Mid-High-Level CV: Object Detection

ENEE 4584/5584 CV Apps in DL

Dr. Alsamman

Slide Credits:
Adaboost: S. Chen, S. Lezebnik
HoG: N. Dalal & B. Triggs, G. Lacey, N. Snavely, K. Grauman, E. Seeman

Object Detection

- A subset of object classification
  - One object of interest
- Added goal of localization
- Practical CV : Image from the “wild”
  - No region of interest
    - Challenging localization
  - Small subset of pixels belong to the object
    - More rejection than detection
  - Multiple instances of the same object
  - 3D variation in pose
  - Variations in scale
  - Variety of occlusions
  - Real-time performance

Traditional Object Detection: Adaboost

- Overview:
  - Proposed for face detection
  - Build strong classifier from many weak classifiers
  - Box function classifier on integral images
  - Train classifier to detect & reject
  - Support vector machine (SVM) classification at the output
  - Cascade classifiers for real-time response

Adaboost

- Adaptive boosting: a machine learning algorithm
- Builds a "strong" classifier iteratively from "weak" classifiers
- Weak classifiers: $h_t(x) \in \{1,-1\}$
  - h_t : Classifier used as iteration t
  - Usually linear binary: Classifies feature x as in/out class .
  - slightly better than random! but fast.
- Strong classifier:

$$H_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

  - Hard classifier: Signum function
  - Use T most beneficial features
  - Each classifier is weighted by α
  - Linear combination: Close to sequential decision making

Algorithm

- Singer & Schapire (1997)
- Given: $(x_1,y_1), (x_2,y_2),\ldots, (x_m,y_m)$
  - $x_i \in X$; $y_i \in \{-1, 1\}$
- Initialize the probab. distribution of X:

$$D_1(i) = 1/m;\ i = 1,..,m$$

- For t = 1:
  - Find a classifier that minimizes the error

$$\varepsilon_t = \sum_{i=1}^{m} D_t(i)\{y_i \neq h_t\}$$

i.e. error is the sum of prob. of each wrongly classified feature.

  - Stop when $\varepsilon_t < 0.5$ (better than chance!)

- Calculate the weight of h_t that minimizes exponential loss:

$$\alpha_t = \frac{1}{2} Ln \frac{1-\varepsilon_t}{\varepsilon_t}$$

- New prob. distribution:

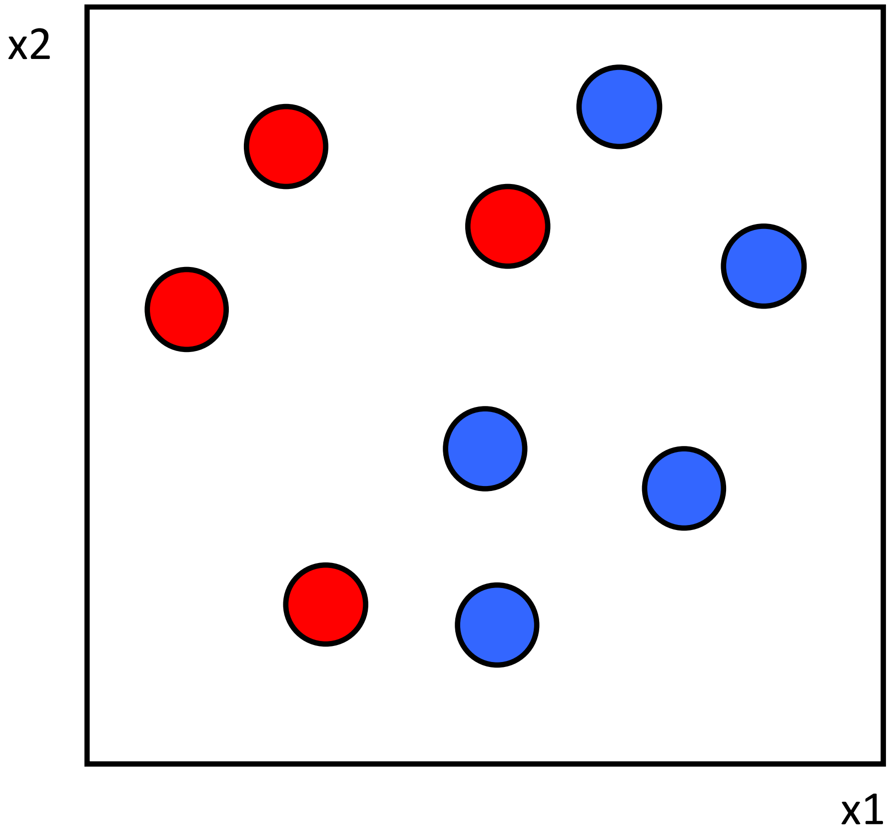
$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

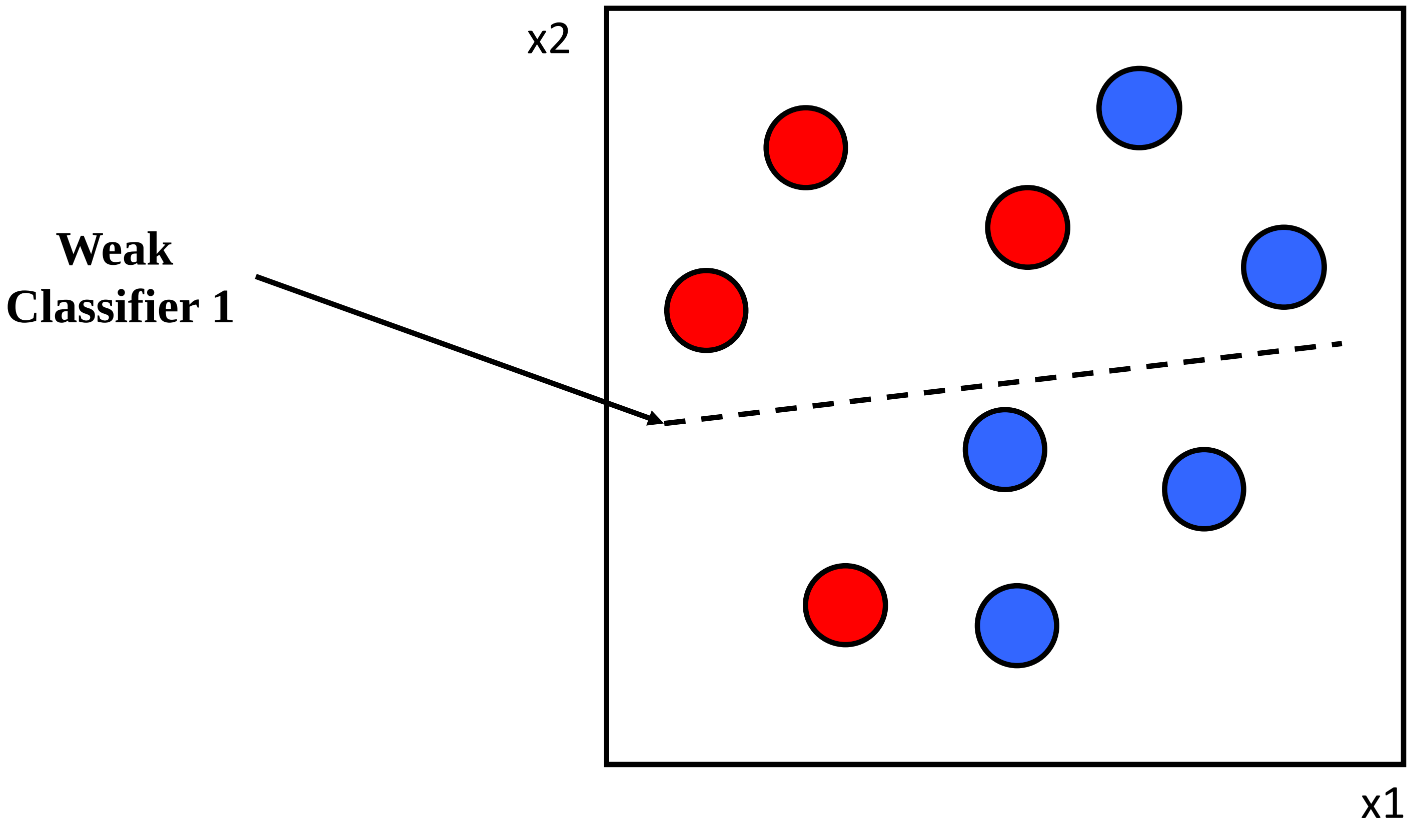
  Z_t is a normalization factor, so that D_{t+1} is a distribution:
  i.e. all probab must add up to 1.

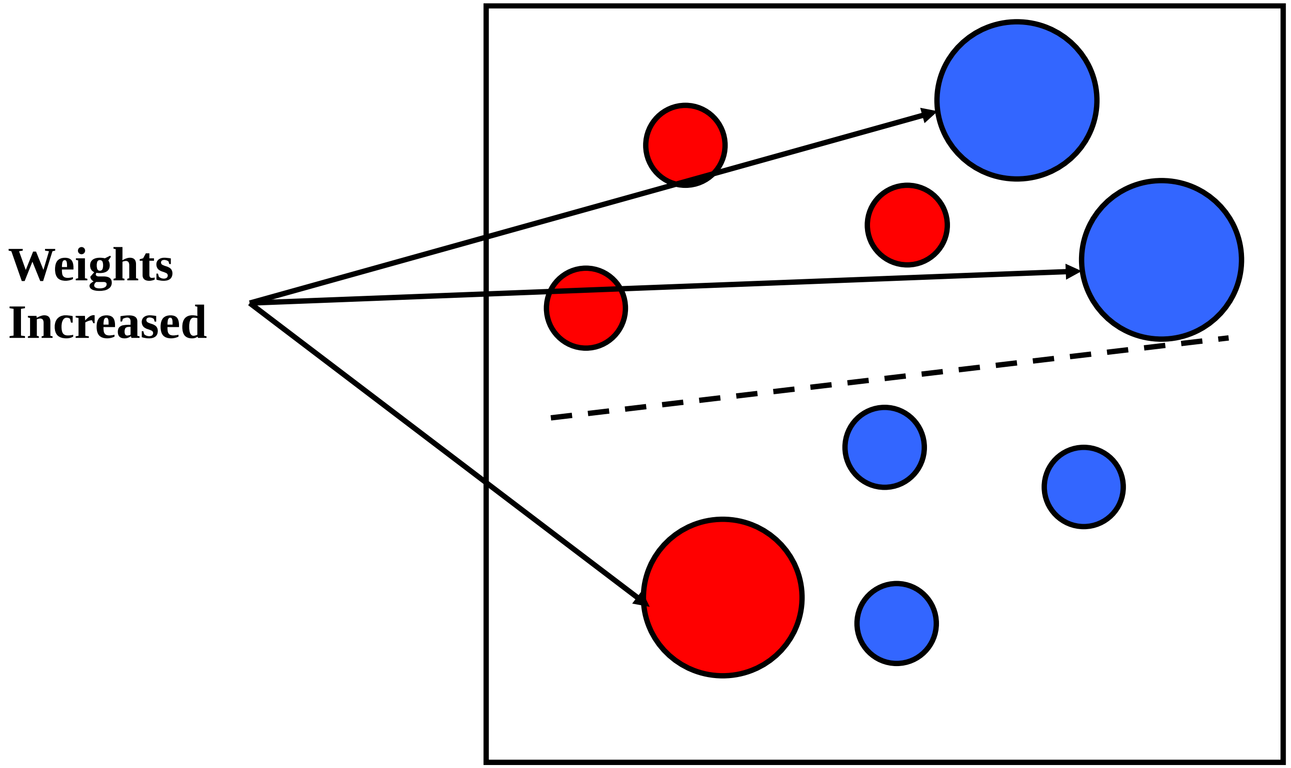
- Repeat for t+1, stop at T: when ε_T=0 (or ε_T<T)
- Sum all classifiers, then use a hard classifier

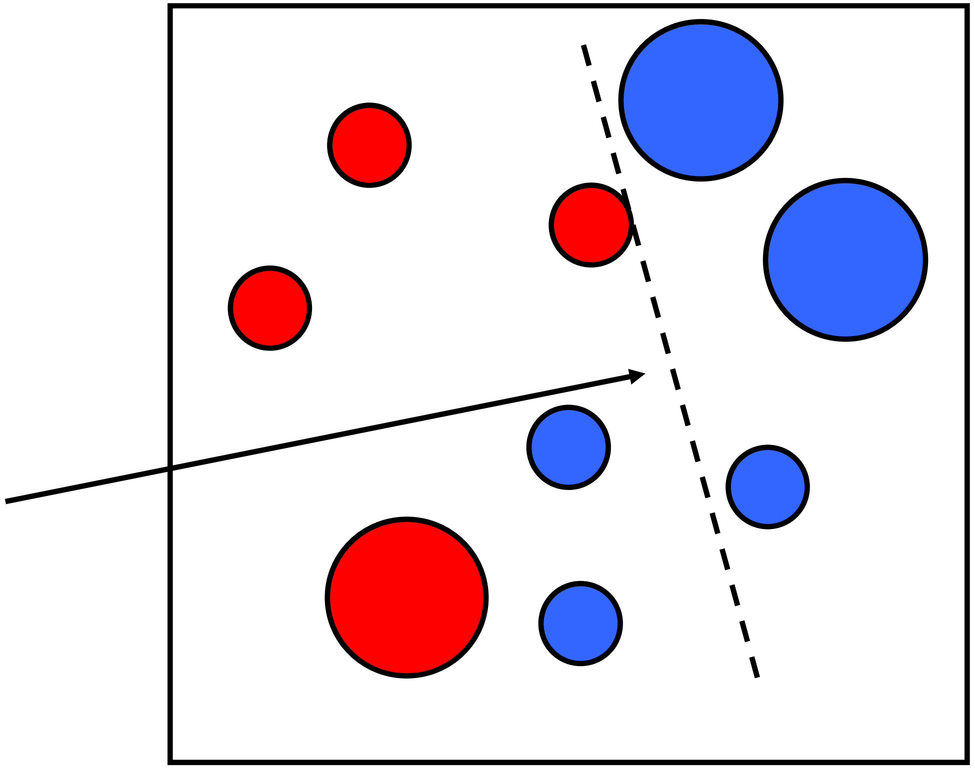
$$H_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Example

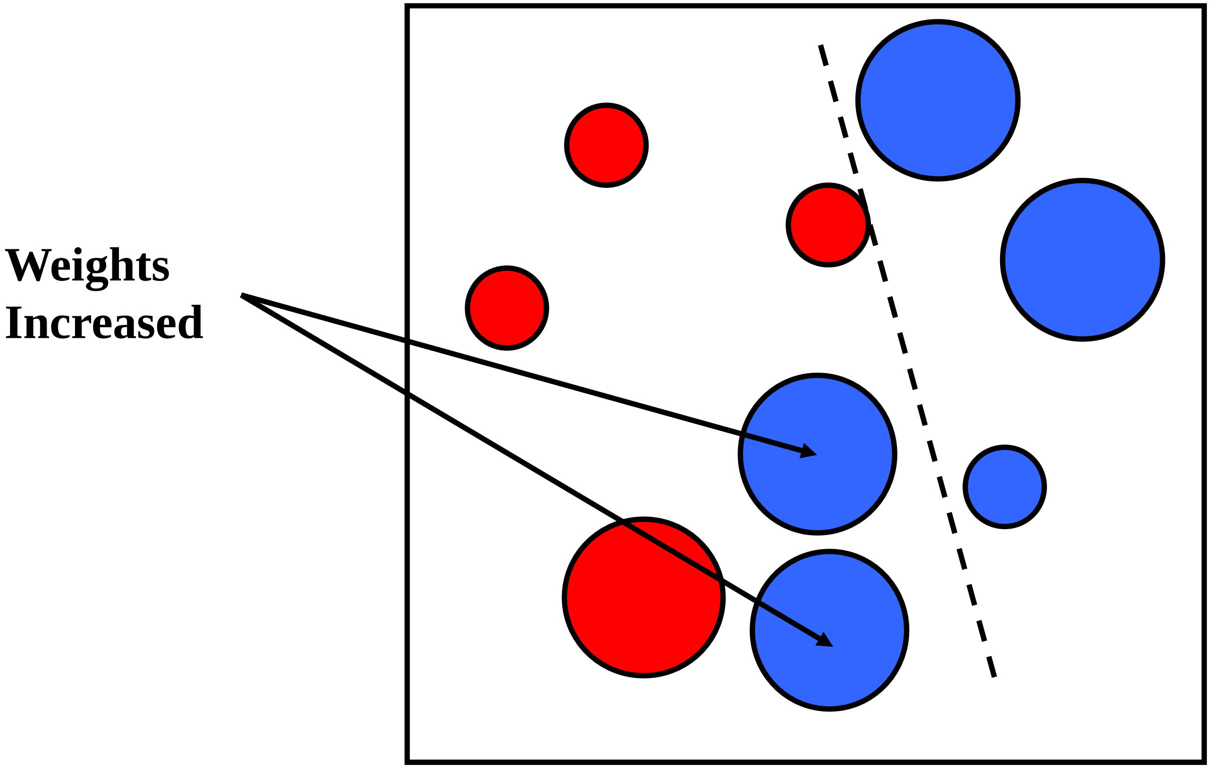


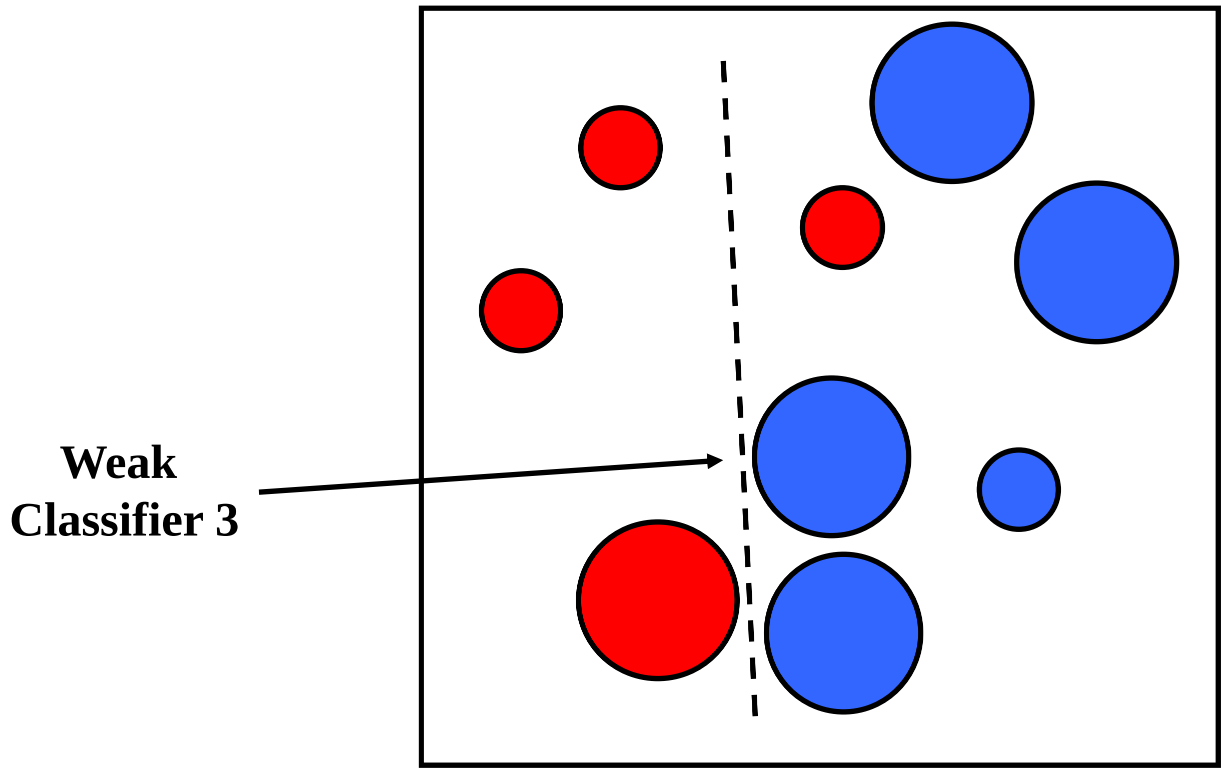

x2
Weak
Classifier 1
x1


Weights
Increased

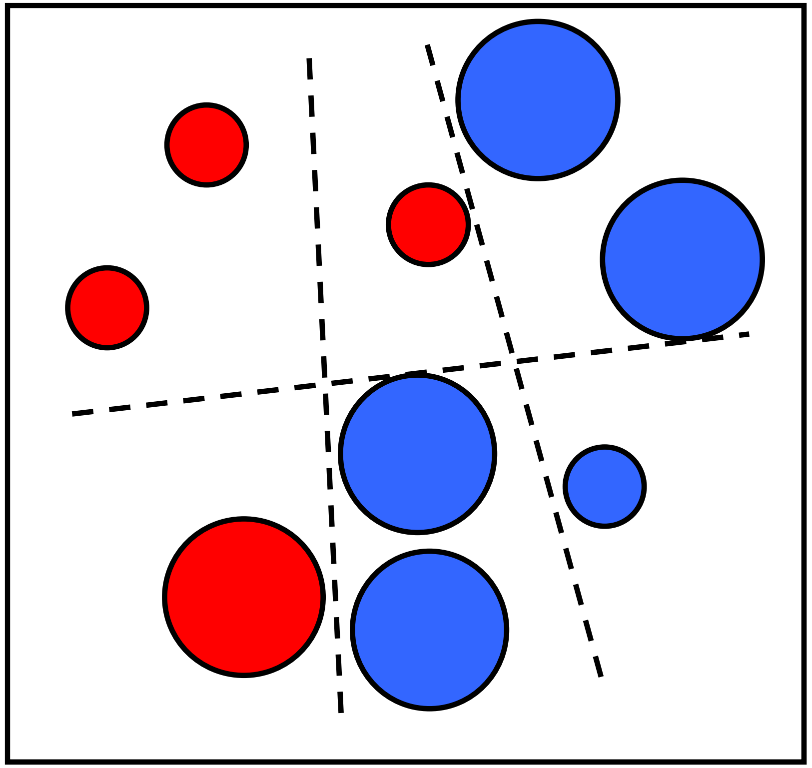


Weak Classifier 2


Weights
Increased


Weak
Classifier 3

Final classifier is
a combination of weak
classifiers



Face Detector

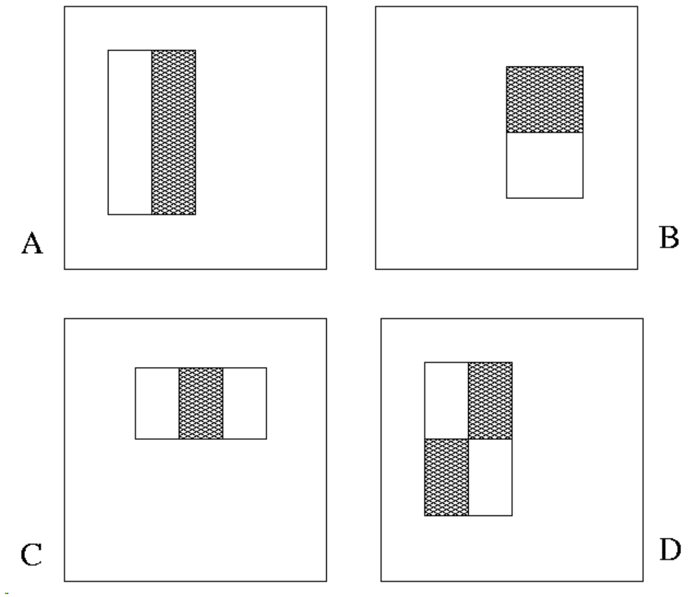
- Basic Idea: Slide a window, evaluate a face model at every location.
- Faces are rare: 0–10 per image
- For computational efficiency, we should try to spend as little time as possible on the non-face windows
- A megapixel image has ~10^6 candidate face locations
- To avoid having a false positive in every image, our false positive rate has to be less than 10^6

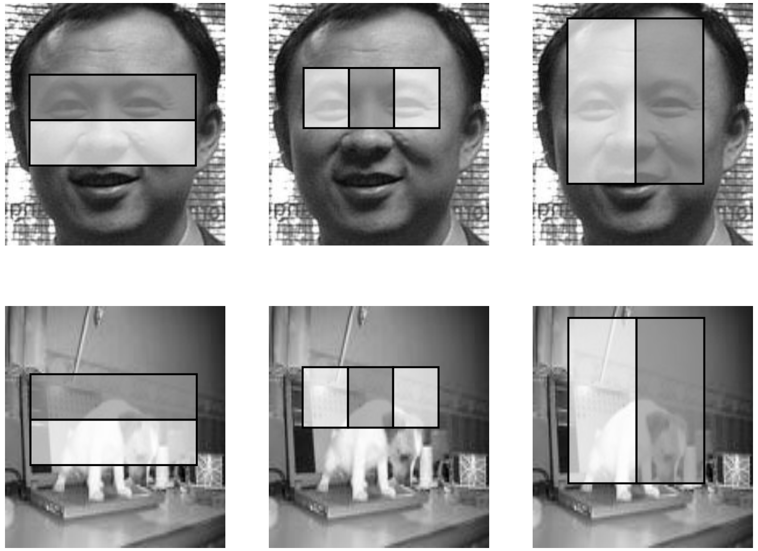
Viola-Jones Face Detector

- S.l.o.w. training, but *fast* detection
- Feature: box function (haar-like) functions
- Weak classifier: Integral image
- Attentional cascade:
  - fast rejection of non-face windows
  - Slower detection of face windows

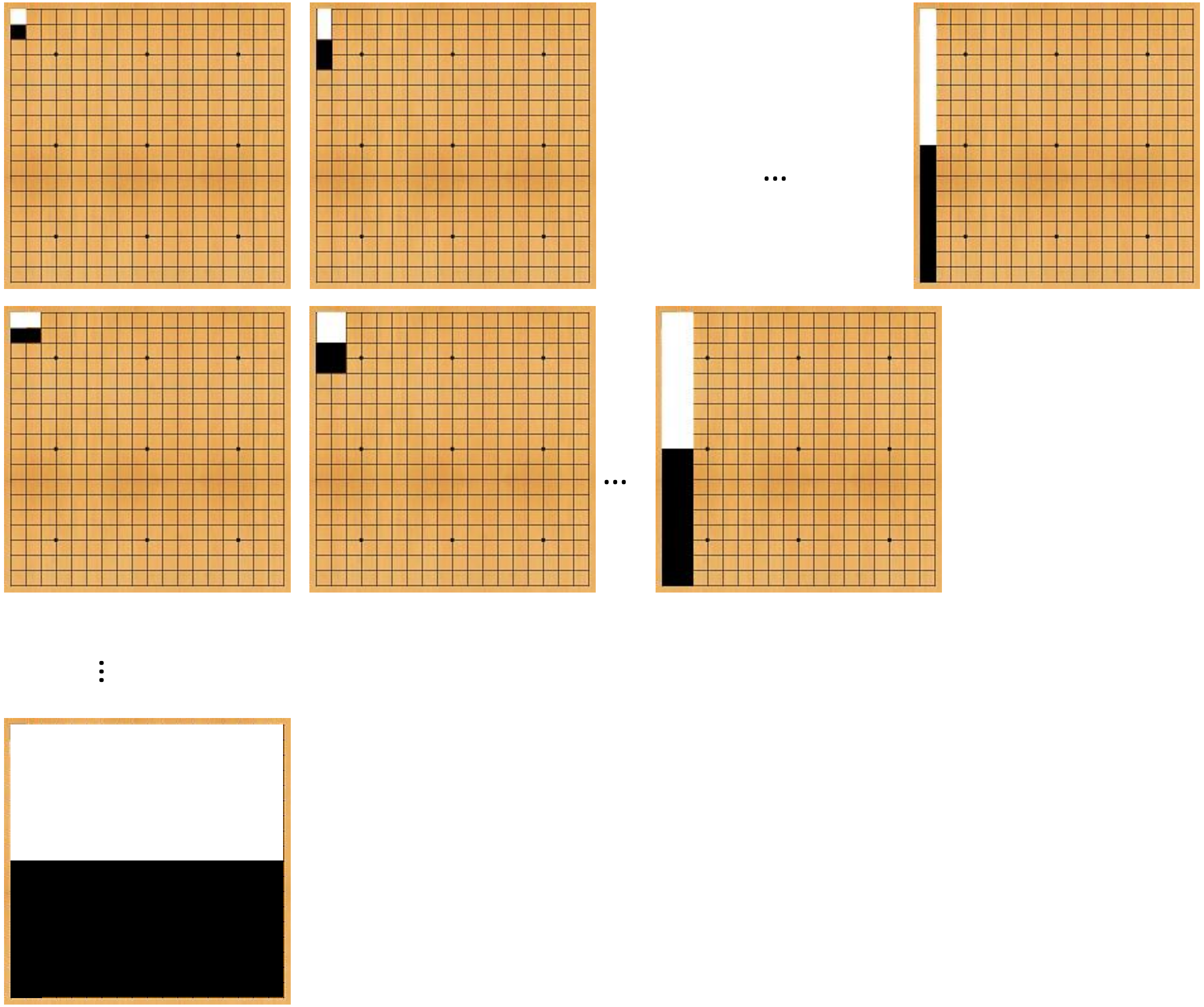
Box-functions

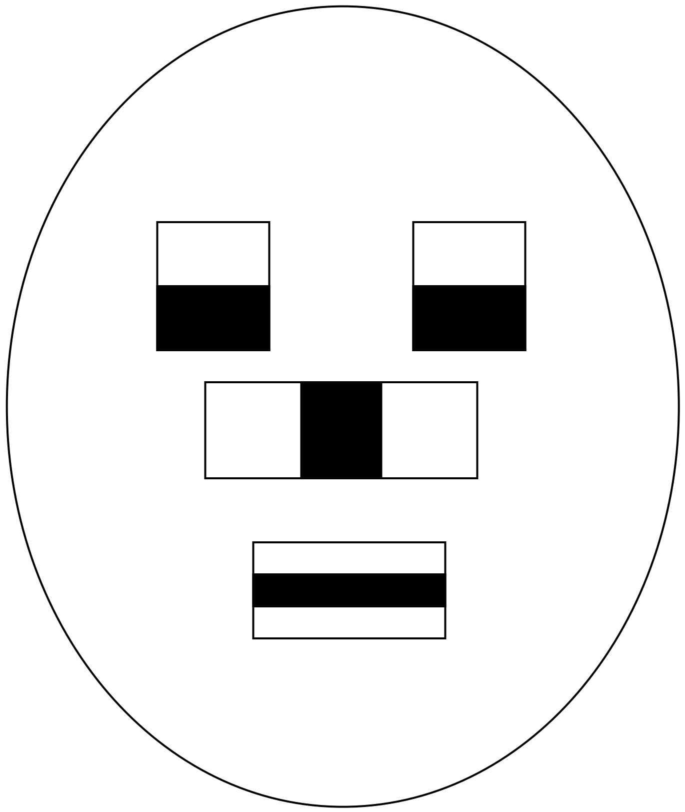
- Rectangle features: sensitive to edges, lines
- Feature value = $\sum$pixels in white area - $\sum$pixels in black area
  - Gray-level images only





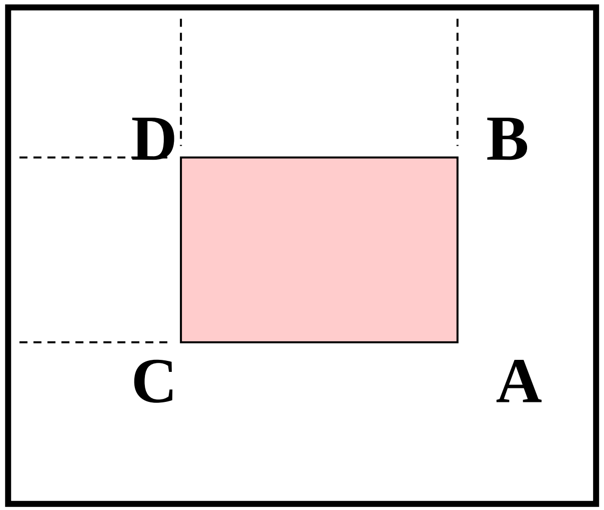
- Feature value is calculate for various sizes of the Haar functions in 24x24 area.
  - Calculate over a sliding window
  - Vary the height and width of the function
- Possible Haar variations in a 24x24 region:
  - Total ~160,000
- Training will reveal that a small subset (~10 features) are sufficient.





Integral Image

- Integral image at (x,y) = $\sum$ pixels values above, to the left
  - $$ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y')$$
  - Can be computed in 1 pass
  - Cumulative sum fuction
- Sum within an area:
  - Sum = $ii_A - ii_B - ii_C + ii_D$
- Evaluation of all rectangular sizes in constant time



Weak Binary Classifiers

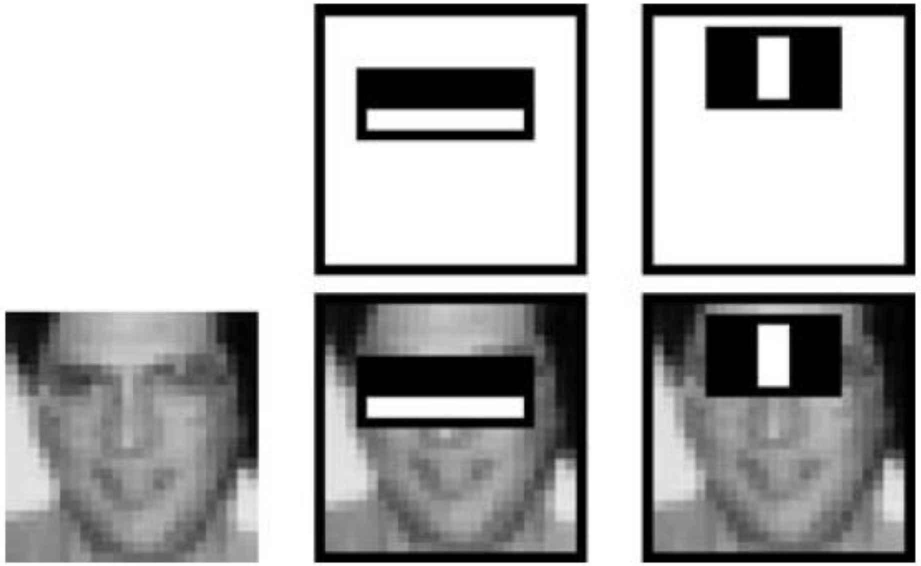
$$h_t(x) = \begin{cases} 1 \;\; if \quad p_t h_t(x) > p_t \theta_t \\ 0 \quad\quad otherwise \end{cases}$$

- $h_t()$: classification function at iteration t
- x: feature = window size = wxh
- $f_t()$: value of Haar feature
- θ_t: threshold (learning objective)
  - There will be a separate threshold for each haar function used
- p_t : polarity {1,-1} for face vs non-face

Adaboost Training

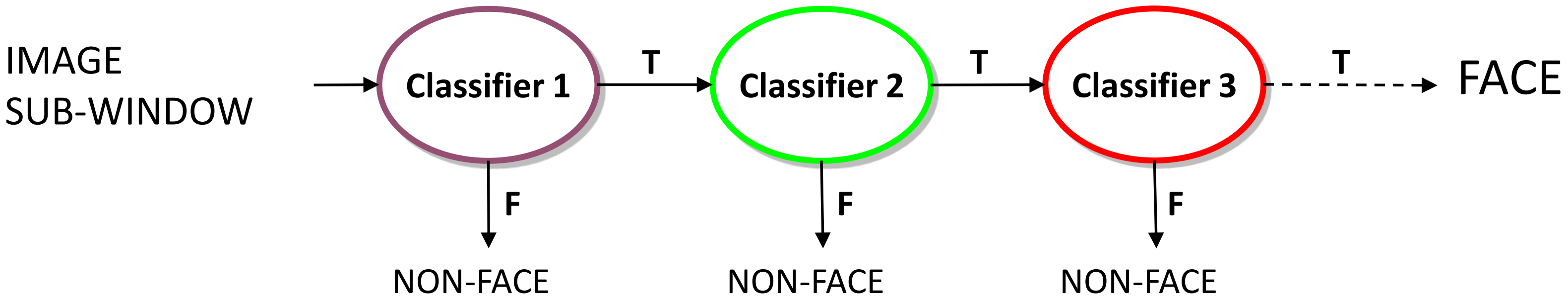
- Create a very large database of face and non-faces
  - 10k-100k in size for greater accuracy
  - Faces should be cropped so only face is showing
  - Faces should include variations in lighting, occlusions, poses, scale
- For each iteration t:
  1. Initialize weights
  2. Evaluate each Haar function on each training sample
  3. Select the threshold θ that reduces the error
  4. Calculate weights, repeat until error = 0.
- Computational complexity ~$O(T \times N \times K)$
  - T iterations, N training samples, K Haar features
  - The key to reducing complexity is K

- With only 2 basic box features, 100% detection of faces is possible
- Big problem: 50% false positive
  - For a megapixel image, error = ?
- With 200 features: 95% detection, 0.007% false positive
  - Better but not good enough



Attentional Cascade

- Start with simple classifiers that reject many of the non-faces (negative) windows, while detecting the almost all of the face (positive) windows
- Positive response triggers classification use a more complex classifier
- Negative response causes rejection of that window and sliding the window.



- Detection rate = $\prod$ false positive rate of each stage
  - Detection rate must be high to maintain successful detection
  - E.g. if each stage has 90% detect, for 10 stages: $0.9^{10} = 0.35$!
- False positive rate = $\prod$ false positive rate of each stage
  - False positive rate should be low

Training Cascade

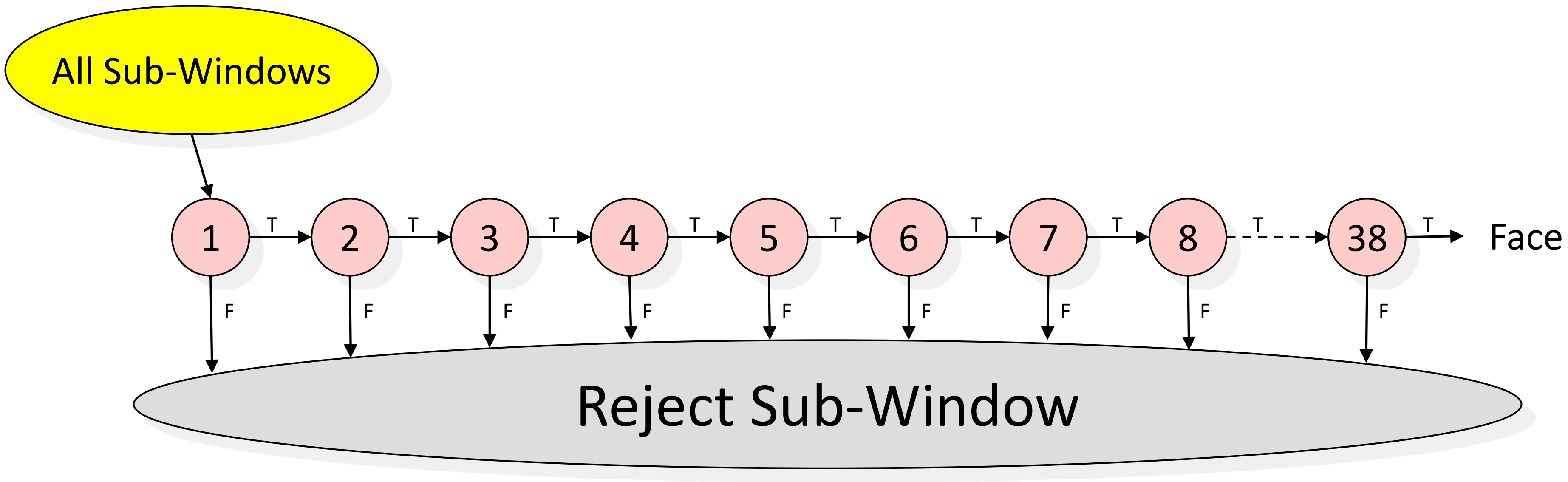
- Set target detection and false positive rates for each stage
- Keep adding Haar features to the current stage until its target rates have been met
  - Need to lower AdaBoost threshold to maximize detection (as opposed to minimizing total classification error)
  - Test on a validation set
- If the overall false positive rate is not low enough, then add another stage
- Use false positives from current stage as the negative training examples for the next stage

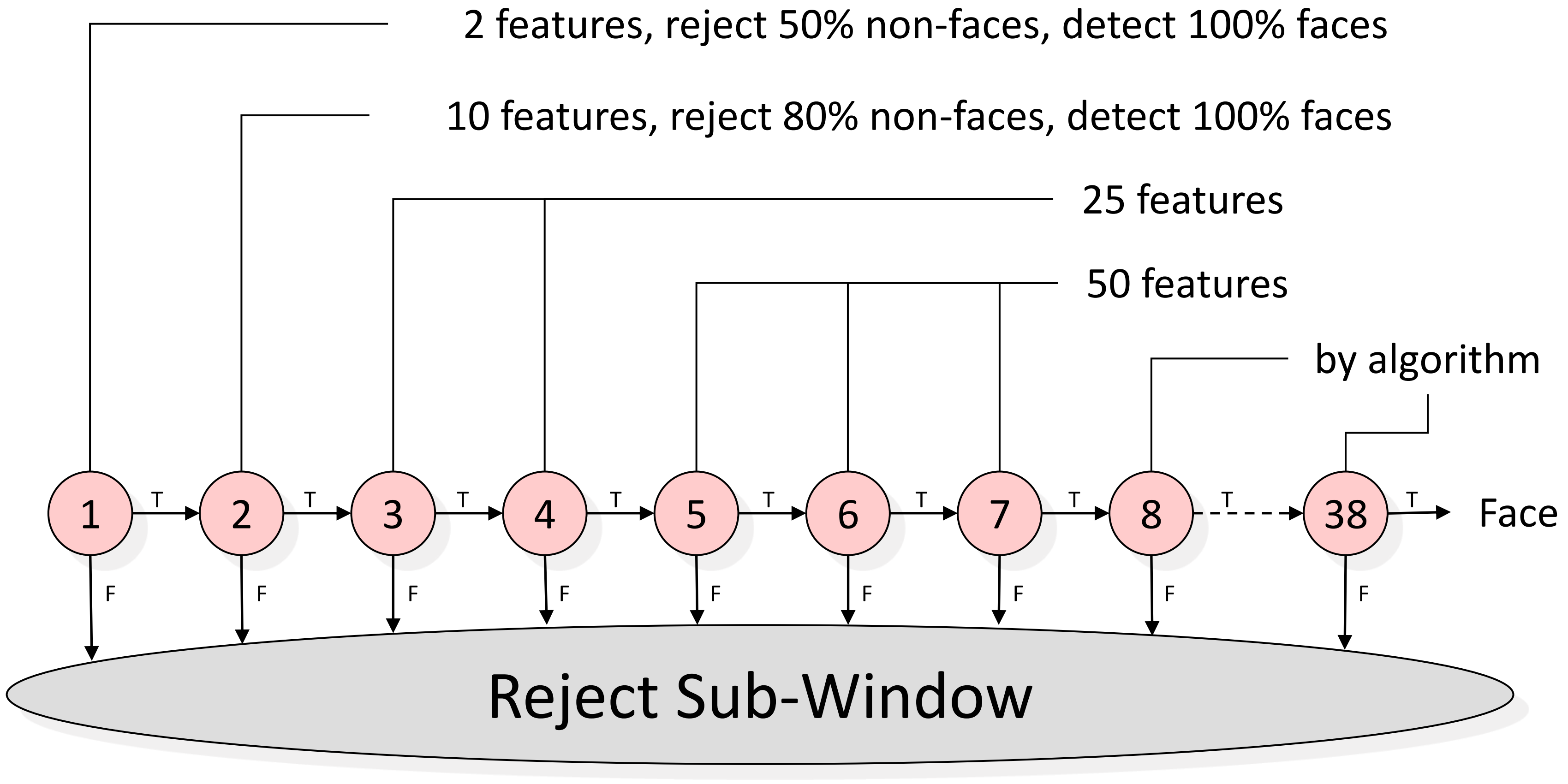
Viola & Jones: Database

- Training Data
  - 5000 faces
    - All frontal, rescaled to 24x24 pixels
  - 300 million non-faces
  - Faces are normalized
    - Scale, translation
- Many variations
  - Across individuals
  - Illumination
  - Pose
  - Not scale!

Structure of the Detector Cascade

- Combining successively more complex classifiers in cascade
  - 38 stages
  - included a total of 6060 features




2 features, reject 50% non-faces, detect 100% faces
10 features, reject 80% non-faces, detect 100% faces
25 features
50 features
by algorithm
1
2
3
4
5
6
7
8
38
T
F
Face
Reject Sub-Window

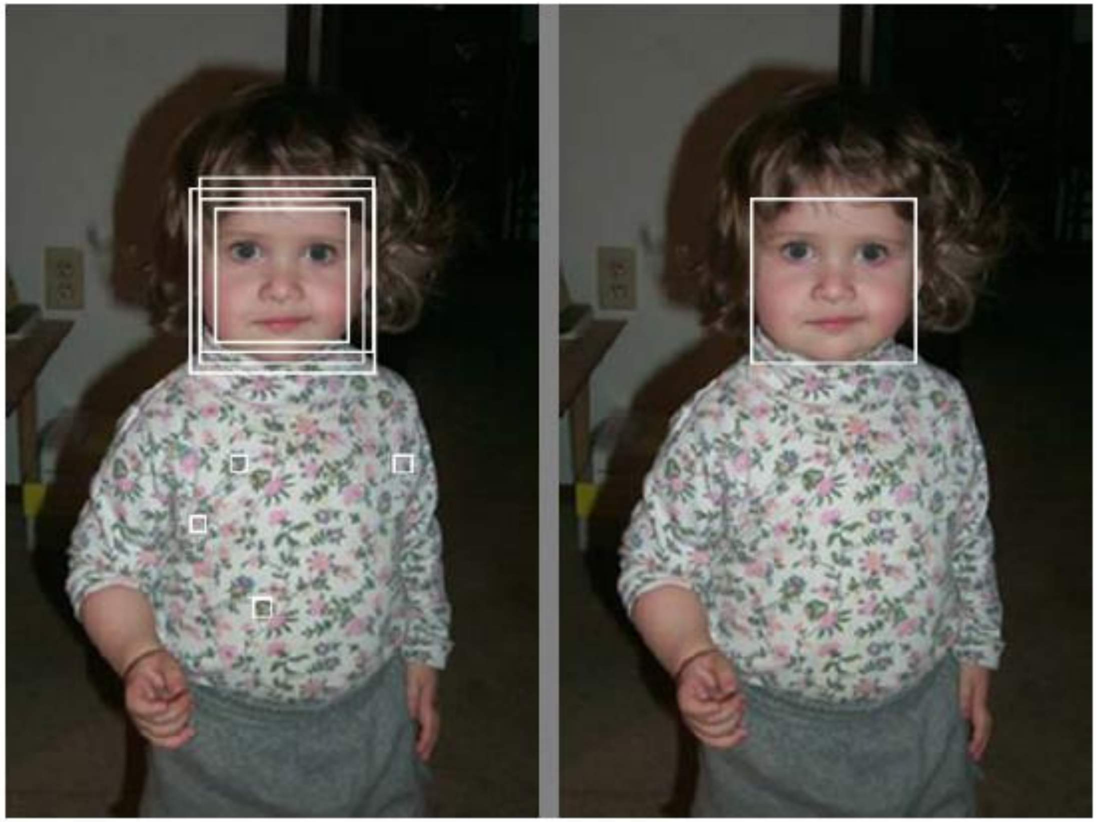
Speed of the Final Detector

- Shifting the window some number of pixels D
  - choice of D affects both speed and accuracy
  - D > 1 decreases the detection rate slightly
  - D > 1 decreases false positives
- Processes large image in milliseconds.
- Average of 8 features evaluated per window on test set

Integration of Multiple Detections

- Combine overlapping detections into a single detection
  - The corners of the final bounding region are the average of the corners of all detections in the set.
  - Decreases the number of false positives.
- A simple Voting Scheme further improves results
  - Retaining regions that produces the same error rate in the final stage.
  - This improves the final detection rate as well as eliminating more false positives.
  - Since detector errors are not uncorrelated, the combination results in a measurable, but modest, improvement over the best single detector.

Merging Multiple Detections



Scale

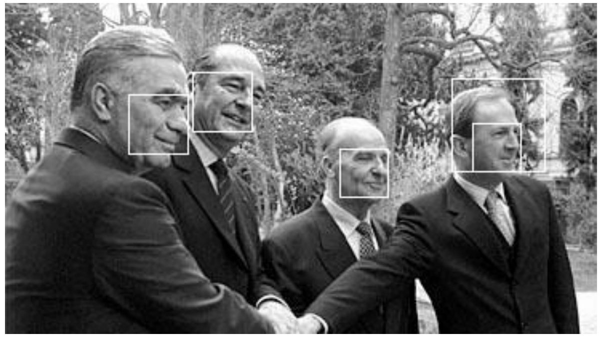
- Don’t scale images, scale features.
- Face detector scans the input at many scales
  - starting at the base scale: detect face at a size of 24 × 24 pixels,
  - Then at 12 scales, 1.25 larger than the last
  - 384 × 288 pixel image is scanned at the top scale

Failure Cases

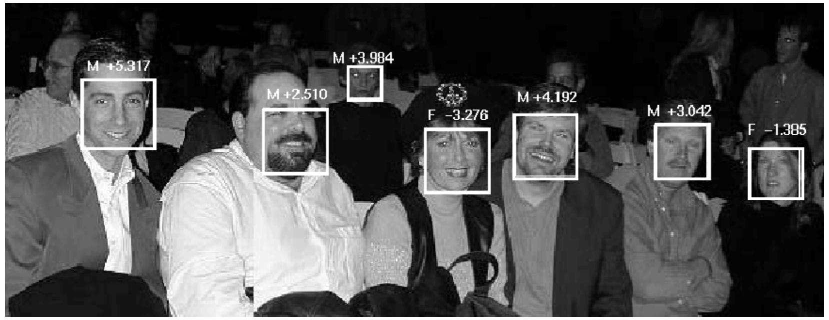
- Large rotations
  - Systems is trained on frontal, upright faces
  - ±15 degrees in plane and about ±45 degrees out of plane is acceptable. More causes failure
- Harsh backlighting: dark faces, light background.
  - Additional but (real-time) prohibitive techniques can be employed.
- Significantly occluded faces
  - Occlusion of the eyes: usually fail.
  - The face with covered mouth will usually still be detected.

Other Applications

- Detection of facial features: eyes, nose, mouth



- Detecting facial profiles in images



- Detection of gender

- Pedestrian detection

Adaboost Overview

- Real-time detector
- Scale and location invariant detector
  - Instead of scaling the image itself (e.g. pyramid-filters), we scale the features.
- Can be trained for other object detection
- Very suitable for rejection
- Requires long training
- Can only detect images similar to training set
  - If no profile images exist in training, it can’t detect them
- Sensitive to lighting conditions
  - Affects integral calculation
- Multiple detections of the same object due to overlapping windows.

Histogram of Oriented Gradients

Motivation

- Object detection for Understanding scenes
- Challenges
  - Wide variety of articulated poses
  - Variable appearance and clothing
  - Complex backgrounds
  - Unconstrained illumination
  - Occlusions
  - Scales
  - Real-time detection

Chronology

- Template based: Edge detection
  - Supports irregular shapes & partial occlusions
  - Window free framework
  - Sensitive to edge detection & edge threshold
  - Not resistant to local illumination changes
- Feature (keypoint) based:
  - E.g. local maxima (SIFT), corners (MOPS)
  - SIFT uses gradient orientation histograms for feature orientation
  - Keypoints difficult to detect on certain objects

- Large positive and negative-heavy databases
  - E.g. Haar and SVM, adaboost
  - Fast detection
  - Robust and invariant
  - Uses large positive and negative-heavy database
  - Very slow learning

Why H.O.G

- Gradients:
  - Edges persist under various lighting changes and transforms
  - Gradients are normal (perpendicular) to edges
- Orientation
  - Angle/direction of gradient
  - Magnitude is susceptible to invariance illumination changes
- Histogram
  - A histogram of orientations can be used to describe a shape
  - Histograms are invariant to transformations (rot, scale, etc.)
  - Coarse binning (sampling) orientation leads to greater invariance
    - e.g. in hand gesture recognition
  - Finer binning is needed in some applications
    - E,g. pedestrian recognition

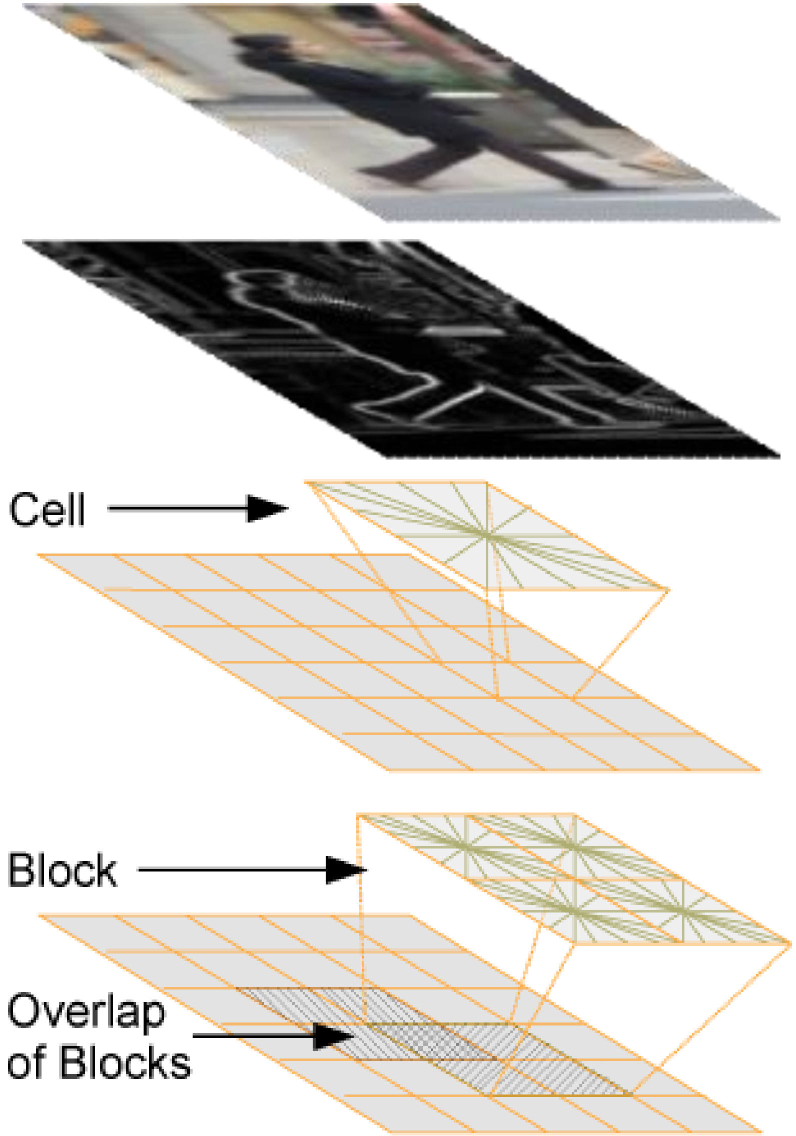
HOG

- Idea: describe shape using localized HOG
  - Widowed approach: HOG applied to local regions
  - Localization provides immunity to occlusions, background changes
- Idea: Use a database to build HOG descriptor for shape and anti-shape
  - Shape descriptor is a HOG vector that describes object of interest
  - Anti-shape descriptor is a vector that described other objects
  - Based on images taken from real environments

Training Algorithm

- Need a LARGE database of images
  - More non-object images than object
  - Need to determine dimension of object

1. Compute a HOG descriptor for each training image
   i. Color Processing : Selection of Grayscale, RGB, etc.
   ii. Gradient Calculation on each pixel
   iii. Orientation Binning of pixel groups (cells)
   iv. Normalization of local regions (blocks of cells)
   v. Generate HOG descriptor for image
2. Use a (linear) SVM classifier
   - 1-level classification: object or non-object
   - Learn classifier boundary equation
   - Reliable and fast, but not optimal


Cell
Block
Overlap
of Blocks

Testing

- For each new (test) image:
  - Use a sliding window,
  - Window size same as object size
  - In each window: compute HOG vector for that window
  - Classify window based on boundary equation

Color Processing

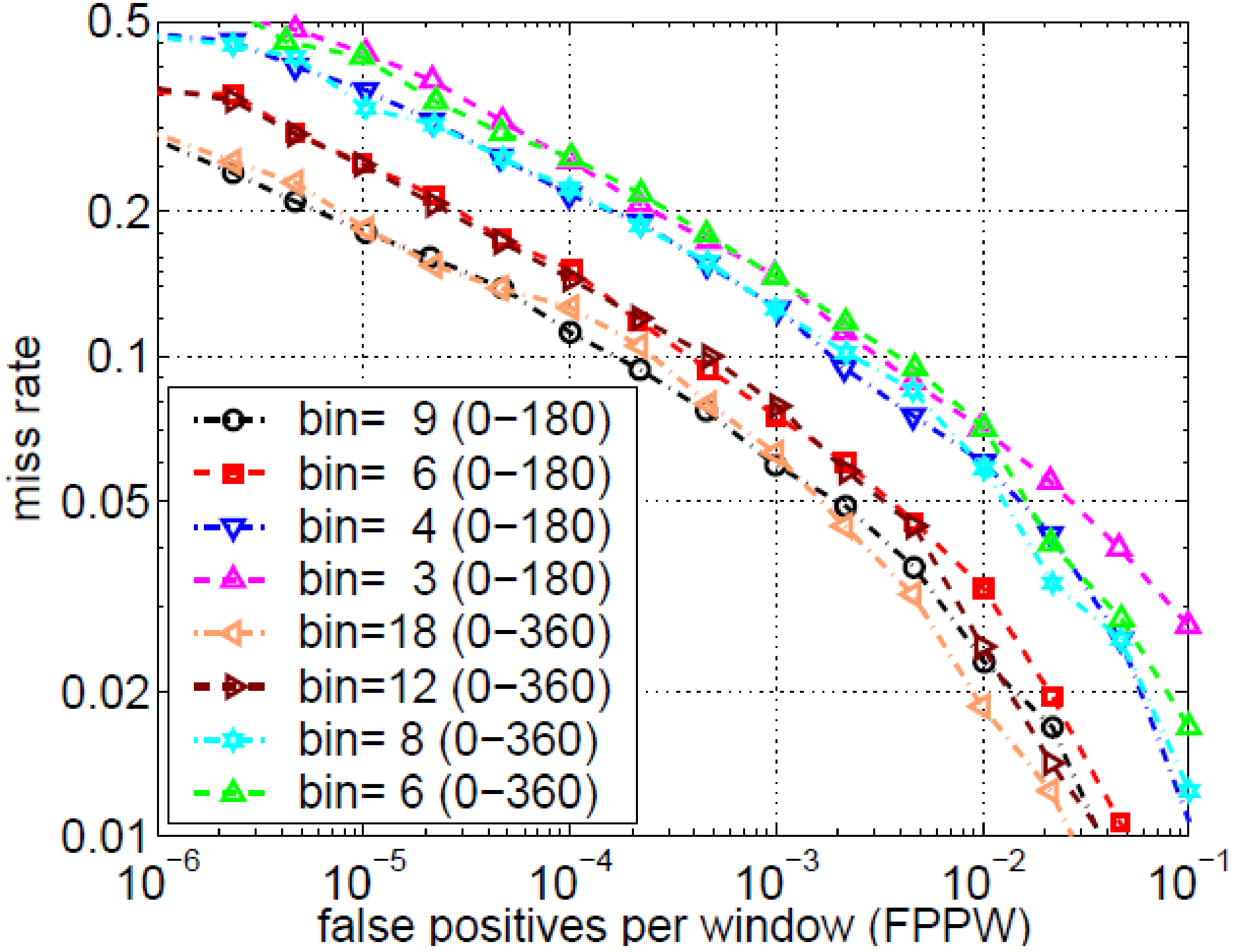
- RGB channels can be processed separately
- No advantage of RGB over LAB
  - CIE LAB:
    - L: lightness (black to white)
    - A: Red to Green
    - B: Yellow to Blue.
- Grayscale has -1.5% effect on results
- Gamma correction has no major effect on image
  - Gamma correction: adjusting gain due to power-law devices

Gradient Calc

- Several gradient detectors tried
  - [1,-1], [1,0,-1], [1,-8,0,8,-1], Sobel
  - Test unfiltered and pre-filtered with Gaussian smoothing
- Simplest [1,0,-1] proved best
- Gaussian smoothing affected results negatively
- For color images
  - Compute each channel separately
  - Choose the largest value as the gradient for that pixel

Orientation Binning

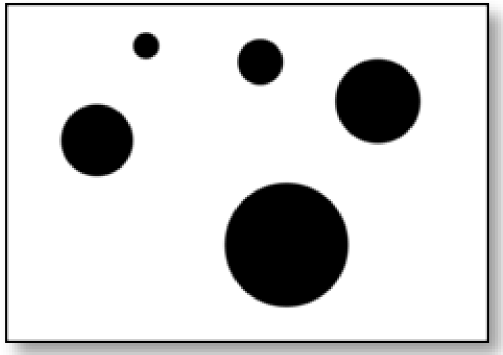
- Sampling of gradient orientation:
  - Highest possible: 360 bins => 1°/bin
  - n bins => 360°/n degrees/bin
  - 0-180° can be used to ignore negative directions and simplify computation
- Compute histogram for gradient orientations for each cell
  - Bin orientations
  - Less bins => better invariance
- Optimum: 9 bins 0-180° or 18 bins 0-360°

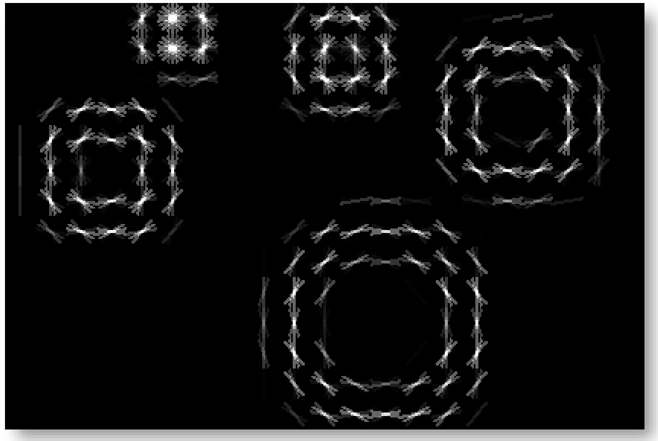


0.5
0.2
0.1
0.05
0.02
0.01
miss rate
bin= 9 (0−180)
bin= 6 (0−180)
bin= 4 (0−180)
bin= 3 (0−180)
bin=18 (0−360)
bin=12 (0−360)
bin= 8 (0−360)
bin= 6 (0−360)
10^{-6} 10^{-5} 10^{-4} 10^{-3} 10^{-2} 10^{-1}
false positives per window (FPPW)

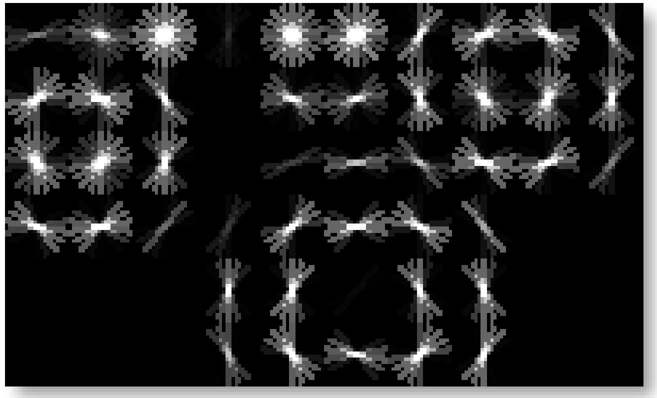
Partitioning: Cell

- Image is partitioned into cells: nxn pixels
  - Smaller n => better gradient resolution
  - The histogram is computed for each cell
  - Histograms from adjacent cells are used to create a vector



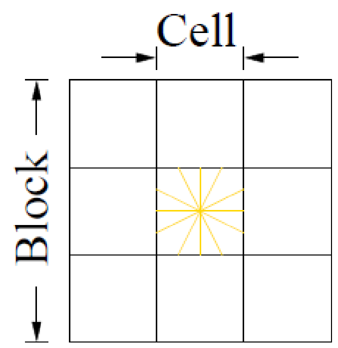


10x10 cells

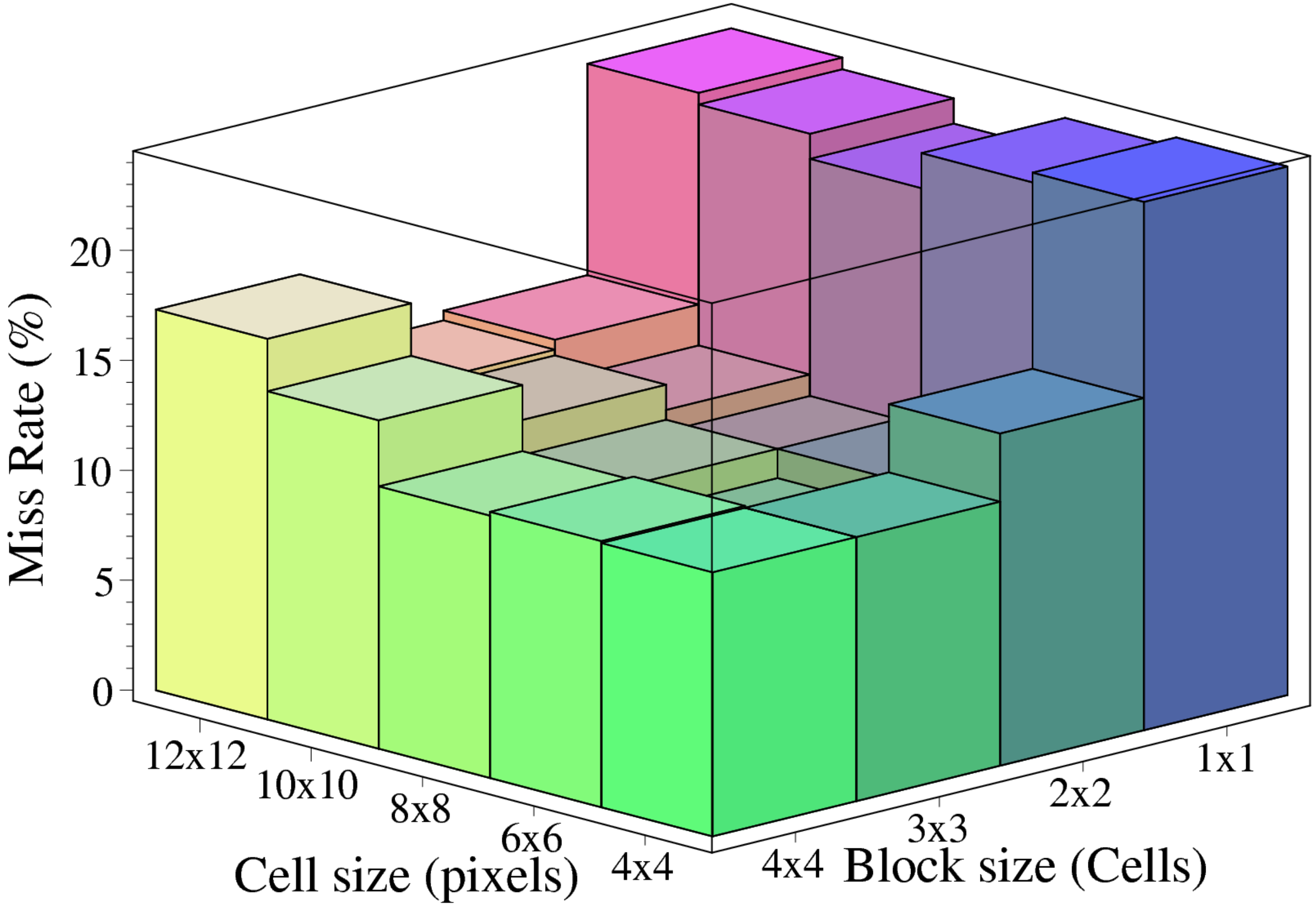


20x20 cells

Partitioning: Block



- Rectangular HOG (R-HOG)
- mxm cells are grouped together to form blocks
  - Local HOG
  - Larger m => better invariance to change
- Optimum for pedestrian images:
  - 6x6 cell size
  - 3x3 block size
  - May vary for other images and image types


Miss Rate (%)
0
5
10
15
20
Cell size (pixels)
12x12
10x10
8x8
6x6
4x4
Block size (Cells)
4x4
3x3
2x2
1x1

Bilinear Smoothing: Within a Cell

- Used to reducing aliasing
  - Aliasing: interference due to improper sampling
- Bilinear smoothing: orientation contributes to the bins that it is closes to
- Example:

positive only, bins = 9

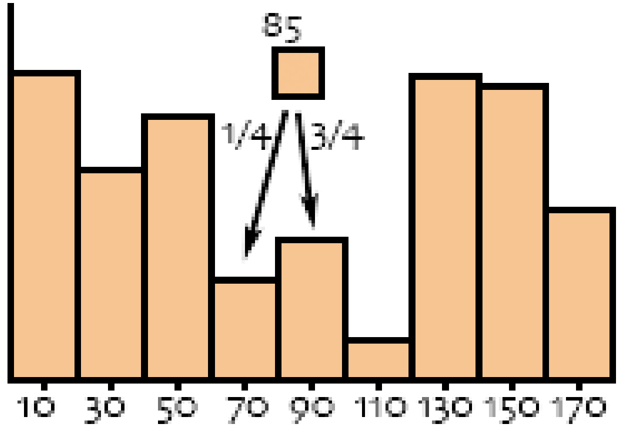
80/9 = 20°/bin.

Bins located at 10, 30, 50, 70, 90, 110°∠, …

if magnitude of grad i = $|G_i|$ = 10, and orientation = $\angle G_i$=85

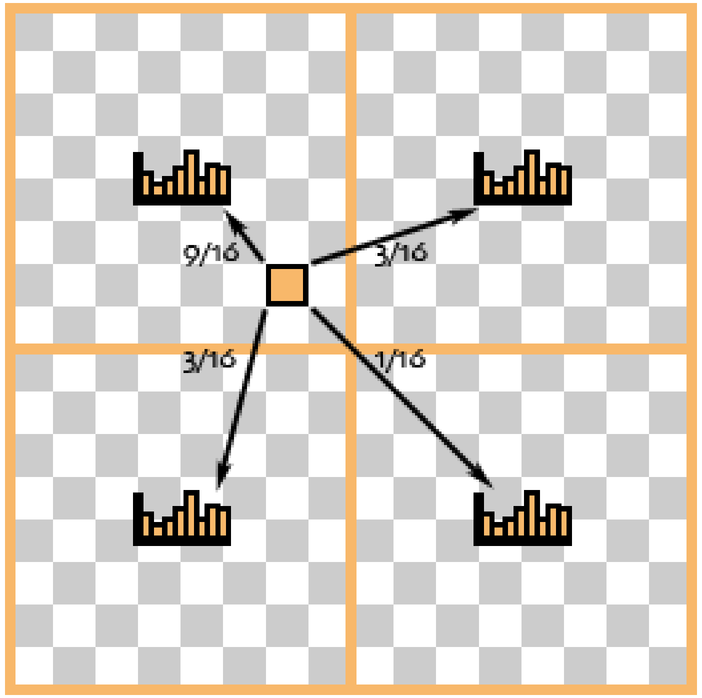
$\Rightarrow$ h(4) = [1-(85-70)/20]*10=2.5;

$\Rightarrow$ h(5) = [1-(90-85)/20]*10=7.5;



Trilinear Interpolation: Between Cells

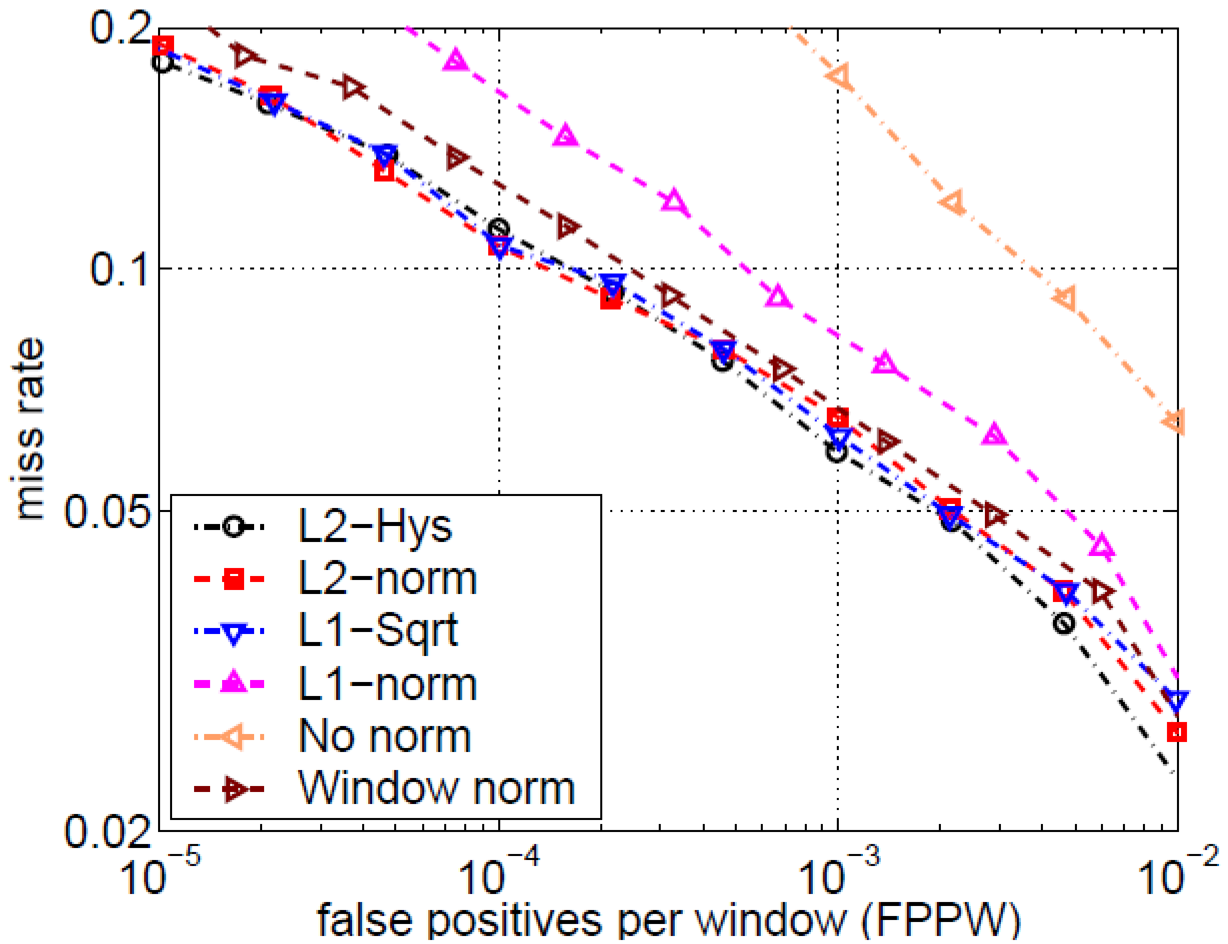
- Example: Given 8x8 cell
  - Centers are (5,5), (5,13), (13,5), (13,13)
  - Mag Gradient i is located at 7,7
  - Row, Column Distance to cell center:
    - Top-left: 2/8, 2/8
    - Top-right: 2/8, 5/8
    - Bottom-left: 6/8, 2/8
    - Bottom-right: 6/8, 6/8
  - Ratios: (1-r)(1-c)
    - Top-left: 6/8*6/8 = 36/64 = 9/16
    - Top-right: 6/8*2/8 = 12/64 = 3/16
    - Bottom-left: 2/8*6/8 = 12/64 = 3/16
    - Bottom-right: 2/8*2/8 = 4/64   = 1/16
- Slow!



Block Normalization

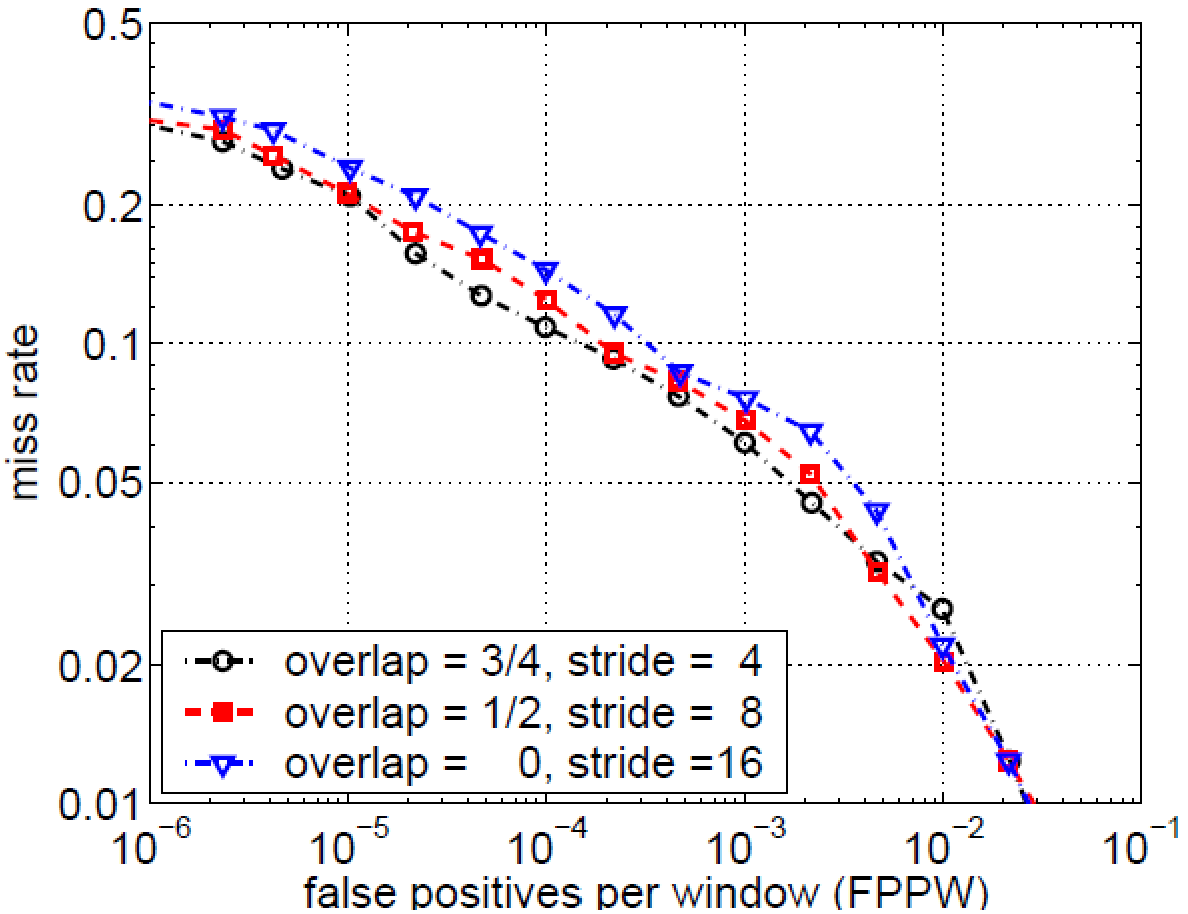
- Gradient magnitude is affected by illumination
  - Low contrast/illumination regions yield small values
  - High intensity regions dominant values
- Normalization:
  - Low contrast regions are stretched
  - High intensity regions are reduced
  - L1-norm: divide by sum of values: $\dfrac{v}{\sqrt{|v|_1 + \epsilon}}$
  - L2-norm: divide by sqrt(sum {values^2}): $\dfrac{v}{\sqrt{|v|_2^2 + \epsilon}}$
  - v = vector concatenated histograms of all cells in a block

Effect of Normalization

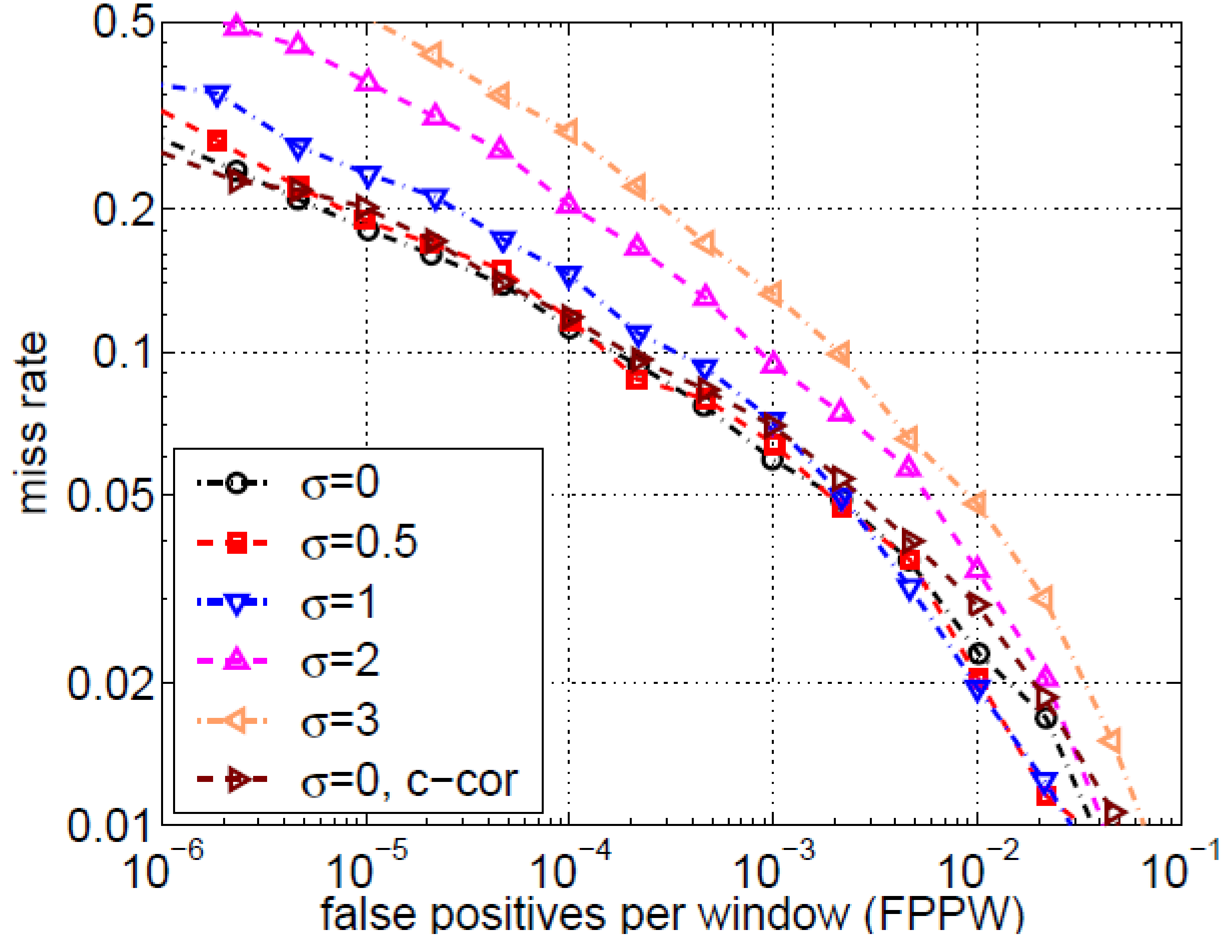


Cell Overlap

- Blocks can overlap
  - Improves performance, but descriptor size increases
  - Ensures consistency across the whole image without the loss of local variations
- Overlap: sliding block
  - 0% overlap: blocks have no shared cells
  - 50% overlap: blocks share 50% of cells with neighboring blocks

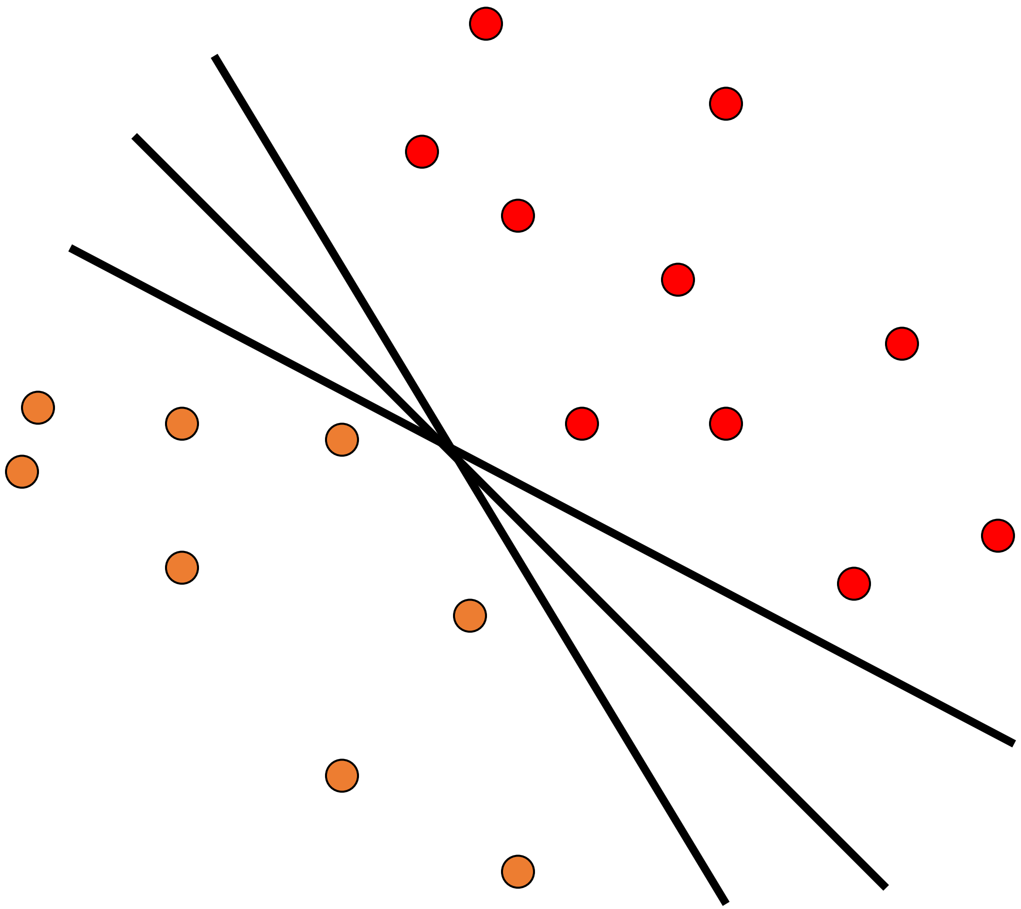


Effect of Scale



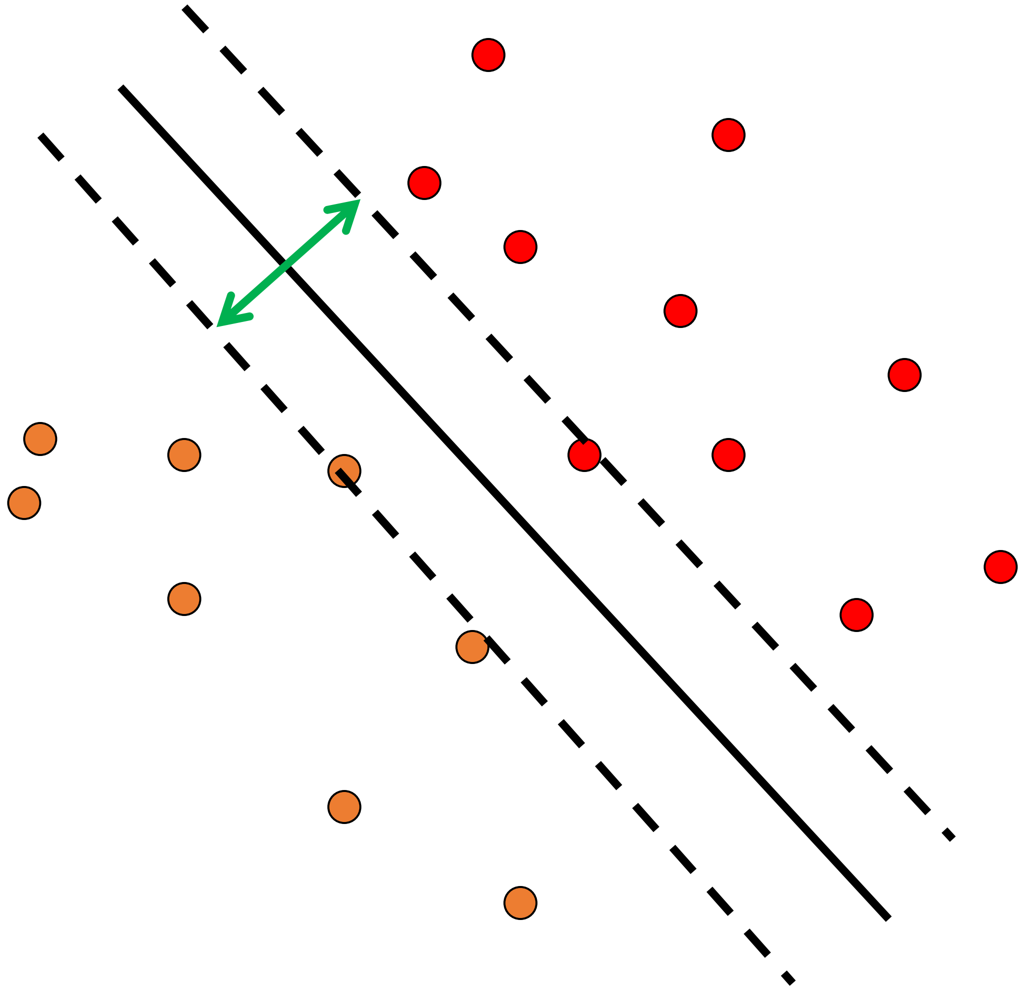
Linear Classifier

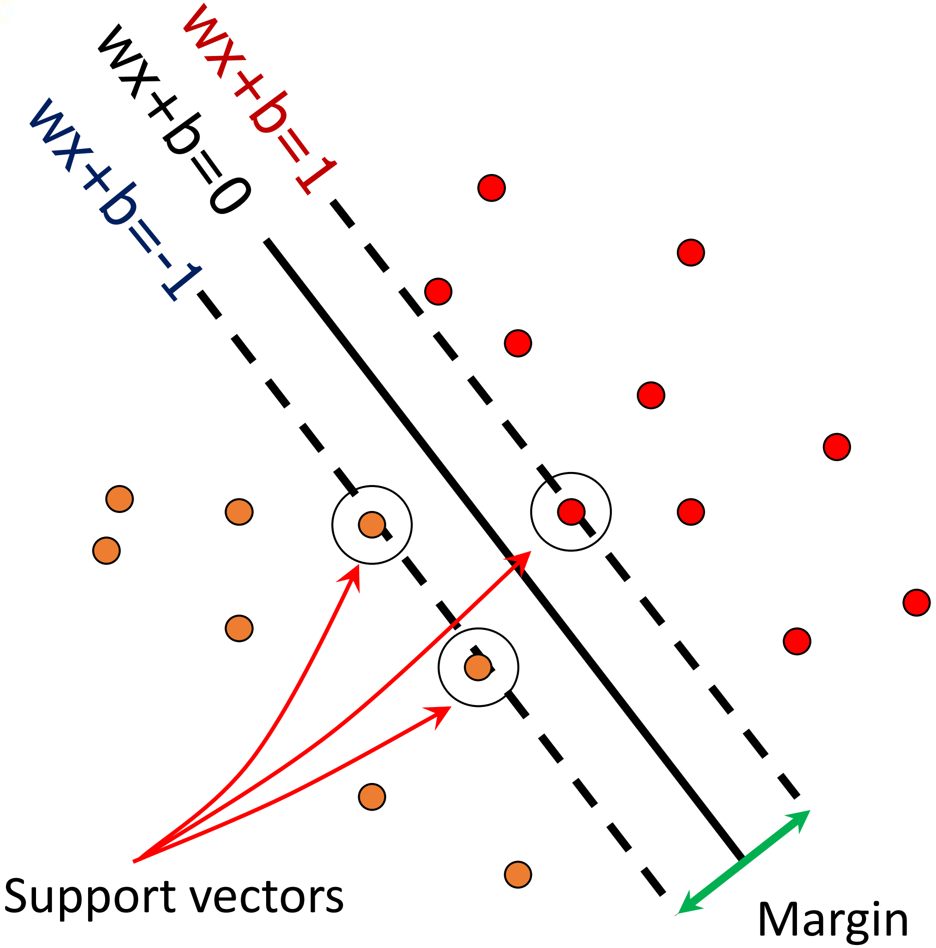
- Find linear function to separate positive and negative examples
- Support Vector Machines:
  - Maximize the margin between the positive and negative training examples
- Hard clipping of SVM scores gives the best results than simple probabilistic mapping of scores



$\mathbf{x}_i$ positive : $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$

$\mathbf{x}_i$ negative : $\quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$





$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

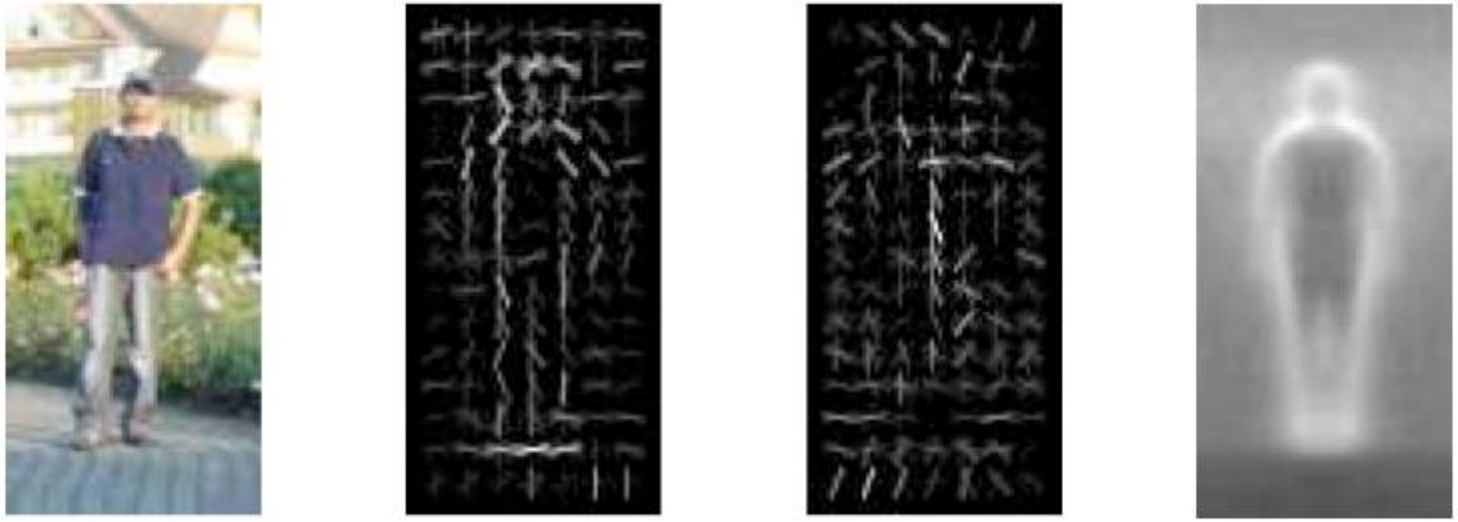
For support, vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

People Detection

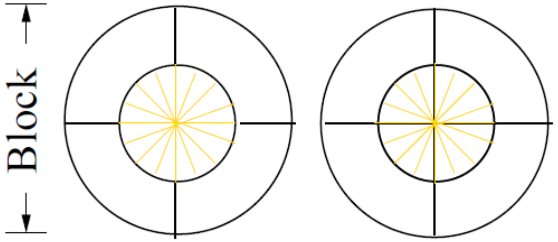
- Detect & localize upright people in static images
  - Main assumption: upright fully visible people
- Challenges
  - Wide variety of articulated poses
  - Variable appearance/clothing
  - Complex backgrounds
  - Unconstrained illumination
  - Occlusions, different scales
- Applications
  - Pedestrian detection for smart cars
  - Film & media analysis
  - Visual surveillance

Database

- http://pascal.inrialpes.fr/data/human/
- Training
  - 614 positive images
  - 1218 negative images
- Testing
  - 288 positive images
  - 453 negative images
- 90% at 10^{-4} false positives per window
- Slower than adaboost (Viola & Jones)

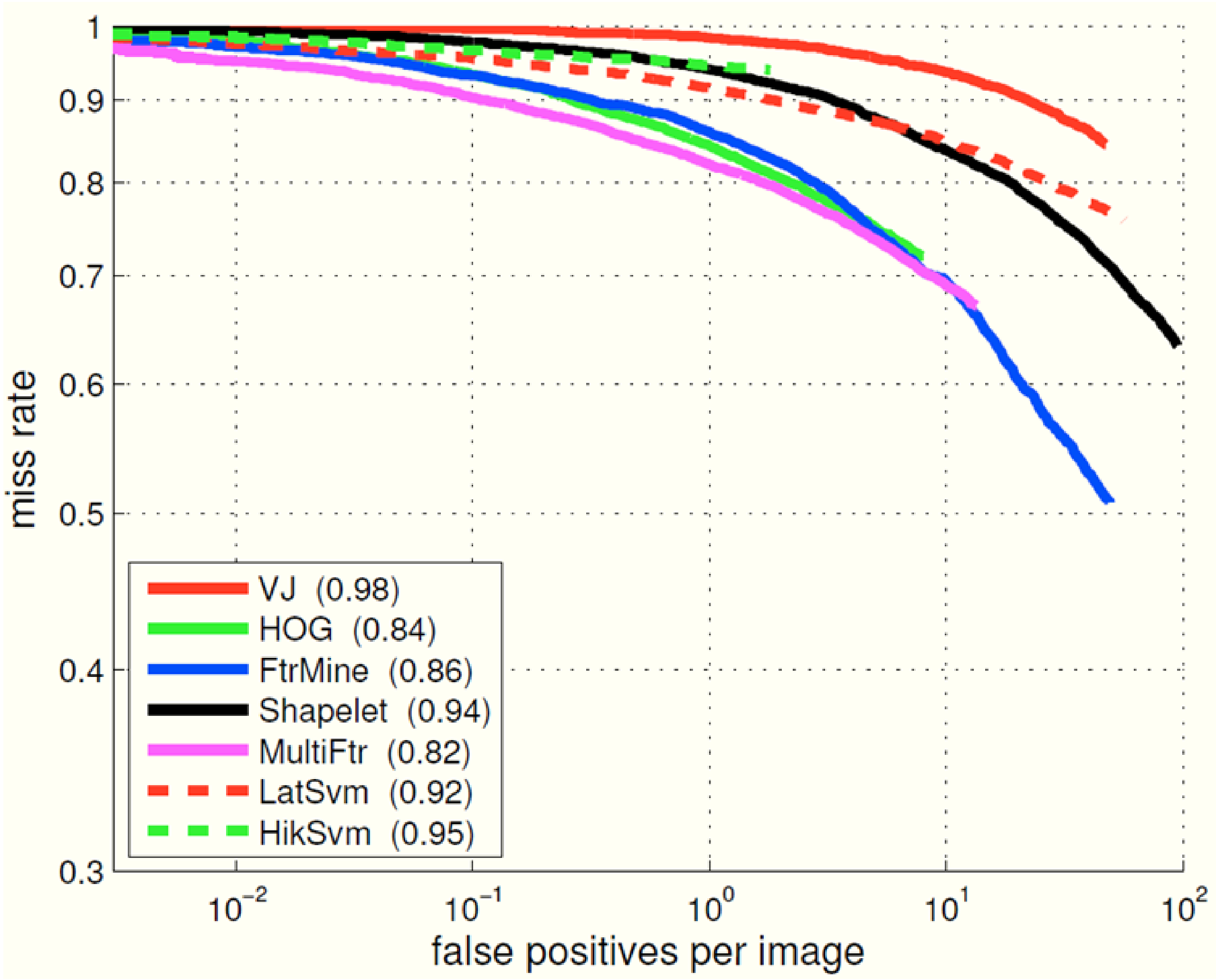


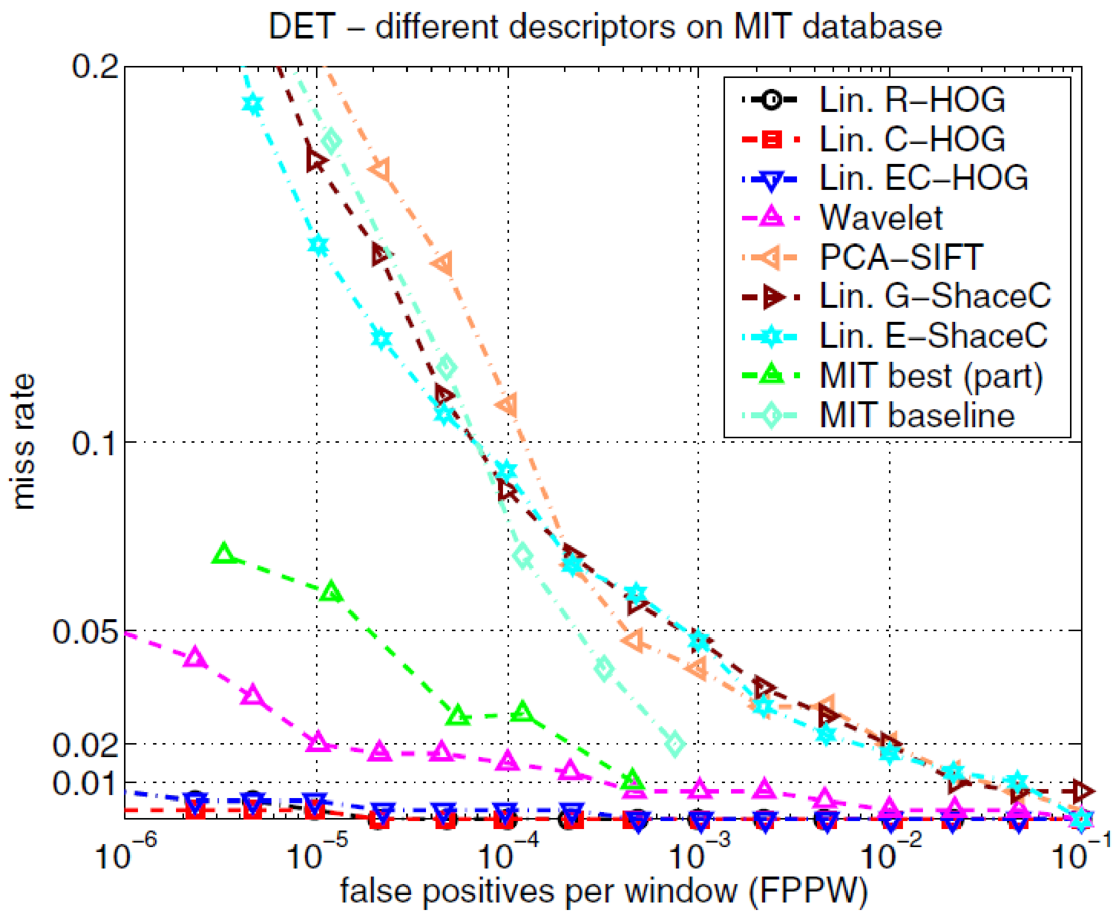
Improvements



- Circular HOG (C-HOG)
  - Circular arrangement of cells
- Multi-scale resolution

Comparisons




DET – different descriptors on MIT database
miss rate
false positives per window (FPPW)
0.2
0.1
0.05
0.02
0.01
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
Lin. R–HOG
Lin. C–HOG
Lin. EC–HOG
Wavelet
PCA–SIFT
Lin. G–ShaceC
Lin. E–ShaceC
MIT best (part)
MIT baseline