Generic Object Detection using AdaBoost

Ben Weber

University of California, Santa Cruz

Last update: March 20, 2008
Outline

1. Introduction
2. Related Work
3. Object Detection
4. Implementation
5. Results
6. Conclusion
Introduction

- Generic object detection is one of the main challenges for computer vision
- Current research focuses on a single object class
- Face detection is a common benchmark problem
- Viola and Jones show good results for face detection using AdaBoost
- This project explores the application of AdaBoost to aircraft detection

[Viola and Jones, 2001]
Outline

1. Introduction
2. Related Work
3. Object Detection
4. Implementation
5. Results
6. Conclusion
Related Work

Detection approaches
- Support vector machines
- Neural networks
- Example-based learning
- Boosting

AdaBoost has been shown to be the fastest detection technique and achieves 95% accuracy

[Viola and Jones, 2001]
Related Problems

- Face recognition
- Rotated face detection
- Side-profile face detection
- Real-time object detection
- Object classification
- Generic object detection
- Active learning
Outline

1. Introduction
2. Related Work
3. Object Detection
4. Implementation
5. Results
6. Conclusion
Object Detection

- Training the detector
  - Select a dataset of positive and negative examples
  - Train the threshold values for each feature
  - Select and train a subset of the weak learners
  - Train the attentional cascade

- Using the detector
  - Exhaustively scan each input image
Features

- Viola and Jones suggest using features rather than pixels
  - Enables encoding of ad-hoc domain knowledge
  - Faster than using pixels directly
- Features are rectangular regions similar to Haar basis functions
- The value of a feature is computed as the difference between the sums of pixels in dark and light regions

[Papageorgiou et al. 1998]
Features

- 3 types of features
  - 2 rectangle features
  - 3 rectangle features
  - 4 rectangle features
- Over 270,000 features
Integral Image

- Provides constant lookup times for the sum of pixels in a rectangular region
- Each value in the integral image contains the sum of the pixels above and to the left:
  \[ ii(x, y) = \sum_{x', y' \leq x, y'} i(x', y') \]
- Can be computed using the following recurrences:
  \[ s(x, y) = s(x, y-1) + i(x, y) \]
  \[ ii(x, y) = ii(x-1, y) + s(x, y) \]
- where \( s(x, 0) = 0 \) and \( ii(0, y) = 0 \)
The sum of any rectangular region can be computed in 4 array references.

The sum within D is \((4 + 1) - (2 + 3)\)
Weak learners are constrained to using a single feature

A learner consists of a feature $f_j$ and a threshold $\theta_j$:

$$h(x) = \begin{cases} 
1 & f_j < \theta_j \\
0 & \text{otherwise}
\end{cases}$$

The threshold value is chosen to minimize the number of misclassified examples

Best weak learner has an error of approximately 0.07
Classifier Training

- 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 \(w_i = 1/2M, 1/2L\) for \(y_i = 0, 1\) respectively, where \(M\) and \(L\) are the number of negatives and positives.

- For \(t = 1\) to \(T\)
  1. Normalize the weights, so that \(\sum_{i=1:n} w_i = 1\)
  2. Choose the classifier, \(h_i\), with the lowest error \(\varepsilon_i\):

\[
\varepsilon_j = \sum_i w_i |h_j(x_i) - y_i|.
\]

3. Update the weights for each example:

\[
w_i = \begin{cases} 
  w_i \beta_t & \text{example } x_i \text{ is classified correctly} \\
  w_i & \text{otherwise}
\end{cases}
\]

where \(\beta_t = \varepsilon_t/(1 - \varepsilon_t)\) and \(\alpha_t = -\log \beta_t\)

- The final strong classifier is:

\[
h(x) = \begin{cases} 
  1 & \sum_{t=1:T} \alpha_t h_t(x) \geq \lambda, \sum_{t=1:T} \alpha_t, \text{ where } \lambda = \frac{1}{2} \\
  0 & \text{otherwise}
\end{cases}
\]
Attentional Cascade

- Degenerate decision tree
- Minimizes the number of features evaluated

[Fleuret and Geman, 2001]
The system exhaustively scans images for objects
- Input images are converted to grayscale and an integral image is generated for each image
- The detector starts with a scale of 1.0 and evaluates every sub-window with the strong classifier
- The scale is then increased and the image is rescanned

Features are scaled rather than the image
Outline

1. Introduction
2. Related Work
3. Object Detection
4. Implementation
5. Results
6. Conclusion
Implementation

- Face detection system
- Training
  - Acquire annotated face dataset
  - Train the feature thresholds
  - Train the classifiers
  - Tune the cascading filters

- Specs
  - Java implementation
  - 5-10 frames per second at 320x240
Face Dataset

- Cal Tech 101 face dataset
  - 101 categories of annotated images
  - 400 faces as positive examples
  - Mirrored versions for 400 more examples
  - 800 negatives selected at random from other categories
Training the Detector

- Threshold values selected by choosing the best result from 50 linearly spaced points
- Classifier trained to 2000 features
### Training the Cascading Filters

- Several filters reject sub-windows
- Modified threshold values to limit number of rejected faces

<table>
<thead>
<tr>
<th></th>
<th>Filter 1</th>
<th>Filter 2</th>
<th>Filter 3</th>
<th>Filter 4</th>
<th>Filter 5</th>
<th>Filter 6</th>
<th>Filter 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of features</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Threshold increase</td>
<td>1</td>
<td>5</td>
<td>10</td>
<td>50</td>
<td>200</td>
<td>1000</td>
<td>2000</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>0.55</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.8</td>
<td>0.8</td>
<td>0.75</td>
</tr>
</tbody>
</table>
Real-Time Face Detection

- java.awt.Robot class for screen captures
- Face detector runs in separate thread
Face Detection Results

- CMU TestSet C
  - 65 images containing 182 faces
  - 54.3% accuracy
  - 83 false positives
Outline

1. Introduction
2. Related Work
3. Object Detection
4. Implementation
5. Results
6. Conclusion

Ben Weber (UCSC)  Generic Object Detection using AdaBoost  Last update: March 20, 2008 25 / 31
Results

- CalTech 101 dataset
  - 800 positive examples from the aircraft category
  - 800 negative examples selected from random categories

- Testing
  - Examples partitioned into two sets
  - Separate classifiers trained for each set
  - Classifiers trained to 5000 weak learners
  - Classifiers validated on the holdout dataset
Aircraft Detection

<table>
<thead>
<tr>
<th></th>
<th>Filter 1</th>
<th>Filter 2</th>
<th>Filter 3</th>
<th>Filter 4</th>
<th>Filter 5</th>
<th>Filter 6</th>
<th>Filter 7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of features</strong></td>
<td>1</td>
<td>5</td>
<td>10</td>
<td>50</td>
<td>200</td>
<td>1000</td>
<td>2000</td>
</tr>
<tr>
<td><strong>Threshold increase</strong></td>
<td>λ</td>
<td>λ</td>
<td>λ</td>
<td>λ</td>
<td>λ</td>
<td>λ</td>
<td>λ</td>
</tr>
<tr>
<td><strong>λ</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ROC Curve for Aircraft Set 1**

**ROC Curve for Aircraft Set 2**
Validation Dataset

- 10 images containing 20 aircraft and 10 negative examples
Aircraft Detections
Conclusion
Conclusion

- AdaBoost provides a technique for fast object detection
- Accuracy for aircraft detection required a high false positive rate
- More training and more filters should improve accuracy
- Some features may be overfitting the data

Future Work
- Try different features
- Heuristics for feature selection
- Parallel processing for detecting multiple object classes