DETECTION AS A BINARY DECISION

Presented by Majd Srour

Agenda

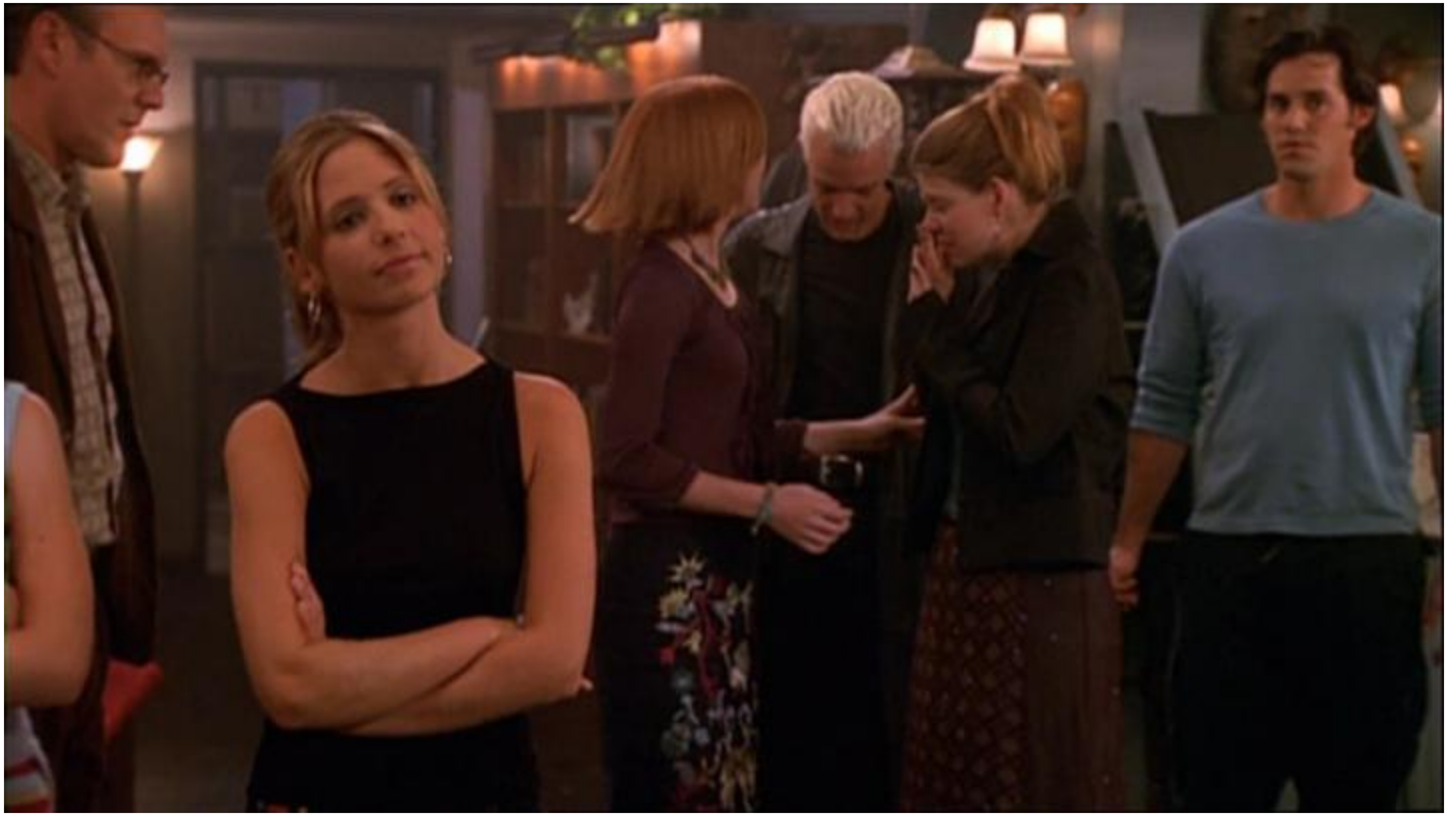
- Histograms of Oriented Gradients for Human Detection
- Rapid Object Detection using a Boosted Cascade of Simple Features

HISTOGRAMS OF ORIENTED GRADIENTS FOR HUMAN DETECTION

Navneet Dalal and Bill Triggs

Presented by Majd Srour

Human Detection



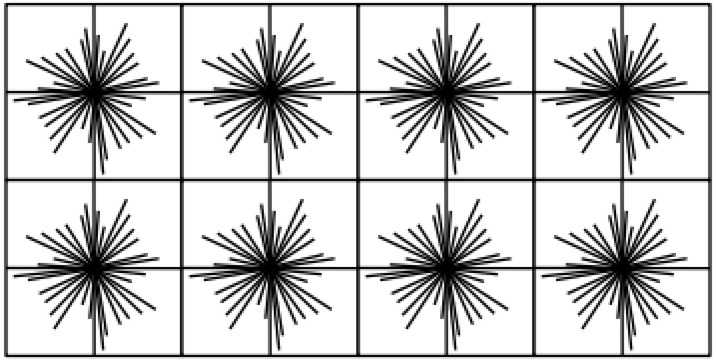
- Find all objects of interest
- Enclose them tightly in a bounding box.

Human Detection

- Find all objects of interest
- Enclose them tightly in a bounding box.

Histograms of Oriented Gradients

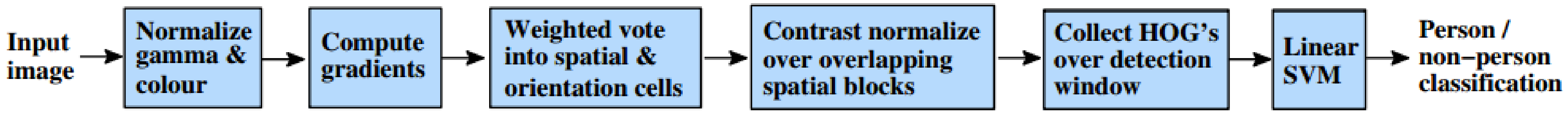
- Objective: object recognition
- Basic idea
  - Local shape information often well described by the distribution of intensity gradients or edge directions even without the precise information about the location of the edges themselves



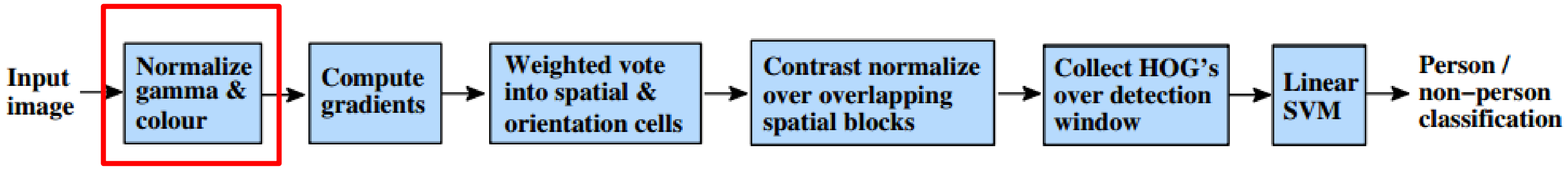
Algorithm Overview

- Divide image into small sub-images: “cells”
  - Cells can be rectangle (R-HOG) or circular (C-HOG)
- Accumulate a histogram of edge orientations (gradients) within that cell
- The combined histogram entries are used as the feature vector describing the object
- To provide better illumination invariance (lighting, shadows, etc.) normalize the cells across larger regions incorporating multiple cells: “blocks”

Algorithm Overview



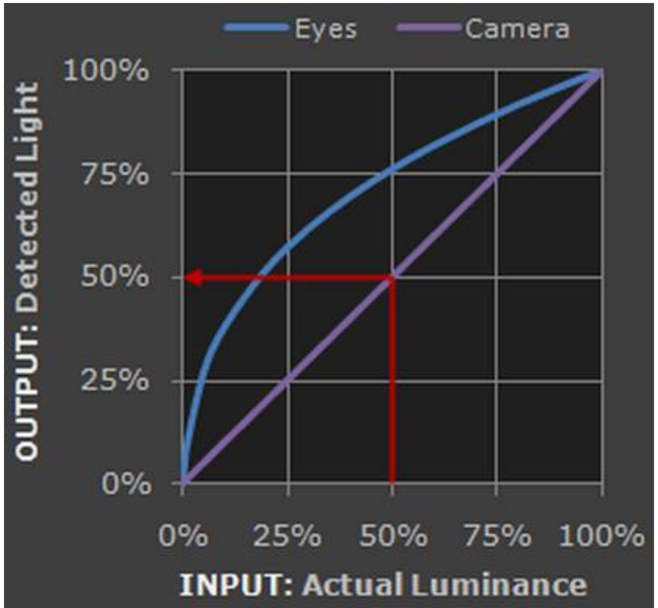
Gamma Correction



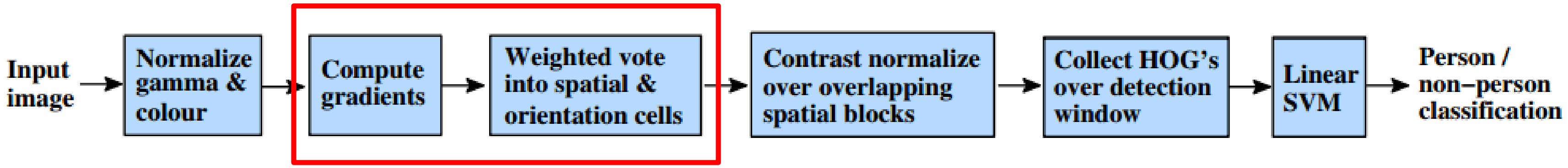
Gamma Correction

- $\mathbf{V_{out} = V_{in}^{gamma}}$
- where V_{out} is the output luminance value and V_{in} is the input/actual luminance value. This formula causes the blue line below to curve. When gamma<1, the line arches upward, whereas the opposite occurs with gamma>1.

- In HOG, Square root gamma (gamma = 0.5), compression of each pixel improves performance by 1%



Gradient Histograms



Gradient Histograms

- The HOG person detector uses a detection window that is 64 pixels wide by 128 pixels tall.

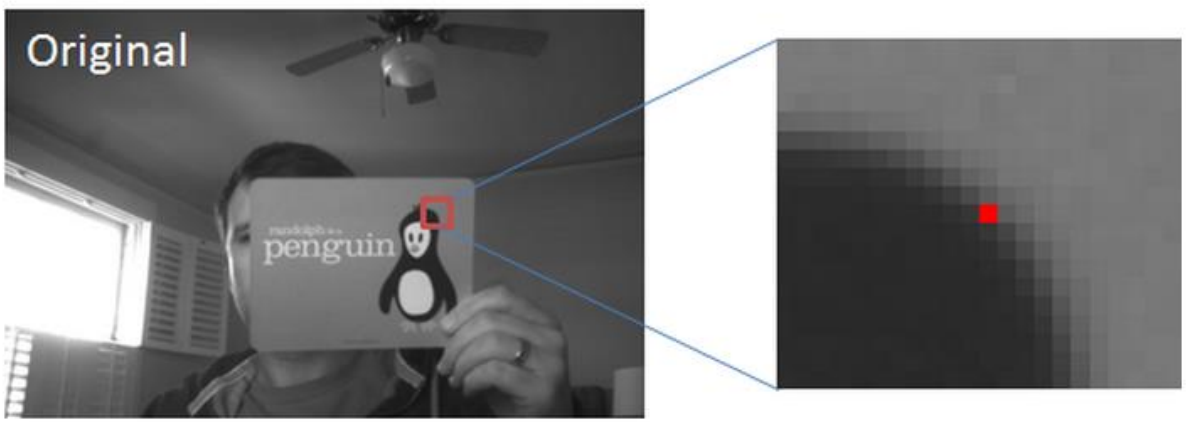
- To computer the HOG descriptor, we operate on a 8×8 pixel cells within the detection window.

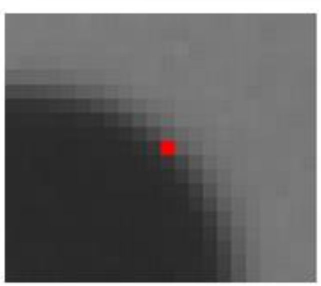
Gradient Histograms

- Zoomed-in version with an 8×8 cell drawn in red.

Gradient Histograms – Gradient Calculation

- Within a cell, we compute the gradient vector at each pixel. Several masks and smoothing scales were tested for calculating the gradient. A simple 1-D mask [1- 0 1] with σ=0 (no smoothing) works best.





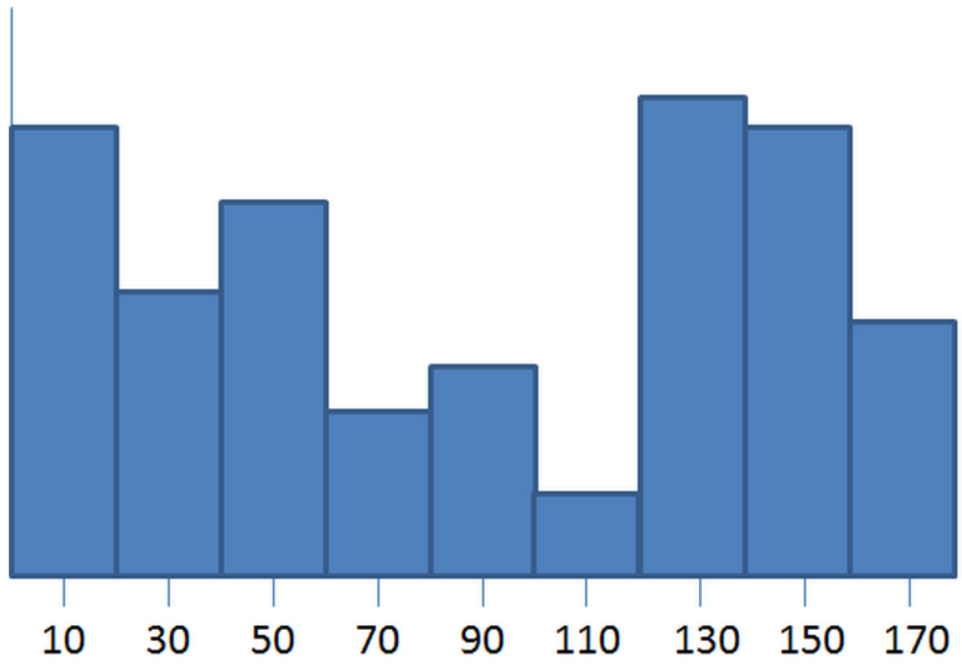
	93	
56		94
	55	

$$\nabla f = \begin{bmatrix} 38 \\ 38 \end{bmatrix}$$

$$|\nabla f| = \sqrt{(38)^2 + (38)^2} = 53.74$$

Gradient Histogram – Histogram Calc.

- We take the 64 gradient vectors (in the 8×8 cell) and put them in a 9-bin histogram. The histogram ranges from 0-180 (unsigned) degrees, so there are 20 degrees per bin:

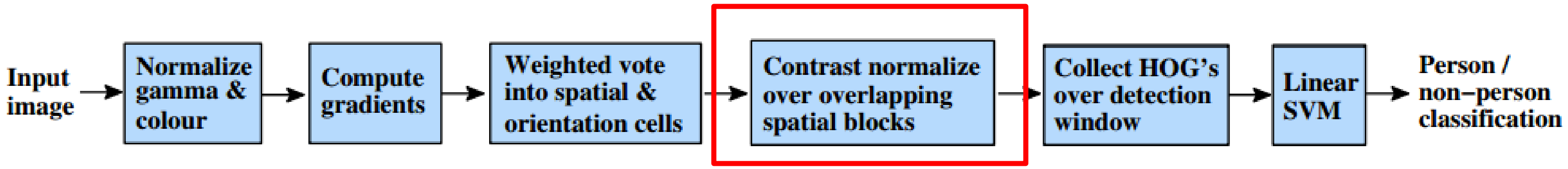


Gradient Histogram – Histogram Calc.

- For each gradient vector, it’s contribution to the histogram is given by the magnitude of the vector (so stronger gradients have a bigger impact on the histogram)

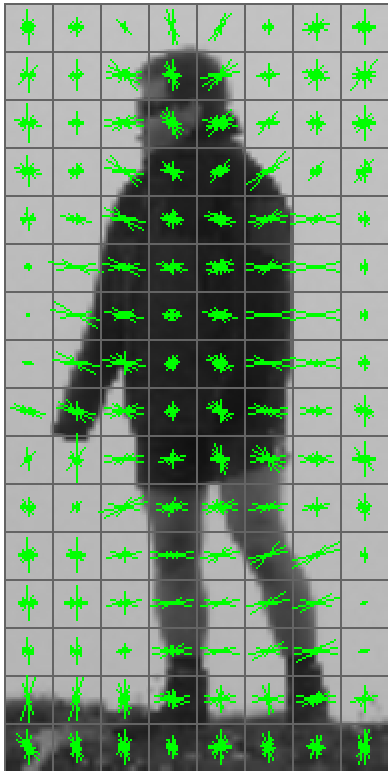
- We split the contribution between the two closest bins. So, for example, if a gradient vector has an angle of 85 degrees, then we add 1/4th of its magnitude to the bin centered at 70 degrees, and 3/4ths of its magnitude to the bin centered at 90.

Normalizing Gradient Vectors



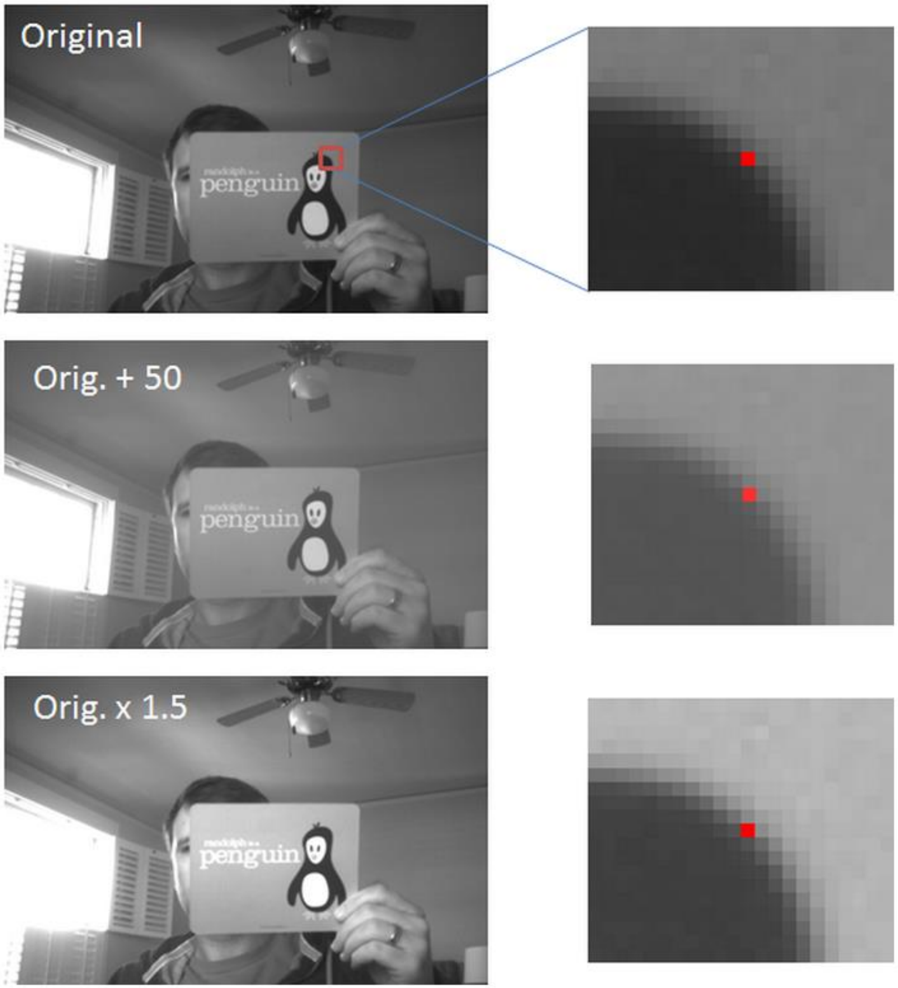
Normalizing Gradient Vectors

- The next step in computing the descriptors is to normalize the histograms. Let’s take a moment to first look at the effect of normalizing gradient vectors in general.

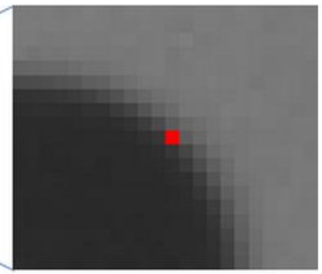


- **Adding** or **subtracting** a fixed amount of brightness to every pixel in the image, and you’ll still get the same gradient vectors at every pixel.
- It turns out that by normalizing your gradient vectors, you can also make them invariant to **multiplications** of the pixel values

Normalizing Gradient Vectors



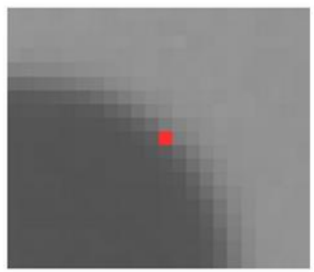
Normalizing Gradient Vectors



	93	
56		94
	55	

$$\nabla f = \begin{bmatrix} 38 \\ 38 \end{bmatrix}$$

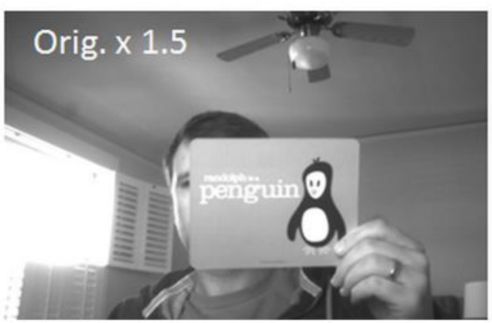
$$|\nabla f| = \sqrt{(38)^2 + (38)^2} = 53.74$$

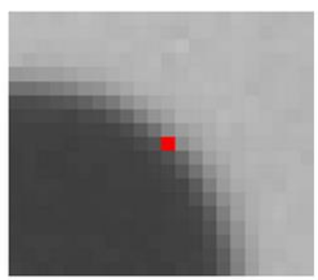


	143	
106		144
	105	

$$\nabla f = \begin{bmatrix} 38 \\ 38 \end{bmatrix}$$

$$|\nabla f| = \sqrt{(38)^2 + (38)^2} = 53.74$$





	140	
84		141
	83	

$$\nabla f = \begin{bmatrix} 57 \\ 57 \end{bmatrix}$$

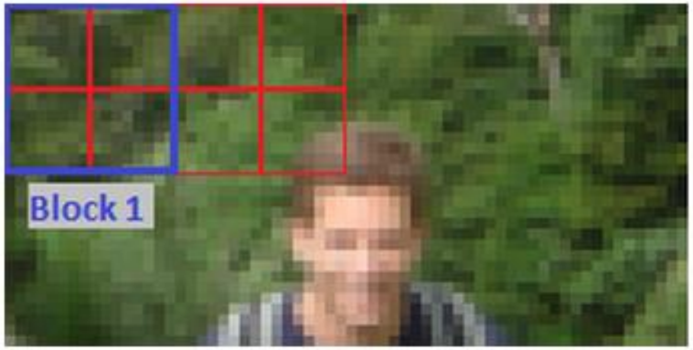
$$|\nabla f| = \sqrt{(57)^2 + (57)^2} = 80.61$$

Normalizing Gradient Vectors

- If you divide all three vectors by their respective magnitudes, you get the same result for all three vectors: [0.71 0.71]‘.
- invariant (or at least more robust) to changes in contrast.
- normalizing the vector to unit length
- does not affect its orientation, only the magnitude.

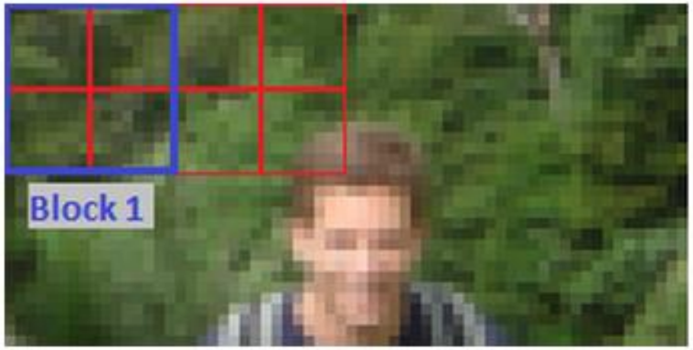
Histogram Normalization

- Rather than normalize each histogram individually, the cells are first grouped into blocks and normalized based on all histograms in the block.
- The blocks used by Dalal and Triggs consisted of 2 cells by 2 cells. The blocks have “50% overlap”, which is best described through the illustration below.

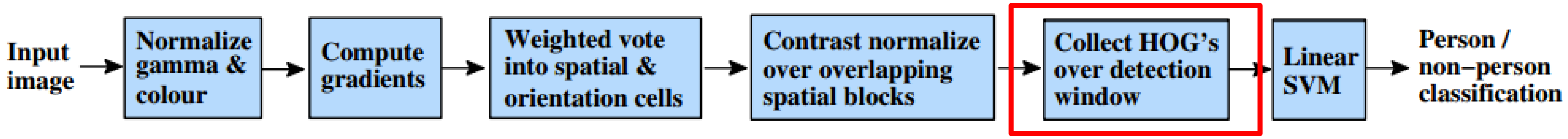


Histogram Normalization

- This block normalization is performed by concatenating the histograms of the four cells within the block into a vector with 36 components (4 histograms x 9 bins per histogram). Divide this vector by its magnitude to normalize it.



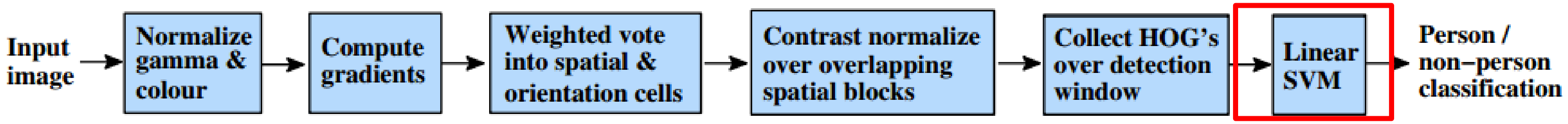
Normalizing Gradient Vectors



Final Descriptor Size

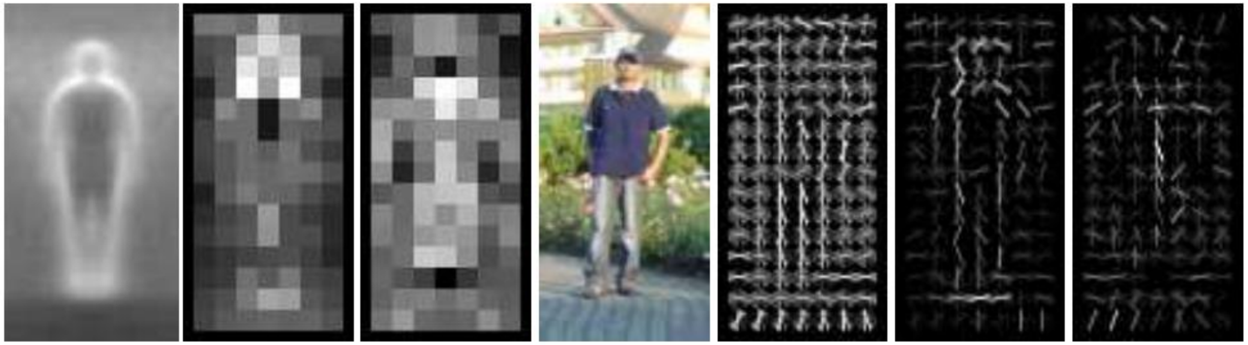
- The 64 x 128 pixel detection window will be divided into 7 blocks across and 15 blocks vertically
- Total of 105 blocks
- Each block contains 4 cells.
- A 9-bin histogram for each cell
- This brings the final descriptor size of 3780 values.

Normalizing Gradient Vectors

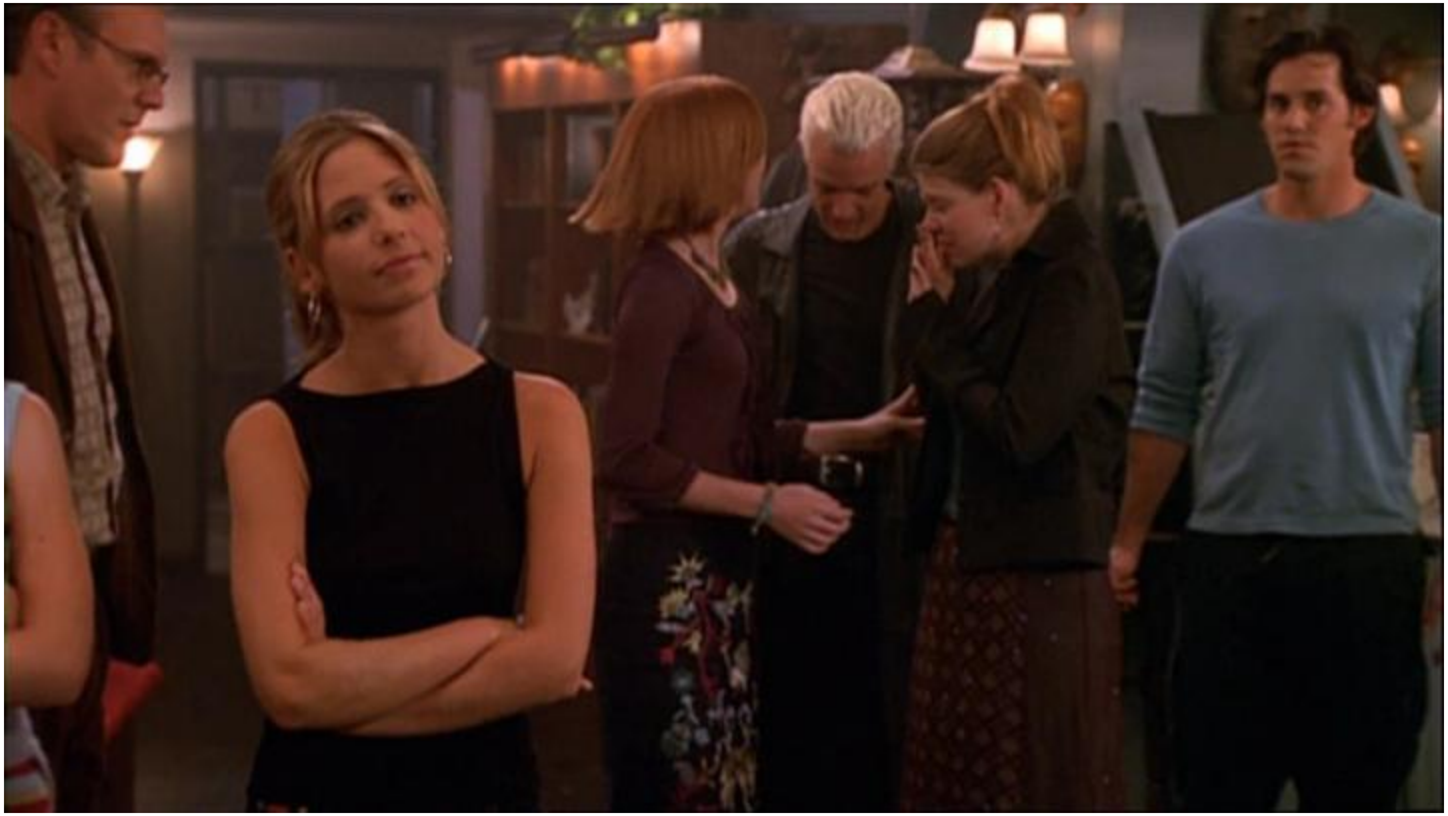


Classifier - Linear SVM

- The final step : feed the descriptors into some recognition system based on supervised learning.
- The Support Vector Machine (SVM) classifier is a binary classifier which looks for an optimal hyperplane as a decision function
- SVM classifier can make decisions regarding the presence of an object in additional test images.

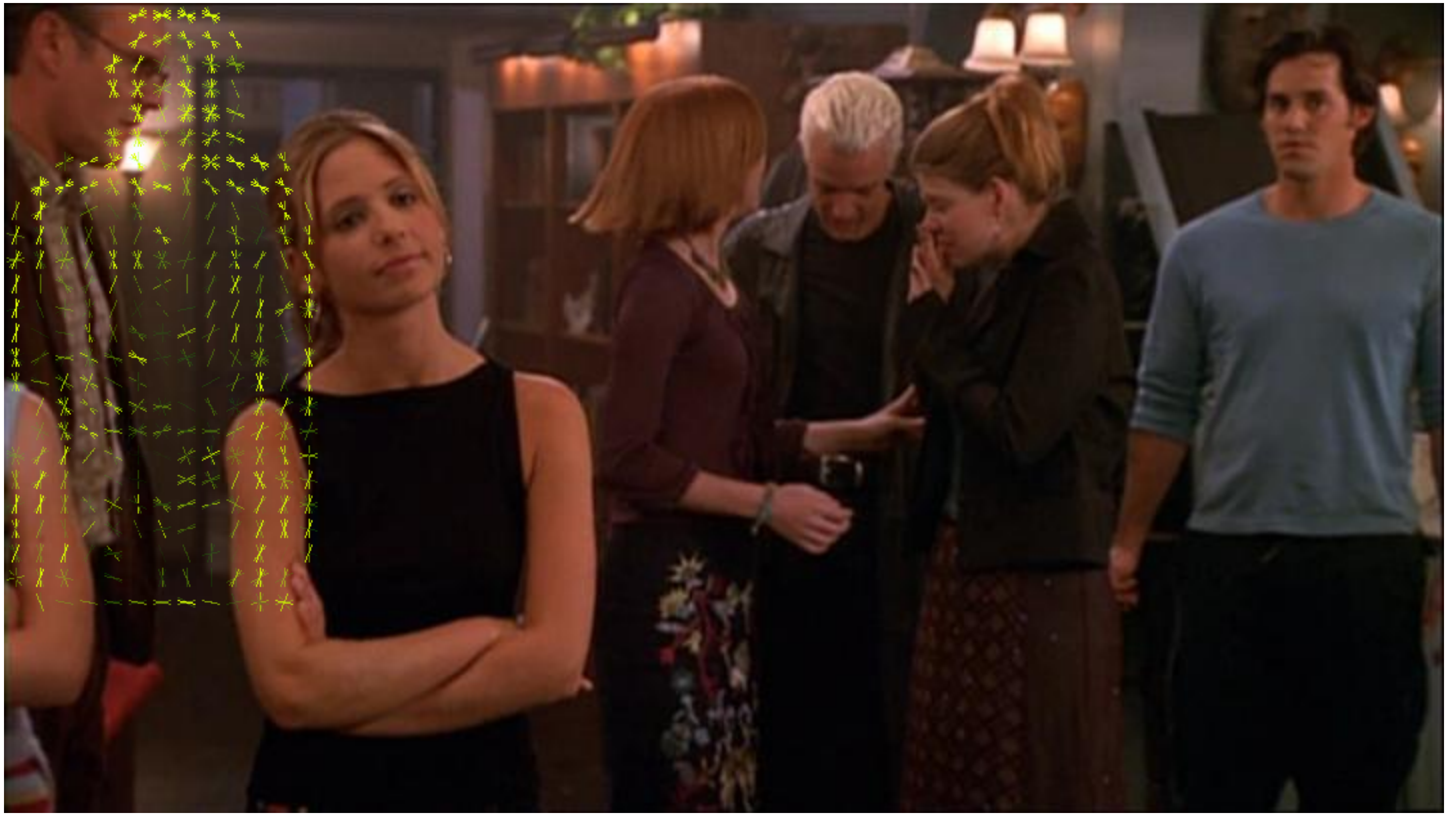


HOG Detector



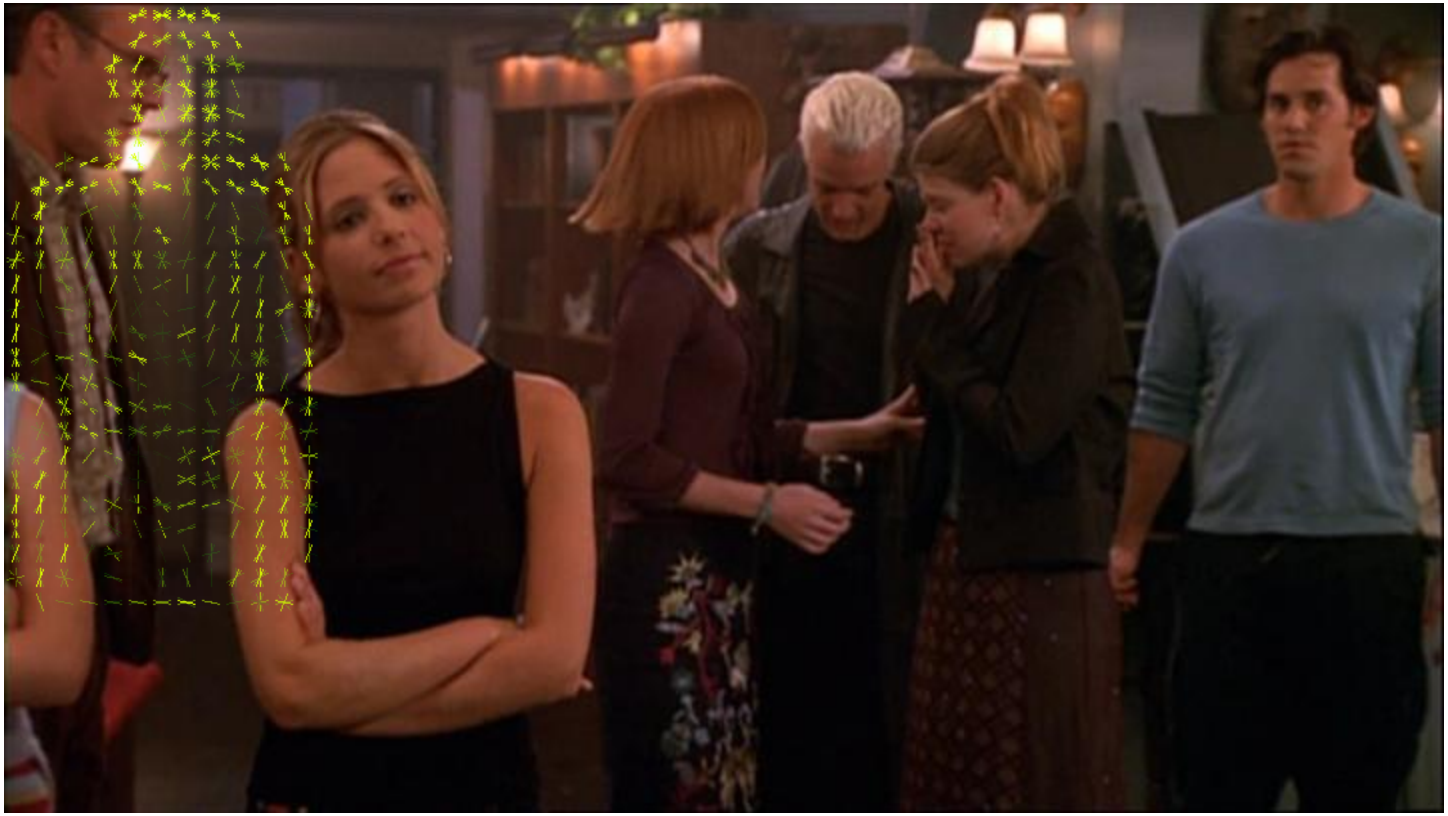
- Sliding window using learnt HOG template

HOG Detector



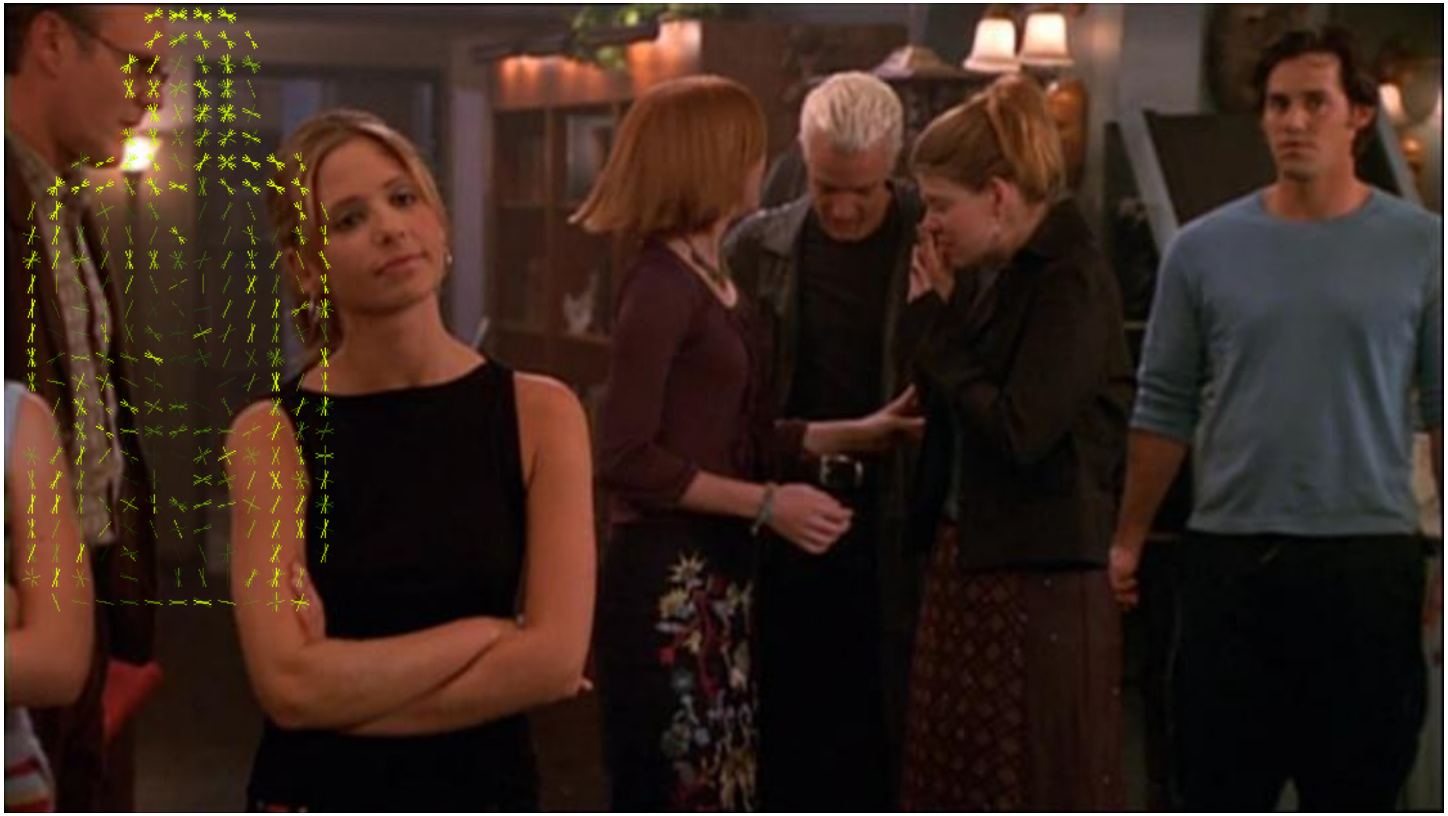
- Sliding window using learnt HOG template

HOG Detector



- Sliding window using learnt HOG template

HOG Detector



- Sliding window using learnt HOG template

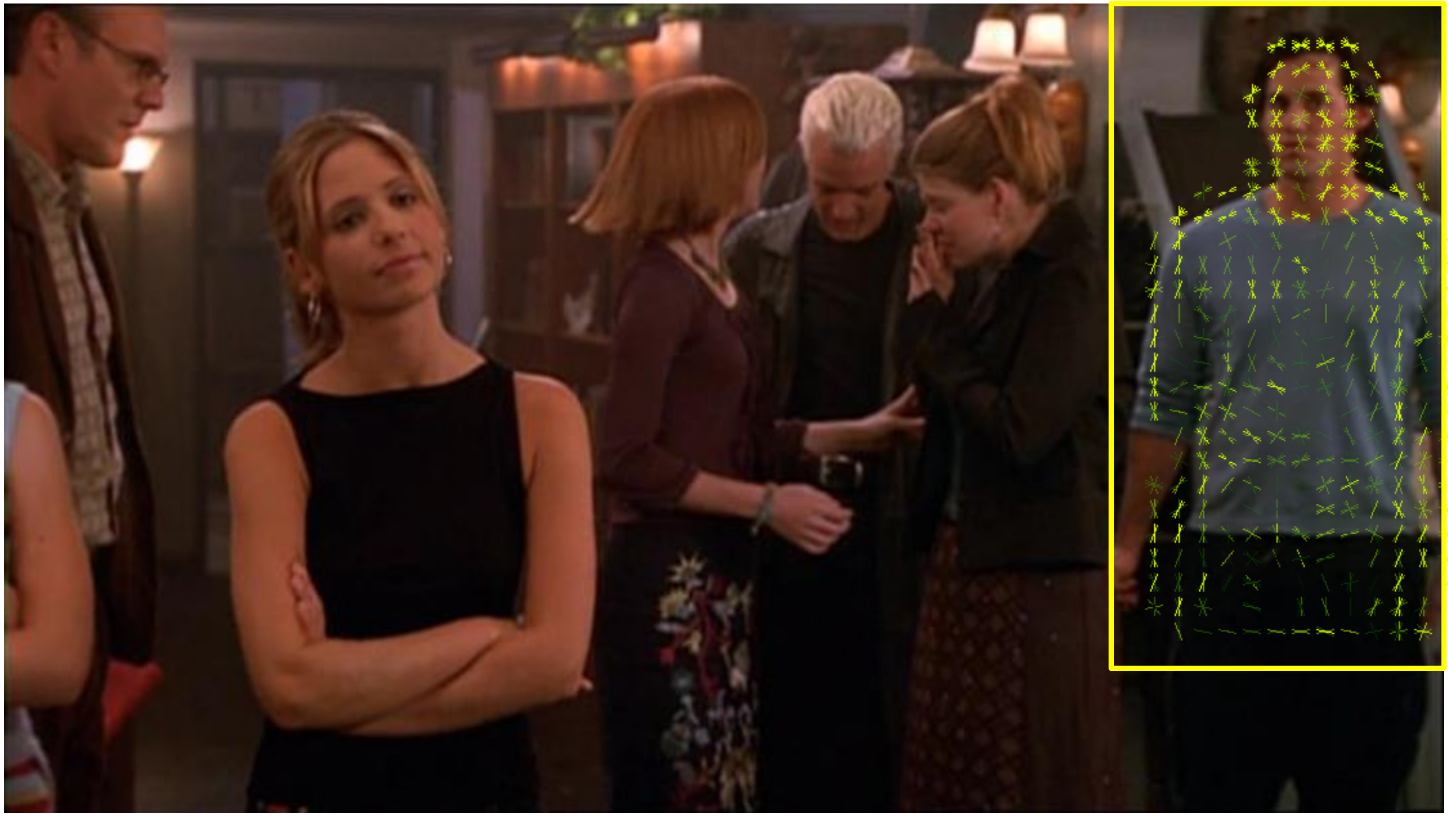
HOG Detector

- Sliding window using learnt HOG template

HOG Detector

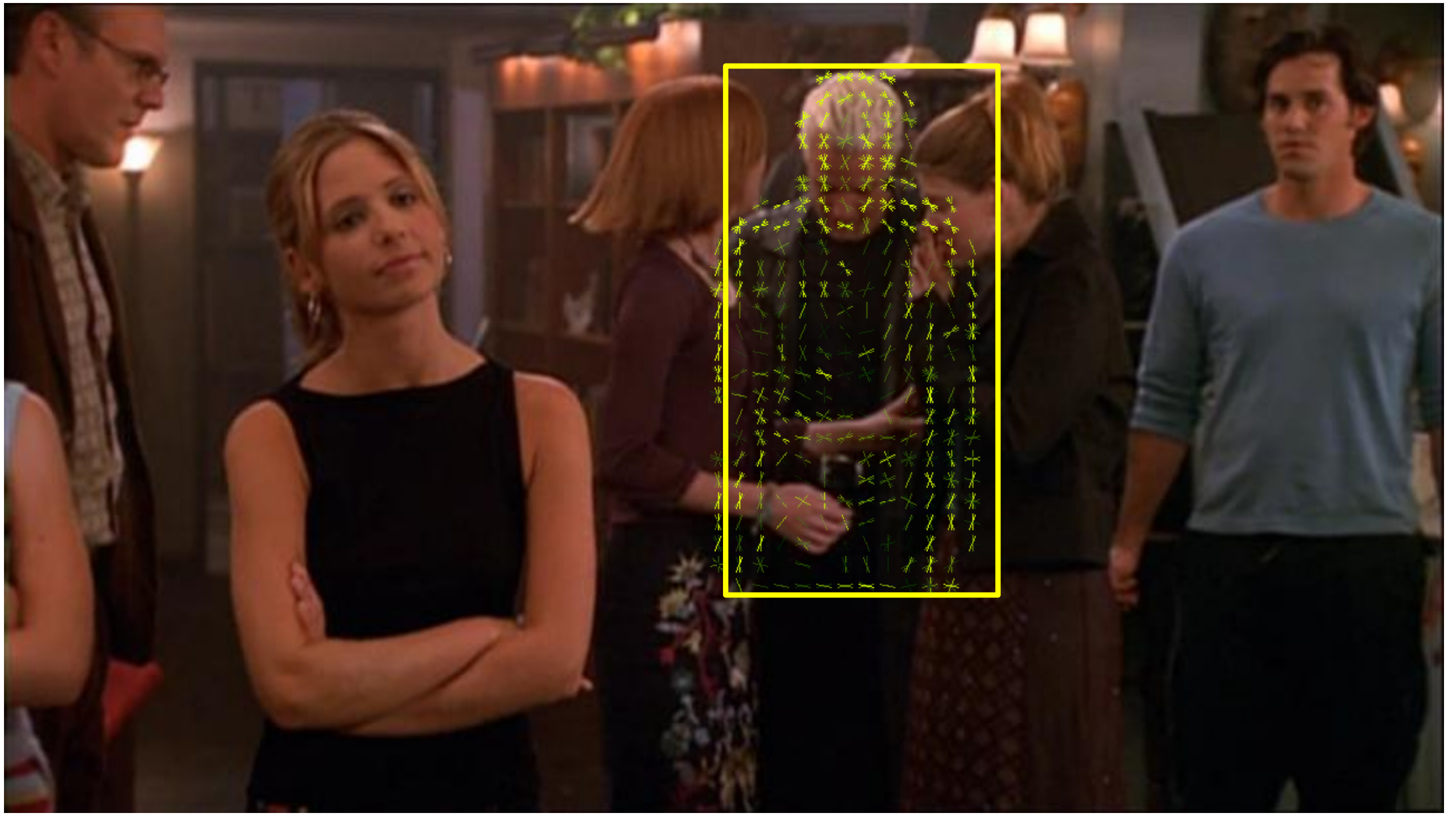
- Sliding window using learnt HOG template

HOG Detector



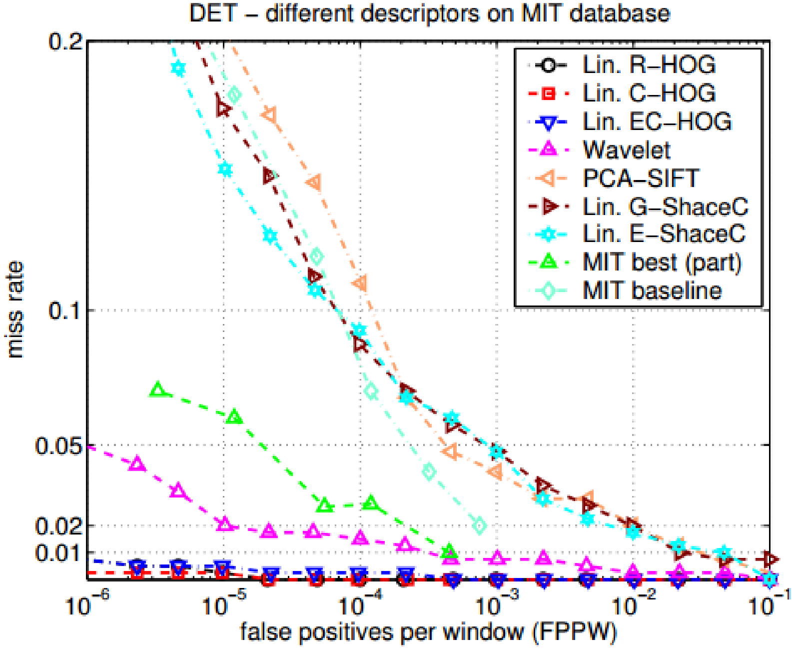
- Sliding window using learnt HOG template

HOG Detector

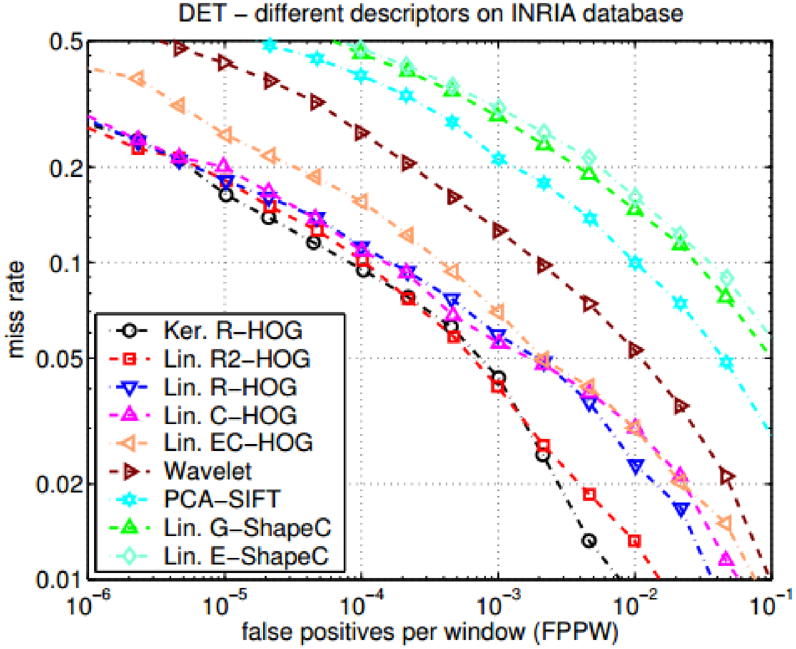


- Sliding window using learnt HOG template

Testing

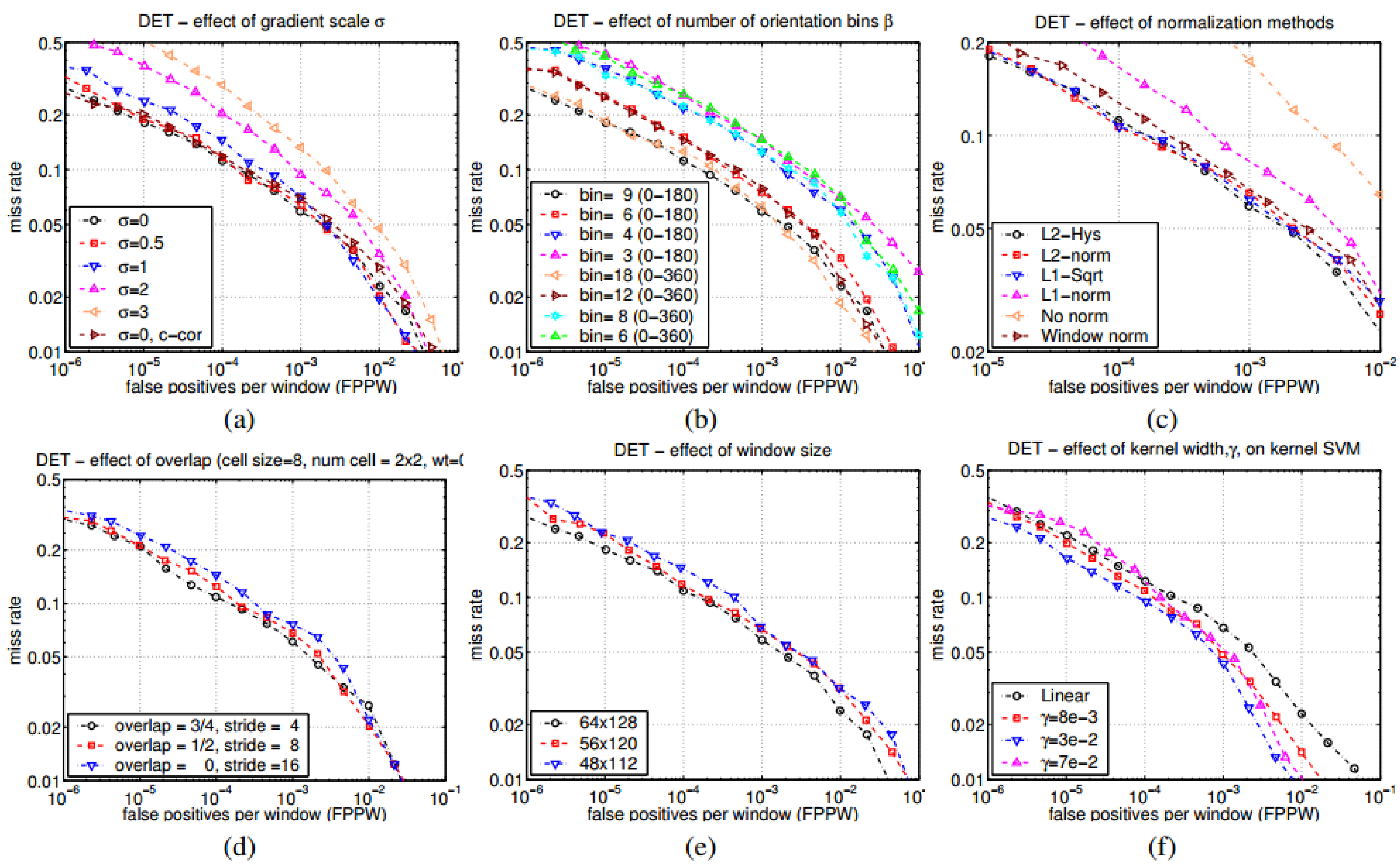


DET – different descriptors on MIT database
miss rate
false positives per window (FPPW)
Lin. R–HOG
Lin. C–HOG
Lin. EC–HOG
Wavelet
PCA–SIFT
Lin. G–ShaceC
Lin. E–ShaceC
MIT best (part)
MIT baseline



DET – different descriptors on INRIA database
miss rate
false positives per window (FPPW)
Ker. R–HOG
Lin. R2–HOG
Lin. R–HOG
Lin. C–HOG
Lin. EC–HOG
Wavelet
PCA–SIFT
Lin. G–ShapeC
Lin. E–ShapeC

Testing – Cont.



- Resources:

- Histograms of Oriented Gradients for Human Detection - Navneet Dalal and Bill Triggs

- Wikipedia on SVM

- Oxford Brookes Vision group

RAPID OBJECT DETECTION USING A BOOSTED CASCADE OF SIMPLE FEATURES

Paul Viola and Michael Jones

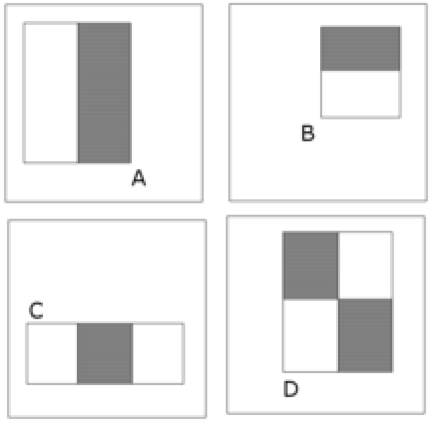
Presented by Majd Srour

Face Detection

- Determining the locations and sizes of human faces in arbitrary images.

Features

- Three kinds of simple features are used.
    1. Two-Rectangles features
    2. Three-Rectangles features
    3. Four-Rectangles features



- Feature value Calculation

  ∑pixel values in white area - ∑pixel values in gray area

Key Contributions

- Three main contributions
  1. Introduction of Integral Image
  2. Learning algorithm based on AdaBoost
  3. Combine Classifiers in Cascade

Integral Image

- This concept was first introduced with this solution framework.
- Integral Image is computed from an image using few operations on pixels.

$$I(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} i(x', y')$$

y

x

10	20	10	20
20	10	10	10
30	10	10	20
10	20	30	20

Original Image

y

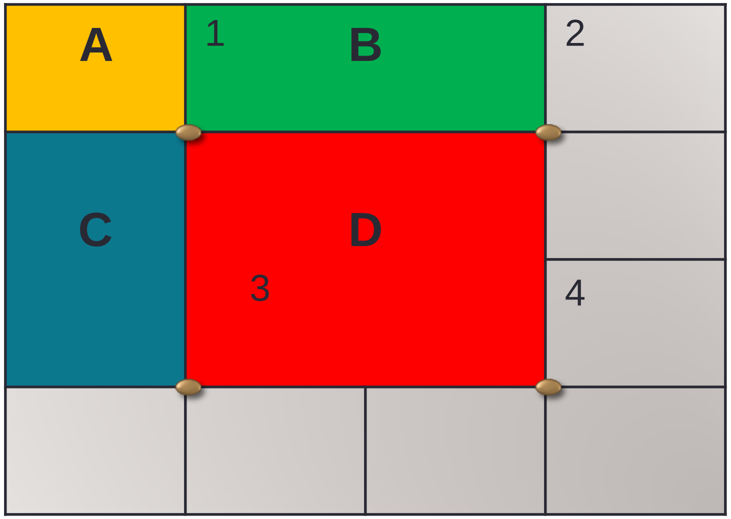
x

10	30	40	60
30	60	80	110
60	100	130	180
70	130	190	260

Integral Image

Integral Image

- Using Integral Image, pixel sum of a rectangle are can be calculated using 4 array references.
- It leads to a rapid evaluation of rectangle features
- Feature evaluation in constant time



$\sum$Pixel sum of area D= ii(4) + ii(1) – ii(2) –ii(3)

Learning Algorithm based on AdaBoost

- AdaBoost is used for feature selection and classifier training
- Capable of selecting a small set of good features from a large number of feature set
- AdaBoost use a set of weak learners to form a strong one
- It guarantees that training error of the strong classifier reach zero exponentially in number of rounds

Learning Algorithm based on AdaBoost

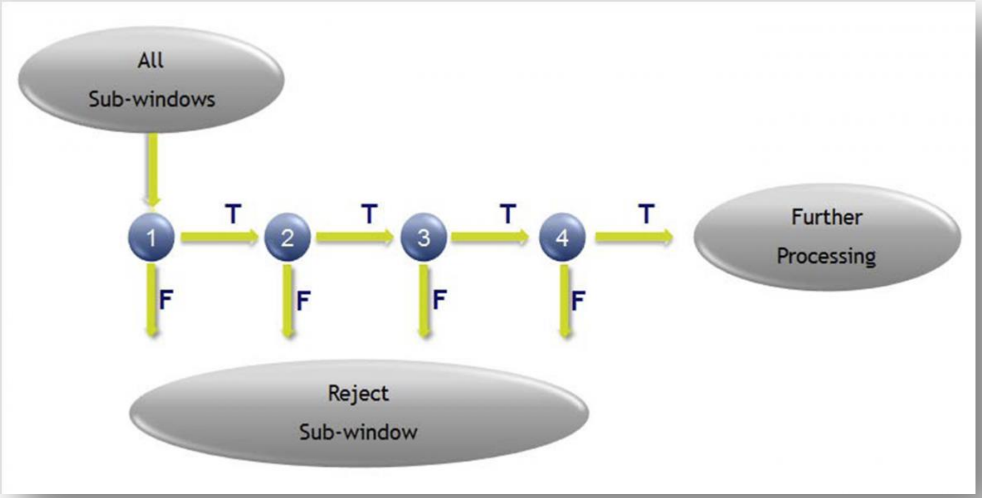
- A weak learner select a single rectangle feature which best separates positive and negative examples
- Weak learner determines the optimal threshold function, such that misclassification is minimized

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

- 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 $\alpha_t = \log \frac{1}{\beta_t}$

Combine Classifiers in Cascade

- Building cascade of classifiers,
  - Increase detection performance
  - Rapidly reduce computation power
- Simpler classifiers apply early and reject majority of sub windows, then apply complex classifiers to achieve low false positive
- Subsequent classifiers are trained using examples, which pass through all the previous stages



Combine Classifiers in Cascade

- Cascade Training process involves two trade-offs
  1. Classifier with more features will achieve higher DR and lower FPR
  2. Classifier with more features need more computations
- Can define a optimization framework in which
  1. Number of classifier stages
  2. Number of features in each stage
  3. Threshold of each stage
- Minimum number of features are selected such that, expected DR and FPR are achieved

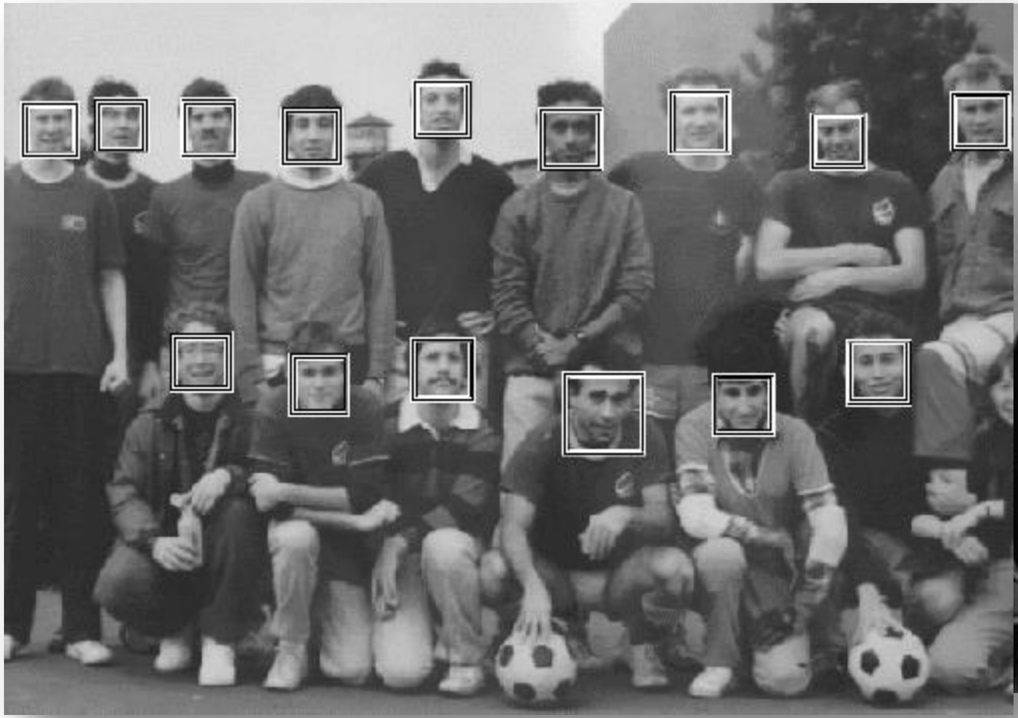
Combine Classifiers in Cascade

- Simple framework is used to produce effective cascade which is highly efficient
  1. User selects maximum acceptable FPR and minimum acceptable DR per each stage
  2. User selects target overall FPR and DR
  3. Each stage is trained by adding features until the target DR and FPRs are met
  4. Stages are added until the overall target for DR and FPR are met

Results

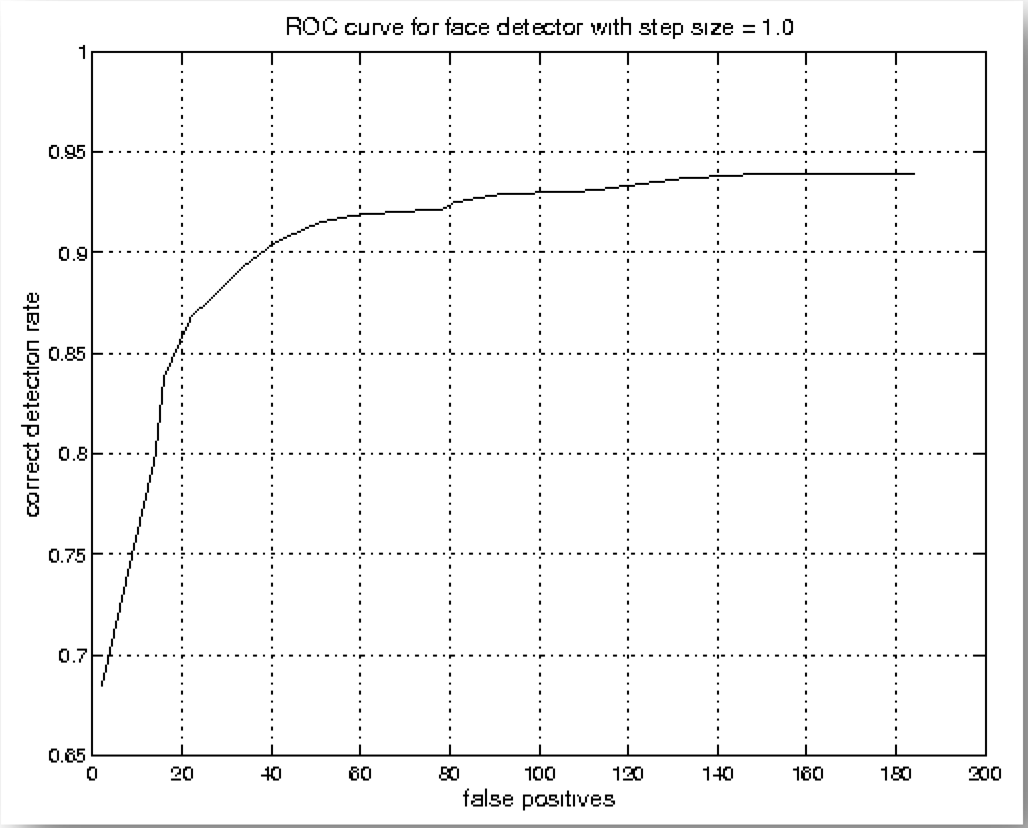
- Testing has done on MIT+CMU test set, which consists with 507 faces in 130 images
- Using a cascade of 38 layers
- Cascade has trained using 4916 facial images and 9544 non-facial images
- Testing has been done with scaling factor of 1.25 and windows shifting scale of 1.0 on images
- On a conventional Pentium III machine with 700Mhz processor.
- **They have achieved Detection Speed of 15 frames/sec on a 700MHz Intel Pentium 3**

Results



Results

- ROC Curve for Face Detector



Results

- Detection Rate Comparison of Cotemporary Solution

Detector \ False detections	10	31	50	65	78	95	167
Viola-Jones	76.1%	88.4%	91.4%	92.0%	92.1%	92.9%	93.9%
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%
Schneiderman-Kanade	-	-	-	94.4%	-	-	-
Roth-Yang-Ahuja	-	-	-	-	(94.8%)	-	-

Conclusion

- Solution achieves the goal of real time object detection
- Conjunction of simple rectangle features and integral image gives a efficient feature representation
- AdaBoost is used for the feature selection and classifier training
- Cascade of classifiers allows to quickly discard background regions and concentrate more on object-like regions