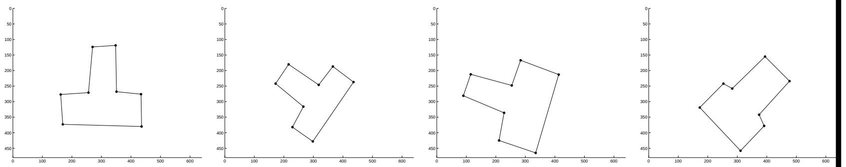
Modeling Classes of Shapes

Suppose you have a class of shapes with a range of variations:



How to represent whole class?

Statistical shape models

System 2 Overview

System processes

Previous Systems: Thresholding, Boundary Tracking, Corner Finding (but here with better threshold)

This System:

Orientation to standard position

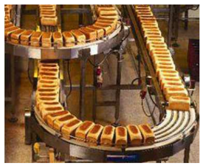
Training: Point Distribution Model calculation

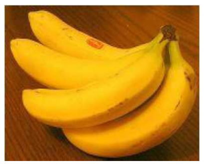
Recognition: likelihood calculation

Motivation

Not all objects

- are made equal: variations in production
- grow equally: eg. fruit classification
- appear equal: movement, configurations

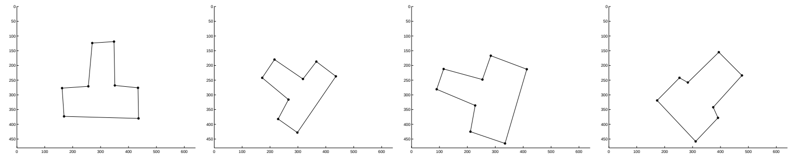


But if they belong to the same **class of object**, we want to recognize them as such

How to recognize shape classes?

All TEE parts belong to the same shape class, but have large variations in shape in common ways



Need to:

1. **Identify common modes (ie. directions) of variation** for each class
2. **Represent the shape class** as statistical variation over these modes
3. Use **statistical recognition** based on comparison to statistical shape representation

Lecture Set Overview

Principal Component Analysis

Point Distribution Models

Model Learning and Data Classification

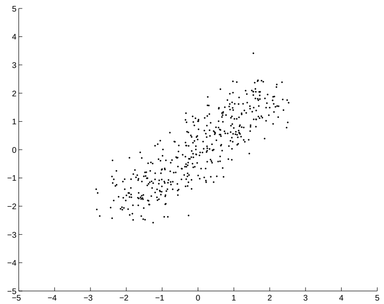
Rotating TEEs to Standard Position

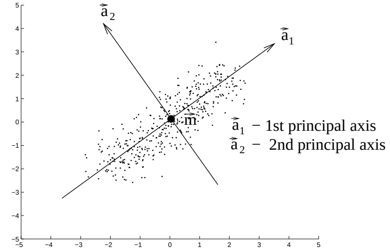
Representing TEEs using Point Distribution Models

Recognize new examples w/statistical classifier

Principal Component Analysis

Given a set of D dimension points $\{\vec{x}_i\}$ with mean $\vec{m}$





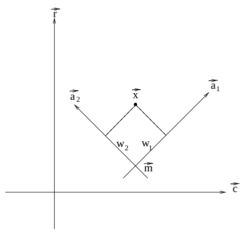
Find a new set of D perpendicular coordinate axes $\{\vec{a}_j\}$ such that

$$\vec{x}_i = \vec{m} + \sum_j w_{ij}\vec{a}_j$$

IE. point $\vec{x}_i$ represented as a mean plus weighted sum of axis directions

Transforming points to the new representation

Transforming points is easy as $\vec{a}_k \cdot \vec{a}_j = 0$ and $\vec{a}_k \cdot \vec{a}_k = 1$ for $k \neq j$

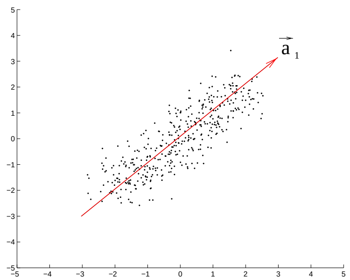


Computing w_{ik} is easy:

$$\vec{a}_k \cdot (\vec{x}_i - \vec{m}) = \vec{a}_k \cdot \sum_j w_{ij}\vec{a}_j = \sum_j w_{ij}\vec{a}_k \cdot \vec{a}_j = w_{ik}$$

How to do PCA I

1. Choose axis $\vec{a}_1$ as the direction of the most variation in the dataset:



2. Project each $\vec{x}_i$ onto a $D-1$ dimensional subspace perpendicular to $\vec{a}_1$ (ie removing the component of variation in direction $\vec{a}_1$) to give $\vec{x}'_i$
3. Calculate the axis $\vec{a}_2$ as the direction of the most remaining variation in $\{\vec{x}'_i\}$

4. Project each $\vec{x}_i'$ onto a $D-2$ dimension subspace
5. Continue like this until all D new axes $\vec{a}_i$ are found.

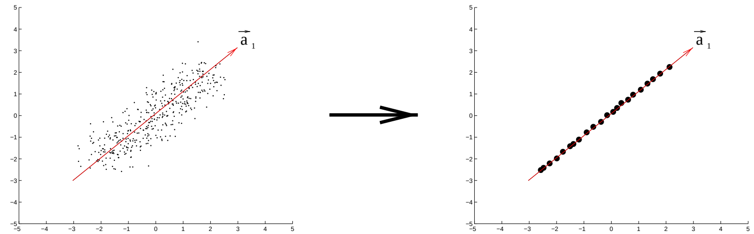
Why PCA

Many possible axis sets $\{\vec{a}_i\}$

PCA chooses axis directions $\vec{a}_i$ in order of largest remaining variation

Gives an ordering on dimensions from most to least significant

Allows us to omit low significance axes. Eg, projecting $\vec{a}_2$ gives:



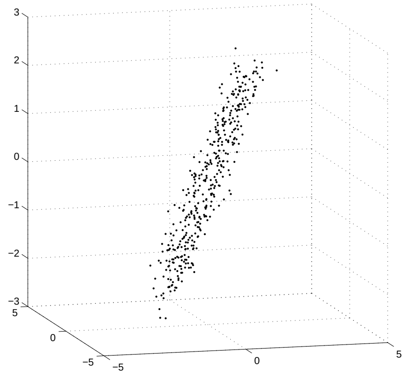
How to Do PCA II

Via Eigenanalysis
Given N D-dimensional points $\{\vec{x}_i\}$

1. Mean $\vec{m} = \frac{1}{N}\sum_i \vec{x}_i$
2. Compute scatter matrix $\mathrm{S} = \sum_i (\vec{x}_i - \vec{m})(\vec{x}_i - \vec{m})'$
3. Compute Singular Value Decomposition (SVD): S = U D V', where D is a diagonal matrix and U' U = V' V = I
4. PCA: i^{th} column of V is axis $\vec{a}_i$ (i^{th} eigenvector of S) d_{ii} of D is a measure of significance (i^{th} eigenvalue)

Midlecture Problem

If you had a 3D dataset like this



How many principal components does it have?

Point Distribution Models

Given:

Set of objects from the same class

Set of point positions $\{\vec{x}_i\}$ for each object instance

Assume:

Point positions have a systematic structural variation plus a Gaussian noise point distribution

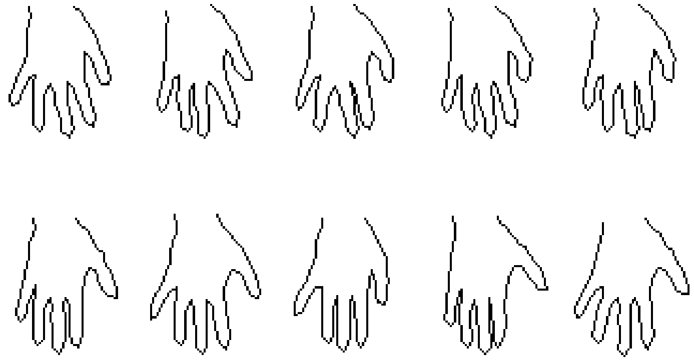
Thus, point position variations are correlated

Goals:

Construct a model that captures structural as well as statistical position variation.
Use model for recognition

Data Example

A family of objects with shape variations



How to represent?

Point Distribution Models - PDMs

Given a set of N observations, each with P boundary points $\{(r_{ik}, c_{ik})\}, k = 1..P, i = 1..N$ in corresponding positions.

Key Trick: rewrite $\{(r_{ik}, c_{ik})\}$ as a new $2P$ vector $\vec{x}_i = (r_{i1}, c_{i1}, r_{i2}, c_{i2}, ..., r_{ip}, c_{ip})'$

Gives N vectors $\{\vec{x}_i\}$ of dimension $2P$

PDMs II

If shape variations are random, then components of $\{\vec{x}_i\}$ will be uncorrelated.

If there is a systematic variation, then components will be correlated.

Use PCA to find correlated variations.

PDM II: The Structural Model

PCA over the set $\{\vec{x}_i\}$ gives a set of $2P$ axes such that

$$\vec{x}_i = \vec{m} + \sum_{j=1}^{2P} w_{ij}\vec{a}_j$$

$2P$ axes gives complete representation for $\{\vec{x}_i\}$.

Approximate shapes using a subset M of the most significant axes (based on the eigenvalue size from PCA):

$$\vec{x}_i \doteq \vec{m} + \sum_{j=1}^{M} w_{ij}\vec{a}_j \quad (1)$$

PDM II: The Structural Model

Represent $\vec{x}_i$ using $\vec{w}_i = (w_{i1}, ..., w_{iM})'$

A smaller representation as $M << 2P$

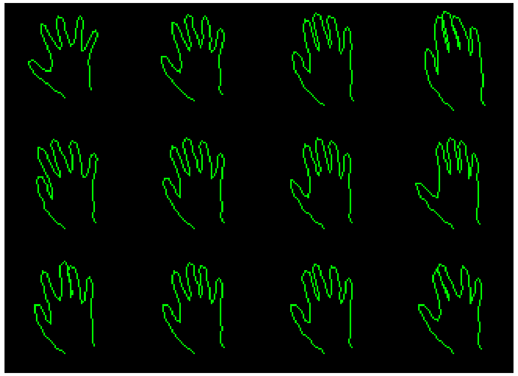
Goal: represent the essential structure variations

Approximate full shape reconstruction using $\vec{w}_i$ and (1)

Can vary $\vec{w}_i$ to vary shape

Structural model - varying the weights

Each row here varies one of top 3 eigenvectors of model from hand outlines



Visualisation of the main modes of structural variation

PDM III: The Statistical Model

If we have a good structural model, then the component weights should characterise the shape.

A family of shapes should have a distribution that characterises normal shapes (and abnormal shapes are outliers).
We assume that the distribution of normal shape weights is Gaussian.

Statistical Model:

Given a set of N component projection vectors $\{\vec{w}_i\}$

Mean vector is $\vec{t} = \frac{1}{N}\sum_i \vec{w}_i$

Covariance matrix $\mathrm{C} = \frac{1}{N-1}\sum_i (\vec{w}_i - \vec{t})(\vec{w}_i - \vec{t})'$

Statistical Model Matlab code

Uses inverse of C (`invcor`):

```
% Vecs(N,D) is N observations of D
%                    dimensional vector
Mean = mean(Vecs)';
diffs = Vecs - ones(N,1)*Mean';
Invcor = inv(diffs'*diffs/(N-1));
```


Classification/Recognition

Given:

- Unknown sample $\vec{x}$
- Structural model: mean $\vec{m}$ + M variation axes $\vec{a}_j$
- Statistical model: class means $\{\vec{t}_i\}$ and associated covariance matrices $\{\mathrm{C}_i\}$ for $i = 1..K$ classes

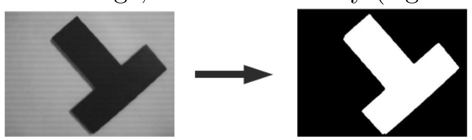
For each class i:

1. Project $\vec{x}$ onto $\vec{a}_j$ to get weights $\vec{w}$ (M dim vector)
2. Compute Mahalanobis distances:
   $d_i(\vec{w}) = ((\vec{w} - \vec{t}_i)'(\mathrm{C}_i)^{-1}(\vec{w} - \vec{t}_i))^{\frac{1}{2}}$

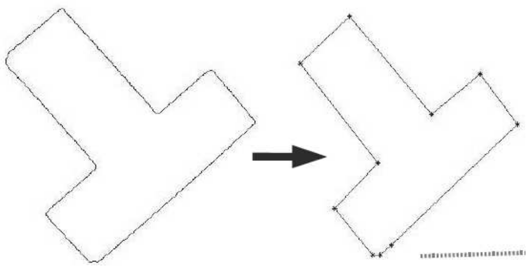
Select class i with smallest distance $d_i(\vec{w})$
Reject if smallest distance is too large

Algorithm Pre-processing

Load image, convert to binary (e.g. IVR)



Get boundary and find corners (system 1)



Problem: poor segmentation → varying numbers of corners

Algorithm Pre-processing

Exploit problem constraints: long lines, meet at given angles → delete short badly placed segments and extend to intersection

Exploit problem constraints: long lines → search for lines directly

Model Learning Summary

1. Load image, threshold, get boundary and find corners (c.f. TASK 1, except with a better corner finding threshold)
2. Point Distribution Model learning method:
   (a) Rotate TEEs to standard position
   (b) Get vertices into vector in a standard order
   (c) Construct structural model using PCA
   (d) Project examples into PCA weight space
   (e) Estimate statistical model of projections

Recognition Summary

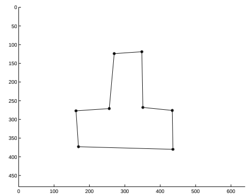
1. Load image, threshold, get boundary and find corners (c.f. TASK 1, except with a better corner finding threshold)
2. Point Distribution Model classification method:
   (a) Constraint: reject if not 8 vertices
   (b) Rotate TEE to standard position for PDM (Constraint: reject if not 2 sets of 4 nearly parallel lines)
   (c) Get vertices into vector in a standard order
   (d) Project vector into PCA weight space (structural model)
   (e) Evaluate statistical likelihood of projection (statistical model)

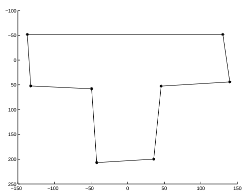
Rotating TEE to standard position

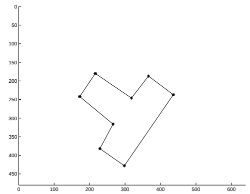
Assumes 8 lines in a rough TEE shape
Use heuristic algorithm (about 160 lines of code)

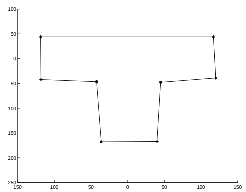
1. Sort 8 lines into 2 sets of 4 mutually nearly parallel lines (reject if not possible): find direction of one line, sort all others by whether angle with this line is $\leq \frac{\pi}{4}$ or not
2. Find which set is the head of TEE (reject if neither or both satisfy criteria). Also sort into positional order: if longest is sufficiently longer than the next and the 3 shortest are about the same length as the longest, the longest is the head of TEE
3. Estimate transformation of TEE to standard position with TEE head top parallel to column axis and center of TEE at origin. Apply transformation to TEE.

Example segmented boundary and standard alignment

1) 

aligned) 

2) 

aligned) 

Training Code

```
function train

  % training phase
  datacount = 0;
  maximages = 31;
  imagestem=input('Training image file stem\n?','s');
  for imagenum = 1 : maximages
     currentimagergb = imread([imagestem, ...
                int2str(imagenum),'.jpg'],'jpg');
     currentimage = rgb2gray(currentimagergb);

     % get corners using TASK2's method
     Image = getbinary(currentimage,0,0,0,0,0); %threshold
     [H,W] = size(Image);
     [r,c] = find( bwperim(Image,4) == 1 );     %perimeter
```



```
     [sr,sc] = removespurs(r,c,H,W,0);           %clean
     [tr,tc] = boundarytrack(sr,sc,H,W,0);       %track
     datalines = zeros(100,4);          % space for results
     numlines = 0;
     findcorners(tr,tc,H,W,9,16);                %segment

     % process boundary, assuming it is a TEE
     if numlines == 8
       % rotate datalines to standard position
       [newlines,flag] = standard_position( ...
                datalines(1:numlines,:),numlines,11);

       % get vertices
       if flag
         % sort vertices into a standard order
         sortvertices=sortvert(newlines(1:numlines,1:2));
```



```
         % add to scatter matrix
         [Vnum,Vcoord] = size(sortvertices);
         if Vnum == 8
           % turn points into long array
           datacount = datacount + 1;
           allvertices(datacount,:) = ...
                reshape(sortvertices,1,Vnum*Vcoord);
         end
       end
     end
  end

  % Create model
  meanvertex = mean(allvertices);
  vertexdev = allvertices - ones(datacount,1)*meanvertex;
  scatter = vertexdev'*vertexdev;
```



```
  [U,D,V]= svd(scatter);
  modeldev = V(:,1:5)'   % use only first 5 components

  % get projections onto data
  vecs = vertexdev*modeldev';

  % get class mean vector and covariance matrix
  [Mean,Invcor] = buildmodel(vecs,maximages);

  % save training data
  save modelfile modeldev meanvertex Mean Invcor
```

Mahalanobis Distance

Vector distance measure that takes account of different range of values for different positions in vector

Given vectors $\vec{a}$, $\vec{b}$ from a set with covariance $\mathbf{C}$, the Euclidean Distance between the vectors is:

$$\| \vec{a} - \vec{b} \| = [(\vec{a} - \vec{b})'(\vec{a} - \vec{b})]^{\frac{1}{2}}$$

The Mahalanobis Distance is:

$$[(\vec{a} - \vec{b})'\mathrm{C}^{-1}(\vec{a} - \vec{b})]^{\frac{1}{2}}$$

IE, scaled differences

Recognition Code

Same as Training code up to where 8×2 sorted point list reshaped into 16-vector

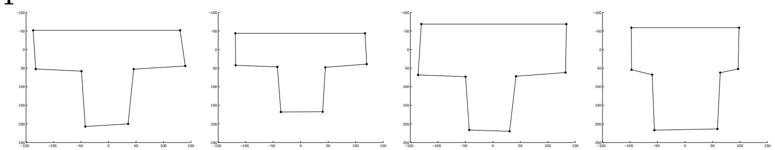
```
% turn 8 2D points into 16 vector
tmp = reshape(sortvertices,1,2*Vnum);

% project onto eigenvectors
vec = (tmp - meanvertex)*modeldev';

% get class distance
dist = mahalanobis(vec',Mean,Invcor);
```


Representing the TEEs using PDMs

Have 8 2D points for each TEE in standard position



Have $N = 31$ instances with variations

Can we make a model of the TEEs? **YES!**

Midlecture Problem

What might the first few principal components encode for the TEE data?

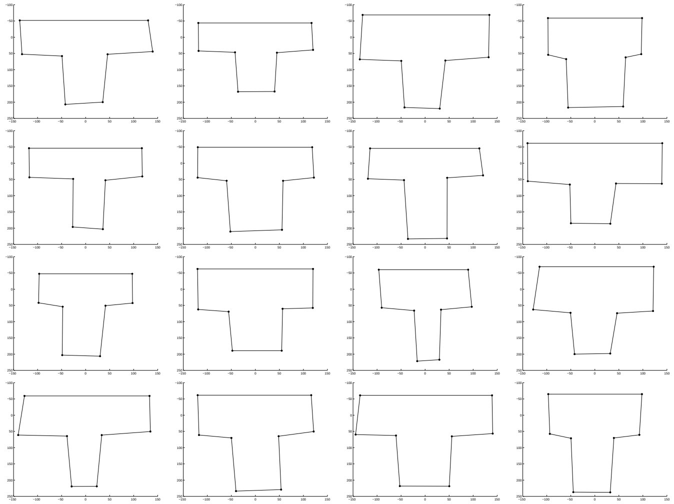
Representing the TEEs using PDMs II

Each corner point in the TEE model has a:

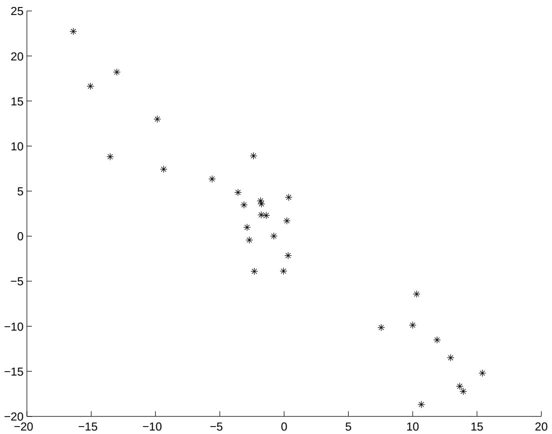
- Standard position
- Modified by shape variations

Use a Point Distribution Model (mean + PCA based main variation vectors) to represent structural variations and statistical model (mean + covariance matrix) to represent in-class variation

Some of Training Data

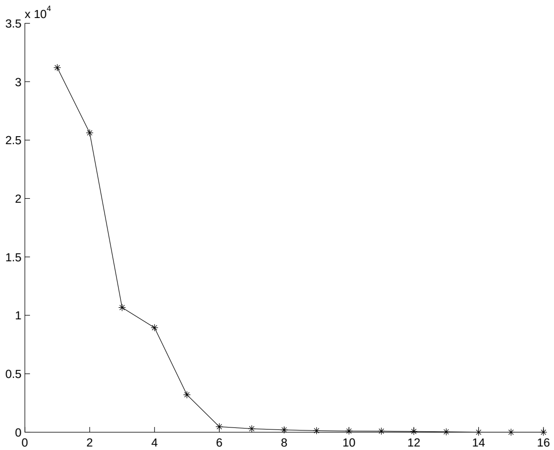


Correlation of x_1 and x_2



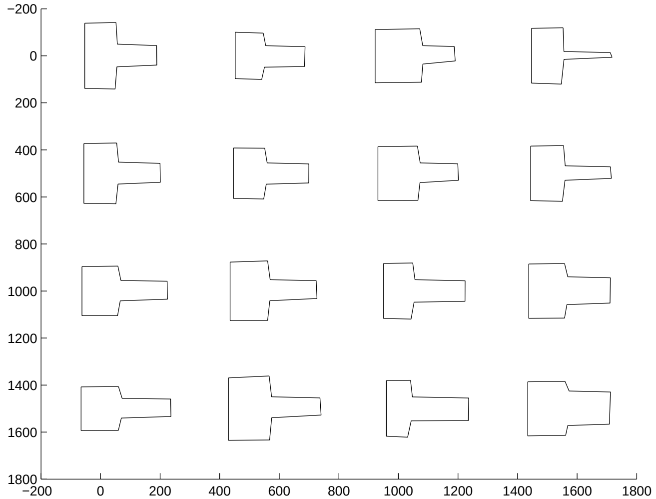
Note strong correlation

Eigenvalues of Scatter Matrix for TEE corners



Use first five

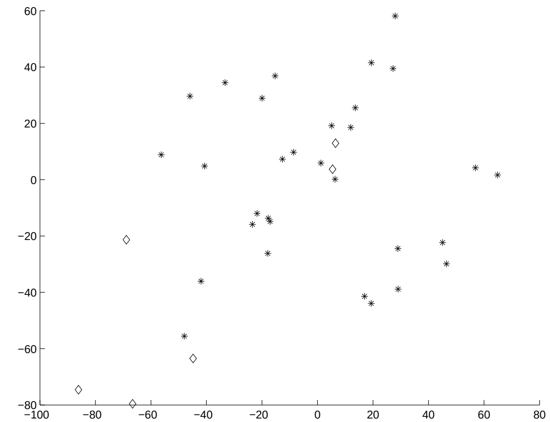
First Four Modes of Variation



Column goes from -2 to +2 standard deviations

Correlation of c_1 and c_2 (of c_1 ... c_5)

* - good data, diamond = bad data

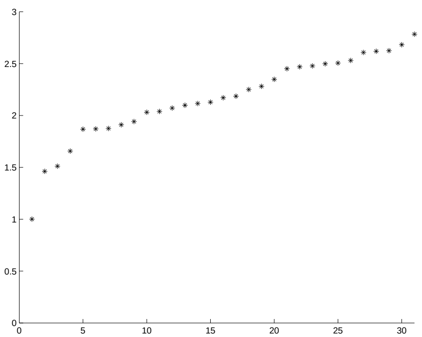


Note 1) decorrelated, 2) bad data tends to be further from mean

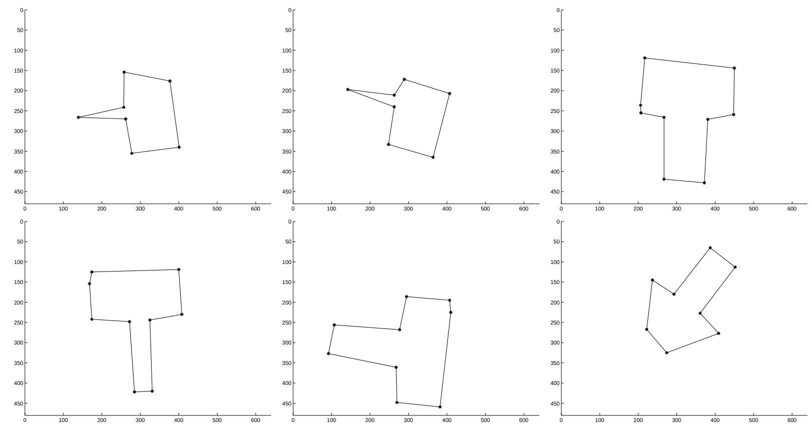
Good TEE Results

All TEEs recognized

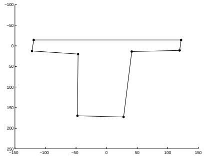
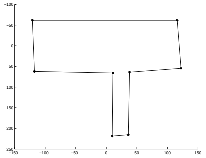
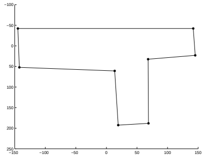
Plot of Mahalanobis distances for training data
Distributed Chi-squared mean: 5/2, std. dev.: sqrt(2*5)
'3-sigma' test threshold: $3\sqrt{Dim} = 3\sqrt{5} = 6.7$

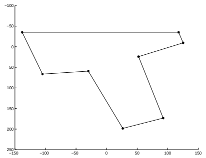
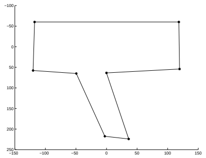
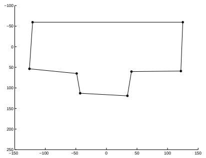


Bad TEE Shapes: Corner (5) & Rotation (1) Failures



‘Invalid’ TEE Shapes Aligned and Classified

5.8: 
5.8: 
8.5: 

9.3: 
2.7: 
5.5: 

Values are Mahalanobis distances

Problems

- Getting same number of points (3 failures on “good” data)
- Getting points in the same positions on smooth curves
- Alignment when shape too extreme

Discussion

- Applicable to smooth curves
- Applicable to other data beside points, 3D, ultrasound
- Covariance matrix links correlations between pairs of features. What about triples, ...?
- Can extend to include greylevel variations
- Training samples need not be in order (batch mode)

What We Have Learned

- Point Distribution Model (PDM) method for representing classes of objects
- Principal Component Analysis (PCA) decomposition for estimating the dominant modes of variation

PCA-based Face Recognition

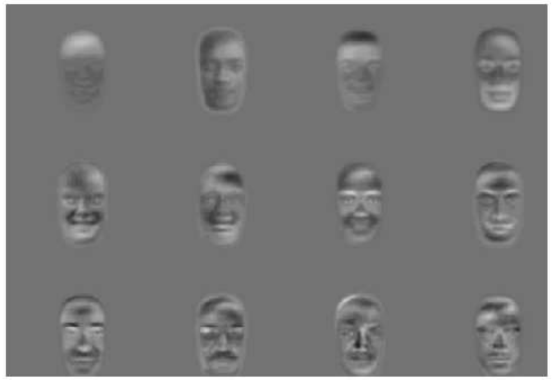
- Eigenfaces (Turk & Pentland 1991)
  Representation of faces using PCA directly on image intensities
  One of most famous uses of PCA in computer vision
  Seminal reference for face recognition (but would work better if we modeled shape variation rather than lightness variation)

- **Key principle**:
  Turn image array into long vector
  Represent sample image (face) as weighted sum of eigenimages (eigenfaces)

Eigenfaces

1. Given set of K registered face images ($R \times C$) with varying capture conditions
2. Represent as $R \times C$ long vectors
3. Do PCA (special trick for large matrices)

Mean face and subset of principle component axes/images [Morris '04]



4. Represent person i by projection weights $\vec{w}_i$

Eigenface Recognition

Given unknown face image F_u

1. Subtract mean face and project onto eigenfaces $\rightarrow \vec{w}_u$
2. Given database of projections $\{\vec{w}_i\}_{i=1}^{K}$, find class c with smallest Mahalanobis distance d_c to $\vec{w}_u$
3. If d_c small enough, return c as identity

Eigenface Results

2500 128×128 image database, varied lighting

- 96% successful recognition over lighting variations
- 85% over orientation variations
- 64% over size variations

Eigenface Discussion

- Variations in position, orientation, scale & occlusion cause problems
- Research topics
- 4-36% failure rate a problem at busy airports