# An Eigenspace Update Algorithm for Image Analysis

S. Chandrasekaran[†], B.S. Manjunath[†], Y.F. Wang[‡], J. Winkeler[†], and H. Zhang[‡]

[‡]Department of Computer Science

[†]Department of Electrical and Computer Engineering

University of California, Santa Barbara, CA 93106

April 18, 1996 9:44 am

**Abstract**

*During the past few years several interesting applications of eigenspace representation of the images have been proposed. These include face recognition, video coding, and pose estimation. However, the vision research community has largely overlooked parallel developments in signal processing and numerical linear algebra concerning efficient eigenspace updating algorithms. These new developments are significant for two reasons: Adopting them will make some of the current vision algorithms more robust and efficient. More important is the fact that incremental updating of eigenspace representations will open up new and interesting research applications in vision such as active recognition and learning. The main objective of this paper is to put these in perspective and discuss a new updating scheme for low numerical rank matrices that can be shown to be numerically stable and fast. A comparison with a non-adaptive SVD scheme shows that our algorithm achieves similar accuracy levels for image reconstruction and recognition at a significantly lower computational cost. We also illustrate applications to adaptive view selection for 3D object representation from projections.*

## 1 Introduction

The eigenspace representation of images has attracted much attention recently among vision researchers [8]-[14]. The basic idea is to represent images or image features in a transformed space where the individual features are uncorrelated. For a given set of (deterministic) images this can be achieved by performing the Singular Value Decomposition (SVD). The statistical equivalent of this is the Karhunen-Loeve Transform (KLT) which is computed by diagonalizing the autocorrelation matrix of the image ensemble. Both are well known techniques in image processing. However, they are computationally expensive.

Since computing SVD is expensive, there is a need for efficient algorithms for SVD updating. In the updating problem, one is interested in computing the new SVD when a row (or a column) is added to a given matrix whose SVD we already know. The idea of SVD updating has been prevalent in signal processing for about two decades. One of the first papers on the *numerical* issues of updating matrix factorizations appeared in 1974 [2]. However, till recently there was no fast and stable updating algorithm for the SVD [4].

In the context of image analysis in eigenspace, this paper makes the following contributions:

- We provide a comparison of some of the popular techniques existing in the vision literature for SVD/KLT computations, and point out the problems associated with those techniques.
- We outline a low-rank SVD update algorithm which is efficient and numerically stable. Using this we suggest a technique for adaptively modifying the number of basis vectors and provide an error analysis.
- We provide preliminary experimental results for the case of 3D object representation using image projections. Other interesting applications in vision are identified.

Although SVD updating techniques have been used by several researchers in the past, to the best of our knowledge this is the first time that a scheme is suggested for adaptively modifying the number of basis vectors.

Let us consider the following scenario: A camera is mounted on a robot which explores a 3D object by viewing it from different angles, and builds an internal representation in terms of image projections. This is a slightly different formulation from the face recognition problem introduced in [13] and later made popular by [14]. In all these cases, we need to be able to recognize an object from its projections only. We assume that image data are directly used in building a representation, but the formulation is valid for any set of image features extracted from the image data. As the sensor acquires each new image, the image is analyzed to determine if it is a salient image (the image saliency is measured by how much new information is embedded within or how different the image is from the current eigenspace representation), and the current representation is updated accordingly. Since the updating algorithm is based on computing the singular values of a matrix composed of images, we begin with a brief review of the eigenimage representation.

### 1.1 Eigenimage Representation

In the following discussion, we shall use the standard Euclidean 2-norm denoted by $\| . \|$ :

$$\|x\| = \sqrt{\sum x_i^2} \quad \text{for } x \in \Re^n . \tag{1}$$

Then, for a matrix $B$,

$$\|B\| = \max_{\|x\| = 1} \|Bx\| . \tag{2}$$

Let $\{A_i\}$ denote a sequence of image vectors, obtained by row-scanning the two-dimensional images with $m$ pixels in each image. Let $B_i$ denote the matrix $[A_1\ A_2\ \ldots\ A_i]$ . Let $\varepsilon$ be a given tolerance and define the $\varepsilon$-rank of $B_i$ to be the number of singular values of $B_i$ greater than $\varepsilon$. Denote the $\varepsilon$-rank of $B_i$ by $k$. Therefore, if $\sigma_j$'s are the singular values of $B_i$ in non-increasing order, then $\sigma_k > \varepsilon \geq \sigma_{k+1}$. In many image processing applications, $k \ll i$. Therefore, $B_i$ can be represented efficiently by its first $k$ singular vectors and singular values. Denote the SVD of $B_i$ by

$$\begin{bmatrix} U_i & * \end{bmatrix} \begin{bmatrix} \Sigma_i & 0 \\ 0 & [\varepsilon] \end{bmatrix} \begin{bmatrix} V_i^T \\ * \end{bmatrix} , \tag{3}$$

where $U_i$ is an $m \times k$ matrix, $\Sigma_i$ is $k \times k$ diagonal matrix, and $V_i$ is an $i \times k$ matrix. Note that $U_i$ and $V_i$ are matrices whose columns are the first $k$ left- and right-singular vectors, respectively.

Note that $B_i$ can be reconstructed to $\varepsilon$ accuracy by $U_i \Sigma_i V_i^T$. That is $\left\| B - \boldsymbol{U}_i \Sigma_i \boldsymbol{V}_i^T \right\| \leq \varepsilon$. The algorithmic requirement in many applications is to compute $\{ U_i, \Sigma_i, V_i \}$ efficiently. This can be directly computed from $B_i$ by using standard SVD algorithms (e.g., Golub-Reinsch [3]). This has a complexity of $O\left(mi^2\right)$. Some researchers ([8],[14]) have suggested computing SVD by computing the eigendecomposition of $B_i^T B_i$. While this has the same complexity, its numerical properties are not as good [3]. Nevertheless, in applications involving a large number of images, computing the SVD of $B_i$ can be too slow.

In many situations (as in face recognition, database browsing, video coding, and active recognition) $\{U_{i-1}, \Sigma_{i-1}, V_{i-1}\}$ is available and this can be used to speedup the computations. We can approximately compute $\{U_i, \Sigma_i, V_i\}$ by computing the SVD of $\left[U_{i-1}\Sigma_{i-1}V_{i-1}\ A_i\right] = \hat{B}_i$. This is the approach taken in [8] but they compute the SVD of $\hat{B}_i$ by computing the eigendecomposition of $\hat{B}_i^T\hat{B}_i$. This costs $O\left(mk + k^3\right)$. While more efficient than computing the SVD of $B_i$, this still suffers from potential numerical instability [3]. Murase [10] advocates the use of iterative methods for computing the SVD/eigendecomposition. But as is well known in numerical linear algebra [3] it is difficult to get robust implementations of such iterative methods.

In this paper, we propose instead the use of a direct update algorithm to compute the SVD of $\hat{B}_i$. This algorithm has good numerical properties, and is as efficient as the approach of [8]. Moreover, for data sets with large $k$, a fast version of the algorithm with time complexity $O(mk)$ is available.

Table 1 summarizes the algorithmic differences among some previous work in vision and ours. Note that these papers address different applications and it is not our intention here to compare those other aspects. From this table, an important conclusion to be drawn is that there exists a powerful technique from numerical linear algebra (fast and stable SVD update) that has important bearings on many vision applications. This is born out by the fact these techniques have been rediscovered several times in the vision literature. There are several other papers concerning various pattern recognition applications of eigenspace representations (for example, [9], Oja's book on subspace methods [11]), but are not very relevant to our discussion here.

## 2 Updating an Eigenspace Representation

If all $N$ images of the data set are not available at the outset (as in an active sensing scenario), we would need to compute the SVD every time a significant new image is obtained. If we did the naive thing and saved all the images and computed a full-fledged SVD from scratch every time, it would cost $O\left(mN^3\right)$ time, which is too slow.

### 2.1 Adaptive eigenspace computation

We now discuss a more efficient way. Let the left-singular vectors computed by the following incremental updating algorithm after obtaining $i$ images be denoted by $\boldsymbol{U}_{k_i}$, and let $\Sigma_{k_i}$ denote the corresponding matrix of singular values, where $k_i$ is the number of columns of $\boldsymbol{U}_{k_i}$. Note that $U_i$ can be different from $\boldsymbol{U}_{k_i}$ and the effect of this approximation is studied in this section. When we acquire the new image $\boldsymbol{A}_{i+1}$ we compute a new SVD

$$\left[\boldsymbol{U}_{k_i}\Sigma_{k_i}V_{k_i}^T \quad \boldsymbol{A}_{i+1}\right] = \boldsymbol{U}'\Sigma'\boldsymbol{V}'^T. \tag{4}$$

We now choose the integer $k_{i+1}$ such that the $k_{i+1}$th singular value of $\Sigma'$ is the smallest singular value bigger than $\varepsilon_i$, where $\varepsilon_i$ depends on $\varepsilon$ (the relation between $\varepsilon$ and $\varepsilon_i$ will be developed further in Section 2.2). We then pick the first $k_{i+1}$ left singular vectors to form $\boldsymbol{U}_{k_{i+1}}$ and the corresponding singular values to form $\Sigma_{k_{i+1}}$. If $\varepsilon_i$ does not depend on $i$ it follows from elementary properties of singular values that $k_{i+1} \geq k_i$, which is intuitively obvious. We now state the algorithm more formally.

**Eigenspace Update Algorithm:**

$U = A_1/\|A_1\|$, $V = 1$, $\Sigma = \|A_1\|$
For $i = 2$ to N
 $\left[\boldsymbol{U}\Sigma V^T \quad A_i\right] = \boldsymbol{U}'\Sigma'\boldsymbol{V}'^T$;
 Find $k$ such that $\sigma_k' > \varepsilon_i \geq \sigma'_{k+1}$;
 Let $\boldsymbol{U}$ equal the first $k$ columns of $\boldsymbol{U}'$;
 Let $V$ equal the first $k$ columns of $V'$;
 Let $\Sigma$ equal the leading $k \times k$ principal submatrix of $\Sigma$;
End

In practice there is no need to update the SVD for every new image. Only those images which are significantly outside the current object eigenspace, or those that cause a large change in the singular values need be updated. We now give more details about the computation, $\left[\boldsymbol{U}\Sigma V^T \quad A_i\right] = \boldsymbol{U}'\Sigma'\boldsymbol{V}'^T$, in the above algorithm.

Let $\boldsymbol{U}_{k_i}\Sigma_{k_i}V_{k_i}^T$ denote an approximate low-rank SVD of $B_i$, where $U_{k_i} \in \Re^{m \times k_i}$, $\Sigma_{k_i} \in \Re^{k_i \times k_i}$, and $V_{k_i} \in \Re^{i \times k_i}$. Assume that $B_i + E_i = \boldsymbol{U}_{k_i}\Sigma_{k_i}V_{k_i}^T$. We now show how to compute an approximate low-rank SVD of $B_{i+1} = (B_i \ \ A_{i+1})$ efficiently, using the available low-rank approximate SVD of $B_i$.

Observe that

$$B_{i+1} = \left(\boldsymbol{U}_{k_i}\Sigma_{k_i}V_{k_i}^T \quad A_{i+1}\right) + (E_i \ \ 0) \tag{5}$$

Therefore we can concentrate on computing the SVD of the matrix $\left(\boldsymbol{U}_{k_i}\Sigma_{k_i}V_{k_i}^T \quad A_{i+1}\right)$, which we denote by $\hat{B}_{i+1}$.

Note that if $U_{k_i}$ and $V_{k_i}$ are square matrices then we can easily diagonalize the first $k_i$ columns of $\hat{B}_{i+1}$ and obtain a *broken arrow-head* matrix, whose SVD can be computed quickly using the techniques suggested by Gu and Eisenstat [4]. But $U_{k_i}$ and $V_{k_i}$ are not square matrices. Notice that $U_{k_i}$ can be extended by adding the part of the new image $A_{i+1}$ which is perpendicular to $U_{k_i}$. It turns out that this is all we need as far as $U_{k_i}$ is concerned. Now for $V_{k_i}$ we extend it by adding a zero row and a zero column with a one in the bottom right entry. These extensions leave the columns of the extended $U_{k_i}$ and $V_{k_i}$ orthogonal. Using these extensions, computing the SVD of $\hat{B}_{i+1}$ can be reduced to computing the SVD of a *broken arrow-head* matrix. The details are given in the algorithm below.

1. $x \leftarrow U_{k_i}^T A_{i+1}$; $x \in \Re^{k_i \times 1}$.
2. $a'_\perp \leftarrow A_{i+1} - U_{k_i}x$; $a_\perp \leftarrow a'_\perp / \|a'_\perp\|$.

3. Compute the SVD of $\begin{pmatrix} \Sigma_{k_i} & U_{k_i}^T A_{i+1} \\ 0 & a_\perp^T A_{i+1} \end{pmatrix} = W \Lambda Q^T$, where $W, \Lambda, Q \in \Re^{(k_i+1) \times (k_i+1)}$.

4. $U'_{k_{i+1}} \leftarrow \begin{pmatrix} U_{k_i} & a_\perp \end{pmatrix} W$.

5. $V'_{k_i+1} \leftarrow \begin{pmatrix} V_{k_i} & 0 \\ 0 & 1 \end{pmatrix} Q$.

6. $\Sigma'_{k_i+1} \leftarrow \Lambda$.

It is straightforward to check that $\hat{B}_{i+1} = \boldsymbol{U}'_{k_{i+1}} \Sigma'_{k_{i+1}} (V'_{k_{i+1}})^T$; i.e., the above algorithm computes the SVD of $\hat{B}_{i+1}$. Once we have the SVD of $\hat{B}_{i+1}$ we can make a rank decision based on a parameter $\varepsilon_{i+1}$. We discard all singular values (and associated singular vectors) which are smaller than $\varepsilon_{i+1}$ to get $\boldsymbol{U}_{k_{i+1}} \Sigma_{k_{i+1}} (V_{k_{i+1}})^T$, the approximate low-rank SVD of $B_{i+1}$. Let $\hat{B}_{i+1} = \boldsymbol{U}_{k_{i+1}} \Sigma_{k_{i+1}} (V_{k_{i+1}})^T + G_{i+1}$. Then the matrix $G_{i+1}$ contains the errors due to truncation of the SVD and also the round-off errors incurred in computing the SVD of $\hat{B}_{i+1}$.

We next look at the various factors affecting the run-time efficiency of the above algorithm.

## 2.2 Flop count

Steps 1 and 2 of the algorithm can be computed in $O(mk_i)$ flops, and they add up to $O(mnk)$ flops for computing the low-rank SVD of the entire matrix $B$.

Step 3 involves the computation of the SVD of a broken-arrowhead matrix. Using standard dense SVD algorithms (see LAPACK manual [1]) this can be done in $O\left(k_i^3\right)$ flops. So the total cost will be $O\left(nk^3\right)$ flops. If we use the fast stable algorithm of Gu and Eisenstat [4] for computing the SVD of broken-arrowhead matrices the cost can be further reduced to $O\left(nk^2\right)$. (But the overheads in the implementation may make this worthwhile only for $k$ bigger than 100.)

Step 4 costs $O\left(mk_i^2\right)$ flops if done in a straightforward manner. Again, this can be speeded up to $O(mk_i)$ flops using the fast and stable algorithm, outlined in [4]. Thus the total cost is $O\left(mnk^2\right)$ flops or $O(mnk)$ flops.

For step 5 the cost is $O\left(ik_i^2\right)$ flops if done in a straightforward manner, and $O(ik_i)$ flops using the technique outlined in [4]. The total cost is either $O\left(n^2k^2\right)$ flops or $O\left(n^2k\right)$ flops respectively.

So we see that the dominant cost of the algorithm is the matrix multiplications. If they are done in a straightforward manner the total cost of the algorithm is $O\left(mnk^2\right)$ flops. This should be compared with the cost of computing the SVD of $B$ once, $O\left(mn^2\right)$ flops. If we use the fast and stable algorithm of [4], the total number of flops can be reduced to $O(mnk)$, though this will be useful only if $k$ is large enough.

In summary, the algorithm is as efficient as possible and all that is left to be considered is its accuracy.

### 2.3 Accuracy

The two primary sources of errors are the round-off errors incurred in computing the SVD of $\hat{B}_{i+1}$ and the error from truncating the SVD. The standard algorithms for computing the SVD (see the LAPACK manual [1]) and the fast algorithm of [4] are backward stable. Therefore step 3 in our algorithm is numerically stable. The potential source of instability is step 2, where we need to ensure that $a_\perp$ is numerically perpendicular to $U_{k_i}$. If $a'_\perp$ is very small, then $a_\perp$ as computed may no longer be numerically perpendicular to $U_{k_i}$. This can lead to serious numerical instability. One way to fix this problem is to monitor $\|a_\perp\|$, and if it falls below a certain threshold to reorthogonalize it against $U_{k_i}$. Other options would be to use modified Gram-Schmidt or QR type orthogonalization techniques. In practice if $\|a'_\perp\|$ gets smaller than $\varepsilon$, it is usually safe to skip the updating of the SVD altogether. Since $\varepsilon$ is fairly larger than the machine precision in our application, by doing this we have never observed any numerical difficulties with our current implementation.

Of more importance than the rounding error is the error due to truncating the SVD. The error matrix $G_i$ represents both these errors. It is straight forward to show that

$$\left\| B_i - \boldsymbol{U}_{k_i} \Sigma_{k_i} V_{k_i}^T \right\| \leq \|G_2\| + \|G_3\| + \ldots + \|G_i\|$$

We can assume that $\|G_i\|$ is approximately equal to $\varepsilon_i$, the truncation parameter in the $i$-th update. Therefore one would like the $\varepsilon_i$ to satisfy $\sum_{i=1}^{n} \varepsilon_i \approx \varepsilon$. But the choice $\varepsilon_i \approx \varepsilon / n$ can be overly conservative leading to a large-rank factorisation. A better choice would be $\varepsilon_i \approx \varepsilon / \sqrt{n}$. The first set of experiments in the next section compares the accuracy in image reconstruction to that of the complete SVD and demonstrates that similar performance can be obtained at a fraction of computational cost.

## 3 Applications

Previous research has already demonstrated that the eigenspace approach is a powerful tool in recognition and pose estimation of objects from image projections [9], [14], [13]. Our objective is to illustrate the efficacy and efficiency of the incremental eigenspace updating algorithm over the traditional *batch* algorithm. While the batch algorithm is in some sense the best-case, it is computationally expensive. Furthermore, the batch algorithm is not suitable for application in a dynamic environment because the inclusion of a single new image into the image set can require a complete recomputation of the basis set. The proposed updating algorithm easily handles any number of new images in an incremental manner, without recomputing the basis set from scratch. We demonstrate the application of this algorithm to the problem of intelligent view selection, which has applications in active vision.

Three sets of experiments were conducted. The first compares the reconstruction accuracy, recognition rate, and run time of the incremental update algorithm with those of the batch algorithm. It is shown that the performance of the incremental algorithm is comparable to that of the batch algorithm at a significant savings of the computation time. The second experiment demonstrates how the incremental eigenspace updating algorithm consistently selects the object views useful for representation of the object. These experiments show that view selection is robust as initial pose varies. The final experiments show how this view selection process can be made adaptive, sampling the "view space" as a function of the object image complexity. The views less useful for object representation are sampled coarsely, while the more useful views are sampled finely.

### 3.1 Incremental Update of the SVD vs. Batch SVD

The first set of experiments compares the reconstruction and recognition performance of the proposed incremental update algorithm with that of the batch algorithm. The batch algorithm performs a single SVD on the matrix containing all images in the ensemble. Hence, it represents the best case scenario in terms of recognition and reconstruction performance and serves as the baseline for comparison.

The U.S. ARMY's FERET face image database was used in this set of experiments. The particular set used consists of 137 images from FERET plus one from a local database (the army database images are not available for publication). All the face images were normalized by registering the eye locations manually without changing the aspect ratio. Two frontal views of each person are available, one of which is used during the training phase in constructing the basis and the other for testing.

In the following experiments, comparisons are made using basis sets whose dimensions are 10% and 20% of the number of images in the collection. Figure 1(a) shows one view of the image added to the FERET database. This image is also used in constructing the basis. The reconstructed images with 10% and 20% basis (13 and 27 bases, respectively) using the incremental update and the batch update are shown in Figure 1(b)-(e). The visual reconstruction quality for the incremental algorithm is comparable with that of the batch one, as the examples show. From our experiments, a 20% basis set is often sufficient to reconstruct images with an acceptable visual quality.

Figure 2 shows the average reconstruction error as a function of the basis dimension, where the basis dimension is 10% and 20% of the size of the image collection, respectively. The average reconstruction error was computed by projecting images in the ensemble onto the subspace represented by the basis images and computing the average per-pixel residue error. As can be seen from the figure, the residue error decreases when the size of the image ensemble increases. The performance of the incremental SVD algorithm closely mimics that of the batch algorithm and the difference in average residue error was generally about 10% for different basis dimensions.

Figure 3 compares the recognition rate as a function of the basis dimension for the incremental and batch algorithms. The recognition rate was calculated as follows: We designated one set of 138 frontal facial images as the training set and the other set as the test set. Images in the training set were used to compute the basis images, where the number of bases used equaled 10% and 20% of the database size. The feature vectors (i.e., the projection coefficients onto the basis images) of the images in the training set were then computed. The feature vectors of images in the test set were computed and the distances between the two sets of feature vectors were calculated to determine the matches between images in the two data sets. Percentage of correct associations of the images in the training and test sets are plotted in Figure 3. As can be seen from this figure, the recognition rate of the incremental algorithm again closely mimics that of the batch algorithm. The recognition rate decreases when larger image databases are used; however, the difference between the two methods is about 5-10%.

The slight drop in the reconstruction and recognition accuracy of the incremental algorithm is made up by the significant savings in the computation time over the batch algorithm. This will be significant for any meaningfully large database. Computation times for the batch algorithm, which did not include efficient adaptive basis computation, are compared to those of the incremental algorithm. The results are plotted in Figure 4. As expected, this new algorithm computes the SVD much faster than the older batch method when the basis dimension is 10-20% of the image count. When the basis dimension becomes a much larger percentage, the batch algorithm will be faster. However, for large databases, constructing a basis of dimension larger than 10-20% of the total image count will not be feasible.

One approach for constructing the eigenspace representations for large databases is to use random selection to pick a subset of the images. The hope is that this randomly chosen subset will be representative of the whole image set, and that a basis set created using this subset of images will perform well for the whole image data. The proposed incremental updating algorithm has computation times comparable to these other methods. The proposed method, however, provides an adaptive means for selecting a more representative data set for computing the basis. Every image in the database is examined and only the most useful views are selected by the updating algorithm.

### 3.2 Salient Views and Basis Dimensions

The previous experiments illustrate that the incremental update algorithm performs nearly as well as the best-case batch algorithm. The rest of our experiments illustrate the strengths of the incremental update algorithm. These experiments are inherently incremental and cannot be efficiently performed with the batch algorithm.

The incremental algorithm consistently selects the most useful views for creating a basis set. To illustrate this, test objects were placed on a rotation stage and their pictures were taken every ten degrees of rotation. Eigenspace representations were constructed by examining the images as they were acquired. At times, the representation may need to be updated. The representation changes under the following three conditions: (1) when the dimensionality of the current basis image space is not sufficient to encode the new image, (2) when the singular vectors are rotated, or (3) when the addition of the new image affects the singular values only. In the first case the number of basis images increases by one. Our experimental studies indicate that when a new representation is required, usually the dimensionality also goes up. This suggests the following simplification to the update algorithm: check to see if an update is necessary, and if necessary, update the representation and enlarge the basis dimensionality by one. This check can be done by comparing the reconstruction error to a threshold. The threshold is a measure of the per pixel mean squared error allowed in the most poorly reconstructed image in the set. Hence, only those images that provided significant new information are used in constructing a new, larger representation. We call these images salient views of the corresponding object. We used this simplified update algorithm in our experiments.

A total of seven objects (Figure 5) were digitized with 36 images per object. Figure 6 and Figure 7 show the salient views recorded for two of the objects. The same threshold was used to select the salient views for both objects. As can be seen from these figures, the number of salient views depends on the complexity of the imaged object. Toy 1 has more complexity in its structure than symmetric Toy 5, and more views are selected as salient. The salient views selected for Toy 1 include only one back view of the object, all others are frontal or profile views. Note that the only information being used are the gray-level pixel intensities. For our digitized objects, a good choice of the threshold in the reconstruction error appears to be in the range 0.06 - 0.1 per pixel.

For the next experiments, three of the objects were digitized at one degree of rotation for a total of 360 images per object. Figure 8 shows the basis dimension as a function of threshold (i.e., desired representation accuracy). As expected, it clearly indicates that the basis dimension grows as the desired reconstruction error is reduced. These results are the average of measurements made at four different starting angles. These results meet our intuitive expectations, since the Toy 2 seems to be the 'simplest' object and it requires the fewest bases for reconstruction to within a certain error. Note that the face images have a much larger basis dimension requirement than the rotating objects. This is because the face-to-face variation is larger than the view-to-view variations for the objects. A more detailed set of results are shown in Table 2. Consider, for example, Toy 2 with the threshold of 0.055. The basis dimension is 19 or 20 for starting views separated by 90 degrees about the object. Nineteen or twenty salient views are needed to represent this object to within a reconstruction error of 0.055 regardless of from which angle the object is first viewed. In most cases the choice of the starting angle has little effect on the number of bases chosen for a fixed threshold. Note also that the maximum residual error in reconstruction is always near the desired threshold, even though each view is only examined once, often before the full basis set is established.

### 3.3 View selection in a dynamic environment

In the previous experiments every image in the set was examined, which may be wasteful. When one view of an object on the rotation stage is found to be well-represented, it is unlikely that the view one-degree away will be salient. Consider now the camera actively positioned about a stationary object. The camera's next position is found from its current position by considering the changes in the eigenspace representation. The camera can now adapt to the view space, skipping over any number of uninteresting views to examine only the salient views of the object. The step size between one view of the object and the next is based on how interesting the current view is and on how well it is already represented.

Our experiments simulated the adaptive positioning of the camera in increments of no less than one degree about an object. The camera was placed at an arbitrary starting point facing an object. Then the system started acquiring objects initially at one degree rotation. If the new image was close to the current eigenspace representation, the step size for the rotation was doubled. On the other hand, if the distance of the current image from the eigenspace representation was large, then the representation was updated by including this current view and the step size was halved. Figures 9-11 show the step size as a function of angle for different starting angles. The step size is large in well-modeled areas and small in areas with significant new information. Note that the step size seems to be large in the same regions of each graph. For instance, in the TOY6 graph, the area around 120 degrees is uninteresting regardless of the starting angle, and the area around 220 degrees is always interesting to the algorithm. Similar experiments were performed with the starting angle held constant and threshold varied and similar results were observed.

Both the specific choice of salient views and the size of the steps about the object are relatively constant even as the initial view or the threshold are varied. The selection of the views necessary for proper representation of the object are as robust for the adaptive algorithm as when all of the views are examined. The main refinements are the speed with which the representation can be constructed and, more importantly, the applicability to an active visual system.

### 3.4 Conclusions

The eigenspace updating technique is suitable for use in an active environment where images are acquired continuously and a representation is incrementally constructed and refined. We have shown its effectiveness in 3D object representation using 2D images which is useful in active recognition and exploration. Further research is needed to quantify image saliency in a more objective way, perhaps requiring higher level visual cues to be incorporated into the scheme. The methodology presented here is also applicable to image features such as wavelet transformed data. Our current research seeks to extend these concepts to video coding and image database problems.

**Acknowledgments.** Portions of the research in this paper use the FERET database of facial images collected under the ARPA/ARL FERET program. This research was partially supported by NSF grant IRI-9411330.

## 4 References

[1] E. Anderson et al., *LAPACK Users' Guide,* Society for Industrial and Applied Mathematics, 1992.

[2] P.E. Gill, G.H. Golub, W. Murray, and M.A. Saunders (1974). "Methods for Modifying Matrix Factorizations," Math. Comp. 28, 505-35

[3] G. H. Golub and C. F. van Loan, *Matrix Computations,* The Johns Hopkins Press, 1989.

[4] M. Gu and S. C. Eisenstat, "A Stable and Fast Algorithm for Updating the Singular Value Decomposition," Research Report YALEU/DCS/RR-966, 1994, Yale University, New Haven, CT.

[5] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. Educ. Psychology* Vol. 24, pp. 417-441 and pp. 448-520, 1933.

[6] A. Jain, Fundamentals of Digital Image Processing, Prentice Hall 1989.

[7] B. S. Manjunath, S. Chandrasekaran, and Y. F. Wang, "Learning in eigenspace: theory and application," CIPR-TR-94-17, ECE Dept., UCSB, November 1994.

[8] H. Murakami and B. V. K. V. Kumar, "Efficient calculation of primary images from a set of images," *IEEE T-PAMI*, Vol. 4, No. 5, September 1982, pp. 511-515.

[9] H. Murase and S. K. Nayar, "Visual learning of object modules from appearance," *Proc. Image Understanding Workshop 1993*, (San Diego, CA), pp. 547-555, 1993.

[10] H. Murase and M. Lindenbaum, "Partial eigenvalue decomposition of large images using the spatial temporal adaptive method," *IEEE T-IP*, Vol. 4 (5), May 1995, pp. 620-629.

[11] E. Oja, *Subspace methods of pattern recognition,* John Wiley, 1983.

[12] A. Pentland, B. Moghaddam, and T. Starner, “View-based and modular eigenspaces for face recognition,” *Proc. IEEE Conf. CVPR ‘94*, (Seattle, Washington), pp. 84-91, June 1994.
[13] L. Sirovich and M. Kirby, “Low dimensional procedure for the characterization of human faces,” J*ournal of Optical Society of America*, Vol. 4, No. 3, pp. 519-524, 1987.
[14] M. Turk and A. Pentland, “Eigenfaces for Recognition,” *Journal of Cognitive Neuroscience*, Vol. 3, No. 1, pp. 71- 86, March 1991.
[15] S. Ullman and R. Basri, “Recognition by linear combination of models,” IEEE T-PAMI, Vol. 13, No. 10, pp. 992-1006, October 1991.

Table 1: Comparison of algorithms

| **Authors** | **Method** | **Update** |
|---|---|---|
| Murakami and Kumar (1982) | $B^T B$ | yes |
| Kirby and Sirkovich (1990) | $BB^T$ | no |
| Turk and Pentland (1991) | $B^T B$ | no |
| Murase and Lindenbaum (1995) | iterative ($BB^T$) | no |
| This paper | SVD of $B$ | yes |

Table 2: Toy 2, Toy 6 and Toy 7 statistics.

| Threshold | | θ | Basis Dimension | | | $\delta max$ | | | time(seconds) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Toy 2 and Toy 6 | Toy 7 | | Toy 2 | Toy 6 | Toy 7 | Toy 2 | Toy 6 | Toy 7 | Toy 2 | Toy 6 | Toy 7 |
| 0.09 | 0.12 | 0 | 2 | 2 | 3 | 0.080 | 0.106 | 0.137 | 3.73 | 3.27 | 5.95 |
| | | 90 | 1 | 3 | 5 | 0.092 | 0.090 | 0.121 | 2.52 | 6.35 | 15.44 |
| | | 180 | 1 | 2 | 4 | 0.096 | 0.093 | 0.134 | 2.54 | 4.09 | 9.63 |
| | | 270 | 2 | 2 | 5 | 0.087 | 0.105 | 0.113 | 3.80 | 3.71 | 16.36 |
| 0.08 | 0.11 | 0 | 3 | 5 | 7 | 0.077 | 0.092 | 0.110 | 6.40 | 15.53 | 32.43 |
| | | 90 | 1 | 7 | 9 | 0.092 | 0.077 | 0.106 | 2.53 | 33.01 | 60.02 |
| | | 180 | 2 | 6 | 8 | 0.090 | 0.084 | 0.102 | 3.27 | 23.90 | 45.32 |
| | | 270 | 3 | 6 | 6 | 0.075 | 0.085 | 0.113 | 6.90 | 22.32 | 120.23 |
| 0.07 | 0.10 | 0 | 5 | 17 | 9 | 0.076 | 0.074 | 0.110 | 16.35 | 313.13 | 59.36 |
| | | 90 | 3 | 14 | 10 | 0.089 | 0.076 | 0.101 | 6.93 | 185.90 | 77.15 |
| | | 180 | 5 | 15 | 10 | 0.079 | 0.076 | 0.101 | 15.51 | 221.09 | 78.4 |
| | | 270 | 4 | 16 | 12 | 0.072 | 0.070 | 0.100 | 10.58 | 259.44 | 91.01 |
| 0.065 | 0.09 | 0 | | 26 | 17 | | | 0.079 | | | 367.77 |
| | | 90 | | 20 | 20 | | | 0.089 | | | 490.38 |
| | | 180 | | 25 | 19 | | | 0.085 | | | 424.44 |
| | | 270 | | 25 | 18 | | | 0.091 | | | 368.72 |
| 0.06 | | 0 | 11 | | | 0.073 | | | 98.82 | | |
| | | 90 | 8 | | | 0.079 | | | 43.50 | | |
| | | 180 | 15 | | | 0.058 | | | 225.32 | | |
| | | 270 | 13 | | | 0.061 | | | 154.25 | | |
| 0.055 | | 0 | 20 | | | 0.053 | | | 483.43 | | |
| | | 90 | 20 | | | 0.050 | | | 483.81 | | |
| | | 180 | 19 | | | 0.059 | | | 425.70 | | |
| | | 270 | 19 | | | 0.055 | | | 428.82 | | |

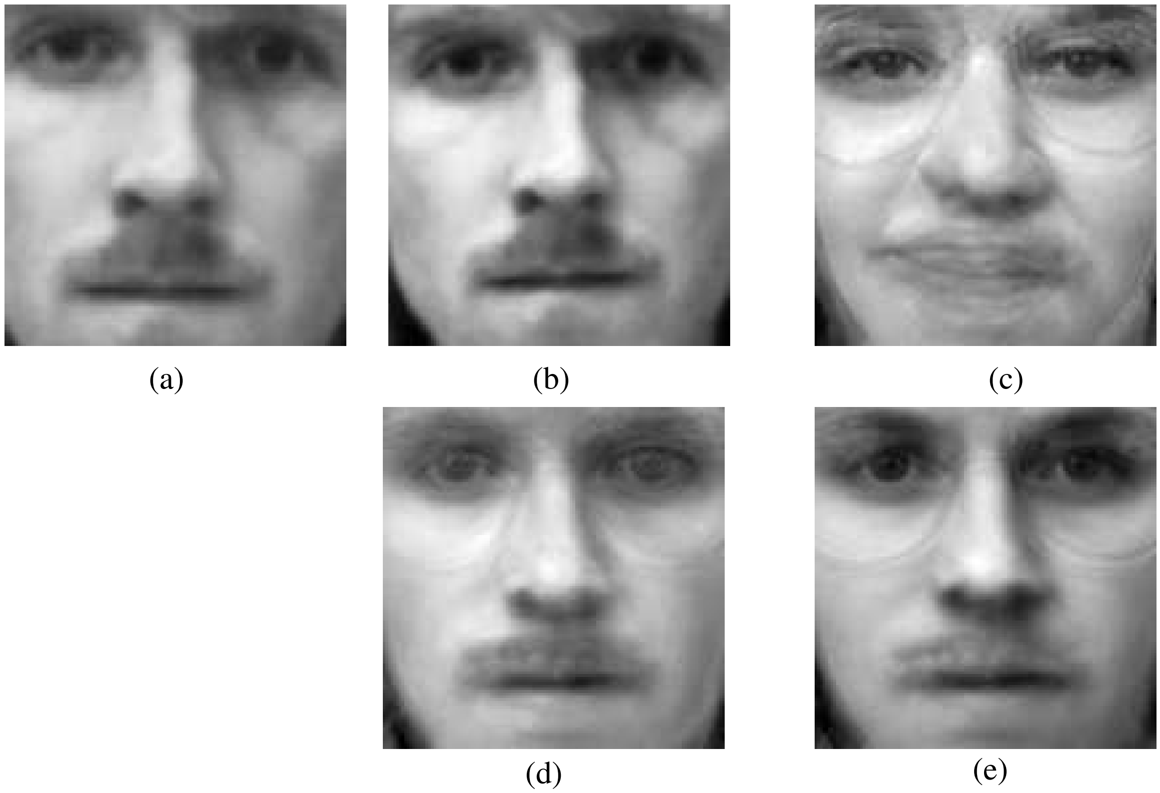



Figure 1: Face reconstructions using a basis dimension equal to 20% and 10% of the image count. The basis sets were created with the incremental and batch algorithms (a) the original image, (b) incremental algorithm with 20% basis, residue error 0.000023, (c) incremental algorithm with 10% basis, residue error 0.211824, (d) batch algorithm with 20% basis, residue error 0.084062, (e) batch algorithm with 10% basis, residue error 0.126791.

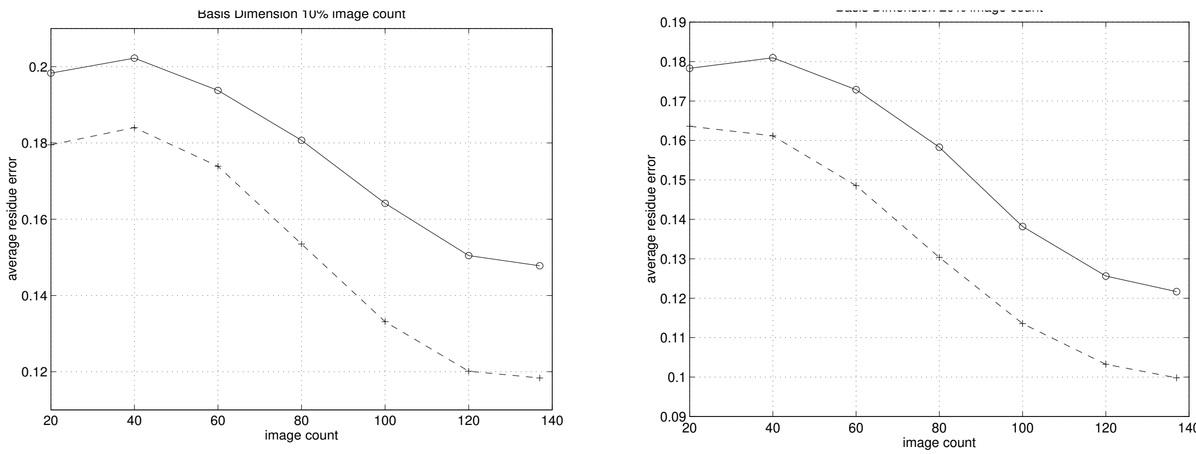



Figure 2: Residue error for basis dimension (a) 10% of image count, and (b) 20% image count. These results are for the test data. Dashed line: batch algorithm, solid line: incremental update algorithm.

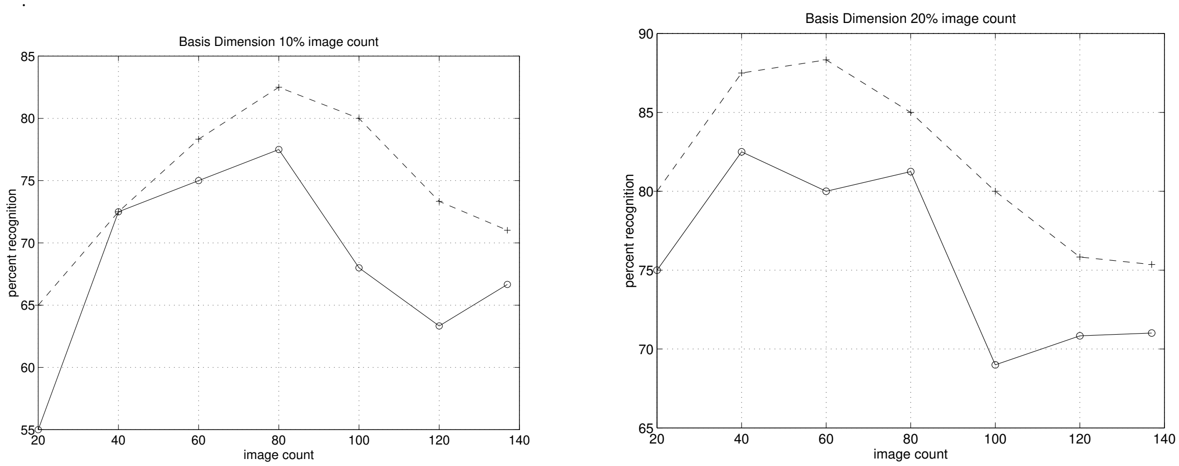



Figure 3: Percent recognition for basis dimension 10% and 20% of image count. Dashed line: batch algorithm, solid line: incremental update algorithm.

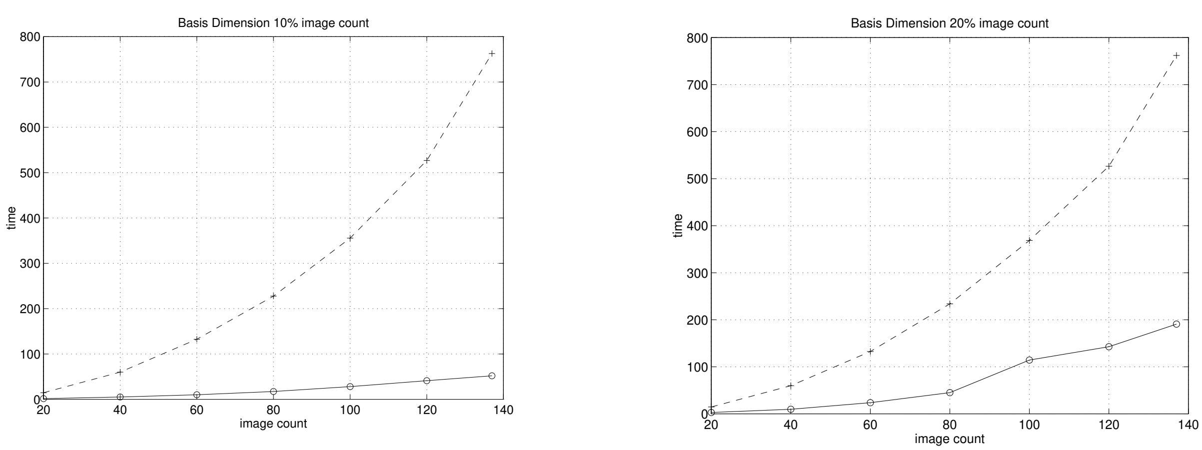
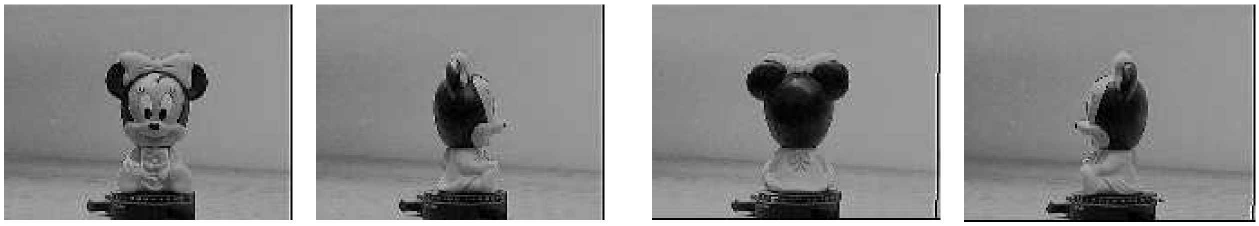
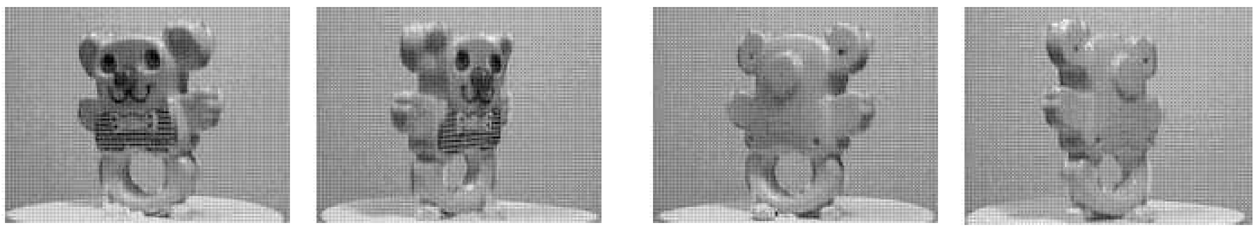


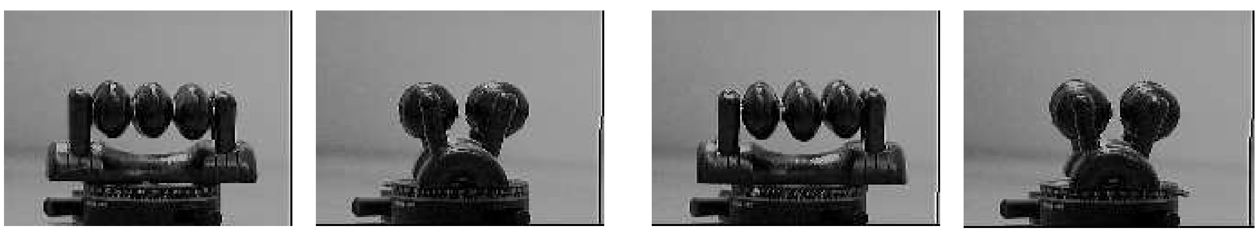

Figure 4: Computation time for basis dimension 10% and 20% of image count. Dashed line: batch algorithm, solid line: incremental update algorithm.



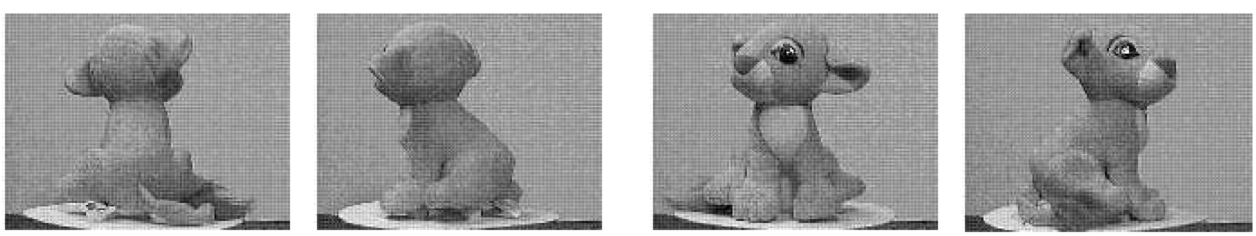
Toy 1



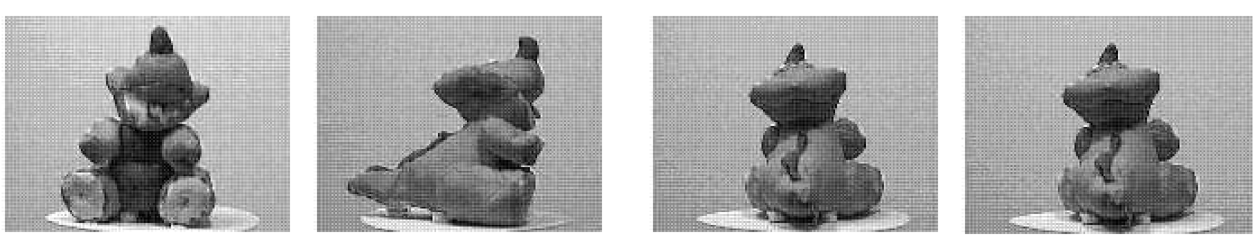
Toy 2



Toy 5



Toy 6



Toy 7

Figure 5: Test objects at 0, 90, 180 and 270 degrees, respectively

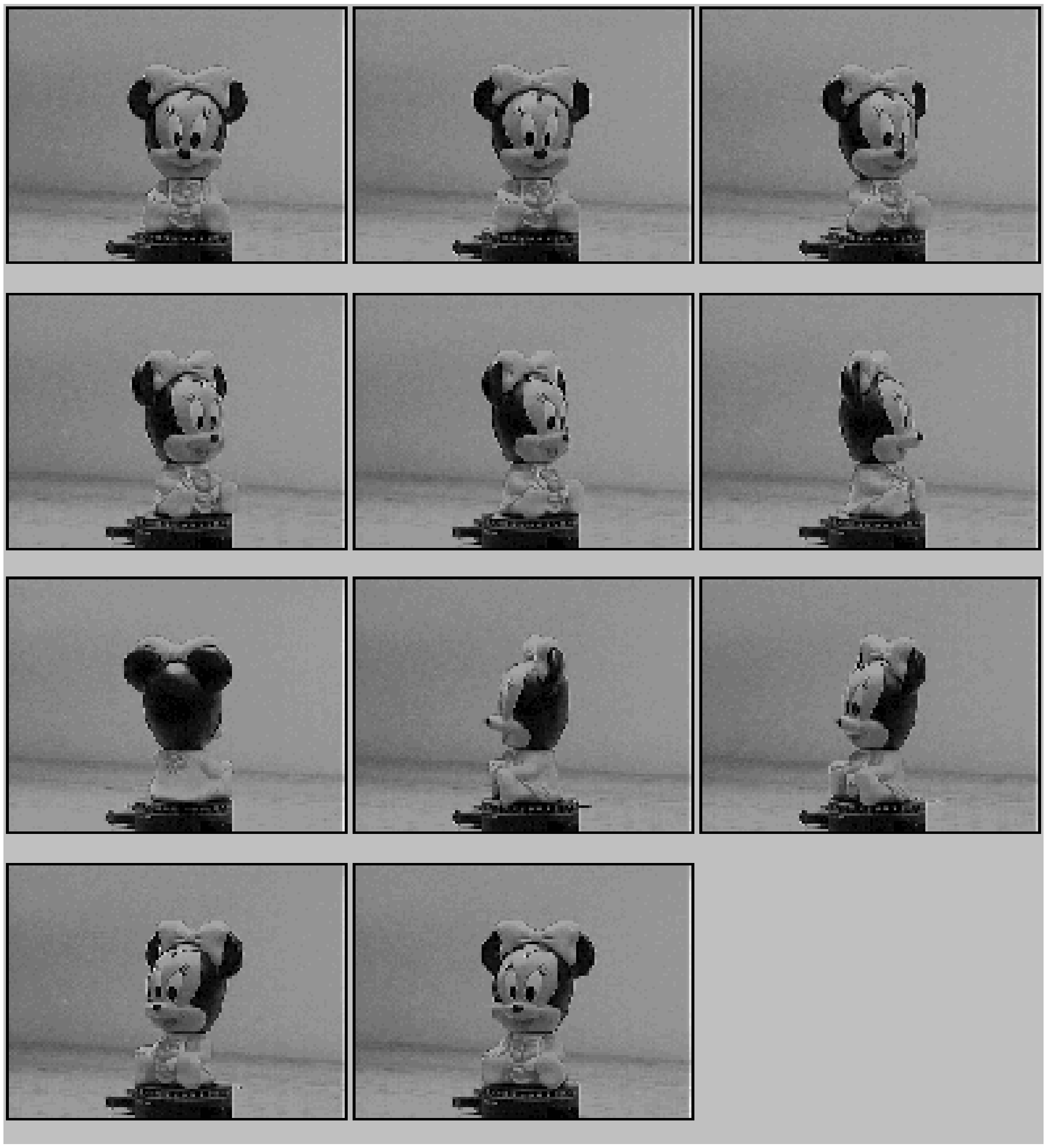


Figure 6: Salient views of Toy 1.

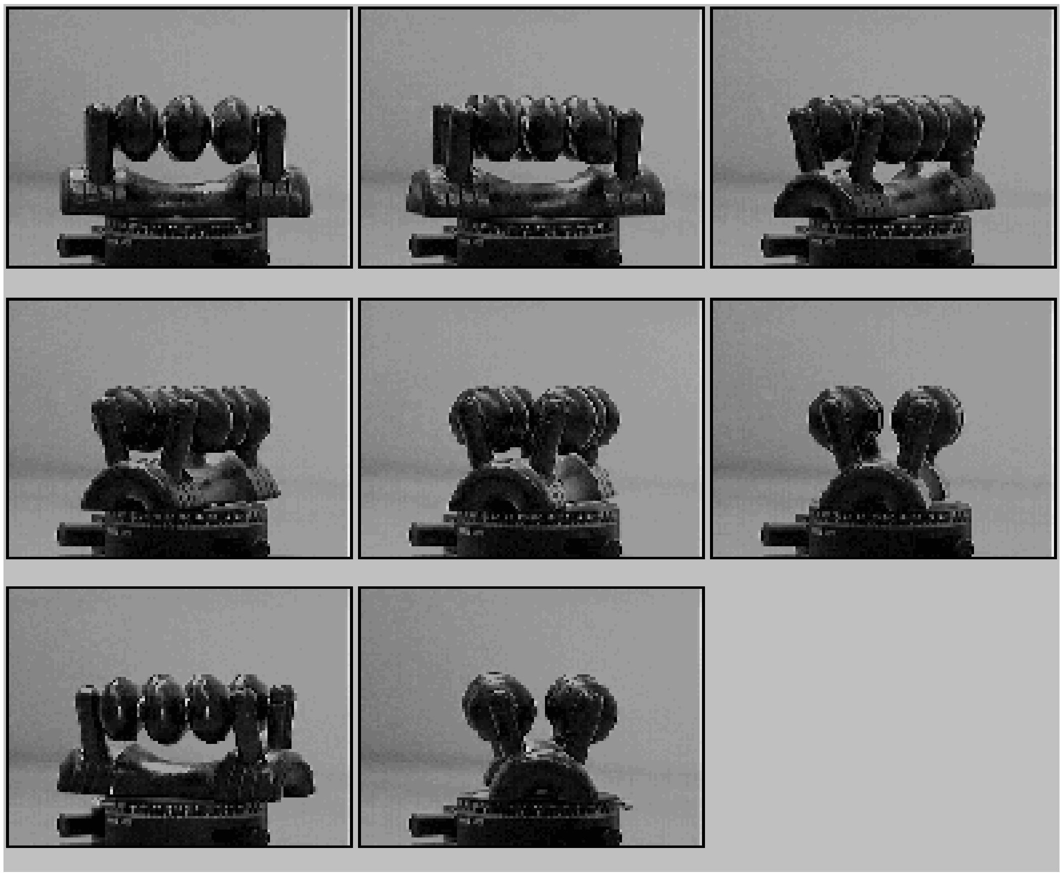


Figure 7: Salient views of Toy 5.

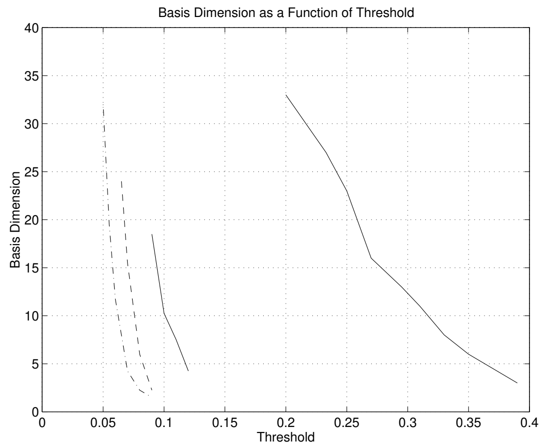



Figure 8: Basis dimension as a function of threshold. Dot-dash line: Toy 2, dashed line: Toy 6, short solid line: Toy 7, long solid line: face database.

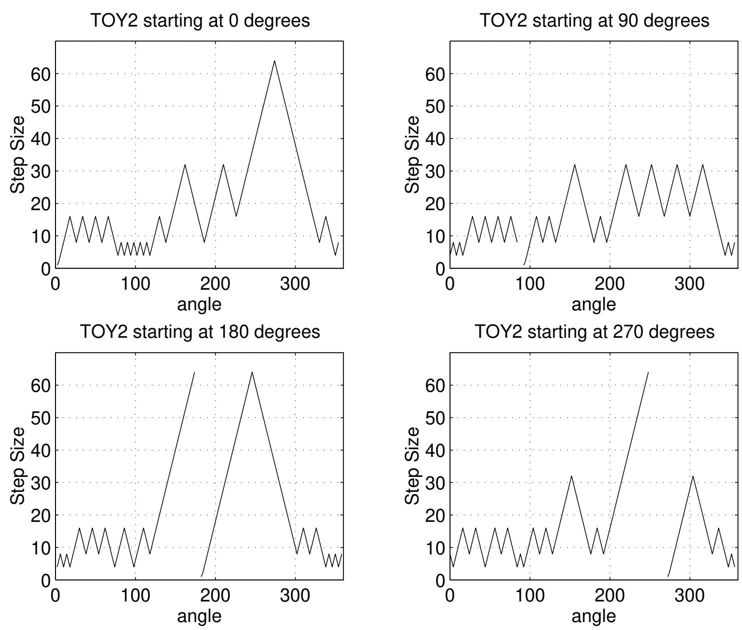



Figure 9: Toy 2 step size for different starting angles.

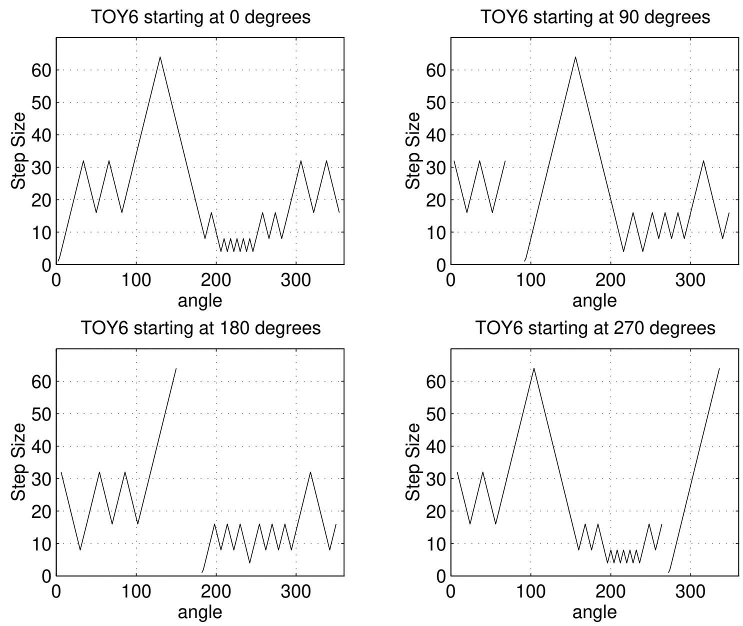



Figure 10: Toy 6 step size for different starting angles.

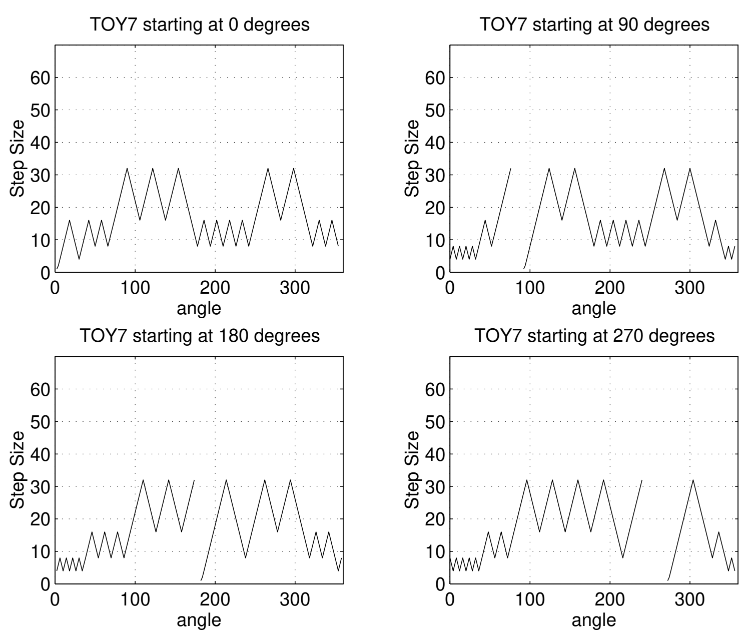



Figure 11: Toy 7 step size for different starting angles.