Lecture-12

Face Recognition

Simple Approach

- Recognize faces (mug shots) using gray levels (appearance)
- Each image is mapped to a long vector of gray levels
- Several views of each person are collected in the model-base during training
- During recognition a vector corresponding to an unknown face is compared with all vectors in the model-base
- The face from model-base, which is closest to the unknown face is declared as a recognized face.

Problems and Solution

- Problems :
  - Dimensionality of each face vector will be very large (250,000 for a 512X512 image!)
  - Raw gray levels are sensitive to noise, and lighting conditions.
- Solution:
  - Reduce dimensionality of face space by finding principal components (eigen vectors) to span the face space
  - Only a few most significant eigen vectors can be used to represent a face, thus reducing the dimensionality

Eigen Vectors and Eigen Values

The eigen vector, x, of a matrix *A* is a special vector, with the following property

$$Ax = lx$$ Where ë is called eigen value

To find eigen values of a matrix A first find the roots of:

$$\det(A - lI) = 0$$

Then solve the following linear system for each eigen value to find corresponding eigen vector

$$(A - lI)x = 0$$

Example

$$A = \begin{bmatrix} -1 & 2 & 0 \\ 0 & 3 & 4 \\ 0 & 0 & 7 \end{bmatrix}$$

Eigen Values

$$l_1 = 7,\ l_2 = 3,\ l_3 = -1$$

$$\mathbf{x_1} = \begin{bmatrix} 1 \\ 4 \\ 4 \end{bmatrix},\ \mathbf{x_2} = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix},\ \mathbf{x_3} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$ Eigen Vectors

Eigen Values

$$\det(A - lI) = 0$$

$$\det(\begin{bmatrix} -1 & 2 & 0 \\ 0 & 3 & 4 \\ 0 & 0 & 7 \end{bmatrix} - l \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}) = 0$$

$$\det(\begin{bmatrix} -1-l & 2 & 0 \\ 0 & 3-l & 4 \\ 0 & 0 & 7-l \end{bmatrix}) = 0$$

$$(-1-l)((3-l)(7-l)-0) = 0$$

$$(-1-l)(3-l)(7-l) = 0$$

$$l = -1, \quad l = 3, \quad l = 7$$

Eigen Vectors

$l = -1$ $\qquad$ $(A - lI)x = 0$

$$(\begin{bmatrix} -1 & 2 & 0 \\ 0 & 3 & 4 \\ 0 & 0 & 7 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}) \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 & 0 \\ 0 & 4 & 4 \\ 0 & 0 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$0 + 2x_2 + 0 = 0$$

$$0 + 4x_2 + 4x_3 = 0$$

$$0 + 0 + 8x_3 = 0$$

$$x_1 = 1, \quad x_2 = 0, \quad x_3 = 0$$

$$\mathbf{x_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Face Recognition

Collect all gray levels in a long vector *u:*

$$u = (I(1,1),\ldots,I(1,N),I(2,1),\ldots,I(2,N),\ldots,I(M,1),\ldots,I(M,N))^T$$

Collect *n* samples (views) of each of *p* persons in matrix A (MN X pn):

$$A = \left[u_1^1,\ldots u_n^1, u_1^2 \ldots, u_n^2, \ldots, u_1^p \ldots, u_n^p\right]$$

Form a correlation matrix L (MN X MN):

$$L = AA^T$$

Compute eigen vectors, $f_1, f_2, f_3, \ldots f n_1$, of L, which form a bases for whole face space

Face Recognition

Each face, *u*, can now be represented as a linear combination of eigen vectors

$$u = \sum_{i=1}^{n_1} a_i f_i$$

Eigen vectors for a symmetric matrix are orthonormal:

$$f_i^T . f_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Face Recognition

$$u_x^T . f_i = \left(\sum_{i=1}^{n} a_i f_i\right)^T . f_i$$

$$= (\mathrm{a}_1 f_1^T + \mathrm{a}_2 f_2^T + \ldots + \mathrm{a}_i f_i^T + \ldots + \mathrm{a}_n f_n^T).f_i$$

$$u_x^T . f_i = (\mathrm{a}_1 f_1^T . f_i + \mathrm{a}_2 f_2^T . f_i + \ldots + \mathrm{a}_i f_i^T . f_i + \ldots + \mathrm{a}_n f_n^T . f_i)$$

$$u_x^T . f_i = a_i$$

Therefore: $a_i = u_x^T . f_i$

Face Recognition

L is a large matrix, computing eigen vectors of a large matrix is time consuming. Therefore compute eigen vectors of a smaller matrix, C:

$$C = A^T A$$

Let a_i be eigen vectors of C, then Aa_i are the eigen vectors of A:

$$C\boldsymbol{a}_i = \boldsymbol{l}_i \boldsymbol{a}_i$$

$$A^T A\boldsymbol{a}_i = \boldsymbol{l}_i \boldsymbol{a}_i$$

$$AA^T (A\boldsymbol{a}_i) = \boldsymbol{l}_i (A\boldsymbol{a}_i)$$

$$L(A\boldsymbol{a}_i) = \boldsymbol{l}_i (A\boldsymbol{a}_i)$$

Training

- Create *A* matrix from training images
- Compute *C* matrix from *A*.
- Compute eigenvectors of *C*.
- Compute eigenvectors of *L* from eigenvectors of *C*.
- Select few most significant eigenvectors of *L* for face recognition.
- Compute coefficient vectors corresponding to each training image.
- For each person, coefficients will form a cluster, compute the mean of cluster.

Recognition

- Create a vector *u* for the image to be recognized.
- Compute coefficient vector for this *u*.
- Decide which person this image belongs to, based on the distance from the cluster mean for each person.

```
load faces.mat
C=A'*A;
[vectorC,valueC]=eig(C);
ss=diag(valueC);
[ss,iii]=sort(-ss);
vectorC=vectorC(:,iii);
vectorL=A*vectorC(:,1:5);
Coeff=A'*vectorL;
for I=1:30
        model(i, :)=mean(coeff((5*(i-1)+1):5*I,:));
end
while (1)
        imagename=input('Enter the filename of the image to
        Recognize(0 stop):');
        if (imagename <1)
        break;
        end;
        imageco=A(:,imagename)'*vectorL;
        disp ('');
        disp ('The coefficients for this image are:');
```

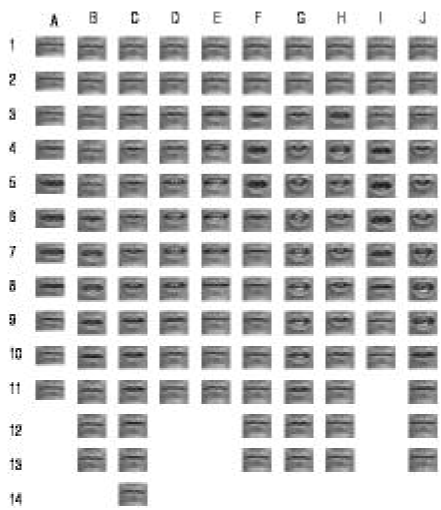
```
        mess1=sprintf('%.2f  %.2f  %.2f  %.2f  %.2f',
        imageco(1),imageco(2),imageco(3),imageco(4),
        imageco(5));
        disp(mess1);
        top=1;
        for I=2:30
                if (norm(model(i,:)-imageco,1)<norm(model
                (top, : )-imageco,1))
                top=i
                end
        end
        mess1=sprintf('The image input was a image of person
        number %d',top);
        disp(mess1);
        end
b=A(:,81);
b=reshape(b,34,51);
imshow(b,gray(255)):
```

Webpage

http://vismod.www.media.mit.edu/vismod/demos/

Visual Lipreading

Image Sequences of "A" to "J"



Particulars

- Problem: Pattern differ spatially
- Solution: Spatial registration using SSD
- Problem : Articulations vary in length, and thus, in number of frames.
- Solution: Dynamic programming for temporal warping of sequences.
- Problem: Features should have compact representation.
- Solution: Principle Component Analysis.

Feature Subspace Generation

- Generate a lower dimension subspace onto which image sequences are projected to produce a vector of coefficients.
- Components
  - Sample Matrix
  - Most Expressive Features

Generating the Sample Matrix

• Consider e letters, each of which has a training set of K sequences. Each sequence is compose of images:

$$I_1, I_2, \ldots, I_P$$

• Collect all gray-level pixels from all images in a sequence into a vector:

$$u = (I_1(1,1), \ldots, I_1(M,N), I_2(1,1), \ldots, I_2(M,N), \ldots I_P(1,1), \ldots, I_P(M,N))$$

. Generating the Sample Matrix

- For letter w , collect vectors into matrix T

$$T_w = \left[u^1, u^2, \ldots u^K\right]$$

- Create sample matrix A:

$$A = \left[T_1, T_2, \ldots T_e\right]$$

•The eigenvectors of a matrix $L = AA^T$ are defined as:

$$Lf_i = l_i f_i$$

The Most Expressive Features

- f is an orthonormal basis of the sample matrix.

•Any image sequence, u, can be represented as:

$$u = \sum_{n=1}^{Q} a_{nf_n} = fa$$

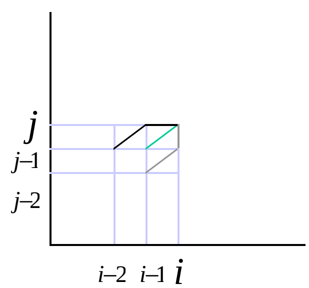
- Use Q most significant eigenvectors.
- The linear coefficients can be computed as:

$$a_n = u^T f_n$$

Training Process

- Model Generation
  - Warp all the training sequences to a fixed length.
  - Perform spatial registration (SSD).
  - Perform PCA.
  - Select Q most significant eigensequences, and compute coefficient vectors “a”.
  - Compute mean coefficient vector for each letter.

Warping



$$A = [a_1, a_2, \ldots, a_i, a_I]$$

$$B = [b_1, b_2, \ldots, b_j, b_J]$$

$$d_{ij} = |a_i - b_j|$$

$$g_{11} = 2d_{11}$$

$$g(i, j) = \min \begin{bmatrix} g(i-1, j-2) + 2d(i, j-1) + d(i, j) \\ g(i-1, j-1) + 2d(i, j) \\ g(i-2, j-1) + 2d(i-1, j) + d(i, j) \end{bmatrix}$$

Recognition

- Warp the unknown sequence.
- Perform spatial registration.
- Compute:

$$a_i^x = u_x^T . \boldsymbol{f}_i$$

$$d^w = \| a^w - a^x \|$$

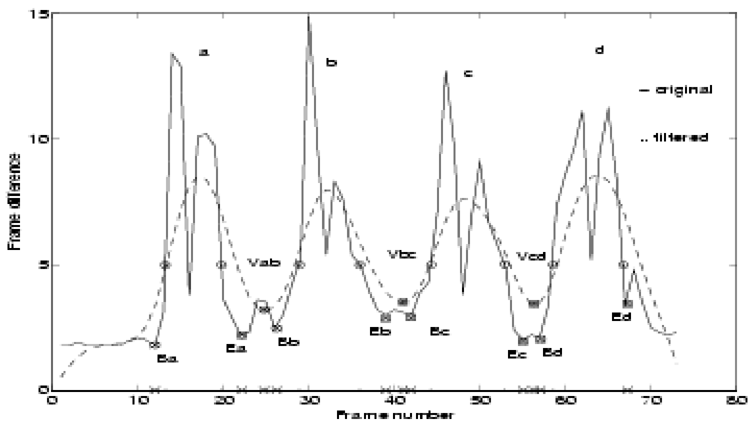
- Determine best match by $\min_w (d^w)$

Extracting letters from Connected Sequences

- Average absolute intensity difference function

$$f(n) = \frac{1}{MN} \sum_{x=1}^{M} \sum_{y=1}^{N} \| I_n(x,y) - I_{n-1}(x,y) \|$$

- f is smoothed to obtain g.
- Articulation intervals correspond to peaks and non-articulation intervals correspond to valleys in “g”.



Extracting letters from Connected Sequences

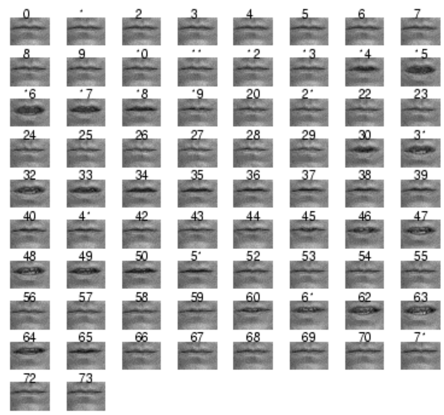
- Detect valleys in g.
- From valley locations in g, find locations where f crosses high threshold.
- Locate beginning and ending frames.

A 12-22

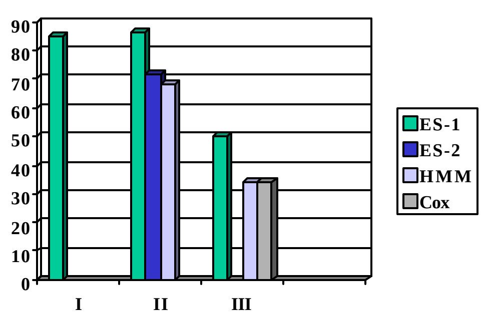
B 26-39

C 42-55

D 57-67

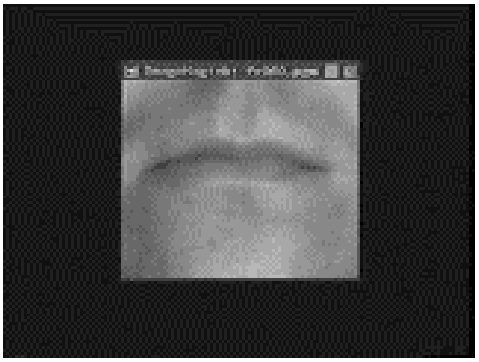


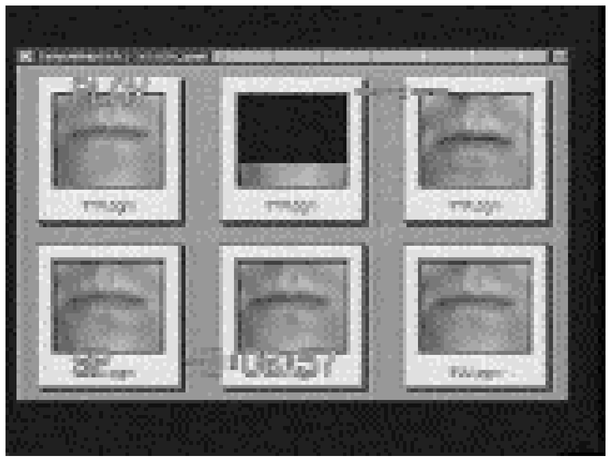
Results

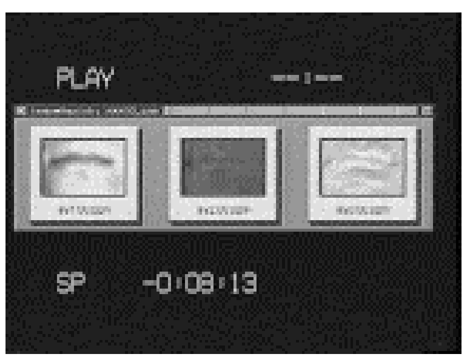


I: “A” to “J” one speaker, 10 training seqs
II. “A” to “M”, one speaker, 10 training seqs
III. “A” to “Z”, ten speakers, two training seqs/letter/person

Show Video Clip






PLAY
SP -0:08:13

paper

http://www.cs.ucf.edu/~vision/papers/shah/97/NDS97.pdf

Program-2 & 3

- For the program-2 you will implement "Synthesizing Realistic Facial Expressions from Photographs" method (Lecture-11).
  - You will assume one view of face is available, the aim is to estimate a pose of camera, translation, rotation, scaling, etc.
  - Do not estimate the changes in "p", vertices.
  - If you have a better face model, like Alias, use it, otherwise use Candide model from the class webpage.
  - Select 13 feature points manually
  - Synthesize a face image from a novel view, once the pose is correctly estimated.
  - Due Nov 7

Program-2 & 3

- For Program-3 implement “Motion Estimation Using Flexible Wireframe Model” (Lecture-9).
  - Use the output of Program-2, conformed wireframe model
  - Assume simple optical flow constraint equation, no need to use generalized optical flow constraint equation
  - Using estimated motion and changes in wireframe mode, sysnthesize image sequence, and compare it with the original sequence for video compression (MPEG-4).
  - Due Nov 30