Lecture 9
Recognition

Dr Christopher Brunner
AMME4710 – Computer Vision and Image Processing
Semester 2, 2013

Resources

• Readings
  – Szeliski, Computer Vision, Ch. 14
    http://szeliski.org/Book/
• Additional material (optional)
  – Viola and Jones, Robust Real-time Object
    Detection, IJCV 2001
• Slides credits
  – S. Lazebnik, K. Grauman, T. Darrell, S. Seitz, L. Fei-
    Fei, R. Fergus, A. Torralba
Outline

• Introduction
• Machine Learning (Classifiers)
• Face Detection
FACE DETECTION
Face detection
Face detection and recognition

Detection

Recognition

“Sally”
Consumer application: Apple iPhoto

http://www.apple.com/ilife/iphoto/
Funny Nikon ads

"The Nikon S60 detects up to 12 faces."
Faces everywhere

http://www.marcofolio.net/imagedump/faces_everywhere_15_images_8_illusions.html
Fig. 1. Unlike current machine-based systems, human observers are able to handle significant degradations in face images. For instance, subjects are able to recognize more than half of all familiar faces shown to them at the resolution depicted here. Individuals shown in order are: Michael Jordan, Woody Allen, Goldie Hawn, Bill Clinton, Tom Hanks, Saddam Hussein, Elvis Presley, Jay Leno, Dustin Hoffman, Prince Charles, Cher, and Richard Nixon.
Why Is Face Detection Difficult?

- **Pose**: Variation due to the relative camera-face pose (frontal, 45 degree, profile, upside down), and some facial features such as an eye or the nose may become partially or wholly occluded.
- **Presence or absence of structural components**: Facial features such as beards, mustaches, and glasses may or may not be present, and there is a great deal of variability amongst these components including shape, color, and size.
- **Facial expression**: The appearance of faces are directly affected by a person's facial expression.
- **Occlusion**: Faces may be partially occluded by other objects. In an image with a group of people, some faces may partially occlude other faces.
- **Image orientation**: Face images directly vary for different rotations about the camera's optical axis.
- **Imaging conditions**: When the image is formed, factors such as lighting (spectra, source distribution and intensity) and camera characteristics (sensor response, lenses) affect the appearance of a face.
Subspace Methods

- **PCA**: Principle Component Analysis.
  - “Eigenfaces”, (Turk and Pentland)
  - Bayesian, (Moghaddam and Pentland)

- **LDA/FLD**: 
  - Linear Discriminant Analysis
  - Fisher Linear Discriminator.
  - “Fisherfaces”, (Belhumeur & Kriegman)

- **ICA**: Independent Component Analysis.
Linear subspaces

Classification can be expensive

- Must either search (e.g., nearest neighbors) or store large PDF's

Suppose the data points are arranged as above... then,

- fit a line,
- classifier measures distance to line.

convert $x$ into $v_1, v_2$ coordinates

$$x \rightarrow ((x - \bar{x}) \cdot v_1, (x - \bar{x}) \cdot v_2)$$

What does the $v_2$ coordinate measure?

- distance to line
- use it for classification—near 0 for orange pts

What does the $v_1$ coordinate measure?

- position along line
- use it to specify which orange point it is

$_{\bar{x}}$ is the mean of the orange points
How to find $v_1$ and $v_2$?
- Matrix decomposition

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

$x$ is the mean of the orange points

$\overline{x}$ is the mean of the orange points

How to find $v_1$ and $v_2$?
- Matrix decomposition
The space of faces

An image is a point in a high dimensional space

- An $N \times M$ image is a point in $\mathbb{R}^{NM}$
- We can define vectors in this space as we did in the 2D case
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 $v_1, v_2, ..., v_K$
  - any face $x \approx \tilde{x} + a_1 v_1 + a_2 v_2 + \ldots + a_K v_K$
Principal component analysis

Suppose each data point is N-dimensional

- Same procedure applies:

\[
\text{var}(v) = \sum_x \| (x - \bar{x})^T \cdot v \|
\]

\[
= v^T A v \quad \text{where} \quad A = \sum_x (x - \bar{x})(x - \bar{x})^T
\]

- The eigenvectors of \( A \) define a new coordinate system
  - eigenvector with largest eigenvalue captures the most variation among data 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”
  - these eigenvectors are known as the \( \textit{principal components} \)
Eigenfaces

PCA extracts the eigenvectors of $A$

- Gives a set of vectors $v_1$, $v_2$, $v_3$, ...
- Each one of these vectors is a direction in face space
  - what do these look like?
The eigenfaces $v_1, ..., v_K$ span the space of faces

- A face is converted to eigenface coordinates by

\[
x \to (\underbrace{(x - \bar{x}) \cdot v_1}, \underbrace{(x - \bar{x}) \cdot v_2}, ..., \underbrace{(x - \bar{x}) \cdot v_K})
\]

\[
a_1 \quad a_2 \quad \ldots \quad a_K
\]

\[
x \approx \bar{x} + a_1v_1 + a_2v_2 + \ldots + a_Kv_K
\]
Computing eigenfaces by SVD

\[ X = \begin{bmatrix} \ldots \end{bmatrix} \]

\[ \text{num. pixels} \]

\[ \text{num. face images} \]

\[ \text{svd}(X,0) \text{ gives } X = U S V^T \]

\[ \text{Covariance matrix } XX^T = U S V^T V S U^T = U S^2 U^T \]

U’s are the eigenvectors of the covariance matrix X

(SVD: Singular value decomposition)
Computing eigenfaces by SVD

\[ X = \underbrace{\ldots}_{\text{num. pixels}} - \frac{1}{\text{num. face images}} = \]

svd(X,0) gives \[ X = U S V^T \]

Covariance matrix \[ XX^T = U S V^T V S U^T = U S^2 U^T \]

U’s are the eigenvectors of the covariance matrix X

Some new face image, \( x \)

\[ x = \underbrace{\ldots}_{\text{eigenfaces}} \ast \underbrace{\ldots}_{S \ast v} + \text{mean face} \]
Summary: Recognition with eigenfaces

Process labeled training images:
• Find mean $\mu$ and covariance matrix $\Sigma$
• Find $k$ principal components (eigenvectors of $\Sigma$) $u_1, \ldots, u_k$
• Project each training image $x_i$ onto subspace spanned by principal components:
  $$(w_{i1}, \ldots, w_{ik}) = (u_1^T(x_i - \mu), \ldots, u_k^T(x_i - \mu))$$

Given novel image $x$:
• Project onto subspace:
  $$(w_1, \ldots, w_k) = (u_1^T(x - \mu), \ldots, u_k^T(x - \mu))$$
• Optional: check reconstruction error $x - \hat{x}$ to determine whether image is really a face
• Classify as closest training face in $k$-dimensional subspace
Choosing the dimension $K$

How many eigenfaces to use?
Look at the decay of the eigenvalues

- the eigenvalue tells you the amount of variance “in the direction” of that eigenface
- ignore eigenfaces with low variance
Limitations

• Global appearance method: not robust to misalignment, background variation

The average of two faces is not another face
The spaces of faces is not convex
Limitations

- PCA assumes that the data has a Gaussian distribution (mean $\mu$, covariance matrix $\Sigma$)

The shape of this dataset is not well described by its principal components
Issues: metrics

What’s the best way to compare images?
- need to define appropriate features
- depends on goal of recognition task

**exact matching**
complex features work well
(SIFT, MOPS, etc.)

**classification/detection**
simple features work well
(Viola/Jones, etc.)
The Viola&Jones Face Detector

• A seminal approach to real-time object detection
• Training is slow, but detection is very fast
• Key ideas
  • Integral images for fast feature evaluation
  • Boosting for feature selection
  • Attentional cascade for fast rejection of non-face windows

Feature extraction

“Rectangular” filters

Feature output is difference between adjacent regions

Efficiently computable with integral image: any sum can be computed in constant time

Avoid scaling images → scale features directly for same cost

Viola & Jones, CVPR 2001

K. Grauman, B. Leibe
Example

Source

Result
Example
Large library of filters

Considering all possible filter parameters: position, scale, and type

Viola & Jones, CVPR 2001

K. Grauman, B. Leibe
Feature selection

• For a 24x24 detection region, the number of possible rectangle features is 180,000+!
• At test time, it is impractical to evaluate the entire feature set
• Can we create a good classifier using just a small subset of all possible features?
• How to select such a subset?

Use Boosting to both select the informative features and to form the classifier
Boosting

- Boosting is a classification scheme that combines *weak learners* into a more accurate *ensemble classifier*

- Training algorithm
  - Initially, weight each training example equally
  - In each boosting round:
    - Find the weak learner that achieves the lowest *weighted* training error
    - Raise the weights of training examples misclassified by current weak learner
  - Compute final classifier as linear combination of all weak learners (weight of each learner is directly proportional to its accuracy)
    - Exact formulas for re-weighting and combining weak learners depend on the particular boosting scheme (e.g., AdaBoost)

Weak Learners: Decision Stumps for classification and feature selection

- Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of weighted error.

Resulting weak classifier:

\[ h_t(x) = \begin{cases} 
  +1 & \text{if } f_t(x) > \theta_t \\
  -1 & \text{otherwise} 
\end{cases} \]

For next round, reweight the examples according to errors, choose another filter/threshold combo.

Viola & Jones, CVPR 2001

K. Grauman, B. Leibe
Each data point has a class label:

\[ y_t = \begin{cases} 
+1 & (\bullet) \\
-1 & (\circ) 
\end{cases} \]

and a weight:

\[ w_t = 1 \]
Toy example

Weak learners from a linear family

Each data point has a class label:

\[ y_t = \begin{cases} 
+1 \ (\bigcirc) \\
-1 \ (\bigotimes) 
\end{cases} \]

and a weight:

\[ w_t = 1 \]

\[ h \rightarrow p(\text{error}) = 0.5 \text{ it is at chance} \]
Toy example

Each data point has a class label:

\[ y_t = \begin{cases} 
+1 & (\bullet) \\
-1 & (\bigcirc) 
\end{cases} \]

and a weight:

\[ w_t = 1 \]

This is a ‘\textit{weak classifier}’: It performs slightly better than chance.
We set a new problem based on the results of the previous weak classifier. Each data point has a class label:

\[ y_t = \begin{cases} +1 & (\bigcirc) \\ -1 & (\bigcirc) \end{cases} \]

We update the weights:

\[ w_t \leftarrow w_t \exp\{-y_t H_t\} \]
Each data point has a class label:

\[ y_t = \begin{cases} +1 & (\color{red}\circ) \\ -1 & (\color{lightgray}\bigcirc) \end{cases} \]

We update the weights:

\[ w_t \leftarrow w_t \exp\{-y_t H_t\} \]

We set a new problem based on the results of the previous weak classifier.
Each data point has a class label:

\[ y_t = \begin{cases} 
+1 & (\bigcirc) \\
-1 & (\bigcirc) 
\end{cases} \]

We update the weights:

\[ w_t \leftarrow w_t \exp\{-y_t H_t\} \]

We set a new problem based on the results of the previous weak classifier.
Toy example

Each data point has a class label:

$$y_t = \begin{cases} 
+1 & (\bullet) \\
-1 & (\circ) 
\end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem based on the results of the previous weak classifier
The strong (non-linear) classifier is built as the combination of all the weak (linear) classifiers.
The AdaBoost Algorithm

Start with uniform weights on training examples

For T rounds

Evaluate weighted error for each feature, pick best.

Re-weight the examples:

Incorrectly classified -> more weight
Correctly classified -> less weight

Final classifier is combination of the weak ones, weighted according to error they had.

Freund & Schapire 1995

- Given example images \((x_1, y_1), \ldots, (x_n, y_n)\) where \(y_i = 0, 1\) for negative and positive examples respectively.
- Initialize weights \(w_{1,i} = \frac{1}{2m}, \frac{1}{2l}\) for \(y_i = 0, 1\) respectively, where \(m\) and \(l\) are the number of negatives and positives respectively.
- For \(t = 1, \ldots, T\):
  1. Normalize the weights,
     \[
     w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}
     \]
     so that \(w_t\) is a probability distribution.
  2. For each feature, \(j\), train a classifier \(h_j\) which is restricted to using a single feature. The error is evaluated with respect to \(w_t\), 
     \[e_j = \sum_i w_i |h_j(x_i) - y_i|\] .
  3. Choose the classifier, \(h_t\), with the lowest error \(e_t\).
  4. Update the weights:
     \[w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}\]
     where \(e_i = 0\) if example \(x_i\) is classified correctly, \(e_i = 1\) otherwise, and \(\beta_t = \frac{e_t}{1-e_t}\).
- The final strong classifier is:
  \[
  h(x) = \begin{cases} 
  1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 
  0 & \text{otherwise}
  \end{cases}
  \]
  where \(\alpha_t = \log \frac{1}{\beta_t}\).
Boosting for face detection

• A 200-feature classifier can yield 95% detection rate and a false positive rate of 1 in 14084

Not good enough!
Attentional cascade

- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows.
- Positive response from the first classifier triggers the evaluation of a second (more complex) classifier, and so on.
- A negative outcome at any point leads to the immediate rejection of the sub-window.
Develop fast, accurate classifier using a cascade

- Given a nested set of classifier hypothesis classes
- Computational Risk Minimization

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001
Cascaded Classifier

- A 1 feature classifier achieves 100% detection rate and about 50% false positive rate.
- A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative) – using data from previous stage.
- A 20 feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001
Viola-Jones Face Detector: Summary

- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade
- 6061 features in final layer
- [Implementation available in OpenCV: http://www.intel.com/technology/computing/opencv/]

K. Grauman, B. Leibe
Speed of Face Detector

- Speed is proportional to the average number of features computed per sub-window.

- On the MIT+CMU test set, an average of 9 features out of a total of 6061 are computed per sub-window.

- On a 700 Mhz Pentium III, a 384x288 pixel image takes about 0.067 seconds to process (15 fps).

- Roughly 15 times faster than Rowley-Baluja-Kanade and 600 times faster than Schneiderman-Kanade.

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001
Viola-Jones Face Detector: Results

• First two features selected by boosting:

This feature combination can yield 100% detection rate and 50% false positive rate
Viola-Jones Face Detector: Results

K. Grauman, B. Leibe
Viola-Jones Face Detector: Results
Viola-Jones Face Detector: Results

K. Grauman, B. Leibe
Detecting profile faces? Detecting profile faces requires training separate detector with profile examples.
Viola-Jones Face Detector: Results

K. Grauman, B. Leibe
Face Recognition

N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar,
Face Recognition

Attributes for training

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Positive Examples</th>
<th>Negative Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asian</td>
<td><img src="image1" alt="Positive Examples" /></td>
<td><img src="image2" alt="Negative Examples" /></td>
</tr>
<tr>
<td>Blond Hair</td>
<td><img src="image3" alt="Positive Examples" /></td>
<td><img src="image4" alt="Negative Examples" /></td>
</tr>
<tr>
<td>Child</td>
<td><img src="image5" alt="Positive Examples" /></td>
<td><img src="image6" alt="Negative Examples" /></td>
</tr>
<tr>
<td>Male</td>
<td><img src="image7" alt="Positive Examples" /></td>
<td><img src="image8" alt="Negative Examples" /></td>
</tr>
</tbody>
</table>

Similes for training

<table>
<thead>
<tr>
<th>Simile</th>
<th>Positive Examples</th>
<th>Negative Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1 Eyebrows</td>
<td><img src="image9" alt="Positive Examples" /></td>
<td><img src="image10" alt="Negative Examples" /></td>
</tr>
<tr>
<td>R1 Eyes</td>
<td><img src="image11" alt="Positive Examples" /></td>
<td><img src="image12" alt="Negative Examples" /></td>
</tr>
<tr>
<td>R1 Nose</td>
<td><img src="image13" alt="Positive Examples" /></td>
<td><img src="image14" alt="Negative Examples" /></td>
</tr>
<tr>
<td>R1 Mouth</td>
<td><img src="image15" alt="Positive Examples" /></td>
<td><img src="image16" alt="Negative Examples" /></td>
</tr>
<tr>
<td>R2 Eyebrows</td>
<td><img src="image17" alt="Positive Examples" /></td>
<td><img src="image18" alt="Negative Examples" /></td>
</tr>
<tr>
<td>R2 Eyes</td>
<td><img src="image19" alt="Positive Examples" /></td>
<td><img src="image20" alt="Negative Examples" /></td>
</tr>
<tr>
<td>R2 Nose</td>
<td><img src="image21" alt="Positive Examples" /></td>
<td><img src="image22" alt="Negative Examples" /></td>
</tr>
<tr>
<td>R2 Mouth</td>
<td><img src="image23" alt="Positive Examples" /></td>
<td><img src="image24" alt="Negative Examples" /></td>
</tr>
</tbody>
</table>

N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar,
"Attribute and Simile Classifiers for Face Verification,"
ICCV 2009.
• What other categories are amenable to window-based representation?
Pedestrian detection

- Detecting upright, walking humans also possible using sliding window’s appearance/texture; e.g.,

  SVM with Haar wavelets [Papageorgiou & Poggio, IJCV 2000]

  Space-time rectangle features [Viola, Jones & Snow, ICCV 2003]

  SVM with HoGs [Dalal & Triggs, CVPR 2005]
Window-based detection: strengths

• Sliding window detection and global appearance descriptors:
  – Simple detection protocol to implement
  – Good feature choices critical
  – Past successes for certain classes
Window-based detection: Limitations

• High computational complexity
  – For example: 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations!
  – If training binary detectors independently, means cost increases linearly with number of classes

• With so many windows, false positive rate better be low
Limitations (continued)

• Not all objects are “box” shaped
Limitations (continued)

• Non-rigid, deformable objects not captured well with representations assuming a fixed 2d structure; or must assume fixed viewpoint

• Objects with less-regular textures not captured well
Limitations (continued)

- If considering windows in isolation, context is lost.
Limitations (continued)

• In practice, often entails large, cropped training set (expensive)
• Requiring good match to a global appearance description can lead to sensitivity to partial occlusions