APPEARANCE-BASED RECOGNITION

- REPRESENT APPEARANCE (IMAGE BRIGHTNESS), NOT GEOMETRY
- WHY?
  AVOIDS PROBLEMS of · GEOMETRIC SHAPE REP DECISIONS, · LEARNING OBJECT MODELS, · DEALING w/ COMPLEX INTERACTIONS B/W SHAPE, ILLUMINATION, REFLECTANCE, ETC., · CALIBRATION
- WHY NOT?
  - TOO MANY POSSIBLE APPEARANCES

    m "visual DOFs" (e.g. pose, lighting)

    R_i discrete samples of i^{th} DOF

    $\Rightarrow \prod_{i=1}^{m} R_i$ images (appearances)

  - HOW TO DISCRETELY SAMPLE THE VISUAL DOFs?
  - HOW TO "PREDICT"/SYNTHESIZE/MATCH w/ NOVEL VIEWS?

- EXAMPLE

  VISUAL DOFs:

  1) Object type P
  2) Pose R
  3) Illumination direction L

$\Rightarrow$ Image Set $= \mathcal{X} = \{ \hat{x}^{(p)}_{R,L} \} =$

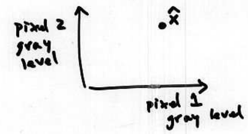
$\{ \hat{x}^{(1)}_{1,1}, \ldots, \hat{x}^{(1)}_{R,1}, \hat{x}^{(1)}_{1,2}, \ldots, \hat{x}^{(1)}_{R,L}, \ldots \}$

$|\mathcal{X}| = RLP$

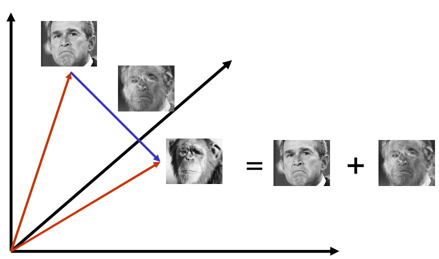
where $\hat{x}$ = image

$= N \times 1$ matrix of pixel values in raster order

$\Rightarrow$ image = point in N-dimensional image space



The Space of Faces



- An image is a point in a high dimensional space
  - An N x M image is a point in R^{NM}

KEY IDEA:

Images in $\{ \hat{x}_i \}$ are highly correlated, so compress them into low-dimensional subspace that captures key appearance characteristics of the visual DOFs.

$\Rightarrow$ EIGENSPACE REPRESENTATION

Given M training images, each w/ N pixels

1) Preprocess images: $\hat{x}_i \rightarrow x_i$

   where $x_i = [x_1, x_2, \ldots, x_N]^T$

   to reduce noise, normalize, etc.

   E.g. brightness normalization $\Rightarrow$

   $x_i = \hat{x}_i / \| x_i \|$

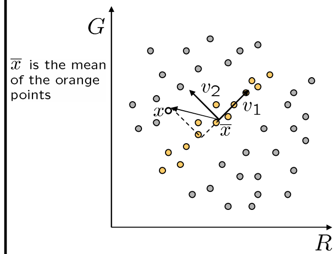
   Make X have 0 mean, 1 variance, scaled, segmented

2) Project image into low-dimensional sub-space

   What sub-space?

   "Best" characteristic feature images defined by best eigenvectors

Linear Subspaces



convert **x** into $\mathbf{v_1}$, $\mathbf{v_2}$ coordinates

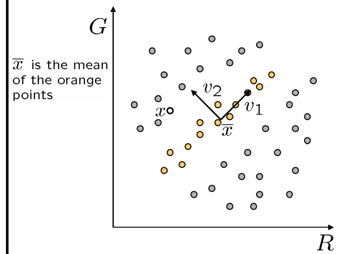
$$\mathbf{x} \to ((\mathbf{x} - \overline{x}) \cdot \mathbf{v}_1, (\mathbf{x} - \overline{x}) \cdot \mathbf{v}_2)$$

What does the $\mathbf{v_2}$ coordinate measure?
- distance to line
- use it for classification—near 0 for orange pts

What does the $\mathbf{v_1}$ coordinate measure?
- position along line
- use it to specify which orange point it is

- Classification is still expensive
  - Must either search (e.g., nearest neighbors) or store large PDF's
- Suppose the data points are arranged as above?
  - Idea—fit a line, classifier measures distance to line

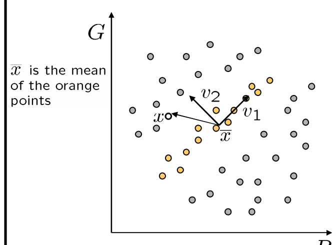
Dimensionality Reduction



How to find $\mathbf{v_1}$ and $\mathbf{v_2}$?

- Dimensionality reduction
  - We can represent the orange points with *only* their $\mathbf{v_1}$ coordinates
    - since $\mathbf{v_2}$ coordinates are all essentially 0
  - This makes it much cheaper to store and compare points
  - A bigger deal for higher dimensional problems

Linear Subspaces



Consider the variation along direction **v** among all of the orange points:

$$var(\mathbf{v}) = \sum_{\text{orange point } \mathbf{x}} \|(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \cdot \mathbf{v}\|^2$$

What unit vector **v** minimizes *var*?

$$\mathbf{v}_2 = min_{\mathbf{v}} \ \{var(\mathbf{v})\}$$

What unit vector **v** maximizes *var*?

$$\mathbf{v}_1 = max_{\mathbf{v}} \ \{var(\mathbf{v})\}$$

$$\begin{aligned} var(\mathbf{v}) &= \sum_{\mathbf{x}} \|(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \cdot \mathbf{v}\| \\ &= \sum_{\mathbf{x}} \mathbf{v}^{\mathrm{T}}(\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}}\mathbf{v} \\ &= \mathbf{v}^{\mathrm{T}}\left[\sum_{\mathbf{x}}(\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}}\right]\mathbf{v} \\ &= \mathbf{v}^{\mathrm{T}}\mathbf{A}\mathbf{v} \quad \text{where } \mathbf{A} = \sum_{\mathbf{x}}(\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \end{aligned}$$

Solution: $\mathbf{v_1}$ is eigenvector of **A** with *largest* eigenvalue
$\mathbf{v_2}$ is eigenvector of **A** with *smallest* eigenvalue

Principal Component Analysis

- Suppose each data point is N-dimensional
  - Same procedure applies:

$$\begin{aligned} var(\mathbf{v}) &= \sum_{\mathbf{x}} \|(\mathbf{x} - \overline{\mathbf{x}})^{\mathbf{T}} \cdot \mathbf{v}\| \\ &= \mathbf{v}^{\mathbf{T}}\mathbf{A}\mathbf{v} \quad \text{where } \mathbf{A} = \sum_{\mathbf{x}}(\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathbf{T}} \end{aligned}$$

  - The eigenvectors of **A** define a new coordinate system
    - eigenvector with largest eigenvalue captures the most variation among training vectors **x**
    - eigenvector with smallest eigenvalue has least variation
  - We can compress the data by only using the top few eigenvectors
    - corresponds to choosing a "linear subspace"
      - represent points on a line, plane, or "hyper-plane"

Best k dimensions defined by
Principal Component Analysis
(also called Karhunen-Loève transform)
and is related to least-squares methods and SVD.

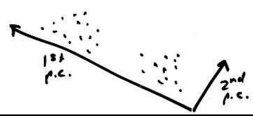
- Maximize info content in compressed data

⇒ Find a set of k orthogonal vectors that account as much as possible for data's variance

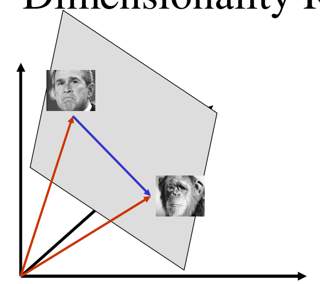
1st principal component = direction w/ max variance

2nd principal component = direction ⊥ to 1st pc and max variance

etc.



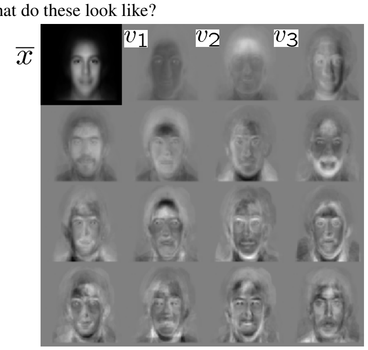
Dimensionality Reduction



- The set of faces is a "subspace" of the set of images
  - Suppose it is K dimensional
  - We can find the best subspace using PCA
  - This is like fitting a "hyper-plane" to the set of faces
    - spanned by vectors $\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_K}$
    - any face $\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_k\mathbf{v_k}$

Eigenfaces

- PCA extracts the eigenvectors of **A**
  - Gives a set of vectors $\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}, ...$
  - Each one of these vectors is a direction in face space
    - what do these look like?

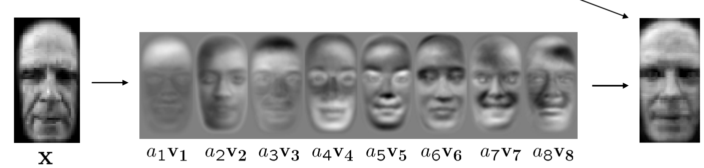


Projecting onto the Eigenfaces

- The eigenfaces $\mathbf{v_1}, ..., \mathbf{v_K}$ span the space of faces
  - A face is converted to eigenface coordinates by

$$\mathbf{x} \rightarrow (\underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot\mathbf{v_1}}_{a_1},\ \underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot\mathbf{v_2}}_{a_2}, \ldots,\ \underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot\mathbf{v_K}}_{a_K})$$

$$\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_K\mathbf{v_K}$$



Computing Subspaces

Given: $\mathcal{X} = \{x_i\} = \begin{bmatrix} \text{Image 1} & \cdots & \text{Image M} \end{bmatrix}$ N pixels

M = RLP images

- Normalize by subtracting Mean Image

$$c = \frac{1}{M} \sum_{i=1}^{M} x_i$$

$$\mathcal{X} = \{x_i - c\}$$

* This ensures that eigenvector w/ largest eigenvalue represents dimension in which variance of images is maximum in correlation sense.

- Compute Covariance Matrix

$$Q = \mathcal{X}\mathcal{X}^T \qquad N \times N \text{ matrix}$$

- Compute Eigenvalues and Eigenvectors

Solve $\lambda_i e_i = Q e_i$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$ eigenvalues

$e_i = N \times 1$ eigenvector (image)

* Eigenvectors ordered "best" to "worst," $e_1, \ldots, e_N$

$\Rightarrow$ k best $= e_1, \ldots, e_k$

- Project each Image to Eigenspace

$$g_j = e_j^T (x_i - c)$$

scalar value representing degree of match between image x_i and eigenvector e_j

$$x_i = \sum_{j=1}^{N} g_j e_j + c$$

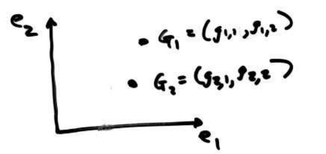
$\Rightarrow$ Image x_i reconstructed exactly by weighted sum of eigenvectors $e_1, \ldots, e_N$

Approximate description using k best eigenvectors

$$x_i \approx \sum_{j=1}^{k} g_j e_j + c$$

$\Rightarrow$ Subspace of k dimensions defined by $e_1, \ldots, e_k$

Image x_i projected to point $G_i = [e_1, \cdots, e_k]^T (x_i - c)$



Recognition with Eigenfaces

- Algorithm
  1. Process the image database (set of images with labels)
     - Run PCA to compute the eigenfaces
     - Calculate the K coefficients for each image
  2. Given a new image (to be recognized) **x**, calculate K coefficients

$$\mathbf{x} \rightarrow (a_1, a_2, \ldots, a_K)$$

  3. Detect if x is a face

$$\|\mathbf{x} - (\bar{\mathbf{x}} + a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \ldots + a_K\mathbf{v}_K)\| > \text{threhold}$$

  4. If it is a face, who is it?
     - Find closest labeled face in database
       - nearest-neighbor in K-dimensional space

Key Property of Eigenspace Rep

Given 2 images $\hat{x}_1$, $\hat{x}_2$ that are used to construct eigenspace, and G_1 is eigenspace projection of $\hat{x}_1$ and G_2 is eigenspace projection of $\hat{x}_2$, then

$$\|\hat{x}_1 - \hat{x}_2\|^2 \approx \|G_1 - G_2\|^2$$

That is, distance in eigenspace is approximately equal to the correlation between 2 images.

FACE RECOGNITION APPS

- IDENTIFICATION
  CREDIT CARDS, DRIVER'S LICENSE, PASSPORT, EMPLOYE ID
- SECURITY
  CROWD SURVEILLANCE, CHECK-CASHING ATMS, BANK OPERATIONS, BUILDING ACCESS
- HEALTH CARE
  VERIFICATION of MEDICAL RECORDS, HOSPITAL RECORD RETRIEVAL
- LAW ENFORCEMENT
  MATCH COMPOSITE IMAGE w/ DB of SUSPECTS
- MARKETING

Turk + Pentland's Recognition Method

"Eigenfaces for face Recognition"

1. Given a set of training images w/ P people and R faces/person
2. Compute k (≈ 20) best eigenvectors ("eigenfaces")
3. Foreach subset of images of <u>same</u> person, compute "average" point in eigenspace
   $$G_{mary} = [\bar{g}_{mary,1}, \bar{g}_{mary,2}, \ldots \bar{g}_{mary,k}]$$
4. Given test image X_{test}, project to eigenspace:
   $$G_{test} = [e_1, \ldots, e_k]^T (x_{test} - c)$$
5. Find training face closest to test
   $$d = \min_{\text{person } p} \|G_{test} - G_p\|^2$$

6. Find distance from "face space":
   $$dffs = \|y - y_f\|^2$$
   where $y = x_{test} - c$
   $$y_f = \sum_{i=1}^{k} g_{test,i}\, e_i$$
7. <u>If</u> $dffs < T_1$
   ; image close enough to
   ; "face space" — not a
   ; tree, etc.
   <u>then</u> <u>if</u> $d < T_2$
   <u>then</u> person k
   <u>else</u> unknown person
   <u>else</u> not a face

Face Recognition Algorithm

- Consider 2 3x3 images:

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 10 & 10 & 10 \\ 0 & 0 & 0 \end{bmatrix}}_{I_1} \qquad \underbrace{\begin{bmatrix} 0 & 10 & 0 \\ 0 & 10 & 0 \\ 0 & 10 & 0 \end{bmatrix}}_{I_2}$$

$$\Rightarrow I_1 = [0\ 0\ 0\ 10\ 10\ 10\ 0\ 0\ 0]^T$$
$$I_2 = [0\ 10\ 0\ 0\ 10\ 0\ 0\ 10\ 0]^T$$

- Say $M' = 1$ and
$$E_1 = [5\ 0\ 5\ 10\ 5\ 10\ 5\ 0\ 5]^T$$

- Compute Average Image, A
$$A = (I_1 + I_2)/2$$
$$= \left[\tfrac{0+0}{2}\ \ \tfrac{0+10}{2}\ \ \cdots\ \ \tfrac{0+0}{2}\right]$$
$$= [0\ 5\ 0\ 5\ 10\ 5\ 0\ 5\ 0]^T$$

- Project I_1 to 1-D "face space"
$$W_1 = [w_{11}]$$
where
$$w_{11} = E_1^T \cdot (I_1 - A)$$
$$I_1 - A = [0{-}0\ \ 0{-}5\ \ \cdots\ \ 0{-}0]^T$$
$$= [0\ {-5}\ 0\ 5\ 0\ 5\ 0\ {-5}\ 0]^T$$
$$\Rightarrow w_{11} = 5\cdot 0 + 0 \cdot {-5} + \cdots + 5 \cdot 0 = 0$$
$$\therefore W_1 = [0]$$

- Project I_2 to 1-D face space
$$W_2 = [-100]$$

- Determine if I_1 or I_2 is closest to test image $I_{test} = \begin{bmatrix} 0 & 7 & 3 \\ 0 & 10 & 10 \\ 0 & 10 & 0 \end{bmatrix}$

$\Rightarrow$ Project I_{test} to face space
$$W_{test} = [w_{t1}]$$
$$w_{t1} = E_1^T \cdot (I_{test} - A)$$
$$= [5\ 0\ 5\ 10\ 5\ 10\ 5\ 0\ 5] \begin{bmatrix} 0 \\ 2 \\ 3 \\ -5 \\ 0 \\ 5 \\ 0 \\ 5 \\ 0 \end{bmatrix}$$
$$= 15$$

$\Rightarrow [15]$ closer to $[0]$ than $[-100]$

Limits of PCA

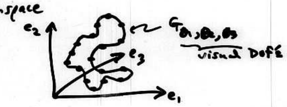
- Attempts to fit a *hyperplane* to the data
  - can be interpreted as fitting a Gaussian, where A is the covariance matrix
  - this is not a good model for some data
- If you know the model in advance, don't use PCA
  - regression techniques to fit parameters of a model
- Several alternatives/improvements to PCA have been developed
  - LLE: http://www.cs.toronto.edu/~roweis/lle/
  - isomap: http://isomap.stanford.edu/
  - kernel PCA: http://www.cs.ucsd.edu/classes/fa01/cse291/kernelPCA_article.pdf
  - For a survey of such methods applied to object recognition
    - Moghaddam, B., "Principal Manifolds and Probabilistic Subspaces for Visual Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, June 2002 (Vol 24, Issue 6, pps 780-788) http://www.merl.com/papers/TR2002-13/

Parametric Eigenspace

Key Idea:
For a given object, as we slowly vary the visual DOFs, the appearance also slowly changes. Furthermore, changes in subspace also slowly change.

Possible Exceptions: When crossing "visual events" where topological change in appearance occurs, or for specular objects

⇒ discrete pts $G_1, \dots, G_m$ in eigenspace are samples on a smooth manifold (surface) in eigenspace



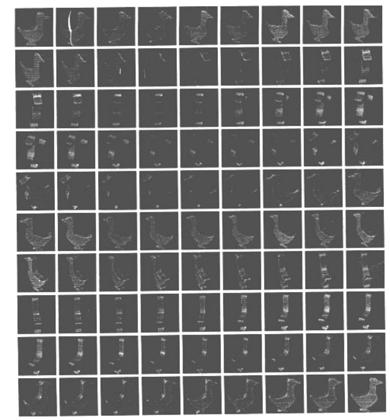


Figure 2: Image set obtained by rotating the object shown in Fig.1 about a single axis. These images are scale and brightness normalized.

Murase and Lindenbaum have compared the performance of the STA algorithm with the conjugate gradient and SVD algorithms described previously. Their results show the STA algorithm to be superior in performance to both algorithms, often 10 or more orders in magnitude faster than the SVD algorithm. Hence, we have used the STA algorithm to compute the eigenvectors of image sets. As an example, Fig.3 shows six eigenvectors (shown as images) computed for the image set shown in Fig.2. The eigenvectors are ordered in descending order of their eigenvalue magnitudes.

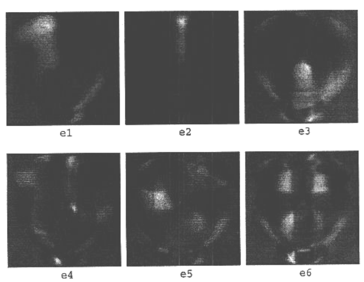


Figure 3: Eigenvectors corresponding to the six largest eigenvalues, computed for the image set shown in Fig.2.

of an input image onto a 10-dimensional space requires 10 dot products of the input image with the 10 orthogonal eigenvectors that constitute the eigenspace. Hence, the projection of an image onto the universal and object eigenspaces can be done in real-time (frame rate of a typical image digitizer) using simple and inexpensive hardware. Once the image has been projected onto the universal eigenspace, we need to find the hypersurface that is closest to it. When the image is projected onto the object eigenspace, we need to find the point of the object hypersurface that is closest to the input point. A variety of algorithms can be used to solve both these problems. In our current implementation, we use an exhaustive search algorithm that computes the distance of the input point from uniformly sampled points on the parametrized hypersurface.

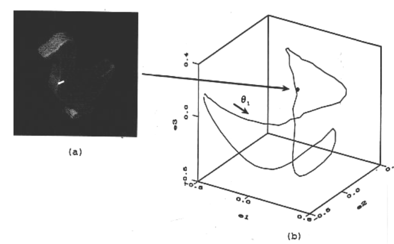


Figure 5: (a) An input image. (b) The input image is mapped to a point in the object eigenspace. The location of the point on the parametric curve determines the pose of the object in the input image.

Use P+1 Eigenspaces:

1. "Universal" Eigenspace
   - uses average image, C, of all images of all objects
   - use to discriminate between different objects ⇒ identify which object
   - ~ 10-dimensional
2. "Object" Eigenspaces
   - uses average image, $C^{(p)}$, of all images of object P
   - P different obj. eigenspaces
   - use to estimate pose of a given object
   - ~ 10-dimensional

- Given a set of visual DOFs, $\theta_1, \ldots, \theta_m$ discretely sample each θ_i and compute $G^{(p)}(\theta_1, \ldots, \theta_m)$ foreach object p.
  (use average, c, of all images)

  universal eigenspace of all objects
- Fit m-dimensional surface to pts in eigenspace (e.g. using cubic-spline interpolation)
- Also, foreach object project into "object eigenspace" ⇒ $F^{(p)}(\theta_1, \ldots, \theta_m)$
  Fit surface to $F^{(p)}$
  (use average, $c^{(p)}$, of all images of p.)

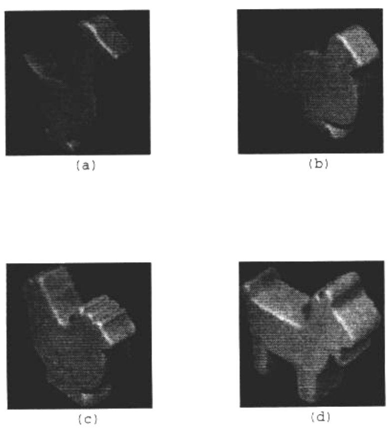


Figure 7: The four objects used in the experiments.

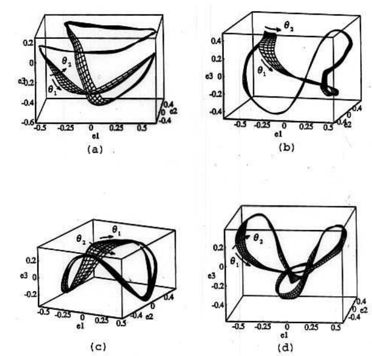


Figure 8: Parametric hypersurfaces in object eigenspace computed for the four objects shown in Fig. 7. For display, only the three most important dimensions of each eigenspace are shown. The hypersurfaces are reduced to surfaces in three-dimensional space.

Recognition Alg

1. Project test image into universal eigenspace

$$z = [e_1, \dots, e_k]^T (x_{test} - c)$$

2. Find closest object surface, p,

$$d_1^{(p)} = \min_{\theta_1, \theta_2} \| z - G^{(p)}(\theta_1, \theta_2) \|$$

3. <u>If</u> $d_1^{(p)} > T_1$ <u>then</u> unknown object so halt

4. Project image into selected object p's eigenspace, $z^{(p)}$

5. Find values of visual DOFs:

$$d_2^{(p)} = \min_{\theta_1, \theta_2} \| z^{(p)} - F^{(p)}(\theta_1, \theta_2) \|$$

$\Rightarrow$ Solve for θ_1, θ_2

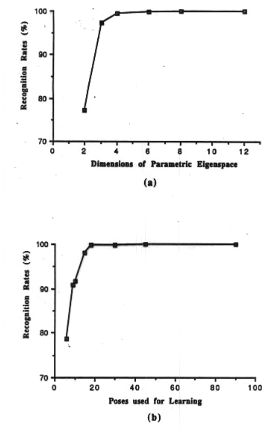


Figure 9: Recognition results for the objects shown in Fig. 7. (a) Recognition rate plotted as a function of the number of universal eigenspace dimensions used to represent the parametric hypersurfaces. (b) Recognition rate plotted as a function of the number of discrete poses of each object used in the learning stage. In both cases the recognition rates were computed using all 1080 input images detailed in Table 1.

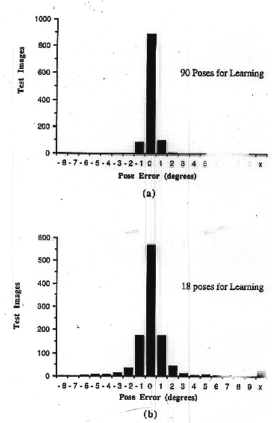


Figure 10: Pose estimation results for the objects shown in Fig. 7. (a) Histogram of the error (in degrees) in computed object pose for the case where 90 poses are used in the learning stage. (b) Pose error histogram for the case where 18 poses are used in the learning stage. The average of the absolute error in pose for the complete set of 1080 test images is 0.5 in the first case and 1.0 in the second case.