shown. The entire trajectory, which includes a few hand-eye rotations, is traversed with high repeatability in a total of 30 s. The appearance representation of the model circuit board, parametrized by travel time t, is shown in Fig. 6(b). In Fig. 6(c), a modem with three missing components (marked A, B, and C) is shown. The inspection algorithm was executed for this defective board and the results are shown in Fig. 6(d), which shows the distance in eigenspace between the novel and model boards plotted as a function of travel time. As seen from the figure, the distance is close to zero in all parts of the trajectory except where the defects pass through the visual input window. From this distance function, the three defects are easily identified by using an error threshold.

REFERENCES

[1] J. S. Albus, "A new approach to manipulator control: The cerebellar model," *Trans. ASME, J. Dyn. Syst. Meas. Contr.*, vol. 97, pp. 220–227, Sept. 1975.

[2] P. K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman, "Trajectory filtering and prediction for automated tracking and grasping of a moving object," in *Proc. IEEE Int. Conf. on Robotics and Automation*, Nice, 1992, pp. 1850–1856.

[3] A. Castano and S. Hutchinson, "Hybrid vision/position servo control of a robotic manipulator," in *Proc. IEEE Int. Conf. on Robotics and Automation*, Nice, May 1992, pp. 1264–1269.

[4] R. T. Chin and C. R. Dyer, "Model-based recognition in robot vision," *ACM Comput. Surv.*, vol. 18, no. 1, pp. 67–108, 1986.

[5] J. Feddema, C. S. G. Lee, and O. Mitchell, "Weighted selection of image features for resolved rate visual feedback control," *IEEE Trans. Robot. Automat.*, vol. 7, no. 1, pp. 31–47, Feb. 1991.

[6] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. London: Academic, 1990.

[7] K. Hashimoto, T. Kimoto, and H. Kimura, "Manipulator control with image-based visual servo," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1991, pp. 2267–2271.

[8] A. S. Householder, *The Theory of Matrices in Numerical Analysis*. New York: Dover, 1964.

[9] R. A. Hummel, "Feature detection using basis functions," *Comput. Graph. Image Proc.*, vol. 9, pp. 40–55, 1979.

[10] A. Koivo and N. Houshangi, "Real-time vision feedback for servoing robotics manipulator with self-tuning controller," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 1, pp. 134–142, Feb. 1991.

[11] M. Kuperstien, "Adaptive visual-motor coordination in multijoint robots using parallel architecture," in *Proc. IEEE Int. Conf on Robotics and Automation*, Raleigh, NC, 1987, pp. 1595–1602.

[12] R. Lenz, "Optimal filters for the detection of linear patterns in 2D and higher dimensional images," *Pattern Recognition*, vol. 20, no. 2, pp. 163–172, 1987.

[13] R. Luo, R. Mullen, and D. Wessel, "An adaptive robotic tracking system using optical flow," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1988, pp. 568–573.

[14] B. W. Mel, "MURPHY: A robot that learns by doing," in *AIP Proc. Neural Information Processing System Conf.*, Denver, CO, 1987.

[15] S. Mersch, "Polarized lighting for machine vision applications," in *Proc. of RI/SME Third Annu. Applied Machine Vision Conf.*, Schaumburg, IL, Feb. 1984, pp. 40–54.

[16] W. T. Miller, "Sensor-based control of robotic manipulators using a general learning algorithm," *IEEE J. Robot. Automat.*, vol. RA-3, no. 2, pp. 157–165, Apr. 1987.

[17] ——, "Real-time application of neural networks for sensor-based control of robots with vision," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 4, pp. 825–831, July/Aug. 1989.

[18] H. Murakami and V. Kumar, "Efficient calculation of primary images from a set of images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 4, no. 5, pp. 511–515, Sept. 1982.

[19] H. Murase and S. K. Nayar, "Learning and recognition of 3D objects from appearance," in *Proc. IEEE Workshop on Qualitative Vision*, June 1993, pp. 39–50.

[20] ——, "Visual learning and recognition of 3D objects from appearance," *Int. J. Comp. Vision*, vol. 14, no. 1, pp. 5–24, 1995.

[21] S. K. Nayar, H. Murase, and S. A. Nene, "Learning, positioning, and tracking visual appearance," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, San Diego, May 1994.

[22] S. K. Nayar, S. A. Nene, and H. Murase, "Subspace methods for robot vision," Tech. Rep. CUCS-06-95, Dept. Computer Science, Columbia University, New York, Feb. 1995.

[23] S. A. Nene, S. K. Nayar, and H. Murase, "SLAM: A software library for appearance matching," in *Proc. ARPA Image Understanding Workshop*, Monterey, CA, Nov. 1994. Also Tech. Rep. CUCS-019-94.

[24] S. A. Nene and S. K. Nayar, "Closest point search in high dimensions," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, San Fransisco, June 1996.

[25] E. Oja, *Subspace Methods of Pattern Recognition*. Hertfordshire, UK: Research Studies Press, 1983.

[26] N. Papanikolopoulos, P. Khosla, and T. Kanade, "Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision," *IEEE Trans. Robot. Automat.*, vol. 9, no. 1, pp. 14–35, Feb. 1993.

[27] A. Pentland, B. Moghaddam, and T. Starner, "View-based and modular eigenspaces for face recognition," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Seattle, June 1994.

[28] W. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C*. Cambridge, UK: Cambridge Univ. Press, 1988.

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

[30] L. Sirovich and M. Kirby, "Low dimensional procedure for the characterization of human faces," *J. Opt. Soc. Amer.*, vol. 4, no. 3, pp. 519–524, 1987.

[31] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, June 1991, pp. 586–591.

[32] J. Walter, T. Martinez, and K. Schulten, "Industrial robot learns visuo-motor coordination by means of neural-gas network," in *Proc. Int. Joint Conf. of Neural Networks*, June 1990.

[33] L. Weiss, A. Sanderson, and C. Neuman, "Dynamic sensor-based control of robots with visual feedback," *IEEE J. Robot. Automat.*, vol. RA-3, no. 5, pp. 404–417, Oct. 1987.