Experiments with Object Detection using Haar-like Features

Harshdeep Singh
Jan 29, 2009
Outline

- Background
- A walkthrough of cascade creation
- Visualizing a couple of cascades
- Detecting different types of objects
- Training with a single image
- Incorporating color information to improve performance (for face detection)
The Detector

- Proposed by [Viola, Jones 2001]
- Using boosted cascades of Haar-like features
- Implementation available in OpenCV
Haar-like features

- feature = \( w_1 \times \text{RecSum}(r_1) + w_2 \times \text{RecSum}(r_2) \)
- Weights can be positive or negative
- Weights are directly proportional to the area
- Calculated at every point and scale
Weak Classifier

- A **weak classifier** \( h(x, f, p, \theta) \) consists of
  - feature \( f \)
  - threshold \( \theta \)
  - polarity \( p \), such that

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

- Requirement
  - Should perform better than random chance
Attentional Cascade

- Initial stages have less features (faster computation)
- More time spent on evaluating more promising sub-windows
Cascade Creation - Walkthrough

Positive Samples
200 distorted versions of a synthetic image
Cascade Creation - Walkthrough

Positive Samples
200 distorted versions of a synthetic image
Cascade Creation - Walkthrough

Negative Samples
100 images from BACKGROUND_Google category of Caltech 101 dataset
Cascade Creation - Walkthrough

• Input Parameters
  – \( d \) = Minimum acceptable detection rate per layer (0.995)
  – \( f \) = Maximum acceptable false positive rate per layer (0.5)
  – \( F_{\text{target}} \) = Target overall false positive rate
    • Or maximum number of stages in the cascade
    • For \( n_{\text{Stages}} = 14 \), \( F_{\text{target}} = f^{n_{\text{Stages}}} = 6.1 \times 10^{-5} \)
$F_0 = 1$

$i = 0$

while $F_i > F_{\text{target}}$ and $i < n\text{Stages}$

\[ i = i + 1 \]

Train Classifier for stage $i$

- Initialize Weights
- Normalize Weights
- Pick the (next) best weak classifier
- Update Weights
- Evaluate $f_i$

if $f_i > f$

\[ \text{go back to Normalize Weights} \]

Combine weak classifiers to form the strong stage classifier

Evaluate $F_i$

If $F_i > F_{\text{target}}$, $N =$ set of negative samples that are labeled positive by current detector
Cascade Creation - Walkthrough

\[ F_0 = 1 \]
\[ i = 0 \]
while \( F_i > F_{\text{target}} \) and \( i < n\text{Stages} \)
\[ i = i + 1 \]
Train Classifier for stage \( i \)
  
  Initialize Weights
  Normalize Weights
  Pick the (next) best weak classifier
  Update Weights
  Evaluate \( f_i \)
  if \( f_i > f \)
    go back to Normalize Weights
  Combine weak classifiers to form the strong stage classifier
  Evaluate \( F_i \)
  If \( F_i > F_{\text{target}} \) N = set of negative samples that are labeled positive by current detector
Cascade Creation - Walkthrough

\[ F_0 = 1 \]
\[ i = 0 \]

while \( F_i > F_{\text{target}} \) and \( i < n_{\text{Stages}} \)

\[ i = i + 1 \]

Train Classifier for stage \( i \)

1. Initialize Weights
2. Normalize Weights
3. Pick the (next) best weak classifier
4. Update Weights
5. Evaluate \( f_i \)

if \( f_i > f \)

\[ \text{go back to Normalize Weights} \]

Combine weak classifiers to form the strong stage classifier

Evaluate \( F_i \)

If \( F_i > F_{\text{target}} \), \( N \) = set of negative samples that are labeled positive by current detector
Cascade Creation - Walkthrough

\[ F_0 = 1 \]
\[ i = 0 \]

while \( F_i > F_{\text{target}} \) and \( i < n_{\text{Stages}} \)

\[ i = i + 1 \]

Train Classifier for stage \( i \)

- **Initialize Weights**
- Normalize Weights
- Pick the (next) best weak classifier
- Update Weights
- Evaluate \( f_i \)
- if \( f_i > f \)
  - go back to Normalize Weights

Combine weak classifiers to form the strong stage classifier

Evaluate \( F_i \)

If \( F_i > F_{\text{target}} \), \( N \) = set of negative samples that are labeled positive by current detector

Weight for each

- positive sample \( 0.5/m \)
- negative sample \( 0.5/n \)

\( m \) – number of positive samples (200)
\( n \) – number of negative samples (100)
Cascade Creation - Walkthrough

\[ F_0 = 1 \]
\[ i = 0 \]
\[ \text{while } F_i > F_{\text{target}} \text{ and } i < n_{\text{Stages}} \]
\[ i = i + 1 \]

Train Classifier for stage \( i \)
- Initialize Weights
- Normalize Weights
- Pick the (next) best weak classifier
- Update Weights
- Evaluate \( f_i \)
  - if \( f_i > f \)
    - go back to Normalize Weights
- Combine weak classifiers to form the strong stage classifier
- Evaluate \( F_i \)
  - If \( F_i > F_{\text{target}} \)

\[ m = \text{set of negative samples that are labeled positive by current detector} \]

Weight for each
- positive sample \( \frac{0.5}{m} \)
- negative sample \( \frac{0.5}{n} \)

\( m \) – number of positive samples (200)
\( n \) – number of negative samples (100)
Cascade Creation - Walkthrough

\[ F_0 = 1 \]
\[ i = 0 \]

while \( F_i > F_{\text{target}} \) and \( i < n\text{Stages} \)
\[ i = i + 1 \]

Train Classifier for stage \( i \)
- Initialize Weights
- Normalize Weights
- Pick the (next) best weak classifier
- Update Weights
- Evaluate \( f_i \)
- if \( f_i > f \)
  go back to Normalize Weights

Combine weak classifiers to form the strong stage classifier
- Evaluate \( F_i \)
- If \( F_i > F_{\text{target}} \), \( N \) = set of negative samples that are labeled positive by current detector

The one with minimum error

\[ \epsilon_t = \min_{f,p,\theta} \sum_i w_i |h(x_i, f, p, \theta) - y_i| \]

\[ \epsilon_t = 0.005 \]
Error minimization

Positive samples

Negative samples
Error minimization
Error minimization

\[ e_1 = S^+ + (T^- - S^-) \]
\[ e_2 = S^- + (T^+ - S^+) \]
\[ e = \min(e_1, e_2) \]

Sum of weights of
\( T^+ \): All +ve examples
\( T^- \): All -ve examples
\( S^+ \): +ve examples below the current one
\( S^- \): -ve examples below the current one
Cascade Creation - Walkthrough

\[ F_0 = 1 \]
\[ i = 0 \]
while \( F_i > F_{\text{target}} \) and \( i < n\text{Stages} \)
\[ i = i + 1 \]
Train Classifier for stage \( i \)
  Initialize Weights
  Normalize Weights
  Pick the (next) best weak classifier
Update Weights
Evaluate \( f_i \)
if \( f_i > f \)
  go back to Normalize Weights
Combine weak classifiers to form the strong stage classifier
Evaluate \( F_i \)
If \( F_i > F_{\text{target}} \) \( N \) = set of negative samples that are labeled positive by current detector

\[ w_{t+1,t} = w_{t,t} \beta_t^{1-\epsilon_t} \]
\[ \epsilon_t = 0, \text{ if example } x_i \text{ is classified correctly} \]
\[ \epsilon_t = 1, \text{ otherwise} \]

\[ \beta_t = \frac{\epsilon_t}{1 - \epsilon_t} \]
Cascade Creation - Walkthrough

\[ F_0 = 1 \]
\[ i = 0 \]

while \( F_i > F_{\text{target}} \) and \( i < n\text{Stages} \)
\[ i = i + 1 \]

Train Classifier for stage \( i \)
  Initialize Weights
  Normalize Weights
  Pick the (next) best weak classifier
  Update Weights
  Evaluate \( f_i \)
  if \( f_i > f \)
    go back to Normalize Weights
  Combine weak classifiers to form the strong stage classifier
  Evaluate \( F_i \)
  If \( F_i > F_{\text{target}} \), \( N = \) set of negative samples that are labeled positive by current detector

\[ f_i = \frac{\text{number of negative samples that were detected by this stage}}{\text{total number of negative samples}} = \frac{1}{100} \]
Cascade Creation - Walkthrough

\[ F_0 = 1 \]
\[ i = 0 \]

while \( F_i > F_{\text{target}} \) and \( i < n_{\text{Stages}} \)

\[ i = i + 1 \]

Train Classifier for stage \( i \)

- Initialize Weights
- Normalize Weights
- Pick the (next) best weak classifier
- Update Weights
- Evaluate \( f_i \)
- if \( f_i > f \)
  - go back to Normalize Weights

Combine weak classifiers to form the strong stage classifier

Evaluate \( F_i \)

If \( F_i > F_{\text{target}} \), \( N \) = set of negative samples that are labeled positive by current detector
Cascade Creation - Walkthrough

\[ F_0 = 1 \]
\[ i = 0 \]

while \( F_i > F_{\text{target}} \) and \( i < n\text{Stages} \)

\[ i = i + 1 \]

Train Classifier for stage \( i \)

- Initialize Weights
- Normalize Weights
- Pick the (next) best weak classifier
- Update Weights
- Evaluate \( f_i \)
  
  if \( f_i > f \)
  
  go back to Normalize Weights

Combine weak classifiers to form the strong stage classifier

Evaluate \( F_i \)

If \( F_i > F_{\text{target}} \), \( N = \) set of negative samples that are labeled positive by current detector

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

\[
\alpha_t = \log \frac{1}{\beta_t} \quad \beta_t = \frac{\epsilon_t}{1 - \epsilon_t}
\]
Cascade Creation - Walkthrough

$$F_0 = 1$$
$$i = 0$$
while $$F_i > F_{\text{target}}$$ and $$i < n\text{Stages}$$
    $$i = i + 1$$
    Train Classifier for stage i
        Initialize Weights
        Normalize Weights
        Pick the (next) best weak classifier
        Update Weights
        Evaluate $$f_i$$
        if $$f_i > f$$
            go back to Normalize Weights
        Combine weak classifiers to form the strong stage classifier
    Evaluate $$F_i$$
    If $$F_i > F_{\text{target}}$$, $$N$$ = set of negative samples that are labeled positive by current detector
Cascade Creation - Walkthrough

\[ F_0 = 1 \]
\[ i = 0 \]

while \( F_i > F_{\text{target}} \) and \( i < n_{\text{Stages}} \)

\[ i = i + 1 \]

Train Classifier for stage \( i \)

- Initialize Weights
- Normalize Weights
- Pick the (next) best weak classifier
- Update Weights
- Evaluate \( f_i \)
- if \( f_i > f \)
  
  go back to Normalize Weights

Combine weak classifiers to form the strong stage classifier

Evaluate \( F_i \)

If \( F_i > F_{\text{target}} \)

\( N = \) set of negative samples that are labeled positive by current detector
If $F_{\text{target}}$ (maximum false alarm rate) is increased from 0.05 to 0.2, a cascade with only the first two stages is created.
If $F_{\text{target}}$ (maximum false alarm rate) is increased from 0.05 to 0.2, a cascade with only the first two stages is created.
Which features actually get selected?

(OpenCV’s default frontal face cascade)

Stage 0

Stage 1

Stage 21

... 10 more

... 206 more
Caltech 101 dataset

- 101 categories
- 40 to 800 images per category
- Each image is roughly 300x200 pixels
“Most images have little or no clutter. The objects tend to be centered in each image. Most objects are presented in a stereotypical pose.”
Detecting different types of objects

1. Train a cascade from:
   - Positive Samples (60% of images from Faces_easy category)
   - Negative Samples (60% of images in Background_Google category)

2. Test on the rest of the images from Faces_easy and Background_Google categories
3. Repeat with another category
Detecting different types of objects
Variation in Training Images

High accuracy categories

Low accuracy categories
Training with a Single Image

Hand label ROI in 40/64 images

Generate 1000 random distortions of a representative image

Negative samples taken from BACKGROUND_Google category of Caltech 101

Some features that get selected
Performance

Hand label ROI

0.05 0.1 0.15 0.2 0.25 0.3

Precision

Recall

Hand label ROI

Random distortions
Skin Color Approximation

• To filter results of face detector
• Derived from [Bradsky 1998]
• Template Image
  – Patches of faces of different subjects under varying lighting conditions
Skin Color Approximation

Replace each pixel’s value by the count in the corresponding histogram bin

Create hue histogram

Normalize [0 – 255]

Back Projection

RGB -> HSV

Face image

$S = \text{Sum of pixel values in the back-projection} / \text{Area}$

$S > \text{Threshold?}$

Y

N
With skin color filter

Without skin color filter

Evaluated on 435 face images in the Caltech 101 dataset
When does it help?

Without skin filter

With skin filter
1. Viola Jones’ technique worked pretty well for faces and some other categories like airplanes and car_sides.

2. Did not work well with some categories. Accuracy depends largely on the amount of variation in training and test images. It also depends on the amount of background clutter in the training images.

3. In some cases, the training algorithm is not able to go below the maximum false alarm rate of a layer, even with a very large number of features.

4. Selected features for the first few stages are more “intuitive” than the later ones.

5. Skin color can be used to increase the precision of face detection at the cost of recall. Dependent on illumination.

6. Training classifiers is slow! Let OpenCV use as much memory as you have.