Outline for today

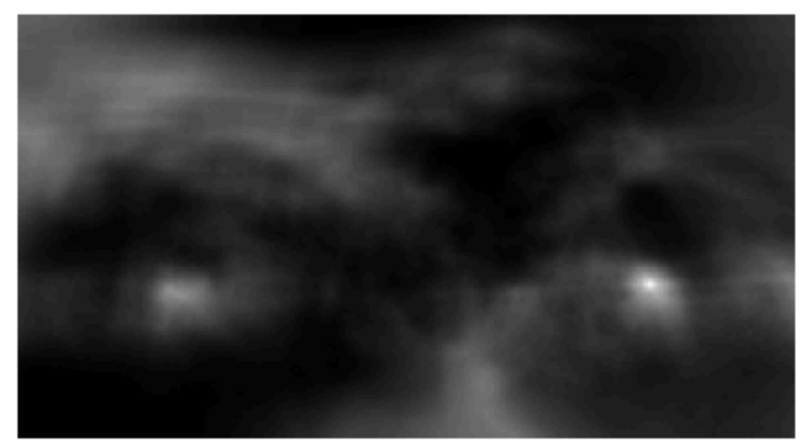
- Histogram of Oriented Gradient (HOG) features
- Pictorial structures (PS) / deformable parts models (DPM)
- Mixtures of deformable models
- Parameter learning with latent SVM
- Possibly more...
  - Cascade detection with DPMs
  - Context rescoring

(Review?) Template matching

- Consider matching with image patches
  - What could go wrong?

template

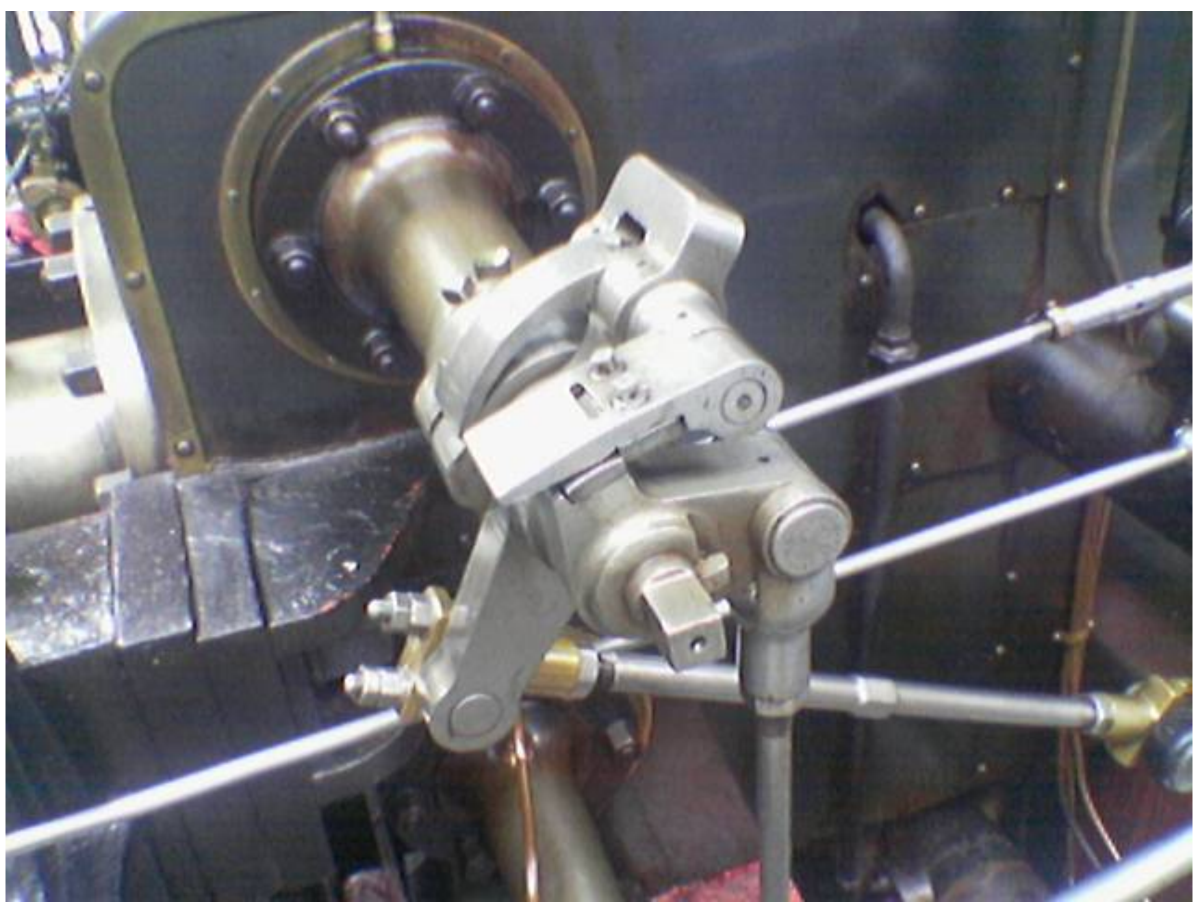
image



match quality
e.g., cross correlation

What is a feature map?

- Any transformation of an image into a new representation
- Example: transform an image into a binary edge map



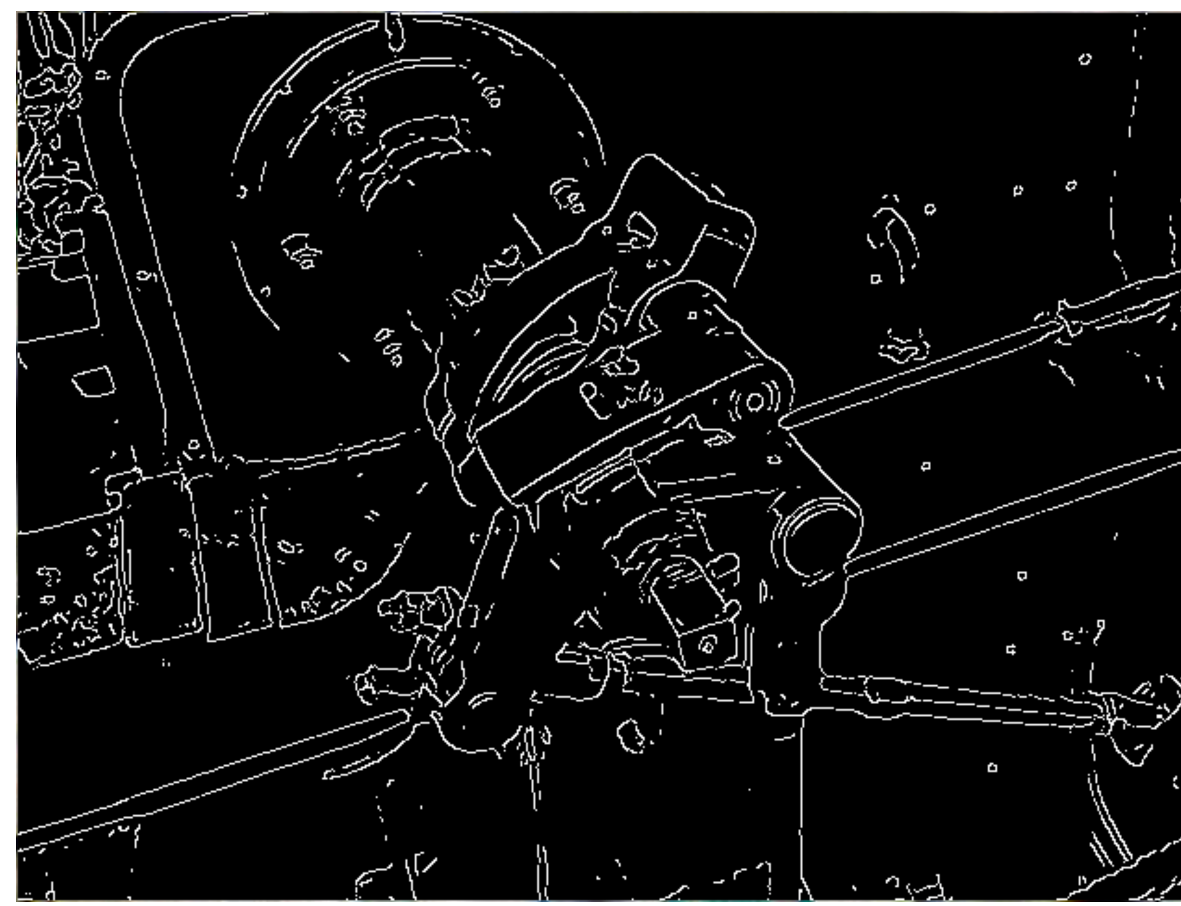


Image source: wikipedia

Feature map goals

- Introduce invariance
  - Bias, gain, nonlinear transformations
  - Small deformations

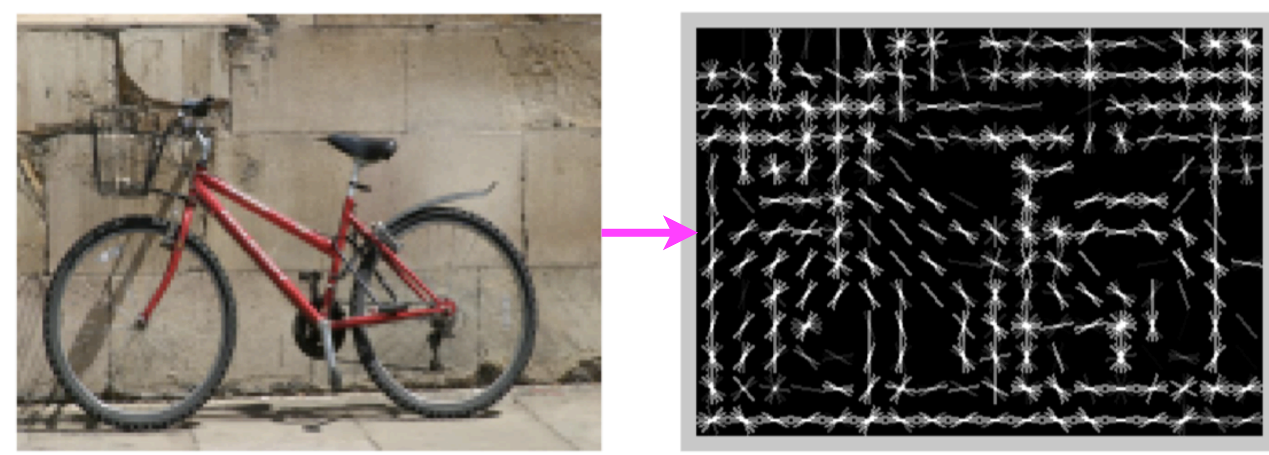
Figure 1.3: Variation in appearance due to a change in illumination

- Preserve larger scale spatial structure

Image: [Fergus05]

Histograms of Oriented Gradients (HOG)

- Introduce invariance
  - Bias / gain / nonlinear transformations
    - bias: gradients / gain: local normalization
    - nonlinearity: clamping magnitude, orientations
  - Small deformations
    - spatial subsampling
    - local "bag" models



- References
  - "Histograms of oriented gradients for human detection." N. Dalal and B. Triggs, CVPR 2005.
  - "Finding people in images and videos." N. Dalal, Ph.D. Thesis, Institut National Polytechnique de Grenoble, 2006.

HOG feature computation

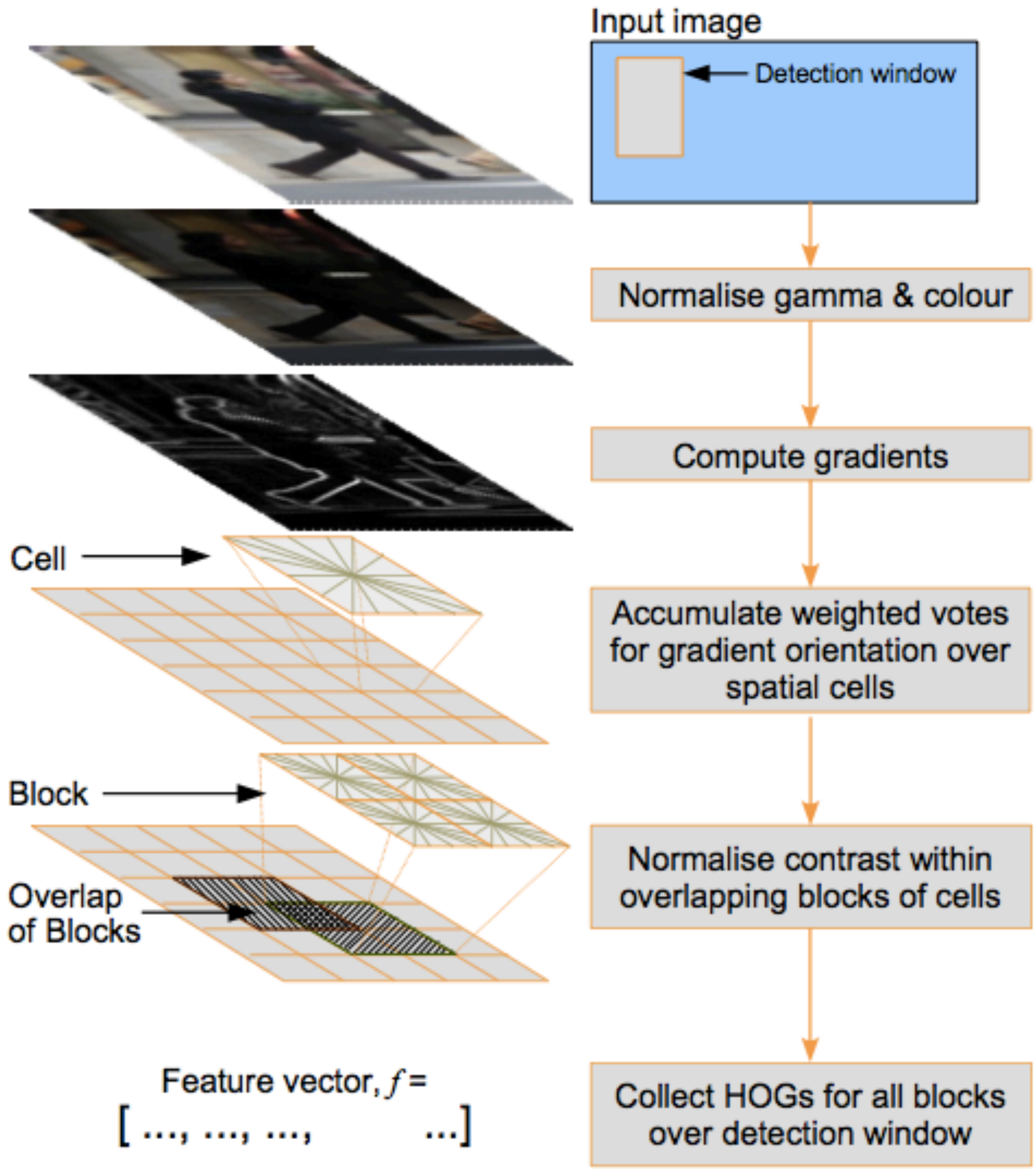
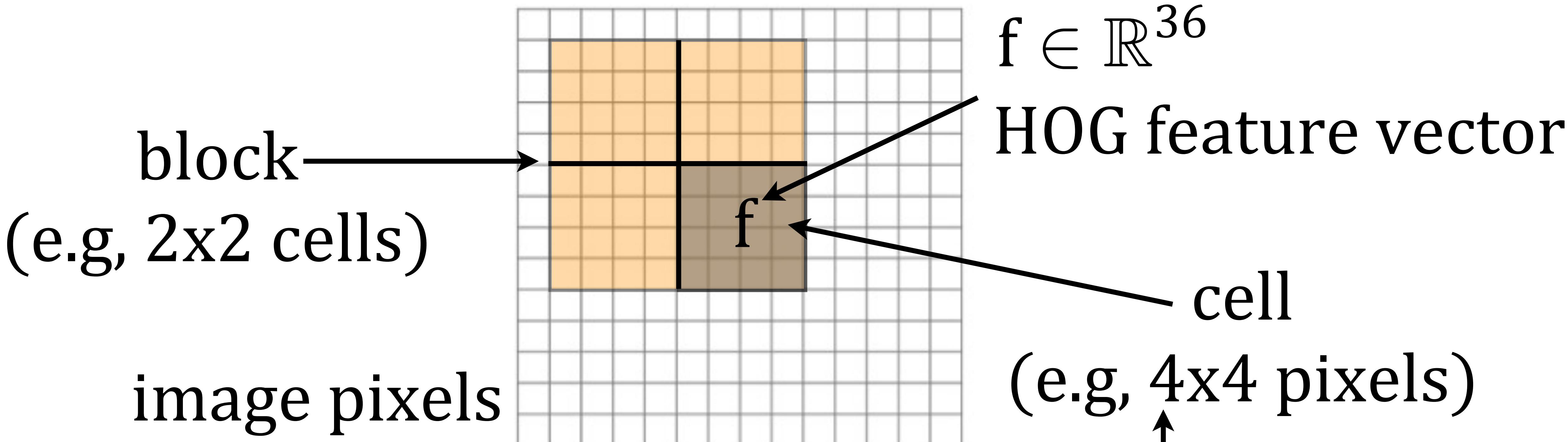
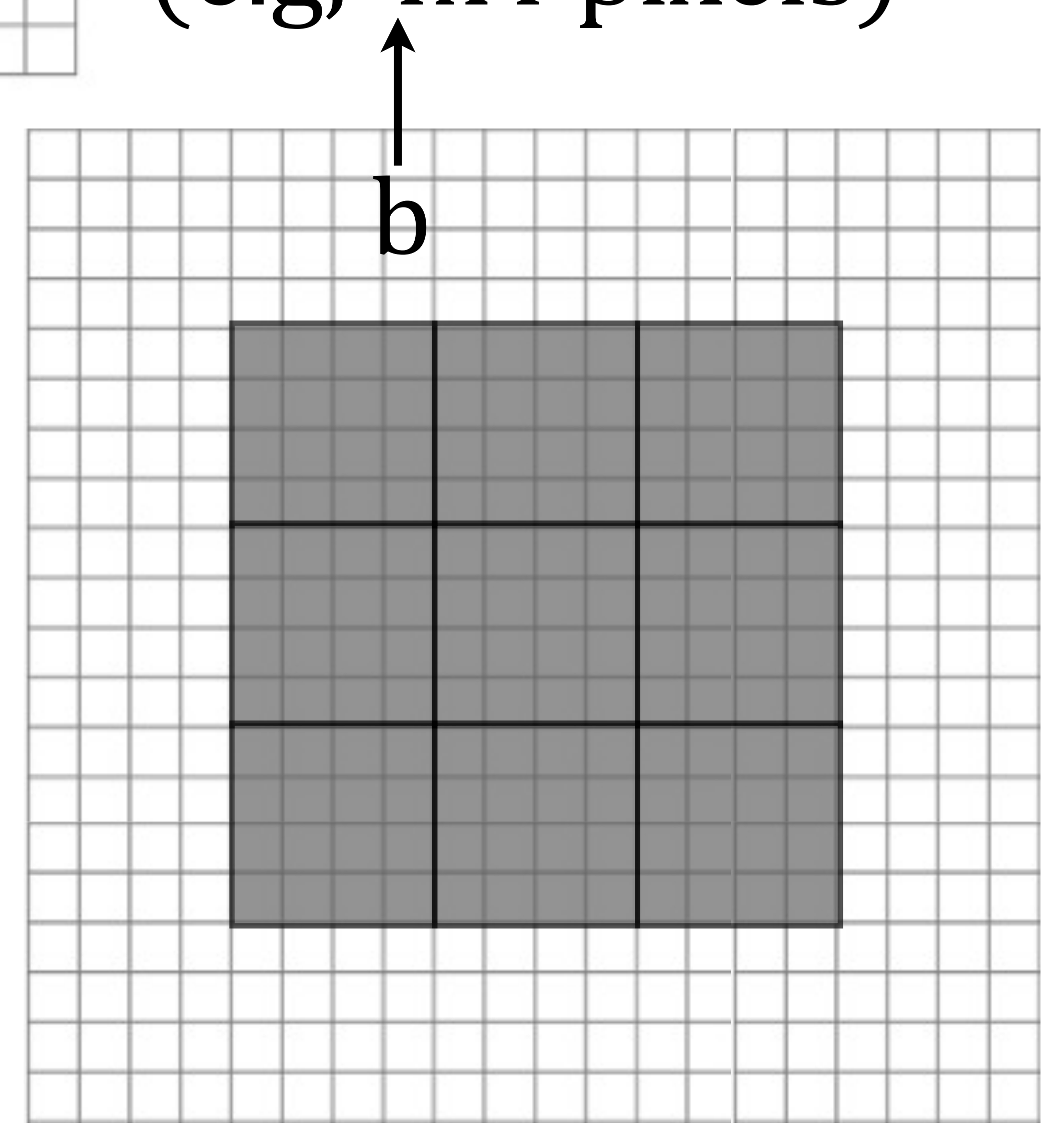


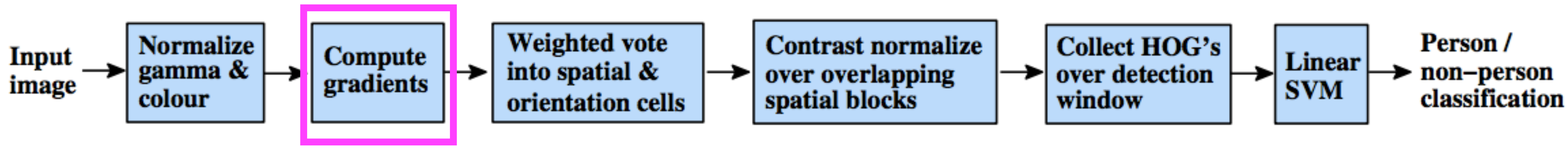
Image: [Dalal06]

HOG terminology

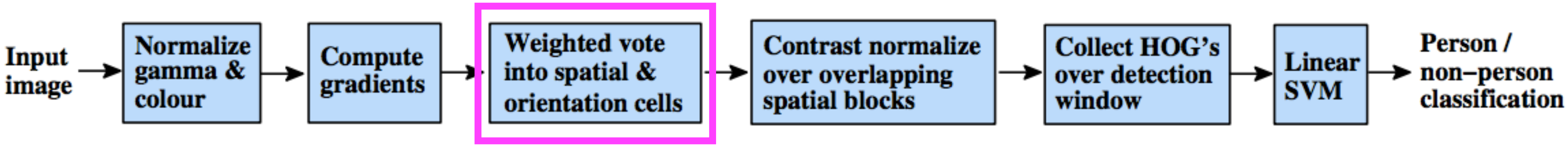


- Original image: H x W x 3
- Feature map: H’ x W’ x D
    - For example
        - H’ = floor(H/b) - 2
        - W’ = floor(W/b) - 2
        - D = 36

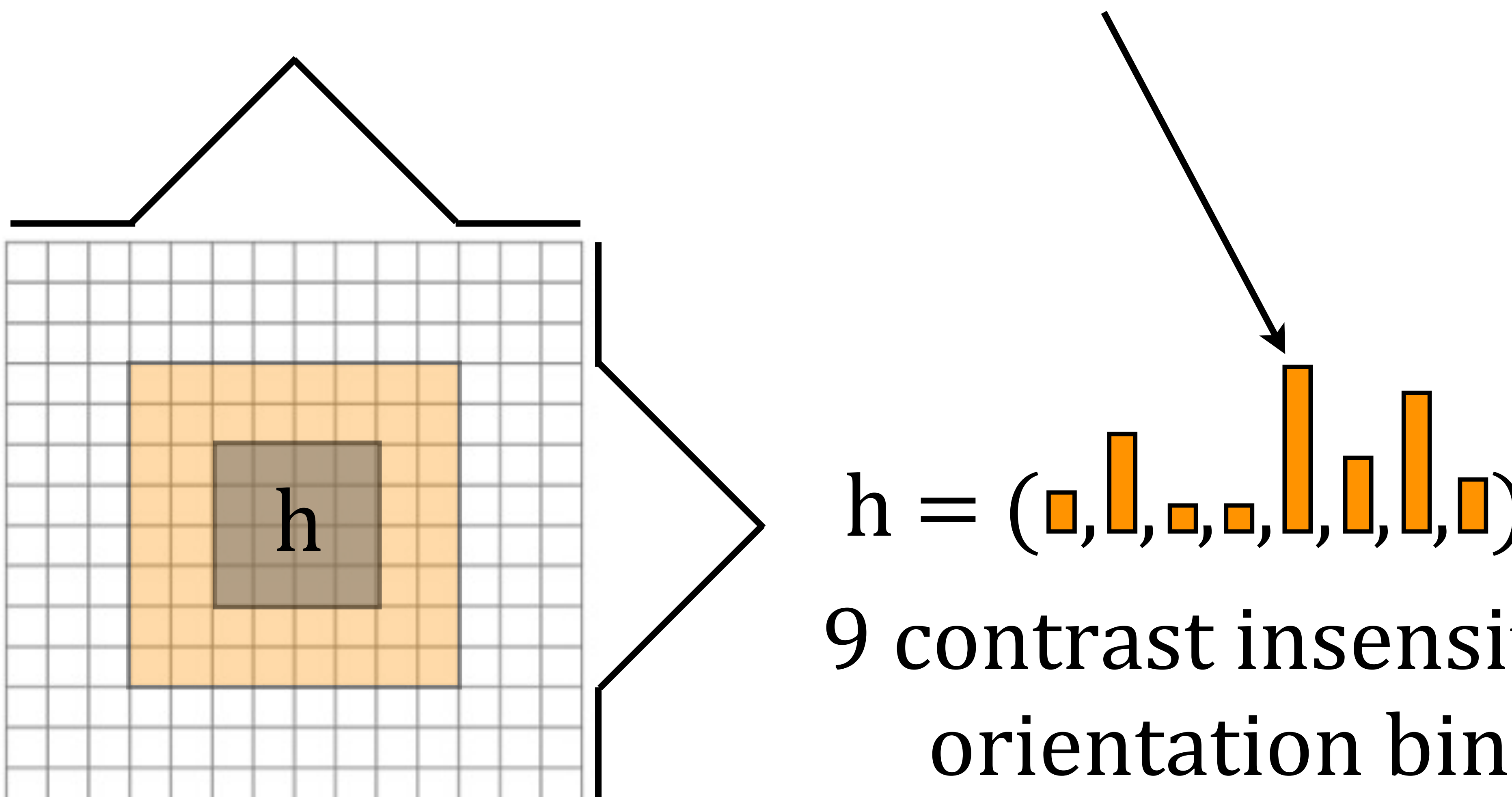




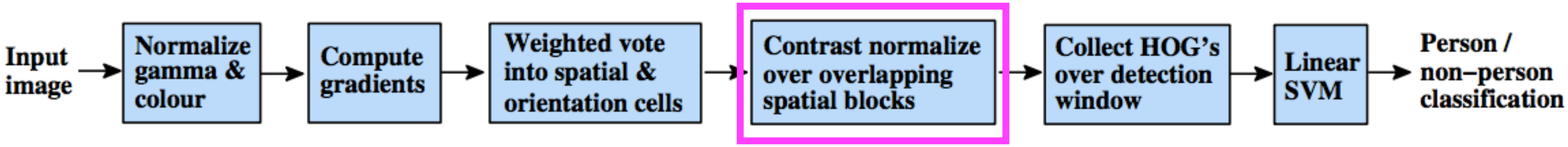
- Many methods
  - (1, 0, -1) centered filter works best
  - Alternatives: uncentered, cubic corrected, Sobel, etc.
- Discrete approx. to partial derivatives
  - $I_x = I[x+1, y] - I[x-1, y]$
  - $I_y = I[x, y+1] - I[x, y-1]$



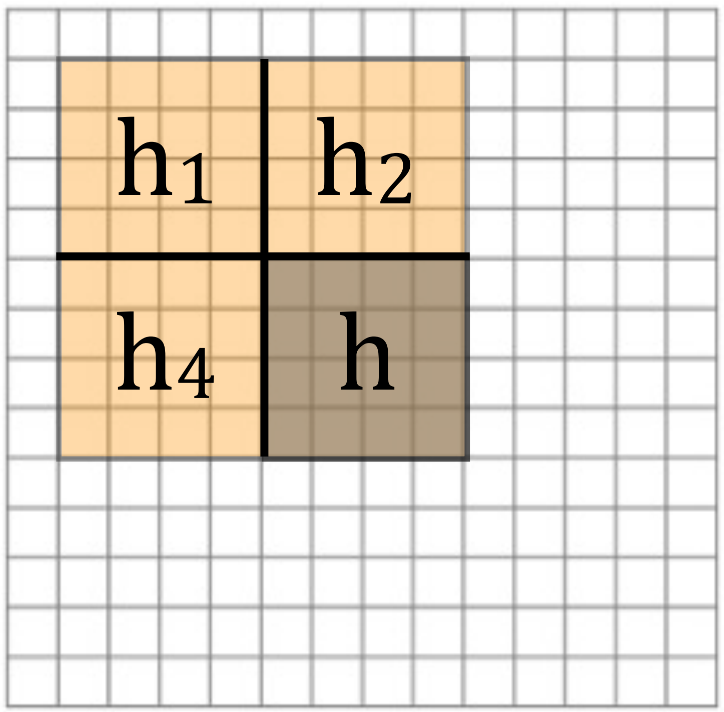
- At each pixel compute
  - Gradient magnitude: $m = ||\,(I_x, I_y)\,||$
  - Gradient orientation: $o = \tan^{-1}(I_y \,/\, I_x)$
  - Quantize orientation; vote into bin (weighted)

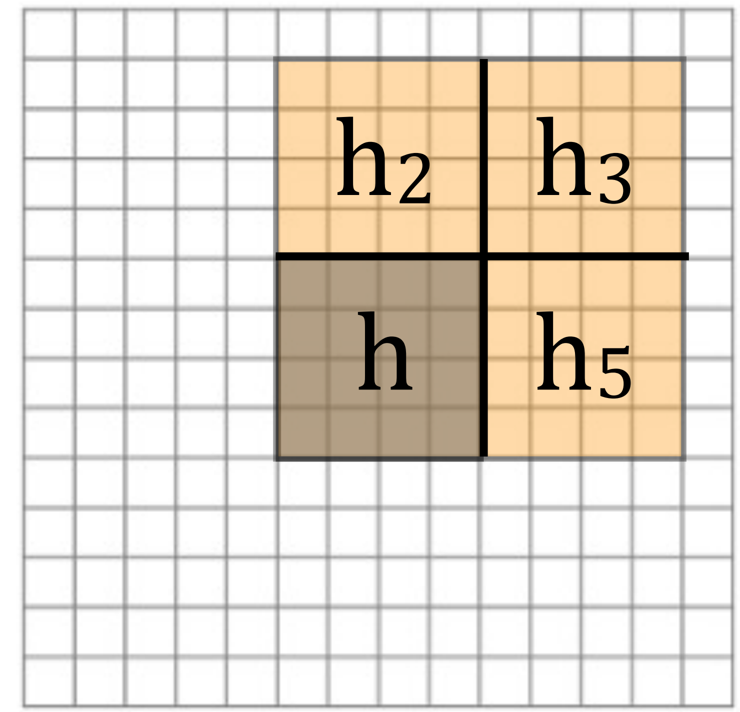


9 contrast insensitive orientation bins



• Local contrast normalization and clipping





$h^1 = \max[\, 0.2, h/||(h; h_1; h_2; h_4)||\,]$

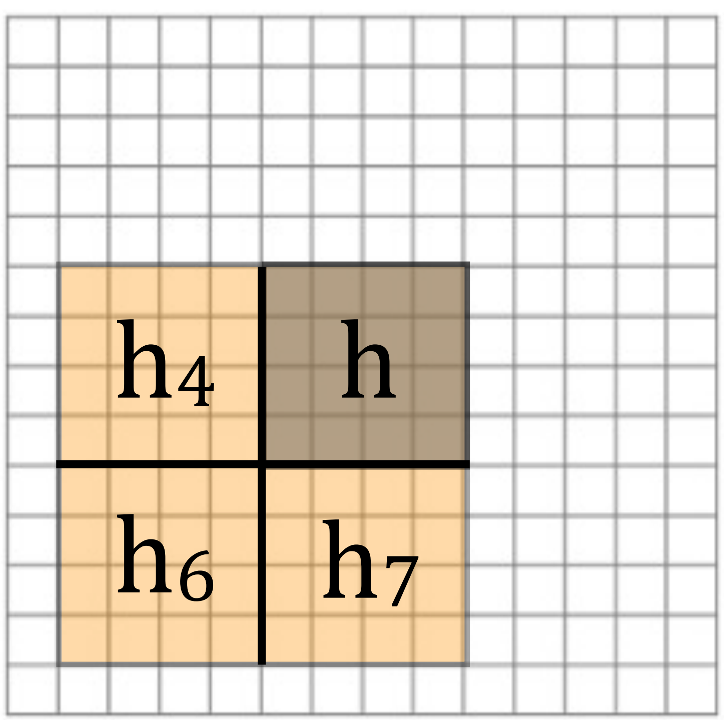
$h^2 = \max[\, 0.2, h/||(h; h_2; h_3; h_5)||\,]$

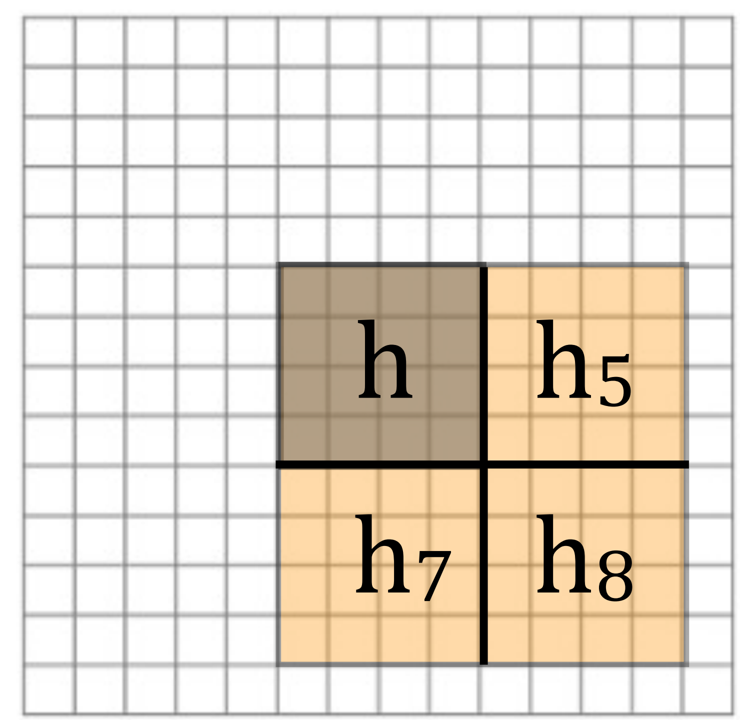
$h^3 = \max[\, 0.2, h/||(h; h_4; h_6; h_7)||\,]$

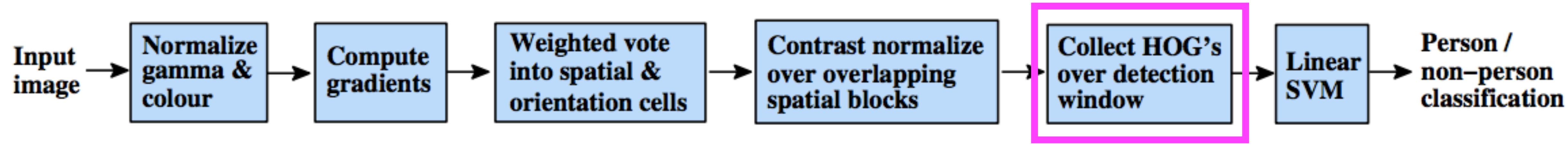
$h^4 = \max[\, 0.2, h/||(h; h_5; h_7; h_8)||\,]$

$f = (h^1; h^2; h^3; h^4)$

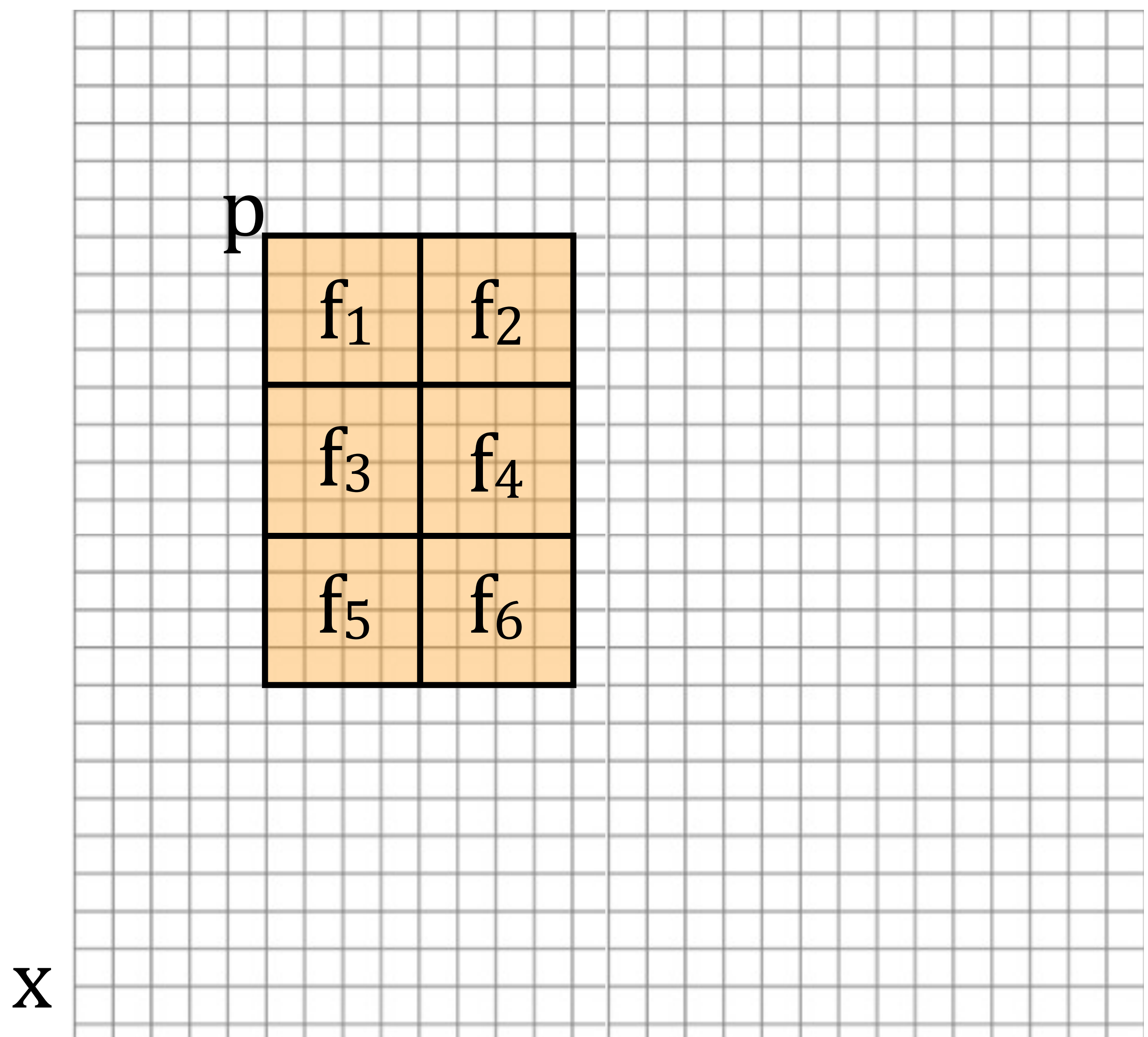
Final dimensionality per cell: 36







- Sliding window feature vector
  - $\Phi(x, p) = (f_1; f_2; f_3; f_4; f_5; f_6)$



Questions?

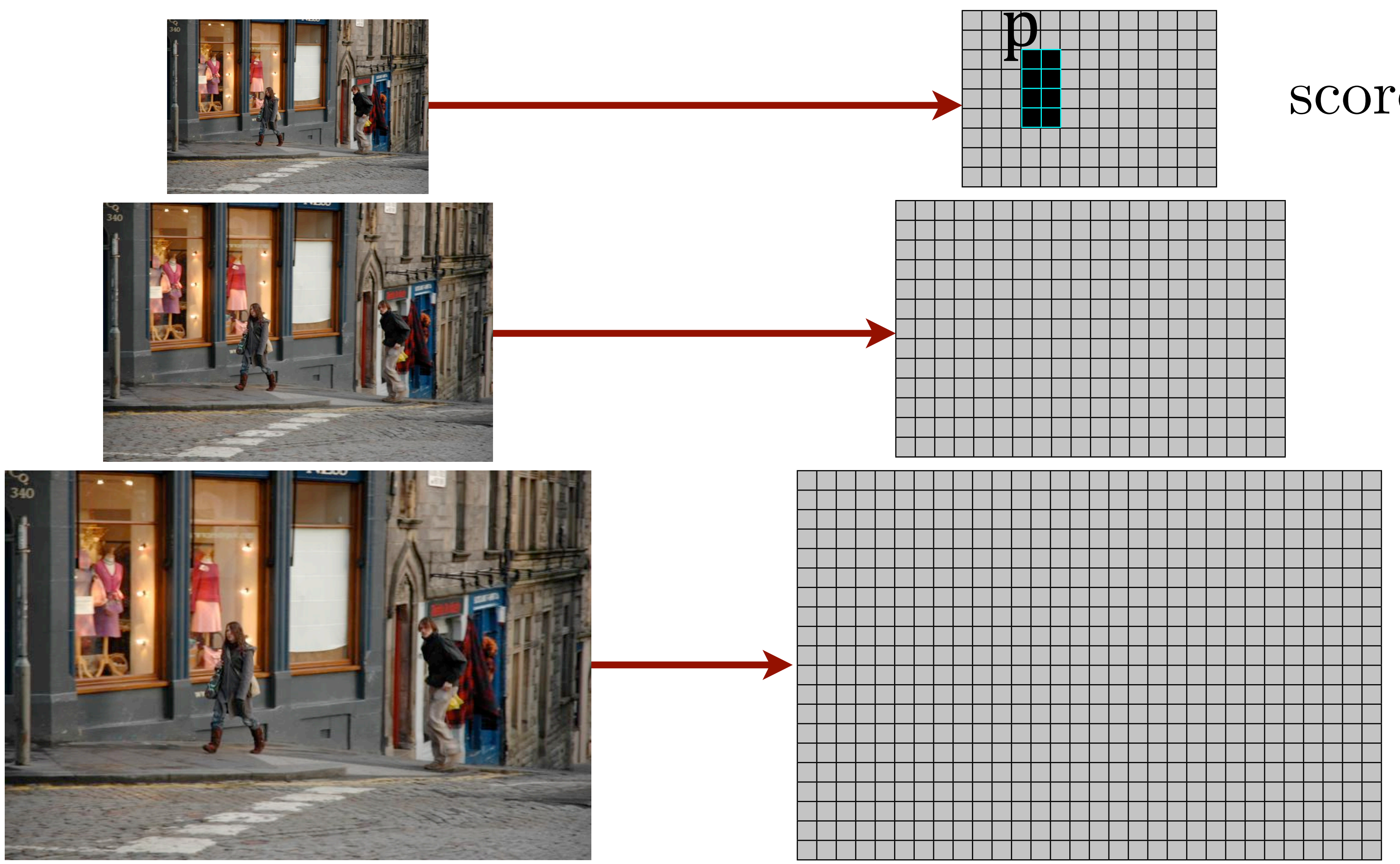


Image pyramid

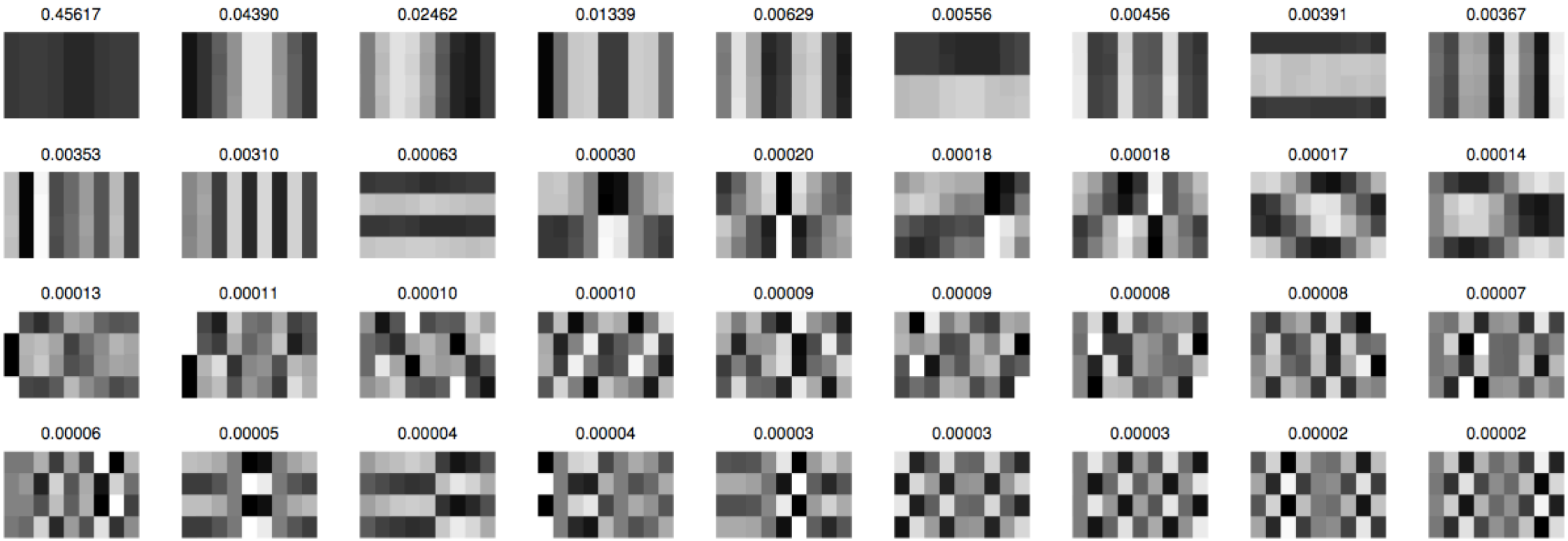
HOG feature pyramid x

$$\text{score}_{\mathbf{w}}(\mathrm{x}, \mathrm{p}) = \mathbf{w} \cdot \Phi(\mathrm{x}, \mathrm{p})$$

[Dalal05]

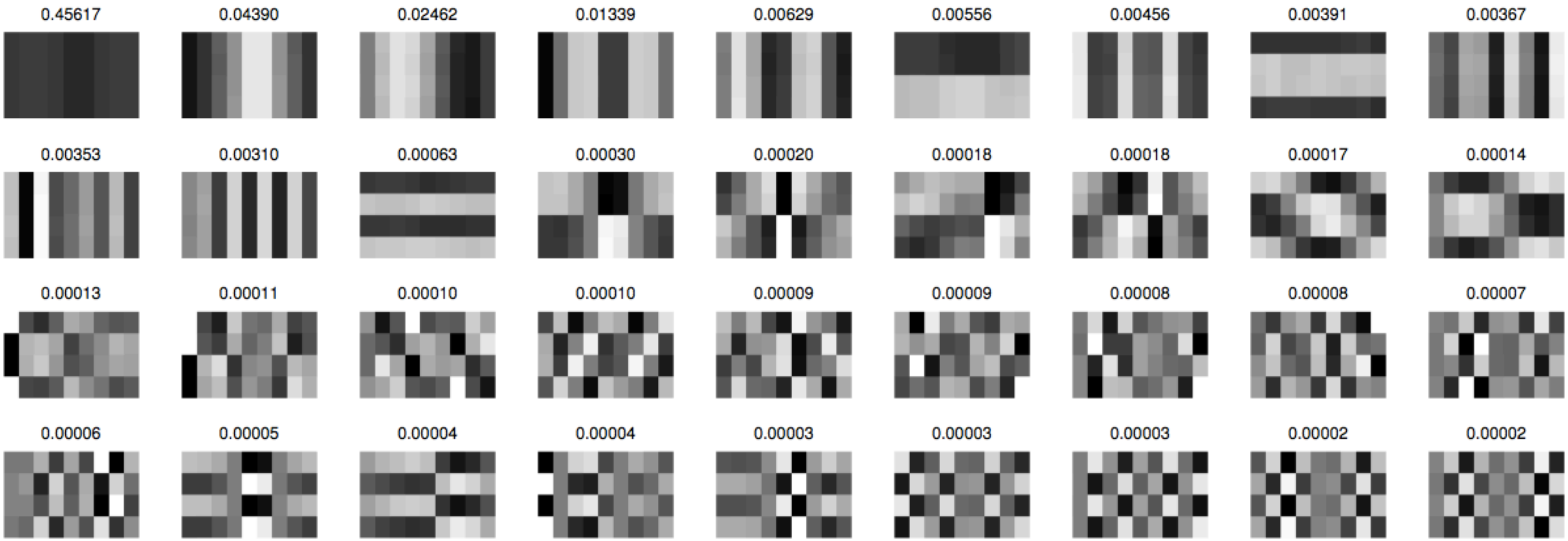
- "Dalal & Triggs detector"
  - HOG feature pyramid
  - Linear filter / sliding-window detector
  - SVM training to learn parameters w

HOG reformulation



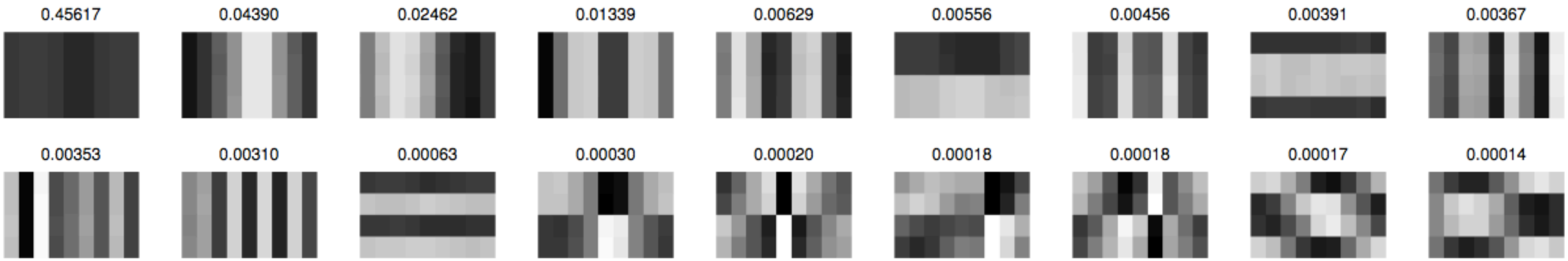
- PCA of HOG features
  - Eigenvectors have a strong structure
  - Dim. reduction to top 12 with no loss in performance

HOG PCA eigenvectors



- Eigenvector structure
  - All rows or columns are (approximately) constant in the top 12 eigenvectors
  - Suggests a different basis

V, a sparse basis for HOG



1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0

u_1

0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0

u_2

...

0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	1

u_9

1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

v_1

...

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1

v_4

New basis $V = \{u_1, \ldots, u_9\} \cup \{v_1, v_2, v_3, v_4\}$

Interpretation of V

Original HOG

$$f = (h^1; h^2; h^3; h^4)$$

Final dimensionality per cell: 36

Analytic projection

$$f = (h^1 + h^2 + h^3 + h^4; 1 \cdot h^1; 1 \cdot h^2; 1 \cdot h^3; 1 \cdot h^4)$$

Final dimensionality per cell: 13

HOG summary

- There's no one true HOG feature
  - Large number of parameters and design choices (see Dalal's thesis)
  - Typical settings
    - cells: 6-8 pixels wide
    - cell blocks: 2x2 or 3x3 rectangular
- The original formulation contains redundant information
  - Efficient and intuitive dimensionality reduction by analytic projection

Pictorial structure models

- Parts — many appearance templates
- “Springs” — spatial connections between parts

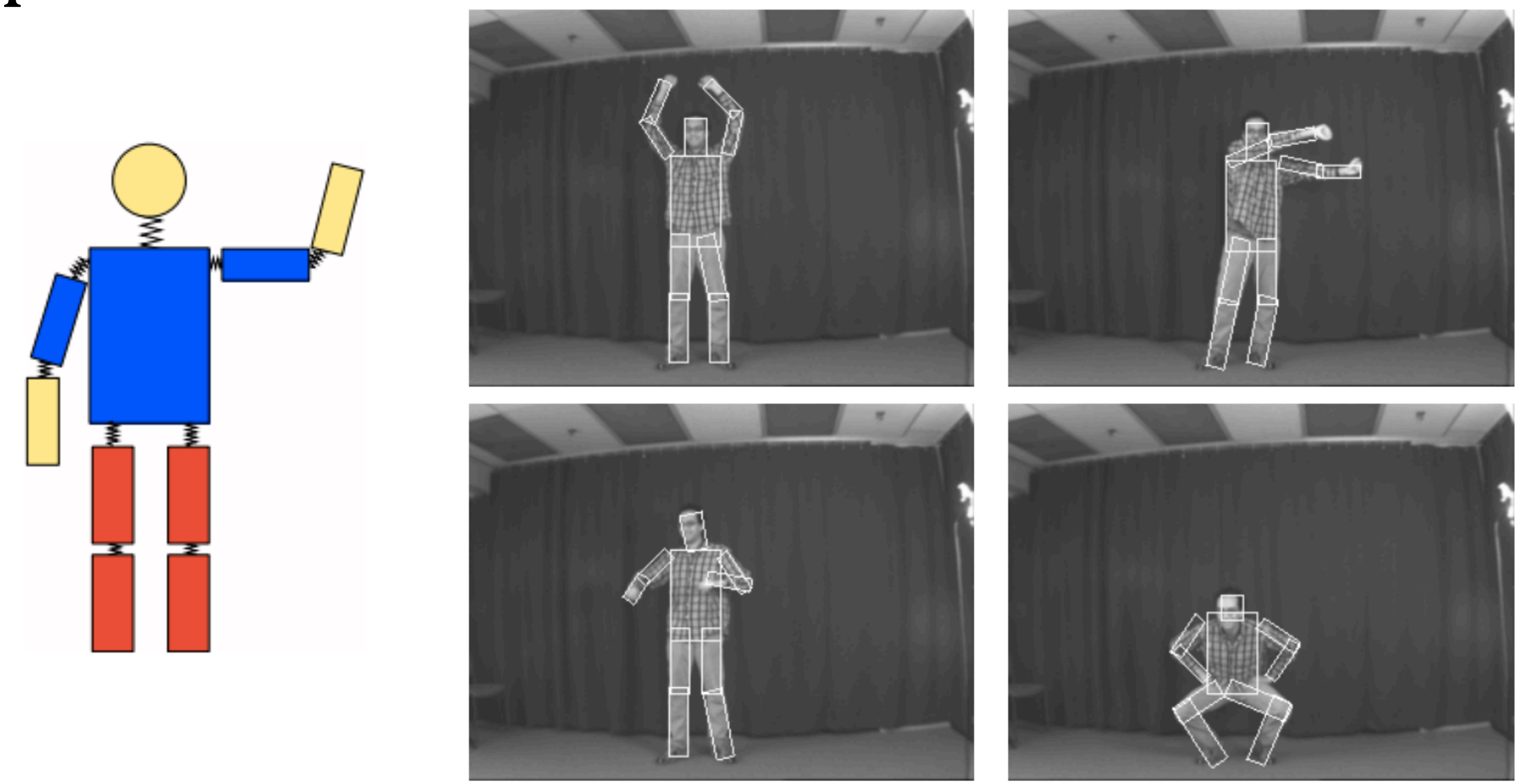


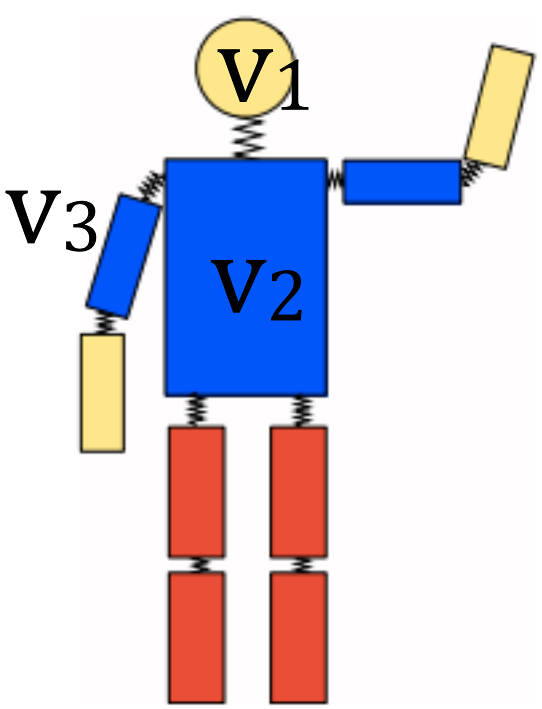
Image: [Felzenszwalb and Huttenlocher 05]

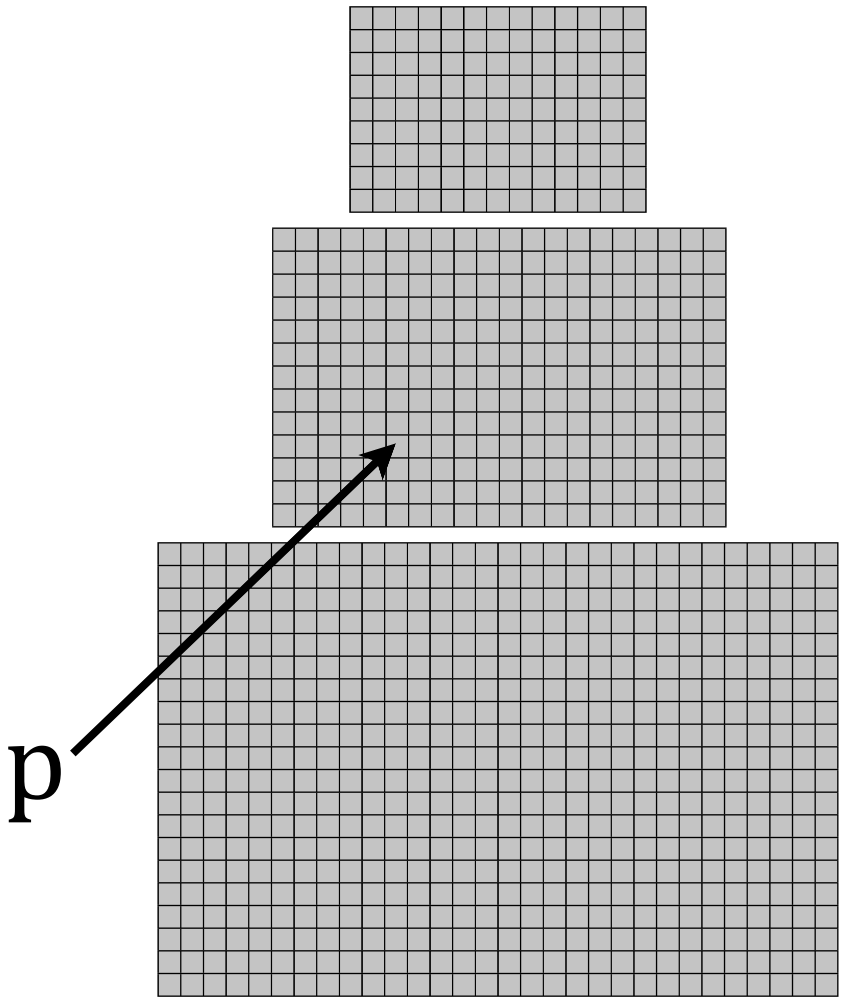
PS formulation

$$G = (V, E)$$

$$V = (v_1, \ldots, v_n) \quad E \subseteq V \times V$$

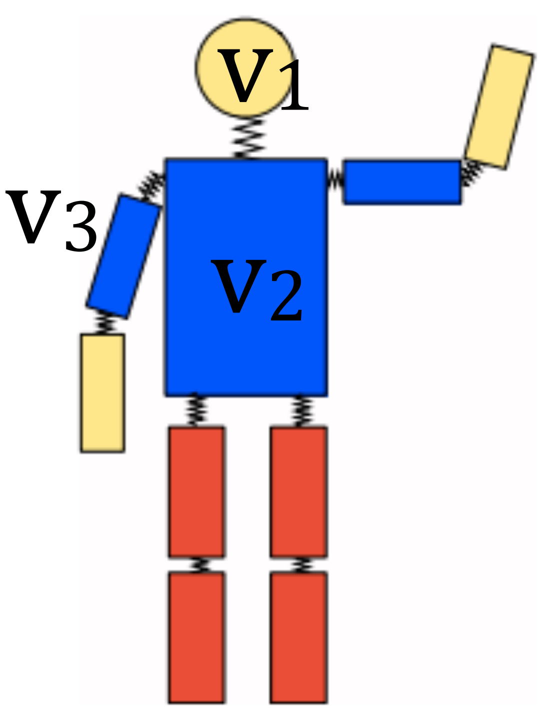
$$(p_1, \ldots, p_n) \in P^n$$

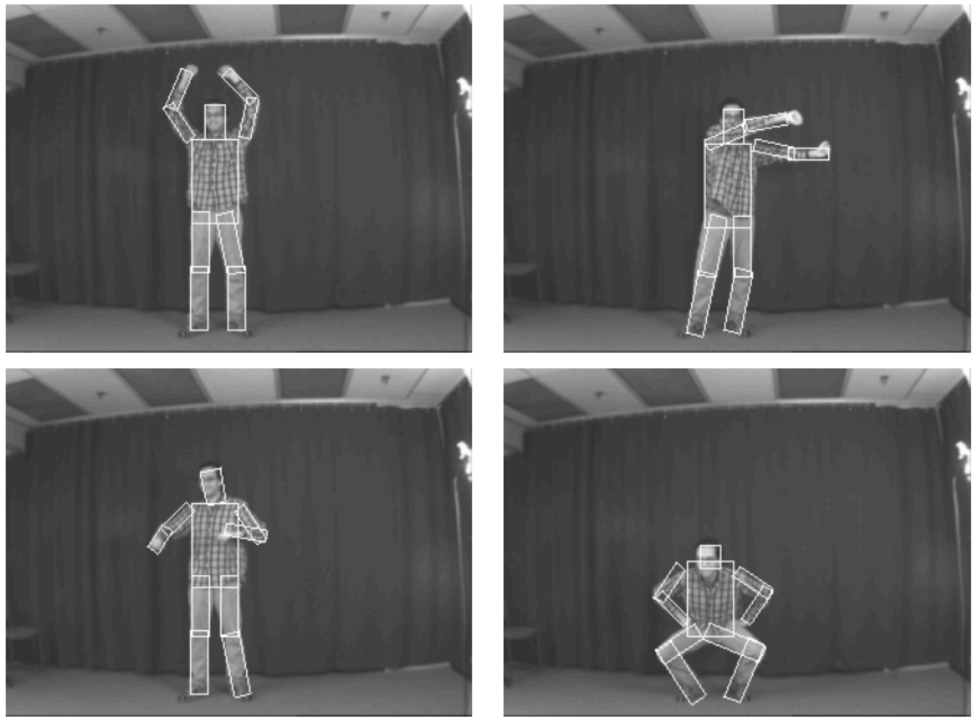


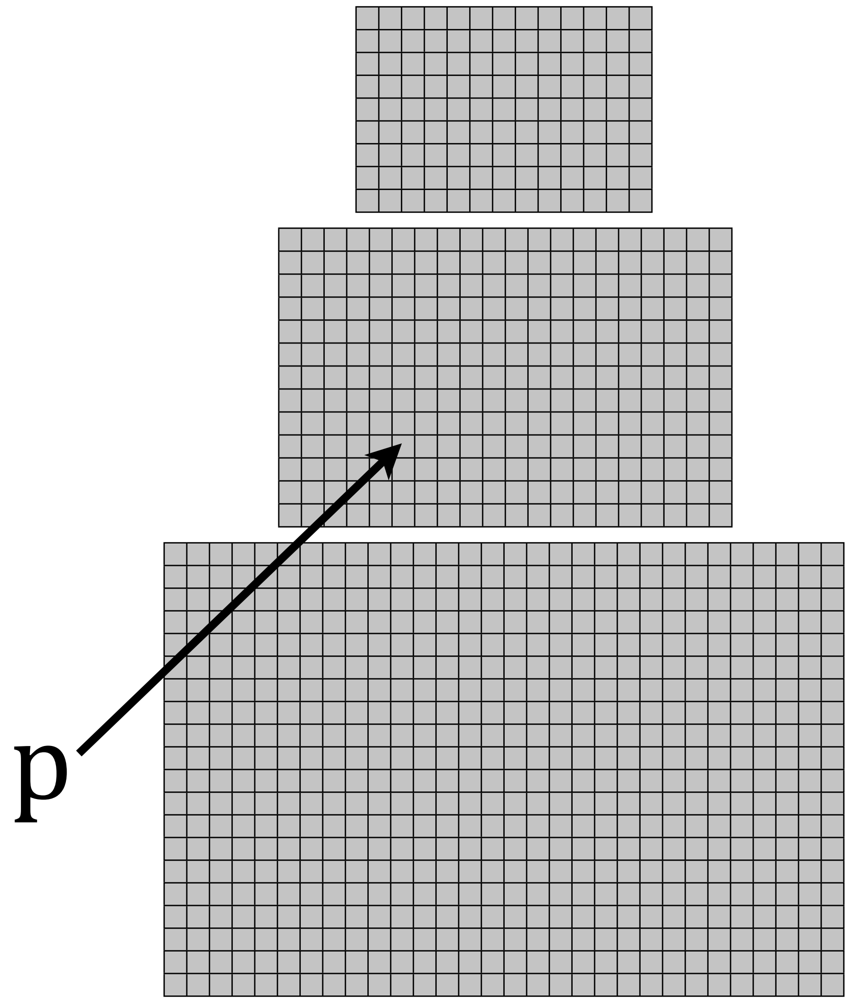


PS score function for matching

$$\text{score}(p_1, \ldots, p_n) = \sum_{i=1}^{n} m_i(p_i) + \sum_{(i,j) \in E} d_{ij}(p_i, p_j)$$







PS score function for matching

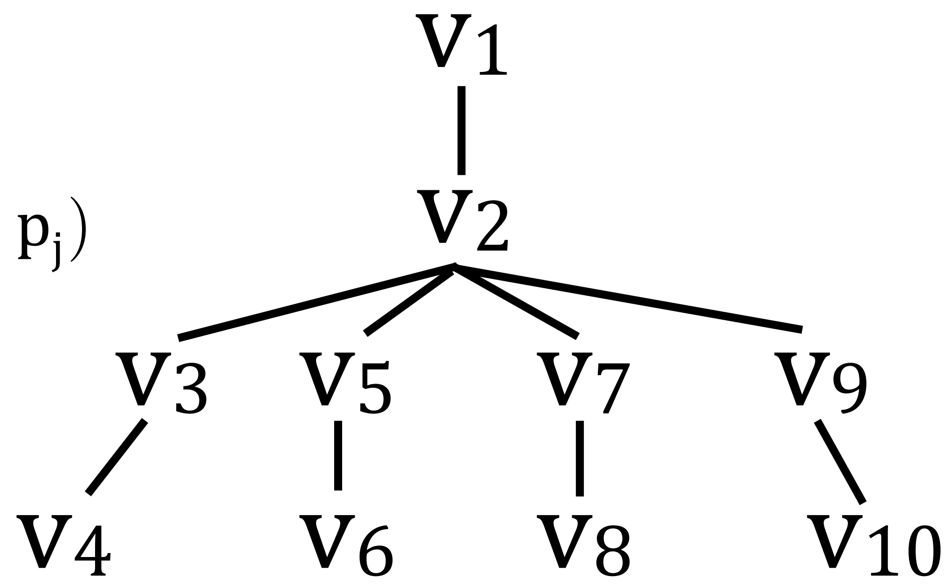
$$\text{score}(p_1, \ldots, p_n) = \sum_{i=1}^{n} m_i(p_i) + \sum_{(i,j) \in E} d_{ij}(p_i, p_j)$$

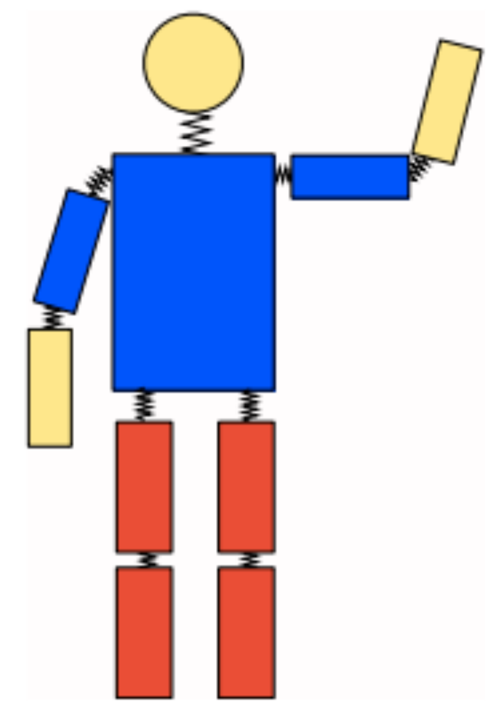
- Objective: maximize score over $p_1,...,p_n$
- h^n configurations! ($h = |P|$)
- If $G = (V,E)$ is a tree, $O(nh^2)$ algorithm
  - $O(nh)$ with some restrictions on d_{ij}

Dynamic programming on a tree

maximize:

$$\text{score}(p_1, \dots, p_n) = \sum_{i=1}^{n} m_i(p_i) + \sum_{(i,j)\in E} d_{ij}(p_i, p_j)$$





$$B_j(p_i) = \max_{p_j} \left[m_j(p_j) + d_{ij}(p_i, p_j) \right]$$ if j is a leaf

$$B_j(p_i) = \max_{p_j} \left[m_j(p_j) + d_{ij}(p_i, p_j) + \sum_{c\in C_j} B_c(p_j) \right]$$

$$B_r = \max_{p_r} \left[m_r(p_r) + \sum_{c\in C_r} B_c(p_r) \right]$$ root part r

Dynamic programming on a tree

- Compute B_j in depth-first order
- When done

$$B_r = \max_{p_1,\ldots,p_n} \text{score}(p_1,\ldots,p_n)$$

Dynamic programming on a tree

$$B_j(p_i) = \max_{p_j} \left[m_j(p_j) + d_{ij}(p_i, p_j) \right]$$

- In general, $O(nh^2)$
- If $d_{ij}(p_i,p_j) = g(p_i - p_j)$, g is convex, can use generalized distance transforms
  - practical O(nh) algorithm [Felzenszwalb and Huttenlocher]
- If $d_{ij}(p_i,p_j)$ is finite over a small, bounded region
  - O(nh) brute force with a small constant

Where do m_i and d_{ij} come from?

- The machine learning approach
  - the computer learns them from training examples
- We'll talk about discriminative training later today

PS summary

- Questions?

Recall the Dalal & Triggs detector

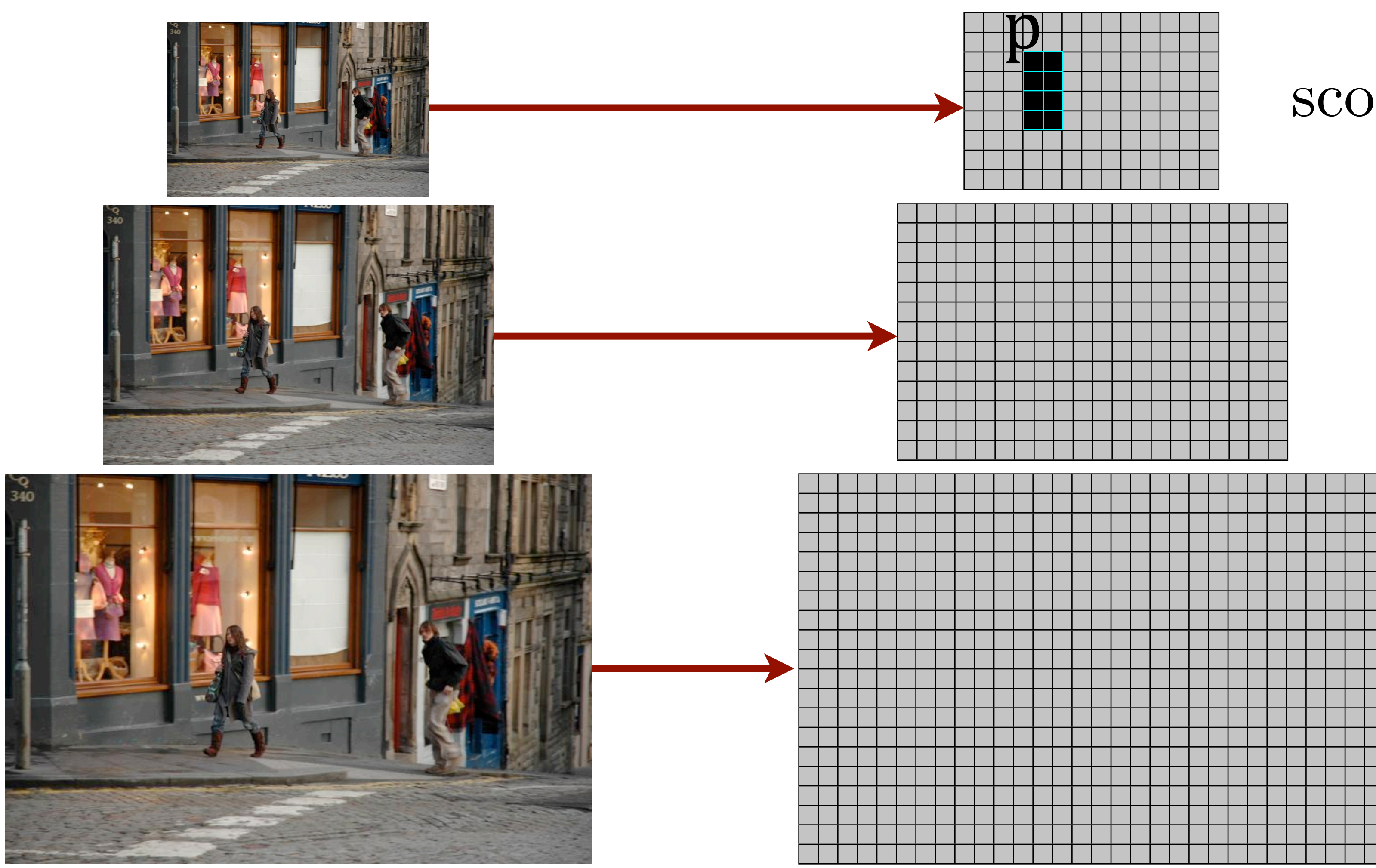
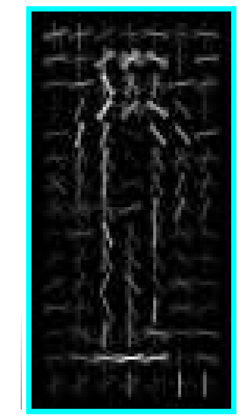


Image pyramid

HOG feature pyramid x

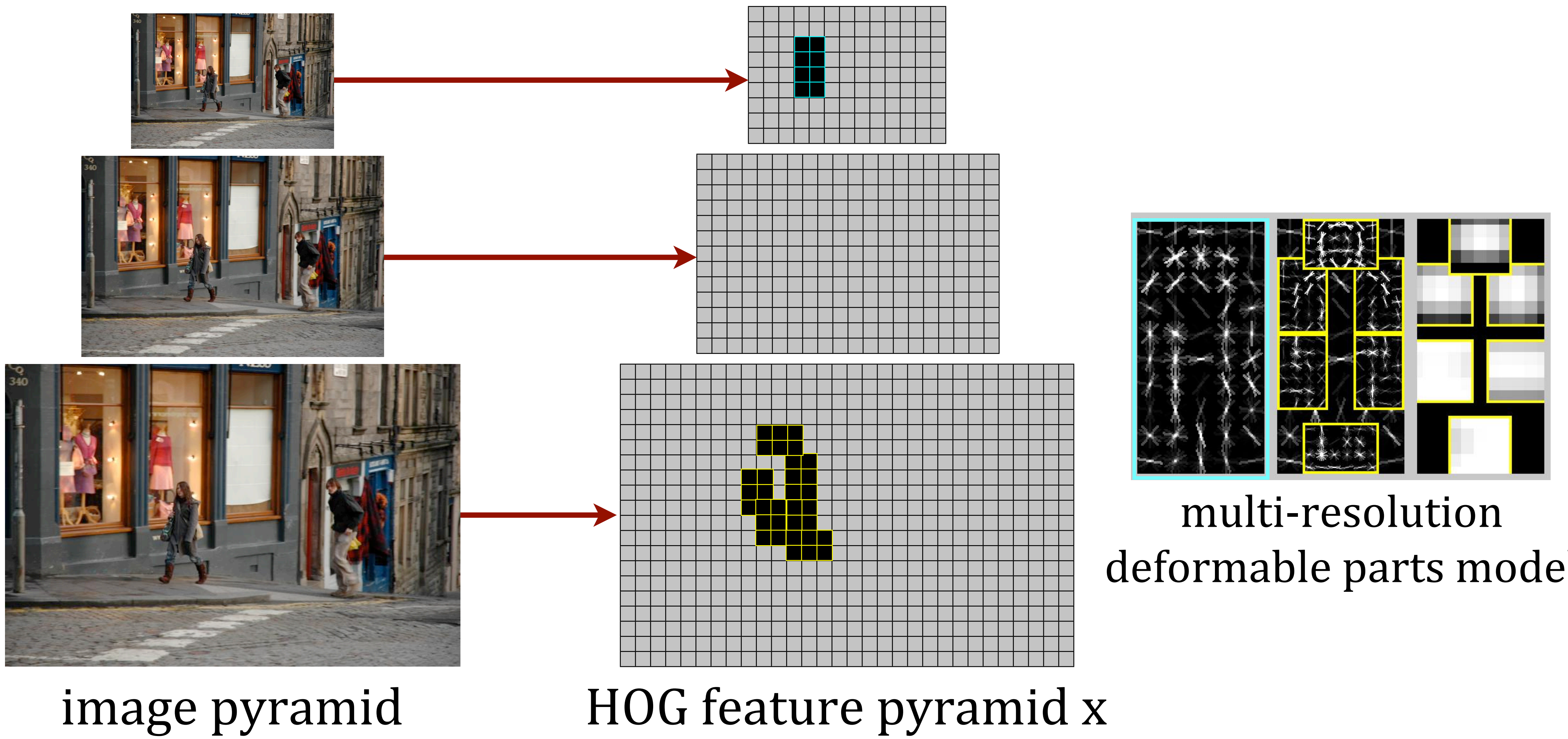
$$\text{score}_{\mathbf{w}}(\mathrm{x}, \mathrm{p}) = \mathbf{w} \cdot \Phi(\mathrm{x}, \mathrm{p})$$



[Dalal05]

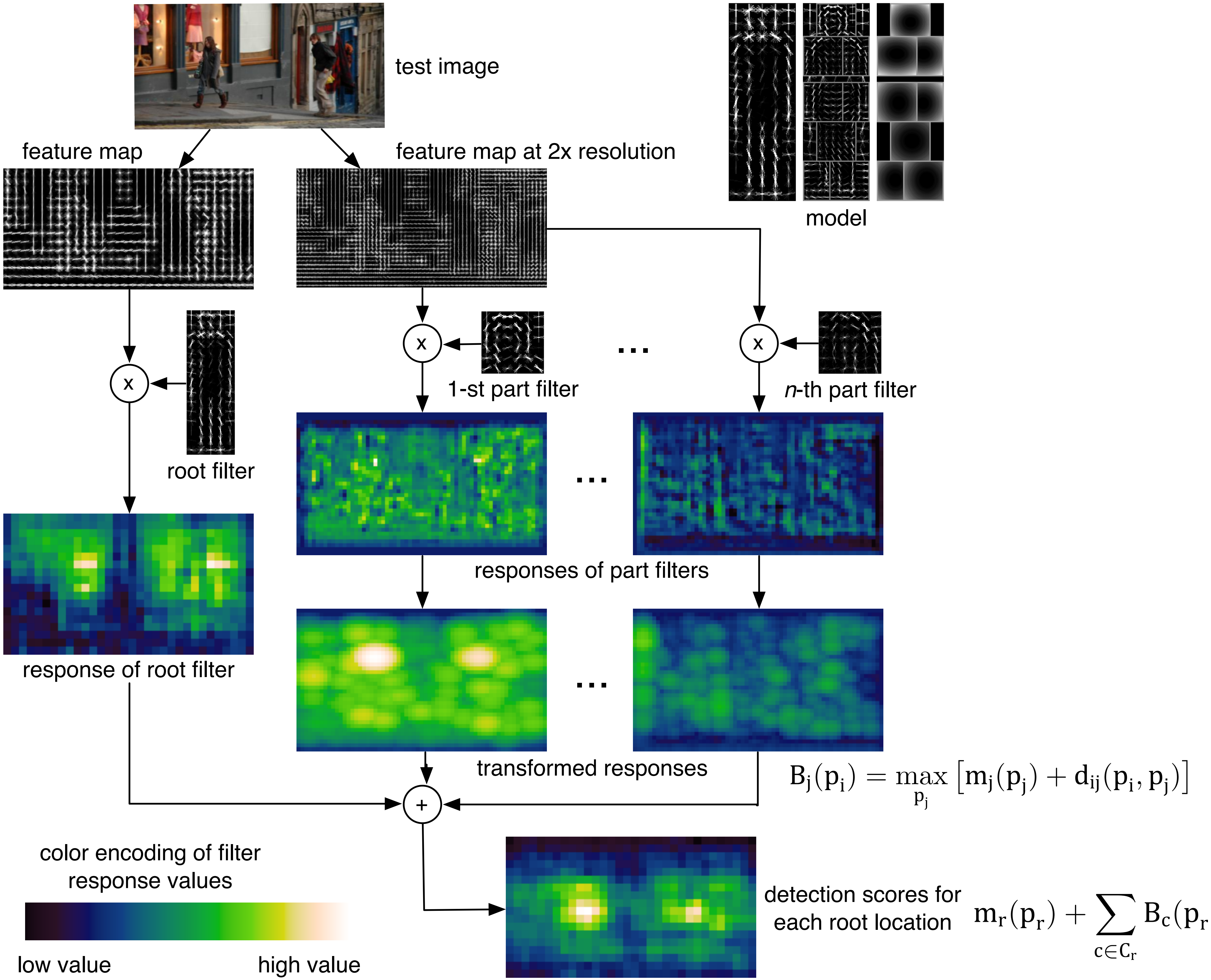
- "Dalal & Triggs detector"
  - HOG feature pyramid
  - Linear filter / sliding-window detector
  - SVM training to learn parameters w

PS + HOG + discriminative training



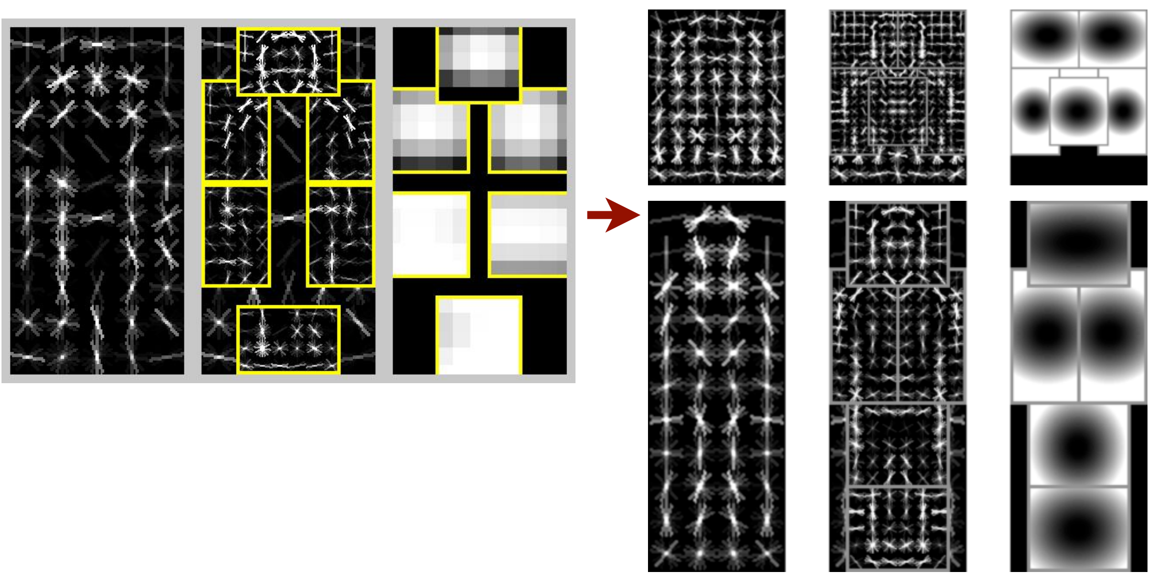
- Combine PS with D&T approach
  - HOG features
  - Linear filters / sliding-window detector
  - Discriminative max-margin (SVM) training

Detection with DPM

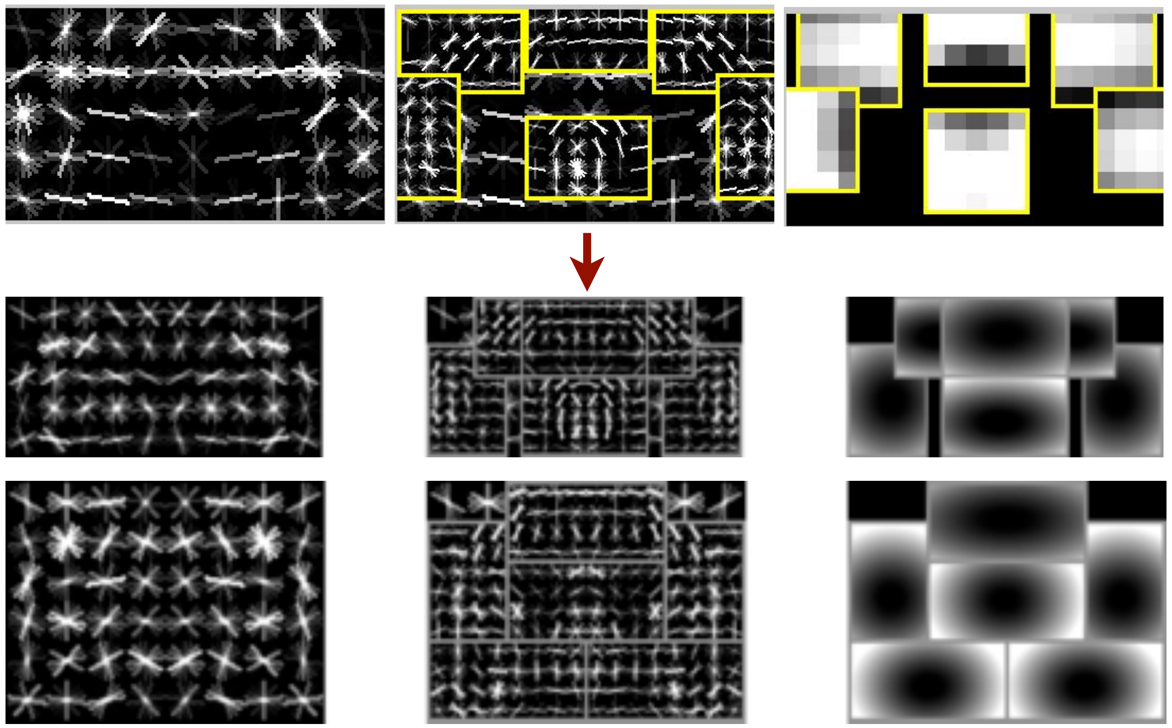


Mixtures of deformable parts models

person



car

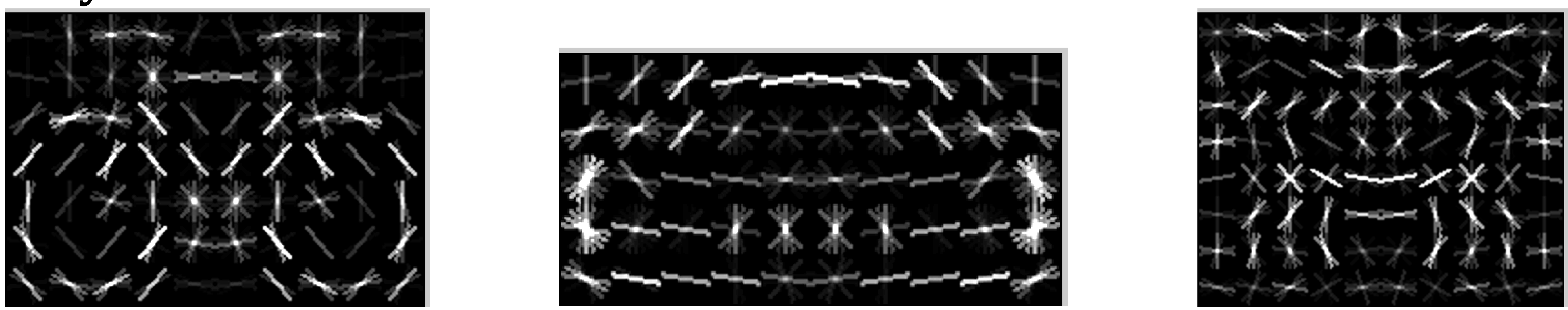


- Captures viewpoint variation and occlusion
- Aspect ratio clustering and discriminative training

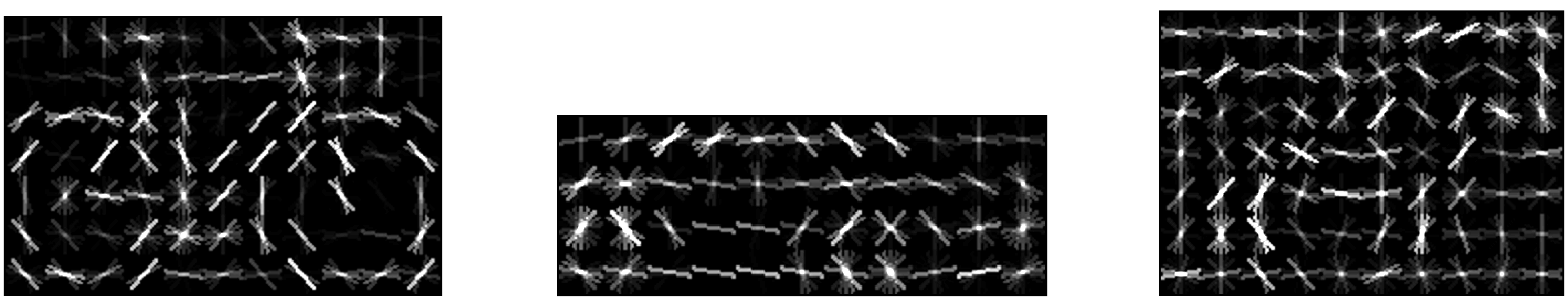
Mixture models: [Weber00, Schneiderman00, Bernstein05]
[Felzenszwalb,Girshick,McAllester,Ramanan in PAMI 10]

Mixtures with latent orientation

bicycle car horse



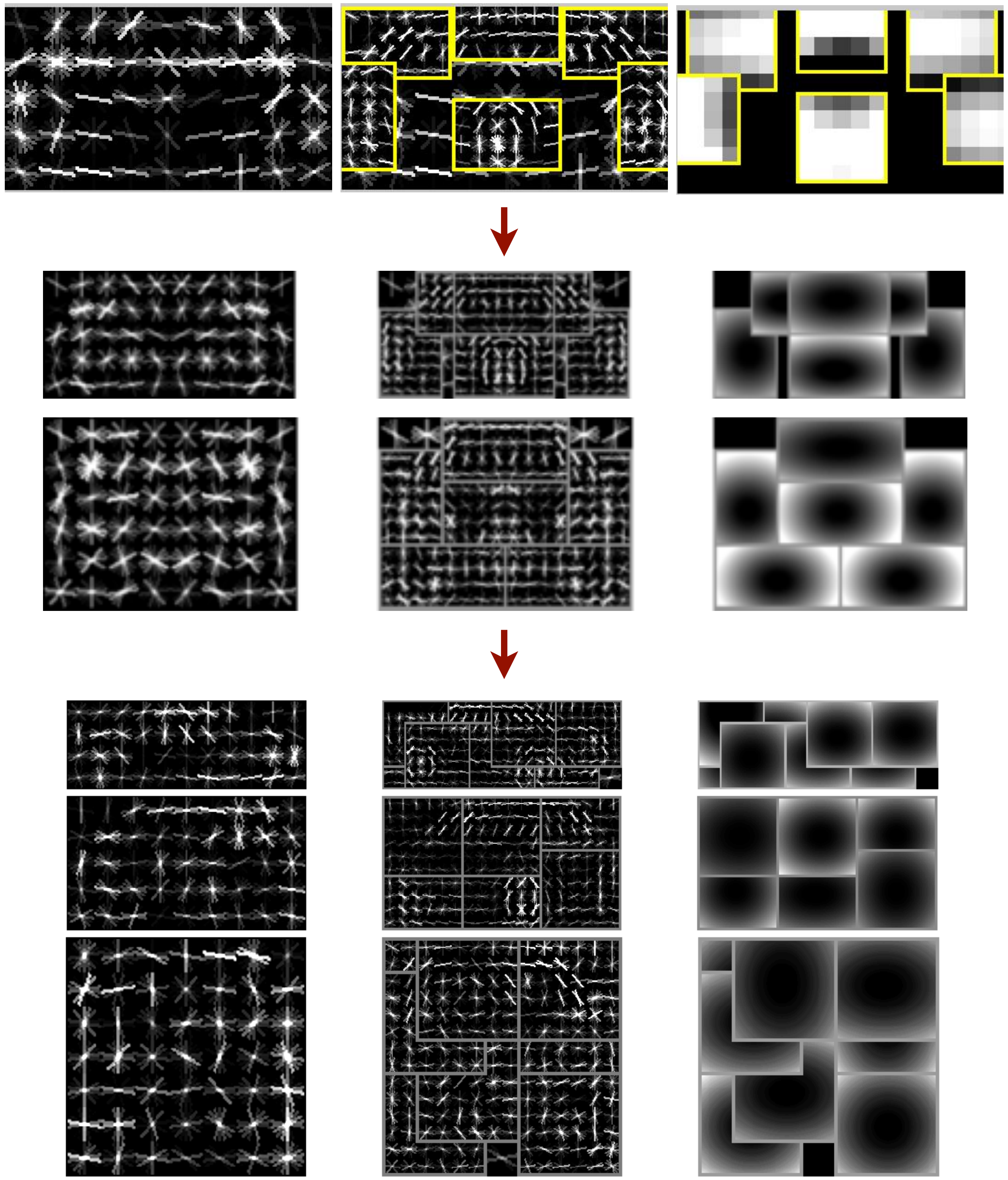
Learning without latent orientation



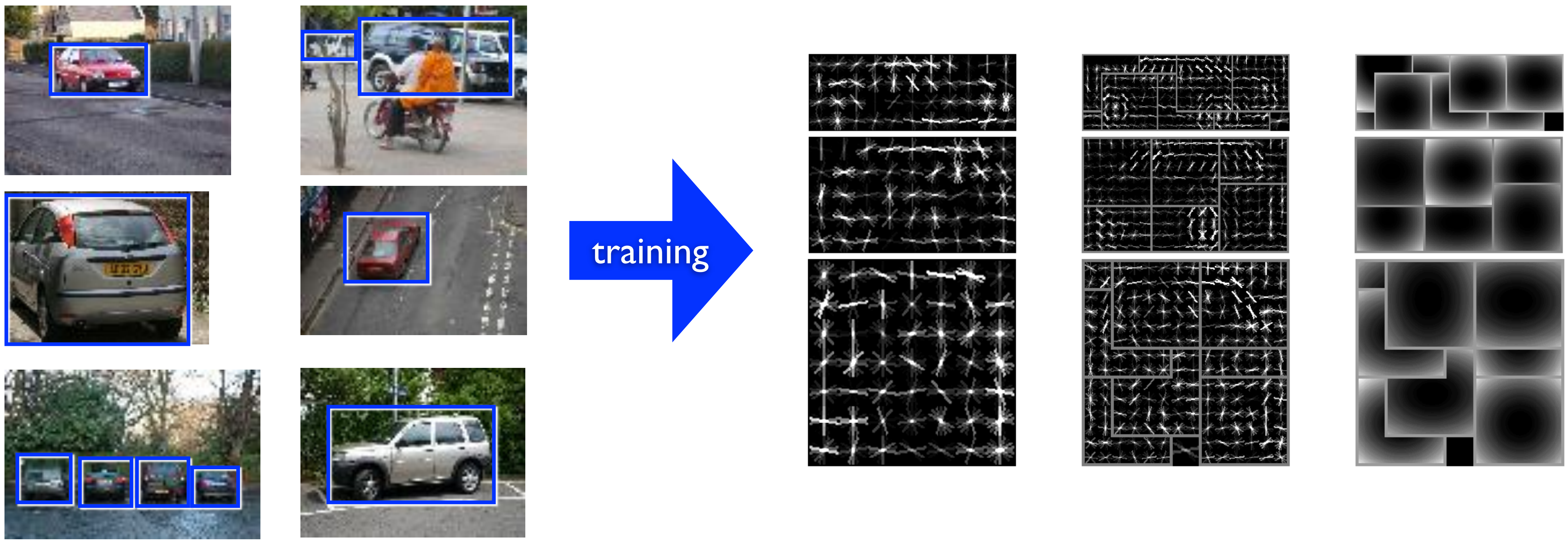
Learning with latent orientation

[Girshick,Felzenszwalb,McAllester voc-release4]

Questions about model structure?



Training models



From images annotated with bounding boxes...

1. learn model structure
2. learn model parameters

(not) Learning model structure

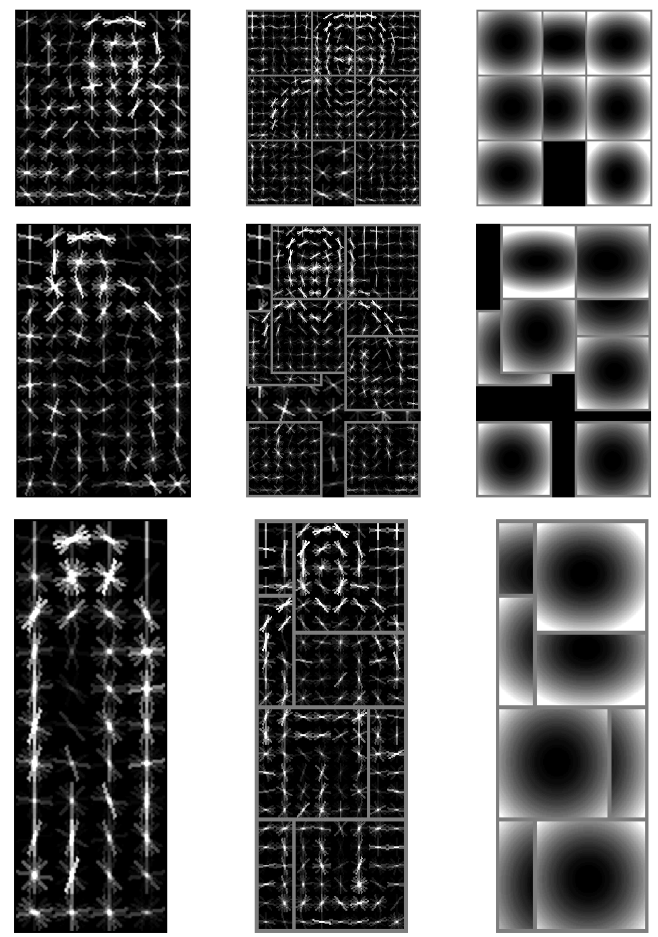
What's the model class?

Number of components?

Number of parts?

Root filter sizes?

Anchor positions?



Root filter shapes?

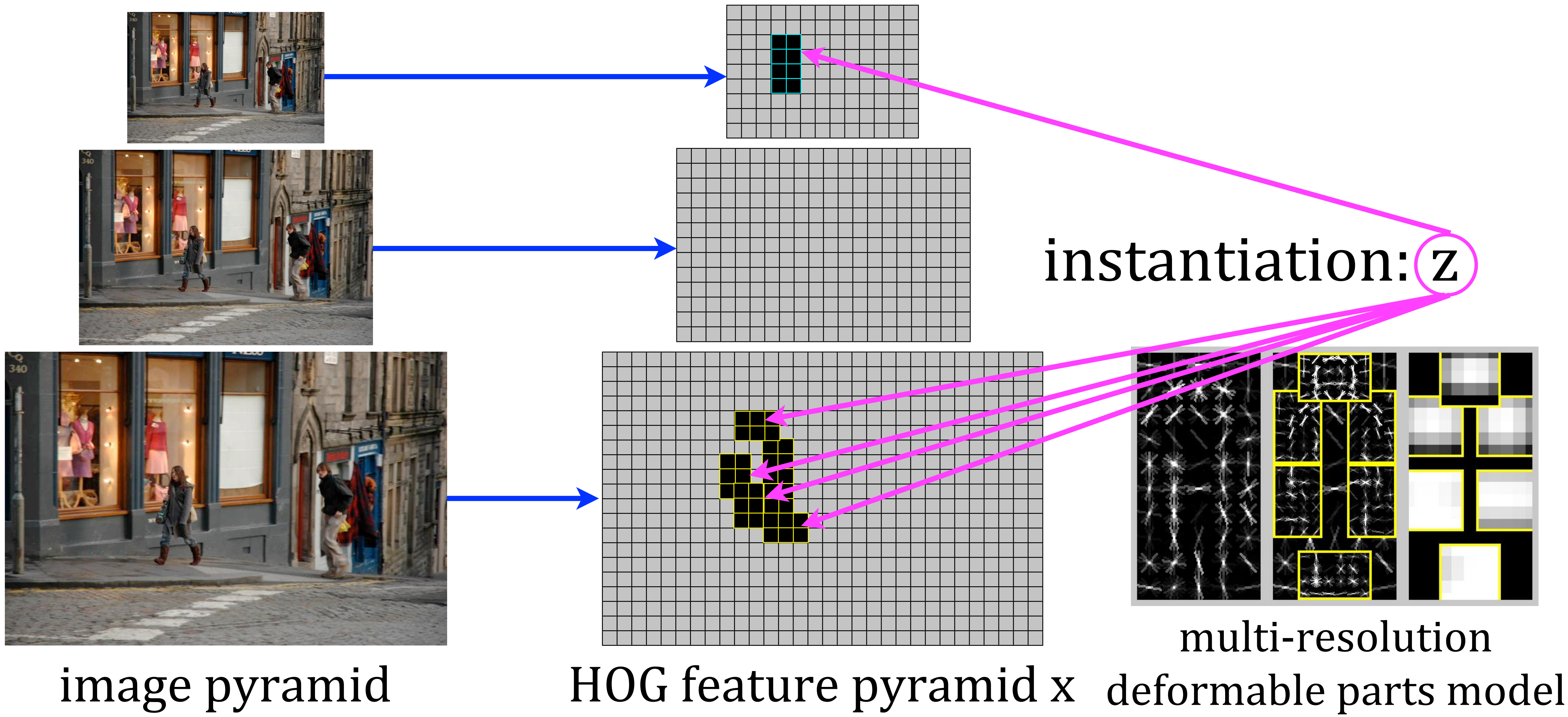
Part shapes and sizes?

Heuristics, cross validation, insight (from humans)

Learning model parameters

- Dalal & Triggs successful combination of
  - HOG features
  - Linear SVM training
- This training problem is different
  - Training data is weakly/partially labeled
  - Several latent (unobserved) variables
    - Filter placement
    - Mixture component
    - Orientation

Linear parameterization

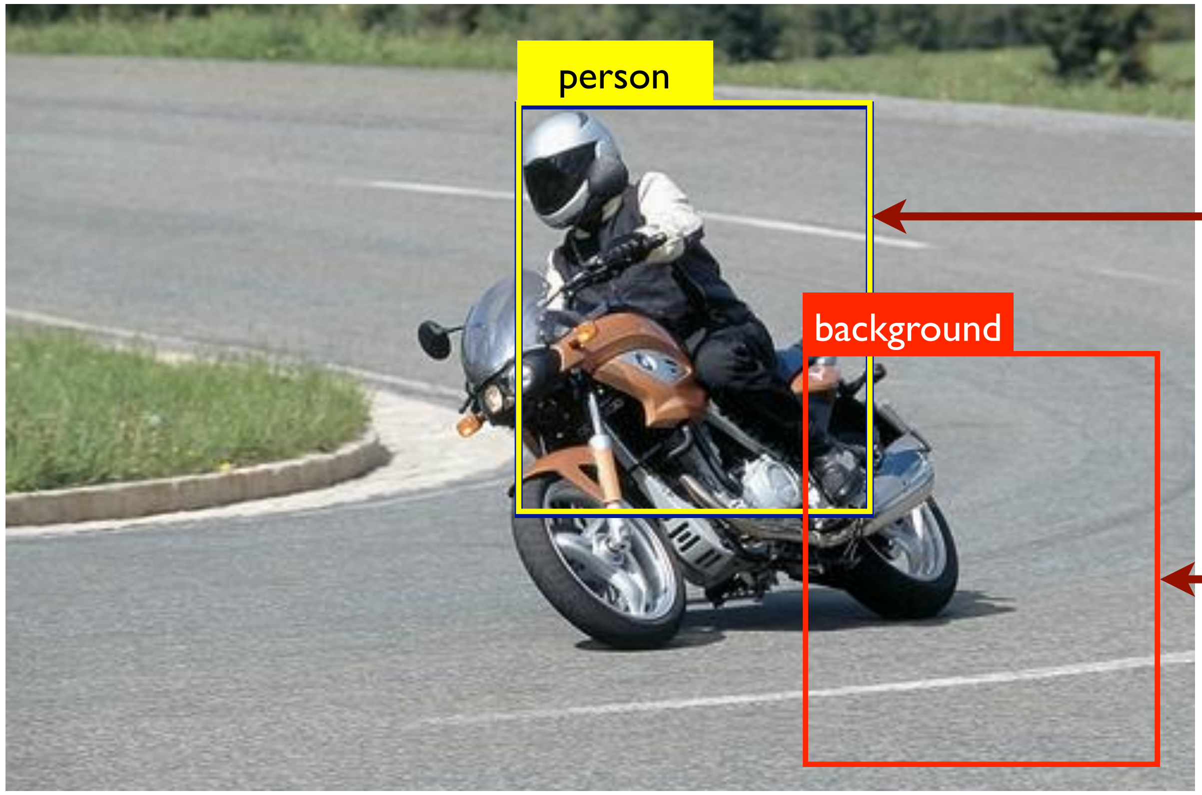


$$m_i(x, p_i) = w_i \cdot \Phi(x, p_i) \qquad d_{ij}(p_i, p_j) = w_{ij} \cdot \Phi(\delta_{ij})$$

$$\text{score}(x, p_1, \ldots, p_n) = \sum_{i=1}^{n} m_i(x, p_i) + \sum_{(i,j)\in E} d_{ij}(p_i, p_j) = w \cdot \Phi(x, \underbrace{p_1, \ldots, p_n}_{z})$$

Learning parameters for detection

$$\text{score}_w(x, z) = w \cdot \Phi(x, z)$$



Intuitive objectives:

some z should score high near the object

all z not near should score low

Training example (x,y)

x is an image

y is a label: +1 for foreground; -1 for background

Z(x) is a set of valid instantiations z

Recall the SVM objective

$$\min_{w} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \max[0, 1 - y_i w \cdot \Phi(x_i)]$$

- Fully supervised
- Goal: extend to handle latent variables
  - Latent SVM

Latent SVM (MI–SVM)

$$\min_{w} \frac{1}{2}||w||^2 + C \sum_{i=1}^{n} \max[0, 1 - y_i F_w(x_i)]$$

$$F_w(x) = \max_{z \in Z(x)} w \cdot \Phi(x, z)$$

- No longer convex
  - Why?
- “Semi-convexity” property
  - Non-convexity comes only from positive examples

[Andrews03, Felzenszwalb08]

Latent SVM (MI–SVM)

$$\min_{w} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \max[0, 1 - y_i F_w(x_i)]$$

$$F_w(x) = \max_{z \in Z(x)} w \cdot \Phi(x, z)$$

- Optimization (to a local minimum)
  - Coordinate descent
  - Convex-concave procedure CCCP