Human Detection

Marco Fiorucci

University of Venice, Italy

Image and Video Understanding

a.y. 2018/19

Human Detection

Detect and localize persons in images regardless of their:

- position;
- scale
- pose and orientation;
- illumination.

Why Is Human Detection Difficult?

Challenges:

- wide variety of articulated poses;
- variable appearance and poses;
- complex background;
- unconstrained illumination;
- occlusion;
- different scales.

From: N. Dalal

Research Issues

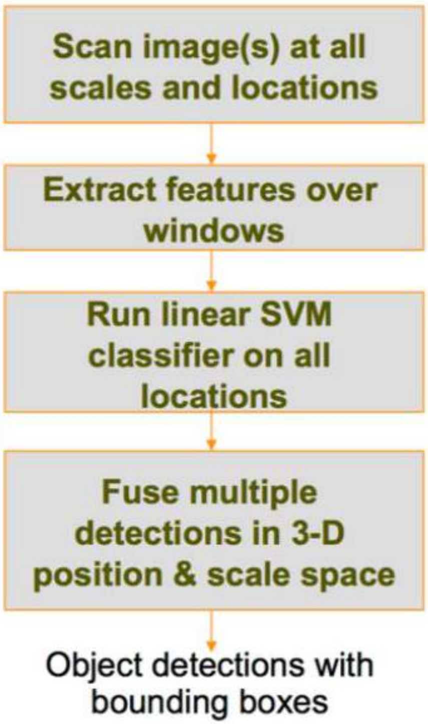
- **Representation:** how to describe a typical person?
- **Scale:** how to deal with persons of different size?
- **Search strategy:** how to spot these persons?
- **Post-processing:** How to combine detection results?

The detection phase

A person detector often works by asking the same question in turn of very possible rectangle (*window*) in the image that might possibly tightly bound one of the instance of interest (persons).

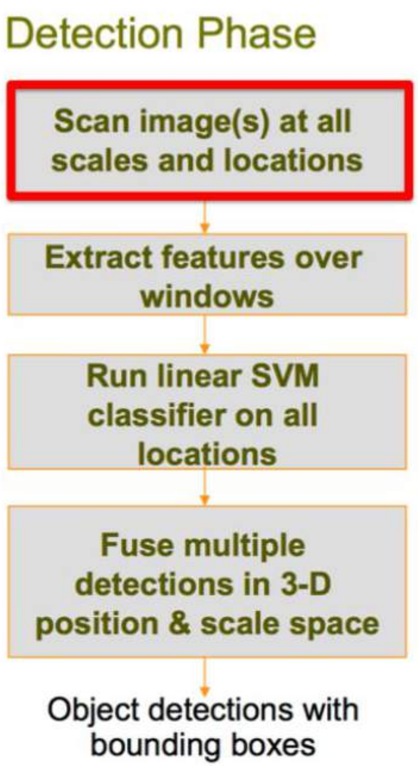
Sliding window detectors find objects in 4 steps:

1. Inspect every windows.
2. Given a window, extract a feature vector (i.e. a vector of numbers that describes the window's contents).
3. Classify each feature vector and accept a window if the score is above a certain threshold.
4. Clean-up the mess (post-processing).



Search Over Space

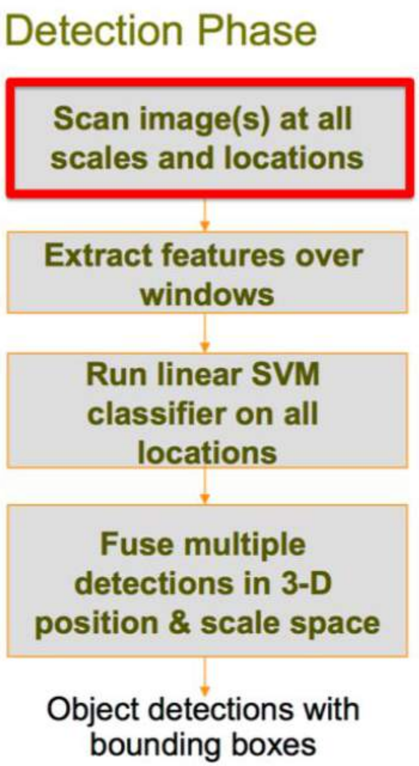
The window is 128 pixels tall and 64 pixels wide: 2 to 1 aspect ratio is a rough compromise between the aspect ratio of a person viewed from the front and one viewed from the side with legs fully extended during a step.

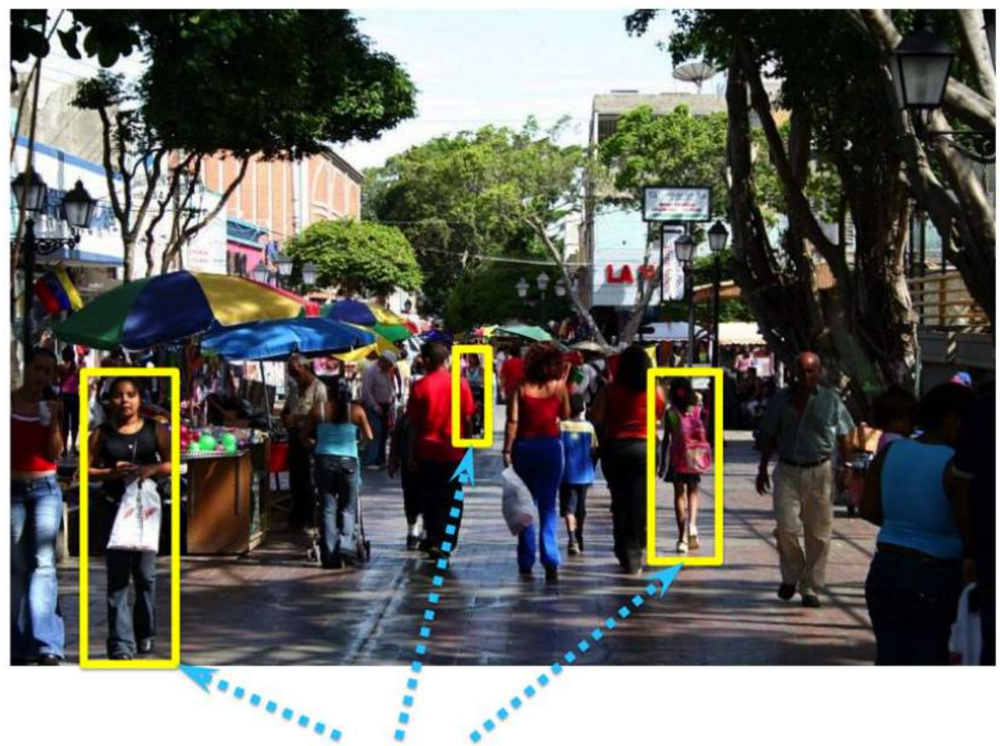


From: S. Fidler

Search Over Scale

Since window is fixed, how to deal with person at different size?



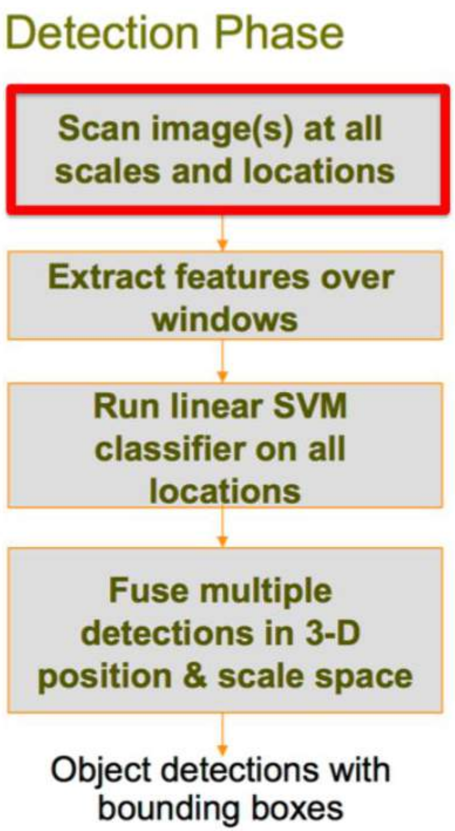


Objects can be of very different sizes (scales), even in the same image. How do we deal with that?

From: S. Fidler

Search Over Scale

Down-scale the image and slide again

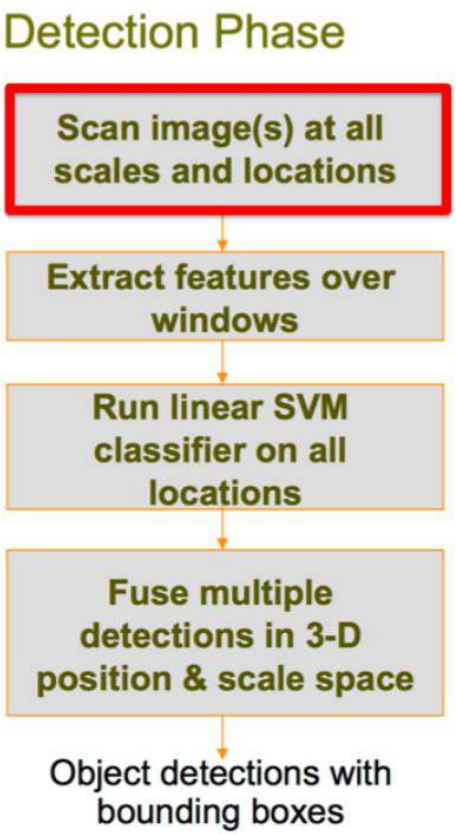


Scale-down the image, and slide the window again (the size of the window is always the same)

From: S. Fidler

Search Over Scale

Down-scale the image and slide again

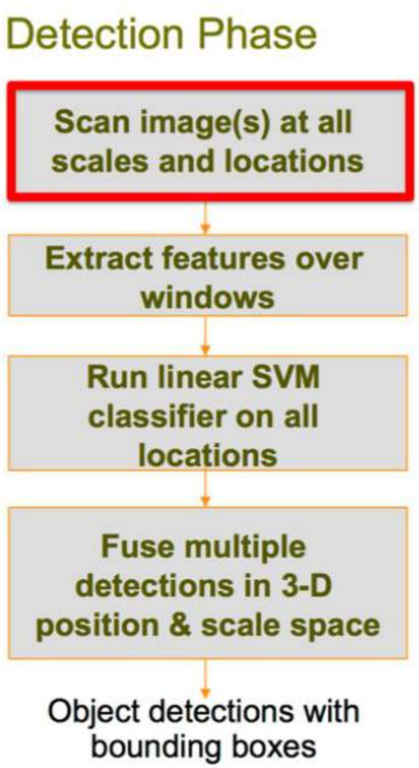


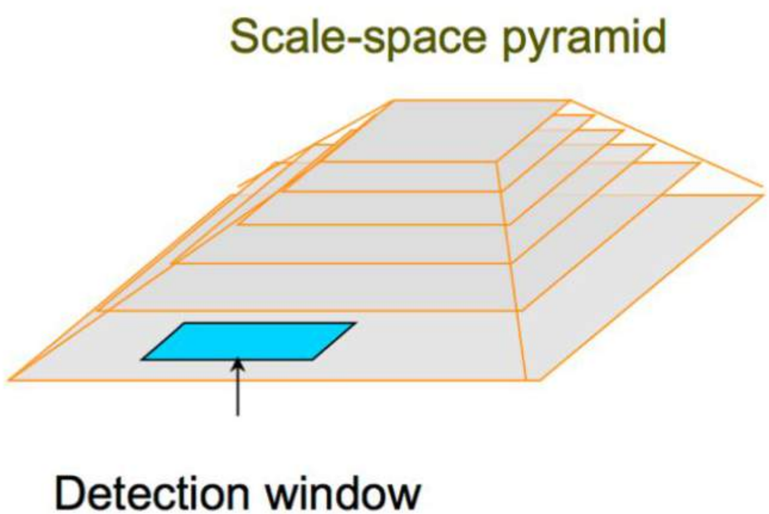
And again…

From: S. Fidler

Search Over Space and Scale

Do a full pyramid, a slide your detector at each scale. Make sure the scale differences across levels are small (do lots of re-scaled images).





From: S. Fidler

Histograms of Oriented Gradients

Histograms of Oriented Gradients for Human Detection

Navneet Dalal and Bill Triggs
INRIA Rhône-Alps, 655 avenue de l'Europe, Montbonnot 38334, France
{Navneet.Dalal,Bill.Triggs}@inrialpes.fr, http://lear.inrialpes.fr

Abstract

We study the question of feature sets for robust visual object recognition, adopting linear SVM based human detection as a test case. After reviewing existing edge and gradient based descriptors, we show experimentally that grids of Histograms of Oriented Gradient (HOG) descriptors significantly outperform existing feature sets for human detection. We study the influence of each stage of the computation on performance, concluding that fine-scale gradients, fine orientation binning, relatively coarse spatial binning, and high-quality local contrast normalization in overlapping descriptor blocks are all important for good results. The new approach gives near-perfect separation on the original MIT pedestrian database, so we introduce a more challenging dataset containing over 1800 annotated human images with a large range of pose variations and backgrounds.

We briefly discuss previous work on human detection in §2, give an overview of our method §3, describe our data sets in §4 and give a detailed description and experimental evaluation of each stage of the process in §5–6. The main conclusions are summarized in §7.

2 Previous Work

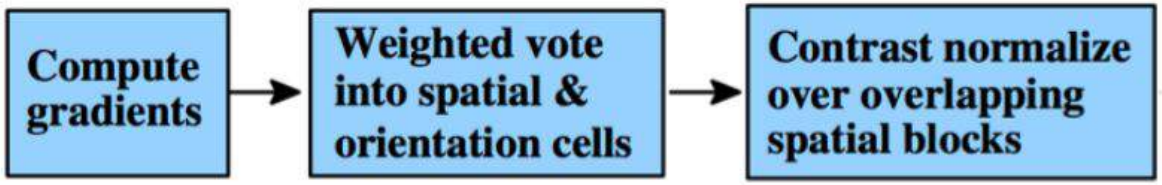
There is an extensive literature on object detection, but here we mention just a few relevant papers on human detection [18, 17, 22, 16, 20]. See [6] for a survey. Papageorgiou *et al* [18] describe a pedestrian detector based on a polynomial SVM using rectified Haar wavelets as input descriptors, with a parts (subwindow) based variant in [17]. Depoortere *et al* give an optimized version of this [2]. Gavrila & Philomen [8] take a more direct approach, extracting edge images and matching them to a set of learned exemplars using chamfer distance. This has been used in a practical real-time pedes-

24415 citations!!!

Histograms of Oriented Gradients

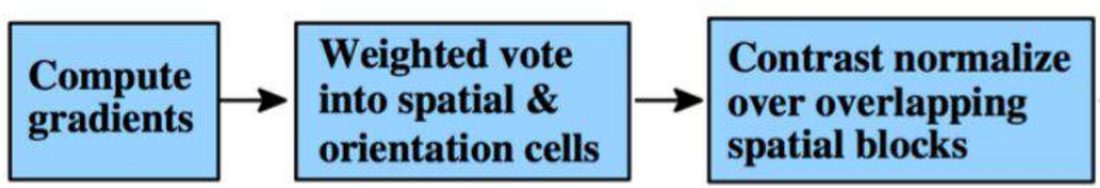
The feature is specifically tuned to person detection.

[L]ocal object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions. In practice this is implemented by dividing the image window into small spatial regions ("cells"), for each cell accumulating a local 1-D histogram of gradient directions or edge orientations over the pixels of the cell. The combined histogram entries form the representation. For better invariance to illumination, shadowing, etc., it is also useful to contrast-normalize the local responses before using them. This can be done by accumulating a measure of local histogram "energy" over somewhat larger spatial regions ("blocks") and using the results to normalize all of the cells in the block. We will refer to the normalized descriptor blocks as *Histogram of Oriented Gradient (HOG)* descriptors [1].



HOG Steps

- HOG feature extraction
  - Compute centered horizontal and vertical gradients with no smoothing
  - Compute gradient orientation and magnitudes
    - For color image, pick the color channel with the highest gradient magnitude for each pixel.
  - For a 64x128 image,
  - Divide the image into 16x16 blocks of 50% overlap.
    - 7x15=105 blocks in total
  - Each block should consist of 2x2 cells with size 8x8.
  - Quantize the gradient orientation into 9 bins
    - The vote is the gradient magnitude
    - Interpolate votes between neighboring bin center.
    - The vote can also be weighted with Gaussian to downweight the pixels near the edges of the block.
  - Concatenate histograms (Feature dimension: 105x4x9 = 3,780)



From: M. Shah

Computing Gradient

Discrete Derivative

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x)$$

$$\frac{df}{dx} = \frac{f(x) - f(x-1)}{1} = f'(x)$$

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

Example

$$f(x) = 10 \quad 15 \quad 10 \quad 10 \quad 25 \quad 20 \quad 20 \quad 20$$
$$f'(x) = 0 \quad 5 \quad -5 \quad 0 \quad 15 \quad -5 \quad 0 \quad 0$$

From: M. Shah

Computing Gradient

Given function	$f(x, y)$
Gradient vector	$\nabla f(x, y) = \begin{bmatrix} \dfrac{\partial f(x, y)}{\partial x} \\ \dfrac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$
Gradient magnitude	$\lvert \nabla f(x, y) \rvert = \sqrt{f_x^2 + f_y^2}$
Gradient direction	$\theta = \tan^{-1} \dfrac{f_x}{f_y}$

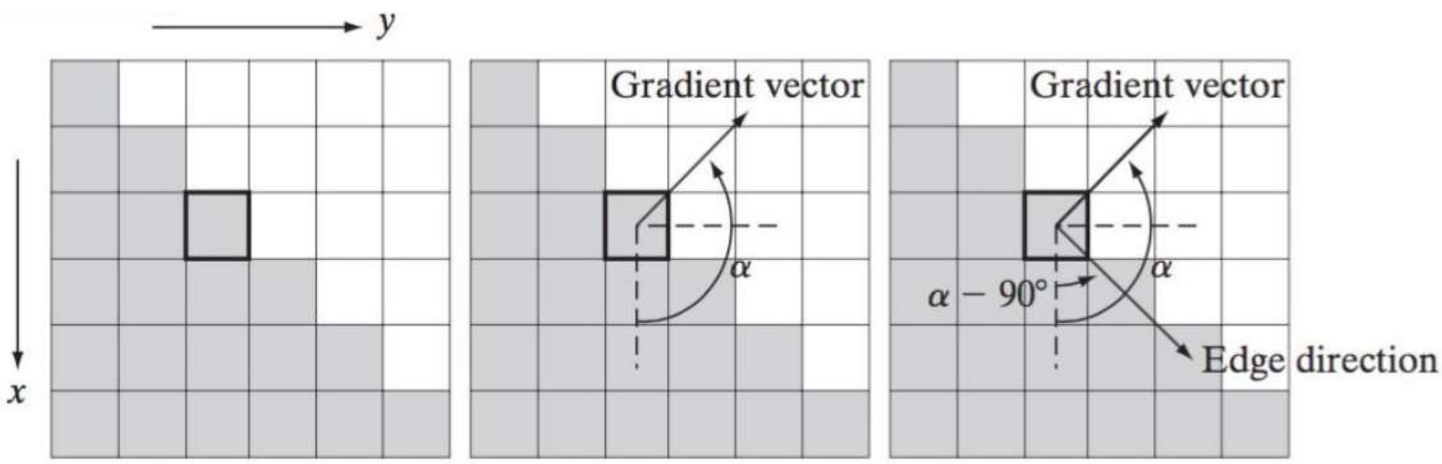
Computing Gradient

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$

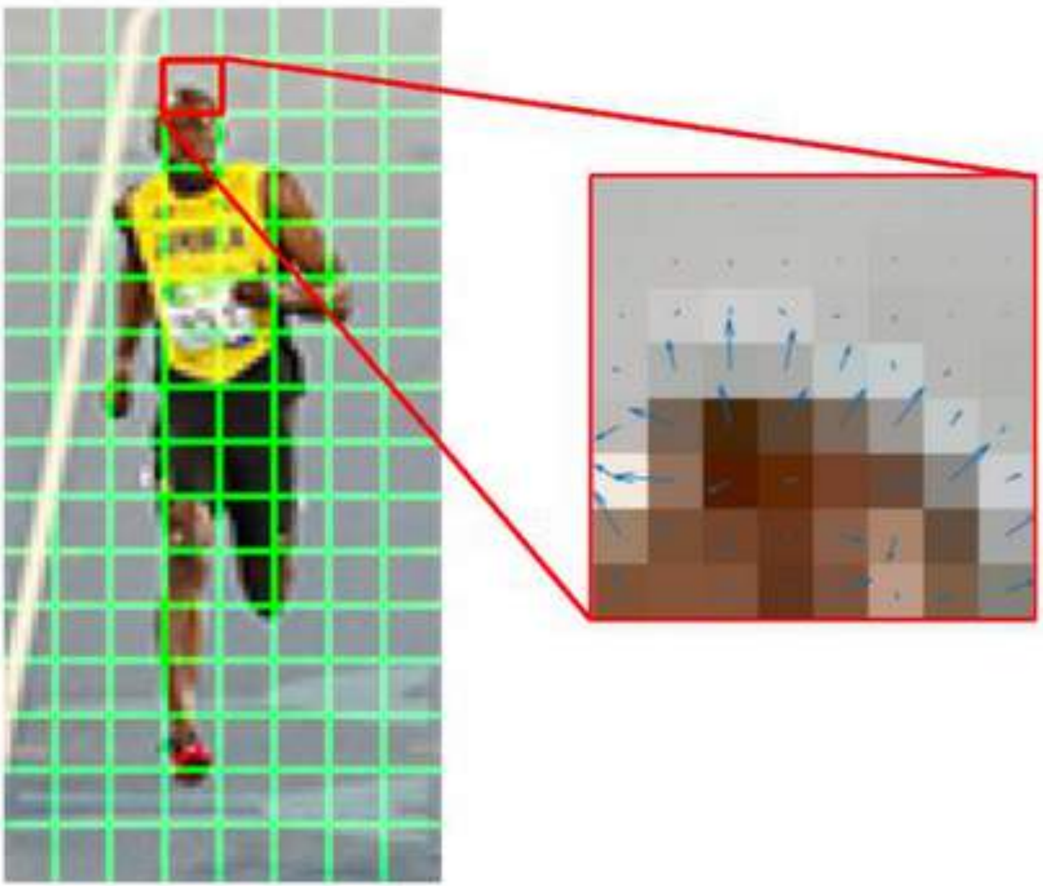
Centered:

-1	0	1

-1
0
1



Computing Gradients



2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

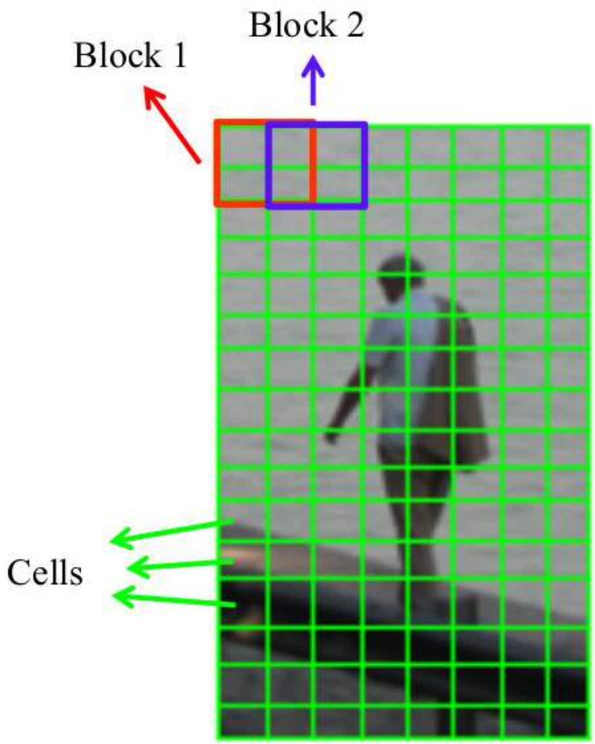
Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

Cells, Blocks

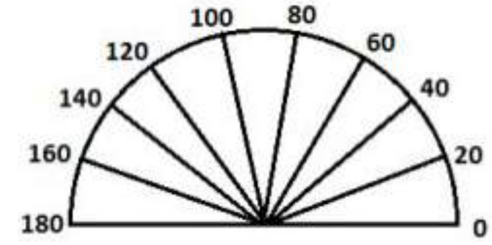
- For a 64x128 image,
- Divide the image into 16x16 blocks of 50% overlap.
  - 7x15=105 blocks in total
- Each block should consist of 2x2 cells with size 8x8.

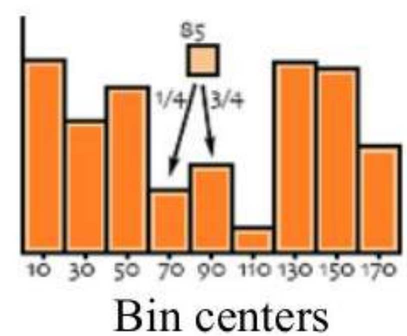


From: M. Shah

Votes

- In each cell, compute histogram of the gradient orientation binned into B bins (B = 9).
- The vote is the gradient magnitude.
- Interpolate votes linearly between neighboring bin centers.
    - Example: if θ=85 degrees.
    - Distance to the bin center Bin 70 and Bin 90 are 15 and 5 degrees, respectively.
    - Hence, ratios are 5/20=1/4, 15/20=3/4.

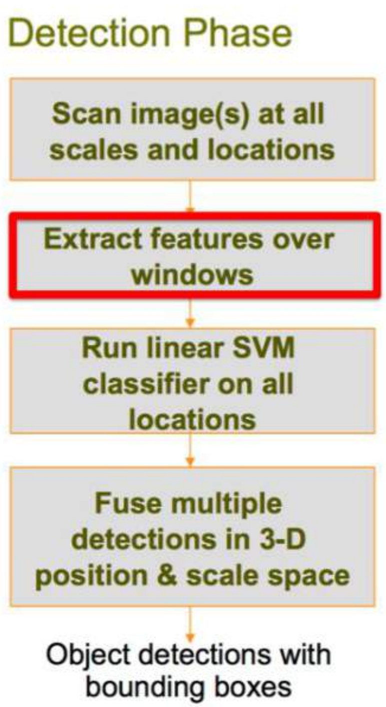


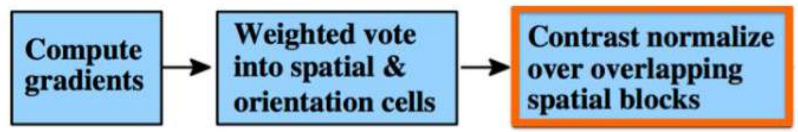


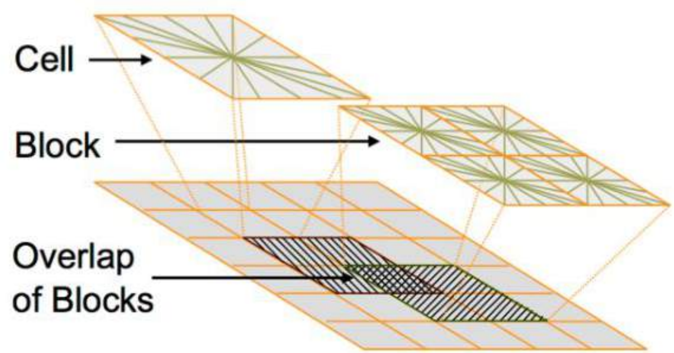
From: C. Tomasi

Block Normalization

Concatenate the four cell histograms in each block into *a single block feature* **f** and normalize the block feature by its Euclidean norm.







Feature vector *f* = [..., ..., ...]

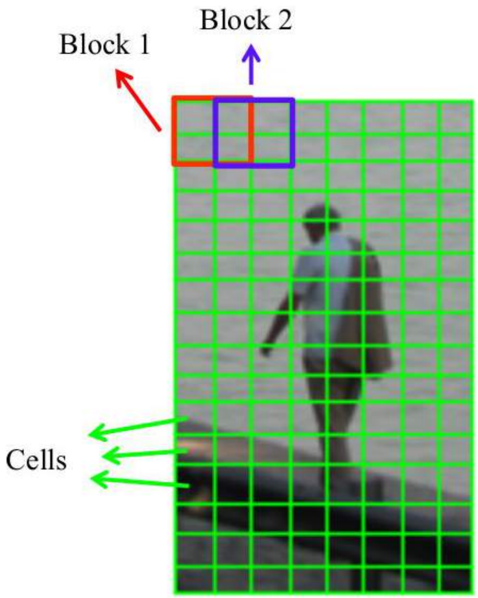
L2 normalization in each block: $\mathbf{f} = \dfrac{\mathbf{f}}{\sqrt{||\mathbf{f}||_2^2 + \epsilon^2}}$

From: S. Fidler

Final Feature Vector

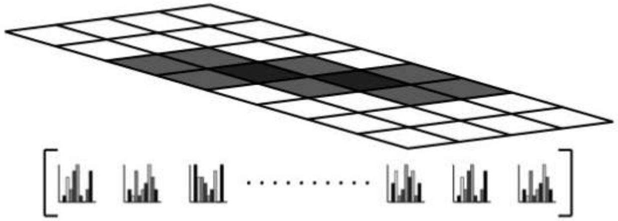
With a 128 × 64 window and cells with 8 × 8 pixels there are 16 cells vertically and 8 horizontally.
With an 8-pixel block stride there are then 15 blocks vertically and 7 horizontally, and with 4 cells per block and 9 orientation bins per histogram.
The length of the HOG feature vector is:

$$15 \times 7 \times 4 \times 9 = 3780$$



Concatenate histograms

- Make it a 1D vector of length 3780.

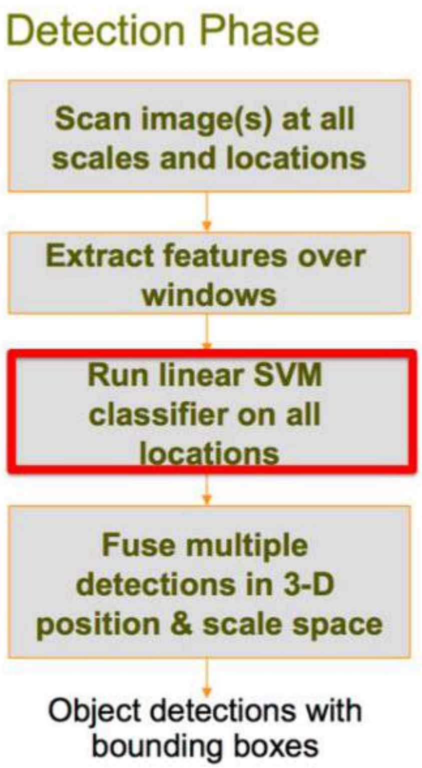


From: m. Shah

HOG Features Visualization

The HOG Detector: Classification

Feature done, we are ready for classification.

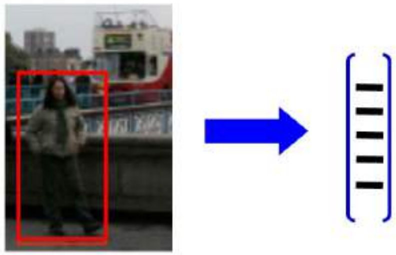


- Train a windows classifier
- Use the trained classifier to predict presence/absence of a person (object class) in each window in the image.

Classification

Learning phase

- Represent each example window by a HOG (Histogram of Oriented Gradients) feature vector:



$$\mathbf{x}_i \in \mathbb{R}^d,$$ with d = 3780

- Train a linear SVM classifier

Testing (Detection)

- Sliding window SVM

From: A. Zisserman

Evaluation Data Sets

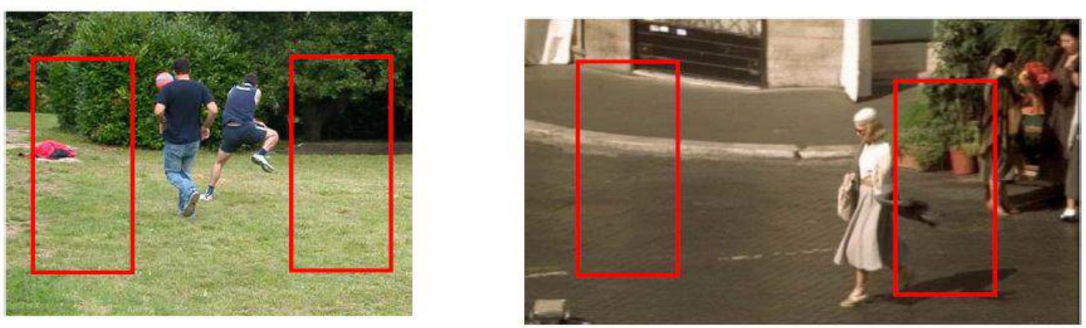
	MIT pedestrian database	INRIA person database
		
Train	507 positive windows Negative data unavailable	1208 positive windows 1218 negative images
Test	200 positive windows Negative data unavailable	566 positive windows 453 negative images
	Overall 709 annotations+ reflections	Overall 1774 annotations+ reflections

From: A. Zisserman

Training data

- Positive data – 1208 positive window examples

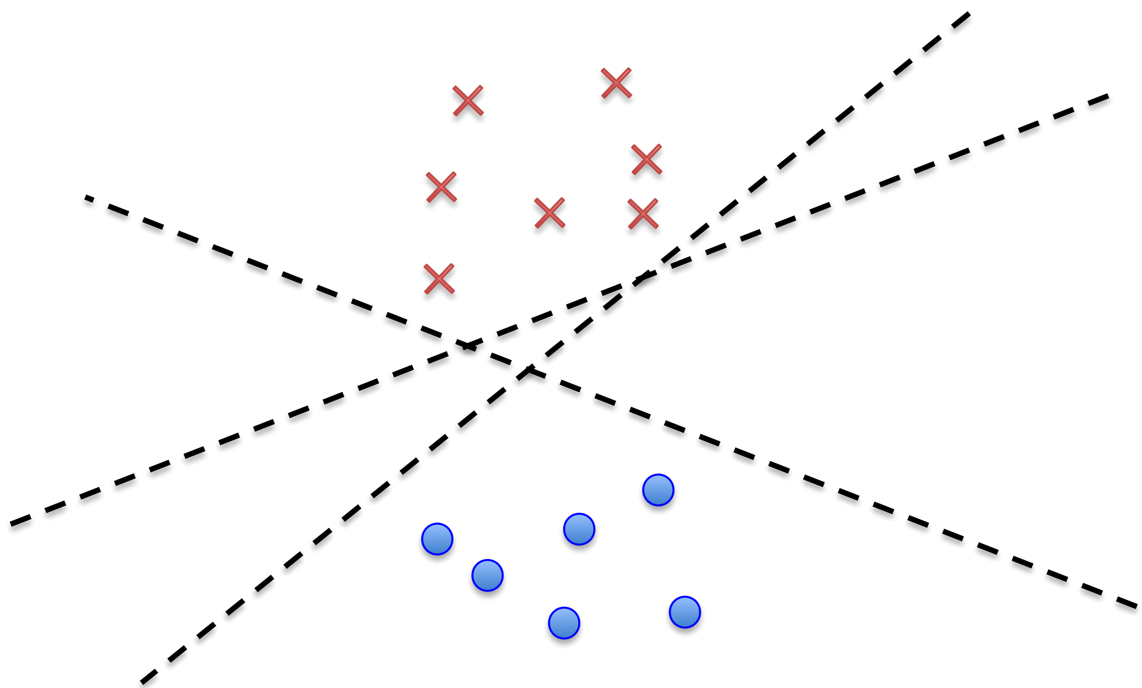
- Negative data – 1218 negative window examples (initially)



From: A. Zisserman

Support Vector Machines

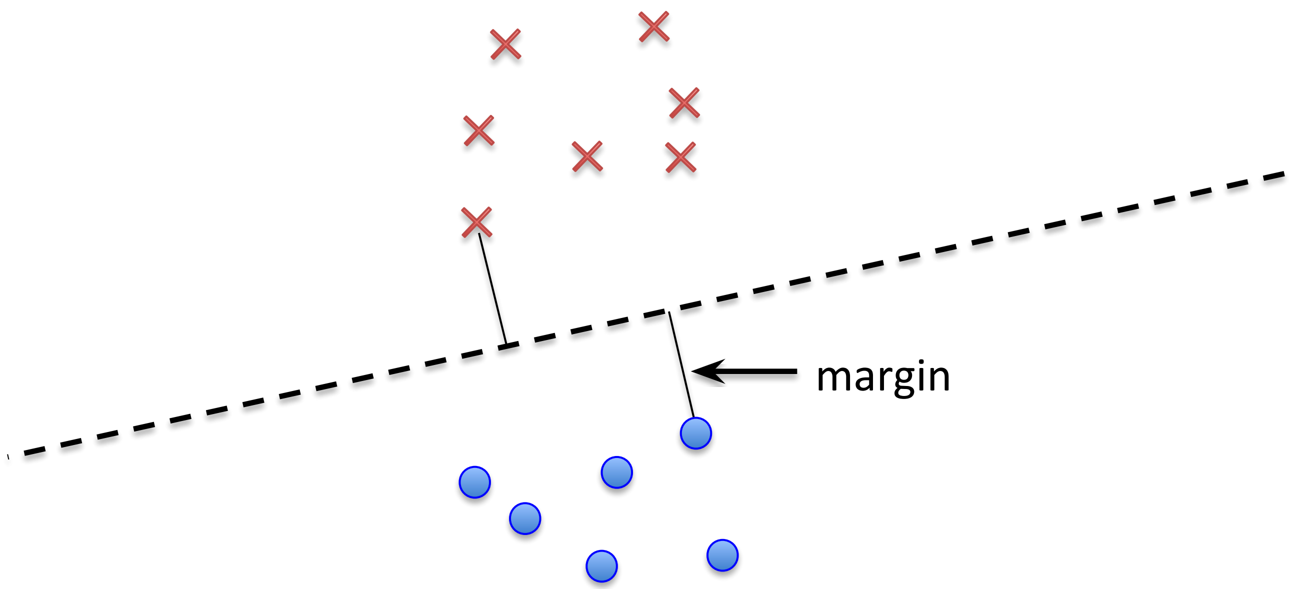
Several possible decision boundaries



All get 100% accuracy on this training set!

Several possible decision boundaries

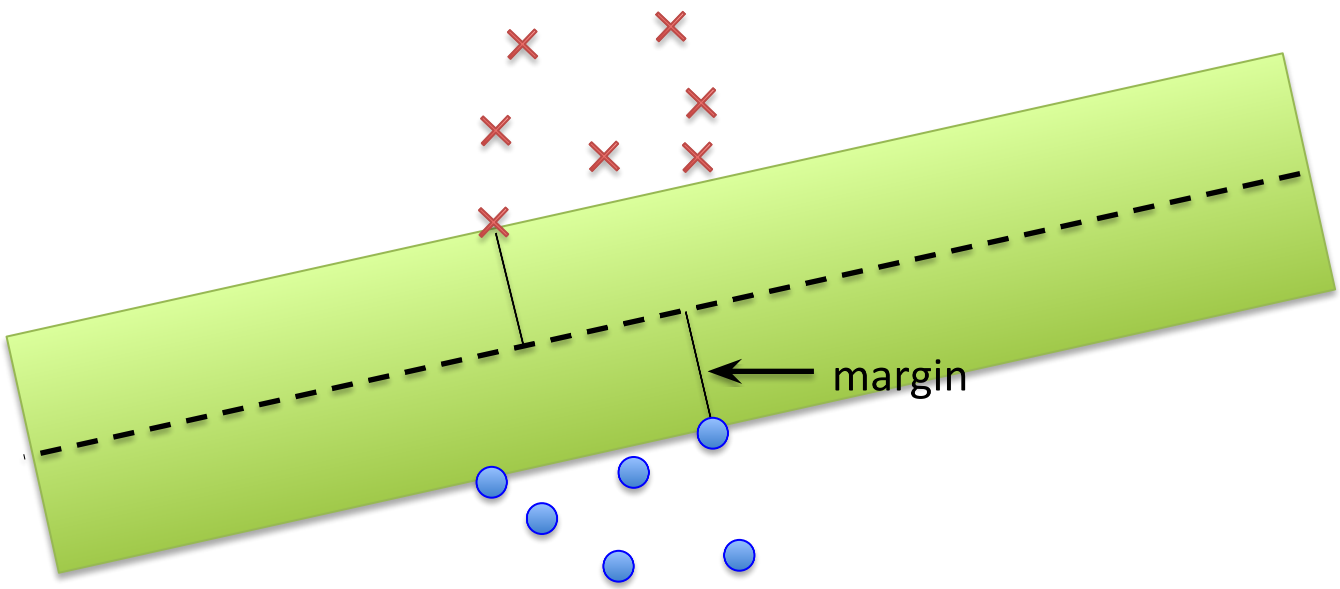
The SVM finds this one – the boundary furthest from the two clusters



Distance to the closest training point is called **the margin**
(equal on both sides of the boundary)

Several possible decision boundaries

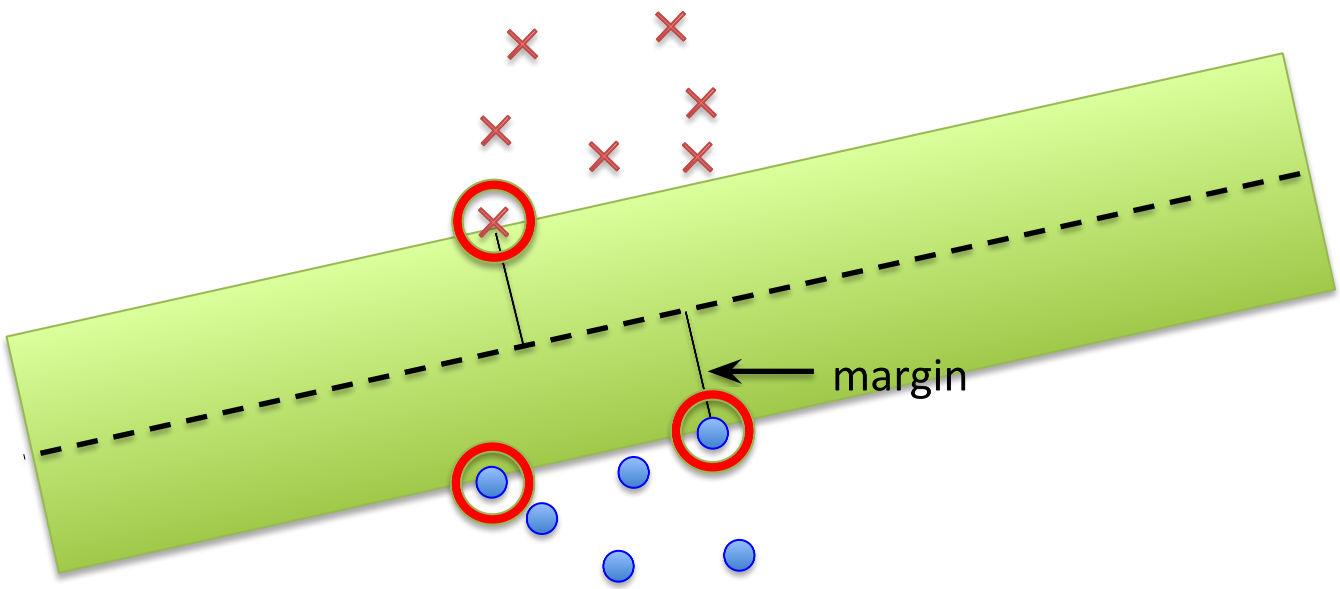
The SVM finds this one – the boundary furthest from the two clusters



Distance to the closest training point is called **the margin**
(equal on both sides of the boundary)

Several possible decision boundaries

The SVM finds this one – the boundary furthest from the two clusters



The circled points are called
SUPPORT VECTORS

All other points can move freely.
Solution only dependent on SVs.

Normalizing the weights

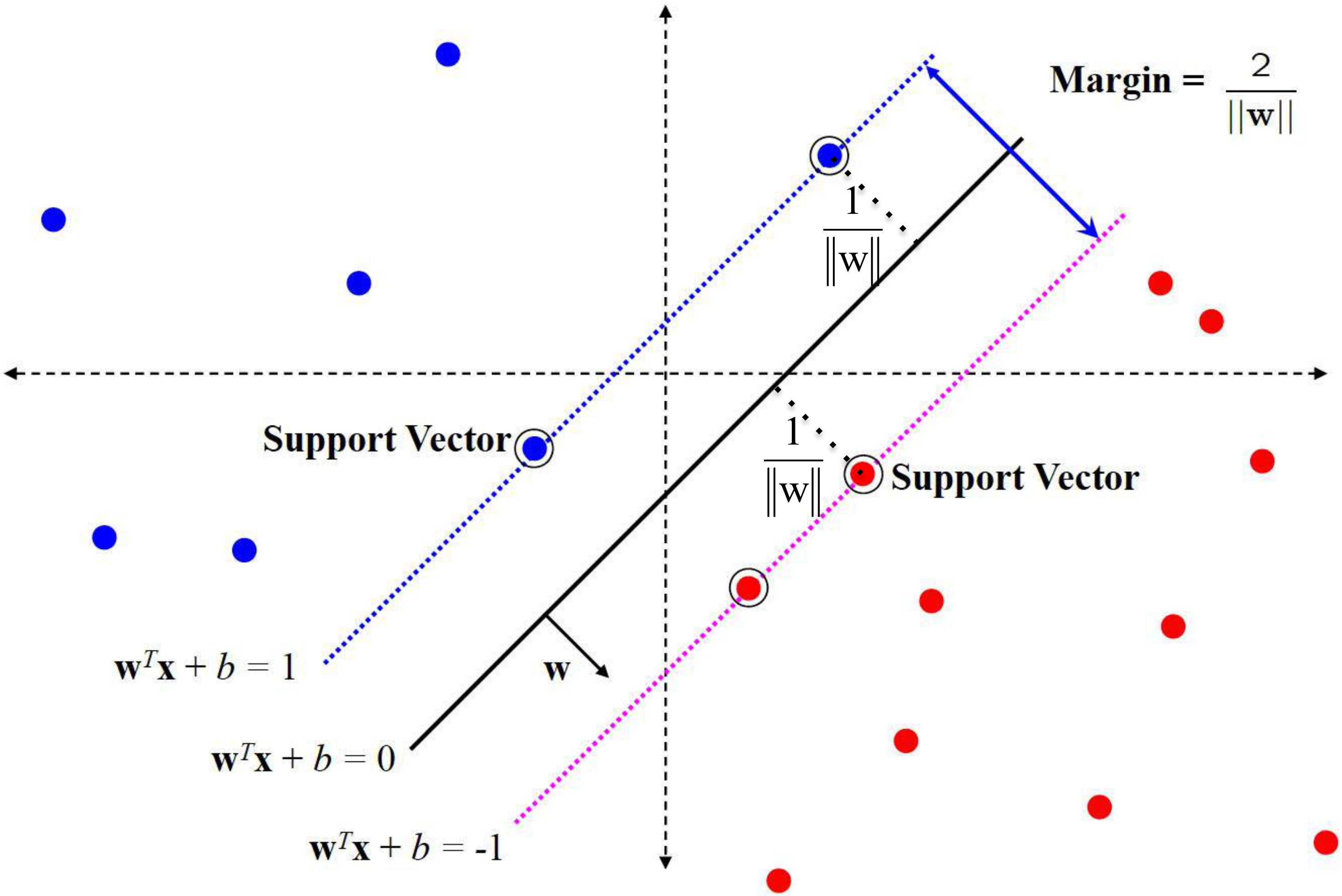
Note that $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0$ and $c(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) = 0$ define the same plane.

Hence we have the freedom to choose the normalization of $\mathbf{w}$ and b.

Choose normalization such that (canonical form):

- $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = +1$ for the *positive* support vectors
- $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = -1$ for the *negative* support vectors

Support Vector Machines



Learning SVMs

Learning the SVM can be formulated as an optimization problem:

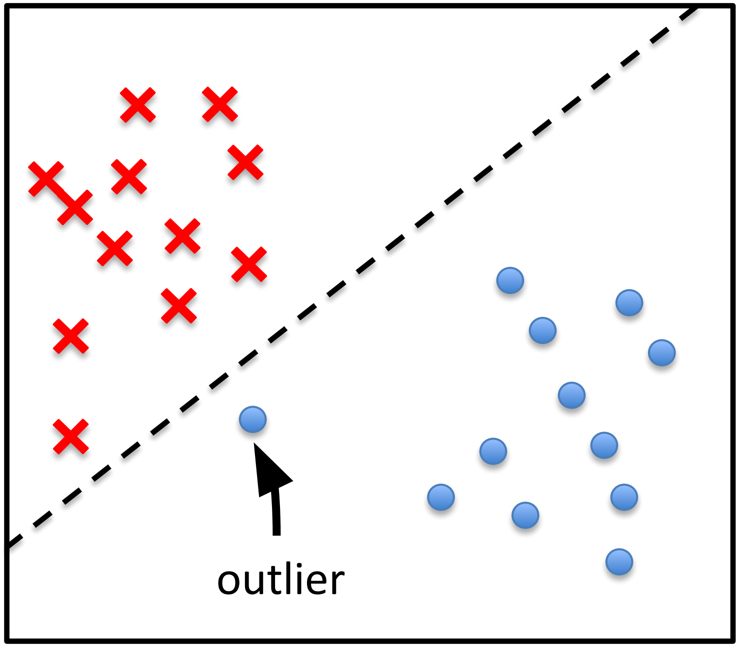
$$\max_{\mathbf{w}} \frac{2}{||\mathbf{w}||} \text{ subject to } \mathbf{w}^\top \mathbf{x}_i + b \begin{array}{ll} \geq 1 & \text{if } y_i = +1 \\ \leq -1 & \text{if } y_i = -1 \end{array} \text{ for } i = 1 \ldots N$$

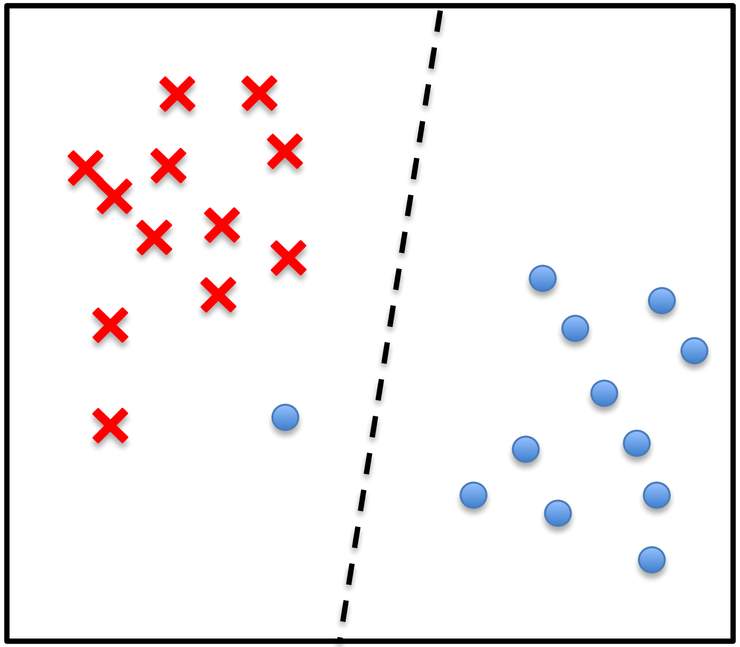
or, equivalently:

$$\min_{\mathbf{w}} ||\mathbf{w}||^2 \text{ subject to } y_i \left(\mathbf{w}^\top \mathbf{x}_i + b \right) \geq 1 \text{ for } i = 1 \ldots N$$

This is a (convex) quadratic optimization problem subject to linear constraints and **there is a unique minimum!**

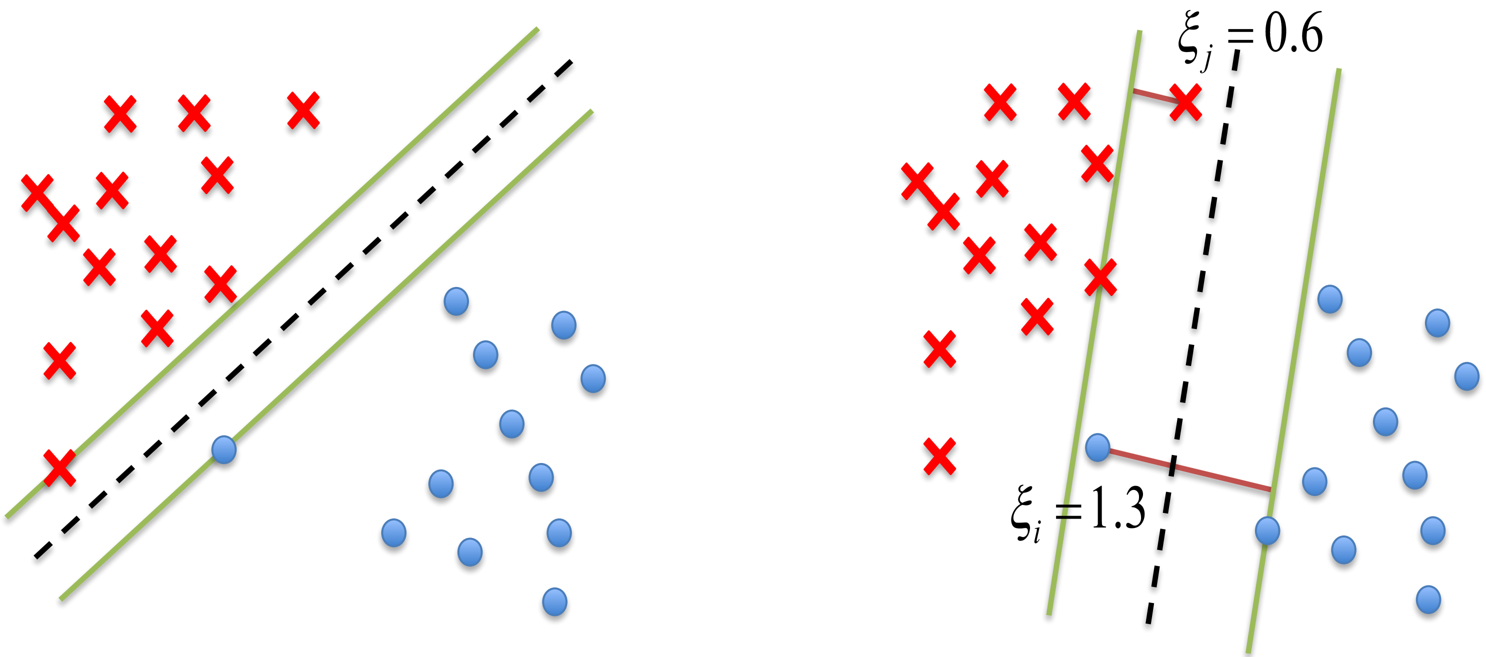
How to manage outliers: Slack variables
(aka soft margins)





How to manage outliers: Slack variables

(aka soft margins)



How to manage outliers: Slack variables
(aka soft margins)

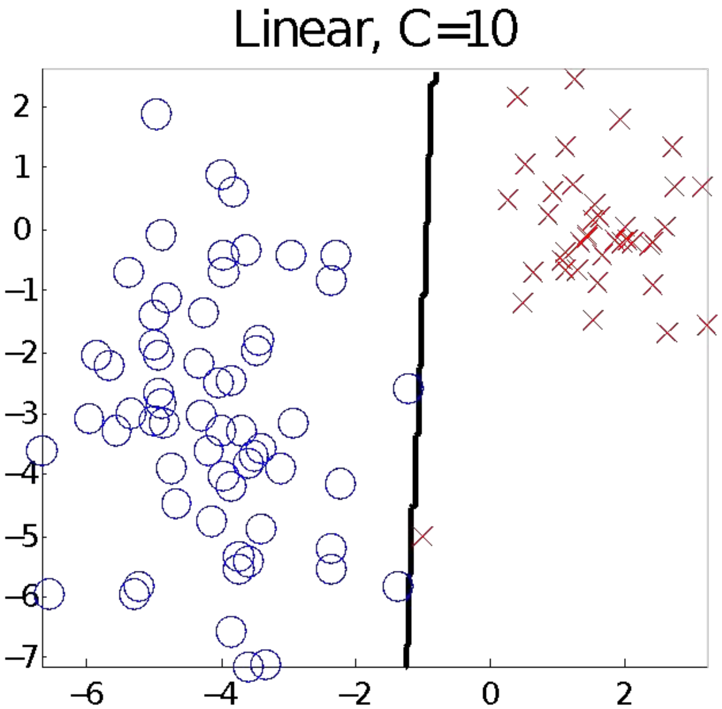
$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ &\text{subject to} && y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) \geq 1 - \xi_i \\ & && \xi_i \geq 0 \qquad\qquad i = 1,\ldots,N \end{aligned}$$

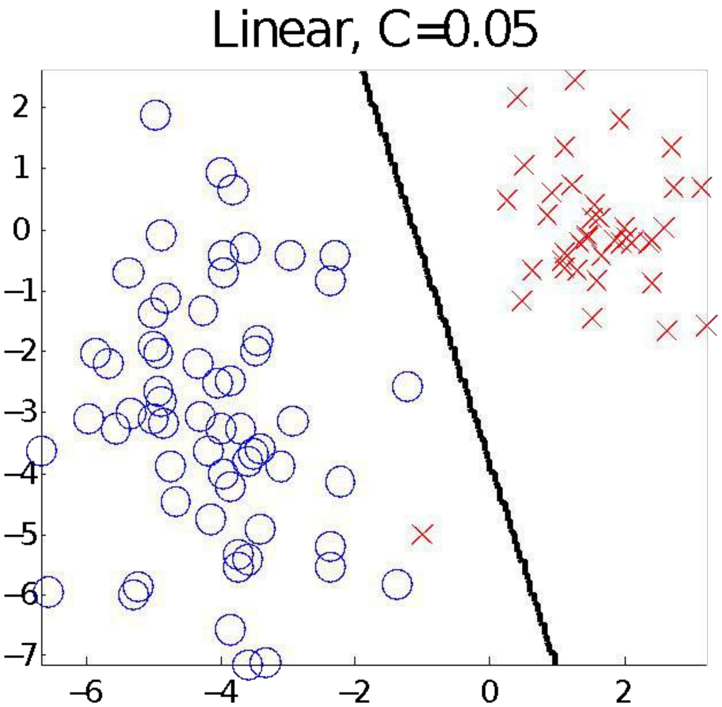
The only parameter C controls the tradeoff between the accuracy w.r.t. to the training data and the maximization of the margin.

It can be interpreted also as a regularization term:

- small C allows constraints to be easily ignored $\rightarrow$ large margin
- large C makes constraints hard to ignore $\rightarrow$ narrow margin
- $C = \infty$ enforces all constraints: hard margin

Example

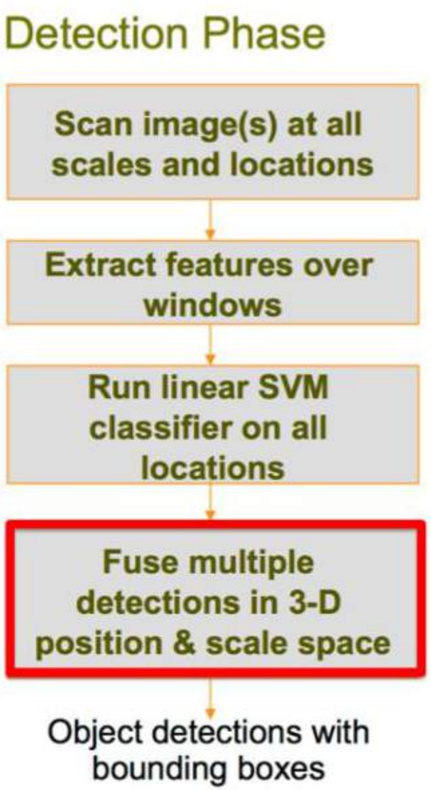




The HOG Detector – Post-processing

Post-processing

Perform *Non-Maxima Suppression* (NMS).



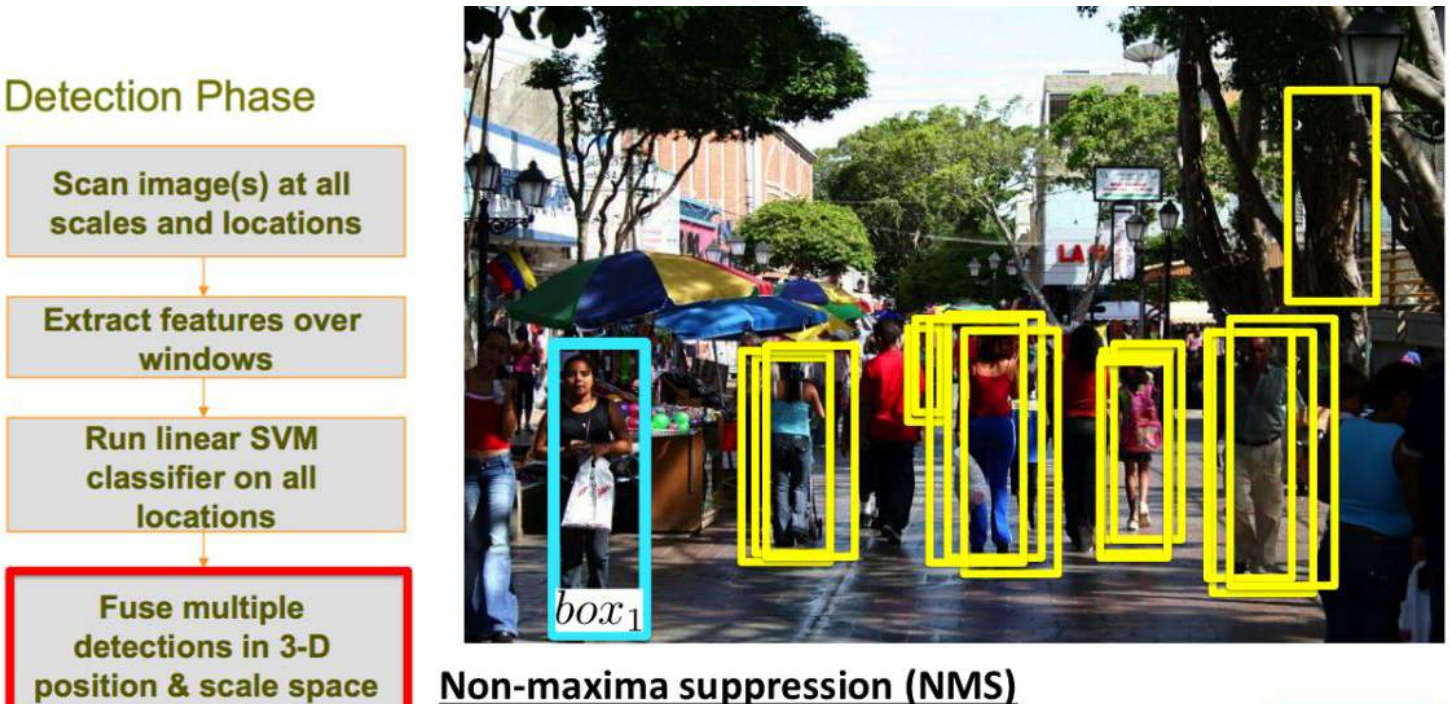
Non-maxima suppression (NMS)

- Greedy algorithm.
- At each iteration pick the highest scoring box.

From: S. Fidler

Post-processing

Perform *Non-Maxima Suppression* (NMS).



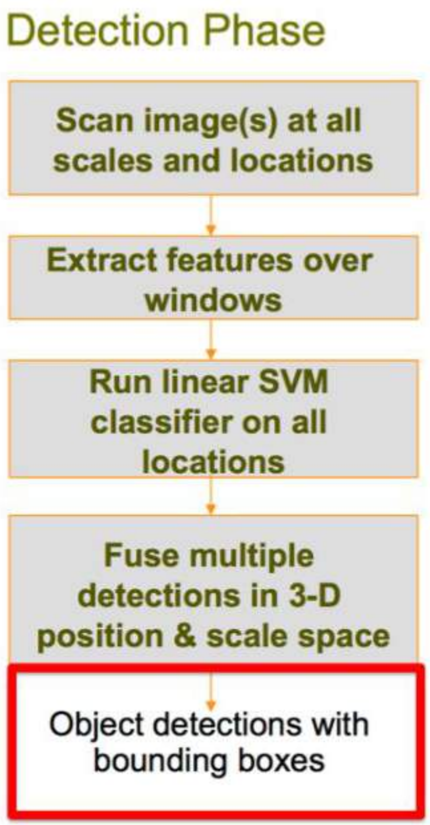
overlap = area(box$_1$ ∩ box$_2$)/ area(box$_1$ ∪ box$_2$)

- Remove all boxes that overlap more than 50% with the chosen box.

From: S. Fidler

Post-processing

Done!



Voila!

(Any idea how you would get rid of that tree detection or the upper right?)

From: S. Fidler

Are We Done?

Single, rigid template usually not enough to represent a category

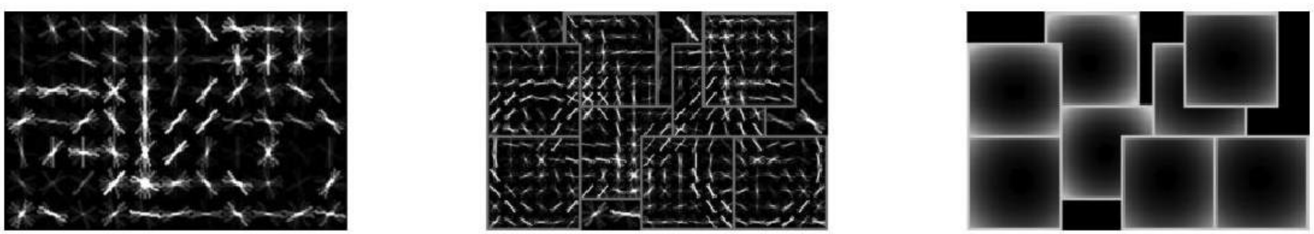
– Many objects (e.g. humans) are articulated, or have parts that can vary in configuration

– Many object categories look very different from different viewpoints, or from instance to instance

From: N. Snavely

Part-Based Model

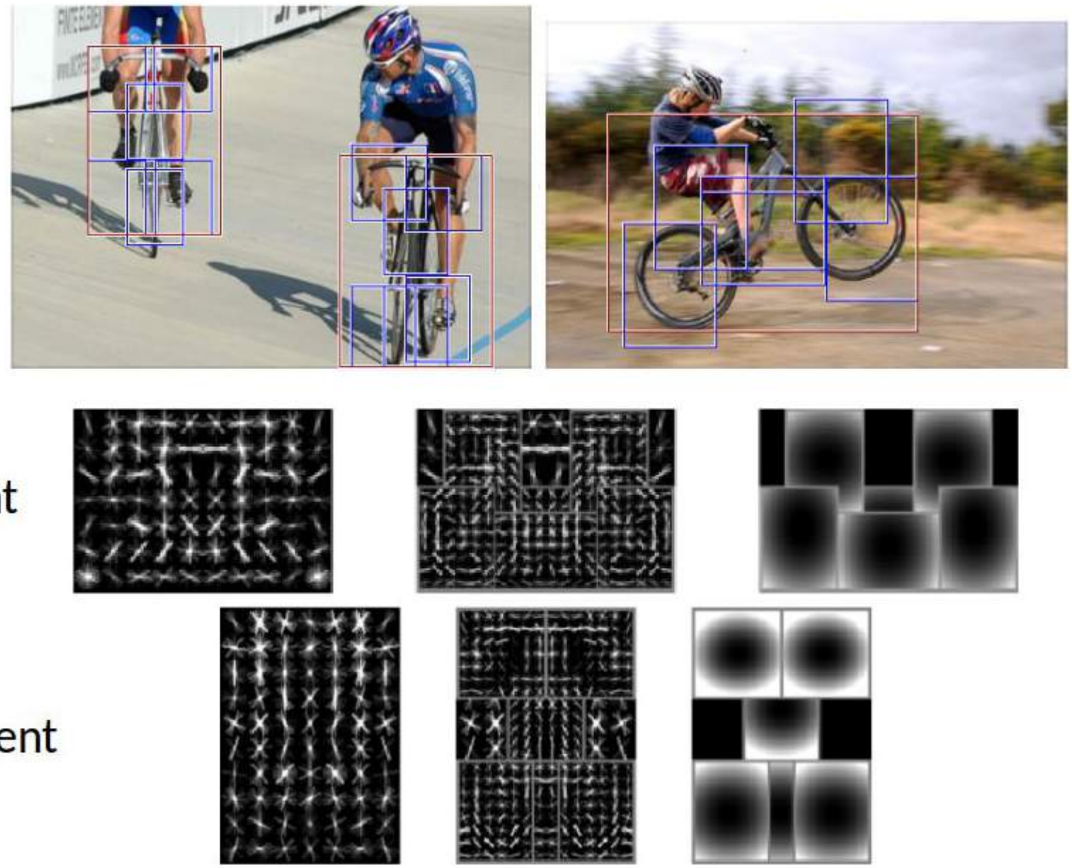
Our first innovation involves enriching the Dalal-Triggs model using a star-structured part-based model defined by a "root" filter (analogous to the Dalal-Triggs filter) plus a set of parts filters and associated deformation models.



Felzenszwalb, et al., **Discriminatively Trained Deformable Part Models**, http://people.cs.uchicago.edu/~pff/latent/

From: N. Snavely

Two-component Bicycle Model

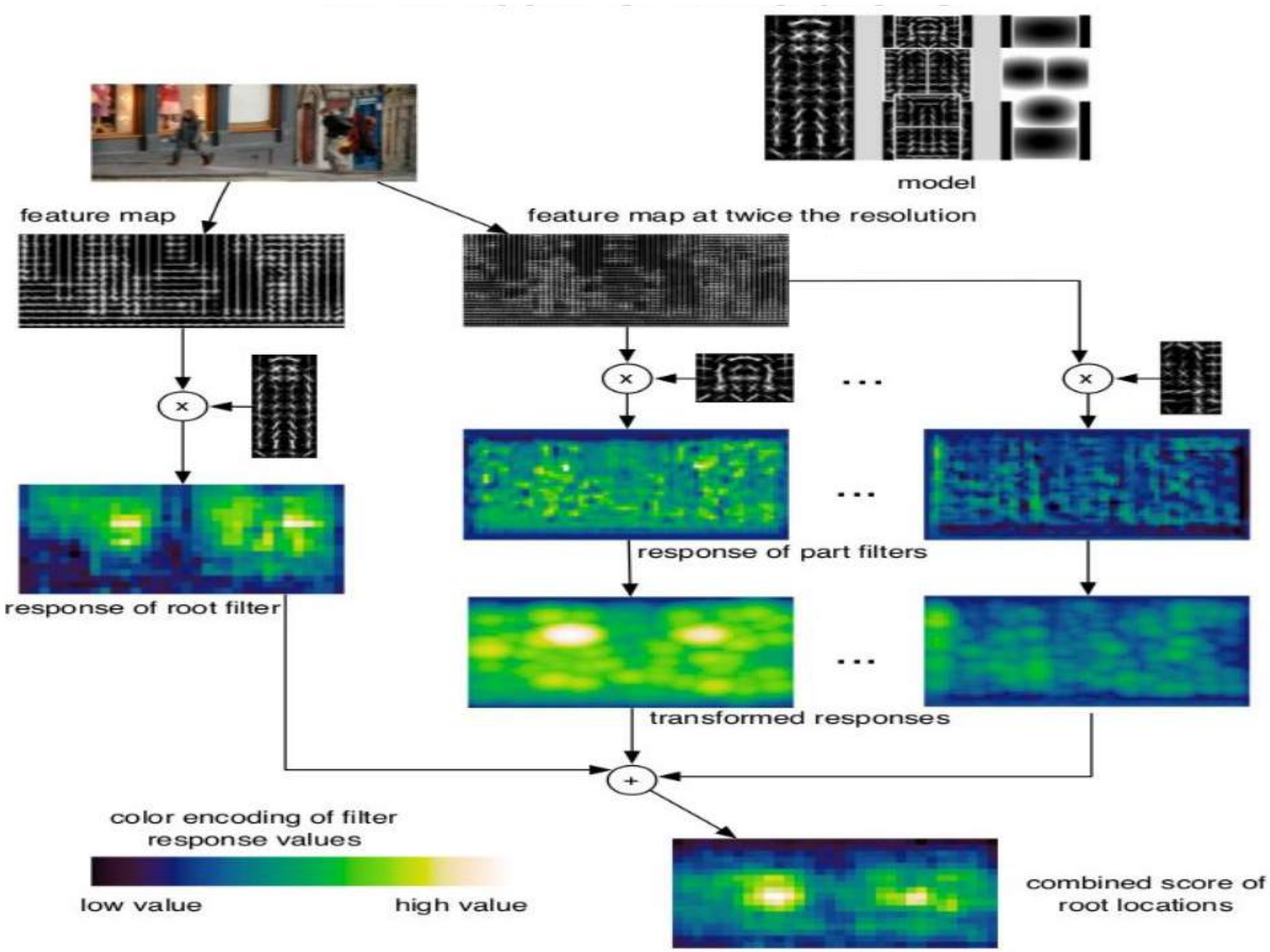


"side" component

"frontal" component

From: N. Snavely

Mixture Model



From: N. Snavely

Latent SVMs

- Rather than training a single linear SVM separating positive examples…
- … cluster positive examples into “component s” and train a classifier for each (using all negative examples)

From: N. Snavely

References

- N. Dalal and B. Triggs, Histograms of oriented gradients for human detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pp. 886-893 vol. 1 (2005).
- P. F. Felzenszwalb, R. B. Girshick, D. McAllester and D. Ramanan, Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 32, no. 9, pp. 1627-1645, (2010).
- C. Burges, A tutorial on support vector machines for pattern recogniton. *Data Mining and Knowledge Discovery* 2(2):121-167 (1998).
- N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other Kernel Based Learning Methods*. Cambridge University Press (2000).

OpenCV Tutorials

- Sliding Windows for Object Detection with Python and OpenCV (Link).
- Histogram of Oriented Gradients with Python and OpenCV (Link).
- Pedestrian Detection with Python and OpenCV (Link).