Face Detection

Vikas.Chandrakant.Raykar
Prithu Sinha
University of Maryland, College Park
Goal

- To detect and localize human faces in any given grayscale/color image.

Challenges: Invariant to

- different illumination conditions
- pose
- camera orientation
Applications

- Face Detection is the first crucial step in face recognition, face tracking, pose estimation and expression recognition.
- Surveillance
- Video indexing/summarization, especially for new broadcasts and videos.
Important Survey papers


Two Approaches:

- Feature based
  No training required
- Image based
  Formulate as a two class problem face vs nonface. Difficulty in getting all nonfaces
Database

- CBCL Face Database #1
  (MIT Center For Biological and Computation Learning)
- Training set : 2,429 faces, 4,548 non-faces
- Test set : 472 faces, 23,573 non-faces
- 19X19 grayscale images as pgm files
- Test set both frontal as well as non frontal and rotated faces.

http://www.ai.mit.edu/projects/cbc
Training face database
Training nonface database
Test Face database
Test nonface database
Evaluation criterion

- Pd Detection probability
- Pf False alarm (a nonface detected as a face)
- Pe Error probability
- Pe = 0.5 * (Pf + (1 - Pd))
- Ideally require high Pd and low Pf
- Compromise between Pd and Pf
Our Approach

- Preprocessing
  - Feature selection (entire image, PCA, KPCA)
  - Training (NN)
  - Classification (NN, KNN)
- Discriminant Analysis (LDA, KLDA, BDA, KBDA)
- Adaboost
- Color based approaches
Preprocessing

- Database has already cropped images
- Lighting Compensation
  Subtract the best linear approximation of the image
- Histogram equalization to improve contrast
Example
Neural Network

- Vectorize the image and use the 361 element vector as an input to the NN
- 3 layer network (h hidden units)
- One output unit(1 face/ -1 nonface)
- sigmoid activation function
- Training : Gradient descent with momentum
Results

- Trained using 2,429 faces 4,548 non-faces
- Tested on 472 faces and 472 nonfaces
- H=25 hidden units
- 500 epochs(rate=0.1 momentum=0.8)
ROC-varying the threshold

25 HIDDEN UNITS
TRAINED FOR 500 EPOCHS

Detection probability

False alarm

threshold = 0

threshold = 0.9

threshold = 0.9
Comments

- The network is biased towards nonfaces since the number of nonfaces is more.
- We can get better detection probability is the number of faces is more than the number of nonfaces used in training. However the false alarm increases.
- We want high detection probability as we can reduce false alarms by certain heuristics.
TRAINED USING 1000 faces 1000 nonfaces
25 HIDDEN UNITS
TRAINED FOR 500 EPOCHS
Effect of num of hidden layers
Principal Component Analysis (PCA)

- Subtract the mean
- Compute the scatter matrix
- Find the eigen values and their corresponding eigen vectors
- Select c largest to capture the desired variance
- Use the projections on the eigenvectors
- Did PCA only on the faces
Eigen Faces
Percentage of variance captured
First three principal components
Classification based on PC’s

- K Nearest Neighbour (KNN)
  - 100 components to capture 95% of the variance
  - Classification based on one nearest neighbour
- Train a Neural network
## Results

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Pd</th>
<th>Pf</th>
<th>Pe</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA KNN</td>
<td>0.6624</td>
<td>0.0828</td>
<td>0.2102</td>
</tr>
<tr>
<td>PCA NN</td>
<td>0.6897</td>
<td>0.0938</td>
<td>0.2025</td>
</tr>
</tbody>
</table>

Threshold = 0
1000 faces
1000 nonfaces
PCA KNN vs PCA NN

- PCA KNN Need to store all the training samples and compare with the test image. Can also use the mean face and nonface.
- However PCA NN gives better results than PCA KNN.
Kernel PCA

- PCA is linear
- Uses only second order statistics
- Can do PCA in feature space
- Express dot product in feature space in terms of kernel functions in the input space
Kernels used

- Gaussian
- Polynomial
Percentage of Variance captured (poly 2 kernel)
First three KPC’s
### Results

<table>
<thead>
<tr>
<th></th>
<th>Pd</th>
<th>Pf</th>
<th>Pe</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPCA KNN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KPCA NN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## PCA vs KPCA

<table>
<thead>
<tr>
<th></th>
<th>Pd</th>
<th>Pf</th>
<th>Pe</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA-KNN</td>
<td>0.6624</td>
<td>0.0828</td>
<td>0.2102</td>
</tr>
<tr>
<td>PCA-NN</td>
<td>0.6897</td>
<td>0.0938</td>
<td>0.2025</td>
</tr>
<tr>
<td>KPCA-KNN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KPCA-NN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Comments

- KPCA gives lower performance than PCA(???)
- KPCA is computationally more intensive
PCA is unsupervised..so features found by PCA need not be discriminating among the classes

LDA finds the direction which maximizes the distance between the projected means and minimizes the within class scatter
LDA

- Equations
Projections-LDA
KLDA

- LDA in feature space
- Use kernels to compute the dot products
Projections
<table>
<thead>
<tr>
<th></th>
<th>Pd</th>
<th>Pf</th>
<th>Pe</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>0.3715</td>
<td>0.1274</td>
<td>0.3779</td>
</tr>
<tr>
<td>KLDA-GAUS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KLDA-POLY</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Biased Discriminant Analysis BDA

- Push all the nonfaces as far away from the face
- Minimize the within class scatter for face only
Projections
KBDA

- BDA if feature space
- Used Gaussian and second degree polynomial kernels
<table>
<thead>
<tr>
<th></th>
<th>Pf</th>
<th>Pd</th>
<th>Pe</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDA</td>
<td>0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KBDA poly2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KBDA gauss</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LDA/BDA/KLDA/KBDA
Adaboost
Comparison of all methods

- Used 1000 faces and 1000 nonfaces for training
- 472 faces and 472 nonfaces for testing
- For NN based methods threshold was set to zero
Error Probability

- PCA NN
- PCA KNN
- KLDA GAUS
- NN (hl2=25)
- KLDA POLY2
- KBDA GAUS
- KPCA KNN POLY2
- KBDA POLY2
- KPCA NN POLY2
- BDA
- LDA
Conclusions
Detecting faces in the entire image

- A 19 x 19 window is slid over the entire image and the windowed image data is sent as a vector to the detector.

- To detect faces of different sizes the scanning is repeated for successively smaller scales of the image by downsampling (typically by 1.2 to 1.4).
Output of the detector
Reducing false alarm

- Multiple detections in areas of the image where there is a face, and false detections only appear in a single position.
- Can be used to significantly reduce false alarm
- Assuming faces do not overlap multiple detections in the image can be assumed to be a measure of high confidence that the detected area to be a face.
Consensus-voting scheme

- For each scale of the image, retain information about the number of times each pixel overlapped due to multiple detections.
- Process the image for all scales possible, we added all the vote matrices to give the final vote matrix.
- Threshold the final vote matrix.
After Consensus voting
After Thresholding
Find Connected components and their bounding box
Sample results
Why not use color info....?

- Detect skin regions.
- In an 8x8 block if the number of skin pixels is less than 32 eliminate the skin region.
- Find all the connected components and label them as face.
Color based face detection

Actual Image

Skin Regions

Processed Skin Regions
How to detect skin regions?

- RGB to Y Cb Cr
- Get the joint distribution of Cb and Cr for skin
- Under normal illuminations skin color occupy small regions of the color space
- $77 < C_b < 127 \quad 133 < C_r < 173$
Sample Results
Eliminate false detections

- Use aspect ratio
- Search for face like features
- Use other detectors to validate face or not
- Use different color distribution models for different illumination conditions (indoor/outdoor)
- Ideal for real time applications where we have one camera and the color distribution model for that camera can be found
Sample result
THANK YOU ALL