

EE482/682 DSP APPLICATIONS

CLASSICAL DETECTION

OVERVIEW

- Recognition Overview
 - Instance Recognition, Image Classification, Object Detection, Semantic Segmentation [Szeliski]
 - Performance Characterization
- Classical Detection
 - Viola and Jones
 - Histogram of Oriented Gradients
 - Deformable Parts Model

RECOGNITION OVERVIEW

SZELISKI 2E CHAPTER6

RECOGNITION OVERVIEW

- Undergone largest changes and fastest developments in the last decade
 - Availability of larger labeled datasets
 - Breakthroughs in deep learning
- Historically, recognition was a “high-level task” built on top of lower-level components (e.g. feature detection and matching)
- With deep learning, there is little distinction between high- and low-level tasks → end-to-end learning

RECOGNITION TASKS

- Instance recognition – find specific objects (exemplars, e.g. a stop-sign)
- Class/category recognition – recognize members of highly variable categories (e.g. any dog)
- Object detection – classify and localize objects
- Segmentation – pixel-level annotation of images into objects/class

INSTANCE RECOGNITION I

- Re-recognize a known 2D/3D rigid object (exemplar)
- Potentially with novel viewpoint, cluttered background, and partial occlusion

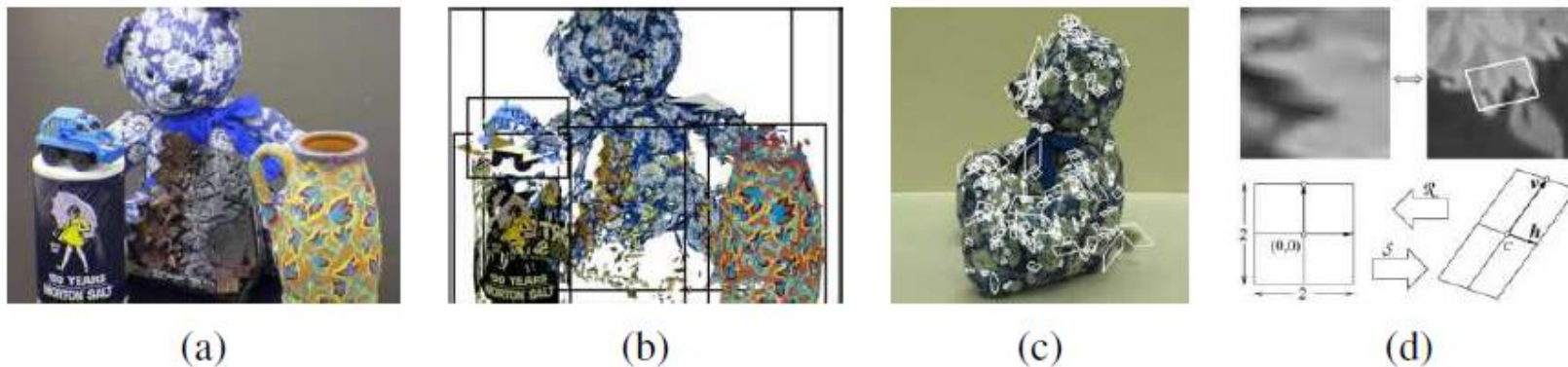


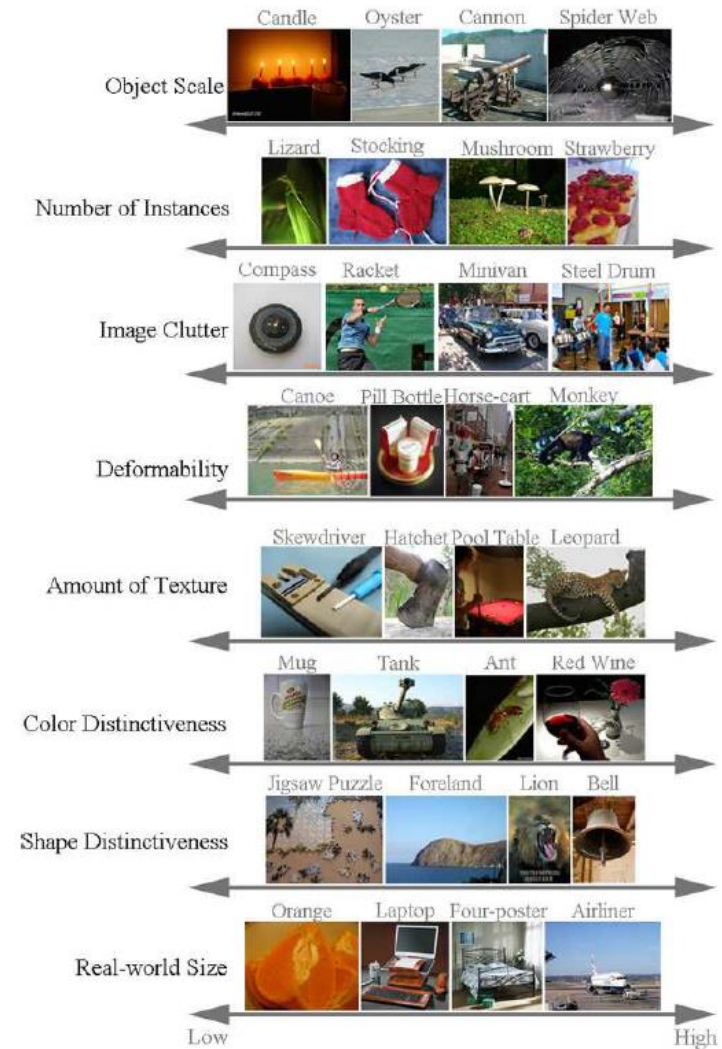
Figure 6.3 3D object recognition with affine regions (Rothganger, Lazebnik et al. 2006) © 2006 Springer: (a) sample input image; (b) five of the recognized (reprojected) objects along with their bounding boxes; (c) a few of the local affine regions; (d) local affine region (patch) reprojected into a canonical (square) frame, along with its geometric affine transformations.

INSTANCE RECOGNITION II

- General approach:
 - Find distinctive features while dealing with local appearance variation
 - Check for co-occurrence and relative positions (e.g. affine transformation)
- More challenging version: instance retrieval (content-based image retrieval) where the number of images to search is very large

IMAGE CLASSIFICATION

- Also known as category/class recognition
 - Must recognize members of highly variable categories
- Much more challenging than instance recognition
 - Same challenges but without known object
- Extensively studied area of CV
 - Where CNNs have dominated
- Note this is whole image classification



CLASSICAL APPROACHES: BOW

- Bag-of-words (features) – simple approach based co-occurrence of collected features
 - Detect features/keypoints
 - Describe keypoints = words
 - Compute histogram (distribution) of words
 - Compare histogram to database for matching
- Note: no geometric verification since not applicable to general objects

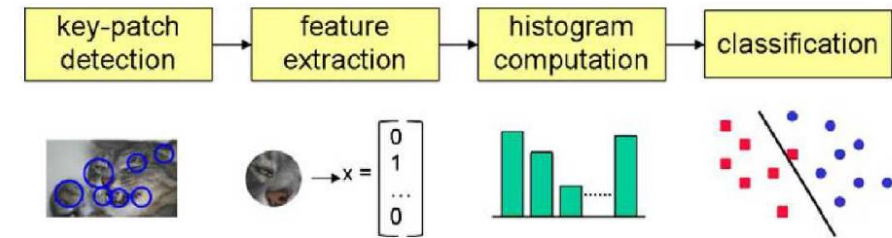


Figure 6.6 A typical processing pipeline for a bag-of-words category recognition system (Csurka, Dance et al. 2006) © 2007 Springer. Features are first extracted at keypoints and then quantized to get a distribution (histogram) over the learned visual words (feature cluster centers). The feature distribution histogram is used to learn a decision surface using a classification algorithm, such as a support vector machine.

CLASSICAL APPROACHES: PARTS

- Approach to find constituent parts and measuring geometric relationships
 - Spring-like connections between subparts that have structure but allow variation
- Basic idea is to have an energy minimization function for subpart arrangements
- Common (graph) structures/topologies include threes and stars for efficiency
- Popular model: Deformable Part Model (DPM) of Felzenszwalb
 - Star model on HOG parts

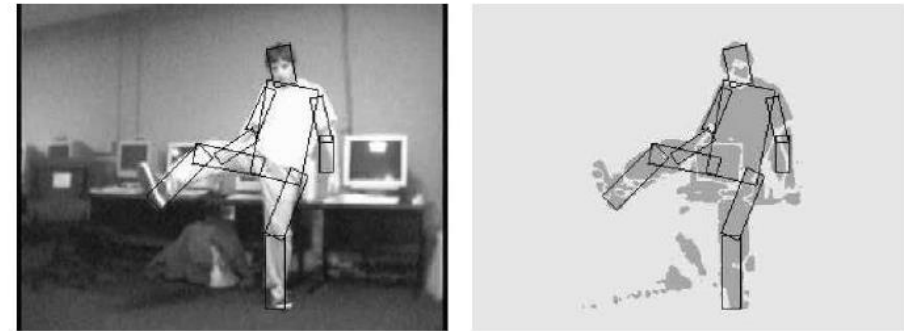


Figure 6.7 Using pictorial structures to locate and track a person (Felzenszwalb and Huttenlocher 2005) © 2005 Springer. The structure consists of articulated rectangular body parts (torso, head, and limbs) connected in a tree topology that encodes relative part positions and orientations. To fit a pictorial structure model, a binary silhouette image is first computed using background subtraction.

CLASSICAL APPROACHES: CONTEXT/SCENE

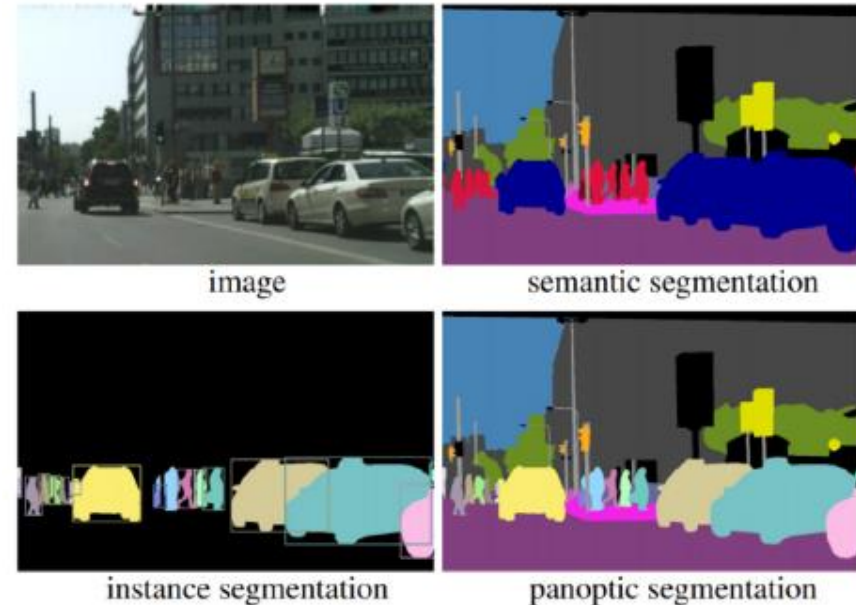
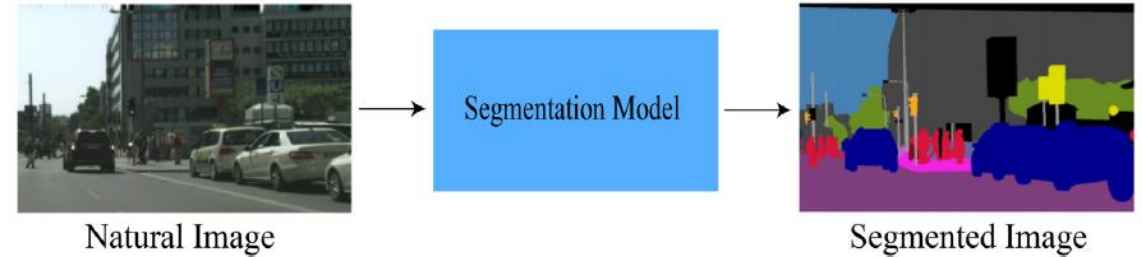
- Previous approaches were object-centric which limits recognition
 - Scene context is very important for disambiguation (e.g. lemon vs. tennis ball)
- Context models combine objects into scenes
 - Number of constituent objects is not known a priori
- The idea of context has been important for deep techniques



Figure 6.8 The importance of context (images courtesy of Antonio Torralba). Can you name all of the objects in images (a–b), especially those that are circled in (c–d). Look carefully at the circled objects. Did you notice that they all have the same shape (after being rotated), as shown in column (e)?

SEGMENTATION

- CV task of segregating an image into multiple regions according to different properties of pixels (e.g. color, intensity, texture)
 - Typically a low-level task that relies on spatial information (neighborhood)
 - Pixel-level class label
- Semantic segmentation – associate a class label for every pixel in an image
- Instance segmentation – mask (segment) each instance of an object in an image independently
- Panoptic segmentation – combination of semantic segmentation and instance segmentation
 - Label both class and separate instances (detection)



QUANTIFYING PERFORMANCE

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

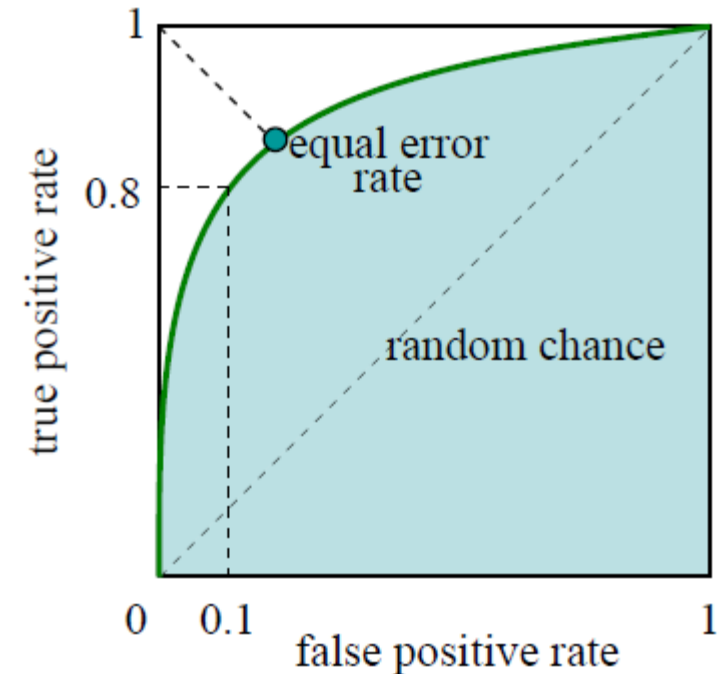
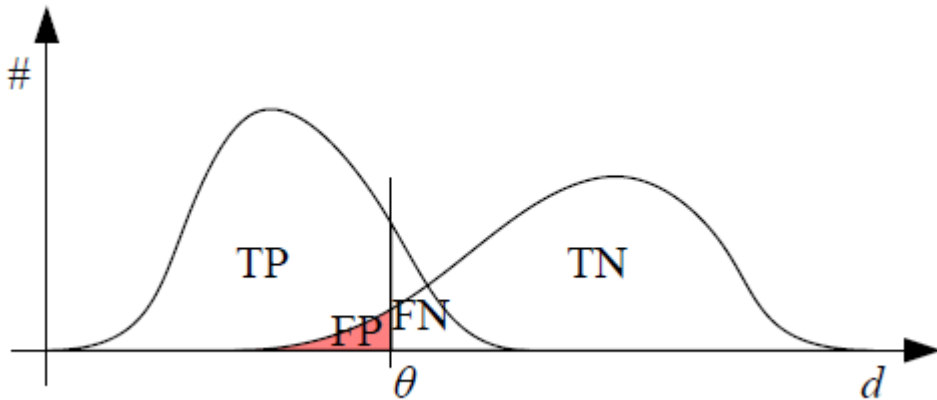
		actual value		
		p	n	total
predicted outcome	p'	TP	FP	P'
	n'	FN	TN	N'
	total	P	N	

- 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



VIOLA AND JONES DETECTOR

CVPR2001

OUTLINE

- Motivation
- Contributions
- Integral Image Features
- Boosted Feature Selection
- Attentional Cascade
- Results
- Summary

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

CONTRIBUTIONS

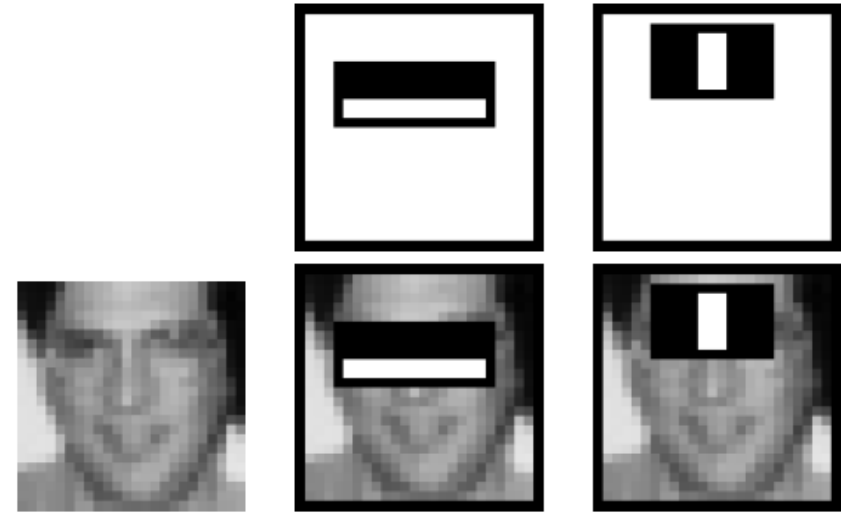
- Robust
 - Very high detection rate and low false positive rate
- Real-time
 - Training is slow, but detection very fast
- Key Ideas
 - Integral images for fast feature evaluation
 - Boosting for intelligent feature selection
 - Attentional cascade for fast rejection of non-face windows

OUTLINE

- Motivation
- Contributions
- Integral Image Features
- Boosted Feature Selection
- Attentional Cascade
- Results
- Summary

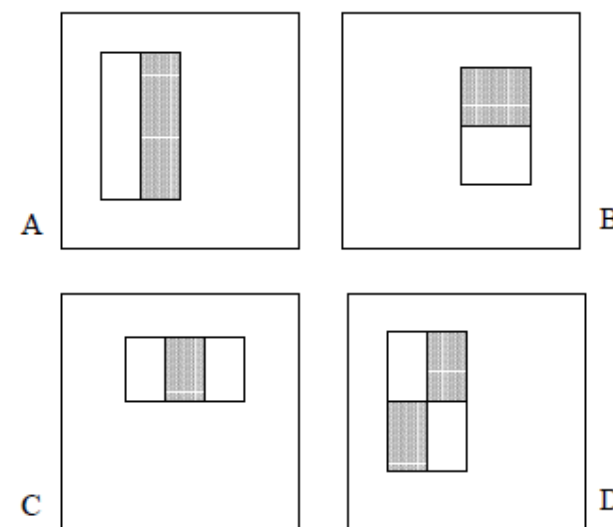
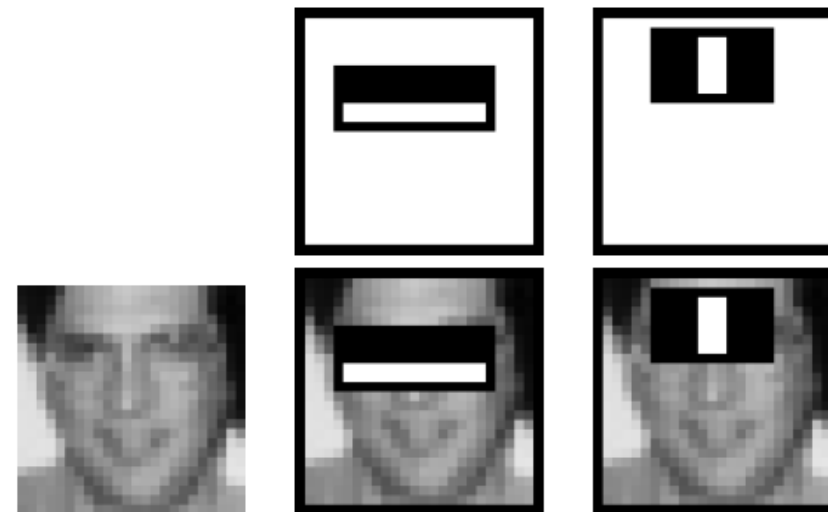
INTEGRAL IMAGE FEATURES

- 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
 - $val = \sum(\text{pixels in black area}) - \sum(\text{pixels in white area})$
- 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



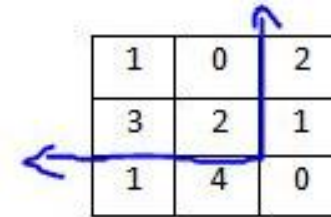
INTEGRAL IMAGE

- 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

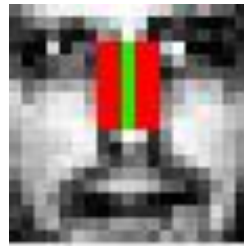
A	B	
	1	2
C	D	
	3	4

OUTLINE

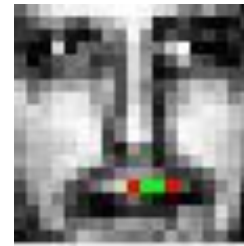
- Motivation
- Contributions
- Integral Image Features
- Boosted Feature Selection
- Attentional Cascade
- Results
- Summary

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 intelligently 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 an 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

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

Strong classifier Image Weight Weak classifier

A diagram illustrating the components of the equation. Four vertical arrows point upwards from labels at the bottom to terms in the equation above. The first arrow points from 'Strong classifier' to 'F(x)'. The second arrow points from 'Image' to 'f_1(x)'. The third arrow points from 'Weight' to 'alpha_1'. The fourth arrow points from 'Weak classifier' to 'f_1(x)'.

↑ ↑ ↑ ↑

IMPLEMENTED ALGORITHM

- Given example images $(x_1, y_1), \dots, (x_n, y_n)$ 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 $t = 1, \dots, T$:

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:

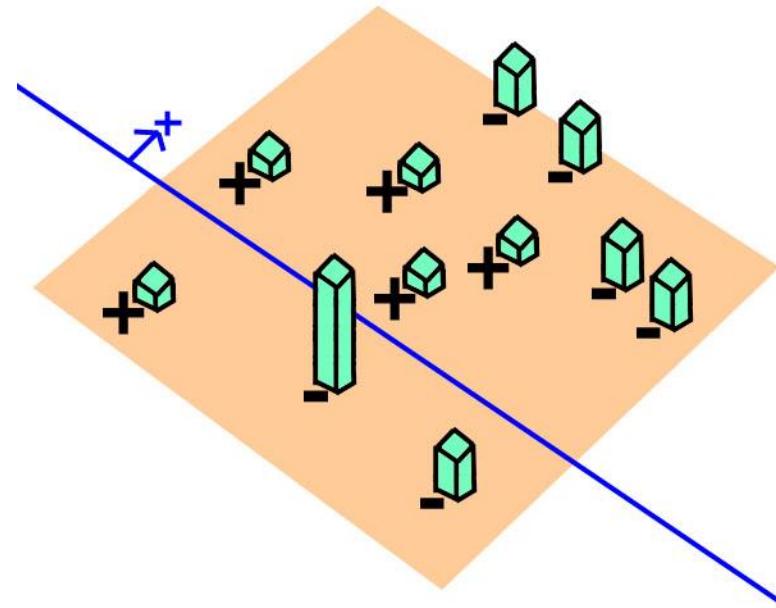
$$h(x) = \begin{cases} 1 & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

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 error
 - 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

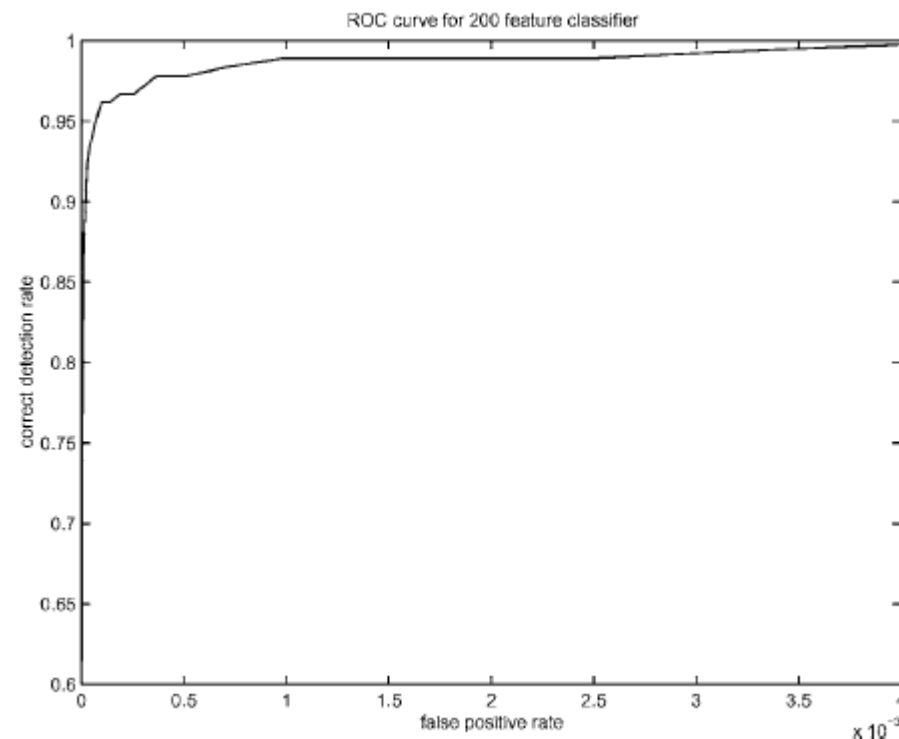
- AdaBoost starts with a uniform distribution of “weights” over training examples.
- Select the classifier with the lowest weighted error (i.e. a “weak” classifier)
- Increase the weights on the training examples that were misclassified.
- (Repeat)
- At the end, carefully make a linear combination of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



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

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

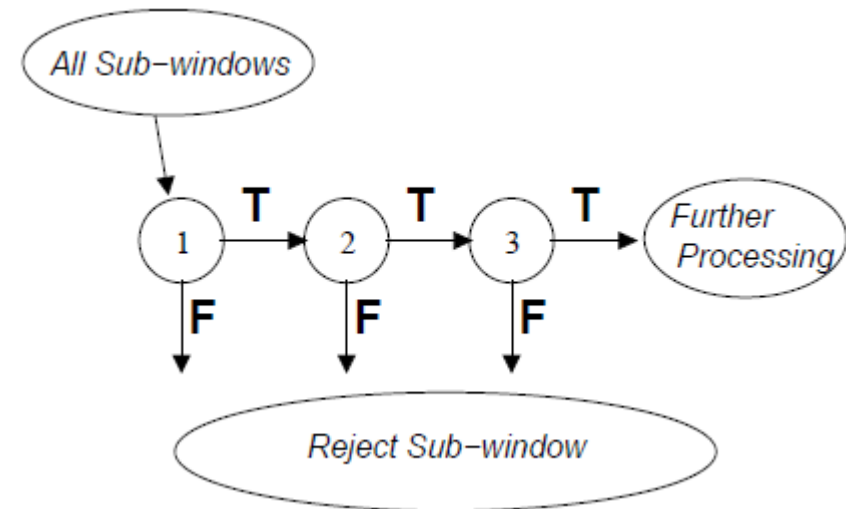


OUTLINE

- Motivation
- Contributions
- Integral Image Features
- Boosted Feature Selection
- Attentional Cascade
- Results
- Summary

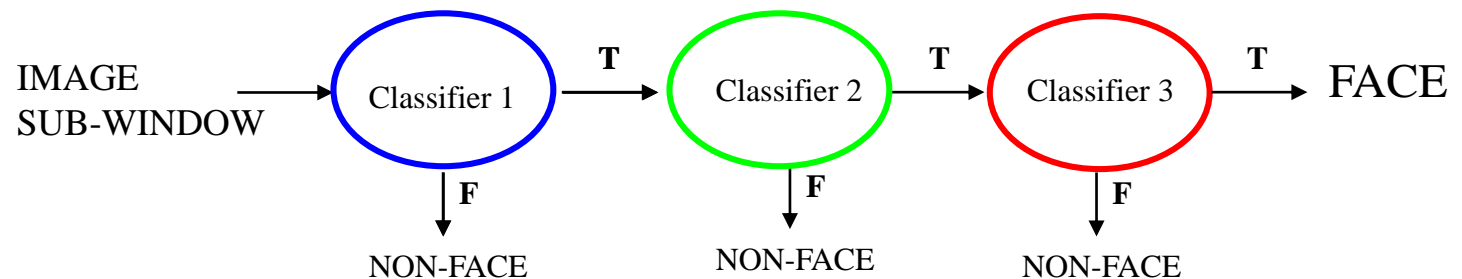
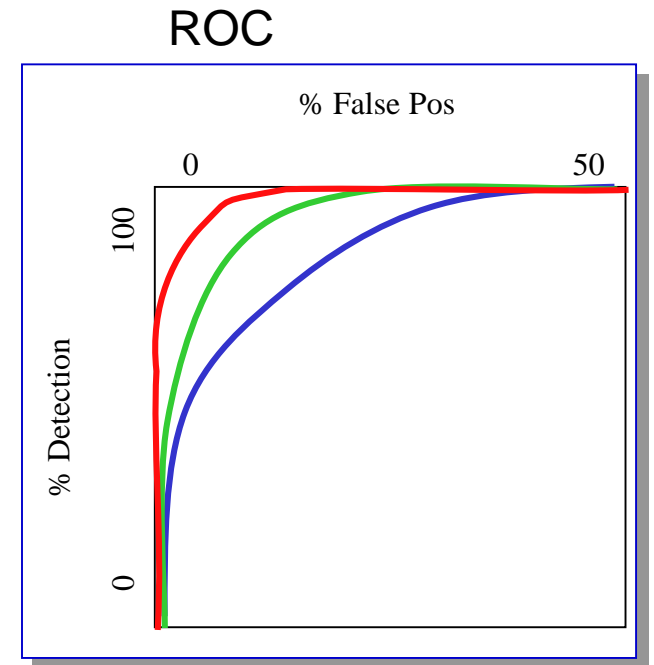
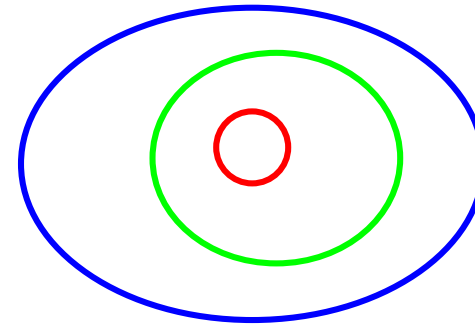
ATTENTIONAL CASCADE

- 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



FACE CASCADE EXAMPLE

Step 1



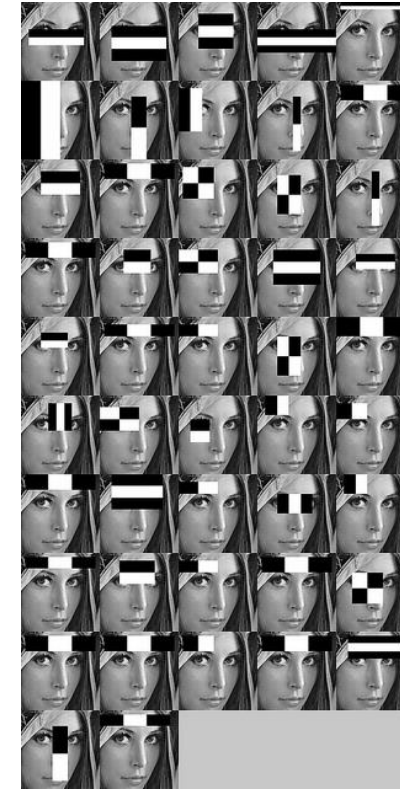
...

Step 4



...

Step N



■ Visualized

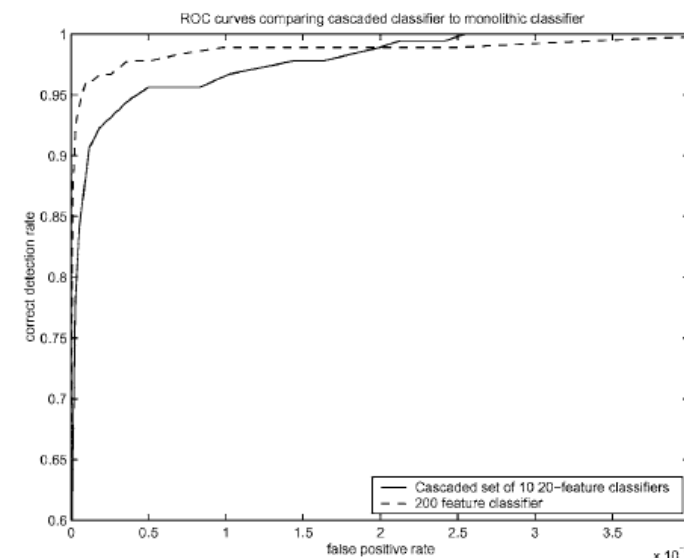
■ <https://vimeo.com/12774628>

OUTLINE

- Motivation
- Contributions
- Integral Image Features
- Boosted Feature Selection
- Attentional Cascade
- Results
- Summary

RESULTS

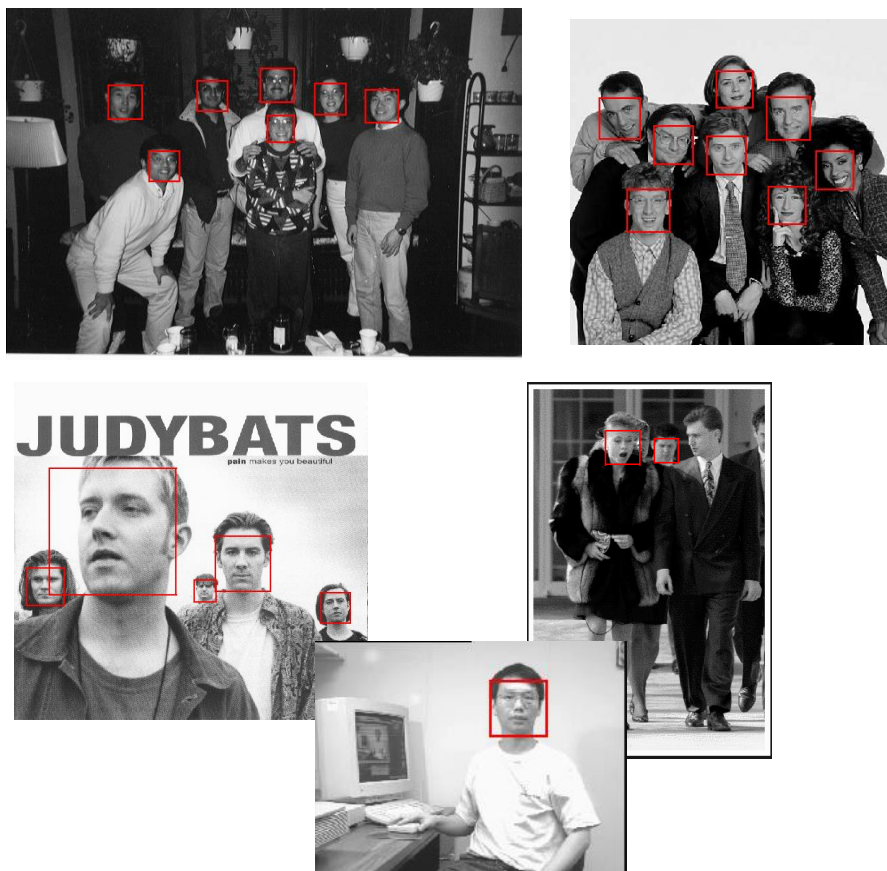
- Training data
 - 4916 labeled faces
 - 9544 non-face images → 350M non-face sub-windows
 - 24×24 pixel size
- Cascade layout
 - 38 layer cascade classifier
 - 6061 total features
 - S1: 1, S2: 10, S3: 25, S4: 25, S5: 50, ...
- Evaluation
 - Avg. 10/6061 features evaluated per sub-window
 - 0.067 sec/image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



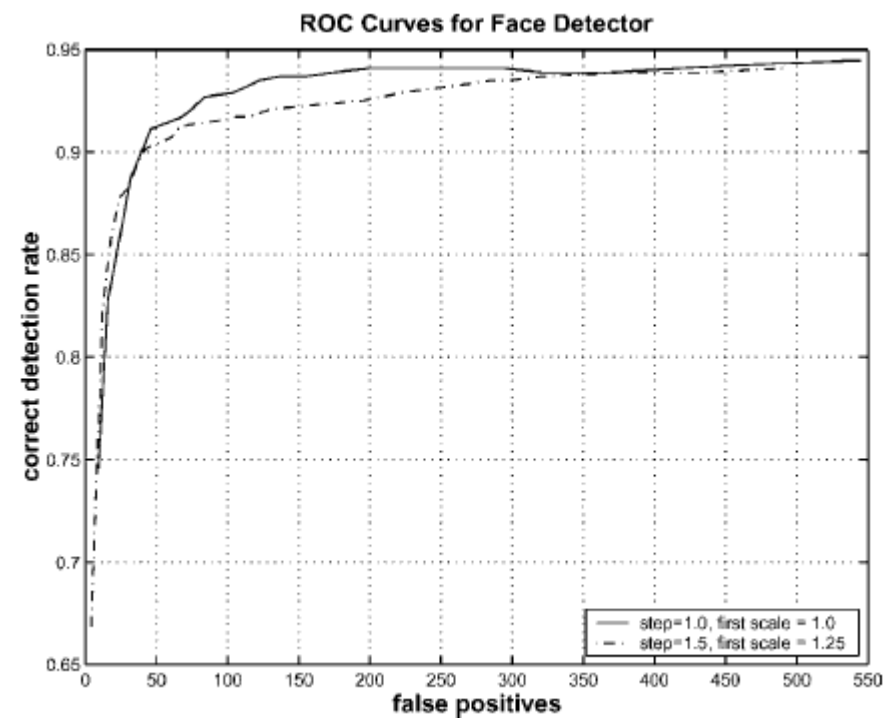
Similar performance between cascade and big classifier, but cascade is $\sim 10\times$ faster

MIT+CMU FACE TEST

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



Detector	False detections							
	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

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

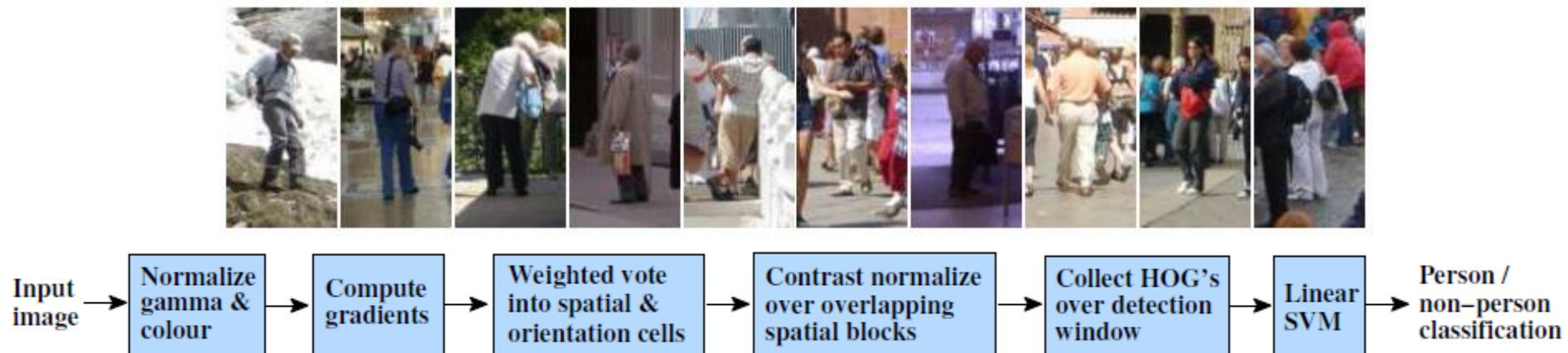
- Detector only works on frontal faces ($< 45^\circ$)
- Sensitive to lighting conditions
- Multiple detections to same face due to overlapping sub-windows

HOG DETECTOR

DALAL AND TRIGGS, CVPR2005

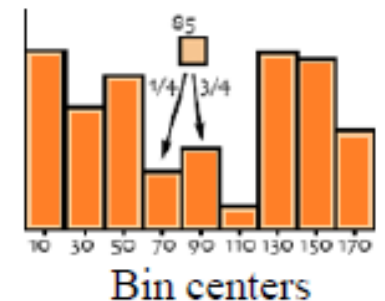
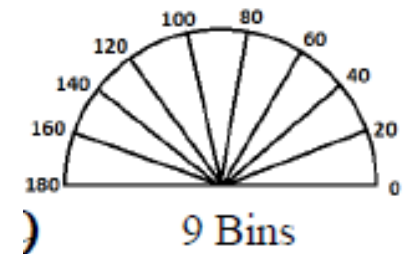
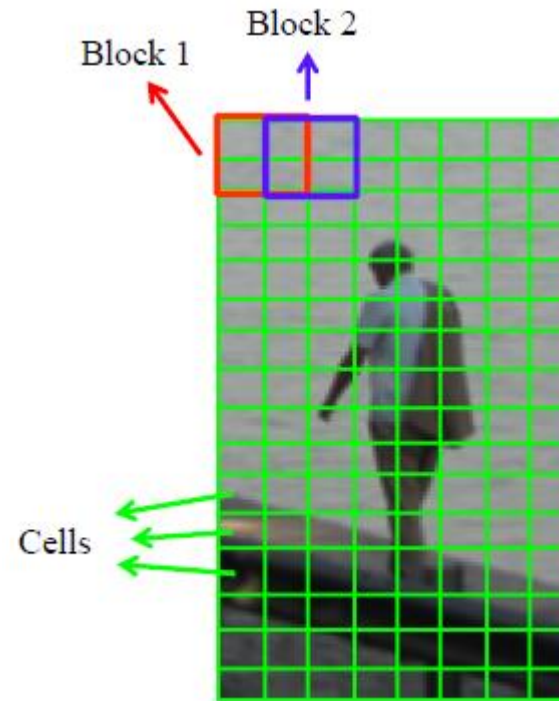
HISTOGRAM OF ORIENTED GRADIENTS

- 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



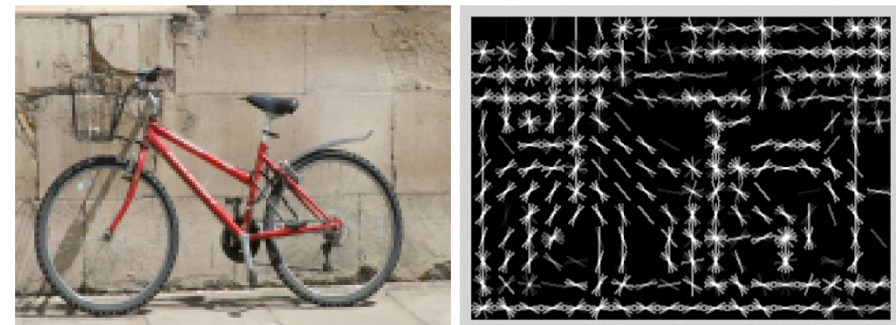
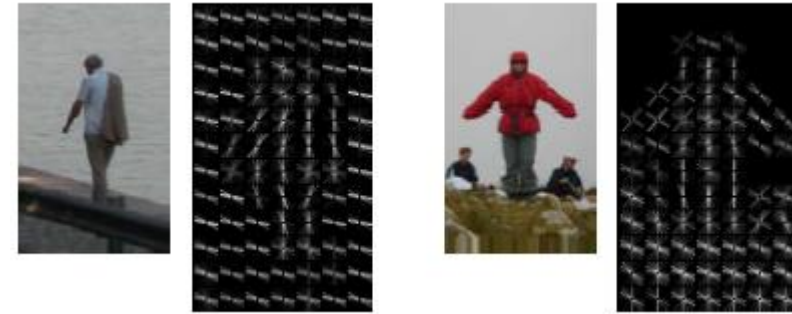
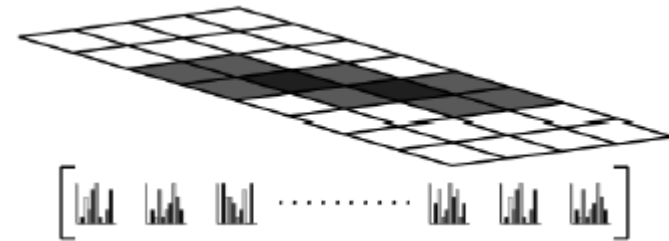
HOG STEPS I

- 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 $\rightarrow 7 \times 15 = 105$ 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
 - $\# \text{features} = (15 \times 7) \times 9 \times 4 = 3780$
 - 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

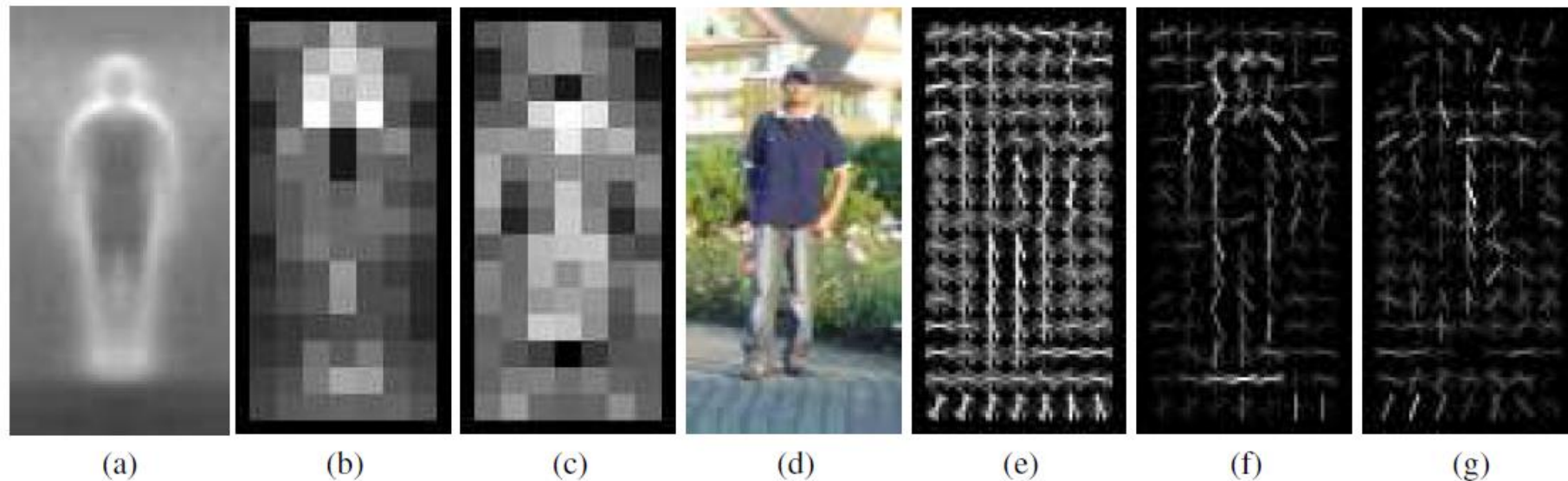


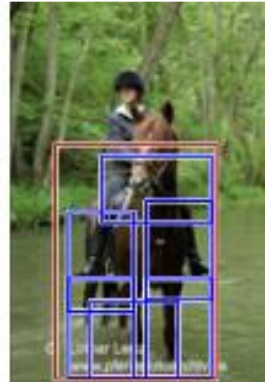
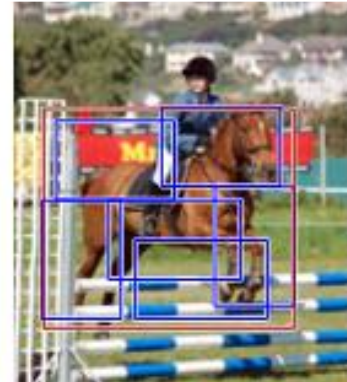
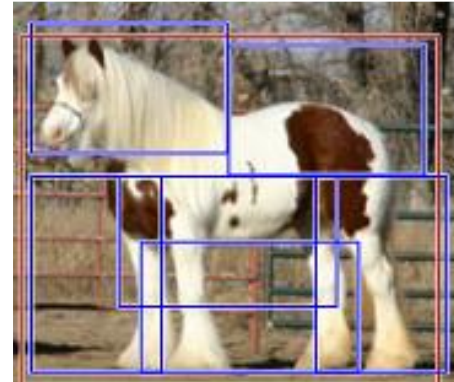
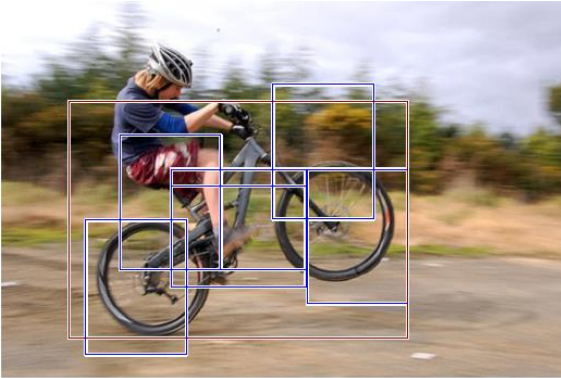
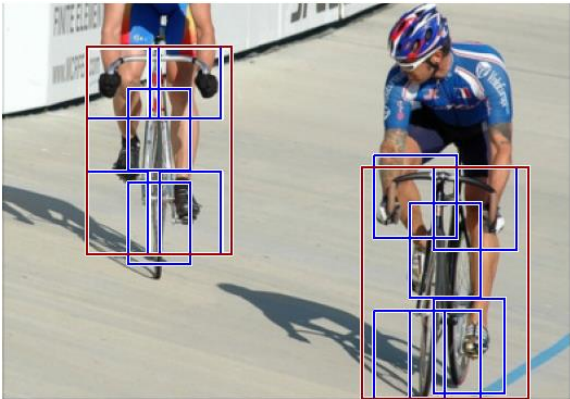
Figure 6. Our HOG detectors cue mainly on silhouette contours (especially the head, shoulders and feet). The most active blocks are centred on the image background just *outside* the contour. (a) The average gradient image over the training examples. (b) Each “pixel” shows the maximum positive SVM weight 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.

DPM DETECTOR

FELZENSZWALB, PAMI2010

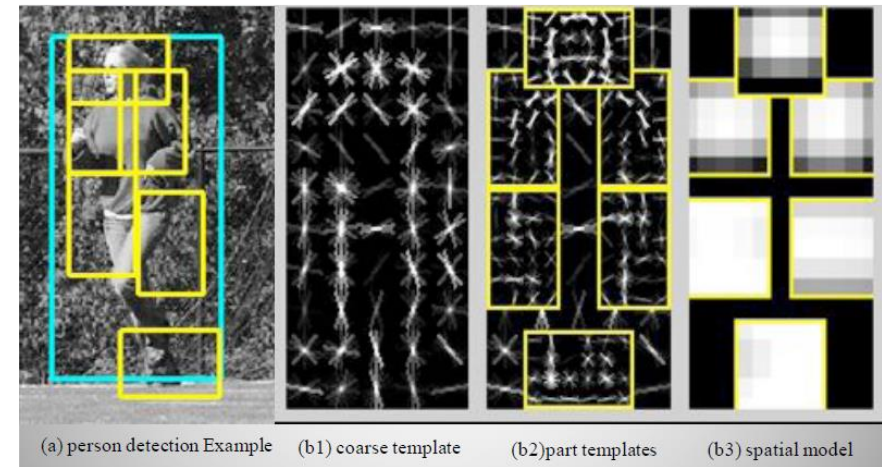
DEFORMABLE PARTS MODEL

- Want to detect all objects of the same category within in image
- Must account for dramatic appearance differences
 - Object is composed of parts in different positions
 - Non-rigid objects



DPM COMPONENTS

- Root – rough appearance of object
- Part – local appearance of object
- Spring – spatial connection between parts
- Use HOG descriptors



DPM SEARCH

- Use pyramid to view image at different scale
 - Coarse level (low resolution) used for root filter (general object outline)
 - Fine level (high resolution) used for parts
- Use a mixture of models to handle wide variation in appearance
 - E.g. model for front and side view of a person/horse/bike

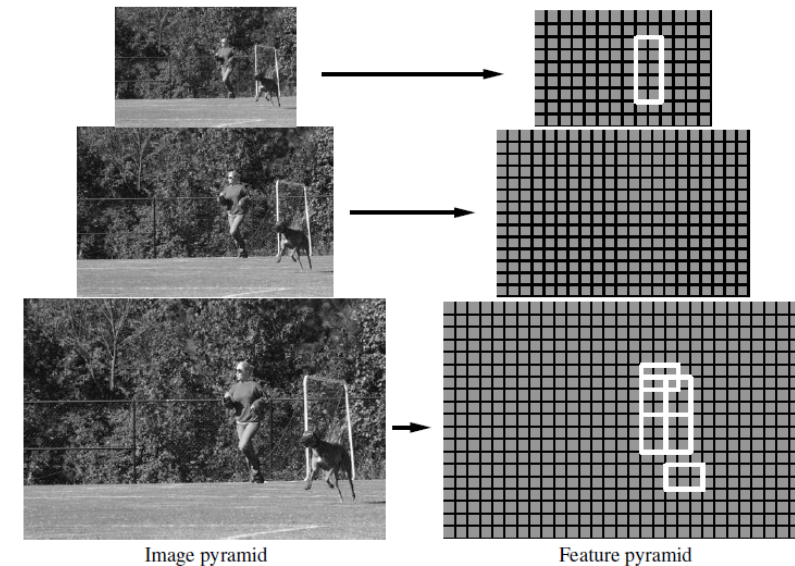


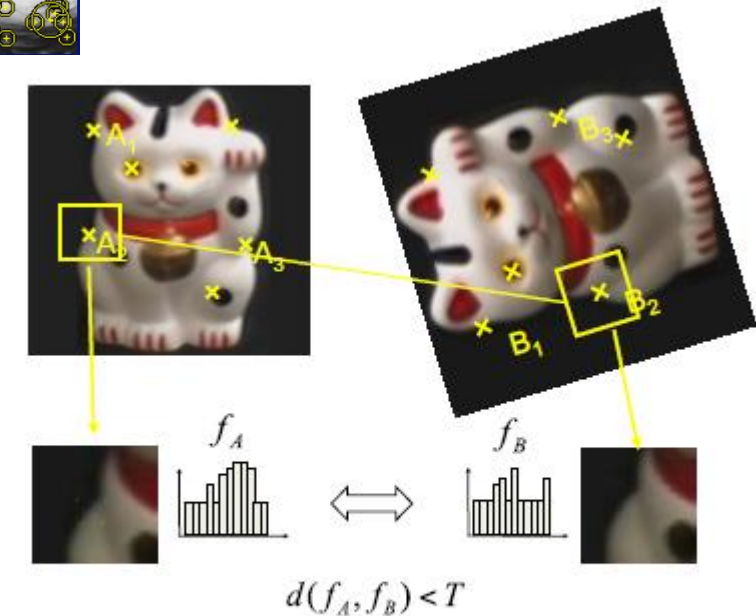
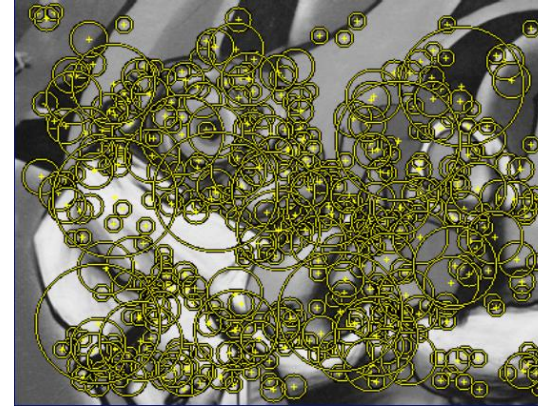
Fig. 3. A feature pyramid and an instantiation of a person model within that pyramid. The part filters are placed at twice the spatial resolution of the placement of the root.

SIFT FEATURES

LOWE, IJCV 1999

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



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

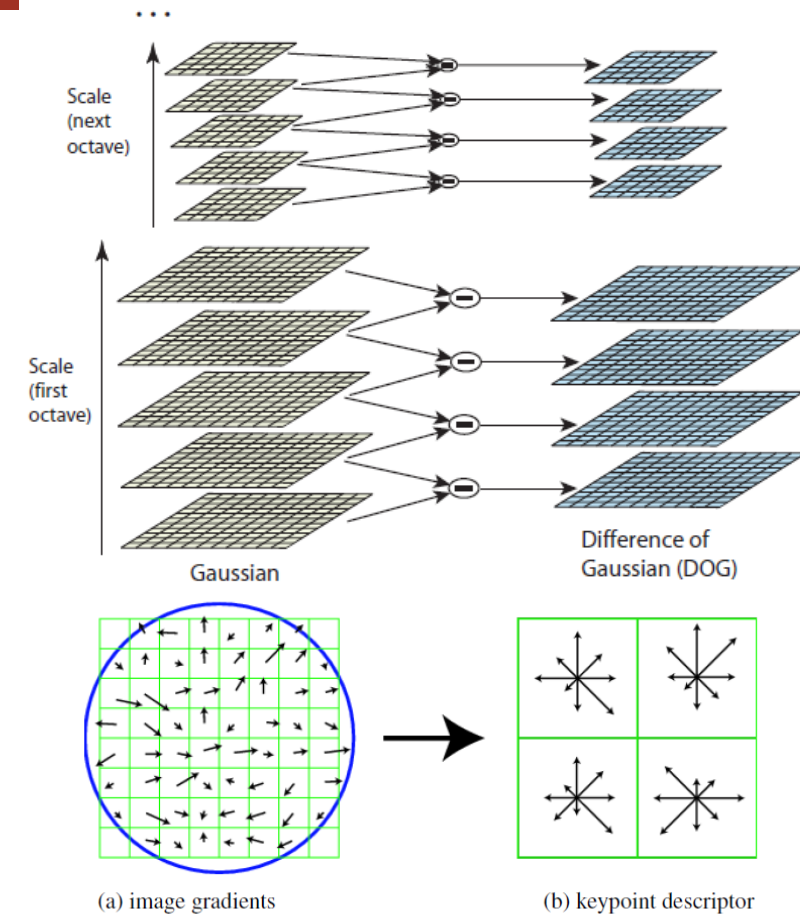


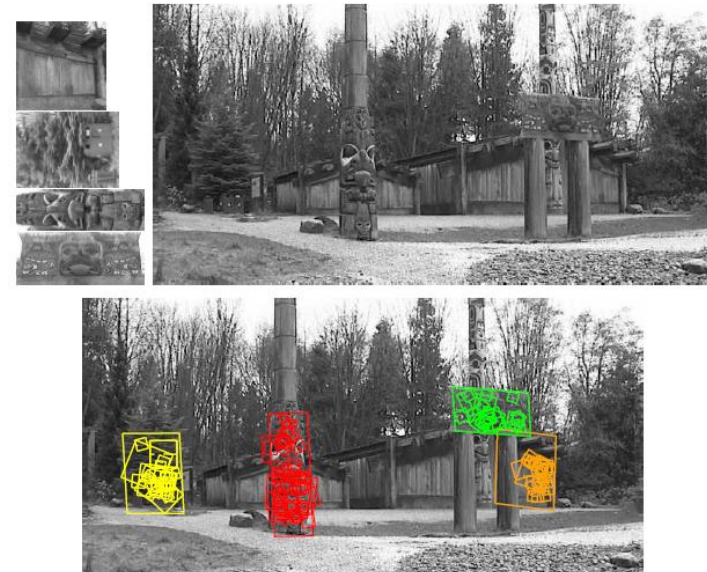
Figure 4.18 A schematic representation of Lowe’s (2004) scale invariant feature transform (SIFT): (a) Gradient orientations and magnitudes are computed at each pixel and weighted by a Gaussian fall-off function (blue circle). (b) A weighted gradient orientation histogram is then computed in each subregion, 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

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 [[cascade classifier](#)][[HOG](#)][[SIFT-like](#)]