Proceedings of the 2001 IEEE/RSJ
International Conference on Intelligent Robots and Systems
Maui, Hawaii, USA, Oct. 29 – Nov. 03, 2001

Eigendecomposition-Based Pose Detection in the Presence of Occlusion

C-Y. Chang, A. A. Maciejewski, V. Balakrishnan, and R. G. Roberts
Purdue University
1285 Electrical Engineering Building
West Lafayette, Indiana 47907-1285
Tel (765) 494-9855
Fax (765) 494-6951
maciejew@ecn.purdue.edu

Abstract— **Eigendecomposition-based techniques are popular for a number of computer vision problems, e.g., object and pose detection, because they are purely appearance-based and they require few on-line computations. Unfortunately, they also typically require an unobstructed view of the object whose pose is being detected. The presence of occlusion precludes the use of the normalizations that are typically applied and significantly alters the appearance of the object under detection. This work presents an algorithm that is based on applying eigendecomposition to a quadtree representation of the image dataset used to describe the appearance of an object. This allows decisions concerning the pose of an object to be based on only those portions of the image in which the algorithm has determined that the object is not occluded. The accuracy and computational efficiency of the proposed approach is evaluated on sixteen different objects with up to 50% of the object being occluded.**

I. Introduction

One of the fundamental problems in computer vision is the recognition and localization of three-dimensional objects. Subspace methods represent one computationally efficient approach for dealing with this class of problems. Variously referred to as eigenspace methods, principal component analysis methods, and Karhunen-Loeve transformation methods [1], these have been used extensively in a variety of applications such as face characterization [2] and recognition [3], lip-reading [4, 5], object recognition, pose detection, visual tracking, and inspection [6, 7, 8, 9]. All of these applications are based on taking advantage of the fact that a set of highly correlated images can be approximately represented by a small set of eigenimages [10]. Once the set of principal eigenimages is determined, online computation using these eigenimages can be performed very efficiently.

Unfortunately, one of the drawbacks associated with using eigendecomposition-based approaches is that they are very sensitive to occlusion. The purpose of this work is to explore the feasibility of applying eigendecomposition to a quadtree representation of correlated images in order to efficiently accommodate the presence of occlusion. The pose detection problem is used here as a representative application. In the next subsection, the fundamentals of applying eigendecomposition to related images are reviewed. This is followed by an overview of the standard approach to solving the pose detection problem using eigendecomposition and a discussion of why occlusion presents such difficulty.

This work was supported by the Sze Tsao Chang Memorial Engineering Fund, the Office of Naval Research under contract no. N00014-97-1-0640, and the National Imagery and Mapping Agency under contract no. NMA201-00-1-1003.

A. Eigendecomposition of Related Images

An image is an $h \times v$ array of square pixels with intensity values normalized between 0 and 1. Thus, an image will be represented by a matrix $\mathcal{X} \in [0,1]^{h \times v}$. Since we will be considering sets of related images, it will be convenient to represent an image equivalently as a vector, obtained simply by "row-scanning", i.e., concatenating the rows to obtain the *image vector* $\mathbf{x}$ of length $m = hv$:

$$\mathbf{x} = \mathrm{vec}(\mathcal{X}^T).$$

The *image data matrix* of a set of images $\mathcal{X}_1, \ldots, \mathcal{X}_n$ is an $m \times n$ matrix, denoted X, and defined as

$$X = [\mathbf{x}_1 \ \cdots \ \mathbf{x}_n],$$

with typically $m \gg n$. We consider only the case where n is fixed, as opposed to cases where X is constantly updated with new images.

The *average image vector* is denoted $\bar{\mathbf{x}}$ and defined as

$$\bar{\mathbf{x}} = (\mathbf{x}_1 + \cdots + \mathbf{x}_n)/n.$$

The corresponding *average image data matrix*, denoted $\bar{X}$, is

$$\bar{X} = [\bar{\mathbf{x}} \ \cdots \ \bar{\mathbf{x}}].$$

The matrix $X - \bar{X}$, which we denote $\hat{X}$, has the interpretation of an "unbiased" image data matrix.

The singular value decomposition (SVD) of $\hat{X}$ is given by

$$\hat{X} = \hat{U}\hat{\Sigma}\hat{V}^T,$$

where $\hat{U} \in \mathbb{R}^{m \times m}$ and $\hat{V} \in \mathbb{R}^{n \times n}$ are orthogonal, and $\hat{\Sigma} \in \mathbb{R}^{m \times n}$, with $\hat{\Sigma} = [\hat{\Sigma}_d \ \mathbf{0}]^T$, where $\hat{\Sigma}_d = \mathrm{diag}(\hat{\sigma}_1, \ldots, \hat{\sigma}_n)$, with $\hat{\sigma}_1 \geq \hat{\sigma}_2 \geq \cdots \geq \hat{\sigma}_n \geq 0$, and $\mathbf{0}$ is an n by $m-n$ zero matrix. The SVD of $\hat{X}$ plays a central role in several important imaging applications such as image compression, pattern recognition and pose detection. The columns of $\hat{U}$, denoted $\hat{\mathbf{u}}_i$, $i = 1, \ldots, m$, are referred to as the eigenimages of $\hat{X}$; these can be interpreted as estimates of the eigenvectors of the covariance matrix of the image vector. The eigenimages provide an orthonormal basis for the columns of $\hat{X}$, ordered in terms of importance; the corresponding singular values measure how "aligned" the columns of $\hat{X}$ are, with the associated eigenimage. The components of the ith column of $\hat{V}$ measure how much each individual image contributes to the ith eigenimage.

B. Eigendecomposition Applied to Pose Detection

The standard application of eigendecomposition to solve the pose detection problem requires the computation of a reduced-order representation of the set of all possible orientations for the object being considered. Fig. 1 is used to illustrate this off-line process with a simple example. In part (a) of the figure, several training images of an object are shown at different orientations (for the purpose of illustration, only one degree of freedom is used in this example). Because the eigenspace representation of an image is very sensitive to changes in size and intensity, each training image is then normalized to account for differences in scale (part (b))and brightness (part (c)). Note that this requires that the image be segmented to determine the boundary of the object. The average normalized training image is then subtracted from each of the normalized training images (parts (d) and (e)) and the eigendecomposition is computed from the resulting images (part (f)). A reduced-order representation of the object's orientation change is then obtained by projecting the normalized training images into the space spanned by the dominant eigenimages, and interpolating to obtain a manifold (part (g)).

To determine the pose of the object in a given test image, that image must undergo the same transformations as a training image, i.e., it must be normalized in both scale and intensity and have the average training image subtracted from it. It can then be projected onto the reduced-order eigenspace and the object's orientation obtained by computing the closest point on the manifold created using the training images. This process is very computationally efficient and reasonably accurate *if* the boundary of the object in the test image can be calculated. Unfortunately, the presence of occlusion complicates this procedure in several ways:

1. The location of the object in the test image cannot be easily determined.
2. Scale normalization cannot be performed on the test image.
3. Brightness normalization is not effective.
4. The occluded region will alter the projection into the eigenspace.

Some of these problems can be addressed by using a hierarchical eigenspace approach [8], for example, one can include object size variation in the set of training images. Alternatively, one can apply an "eigen windows"[11] approach in which small windows around "feature points" are used for both training and detection. Unfortunately, this approach relies on appropriate feature selection as well as detection, and thus loses the advantages associated with purely appearance-based techniques.

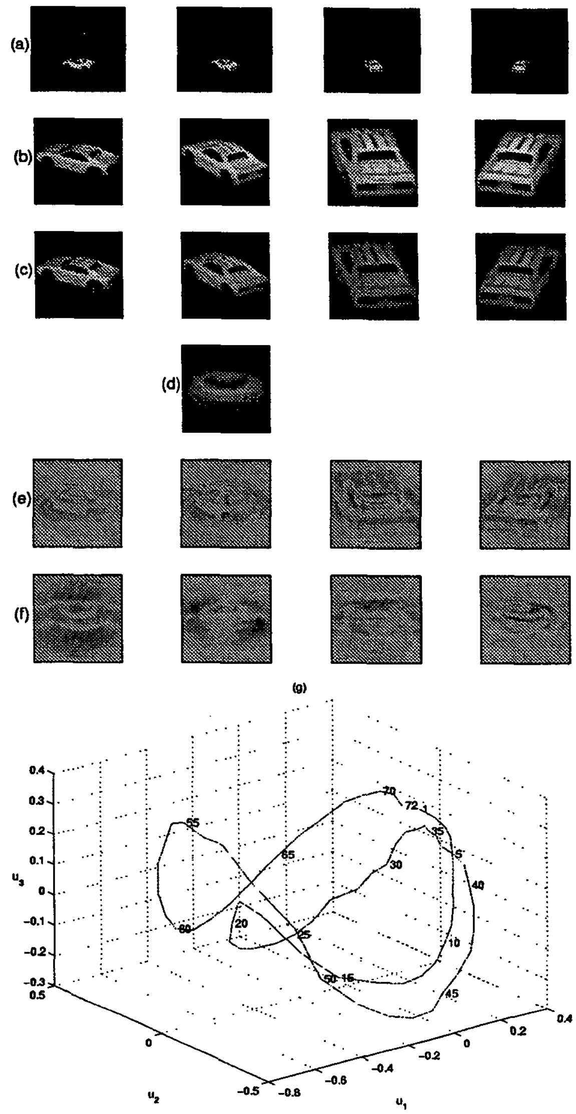

Fig. 1. An example of the off-line training process for a simple one-dimensional pose detection scenario: (a) images of the object at different orientations (b) size normalized images (c) intensity normalized images (To improve the contrast in this subfigure, each image was separately normalized.) (d) average of normalized image (e) normalized images with average image subtracted (f) eigenimages (g) manifold consisting of the normalized training images projected onto the subspace spanned by the first three eigenimages. In cases where both positive and negative values are possible, i.e., (e) and (f), pure red is used to represent positive one, pure blue to represent negative one, and pure green to represent zero.

The goal of the work presented here is to solve the pose detection problem in the presence of occlusion, while retaining the framework of an eigendecomposition approach and all its attendant advantages. The next section presents an outline of our approach, which first identifies candidate locations for the object in a test image, and then performs pose detection using eigendecomposition on a quadtree representation of the training images. The efficacy of our approach, both in terms of accuracy and computational efficiency, as a function of the degree of occlusion, is then explored through a number of experiments.

II. Algorithm Description

We consider the problem of object localization and pose detection under the assumption that the target object is partially occluded but is in an environment where the background can be controlled. A two-step approach is proposed to solve this problem. The first step is to determine the likely candidate locations of the object in the test image. The second step is to evaluate the candidate locations by using eigendecomposition on a quadtree structure of the training images to simultaneously determine if the object is present at a given candidate location, and if so, its pose.

A. Localization

Let $\mathbf{y}$ be an image vector of the same size as the training images that represents a window within the test image offset by (v, h) pixels in the vertical and horizontal directions, respectively. Then, if there is no occlusion, one can identify the location of the desired object within the test image, given by (v, h), by comparing $\mathbf{y}$ to the training images for every possible value of (v, h). However, because an eigendecomposition of the training images exists, it is much more computationally efficient to simply compute the amount of $\mathbf{y}$ that can be represented in a smaller eigenspace, i.e.,

$$y^{(k)} = \sqrt{\sum_{i=1}^{k} (\mathbf{u}_i^T \mathbf{y})^2}.$$

where k represents the size of the reduced order representation and $\mathbf{u}_i$ represents the ith eigenimage. Because the brightness within the window $\mathbf{y}$ will vary for different values of (v, h), the normalized measure

$$m_1 = \frac{y^{(k)}}{\|\mathbf{y}\|} \qquad (1)$$

is more useful for comparing different locations within the test image. Because most of the energy of the training images is preserved by its eigenspace, this measure is likely to be maximized when the image represented by $\mathbf{y}$ is similar to one of the training images, thus identifying the location of the object with the test image.

The major computational expense in evaluating (1) consists of the dot products of the eigenimages $\mathbf{u}_i$ with the image vector $\mathbf{y}$ associated with the window at all possible locations, i.e., all values of (v, h). It was shown in [12] that these projections can be efficiently computed by using a 2-D FFT. That is, if

$$P = \mathcal{F}^{-1}(\mathcal{F}(\mathcal{X})\mathcal{F}(U_i)^*)$$

then

$$\mathbf{u}_i^T \mathbf{y} = p_{(v,h)}$$

where $\mathcal{X}$ is the test image, U_i is obtained from the eigenimage matrix by padding it with zeros to the size of $\mathcal{X}$, the $*$ represents the conjugate and $p_{(v,h)}$ represents the (v, h) entry of the matrix P. The 2-D FFT of all the eigenimages can be pre-calculated and stored during the off-line process. The major on-line computation involved in evaluating m_1, for every possible location in the test image, requires one 2-D FFT of the test image and k 2D inverse FFTs where k is the eigenspace dimension. This is much more efficient than performing a brute force match of the test image with all the training images.

While this method works well for a controlled environment, it is not as effective when occlusion is present. This is illustrated in Fig. 2 where the above approach is applied to the same object, both with and without occlusion. Large values of m_1 do occur when the training images are correctly registed with the test image, however, they do not necessarily correspond to the largest values. In fact, in Fig. 2(e) there were 202 locations that had a higher or equal value of m_1 than that of the correct location because of the occluding object. It is still possible, however, to differentiate between large values of m_1 that are due to the desired object and those that are due to occlusion. This is due to the likelihood that the value of $y^{(k)}$ will be much more sensitive to small registration errors for the desired object than for the occluding object. This motivates the use of a measure based on the second derivative of $y^{(k)}$, namely,

$$m_2 = \sqrt{\left(\frac{\partial^2 y^{(k)}}{\partial v^2}\right)^2 + \left(\frac{\partial^2 y^{(k)}}{\partial h^2}\right)^2}. \qquad (2)$$

This measure is clearly effective in identifying the correct object location for the example shown in Fig. 2.

Because the measure m_2 can tend to be "noisy" due to the use of derivatives, it is combined with the value-based measure m_1 to form the measure:

$$M = \begin{cases} m_2 & \text{if } m_1 \geq \rho \\ 0 & \text{if } m_1 < \rho \end{cases}$$

where ρ is a preset threshold, which is used to identify candidate locations of the desired object even under the influence of occlusion.

An experiment was conducted to evaluate the accuracy of measure M for use in identifying the location of a desired object in a test image as a function of the percent of occlusion. (The percent of occlusion for a test image is defined as the area of the object that is occluded divided by the area of the entire object.) A total of 800 cases were examined, with the percent of occlusion evenly distributed between 0 to 80 percent. The target object used in this

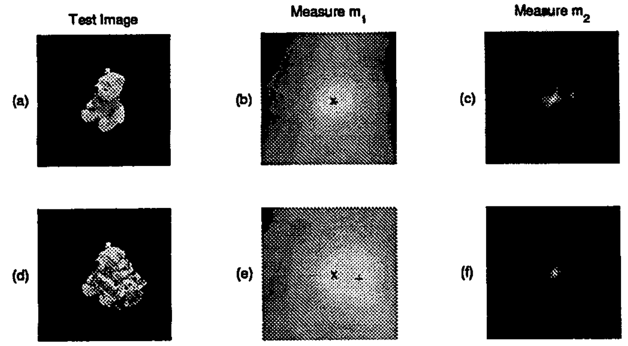

Fig. 2. Test images without and with occlusion are shown in (a) and (d) respectively. Their values of measure m_1 are shown in (b) and (e) with the peak location of the measure marked by a "+" and the correct location of the object marked by an "x". Note that the peak has shifted from the correct location due to the occlusion. However, the peak for measure m_2, shown in (c) and (f), correctly registered the location of the object even when occlusion is present.

experiment is shown in Fig. 2(a) with the image of an occluding object randomly selected from a pool of 15 other objects (see Fig. 5) to create the desired level of occlusion. The test images were of size 256 × 256 and the training images were of size 128 × 128. With a total of 90 training images used to create a 12-dimensional eigenspace.

The measure M was evaluated at a resolution of one pixel in both horizontal and vertical directions. The number of locations that have a measure higher than or equal to that of the correct location, will be referred to as the rank of the correct location. In 60% of all the cases, the rank of the correct location was one, i.e., it had the highest value of M. In addition, the rank of the correct location was less than fifty for over 90% of all cases. In all cases where the occlusion was less than 30%, the average rank was less than 5, and the average rank was still less than 50 even when up to 50% of the object was occluded. The average rank was never more than 100, even for the maximum occlusion of 80%. This suggests that an object registration and pose detection scheme based on candidate locations identified using the measure M may be very efficient. This is the topic of the next subsection.

B. Quadtree Based Detection

Once a number of candidate locations have been determined using the measure M, these locations are evaluated using eigendecomposition on a quadtree representation of the training images. Eigenimages are calculated for each level of a quadtree decomposition of the training images. At a level l, each training image is broken into $4^{(l-1)}$ sub-images (see Fig. 3). Let $\mathbf{x}_{i,l,j}$ denote the image vector associated with the jth sub-image in level l of the ith training image. Then the image data matrix associated with the jth sub-image in level l is formed as:

$$X_{l,j} = [\mathbf{x}_{1,l,j}\ \mathbf{x}_{2,l,j}\ \cdots\ \mathbf{x}_{n,l,j}].$$

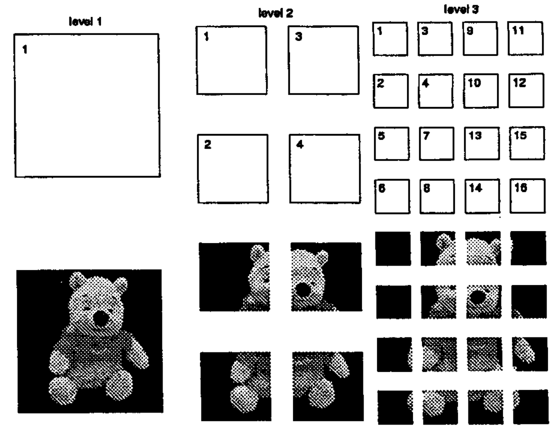

Fig. 3. This figure shows how an image is decomposed into a quadtree structure, and how the sub-images in each level are numbered.

Eigendecomposition is applied to $\hat{X}_{l,j} = X_{l,j} - \bar{X}_{l,j}$ for each sub-image in each level where $\bar{X}_{l,j}$ is the average image data matrix for the image vectors in $X_{l,j}$. If a k-dimensional eigenspace is used, then there will be $4^{(l-1)} \times k$ eigenimages for this level. The image data matrices $X_{l,j}$ that contain very little information about an object, i.e., contain mostly background, are discarded (for example, $X_{3,1}$ in Fig. 3). The projection of $\mathbf{x}_{i,l,j} - \bar{\mathbf{x}}_{l,j}$ onto the corresponding eigenspace is calculated to form the pose manifold that is used in the on-line process.

The on-line process consists of performing image comparisons in the eigenspace for each of the candidate locations until a "match" is found. For pose detection without occlusion, the orientation of the object is obtained from the point on the manifold that is closest to the projection of the test image because both the test image and training images can be normalized. When occlusion is present, the images are not automatically registered so that the distance to the manifold of training images is used to simultaneously determine if the object is present at this location in addition to determining its orientation. The following normalized distance is used:

$$d = ||\mathbf{t}_\theta - \mathbf{p}_q|| / ||\mathbf{t}_\theta|| \tag{3}$$

where $\mathbf{t}_\theta$ represents a point on the manifold of training images at orientation θ for a particular sub-image and $\mathbf{p}_q$ represents the eigenspace projection of the corresponding sub-image in the window associated with the qth candidate location. The projection $\mathbf{p}_q$ is considered to match $\mathbf{t}_i$, the ith training image whose orientation is closest to θ, when d is less than a preset threshold. When the training images cover all the possible variations in orientation, illumination, scaling and other factors, this threshold can be set very small; however this is usually not feasible and interpolation is used between samples. (A threshold of 0.1 was used for the examples presented in the subsequent sections, which worked well for a variety of objects.)

For each candidate location, the image comparison in the eigenspace is evaluated based on equation (3). The comparison starts at the first level and proceeds to smaller images at higher levels in the quadtree. At each level, all the sub-images that do not consist of only the background are examined. If d is smaller than a preset threshold for orientation θ, the orientation associated with training image i whose projection is closest to $\mathbf{t}_\theta$ will receive a vote. The value of this vote is equal to the percentage of the corresponding training sub-image that is occupied by the object. For example, the vote associated with sub-image 4 at level 3 in Fig. 3 is much smaller than that of sub-image 7 at level 3. This mechanism is used to de-emphasize the vote from a sub-image corresponding to a background area or containing very limited information about the object. If the normalized distances for all orientations of a sub-image are greater than a preset threshold (0.9 is used), all child notes in the following level will be skipped to save computation time. If the vote for a particular orientation exceeds a preset threshold, the process is terminated and the orientation of the object is determined by the orientation receiving the largest vote. The process moves to the next level if the maximum vote does not exceed the preset threshold. This threshold is set to $\frac{2}{4(l_{max}-l)}$, where l_{max} is the maximum level allowed. (In this work $l_{max} = 4$ is used with a training image of size 128×128.) If the level l_{max} is completed and no voting exceeds this threshold, the next candidate location is then assessed.

An example of this process is illustrated in Fig. 4. The algorithm starts from the first level, with the corresponding area of the test image displayed on the left and the normalized distance for different orientations on the right. The acceptance threshold is set at 0.1 and the rejection threshold is set at 0.9. In this example, neither the acceptance nor the rejection criterion is satisfied at the first level; therefore, the search proceeds to the second level. In the second level, the third sub-image exceeded the rejection threshold, and thus none of its child nodes are evaluated. In level 3, sub-images $6, 8, 14$, and 16 were not occluded and thus were successfully detected. All four of these sub-images voted for an orientation of 355 degrees, although the last sub-image also voted for other orientations. However, because the value of a vote is based on the percent of the matched training sub-image that is occupied by the object, the value of the vote for this sub-image is equal to zero. The process stops at this level with the conclusion that the object is at an orientation of 355 degrees.

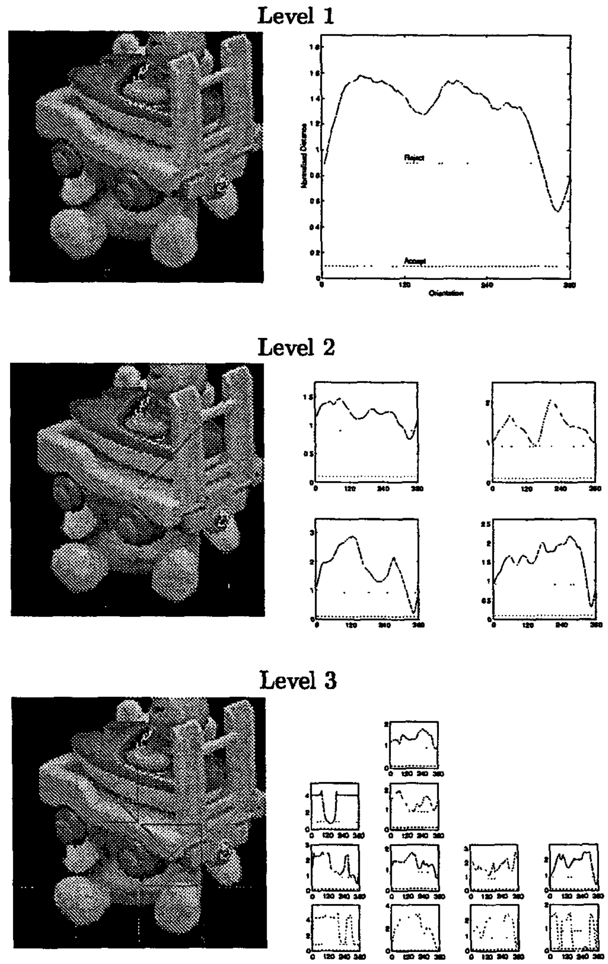

Fig. 4. This figure shows an example of the quadtree eigenspace object/pose detection process. The sub-figures on the left show the test image and the result of the process for all sub-images in each level. A green box indicates that a sub-image is accepted and a red box with a cross through it indicates the rejected sub-images. The sub-figures on the right show the normalized distance as a function of orientation, for each sub-image in each level. The acceptance and rejection thresholds are set at 0.1 and 0.9 respectively.

III. Experimental Evaluation

To evaluate the accuracy and computational efficiency of the proposed algorithm, a number of experiments were performed. For these experiments, it was assumed that the target object whose pose is desired is partially occluded but it is located in a controlled environment with no background clutter. A variety of objects (see Fig. 5) were used to test the robustness of the proposed algorithm. For each target object, 90 training images were obtained by rotating the object by 4 degrees between images. All training images are of size 128×128 pixels with 8 bits used to represent intensity. A 12-dimensional eigenspace was used for every sub-image at each level of the quadtree. All test images (of size 256×256) were generated by superimposing the image of a randomly selected occluding object on top of a randomly selected target object. The target objects were selected from 360 different possible images, i.e., a 1 degree rotation between successive images, in order to include poses that were not part of the training set. The percent of occlusion used in the test images was equally distributed, with 100 cases selected within each 10 percent range. The rejection threshold was set at 0.9 and the acceptance threshold was set at 0.1.

The first set of tests was designed to evaluate the performance of the quadtree approach, independent of the localization problem. For these tests, the size of the target object was the same as for the training images and the objects location in the test image was specified. Fig. 6 shows

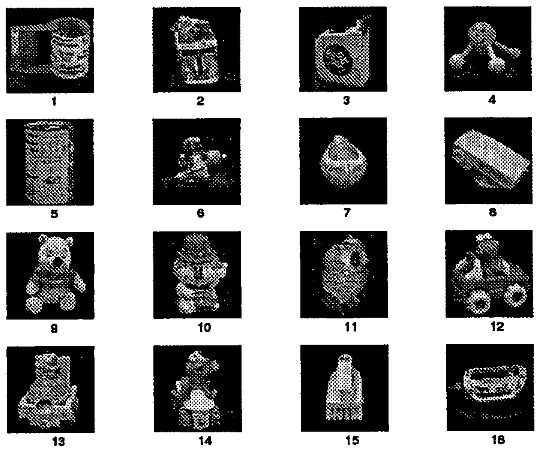

Fig. 5. The 16 objects used for the experiments in this section.

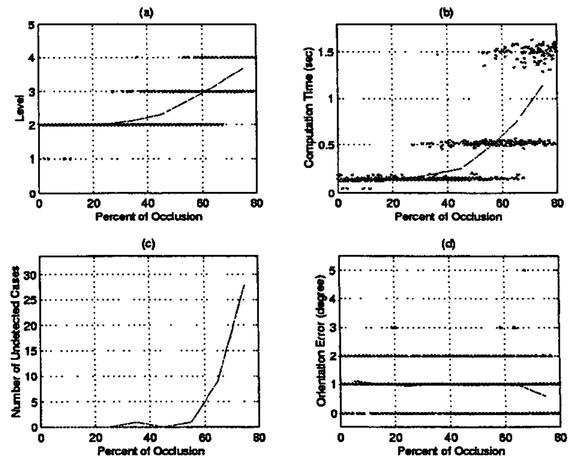

Fig. 6. This figure provides data on the accuracy and computational efficiency of the proposed algorithm as a function of the percent occlusion when evaluated on 800 random test cases of correctly registered images. (a) depth to which the quadtree is searched (average plotted with a solid line) (b) computation time required for each case (average plotted in red) (c) number of cases where the pose cannot be determined (d) orientation error when a pose is determined (average plotted with a solid line)

the results of these tests where the percent of occlusion was varied from 0 to 80 percent, i.e., 800 test cases. As would be expected, the amount of work that the algorithm must perform, i.e., the depth to which the quadtree must be evaluated, is monotonically related to the difficulty of the problem, i.e., the percent occlusion. Both the average depth and average computation time markedly increase for objects that are occluded by more than 50 percent. The difficulty of determining the pose of objects that are more than 50 percent occluded is even more strikingly evident in part (c) of the figure which plots the number of cases in which the algorithm cannot determine the objects orientation. However, it is important to note that even for the 100 cases with occlusions between 70-80 percent, in 72 of them the pose of the object was able to be determined. Even more importantly, the average accuracy to which the algorithm determines an object's orientation is essentially independent of the amount of occlusion. In other words, if the algorithm can make a decision regarding an objects pose, it is usually quite accurate.

The next set of experiments was designed to evaluate the performance of the quadtree decomposition approach when applied to candidate object locations identified by using measure M on the test images. To account for the fact that size normalization cannot be performed, an additional 180 training images were used for each object, where the size of the object was enlarged and reduced by 5 percent, resulting in an image data matrix of 270 images. Two sets of test images were generated in the manner described above. In one set, referred to as the perturbed set, the object size, the brightness of the background and the brightness of the object itself were all randomly perturbed by a value between 0 and 5 percent. In the other set all of these factors were held constant (referred to as the unperturbed set). Because the quadtree-based pose detection approach started to degrade when the occlusion was greater than 50 percent, 500 test cases were used for both the perturbed and unperturbed sets with the percent occlusion equally distributed between 0 and 50 percent. The measure M was used to select the top 100 candidate locations with the constraint that two candidate locations cannot be adjacent.

The performance of the measure M for localization is shown in Fig. 7, where the percent of cases that have the rank of the correct location smaller than the value of the x-axis is displayed. The rank displayed here is before the adjacent pixels were removed, so that the locations with the top 200 values of M are candidate locations that will be evaluated by the quadtree approach. Thus for all objects in the unperturbed set, 90 percent of the cases will have the correct location of the object evaluated. This percentage is reduced significantly if perturbation in the test images is allowed. In particular, in the worst case, i.e., object five, the correct location was a candidate in only 70 percent of the perturbed cases. This is due to object five being rotationally symmetric and therefore having its eigenimages contain sharp edges that are very sensitive to size variations. This situation can be addressed by including more size variation into the training set. (Note that the correct location for object five had a rank of less than 10 for all cases in the unperturbed set.)

The performance of the complete algorithm that includes the quadtree detection applied to the candidate locations identified using the measure M is summarized in Table I and Table II for unperturbed and perturbed test sets, respectively. Unregistered cases refer to those test cases where the pose detection procedure did not identify the object at any of the candidate locations. This is either due to the correct location not being one of the candidate locations (which was typically the case for objects 5, 7, and 11) or due to a significant difference in the appearance of the test image (either due to a pose that is not represented in the training images, occlusion, or perturbation) that prevents detection even at the correct location (which was typically the case for objects 6 and 16). Misregistered cases refer to those cases where the algorithm identified the presence of the object but in an incorrect location. This typically occurred for objects that either contain large areas of uniform intensity (objects 3, 7, 8 and 15) or have different portions of the object that appear similar (objects 1 and 4). For those cases where the object was identified in its correct location, the error in computing the objects pose was calculated. Note that because the training images are taken every 4 degrees, an error of up to 4 degrees still implies that the algorithm identified the correct interval in the manifold of object poses. (Thus a more accurate orientation could be determined by doing a local optimization.) Therefore, only orientation errors of greater than 8 degrees are considered pose detection errors. The total number of errors is the sum of the cases where either the location or the pose was incorrectly determined.

In general, the percentage of cases where the algorithm correctly identified both the location and the pose of the target object was quite high. In particular, most objects were correctly identified (95 percent in the unperturbed case and 90 percent in the perturbed case), or if they were not, then the algorithm effectively declared that the problem was too difficult, i.e. it could not register the test image (e.g., for objects 5, 6, 7, 11, and 16). The only objects that created a difficulty for the algorithm in terms of true errors were object 8 and, to a lesser extent, objects 3 and 4 for the perturbed case. (The localization of object 8 is inherently difficult due to the large areas of uniform appearance that are present at many different poses.) The amount of work that the algorithm performs, i.e., the computation time, is directly related to the difficulty of the detection problem (see Fig. 8). In general, objects that are more difficult to localize (objects 5, 7, and 11 in the perturbed set) require the most computation time.

IV. Conclusion

This paper has presented an algorithm based on applying eigenspace methods to a quadtree representation of a set of related images to solve the pose detection problem in the presence of occlusion. Because the algorithm relies purely on the appearance of the objects in the training set of images, it is very general and easy to apply. The difficulties that are created due to the presence of occlusion, i.e., the inability to easily locate the desired object and apply the appropriate normalizations, are efficiently overcome by the recursive quadtree procedure. While on-line detection times can be an order of magnitude larger than for unoccluded images, the amount of work is proportional to the difficulty of the problem, i.e., the extent of the occlusion. In addition, the algorithm rarely makes an error in detecting the location and pose of the desired object, preferring to declare the detection problem too difficult when too much information is occluded.

References

[1] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, London, second edition, 1990.

[2] L. Sirovich and M. Kirby, "Low-dimensional procedure for the characterization of human faces," *Journal of Optical Society of America*, vol. 4, no. 3, pp. 519–524, March 1987.

[3] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, March 1991.

[4] H. Murase and R. Sakai, "Moving object recognition in eigenspace representation: Gait analysis and lip reading," *Pattern Recognition Letters*, vol. 17, no. 2, pp. 155–162, Feb. 1996.

[5] G. Chiou and J.-N. Hwang, "Lipreading from color video," *IEEE Trans. Image Processing*, vol. 6, no. 8, pp. 1192–1195, Aug. 1997.

[6] H. Murase and S. K. Nayar, "Illumination planning for object recognition using parametric eigenspaces," *IEEE Trans. PAMI*, vol. 16, no. 12, pp. 1219–1227, Dec. 1994.

[7] H. Murase and S. K. Nayar, "Visual learning and recognition of 3-D objects from appearance," *Int. J. Computer Vision*, vol. 14, no. 1, pp. 5–24, 1995.

[8] H. Murase and S. K. Nayar, "Detection of 3D objects in cluttered scenes using hierarchical eigenspace," *Pattern Recognition Letters*, vol. 18, no. 4, pp. 375–384, 1997.

[9] S. K. Nayar, S. A. Nene, and H. Murase, "Subspace method for robot vision," *IEEE Trans. Robot. Automat.*, vol. 12, no. 5, Oct. 1996.

[10] C-Y. Chang, A. A. Maciejewski, and V. Balakrishnan, "Fast Eigenspace Decomposition of Correlated Images," *IEEE Trans. Image Processing*, Vol. 9, No. 11, pp. 1937-1949, Nov. 2000.

[11] K. Ohba and K. Ikeuchi, "Detectability, uniqueness, and reliability of eigen windows for stable verification of partially occluded objects," *IEEE Trans. PAMI*, vol. 19, no. 9, pp. 1043–1048, Sept. 1997.

[12] M. Uenohara and T. Kanade, "Use of Fourier and Karhunen-Loeve decomposition for fast pattern matching with a large set of templates," *IEEE Trans. PAMI*, vol. 19, no. 8, pp. 891–898, Aug. 1997.

TABLE I
Algorithm performance on 500 UNPERTURBED test images

Object	No. Un-registered	No. Mis-registered	Orientation Error $> 4^o$	Orientation Error $> 8^o$	Total Errors	Percent Correct
1	1	3	15	3	6	98.6
2	1	0	0	0	0	99.8
3	4	9	25	7	16	96.0
4	8	10	28	4	14	95.6
5	8	0	12	2	2	98.0
6	36	0	4	0	0	92.8
7	62	7	15	7	14	84.8
8	6	39	21	6	45	89.8
9	3	1	12	0	1	99.2
10	4	0	2	2	2	98.8
11	35	0	6	0	0	93.0
12	2	5	9	5	10	97.6
13	0	2	9	4	6	98.8
14	1	0	19	6	6	98.6
15	4	19	40	1	20	95.2
16	24	0	3	0	0	95.2

TABLE II
Algorithm performance on 500 PERTURBED test images

Object	No. Un-registered	No. Mis-registered	Orientation Error $> 4^o$	Orientation Error $> 8^o$	Total Errors	Percent Correct
1	7	22	47	7	29	92.8
2	6	0	6	0	0	98.8
3	7	28	78	27	55	87.6
4	7	19	89	26	45	89.6
5	87	8	38	8	16	79.4
6	50	1	26	3	4	89.2
7	60	24	32	18	42	79.6
8	5	78	36	9	87	81.6
9	8	0	45	9	9	96.6
10	11	0	55	15	15	94.8
11	60	2	19	1	3	87.4
12	6	6	22	6	12	96.4
13	1	5	25	4	9	98.0
14	6	6	44	9	15	95.8
15	1	36	57	6	42	91.4
16	33	2	8	1	3	92.8

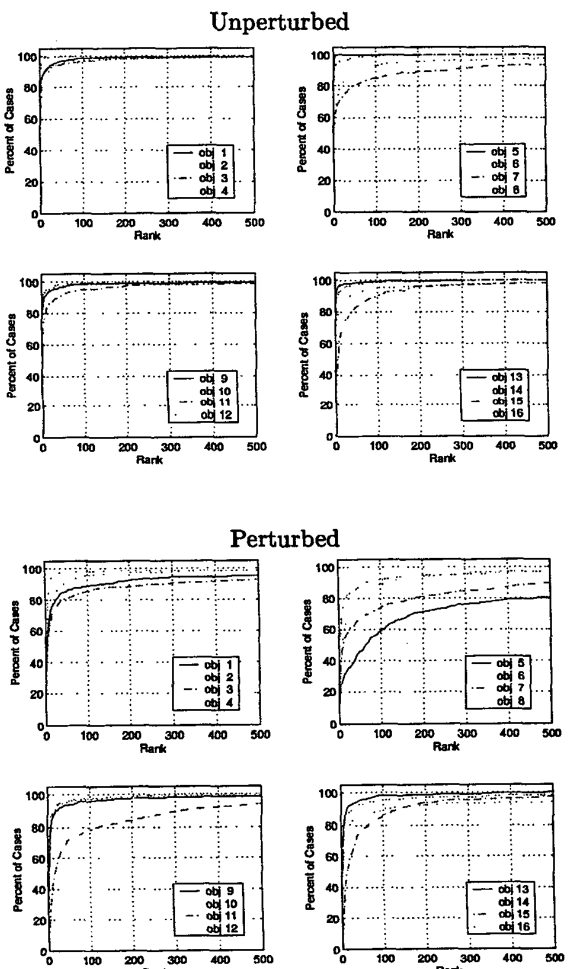

Fig. 7. Performance of the proposed algorithm when using the measure M to select candidate locations. The percent of cases that have the rank of the correct location smaller than the value of the x-axis is displayed.

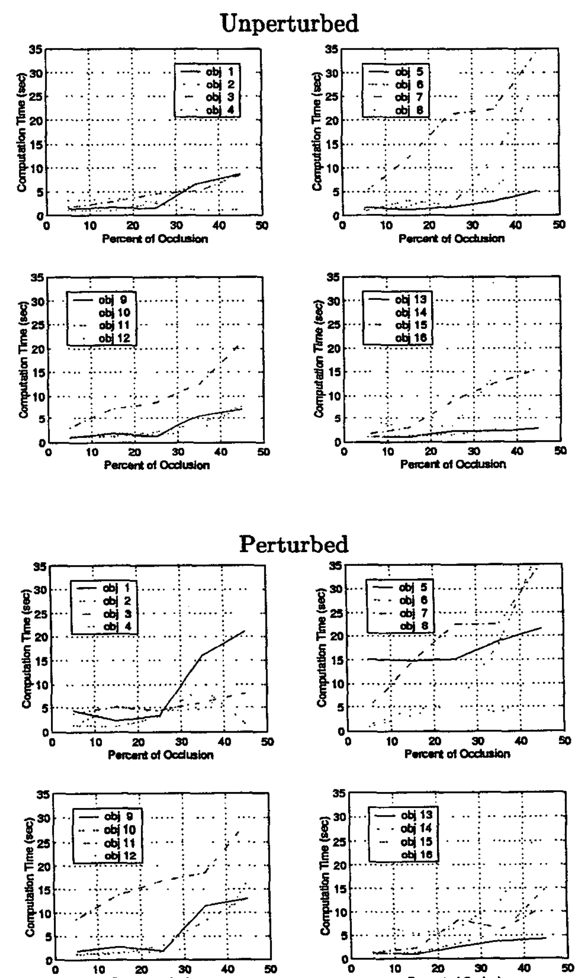

Fig. 8. Average computation time as a function of percent of occlusion. (All programs are written in MATLAB and executed on an HP9000/C110 workstation.) This result is for the experiment in section III.