Today

- Review: alignment-based recognition
- Appearance-based recognition
  - Classification
    - Skin color detection example
  - Sliding window detection
    - Face detection example

Hypothesize and test: main idea

- Given model of object
- New image: hypothesize object identity and pose
- Render object in camera
- Compare rendering to actual image: if close, good hypothesis.

How to form a hypothesis?

All possible assignments of model features to image features?

Example: 2d affine mappings

- Say camera is looking down perpendicularly on planar surface
- We have two coordinate systems (object and image), and they are related by some affine mapping (rotation, scale, translation, shear).

Pose consistency / alignment

- Key idea:
  - If we find good correspondences for a small set of features, it is easy to obtain correspondences for a much larger set.
- Strategy:
  - Generate hypothesis transformation using small numbers of correspondences
  - Backproject: Transform all model features to image features
  - Verify: see if for this alignment the model and image agree
Alignment: backprojection

- Having solved for this transformation from some number of detected matches (3+ here), can compute (hypothesized) location of any other model point in the image space.

\[
\begin{bmatrix}
  u \\
  v
\end{bmatrix} = \begin{bmatrix}
  m_1 & m_2 \\
  m_3 & m_4
\end{bmatrix} \begin{bmatrix}
  x \\
  y
\end{bmatrix} + \begin{bmatrix}
  f_u \\
  f_v
\end{bmatrix}
\]

- Verify, e.g., based on edge agreement

Issue with hypothesis & test alignment approach

- May have false matches
  - We want reliable features to form the matches
    - Local invariant features useful to find matches, and to verify hypothesis
  - May be too many hypotheses to consider
    - We want to look at the most likely hypotheses first
    - Pose clustering (i.e., voting): Narrow down number of hypotheses to consider by letting features vote on model parameters.

Pose clustering and verification with SIFT [Lowe]

To detect instances of objects from a model base:

1) Index descriptors (distinctive features narrow possible matches)

Indexing local features

To detect instances of objects from a model base:

1) Index descriptors (distinctive features narrow possible matches)
2) Generalized Hough transform to vote for poses (keypoints have record of parameters relative to model coordinate system)
3) Affine fit to check for agreement between model and image features (fit and verify using features from Hough bins with 3+ votes)

Planar objects

Model images and their SIFT keypoints

Input image

Model keypoints that were used to recognize, get least squares solution.

Recognition result
Recall: difficulties of voting

- Noise/clutter can lead to as many votes as true target
- Bin size for the accumulator array must be chosen carefully
- (Recall Hough Transform)
- In practice, good idea to make broad bins and spread votes to nearby bins, since verification stage can prune bad vote peaks.

Today

• Review: alignment-based recognition

• Appearance-based recognition
  - Classification
    • Skin color detection example
  - Sliding window detection
    • Face detection example

Supervised classification

• Given a collection of labeled examples, come up with a function that will predict the labels of new examples.

“four”

“nine”

Training examples Novel input

• How good is some function we come up with to do the classification?
• Depends on
  - Mistakes made
  - Cost associated with the mistakes

Supervised classification

• Given a collection of labeled examples, come up with a function that will predict the labels of new examples.

Consider the two-class (binary) decision problem
- \( L(4 \rightarrow 9) \): Loss of classifying a 4 as a 9
- \( L(9 \rightarrow 4) \): Loss of classifying a 9 as a 4

• Risk of a classifier \( s \) is expected loss:
  \[
  R(s) = \Pr(4 \rightarrow 9 \mid s) L(4 \rightarrow 9) + \Pr(9 \rightarrow 4 \mid s) L(9 \rightarrow 4)
  \]
  • We want to choose a classifier so as to minimize this total risk
Supervised classification

Optimal classifier will minimize total risk.

At decision boundary, either choice of label yields same expected loss.

So, best decision boundary is at point \( x \) where

\[
P(\text{class} = 9 | x) L(9 \rightarrow 4) = P(\text{class} = 4 | x) L(4 \rightarrow 9)
\]

To classify a new point, choose class with lowest expected loss; i.e., choose “four” if

\[
P(4 | x) L(4 \rightarrow 9) > P(9 | x) L(9 \rightarrow 4)
\]

How to evaluate these probabilities?

Example: learning skin colors

We can represent a class-conditional density using a histogram (a “non-parametric” distribution)

\[
P(x | \text{skin})
\]

Percentage of skin pixels in each bin

Feature \( x = \text{Hue} \)

What if feature dimension is high?

Bayes rule

\[
P(\text{skin} | x) = \frac{P(x | \text{skin}) P(\text{skin})}{P(x)}
\]

\[
P(\text{skin} | x) \propto P(x | \text{skin}) P(\text{skin})
\]

Where does the prior come from?

Example: learning skin colors

We can represent a class-conditional density using a histogram (a “non-parametric” distribution)

Now we get a new image, and want to label each pixel as skin or non-skin. What’s the probability we care about to do skin detection?
Example: classifying skin pixels

Now for every pixel in a new image, we can estimate probability that it is generated by skin.

Classify pixels based on these probabilities
- if \( p(\text{skin}|x) > \theta \), classify as skin
- if \( p(\text{skin}|x) < \theta \), classify as not skin
- if \( p(\text{skin}|x) = \theta \), choose classes uniformly and at random

Figure from Gary Bradski

Example: classifying skin pixels

• Black = pixels classified as skin

Jones and Rehg, CVPR 1999.

Example: classifying skin pixels

Using skin color-based face detection and pose estimation as a video-based interface

Gary Bradski, 1998

Today

• Review: alignment-based recognition

• Appearance-based recognition
  – Classification
    - Skin color detection example
  – Sliding window detection
    - Face detection example

Detection via classification: Main idea

Basic component: a binary classifier

Car/non-car Classifier

No\text{jeq:car}.
Detection via classification: Main idea

If object may be in a cluttered scene, slide a window around looking for it. (Essentially, our skin detector was doing this, with a window that was one pixel big.)

K. Grauman, B. Leibe

Detection via classification: Main idea

Fleshing out this pipeline a bit more, we need to:
1. Obtain training data
2. Define features
3. Define classifier

Feature extraction: global appearance

Simple holistic descriptions of image content
- grayscale / color histogram
- vector of pixel intensities

Feature extraction: global appearance

Pixel-based representations sensitive to small shifts
- Color or grayscale-based appearance description can be sensitive to illumination and intra-class appearance variation

K. Grauman, B. Leibe

Eigenfaces: global appearance description

An early appearance-based approach to face recognition

Generate low-dimensional representation of appearance with a linear subspace.

Project new images to “face space”. Recognition via nearest neighbors in face space

Turk & Pentland, 1991
**Gradient-based representations**

- Consider edges, contours, and (oriented) intensity gradients

**Gradient-based representations:**

**Matching edge templates**

- Example: Chamfer matching

At each window position, compute average min distance between points on template (T) and input (I).

\[ D_{\text{Chamfer}}(T, I) = \frac{1}{|T|} \sum_{(t, i) \in T \times I} d(t, i) \]

**Gradient-based representations:**

- Consider edges, contours, and (oriented) intensity gradients

**Gradient-based representations:**

**Histograms of oriented gradients (HoG)**

Map each grid cell in the input window to a histogram counting the gradients per orientation.

- Locally orderless: offers invariance to small shifts and rotations
- Contrast-normalization: try to correct for variable illumination

**Gradient-based representations:**

**Rectangular features**

Compute differences between sums of pixels in rectangles

- Captures contrast in adjacent spatial regions, efficient to compute
- Each feature parameterized by scale, position, type.
Classifier construction

- How to compute a decision for each subwindow?

Boosting

- Build a strong classifier by combining number of “weak classifiers”, which need only be better than chance
- Sequential learning process: at each iteration, add a weak classifier
  - Flexible to choice of weak learner
    - Including fast simple classifiers that alone may be inaccurate
- We’ll look at Freund & Schapire’s AdaBoost algorithm
  - Easy to implement
  - Base learning algorithm for Viola-Jones face detector

AdaBoost: Intuition

Consider a 2-d feature space with positive and negative examples.
Each weak classifier splits the training examples with at least 50% accuracy.
Examples misclassified by a previous weak learner are given more emphasis at future rounds.

Final classifier is combination of the weak classifiers
AdaBoost Algorithm

1. Start with\[w_i = \frac{1}{N}\] for \(i = 1, \ldots, N\), where \(N\) is the number of training examples.
2. For \(T\) rounds:
   a. For each training example \(x_i\), evaluate the weighted error of the current classifier \(h(x_i)\) on \(\{x_1, \ldots, x_N\}\).
   b. Update the weights:
      \[w_i' = \frac{w_i}{\sum_{j=1}^{N} e_j^{(t)} w_j} \quad \text{where} \quad e_j^{(t)} = \begin{cases} 0 & \text{if correctly classified} \\ 1 & \text{otherwise} \end{cases}\]
   c. Choose the weak classifier \(h(x_i)\) with lowest weighted error.

Final classifier is a combination of the weak classifiers, weighted according to the error they had:

\[h(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)\]

Freund & Schapire 1995

Example: Face detection

- Frontal faces are a good example of a class where global appearance models + a sliding window detection approach fit well:
  - Regular 2D structure
  - Center of face almost shaped like a “patch”/window
- Now we’ll take AdaBoost and see how the Viola-Jones face detector works

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

Large library of filters

Considering all possible filter parameters: position, scale, and type:
180,000+ possible features associated with each 24 x 24 window

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

AdaBoost for feature-classifier 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:

\[h(x) = \begin{cases} +1 & \text{if } f(x) > \theta_i \\ -1 & \text{otherwise} \end{cases}\]

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

Faces: terminology

- Detection: given an image, where is the face?
- Recognition: whose face is it?
AdaBoost Algorithm

Start with uniform weights on training examples

For \( T \) rounds

1. Normalize the weights,
   \[ w_i = \frac{1}{\sum_{x} w(x)} \]
   so that \( w_i \) is a probability distribution.
2. For each feature, \( j \), train a classifier \( A_j \) which is restricted to using a single feature. The error is evaluated with respect to \( w \),
   \[ \epsilon_j = \frac{1}{\sum_{x} w(x)} \sum_{x} w(x) \cdot A_j(x) \cdot \epsilon_j \]
3. Choose the \( A_j \) with lowest \( \epsilon_j \).
4. Update the weights:
   \[ w_{i+1} = \frac{w_i}{Z} \cdot \epsilon_j \]
   where \( Z \) is the normalizing constant.

The final strong classifier is:
\[
h(x) = \sum_{i=1}^{T} \alpha_i A_i(x) \]
where \( \alpha_i = \log \frac{1}{\epsilon_i} \).

Freund & Schapire 1995

---

AdaBoost for Efficient Feature Selection

- Image Features = Weak Classifiers
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Sort examples by filter values
  - Select best threshold for each filter (min error)
    - Sorted list can be quickly scanned for the optimal threshold
  - Select best filter/threshold combination
  - Weight on this feature is a simple function of error rate
  - Re-weight examples

---

Cascading classifiers for detection

For efficiency, apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative; e.g.,

- Filter for promising regions with an initial inexpensive classifier
- Build a chain of classifiers, choosing cheap ones with low false negative rates early in the chain

Fleuret & Geman, IJCV 2001
Rowley et al., PAMI 1998
Viola & Jones, 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/)

---

Viola-Jones Face Detector: Results

First two features selected
Detecting profile faces?

Detecting profile faces requires training separate detector with profile examples.

Example application

Frontal faces detected and then tracked, character name inferred with alignment of script and subtitles.

Everingham, M., Sivic, J. and Zisserman, A.
“Hello! My name is... Buffy” - Automatic naming of characters in TV video, BMVC 2006.
http://www.robots.ox.ac.uk/~vgg/research/nf/face/index.html
Example application: faces in photos

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

- SVM with HOGs [Dalal & Triggs, CVPR 2005]
- SVM with Haar wavelets [Papageorgiou & Poggio, IJCV 2000]
- Space-time interest features [Viola, Jones & Snow, ICCV 2003]

Highlights
- Sliding window detection and global appearance descriptors:
  - Simple detection protocol to implement
  - Good feature choices critical
  - Past successes for certain classes

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 with holistic appearance-based descriptions

Limitations (continued)

- If considering windows in isolation, context is lost

Models based on local features will alleviate some of these limitations...