Object Category Detection: Sliding Windows

Computer Vision
CS 543 / ECE 549
University of Illinois

Derek Hoiem
Goal: Detect all instances of objects
Influential Works in Detection

• Sung-Poggio (1994, 1998) : ~1450 citations
  – Basic idea of statistical template detection (I think), bootstrapping to get “face-like” negative examples, multiple whole-face prototypes (in 1994)

  – “Parts” at fixed position, non-maxima suppression, simple cascade, rotation, pretty good accuracy, fast

  – Careful feature engineering, excellent results, cascade

  – Haar-like features, Adaboost as feature selection, hyper-cascade, very fast, easy to implement

• Dalal-Triggs (2005) : 1025
  – Careful feature engineering, excellent results, HOG feature, online code

• Felzenszwalb-McAllester-Ramanan (2008)? 105 citations
  – Excellent template/parts-based blend
Sliding window detection
What the Detector Sees
Statistical Template

- Object model = log linear model of parts at fixed positions

\[ +3 +2 -2 -1 -2.5 = -0.5 > 7.5 \]
Non-object

\[ +4 +1 +0.5 +3 +0.5 = 10.5 > 7.5 \]
Object
Design challenges

• Part design
  – How to model appearance
  – Which “parts” to include
  – How to set part likelihoods

• How to make it fast

• How to deal with different viewpoints

• Implementation details
  – Window size
  – Aspect ratio
  – Translation/scale step size
  – Non-maxima suppression
Schneiderman and Kanade

Schneiderman and Kanade

Decision function:

\[
\frac{P(image|object)}{P(image|non-object)} > \lambda \\
\lambda = \frac{P(non-object)}{P(object)}
\]
Parts model

- Part = group of wavelet coefficients that are statistically dependent
Parts: groups of wavelet coefficients

- Fixed parts within/across subbands

- 17 types of “parts” that can appear at each position

- Discretize wavelet coefficient to 3 values

- E.g., part with 8 coefficients has $3^8 = 6561$ values
Part Likelihood

• Class-conditional likelihood ratio

\[
\prod_{x, y \in \text{region}_k=1}^{17} \frac{\prod_{x, y \in \text{region}_k=1}^{17} P_k(\text{pattern}_k(x, y), x, y | \text{object})}{\prod_{x, y \in \text{region}_k=1}^{17} P_k(\text{pattern}_k(x, y), x, y | \text{non-object})} > \lambda
\]

• Estimate \( P(\text{part} | \text{object}) \) and \( P(\text{part} | \text{non-object}) \) by counting over examples

\[
P(\text{part} | \text{object}) = \frac{\text{count}(\text{part} & \text{object})}{\text{count}(\text{object})}
\]

• Adaboost tunes weights discriminatively
Training

1) Create training data
   a) Get positive and negative patches
   b) Pre-process (optional), compute wavelet coefficients, discretize
   c) Compute parts values

2) Learn statistics
   a) Compute ratios of histograms by counting for positive and negative examples
   b) Reweight examples using Adaboost, recount, etc.

3) Get more negative examples (bootstrapping)
Training multiple viewpoints

Train new detector for each viewpoint.
Testing

1) Processing:
   a) Lighting correction (optional)
   b) Compute wavelet coefficients, quantize

2) Slide window over each position/scale (2 pixels, $2^{1/4}$ scale)
   a) Compute part values
   b) Lookup likelihood ratios
   c) Sum over parts
   d) Threshold

3) Use faster classifier to prune patches (cascade...more on this later)

4) Non-maximum suppression
Results: faces

Table 1. Face detection with out-of-plane rotation

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>Detection (all faces)</th>
<th>Detection (profiles)</th>
<th>False Detections</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>92.7%</td>
<td>92.8%</td>
<td>700</td>
</tr>
<tr>
<td>1.5</td>
<td>85.5%</td>
<td>86.4%</td>
<td>91</td>
</tr>
<tr>
<td>2.5</td>
<td>75.2%</td>
<td>78.6%</td>
<td>12</td>
</tr>
</tbody>
</table>

208 images with 441 faces, 347 in profile
Results: cars

Table 3. Car detection

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>Detections</th>
<th>False Detections</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.05</td>
<td>83%</td>
<td>7</td>
</tr>
<tr>
<td>1.0</td>
<td>86%</td>
<td>10</td>
</tr>
<tr>
<td>0.9</td>
<td>92%</td>
<td>71</td>
</tr>
</tbody>
</table>
Results: faces today

http://demo.pittpatt.com/
Viola and Jones

**Fast** detection through two mechanisms

Integral Images

• “Haar-like features”
  – Differences of sums of intensity
  – Thousands, computed at various positions and scales within detection window

-1 +1

Two-rectangle features

Three-rectangle features

Etc.
Integral Images

\[ ii = \text{cumsum} (\text{cumsum} (Im, 1), 2) \]

\[ ii(x,y) = \text{Sum of the values in the grey region} \]

How to compute \( A + D - B - C \)?

How to compute \( B - A \)?

How to compute \( A + D - B - C \)?
Adaboost as feature selection

- Create a large pool of parts (180K)
- “Weak learner” = feature + threshold + parity

\[ h_j(x) = \begin{cases} 
1 & \text{if } p_j f_j(x) < p_j \theta_j \\
0 & \text{otherwise} 
\end{cases} \]

- Choose weak learner that minimizes error on the weighted training set
- Reweight
Adaboost

- 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 \(w_{1,i} = \frac{1}{m}, \frac{1}{n}\) 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 \(\alpha_t = \log \frac{1}{\beta_t}\)
Adaboost

“RealBoost”

Given: \((x_1, y_1), \ldots, (x_m, y_m)\) where \(x_\ell \in X, y_\ell \in Y = \{-1, +1\}\)
Initialize \(D_1(\ell) = 1/m\).
For \(t = 1, \ldots, T\):

- Train base learner using distribution \(D_t\).
- Get base classifier \(h_\ell : X \rightarrow \mathbb{R}\).
- Choose \(\alpha_t \in \mathbb{R}\).
- Update:

\[
D_{t+1}(\ell) = \frac{D_t(\ell) \exp(-\alpha_t y_\ell h_\ell(x_\ell))}{Z_t}
\]

where \(Z_t\) is a normalization factor (chosen so that \(D_{t+1}\) will be a distribution).

Output the final classifier:

\[
H(x) = \text{sign} \left( \sum_{\ell=1}^T \alpha_\ell h_\ell(x) \right).
\]

Figure 1: The boosting algorithm AdaBoost.

Important special case: \(h_t\) partitions input space:

\[
c_j = \frac{1}{2} \ln \left( \frac{W_+}{W_-} \right)
\]

\(\alpha_t\)
Adaboost: Immune to Overfitting?
Interpretations of Adaboost

• Additive logistic regression (Friedman et al. 2000)
  – LogitBoost from Collins et al. 2002 does this more explicitly

• Margin maximization (Schapire et al. 1998)
  – Ratch and Warmuth 2002 do this more explicitly
Adaboost: Margin Maximizer

Test error

Train error

margin
Cascade for Fast Detection

- Choose threshold for low false negative rate
- Fast classifiers early in cascade
- Slow classifiers later, but most examples don’t get there
Viola-Jones details

- 38 stages with 1, 10, 25, 50 ... features
  - 6061 total used out of 180K candidates
  - 10 features evaluated on average

- Examples
  - 4916 positive examples
  - 10000 negative examples collected after each stage

- Scanning
  - Scale detector rather than image
  - Scale steps = 1.25, Translation 1.0*s to 1.5*s

- Non-max suppression: average coordinates of overlapping boxes

- Train 3 classifiers and take vote
Viola Jones Results

<table>
<thead>
<tr>
<th>Detector</th>
<th>10</th>
<th>31</th>
<th>50</th>
<th>65</th>
<th>78</th>
<th>95</th>
<th>167</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viola-Jones</td>
<td>76.1%</td>
<td>88.4%</td>
<td>91.4%</td>
<td>92.0%</td>
<td>92.1%</td>
<td>92.9%</td>
<td>93.9%</td>
</tr>
<tr>
<td>Viola-Jones (voting)</td>
<td>81.1%</td>
<td>89.7%</td>
<td>92.1%</td>
<td>93.1%</td>
<td>93.1%</td>
<td>93.2%</td>
<td>93.7%</td>
</tr>
<tr>
<td>Rowley-Baluja-Kanade</td>
<td>83.2%</td>
<td>86.0%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>89.2%</td>
<td>90.1%</td>
</tr>
<tr>
<td>Schneiderman-Kanade</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>94.4%</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Roth-Yang-Ahuja</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>(94.8%)</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

MIT + CMU face dataset
Schneiderman later results

<table>
<thead>
<tr>
<th></th>
<th>89.7%</th>
<th>93.1%</th>
<th>94.4%</th>
<th>94.8%</th>
<th>95.7%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayesian Network</td>
<td>1</td>
<td>8</td>
<td>19</td>
<td>36</td>
<td>56</td>
</tr>
<tr>
<td>Semi-Naive Bayes*</td>
<td>6</td>
<td>19</td>
<td>29</td>
<td>35</td>
<td>46</td>
</tr>
<tr>
<td>[6]</td>
<td>31</td>
<td>65</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>[7]*</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>78</td>
<td>--</td>
</tr>
<tr>
<td>[16]*</td>
<td>--</td>
<td>--</td>
<td>65</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

Table 2. False alarms as a function of recognition rate on the MIT-CMU Test Set for Frontal Face Detection. * indicates exclusion of the 5 images of hand-drawn faces.
Speed: frontal face detector

- Schneiderman-Kanade (2000): 5 seconds
- Viola-Jones (2001): 15 fps
Occlusions?

• A problem

• Objects occluded by > 50% considered “don’t care”

• PASCAL VOC changed this
Strengths and Weaknesses of Statistical Template Approach

Strengths

• Works very well for non-deformable objects: faces, cars, upright pedestrians
• Fast detection

Weaknesses

• Not so well for highly deformable objects
• Not robust to occlusion
• Requires lots of training data
SK vs. VJ

Schneiderman-Kanade
- Wavelet features
- Log linear model via boosted histogram ratios
- Bootstrap training
- Two-stage cascade
- NMS: Remove overlapping weak boxes
- Slow but very accurate

Viola-Jones
- Similar to Haar wavelets
- Log linear model via boosted stubs
- Bootstrap training
- Multistage cascade, integrated into training
- NMS: average coordinates of overlapping boxes
- Less accurate but very fast
Things to remember

- Excellent results require careful feature engineering
- Sliding window for search
- Features based on differences of intensity (gradient, wavelet, etc.)
- Boosting for feature selection (also L1-logistic regression)
- Integral images, cascade for speed
- Bootstrapping to deal with many, many negative examples