Recognizing Instances of Deforming Models

Robert B. Fisher
School of Informatics
University of Edinburgh

PDM Based 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

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) using Mahalanobis distance

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 and correlation between dimensions

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 and de-correlated 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);
```

What We Have Learned

1. Use statistical classifier once deformation projected into space of expected values