Real-time face detection with Haar cascades

Nora Kassner
Outline

- The Algorithm:

  2001 by Paul Viola & Michael Jones

- Training & the final detector
- OpenCV

Haar-like features
Integral image
Cascade
AdaBoost
Choice of features

- How many features should be used?
- What makes a good feature?

- Pixels

- Histograms

- Haar-like features: local, oriented intensity differences
Haar wavelets

Original-Matrix

<table>
<thead>
<tr>
<th>10</th>
<th>14</th>
<th>7</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>12</td>
<td>13</td>
<td>19</td>
</tr>
<tr>
<td>14</td>
<td>12</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Zweidimensionale Basisfunktionen für ein 4×4-Bild

http://www.tilman.de/uni/ws05/scivis/2d-transformation.html
Haar-like features

→ 160,000 features per 24x24 px window

1. Edge features
   (a) (b) (c) (d)

2. Line features
   (a) (b) (c) (d) (e) (f) (g) (h)

3. Center-surround features
   (a) (b)

→ Features encode knowledge
→ Sensitive to edges, bars, simple structure
Integral Image

→ Very fast

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Input image

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
</tr>
</tbody>
</table>

Integral image

![Diagram ofIntegral Image with points A, B, C, D]

Sum of grey rectangle = D - (B + C) + A

Number of array references: 6 8 9
AdaBoost

→ narrowing down number of features to only a few useful ones

- Weak classifier: perform at least better than random:

\[ h(x, f, p, \theta) = \begin{cases} 
1 & \text{if } pf(x) > p\theta \\
0 & \text{otherwise} 
\end{cases} \]

- Combining weak classifiers in a weighted sum to form a strong classifier:

\[ C(x) = \begin{cases} 
1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\
0 & \text{otherwise} 
\end{cases} \]
AdaBoost

- Given examples images \((x_1, y_1), \ldots, (x_n, y_n)\) where \(y_1 = 0, 1\) for negative and positive examples.
- Initialize weights \(w_{i,1} = \frac{1}{2m}, \frac{1}{2l}\) for \(y_1 = 0, 1\), where \(m\) and \(l\) are the numbers of positive and negative examples.
- For \(t = 1, \ldots, T\):
  1) Normalize the weights, \(w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{\pi} w_{t,j}}\)
  2) Select the best weak classifier with respect to the weighted error:
     \[ e_t = \min_{f, p, \theta} \sum_{i} w_i |h(x_i, f, p, \theta) - y_i| \]
  3) Define \(h_t(x) = h(x, f_t, p_t, \theta_t)\) where \(f_t, p_t\) and \(\theta_t\) are the minimizers of \(e_t\).
  4) Update the weights:
     \[ w_{t+1,i} = w_{t,i} \beta^{1-e_t} \]
     where \(e_t = 0\) if example \(x_i\) is classified correctly and \(e_t = 1\) otherwise, and \(\beta_t = \frac{e_t}{1-e_t}\)
- The final strong classifier is:
  \[ C(x) = \begin{cases} 
  1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\
  0 & \text{otherwise}
  \end{cases} \]
  where \(\alpha_t = \log \frac{1}{\beta_t}\)
The cascade

→ Focus of attention

![Image of a cascade diagram]

Training the cascade

- AdaBoost
  → minimize false negative

- Parameters:
  - # stages
  - # features per stage
  - Threshold of each stage

Select:
- Max. false positive / stage
- Min. true positive / stage
- Target overall false positive

See how the cascade looks like:
http://www.makematics.com/research/viola-jones/
Training the Cascade

- User selects values for $f$, the maximum acceptable false positive rate per layer and $d$, the minimum acceptable detection rate per layer.
- User selects target overall false positive rate, $F_{\text{target}}$.
- $P =$ set of positive examples
- $N =$ set of negative examples
- $F_0 = 1.0; D_0 = 1.0$
- $i = 0$
- while $F_i > F_{\text{target}}$
  - $i \leftarrow i + 1$
  - $n_i = 0; F_i = F_{i-1}$
  - while $F_i > f \times F_{i-1}$
    - $n_i \leftarrow n_i + 1$
    - Use $P$ and $N$ to train a classifier with $n_i$ features using AdaBoost
    - Evaluate current cascaded classifier on validation set to determine $F_i$ and $D_i$.
    - Decrease threshold for the $i$th classifier until the current cascaded classifier has a detection rate of at least $d \times D_{i-1}$ (this also affects $F_i$)
- $N \leftarrow \emptyset$
- If $F_i > F_{\text{target}}$ then evaluate the current cascaded detector on the set of non-face images and put any false detections into the set $N$
The final detector

- 6000 features
- 38 stages
- Input parameters:
  - Cascade containing features
  - Starting scale
  - Starting delta
  - Scale increment
- 15 frames/s
OpenCV

- Free for academic & commercial use
- Link installation instruction: http://docs.opencv.org/master/d9/df8/tutorial_root.html
- C++, C, Python and Java interfaces
- Supports Windows, Linux, Mac OS, iOS and Android

Stuff you can do with it:

https://www.youtube.com/watch?v=oJAl9Yd3kNo
https://www.youtube.com/watch?v=8h9vU1pnNZA
Do it yourself!
Do it yourself!

```
main@NorasT520 ~/OpenCV/TrainCars $ ls
bg.txt cars.info neg pos
main@NorasT520 ~/OpenCV/TrainCars $ opencv_createsamples -info cars.info -num 500 -w 48 -h 24 -vec cars.vec

main@NorasT520 ~/OpenCV/TrainCars $ opencv_traincascade -data data -vec cars.vec -bg bg.txt -numPos 500 -numNeg 500 -numStages 10 -w 48 -h 24
```

http://imgur.com/gallery/QWQiBYU
Sources


- http://docs.opencv.org/master/index.html


- http://www.makematics.com/research/viola-jones/

- http://www.tilman.de/uni/ws05/scivis/wavelet-transformation.html