Lit part of blue dress and shadowed part of white dress are the same color
Recap: Viola-Jones sliding window detector

**Fast** detection through two mechanisms

- Quickly eliminate unlikely windows
- Use features that are fast to compute

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
Features that are fast to compute

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

- \( ii = \text{cumsum}(\text{cumsum}(im, 1), 2) \)

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

SUM within Rectangle D is
\[ ii(4) - ii(2) - ii(3) + ii(1) \]
Feature selection with Adaboost

- Create a large pool of features (180K)
- Select features that are discriminative and work well together
  - “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
Viola Jones Results
Speed = 15 FPS (in 2001)

MIT + CMU face dataset

<table>
<thead>
<tr>
<th>Detector</th>
<th>False detections</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Viola-Jones</td>
<td>76.1%</td>
</tr>
<tr>
<td>Viola-Jones (voting)</td>
<td>81.1%</td>
</tr>
<tr>
<td>Rowley-Baluja-Kanade</td>
<td>83.2%</td>
</tr>
<tr>
<td>Schneiderman-Kanade</td>
<td>-</td>
</tr>
<tr>
<td>Roth-Yang-Ahuja</td>
<td>-</td>
</tr>
</tbody>
</table>
Object Detection

- Overview
- Viola-Jones
- Dalal-Triggs
- Deformable models
- Deep learning
Statistical Template

Object model = sum of scores of features at fixed positions

\[ +3 +2 -2 -1 -2.5 = -0.5 \geq 7.5 \]

Non-object

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

Object
Example: Dalal-Triggs pedestrian detector

1. Extract fixed-sized (64x128 pixel) window at each position and scale
2. Compute HOG (histogram of gradient) features within each window
3. Score the window with a linear SVM classifier
4. Perform non-maxima suppression to remove overlapping detections with lower scores

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05
Input image → Normalize gamma & colour → Compute gradients → Weighted vote into spatial & orientation cells → Contrast normalize over overlapping spatial blocks → Collect HOG’s over detection window → Linear SVM → Person / non-person classification
• Tested with
  – RGB
  – LAB
  – Grayscale

• Gamma Normalization and Compression
  – Square root  Slightly better performance vs. grayscale
  – Log  Very slightly better performance vs. no adjustment
Outperforms

-1 0 1
centered

-1 1
uncentered

1 -8 0 8 -1
cubic-corrected

0 1
diagonal

-1 0 1
-2 0 2
-1 0 1
Sobel

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05
• Histogram of gradient orientations

Orientation: 9 bins (for unsigned angles 0 - 180)

- Votes weighted by magnitude
- Bilinear interpolation between cells

Histograms in \(k \times k\) pixel cells
Normalize with respect to surrounding cells

\[ L2 - \text{norm}: v \rightarrow \frac{v}{\sqrt{||v||_2^2 + \epsilon^2}} \]
Original Formulation

\[ X = \]

\# features = \(15 \times 7 \times 9 \times 4 = 3780\)

\# orientations

\# cells

\# normalizations by neighboring cells
$0.16 = w^T x - b$

$\text{sign}(0.16) = 1$

$\implies \text{pedestrian}$
Pedestrian detection with HOG

- Train a pedestrian template using a linear support vector machine
- At test time, convolve feature map with template
- Find local maxima of response
- For multi-scale detection, repeat over multiple levels of a HOG pyramid

Something to think about...

• Sliding window detectors work
  – very well for faces
  – fairly well for cars and pedestrians
  – badly for cats and dogs

• Why are some classes easier than others?
Strengths and Weaknesses of Statistical Template Approach

Strengths

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

Weaknesses

• Not so well for highly deformable objects or “stuff”
• Not robust to occlusion
• Requires lots of training data
Tricks of the trade

• Details in feature computation really matter
  – E.g., normalization in Dalal-Triggs improves detection rate by 27% at fixed false positive rate

• Template size
  – Typical choice is size of smallest detectable object

• “Jittering” to create synthetic positive examples
  – Create slightly rotated, translated, scaled, mirrored versions as extra positive examples

• Bootstrapping to get hard negative examples
  1. Randomly sample negative examples
  2. Train detector
  3. Sample negative examples that score > -1
  4. Repeat until all high-scoring negative examples fit in memory
Things to remember

• Sliding window for search

• Features based on differences of intensity (gradient, wavelet, etc.)
  – Excellent results require careful feature design

• Boosting for feature selection

• Integral images, cascade for speed

• Bootstrapping to deal with many, many negative examples