Rapid Object Detection using a Boosted Cascade of Simple Features

Paul Viola and Michael Jones
CVPR 2001

Brendan Morris

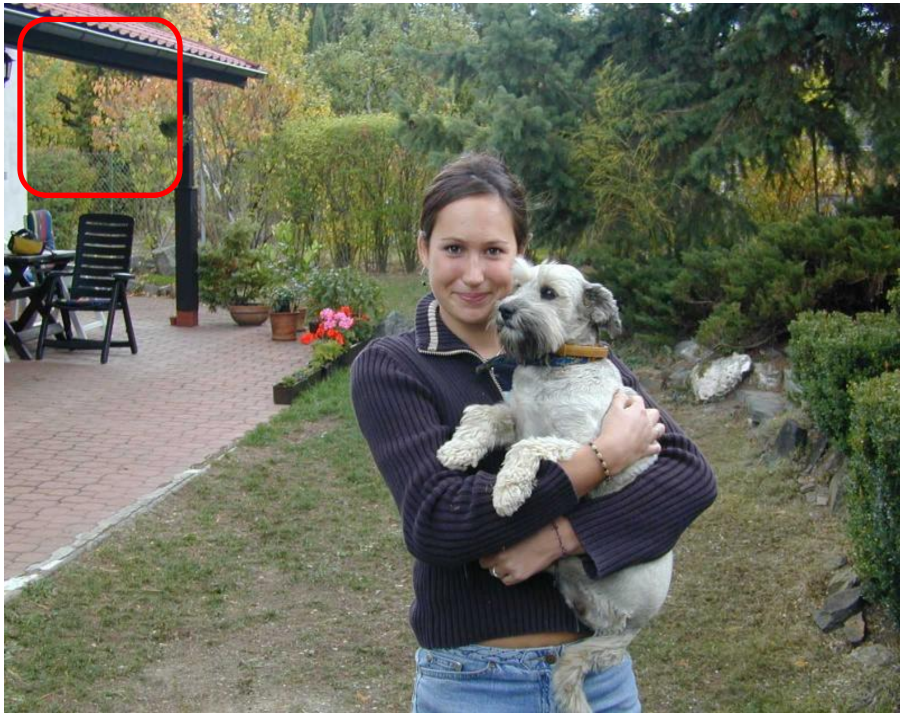
http://www.ee.unlv.edu/~b1morris/ecg782/

Outline

- Motivation
- Contributions
- Integral Image Features
- Boosted Feature Selection
- Attentional Cascade
- Results
- Summary
- Other Object Detection
  - Scale Invariant Feature Transform (SIFT)
  - Histogram of Oriented Gradients (HOG)

Face Detection

- Basic idea: slide a window across image and evaluate a face model at every location



Challenges

- Sliding window detector must evaluate tens of thousands of locations/scale combinations
  - Computationally expensive → worse for complex models
- Faces are rare → usually only a few per image
  - 1M pixel image has 1M candidate face locations (ignoring scale)
  - For computational efficiency, need to minimize time spent evaluating non-face windows
  - False positive rate (mistakenly detecting a face) must be very low ($< 10^{-6}$) otherwise the system will have false faces in every image tested

Outline

- Motivation
- Contributions
- Integral Image Features
- Boosted Feature Selection
- Attentional Cascade
- Results
- Summary
- Other Object Detection
  - Scale Invariant Feature Transform (SIFT)
  - Histogram of Oriented Gradients (HOG)

Contributions of Viola/Jones Detector

- Robust
  - Very high detection rate and low false positive rate
- Real-time
  - Training is slow, but detection very fast

- Key Ideas
  - Integral images for fast feature evaluation
  - Boosting for intelligent feature selection
  - Attentional cascade for fast rejection of non-face windows

Outline

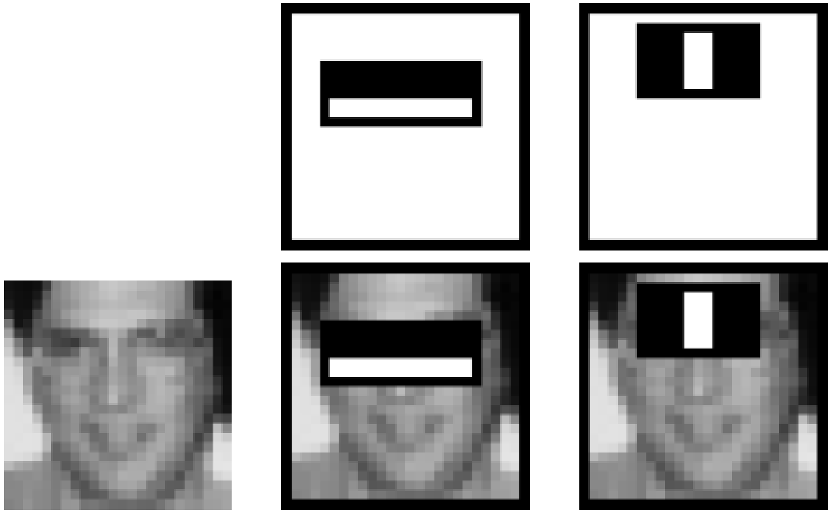
- Motivation
- Contributions
- Integral Image Features
- Boosted Feature Selection
- Attentional Cascade
- Results
- Summary
- Other Object Detection
    - Scale Invariant Feature Transform (SIFT)
    - Histogram of Oriented Gradients (HOG)

Integral Image Features

- Want to use simple features rather than pixels to encode domain knowledge

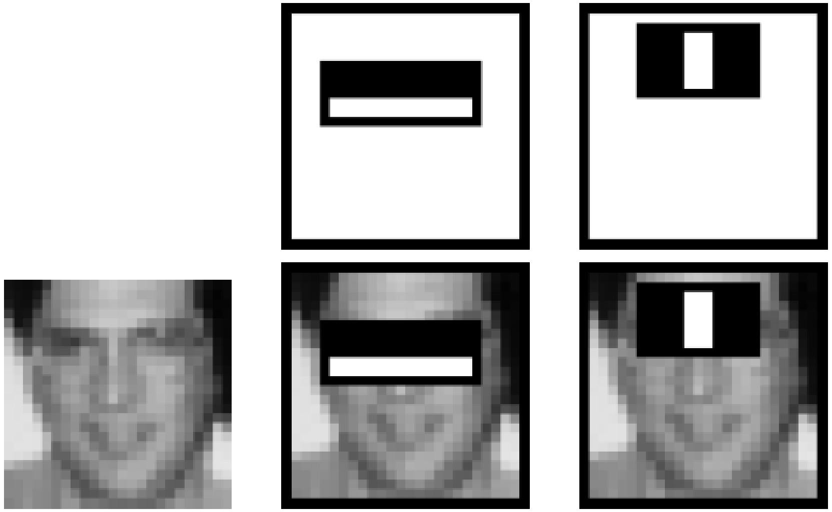
- Haar-like features
  - Encode differences between two, three, or four rectangles
  - Reflect similar properties of a face
    - Eyes darker than upper cheeks
    - Nose lighter than eyes

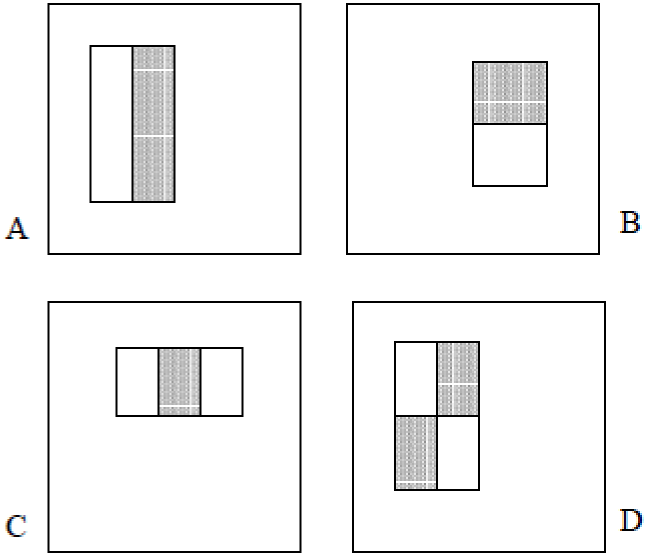
- Believe that these simple intensity differences can encode face structure



Rectangular Features

- Simple feature
  - $val = \sum(pixels\ in\ black\ area) - \sum(pixels\ in\ white\ area)$
- Computed over two-, three-, and four-rectangles
  - Each feature is represented by a specific sub-window location and size
- Over 180k features for a 24×24 image patch
  - Lots of computation





Integral Image

- Need efficient method to compute these rectangle differences
- Define the integral image as the sum of all pixels above and left of pixel (x, y)

$$ii(x,y) = \sum_{x'<x,y'<y} i(x',y')$$

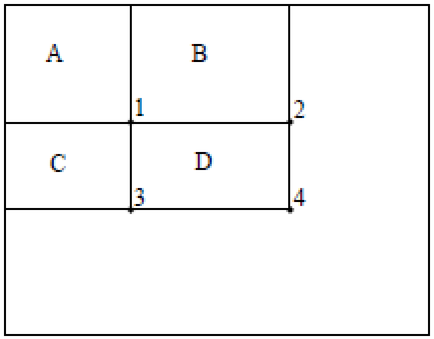
  - Can be computed in a single pass over the image
- Area of a rectangle from four array references
  - $D = ii(4) + ii(1) - ii(2) - ii(3)$
  - Constant time computation

- Integral image

1	0	2
3	2	1
1	4	0

1	1	3
4	6	9
5	11	20

- Rectangle calculation



Outline

- Motivation
- Contributions
- Integral Image Features
- Boosted Feature Selection
- Attentional Cascade
- Results
- Summary
- Other Object Detection
  - Scale Invariant Feature Transform (SIFT)
  - Histogram of Oriented Gradients (HOG)

Boosted Feature Selection

- There are many possible features to compute
  - Individually, each is a “weak” classifier
  - Computationally expensive to compute all
- Not all will be useful for face detection


Relevant feature


Irrelevant feature

- Use AdaBoost algorithm to intelligent select a small subset of features which can be combined to form an effective “strong” classifier

AdaBoost (Adaptive Boost) Algorithm

- Adaptive Boost algorithm
  - Iterative process to build a complex classifier in efficient manner
- Construct a "strong" classifier as a linear combination of weighted "weak" classifiers
  - Adaptive: subsequent weak classifiers are designed to favor misclassifications of previous ones

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

Weak classifier

Strong classifier

Image

Weight

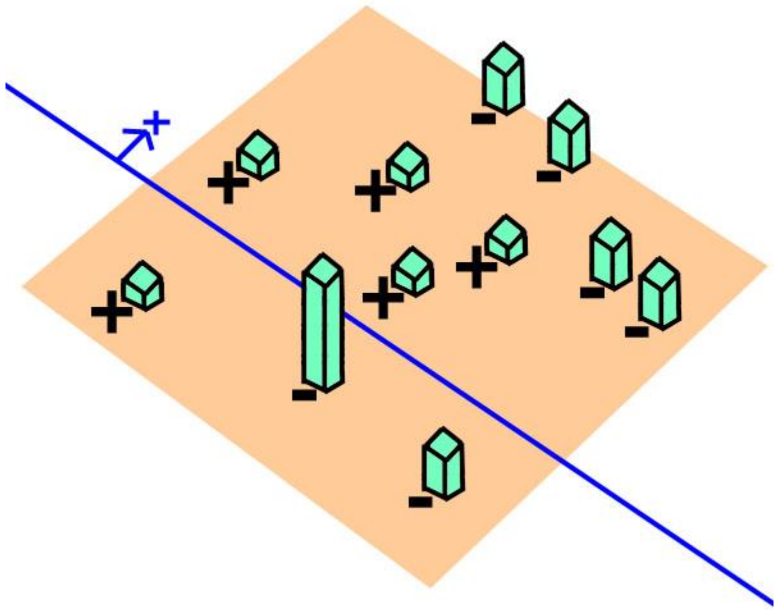
Implemented Algorithm

- 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 $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ 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, $\epsilon_j = \sum_i w_i \, |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_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 $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- 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 $\alpha_t = \log \frac{1}{\beta_t}$

- Initialize
  - All training samples weighted equally
- Repeat for each training round
  - Select most effective weak classifier (single Haar-like feature)
    - Based on weighted eror
  - Update training weights to emphasize incorrectly classified examples
    - Next weak classifier will focus on “harder” examples
- Construct final strong classifier as linear combination of weak learners
  - Weighted according to accuracy

AdaBoost example

- AdaBoost starts with a uniform distribution of “weights” over training examples.
- Select the classifier with the lowest weighted error (i.e. a “weak” classifier)
- Increase the weights on the training examples that were misclassified.
- (Repeat)
- At the end, carefully make a linear combination of the weak classifiers obtained at all iterations.



$$h_{\text{strong}}(\mathbf{x}) = \begin{cases} 1 & \alpha_1 h_1(\mathbf{x}) + \ldots + \alpha_n h_n(\mathbf{x}) \geq \frac{1}{2}(\alpha_1 + \ldots + \alpha_n) \\ 0 & \text{otherwise} \end{cases}$$

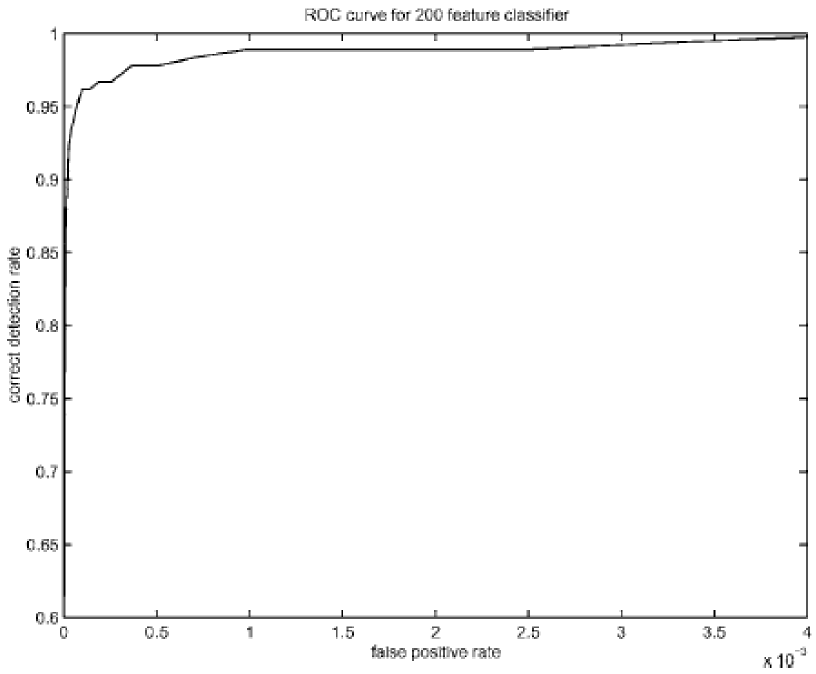
Slide taken from a presentation by Qing Chen, Discover Lab, University of Ottawa

Boosted Face Detector

- Build effective 200-feature classifier

- 95% detection rate
- 0.14×10^{-3} FPR (1 in 14084 windows)
- 0.7 sec / frame

- Not yet real-time



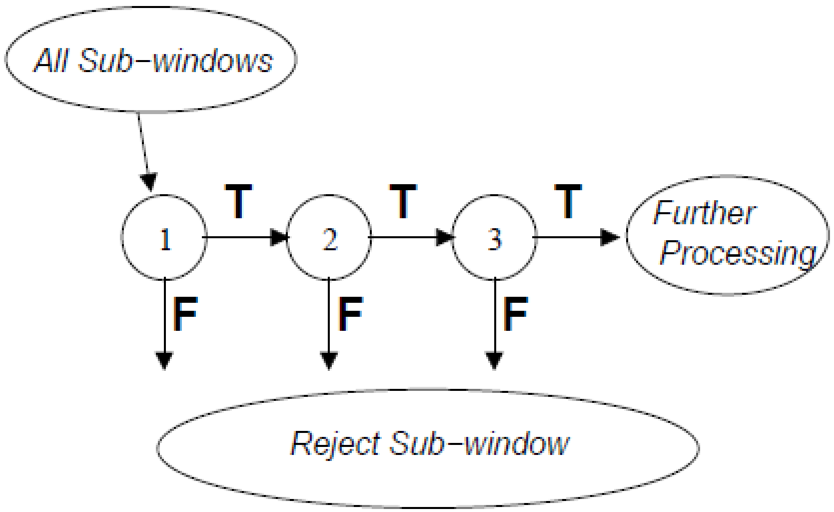
Outline

- Motivation
- Contributions
- Integral Image Features
- Boosted Feature Selection
- Attentional Cascade
- Results
- Summary
- Other Object Detection
  - Scale Invariant Feature Transform (SIFT)
  - Histogram of Oriented Gradients (HOG)

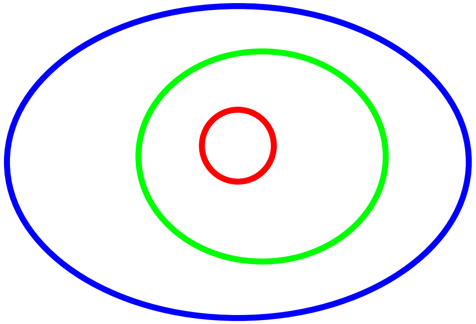
Attentional Cascade

- Boosted strong classifier is still too slow
  - Spends equal amount of time on both face and non-face image patches
  - Need to minimize time spent on non-face patches

- Use cascade structure of gradually more complex classifiers
  - Early stages use only a few features but can filter out many non-face patches
  - Later stages solves “harder” problems
  - Face detected after going through all stages



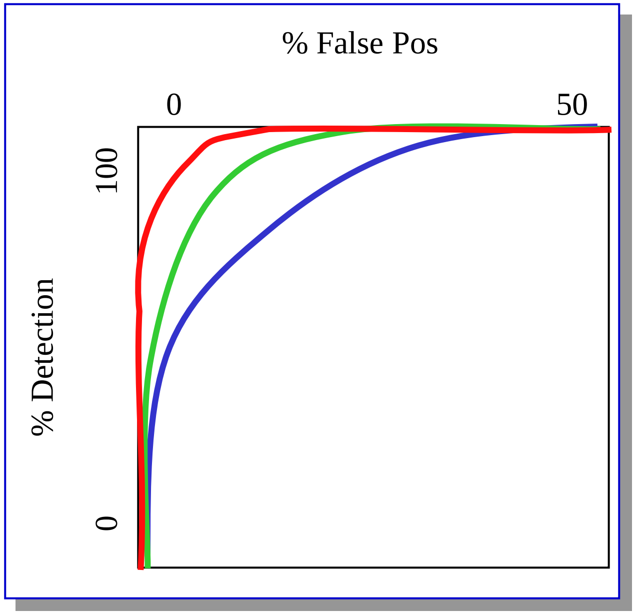
Attentional Cascade

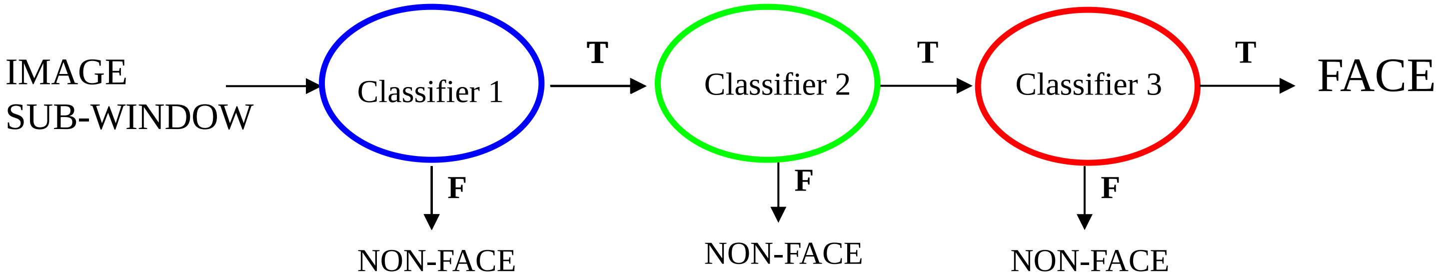


- Much fewer features computed per sub-window
  - Dramatic speed-up in computation
- See IJCV paper for details
  - #stages and #features/stage

- Chain classifiers that are progressively more complex and have lower false positive rates

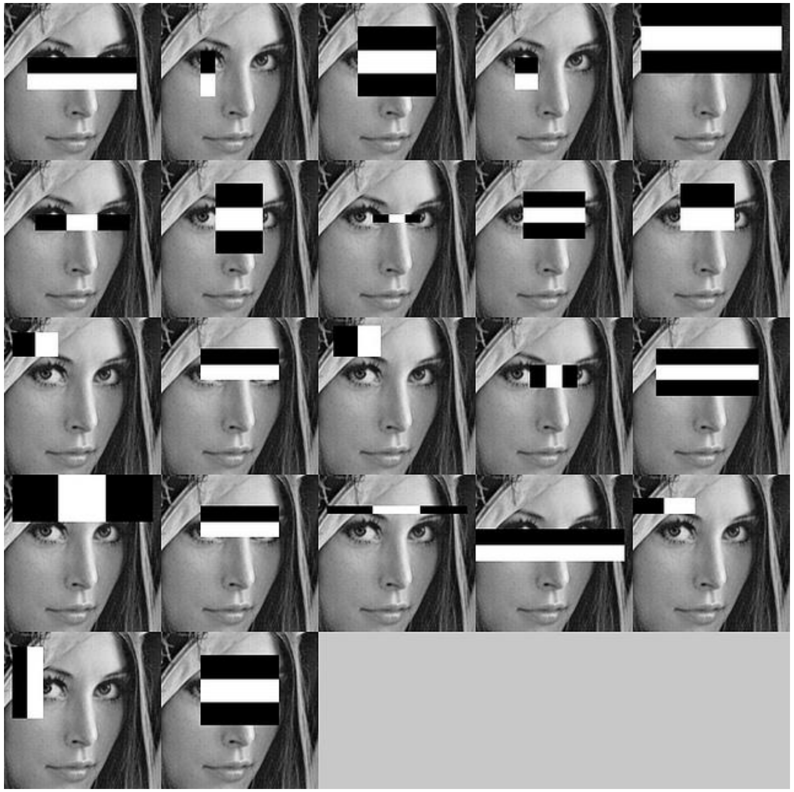
ROC

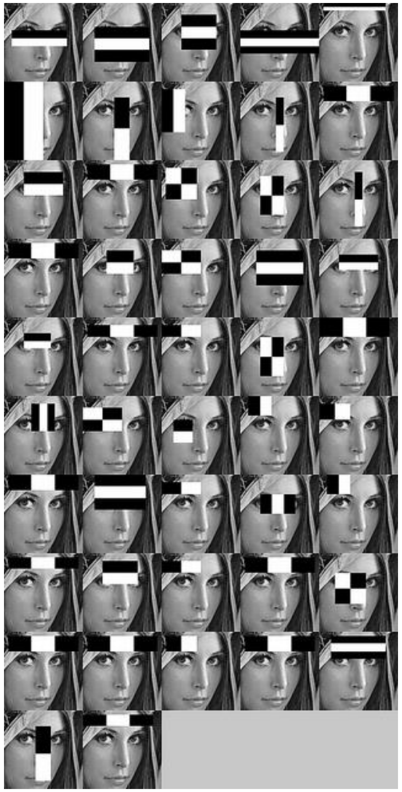




Face Cascade Example

Step 1 … Step 4 … Step N





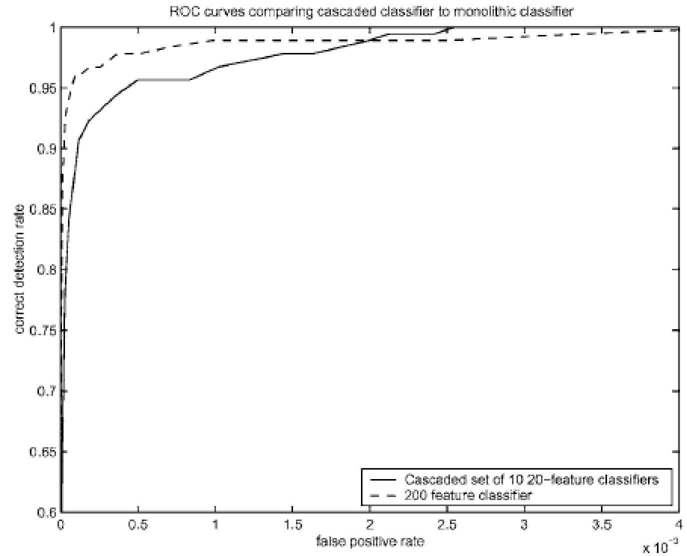
- Visualized
  - https://vimeo.com/12774628

Outline

- Motivation
- Contributions
- Integral Image Features
- Boosted Feature Selection
- Attentional Cascade
- Results
- Summary
- Other Object Detection
    - Scale Invariant Feature Transform (SIFT)
    - Histogram of Oriented Gradients (HOG)

Results

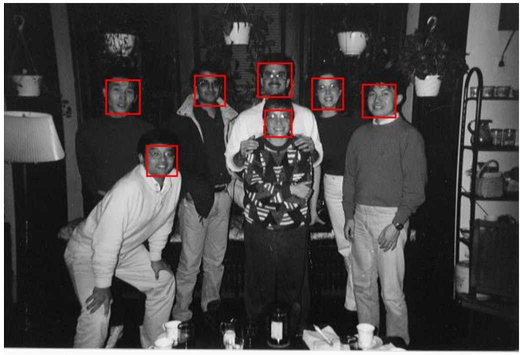
- Training data
  - 4916 labeled faces
  - 9544 non-face images → 350M non-face sub-windows
  - 24 × 24 pixel size
- Cascade layout
  - 38 layer cascade classifier
  - 6061 total features
  - S1: 1, S2: 10, S3: 25, S4: 25, S5: 50, …
- Evaluation
  - Avg. 10/6061 features evaluated per sub-window
  - 0.67 sec/image
    - 700 MHz PIII
    - 384 × 388 image size
    - With various scale
  - Much faster than existing algorithms

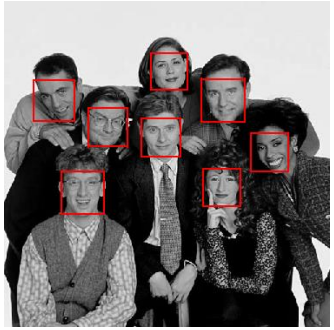


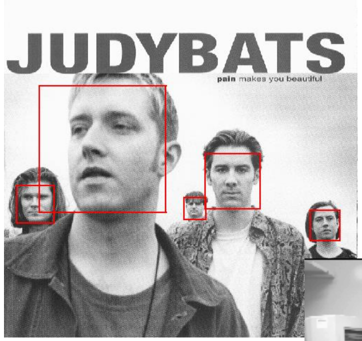
Similar performance between cascade and big classifier, but cascade is ~10x faster

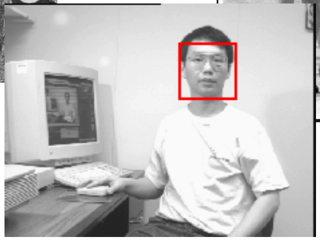
MIT+CMU Face Test

- Real-world face test set
  - 130 images with 507 frontal faces

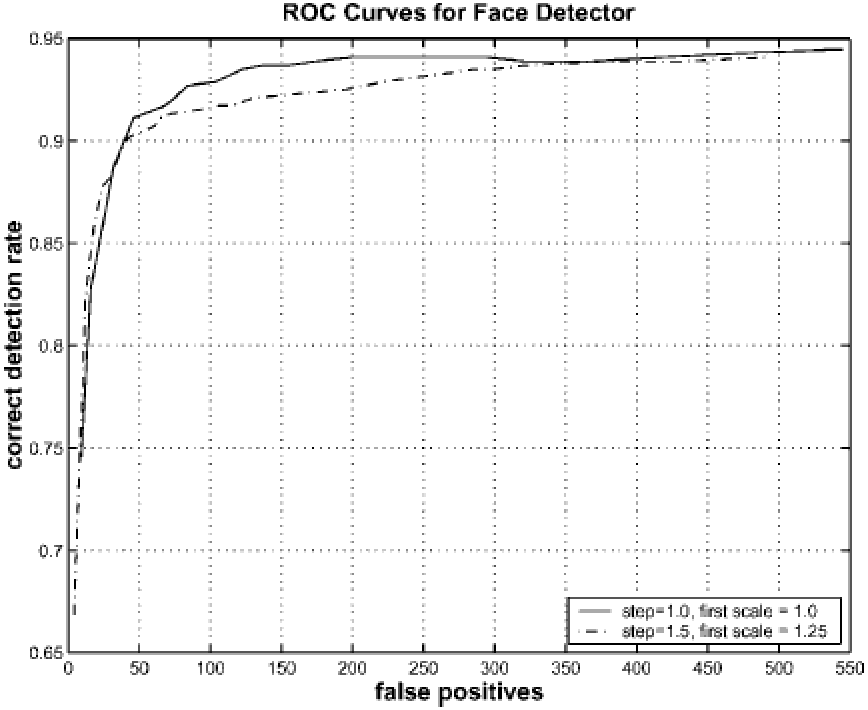








	False detections							
Detector	10	31	50	65	78	95	167	422
Viola-Jones	76.1%	88.4%	91.4%	92.0%	92.1%	92.9%	93.9%	94.1%
Viola-Jones (voting)	81.1%	89.7%	92.1%	93.1%	93.1%	93.2%	93.7%	–
Rowley-Baluja-Kanade	83.2%	86.0%	–	–	–	89.2%	90.1%	89.9%
Schneiderman-Kanade	–	–	–	94.4%	–	–	–	–
Roth-Yang-Ahuja	–	–	–	–	(94.8%)	–	–	–



Outline

- Motivation
- Contributions
- Integral Image Features
- Boosted Feature Selection
- Attentional Cascade
- Results
- Summary
- Other Object Detection
    - Scale Invariant Feature Transform (SIFT)
    - Histogram of Oriented Gradients (HOG)

Summary

- Pros
  - Extremely fast feature computation
  - Efficient feature selection
  - Scale and location invariant detector
    - Scale features not image (e.g. image pyramid)
  - Generic detection scheme → can train other objects
- Cons
  - Detector only works on frontal faces ($< 45°$)
  - Sensitive to lighting conditions
  - Multiple detections to same face due to overlapping sub-windows

Outline

- Motivation
- Contributions
- Integral Image Features
- Boosted Feature Selection
- Attentional Cascade
- Results
- Summary
- Other Object Detection
    - Scale Invariant Feature Transform (SIFT)
    - Histogram of Oriented Gradients (HOG)

Quantifying Performance

- Confusion matrix-based metrics
  - Binary {1,0} classification tasks

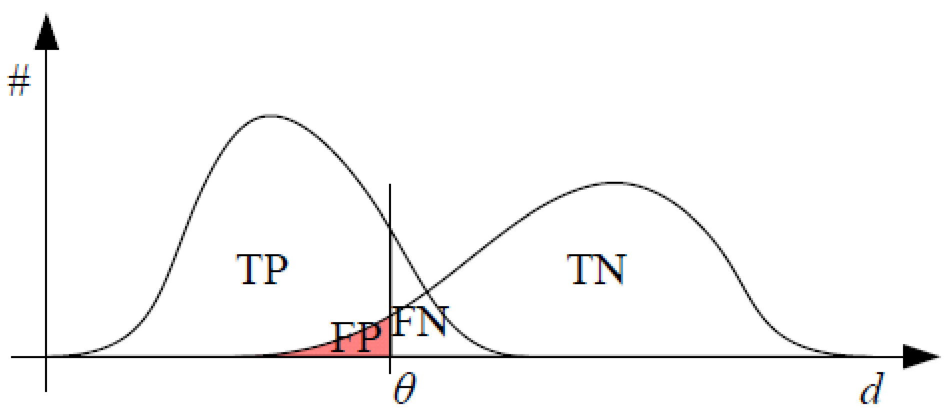
		actual value		
		p	n	total
predicted outcome	p'	TP	FP	P'
	n'	FN	TN	N'
	total	P	N	

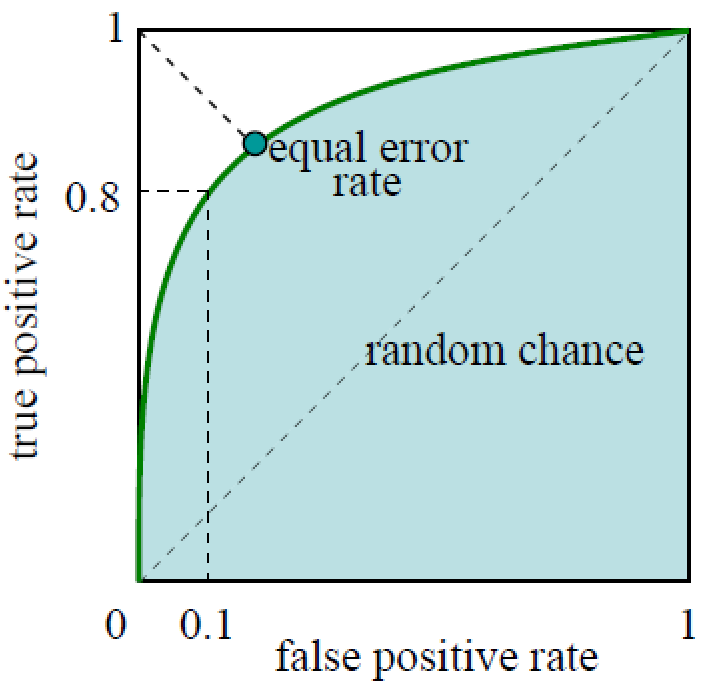
- True positives (TP) - # correct matches
- False negatives (FN) - # of missed matches
- False positives (FP) - # of incorrect matches
- True negatives (TN) - # of non-matches that are correctly rejected

- A wide range of metrics can be defined
- True positive rate (TPR) (sensitivity)
  - $TPR = \frac{TP}{TP+FN} = \frac{TP}{P}$
  - Document retrieval → recall – fraction of relevant documents found
- False positive rate (FPR)
  - $FPR = \frac{FP}{FP+TN} = \frac{FP}{N}$
- Positive predicted value (PPV)
  - $PPV = \frac{TP}{TP+FP} = \frac{TP}{P'}$
  - Document retrieval → precision – number of relevant documents are returned
- Accuracy (ACC)
  - $ACC = \frac{TP+TN}{P+N}$

Receiver Operating Characteristic (ROC)

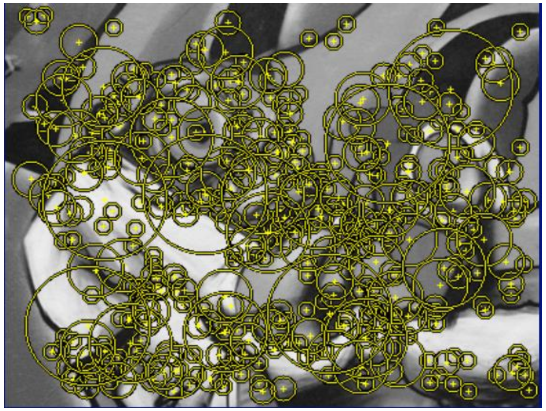
- Evaluate matching performance based on threshold
  - Examine all thresholds θ to map out performance curve
- Best performance in upper left corner
  - Area under the curve (AUC) is a ROC performance metric





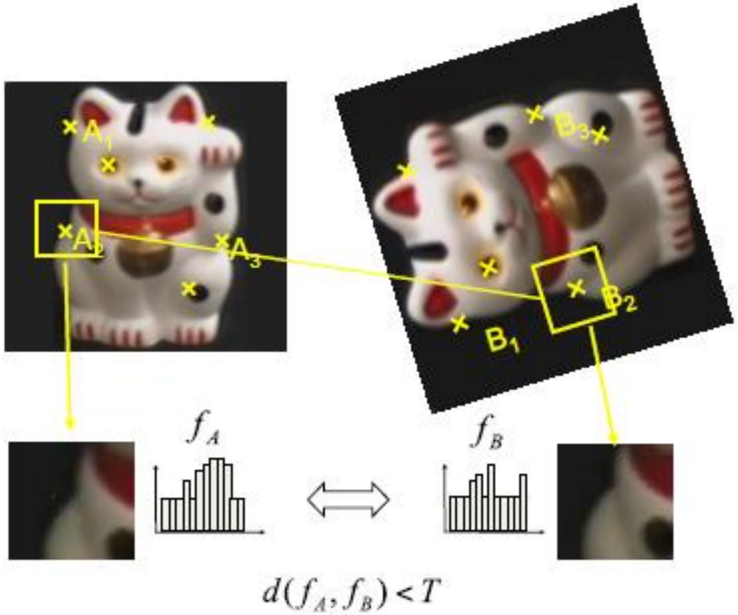
Scale Invariant Feature Transform (SIFT)

- One of the most popular feature descriptors [Lowe 2004]
  - Many variants have been developed



- Descriptor is invariant to uniform scaling, orientation, and partially invariant to affine distortion and illumination changes

- Used for matching between images



K. Grauman, B. Leibe

SIFT Steps I

- Identify keypoints
  - Use difference of Gaussians for scale space representation
  - Identify "stable" regions
    - Location, scale, orientation
- Compute gradient 16 × 16 grid around keypoint
  - Keep orientation and down-weight magnitude by a Gaussian fall off function
    - Avoid sudden changes in descriptor with small position changes
    - Give less emphasis to gradients far from center
- Form a gradient orientation histogram in each 4 × 4 quadrant
  - 8 bin orientations
  - Trilinear interpolation of gradient magnitude to neighboring orientation bins
  - Gives 4 pixel shift robustness and orientation invariance

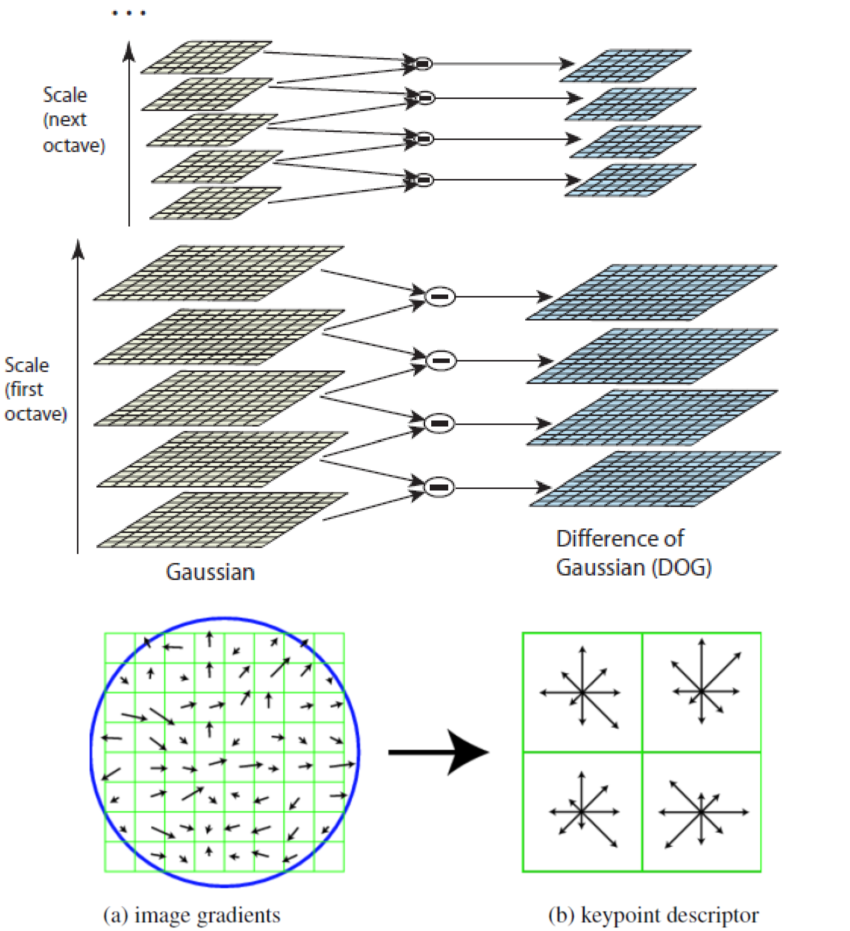
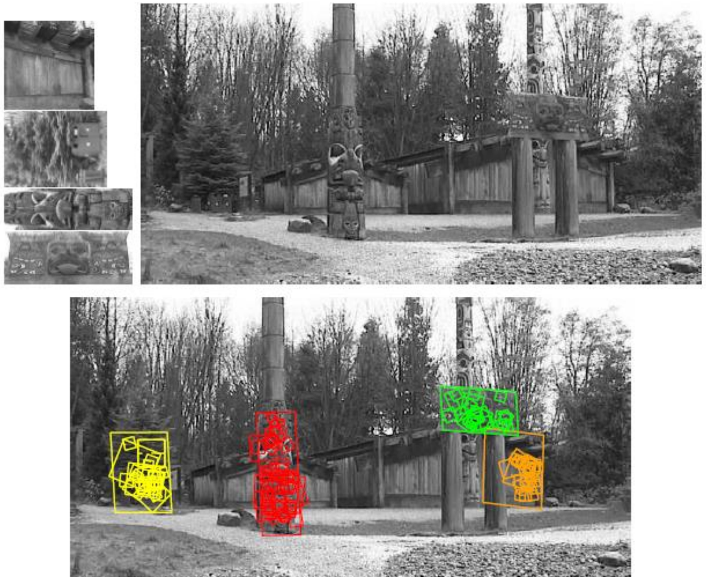


Figure 4.18 A schematic representation of Lowe's (2004) scale invariant feature transform (SIFT): (a) Gradient orientations and magnitudes are computed at each pixel and weighted by a Gaussian fall-off function (blue circle). (b) A weighted gradient orientation histogram is then computed in each subregion, using trilinear interpolation. While this figure shows an 8×8 pixel patch and a 2×2 descriptor array, Lowe's actual implementation uses 16×16 patches and a 4×4 array of eight-bin histograms.

SIFT Steps II

- Final descriptor is $4 \times 4 \times 8 = 128$ dimension vector
  - Normalize vector to unit length for contrast/gain invariance
  - Values clipped to 0.2 and renormalized to remove emphasis of large gradients (orientation is most important)
- Descriptor used for object recognition
  - Match keypoints
  - Hough transform used to “vote” for 2D location, scale, orientation
  - Estimate affine transformation

Figure 12: The training images for two objects are shown on the left. These can be recognized in a cluttered image with extensive occlusion, shown in the middle. The results of recognition are shown on the right. A parallelogram is drawn around each recognized object showing the boundaries of the original training image under the affine transformation solved for during recognition. Smaller squares indicate the keypoints that were used for recognition.

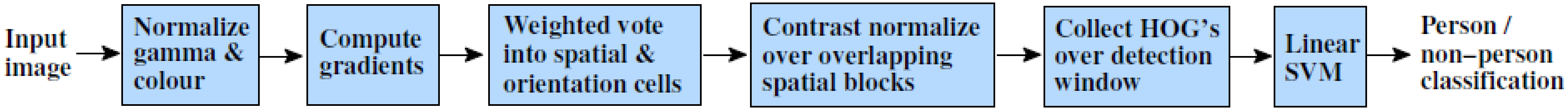


Other SIFT Variants

- Speeded up robust features (SURF) [Bay 2008]
  - Faster computation by using integral images (Szeliski 3.2.3 and later for object detection)
  - Popularized because it is free for non-commercial use
    - SIFT is patented
- OpenCV implements many
  - FAST
  - ORB
  - BRISK
  - FREAK
- OpenCV is a standard in vision research community
  - Emphasis on fast descriptors for real-time applications

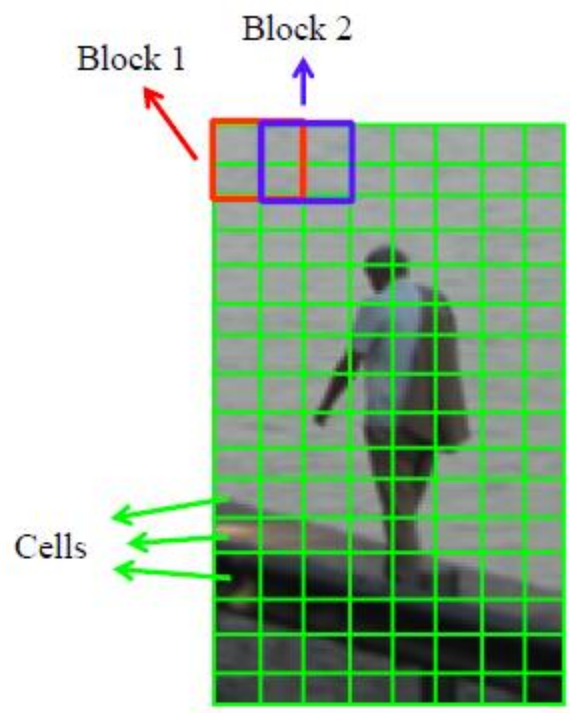
Histogram of Oriented Gradients

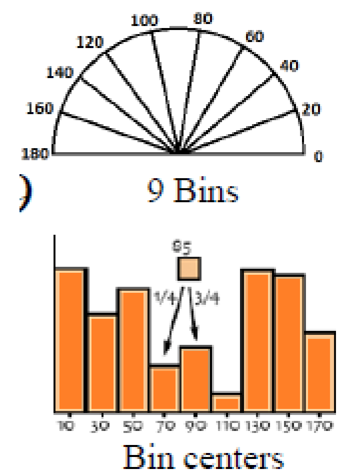
- Want descriptor for a full object rather than keypoints
  - Geared toward detection/classification rather than matching
- Designed by Dalal and Triggs for pedestrian detection
  - Must handle various pose, variable appearance, complex background, and unconstrained illumination



HOG Steps I

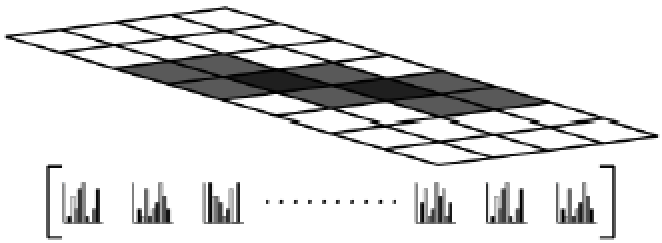
- Compute horizontal and vertical gradients (with no smoothing)
- Compute gradient orientation and magnitude
- Divide image into 16 × 16 blocks of 50% overlap
  - For 64 × 128 image → 7 × 15 = 105 blocks
  - Each block consists of 2 × 2 cells of size 8 × 8 pixels
- Histogram of gradient orientation of cells
  - 9 bins between 0-180 degrees
  - Bin vote is gradient magnitude
  - Interpolate vote between bins

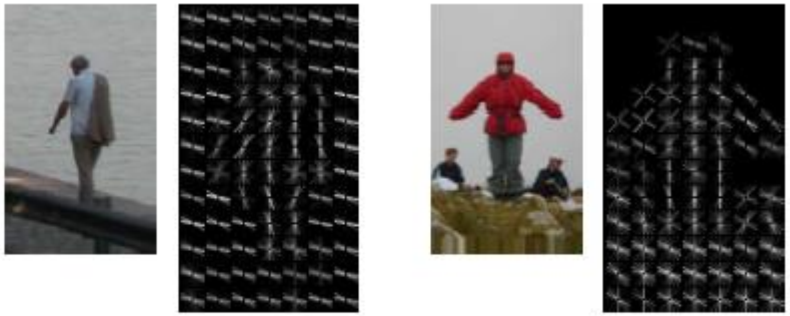


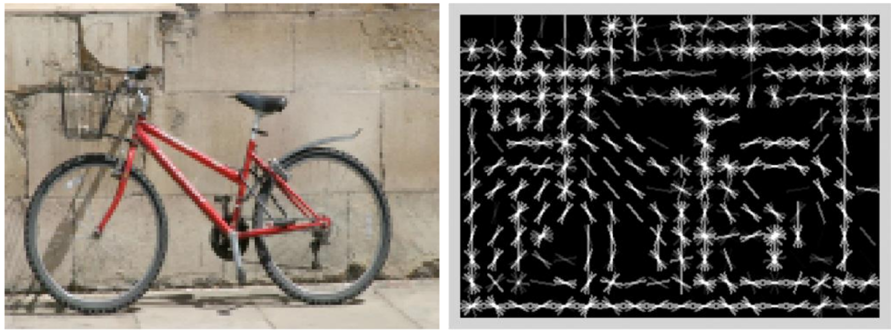


HOG Steps II

- Group cells into large blocks and normalize
- Concatenate histograms into large feature vector
  - #features = (15*7)*9*4 = 3780
    - 15*7 blocks
    - 9 orientation bins
    - 4 cells per block
- Use SVM to train classifier
  - Unique feature signature for different objects
  - Computed on dense grids at single scale and without orientation alignment

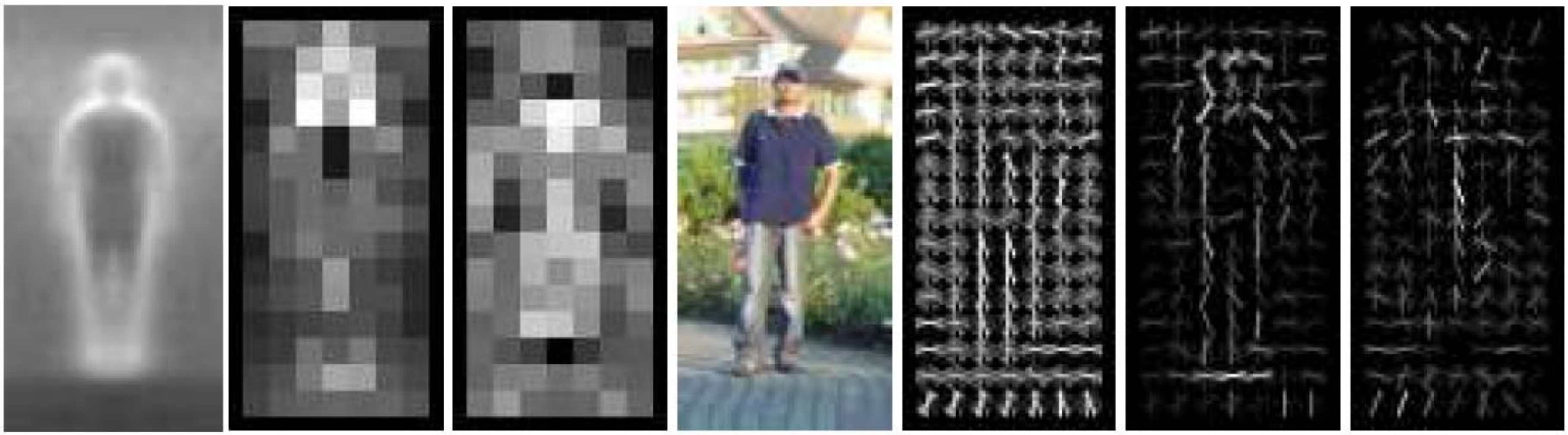






HOG Overview

- Note: emphasizes contours/silhouette of object so robust to illumination



(a) (b) (c) (d) (e) (f) (g)

Figure 6. Our HOG detectors cue mainly on silhouette contours (especially the head, shoulders and feet). The most active blocks are centred on the image background just *outside* the contour. (a) The average gradient image over the training examples. (b) Each "pixel" shows the maximum positive SVM weight in the block centred on the pixel. (c) Likewise for the negative SVM weights. (d) A test image. (e) It's computed R-HOG descriptor. (f,g) The R-HOG descriptor weighted by respectively the positive and the negative SVM weights.

SIFT vs HOG

Powerful orientation-based descriptors
Robust to changes in brightness

- SIFT
  - 128 dimensional vector
  - 16x16 window
  - 4x4 sub-window (16 total)
  - 8 bin histogram (360 degree)
  - Computed at sparse, scale-invariant keypoints of image
  - Rotated and aligned for orientation
  - Good for matching
- HOG
  - 3780 dimensional vector
  - 64x128 window
  - 16x16 blocks with overlap
  - Each block in 2x2 cells of 8x8 pixels
  - 9 bin histogram (180 degree)
  - Appears similar in spirit to SIFT
  - Computed at dense grid at single scale
  - No orientation alignment
  - Good for detection

Thank You

- Questions?

References

- Reading
  - P. Viola and M. Jones, Rapid object detection using a boosted cascade of simple features, CVPR 2001
  - P. Viola and M. Jones, Robust real-time face detection, IJCV 57(2), 2004
  - Dalal and Triggs, "Histogram of Oriented Gradients for Human Detection", CVPR 2005
  - Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", IJCV 60(2) 1999
- Code
  - OpenCV has implementations