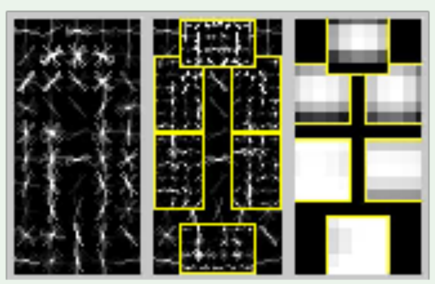
Deformable Part Models (DPM)

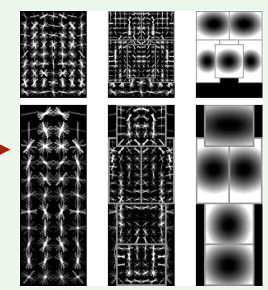
Felzenswalb, Girshick, McAllester & Ramanan (2010)

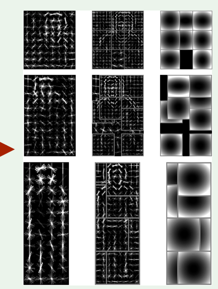
Slides drawn from a tutorial By R. Girshick

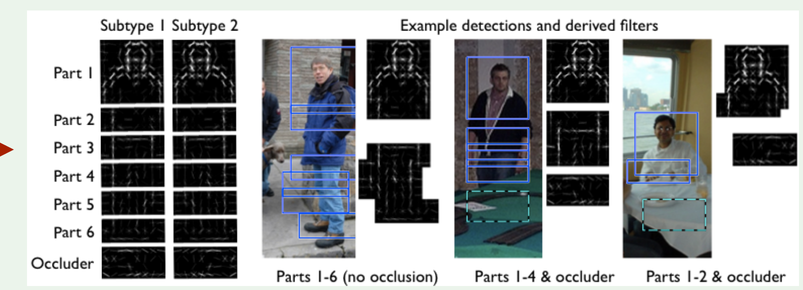
Part 1: modeling

Part 2: learning









AP	12%	27%	36%	45%	49%
	2005	2008	2009	2010	2011

Person detection performance on PASCAL VOC 2007

The Dalal & Triggs detector

Image pyramid

The Dalal & Triggs detector

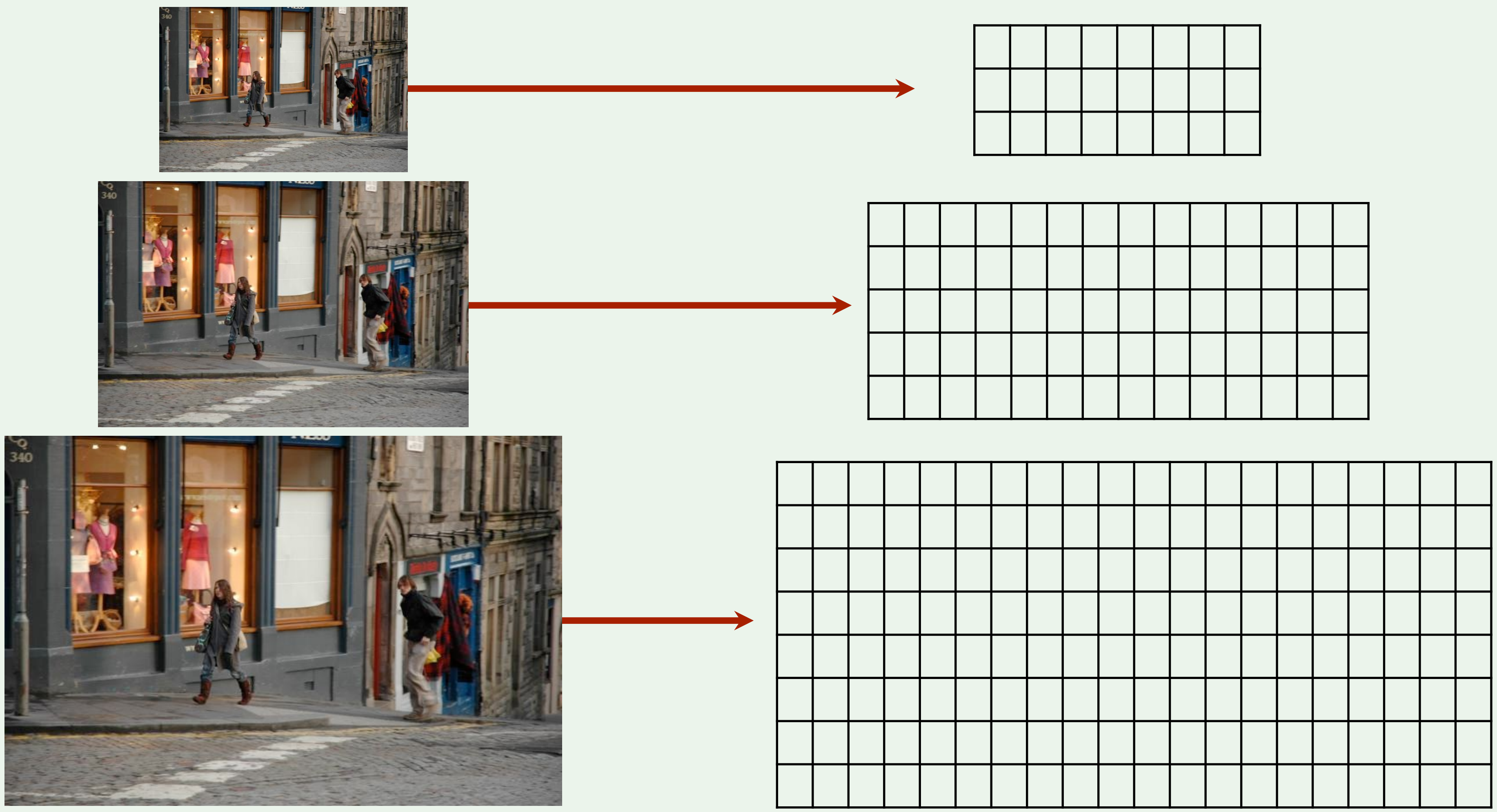
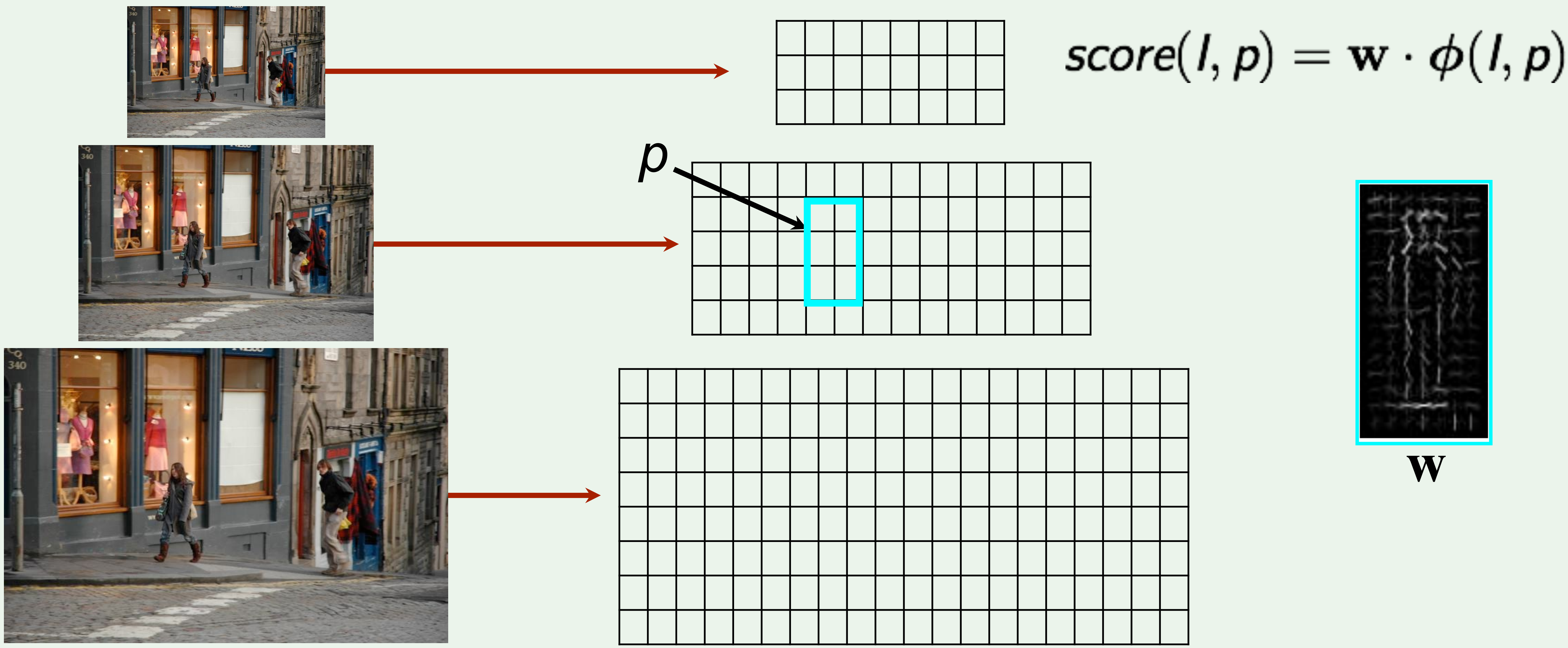


Image pyramid   HOG feature pyramid

- Compute HOG of the whole image at multiple resolutions

The Dalal & Triggs detector



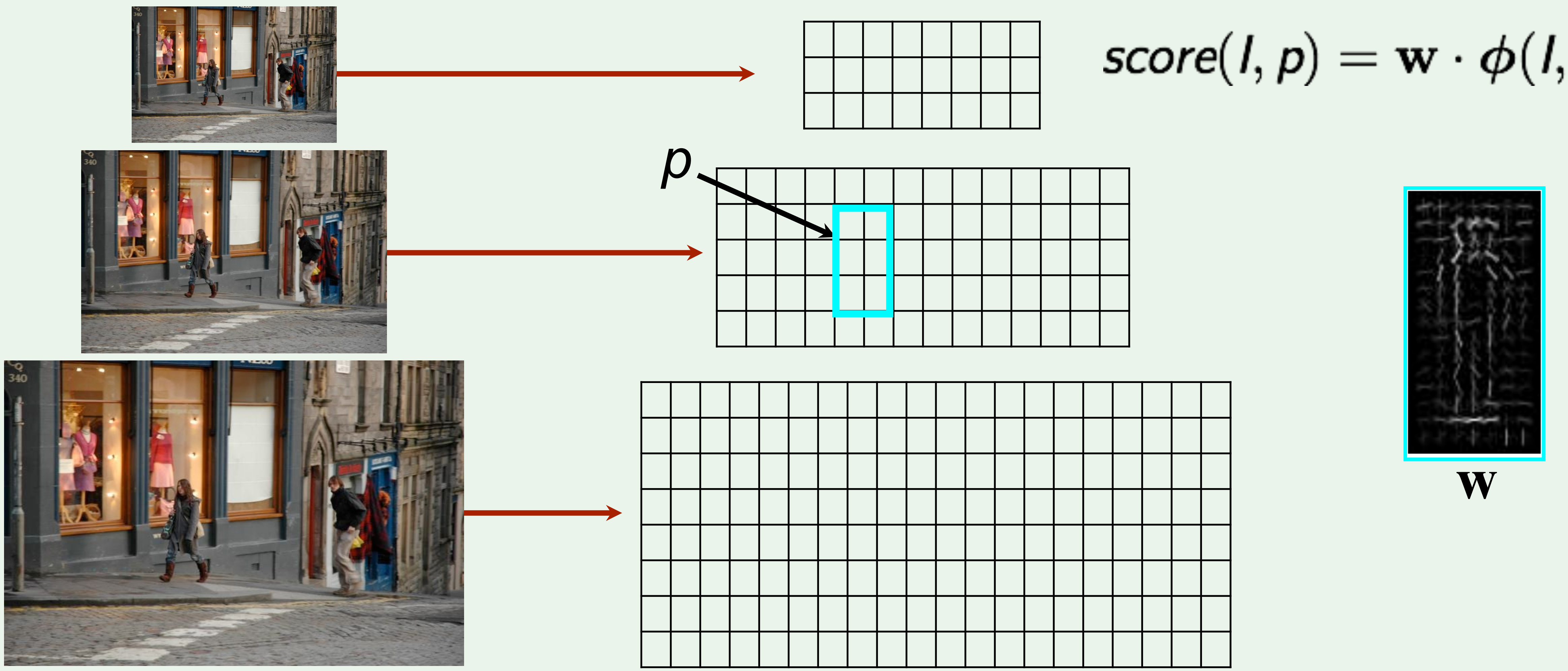
$$score(I, p) = \mathbf{w} \cdot \phi(I, p)$$

Image pyramid   HOG feature pyramid

- Compute HOG of the whole image at multiple resolutions
- Score every window of the feature pyramid

How much does the window at *p* look like a pedestrian?

The Dalal & Triggs detector



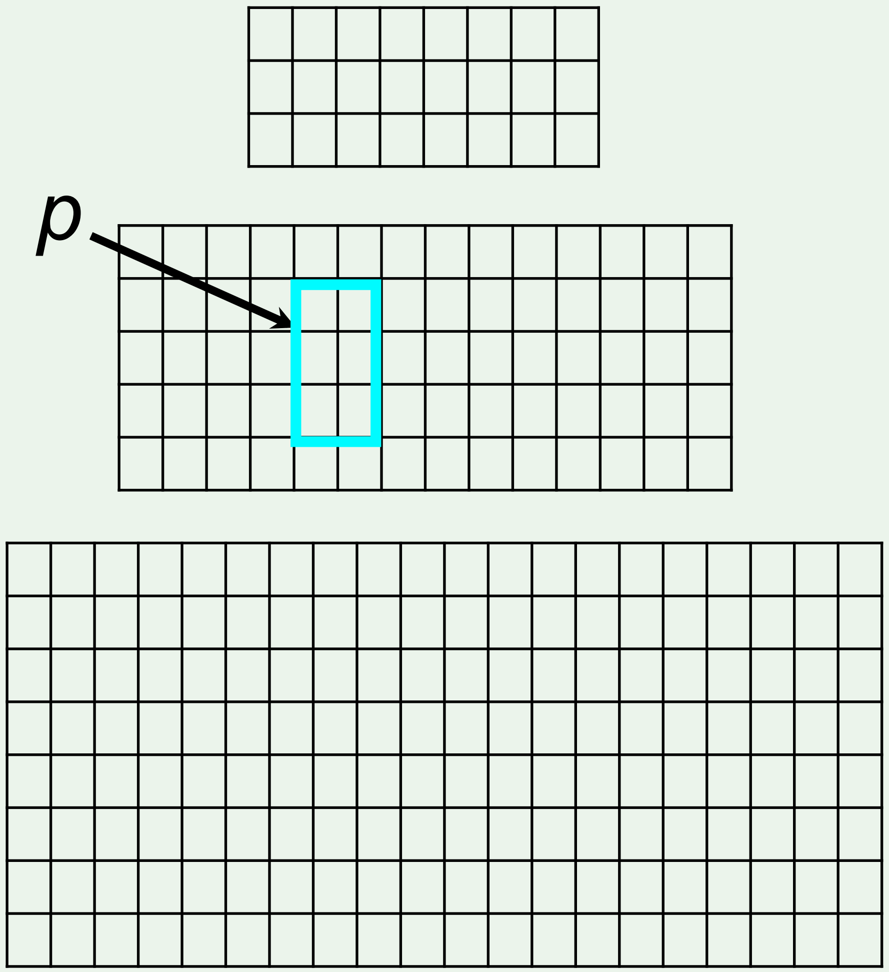
$$score(I, p) = \mathbf{w} \cdot \phi(I, p)$$

Image pyramid

HOG feature pyramid

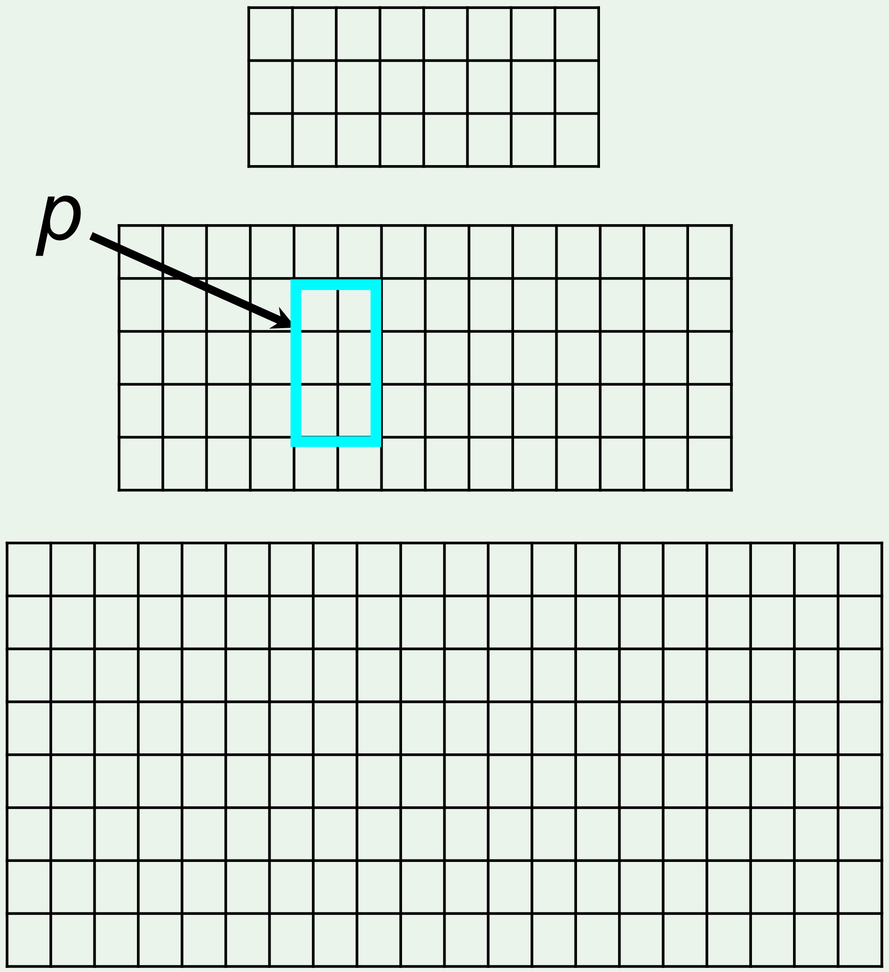
- Compute HOG of the whole image at multiple resolutions
- Score every window of the feature pyramid
- Apply non-maximal suppression

Detection



number of locations *p* ~ 250,000 per image

Detection

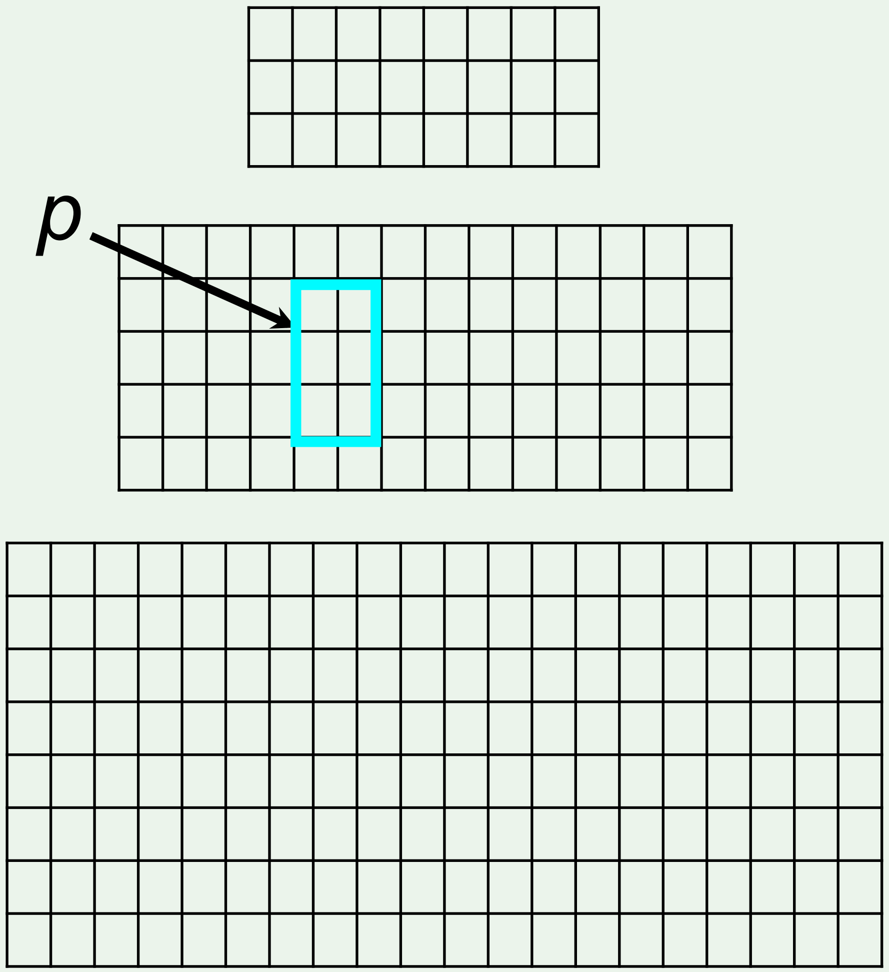


number of locations p ~ 250,000 per image

test set has ~ 5000 images

>> 1.3×10^9 windows to classify

Detection



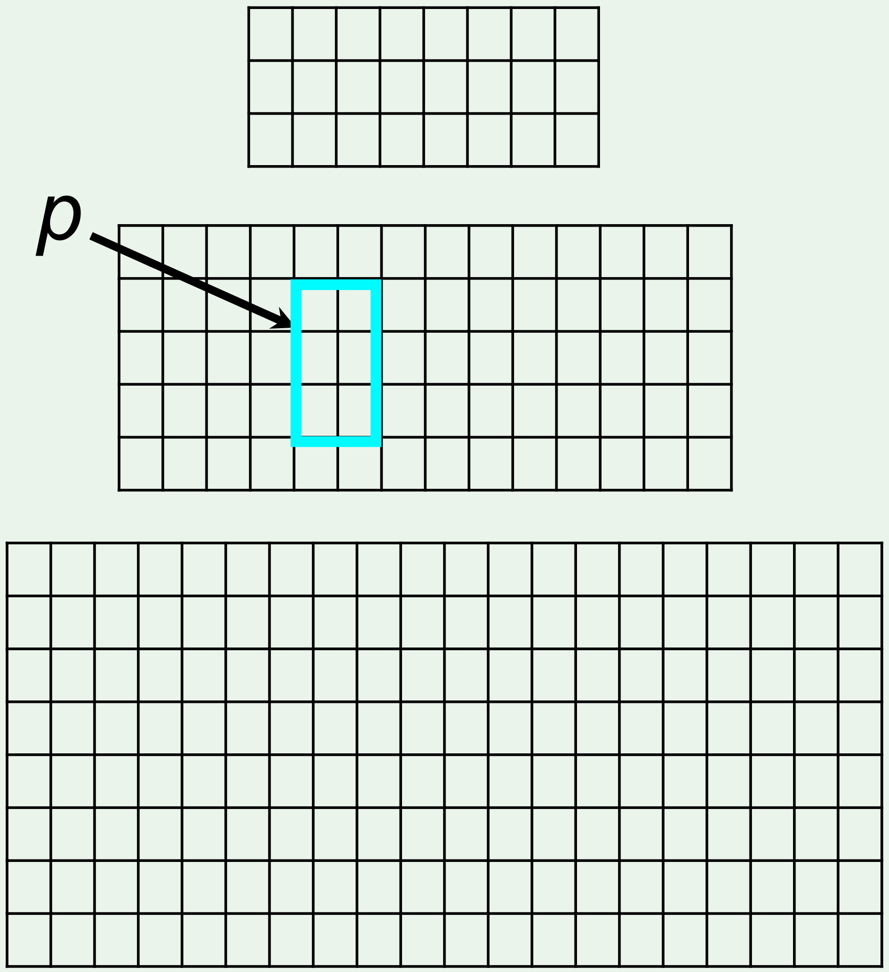
number of locations p ~ 250,000 per image

test set has ~ 5000 images

>> 1.3×10^9 windows to classify

typically only ~ 1,000 true positive locations

Detection



number of locations *p* ~ 250,000 per image

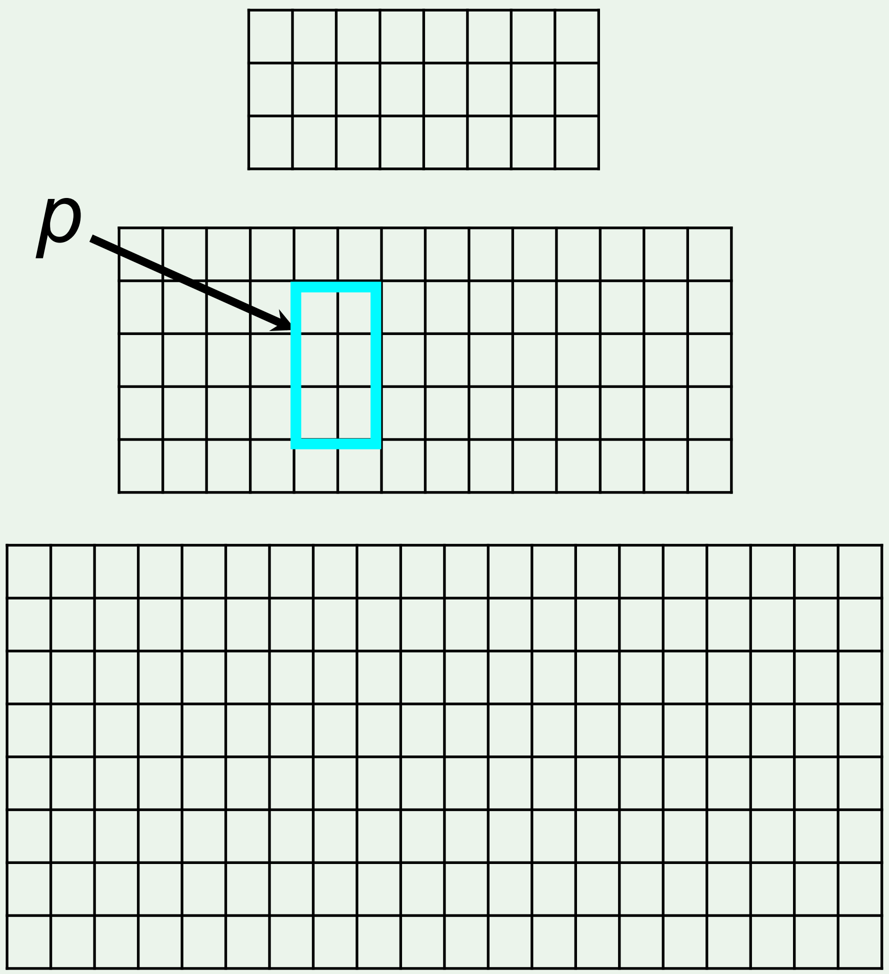
test set has ~ 5000 images

>> 1.3×10^9 windows to classify

typically only ~ 1,000 true positive locations

Extremely unbalanced binary classification

Detection



number of locations p ~ 250,000 per image

test set has ~ 5000 images

>> $1.3\text{x}10^9$ windows to classify

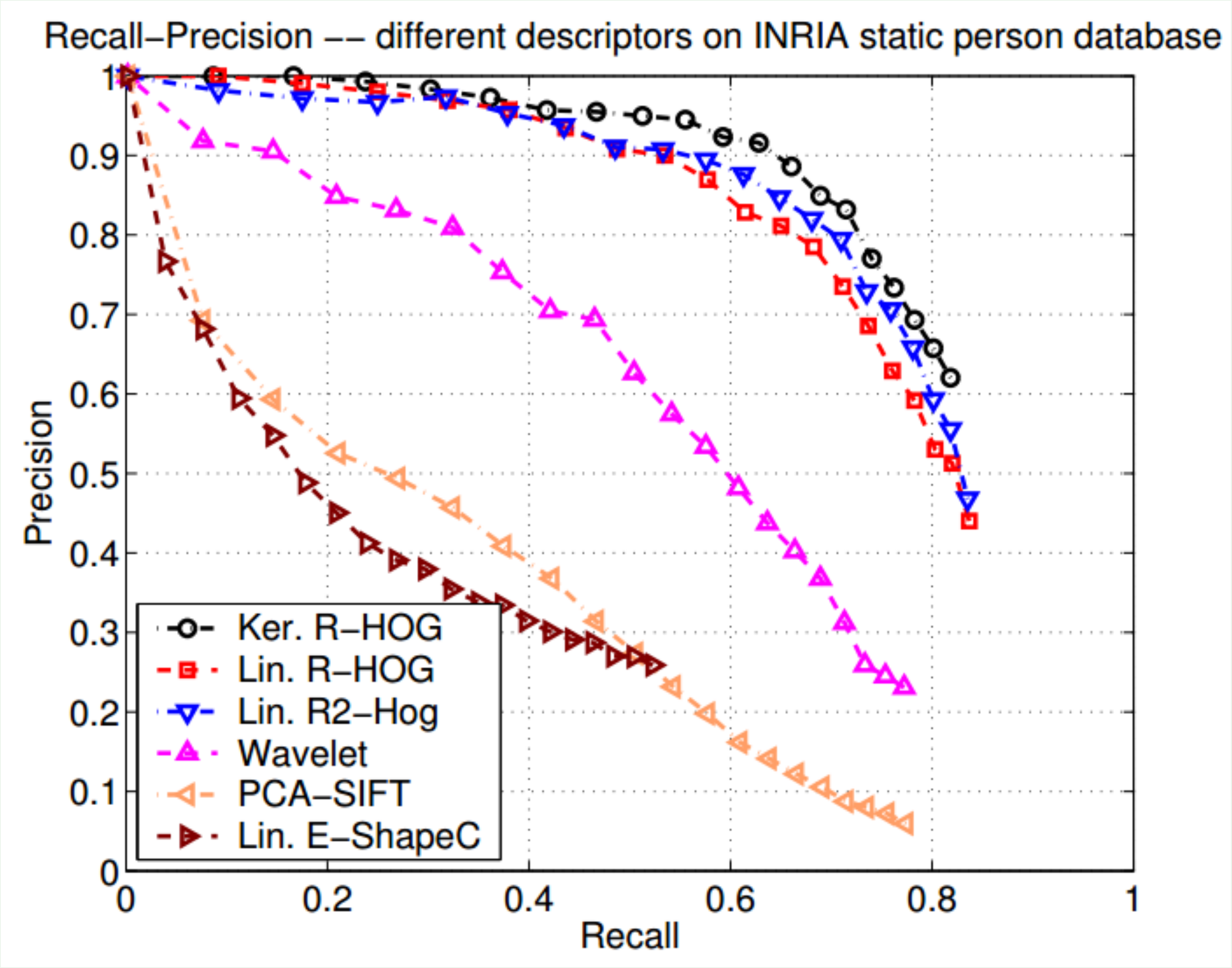
typically only ~ 1,000 true positive locations

w

Learn **w** as a Support Vector Machine (SVM)

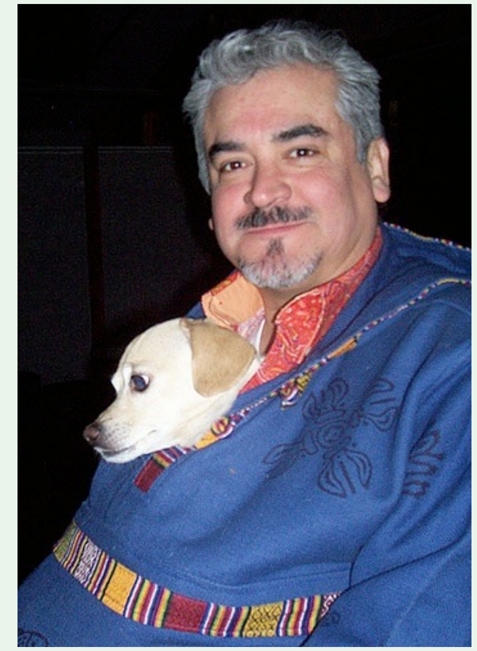
Extremely unbalanced binary classification

Dalal & Triggs detector on INRIA pedestrians



- AP = 75%
  - Very good
  - **Declare victory and go home?**

Dalal & Triggs on PASCAL VOC 2007

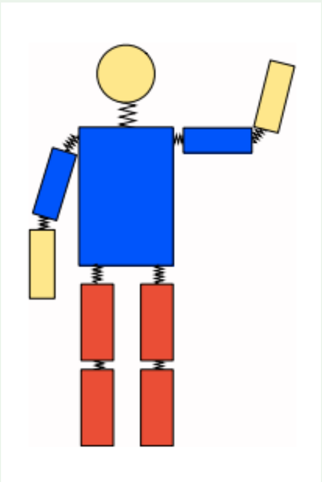


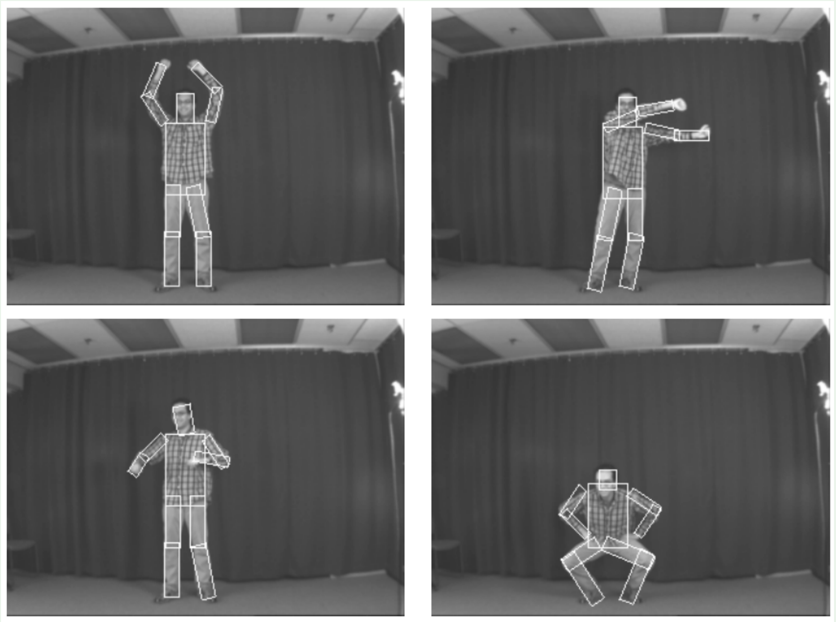
AP = 12%

(using my implementation)

How can we do better?

Revisit an old idea: part-based models
"pictorial structures"



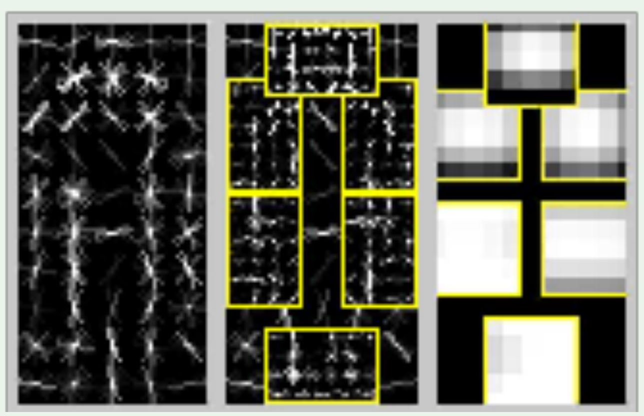


Fischler & Elschlager '73
Felzenszwalb & Huttenlocher '00
- Pictorial structures
- *Weak appearance models*
- *Non-discriminative training*

Combine with modern features and machine learning

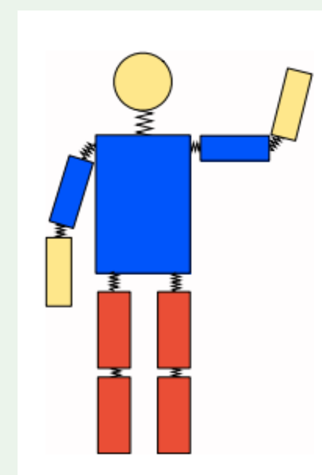
DPM key idea

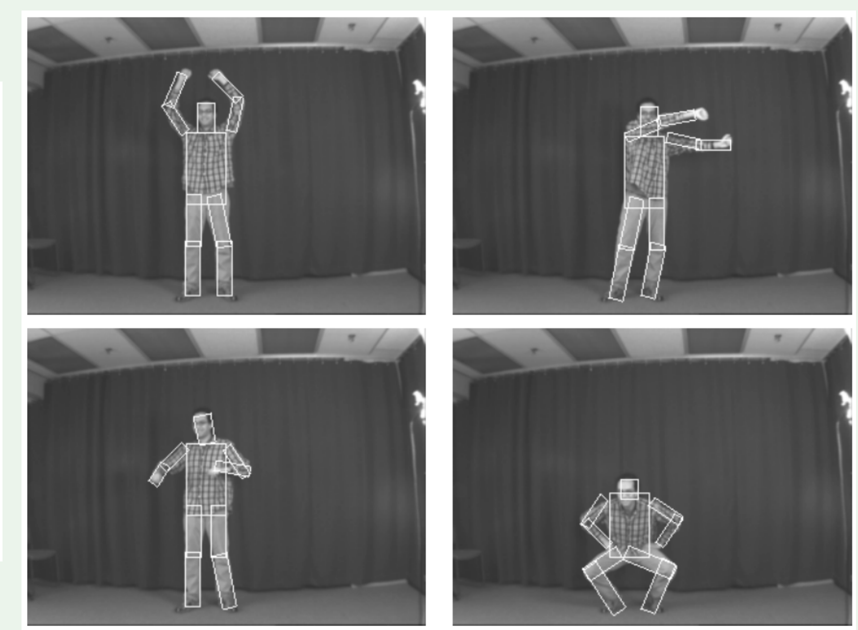
Port the success of Dalal & Triggs
into a part-based model



=

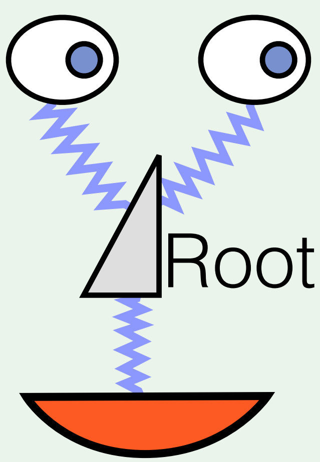
+

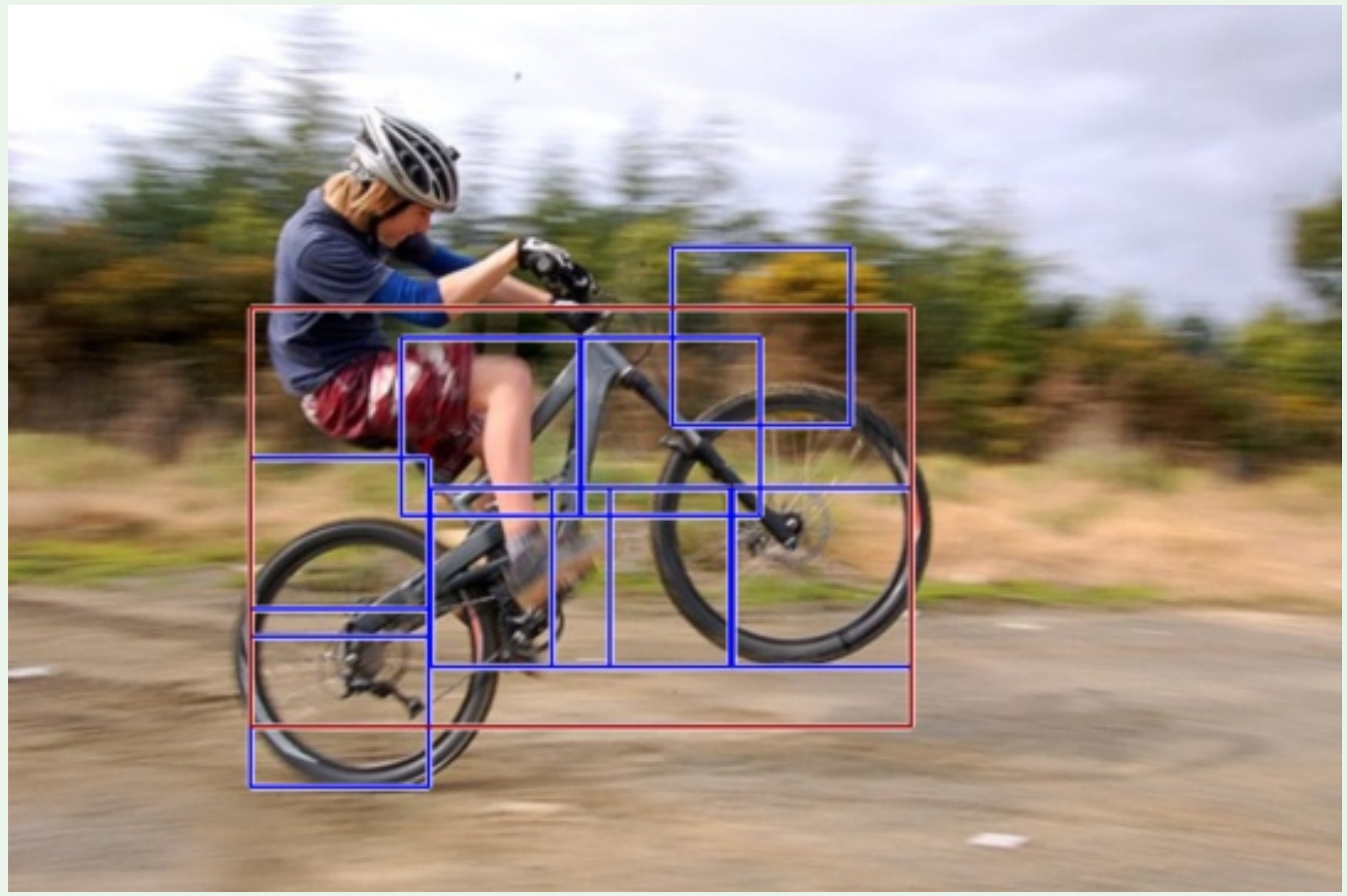


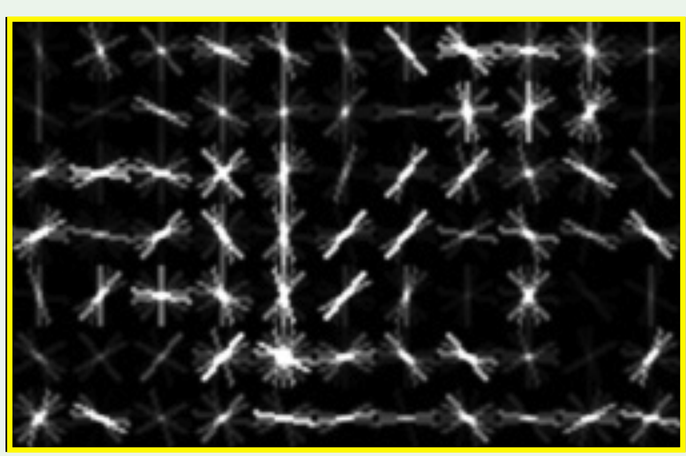


DPM D&T PS

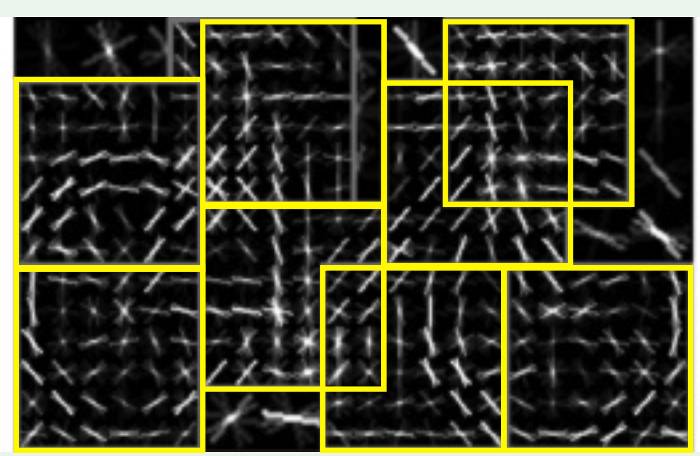
Example DPM (most basic version)



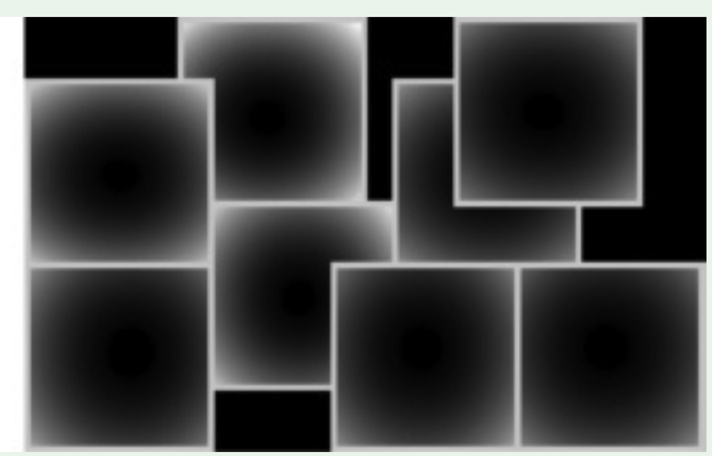




Root filter



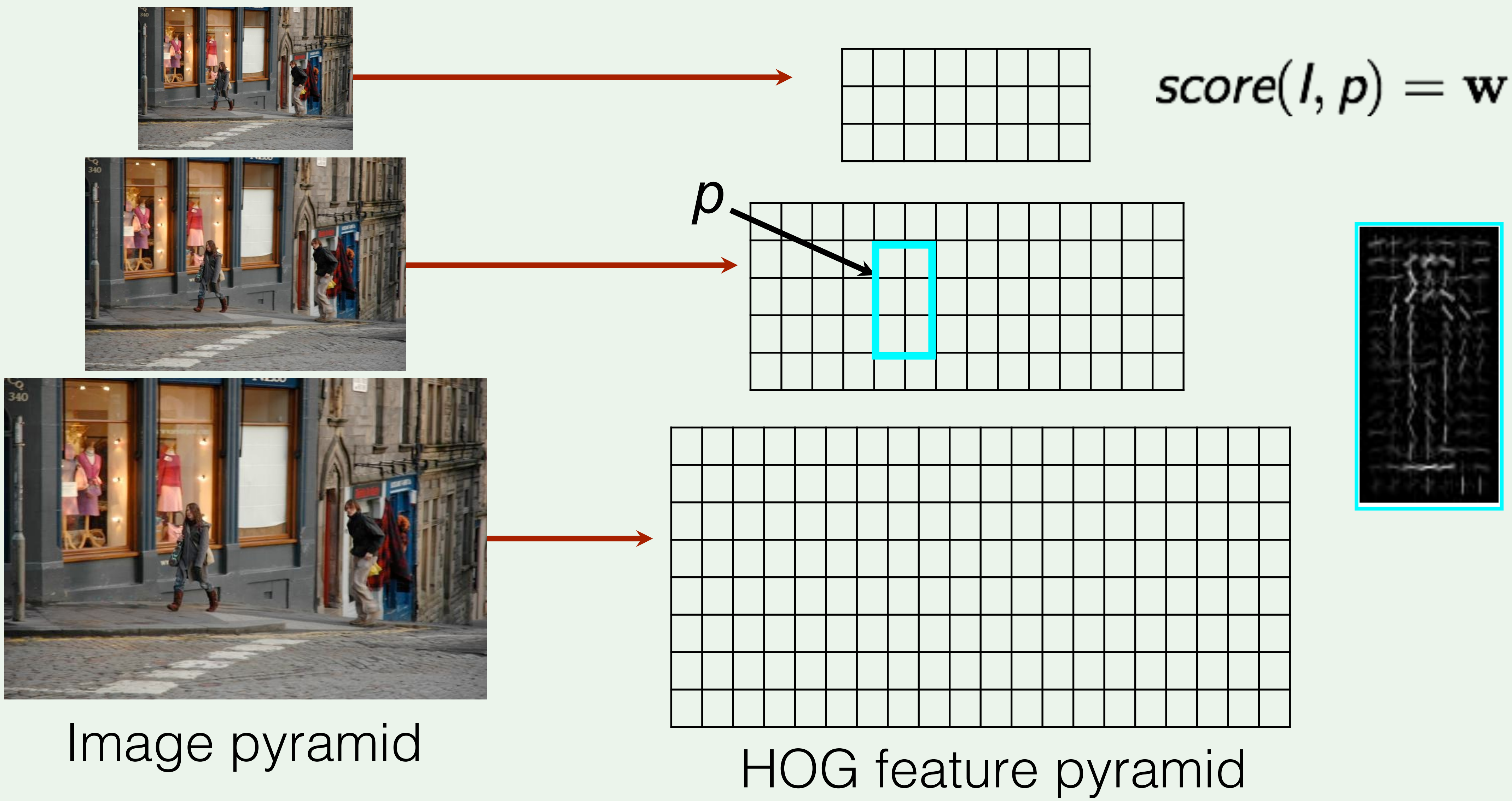
Part filters



Deformation costs

Recall the Dalal & Triggs detector

$$score(I, p) = \mathbf{w} \cdot \phi(I, p)$$



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

DPM = D&T + parts

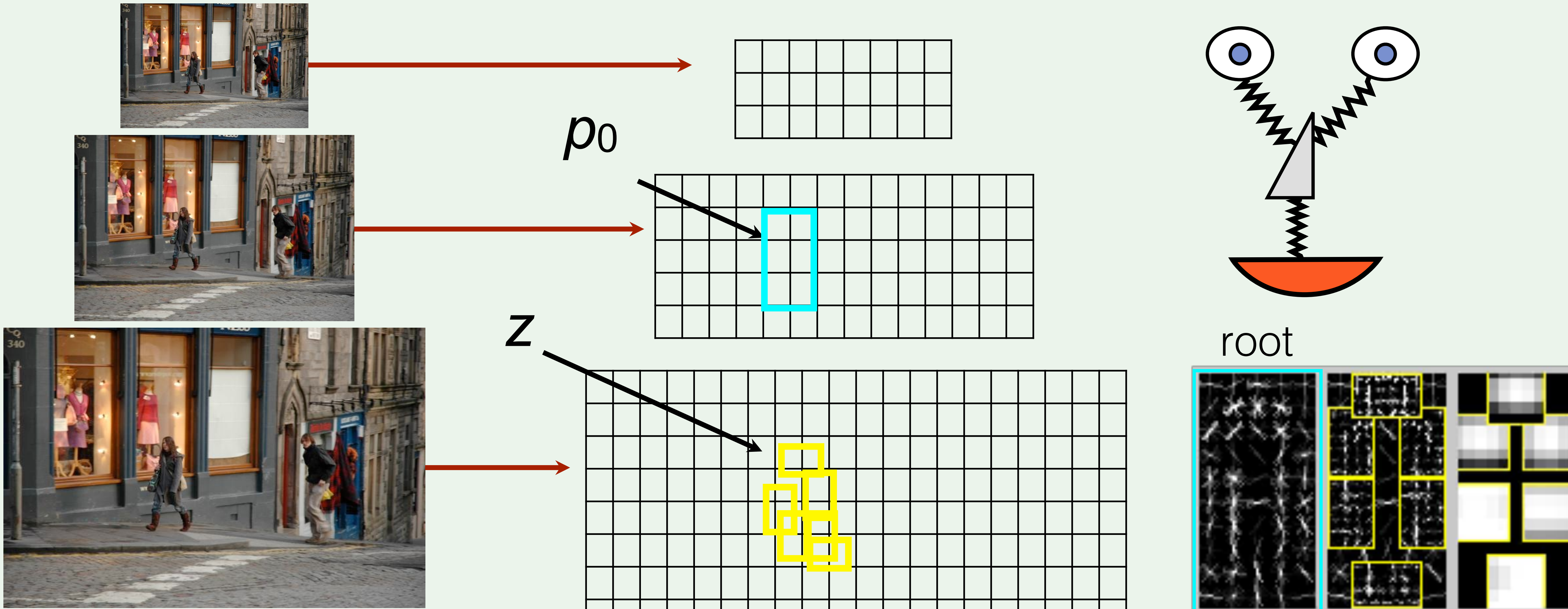


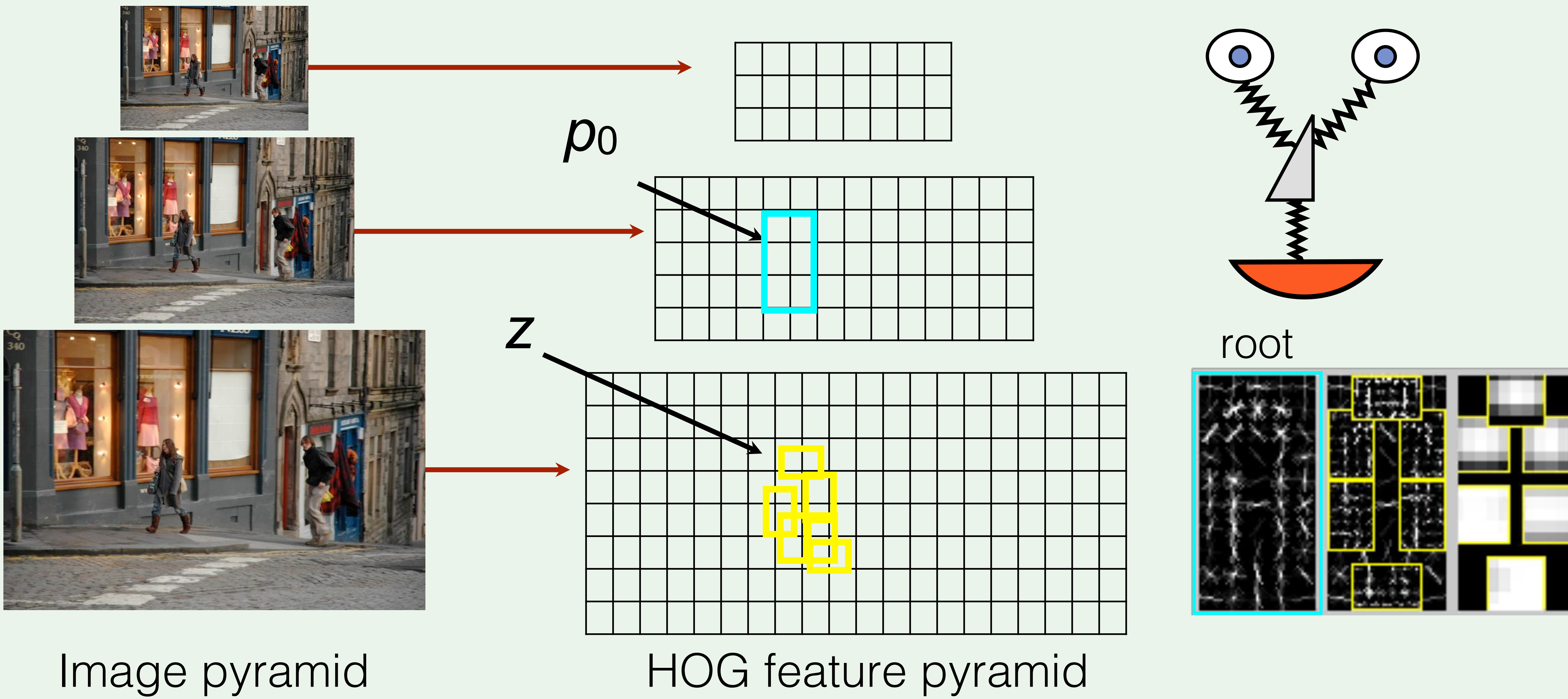
Image pyramid

HOG feature pyramid

[FMR CVPR'08]
[FGMR PAMI'10]

- Add parts to the Dalal & Triggs detector
  - HOG features
  - Linear filters / sliding-window detector
  - Discriminative training

Sliding window detection with DPM



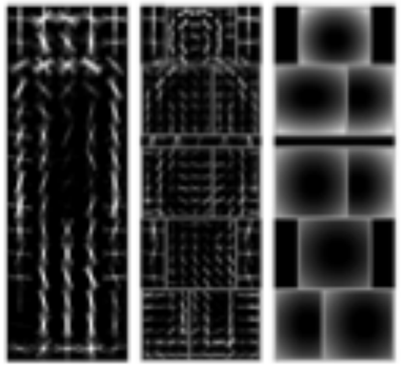
$$z = (p_1, \ldots, p_n)$$

$$\text{score}(I, p_0) = \max_{p_1, \ldots, p_n} \sum_{i=0}^{n} m_i(I, p_i) - \sum_{i=1}^{n} d_i(p_0, p_i)$$

Filter scores — Spring costs

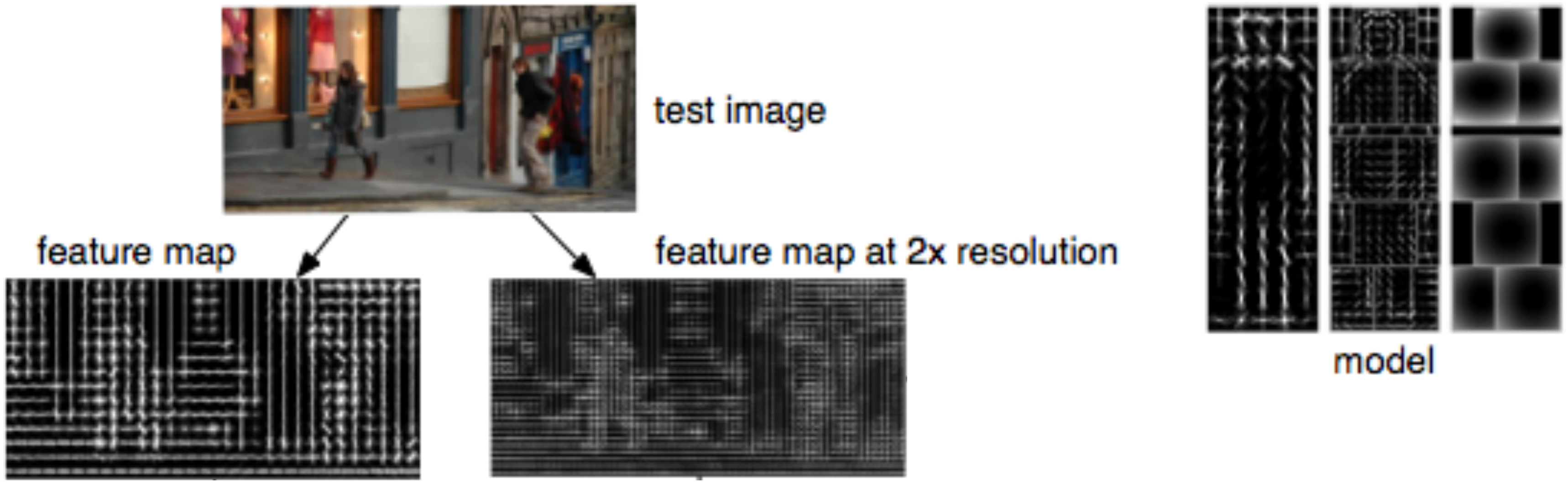
DPM detection

test image



model

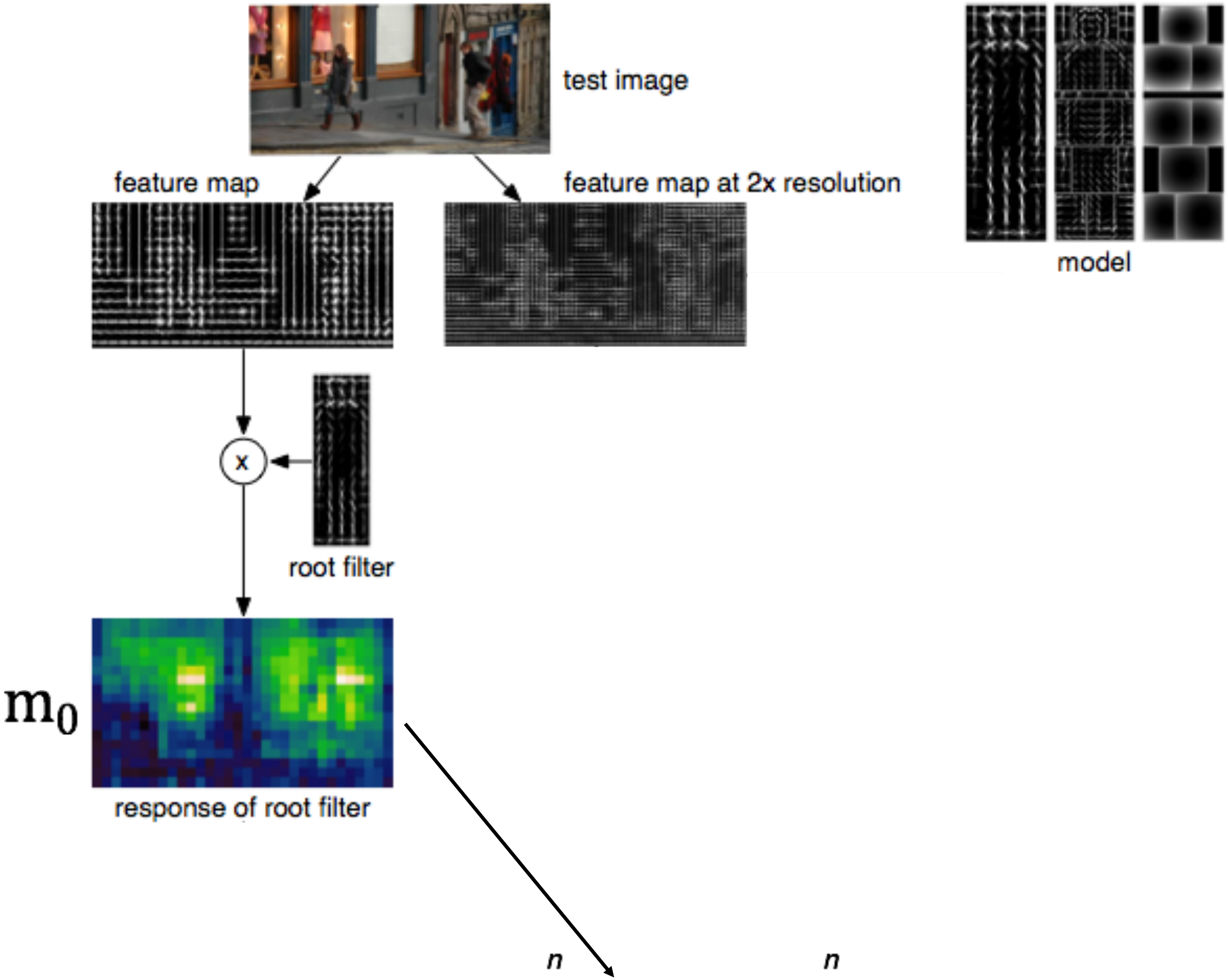
DPM detection



Root scale　　Part scale

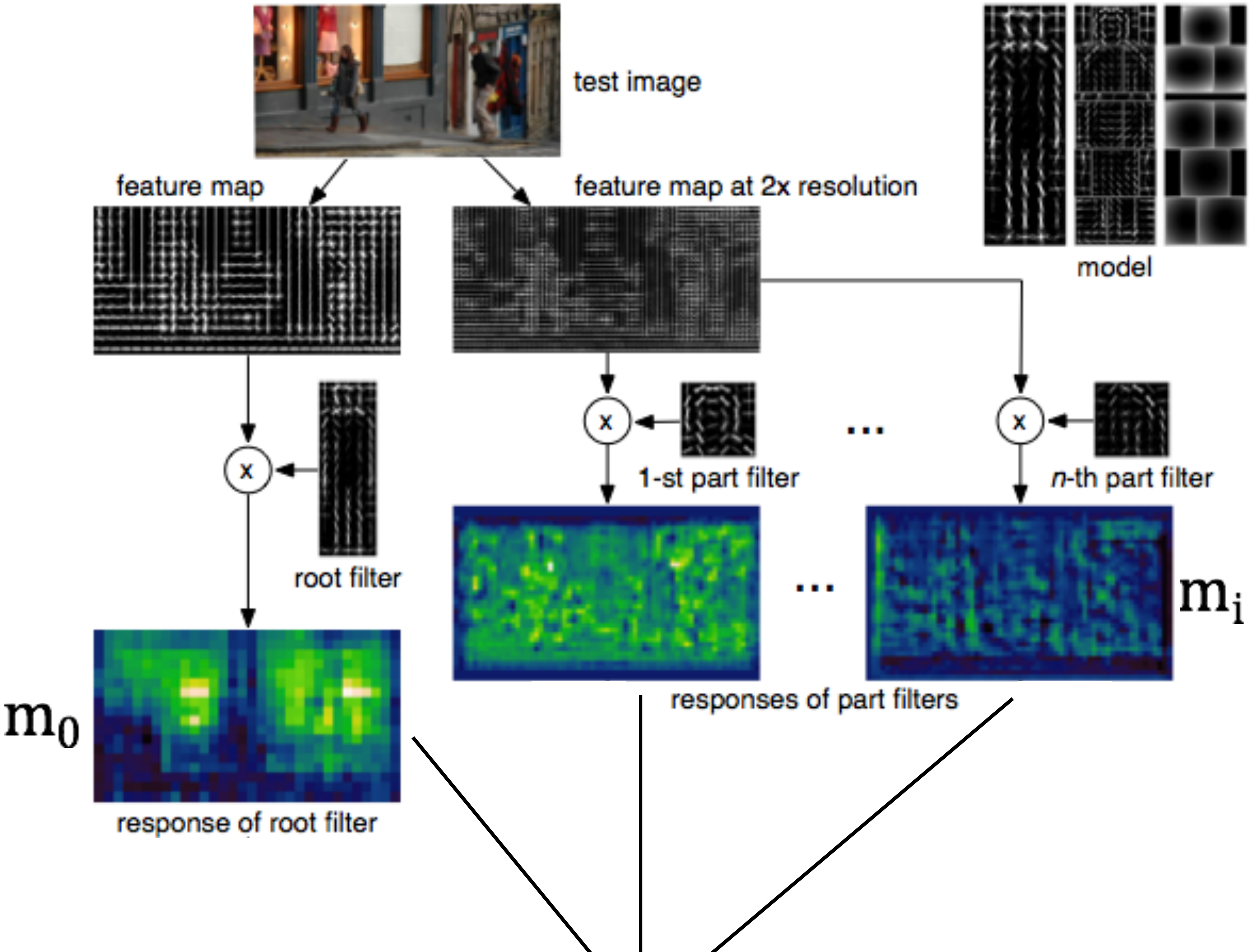
repeat for each level in pyramid

DPM detection



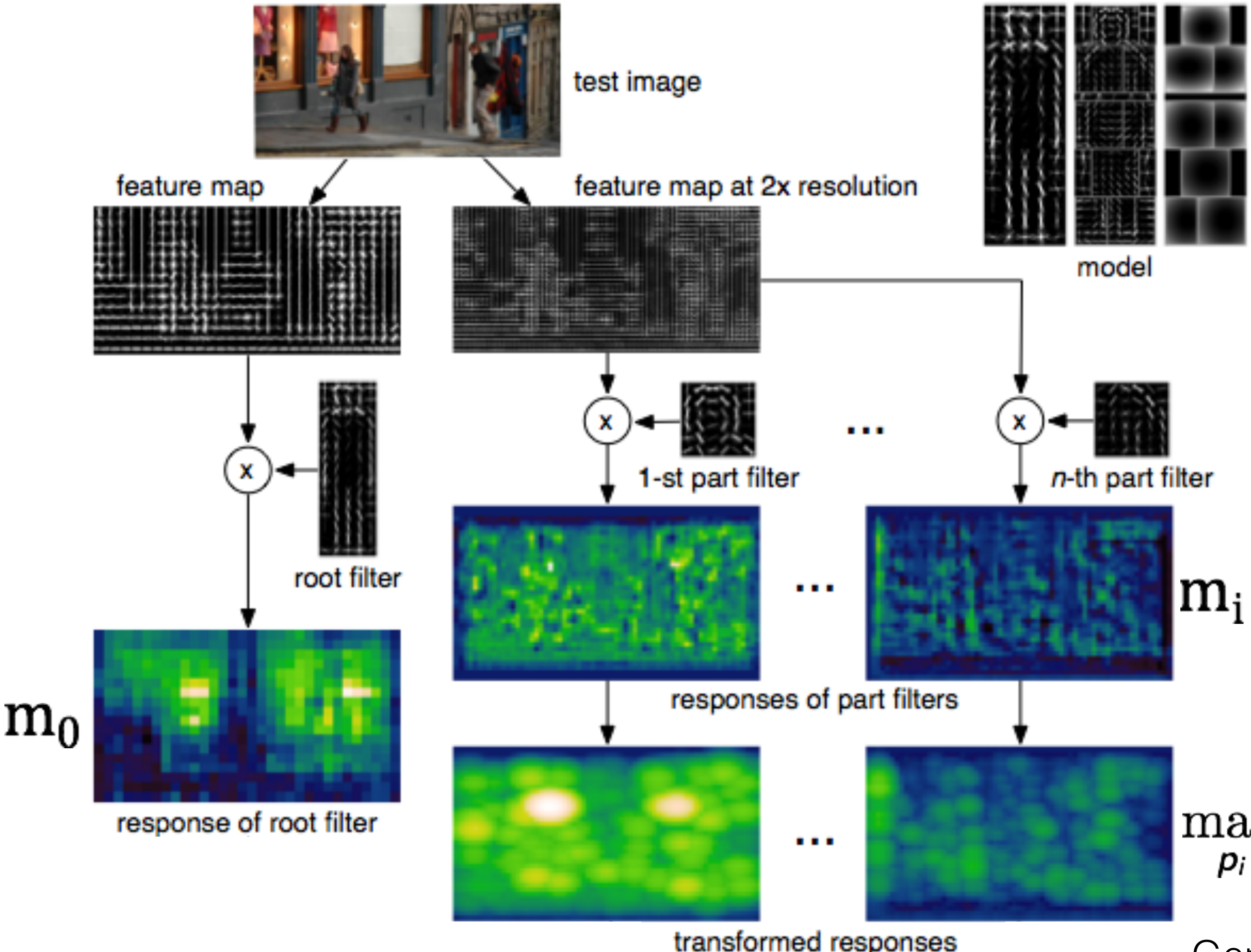
$$\text{score}(I, p_0) = \max_{p_1,\ldots,p_n} \sum_{i=0}^{n} m_i(I, p_i) - \sum_{i=1}^{n} d_i(p_0, p_i)$$

DPM detection



$$\text{score}(I, p_0) = \max_{p_1,\ldots,p_n} \sum_{i=0}^{n} m_i(I, p_i) - \sum_{i=1}^{n} d_i(p_0, p_i)$$

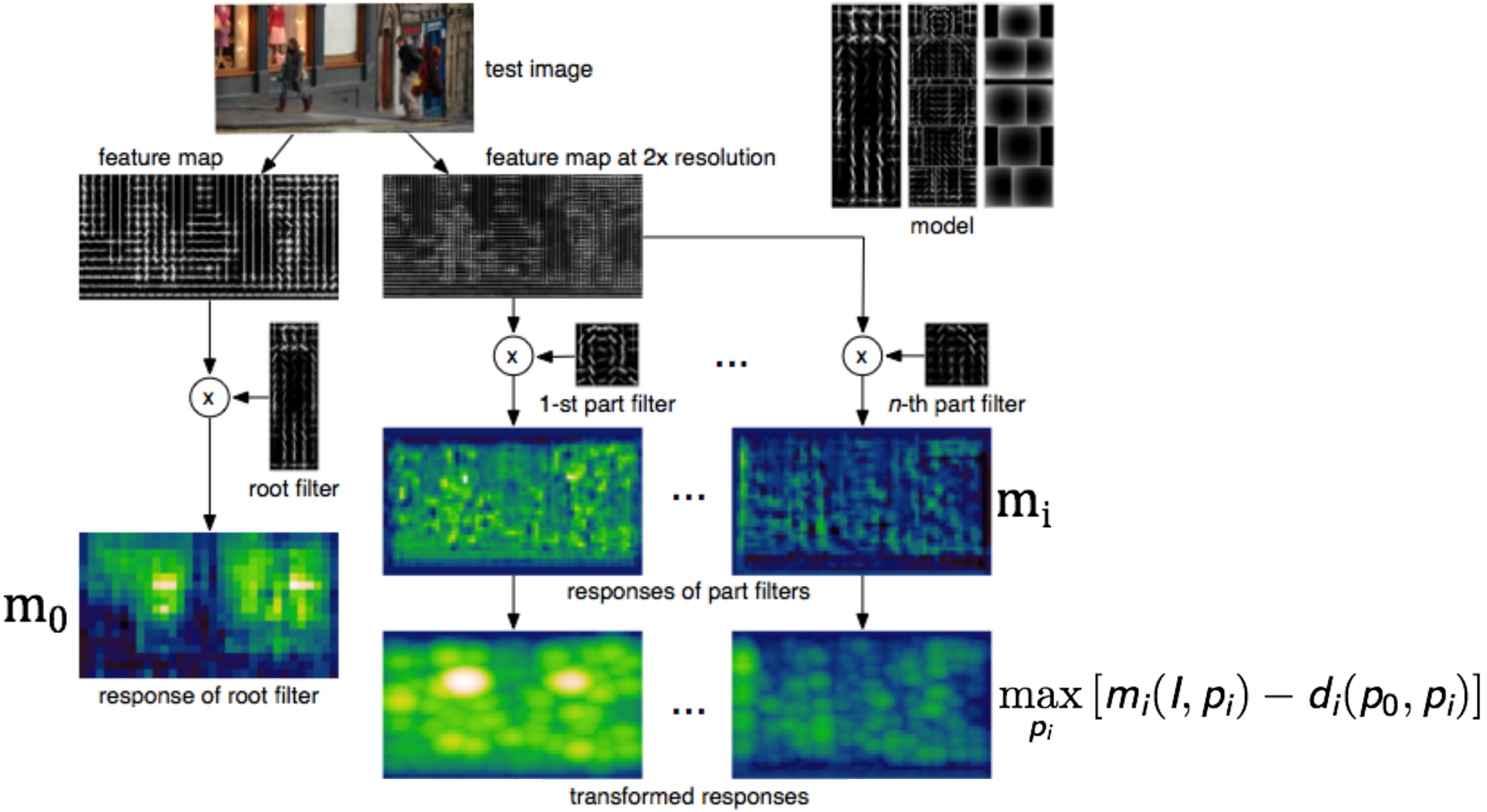
DPM detection



$$\max_{p_i} \left[m_i(I, p_i) - d_i(p_0, p_i) \right]$$

Generalized distance transform
Felzenszwalb & Huttenlocher '00

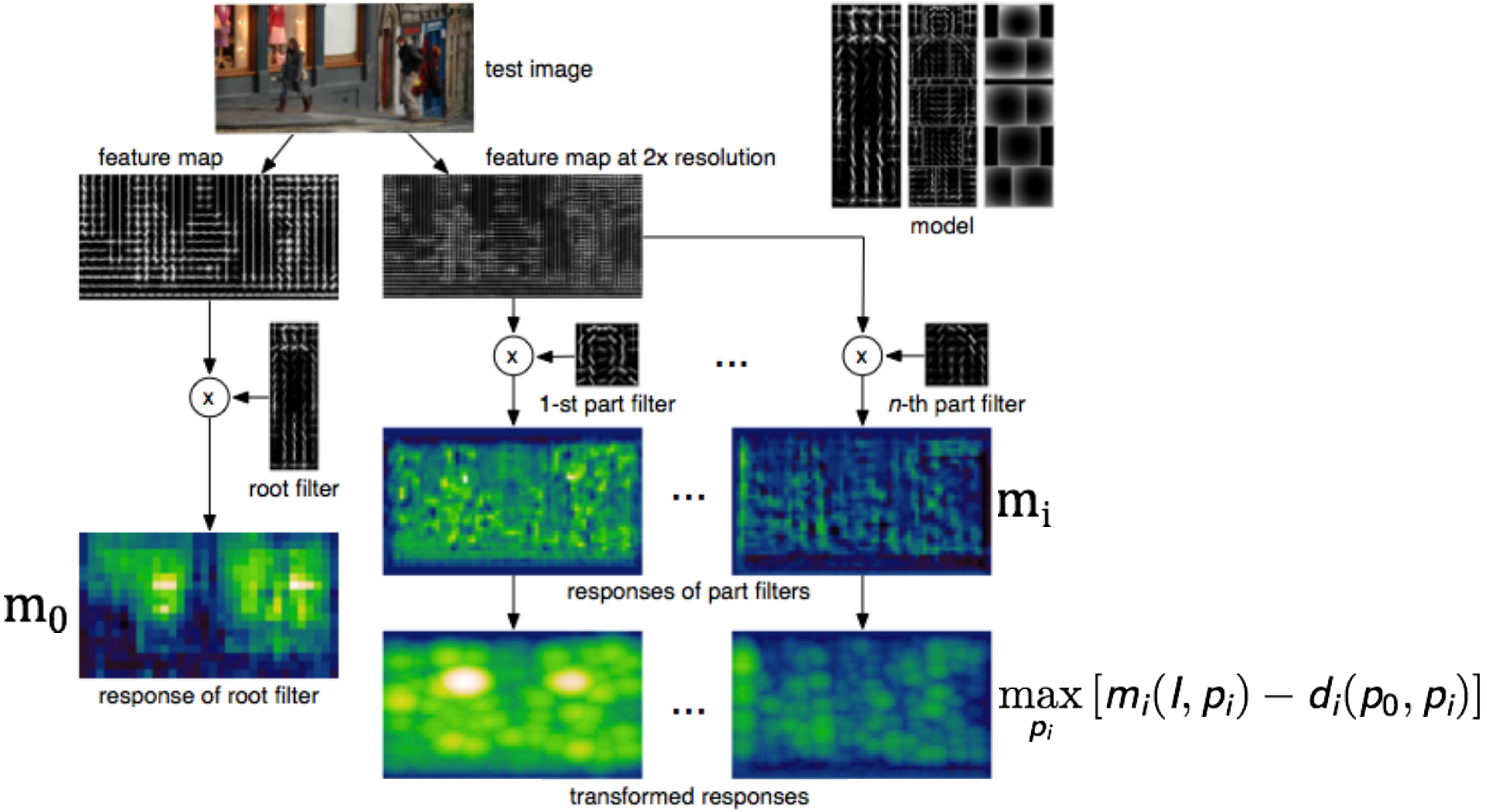
DPM detection



$$\mathrm{score}(I, p_0) = \max_{p_1,\ldots,p_n} \sum_{i=0}^{n} m_i(I, p_i) - \sum_{i=1}^{n} d_i(p_0, p_i)$$

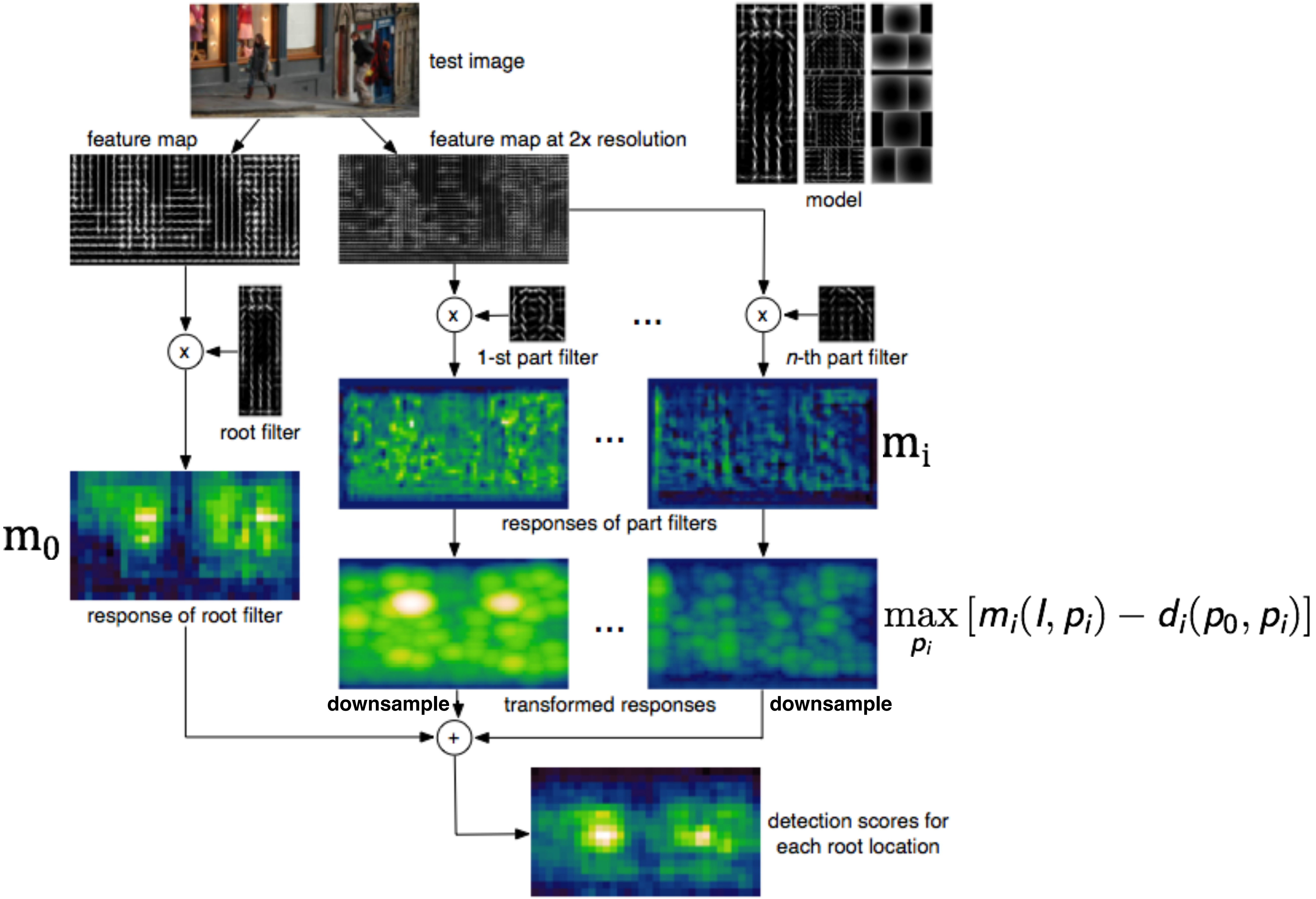
$$= m_0(I, p_0) + \sum_{i=1}^{n} \max_{p_i} [m_i(I, p_i) - d_i(p_0, p_i)]$$

DPM detection



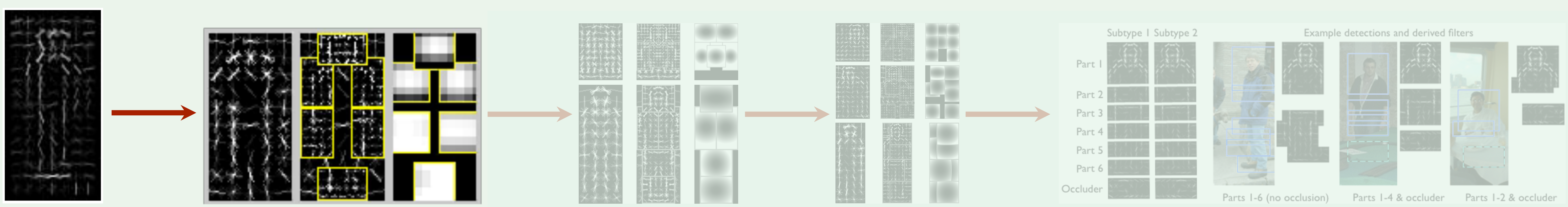
All that's left: combine evidence

DPM detection



Person detection progress

Progress bar:

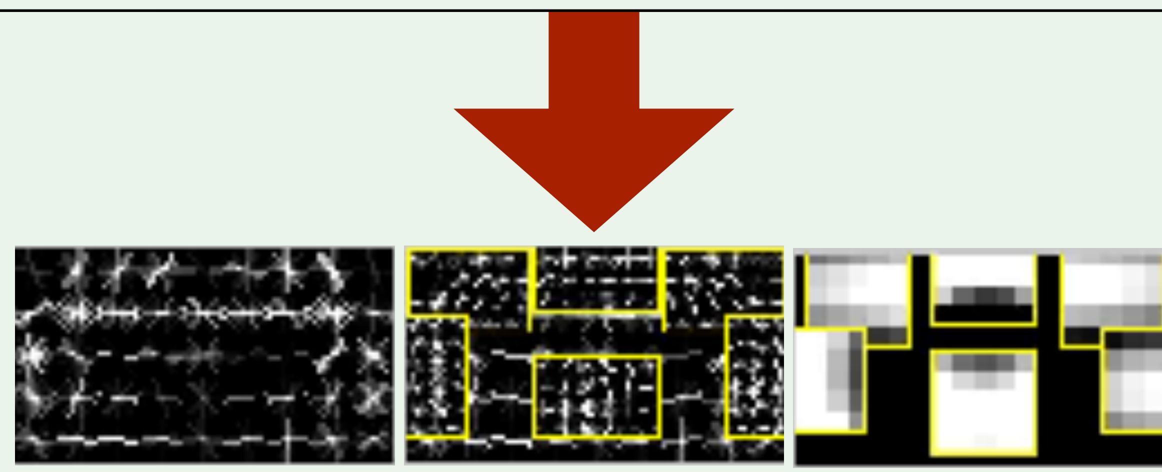


AP 12% 27% 36% 45% 49%

2005 2008 2009 2010 2011

One DPM is not enough: **What are the parts?**

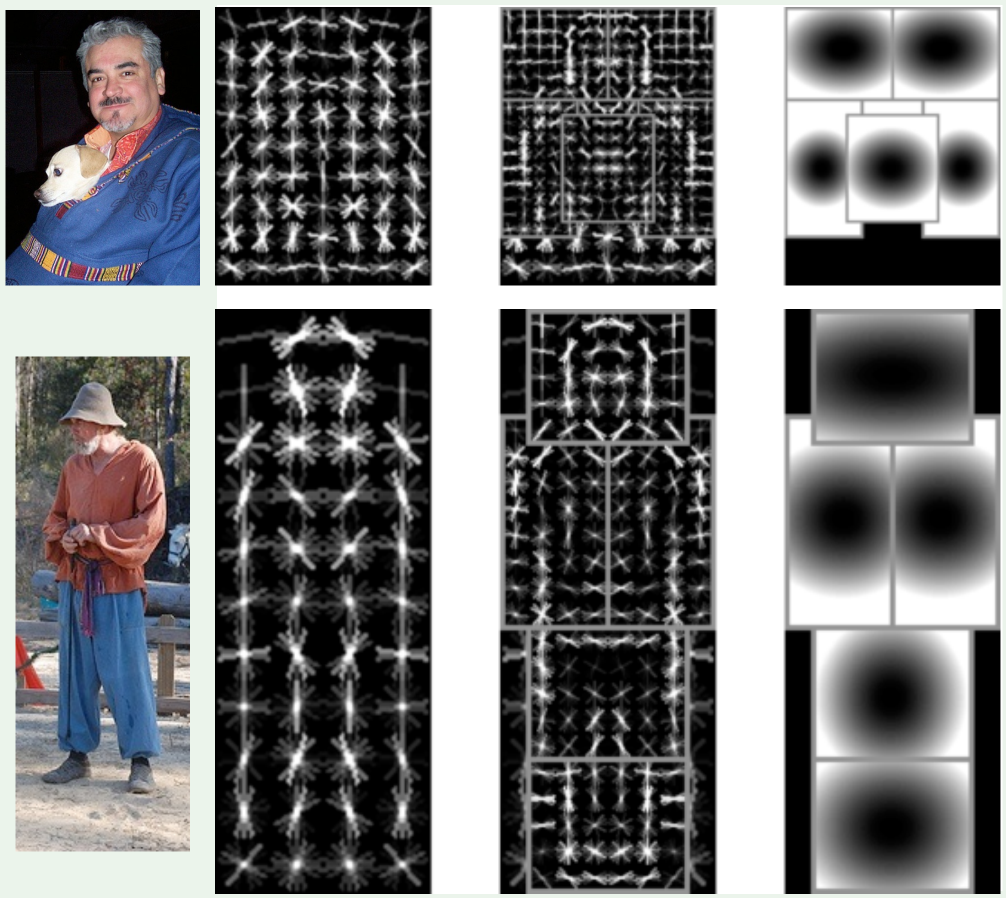
Aspect soup

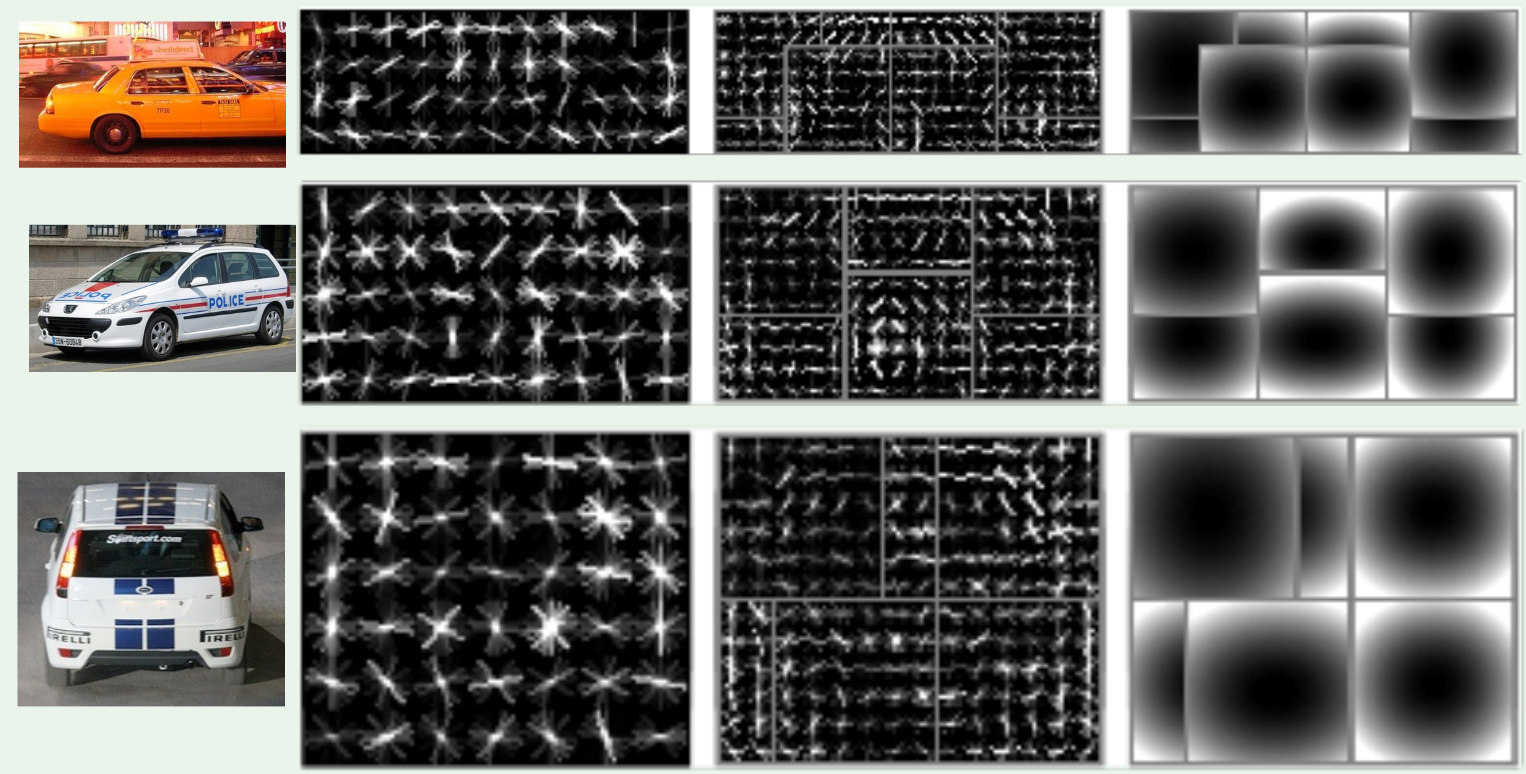


General philosophy: enrich models to better represent the data

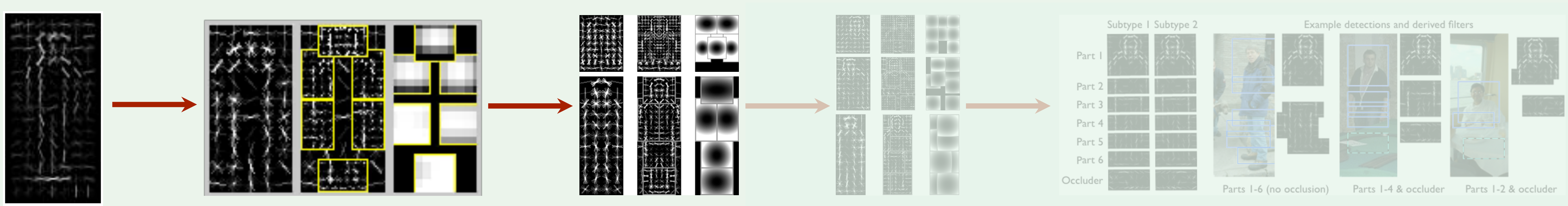
Mixture models

Data driven: aspect, occlusion modes, subclasses





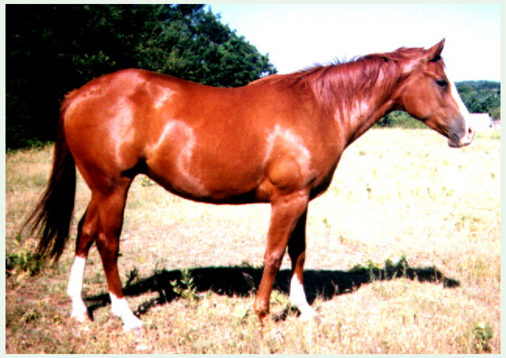
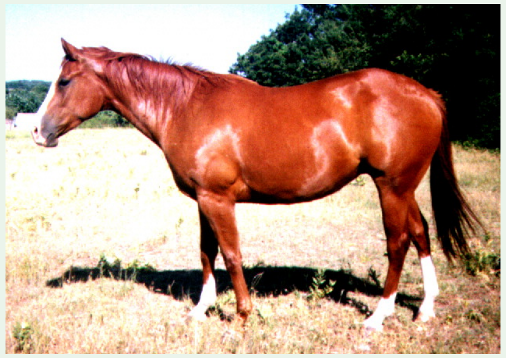
Progress bar:

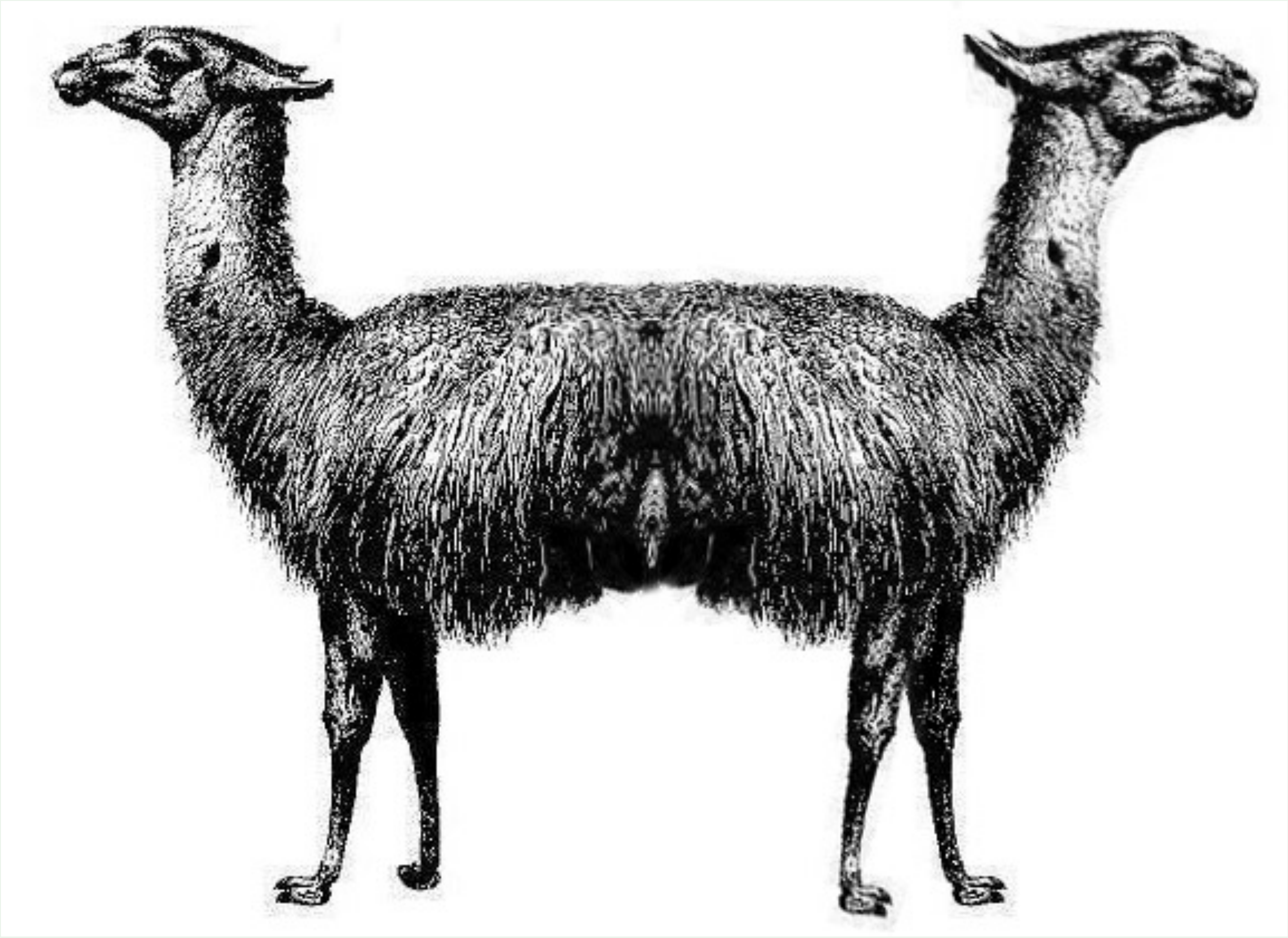


AP	12%	27%	36%	45%	49%
	2005	2008	2009	2010	2011

Pushmi–pullyu?

Good generalization properties on Doctor Dolittle's farm

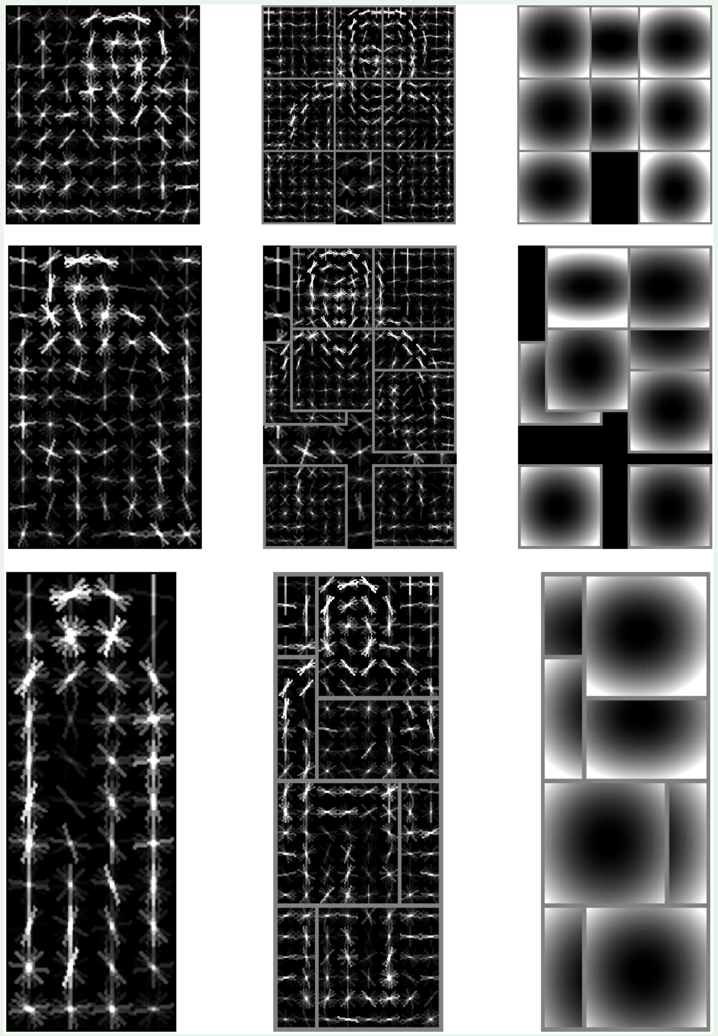
( + ) / 2 =



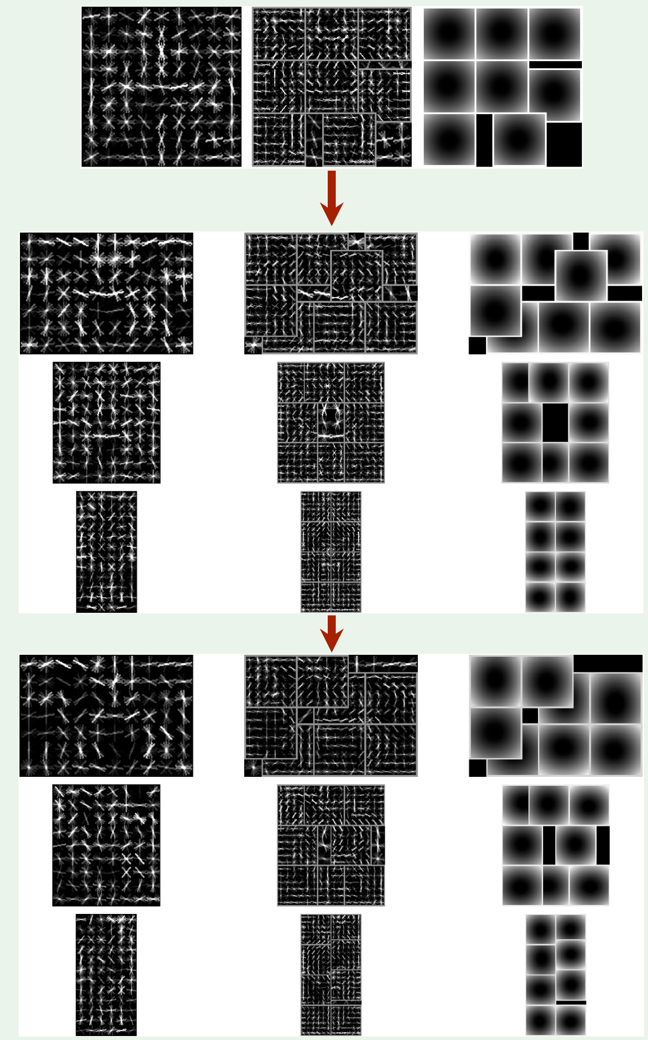
This was supposed to detect horses

Latent orientation

Unsupervised left/right orientation discovery



horse AP

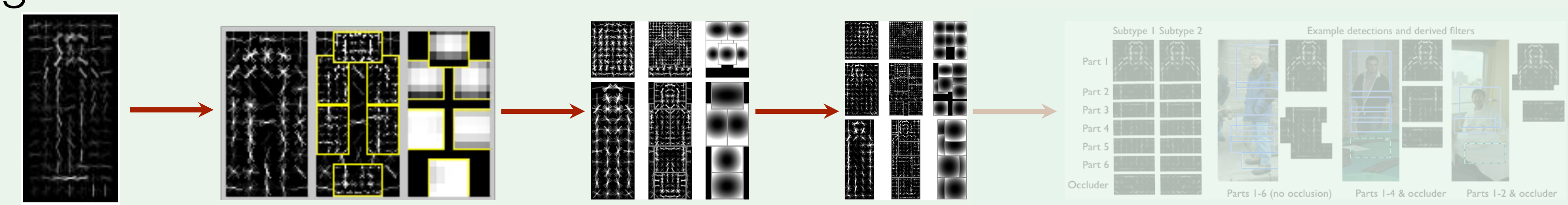


0.42

0.47

0.57

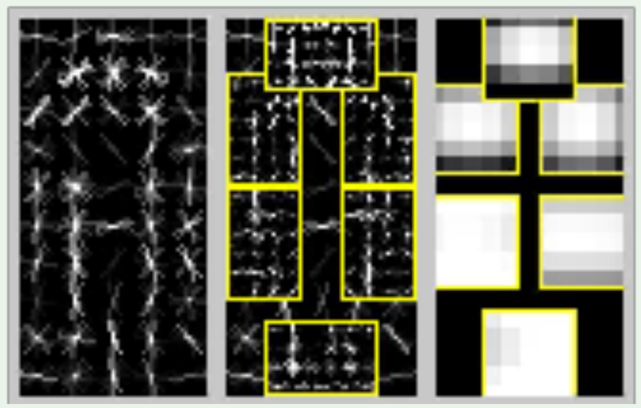
Progress bar:



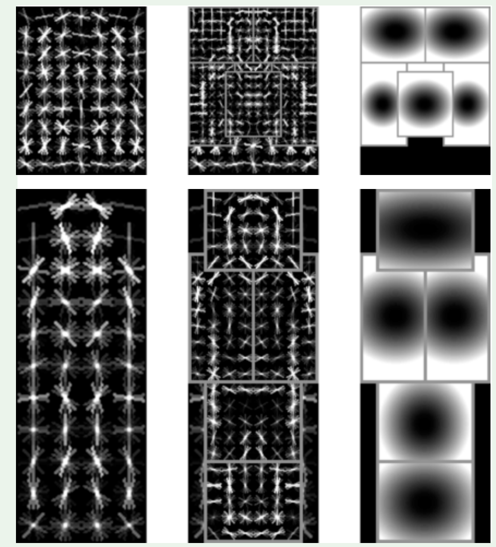
AP	12%	27%	36%	45%	49%
	2005	2008	2009	2010	2011

Summary of results

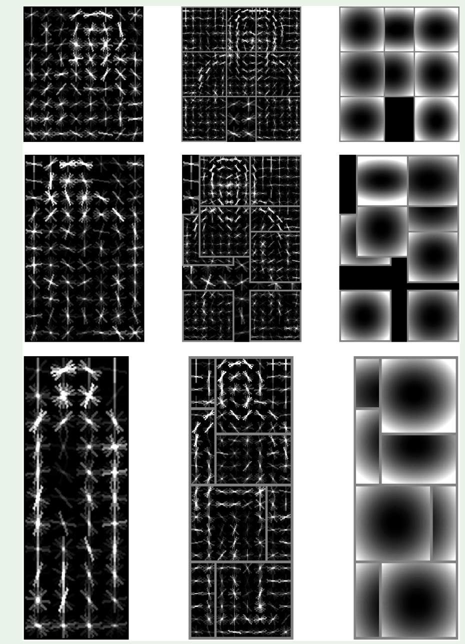
[DT'05]
AP 0.12



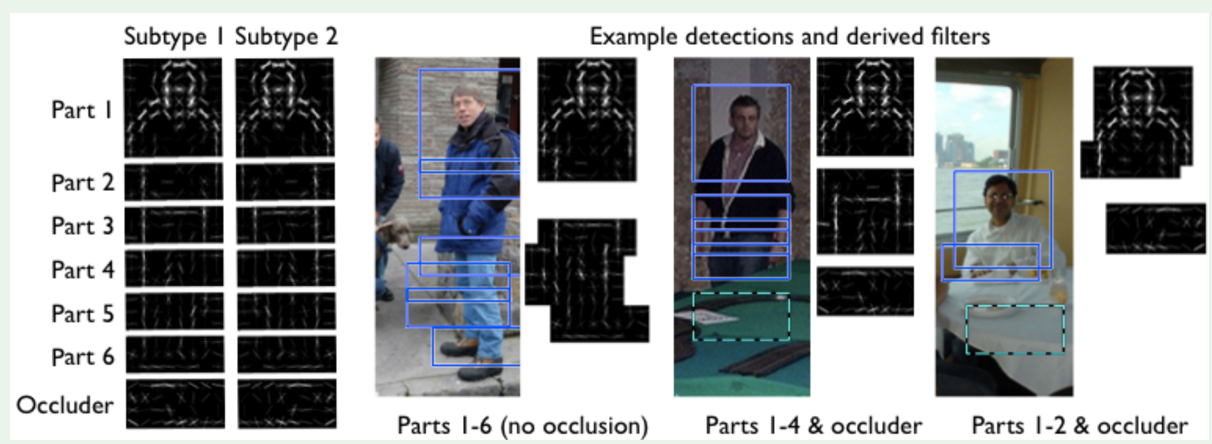
[FMR'08]
AP 0.27



[FGMR'10]
AP 0.36



[GFM voc-release5]
AP 0.45



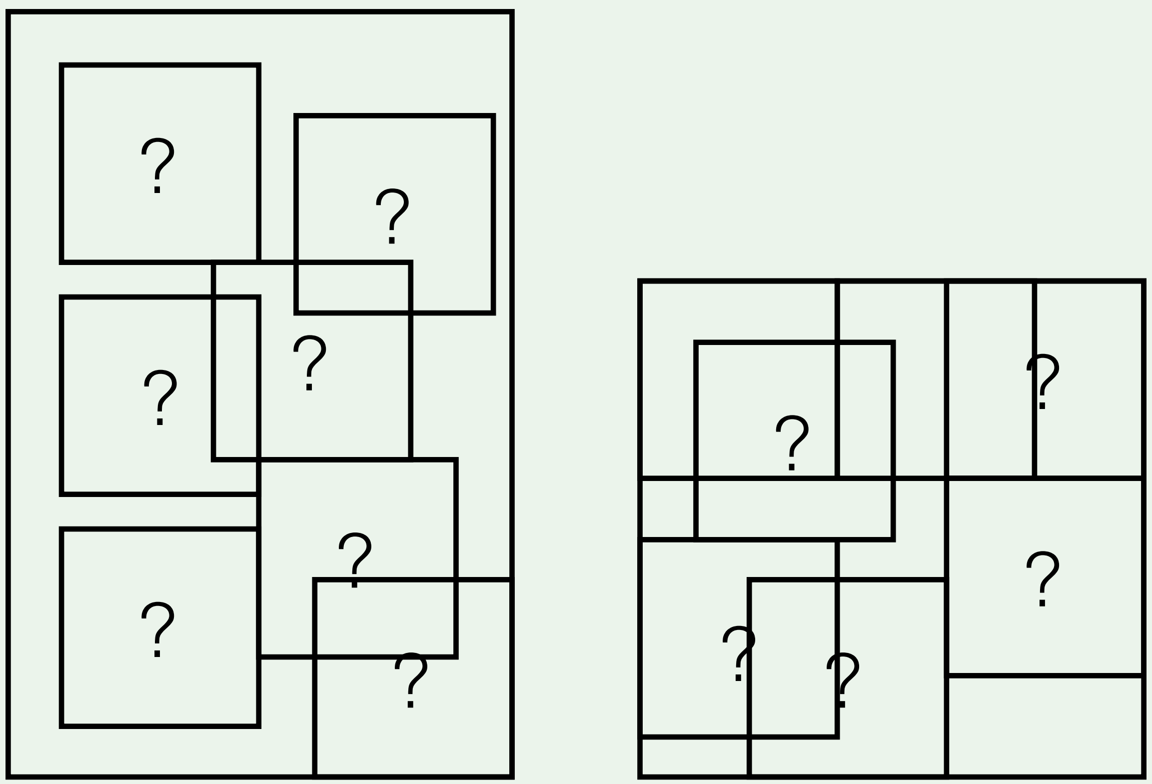
[Girshick, Felzenszwalb, McAllester '11]
AP 0.49

Object detection with grammar models

Code at www.cs.berkeley.edu/~rbg/voc-release5

Part 2: DPM parameter learning

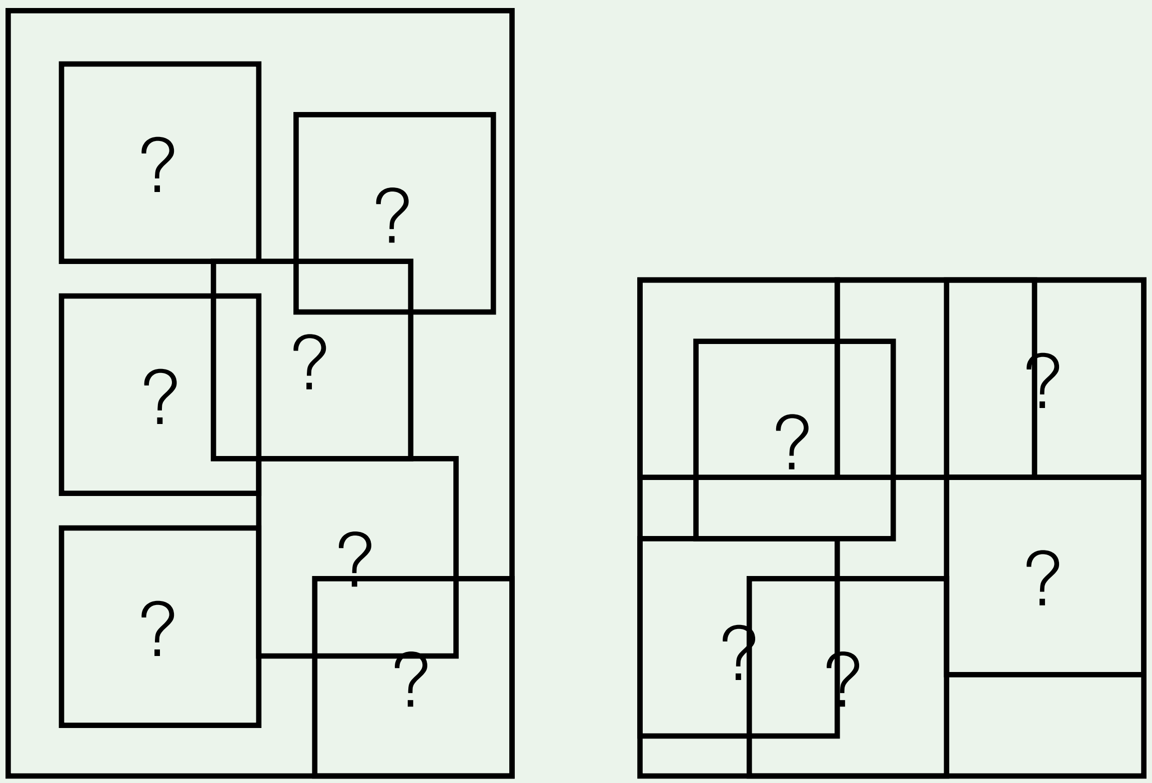
given fixed model *structure*



component 1 component 2

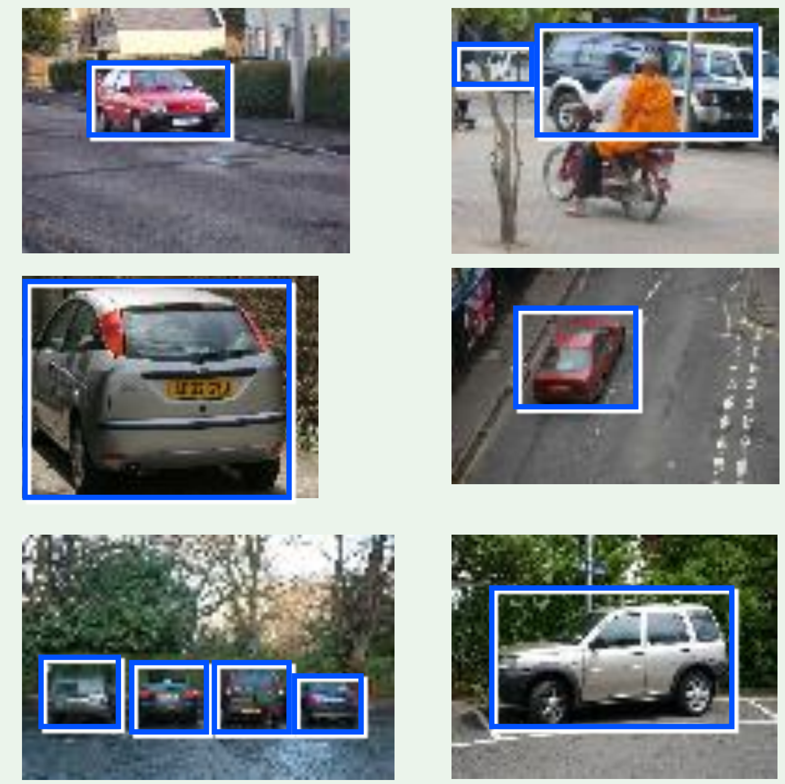
Part 2: DPM parameter learning

given fixed model *structure*



component 1 component 2

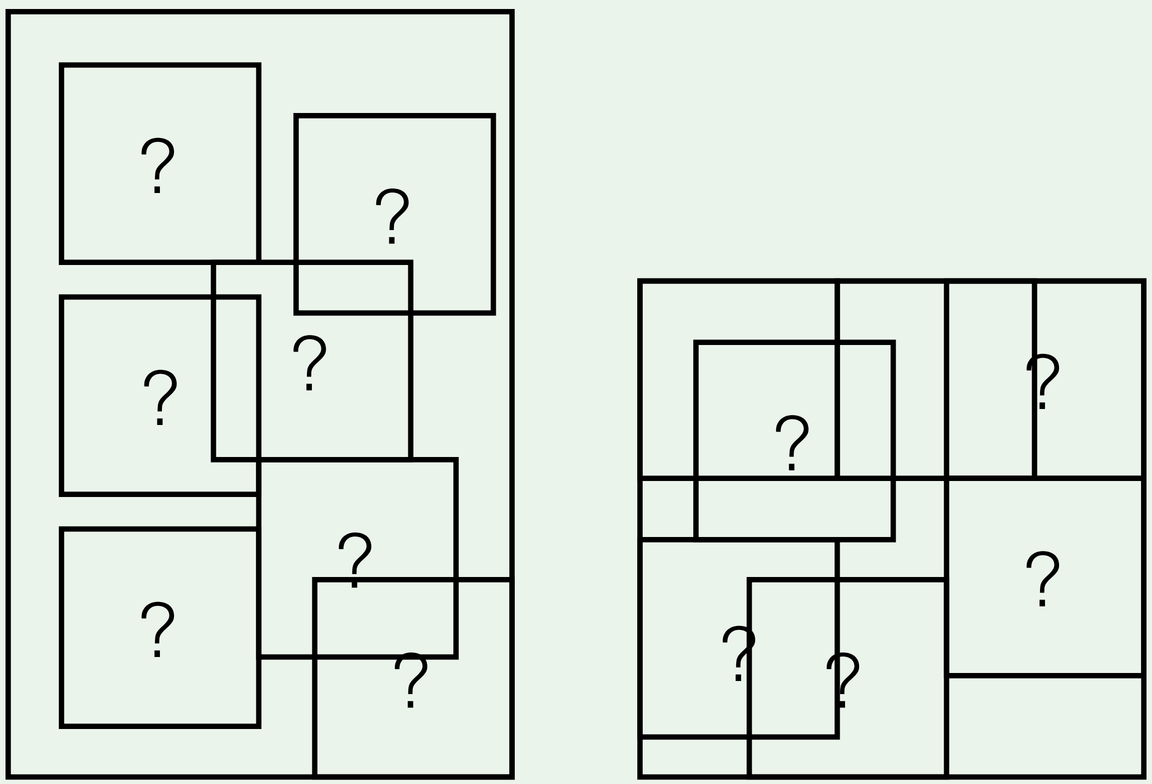
training images y



+1

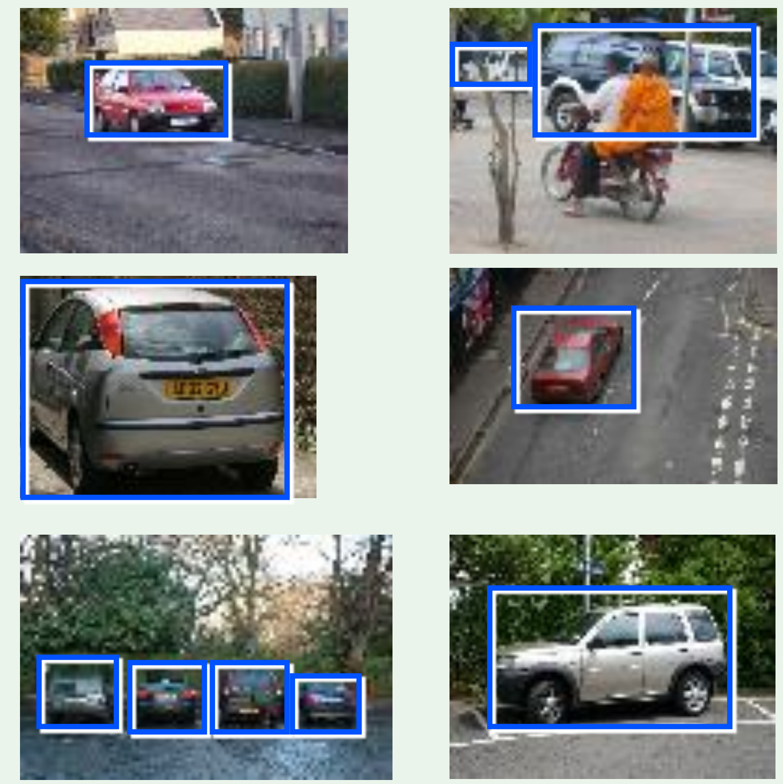
Part 2: DPM parameter learning

given fixed model *structure*



component 1 component 2

training images **y**

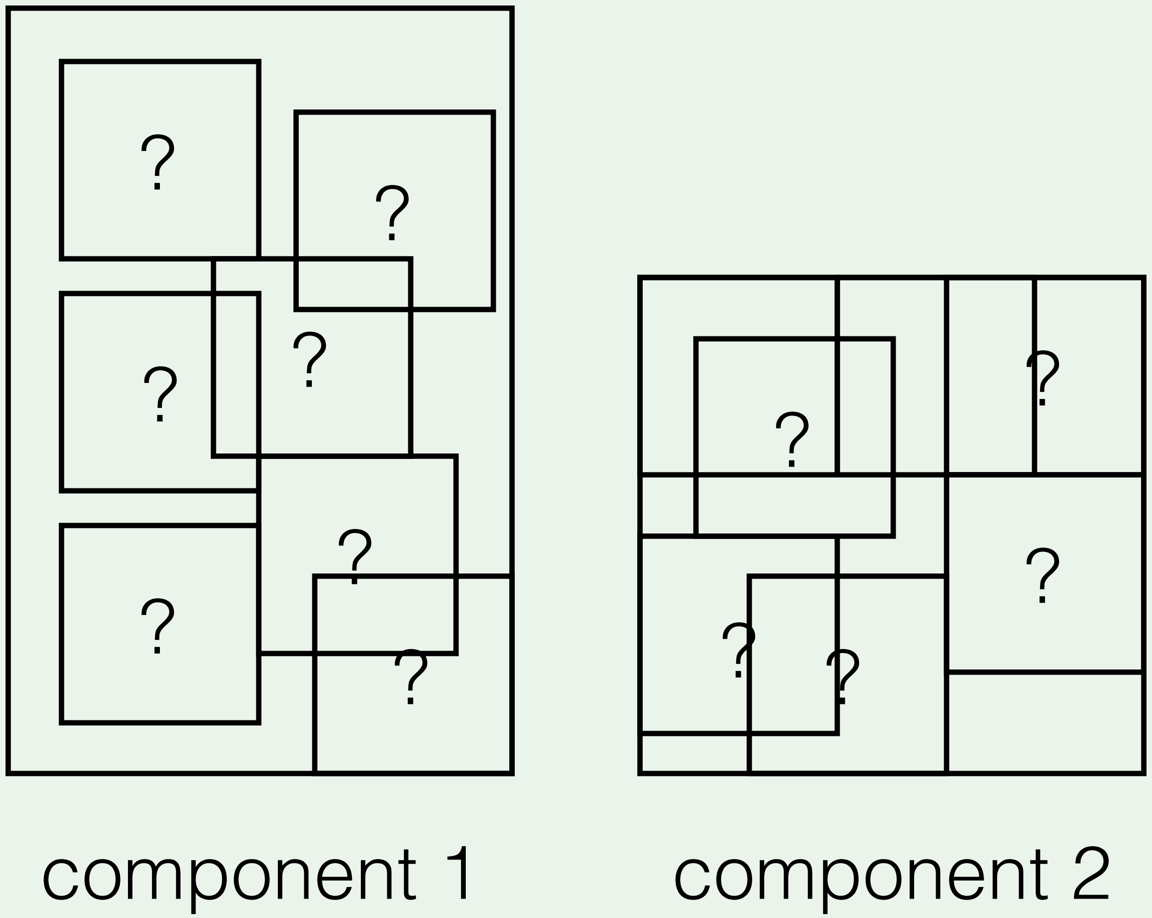


+1

-1

Part 2: DPM parameter learning

given fixed model *structure*



training images y

Parameters to learn:

– biases (per component)
– deformation costs (per part)
– filter weights

Linear parameterization of sliding window score

$$z = (p_1, \ldots, p_n)$$

$$\text{score}(I, p_0) = \max_{p_1,\ldots,p_n} \sum_{i=0}^{n} m_i(I, p_i) - \sum_{i=1}^{n} d_i(p_0, p_i)$$

Filter scores Spring costs

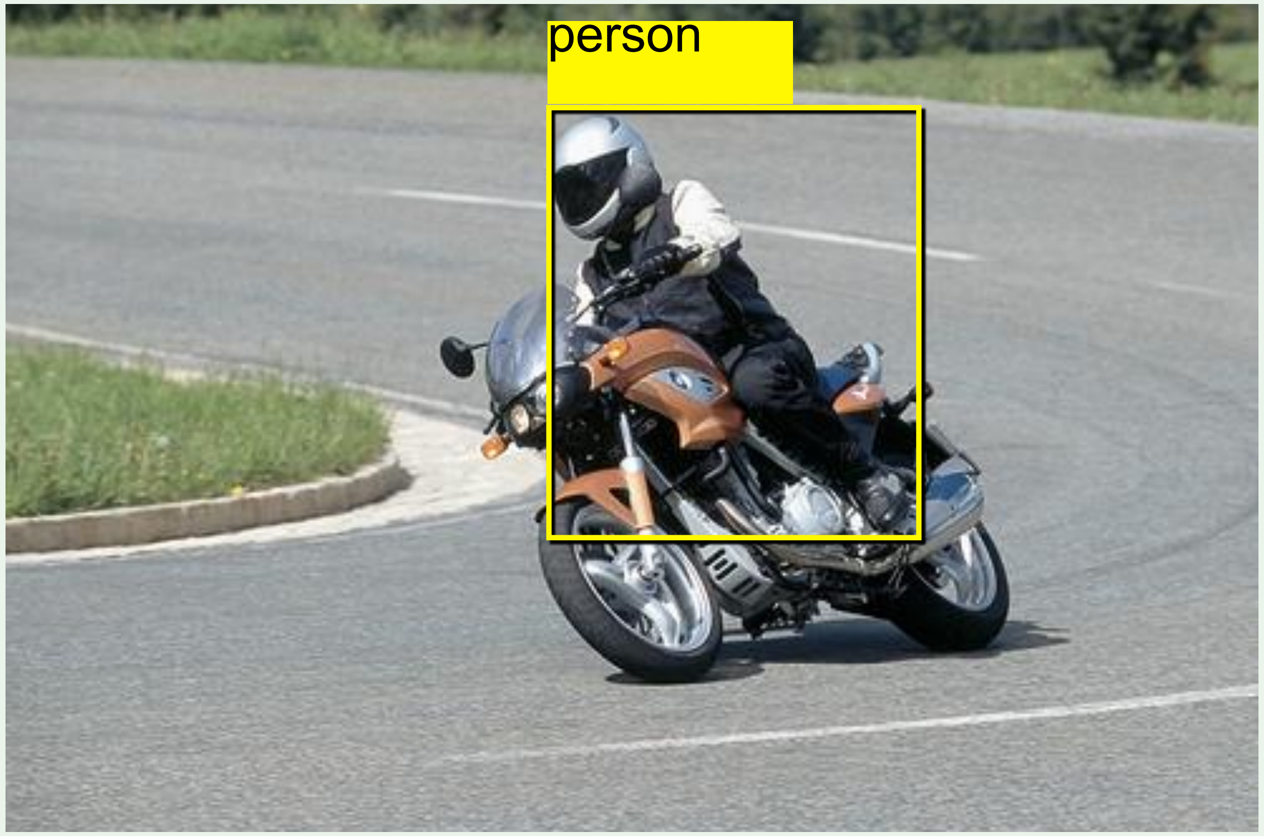
Filter scores $$m_i(I, p_i) = \mathbf{w}_i \cdot \phi(I, p_i)$$

Spring costs $$d_i(p_0, p_i) = \mathbf{d}_i \cdot (dx^2, dy^2, dx, dy)$$

$$score(I, p_0) = \max_{z} \mathbf{w} \cdot \Phi(I, (p_0, z))$$

Positive examples ($y = +1$)

x specifies an image and bounding box



We want

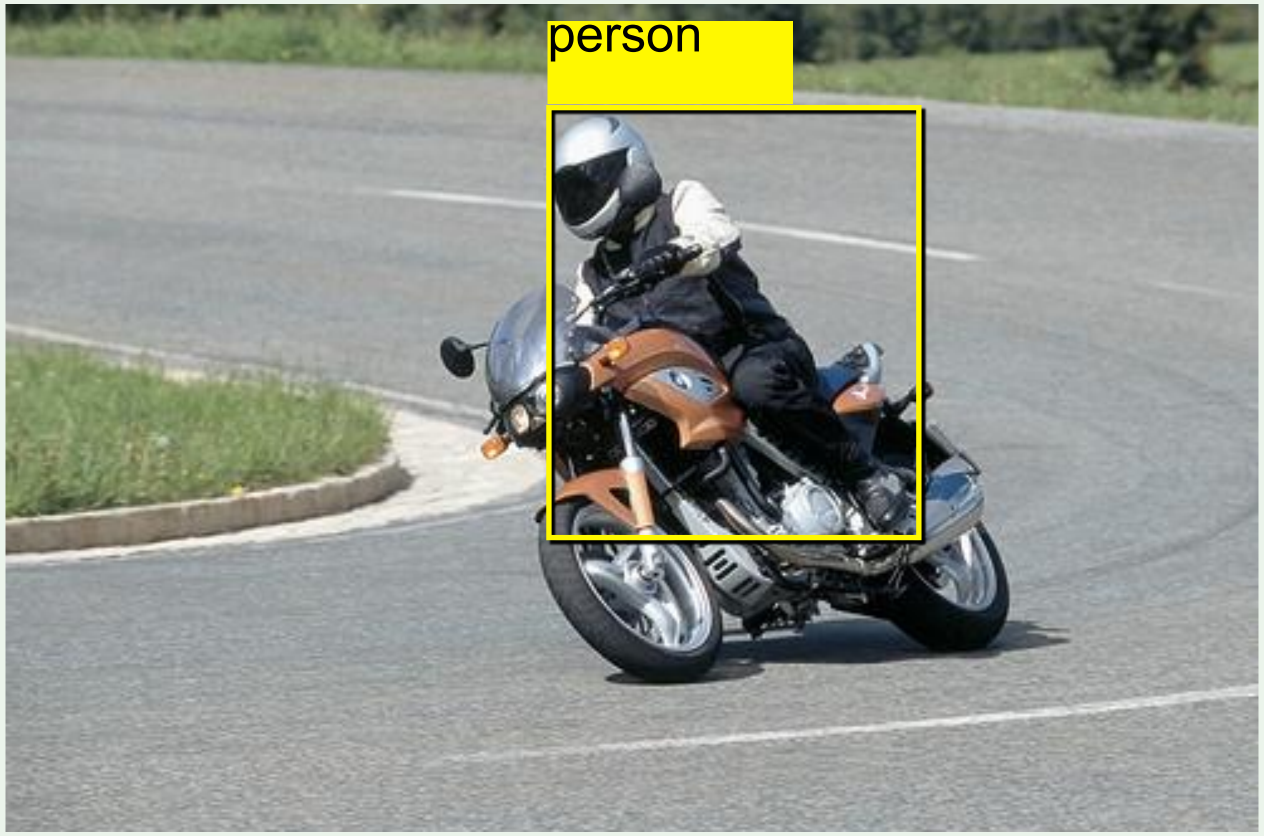
$$f_{\mathbf{w}}(x) = \max_{z \in Z(x)} \mathbf{w} \cdot \Phi(x, z)$$

to score >= +1

$Z(x)$ includes all z with more than 70% overlap with ground truth

Positive examples ($y = +1$)

x specifies an image and bounding box



We want

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

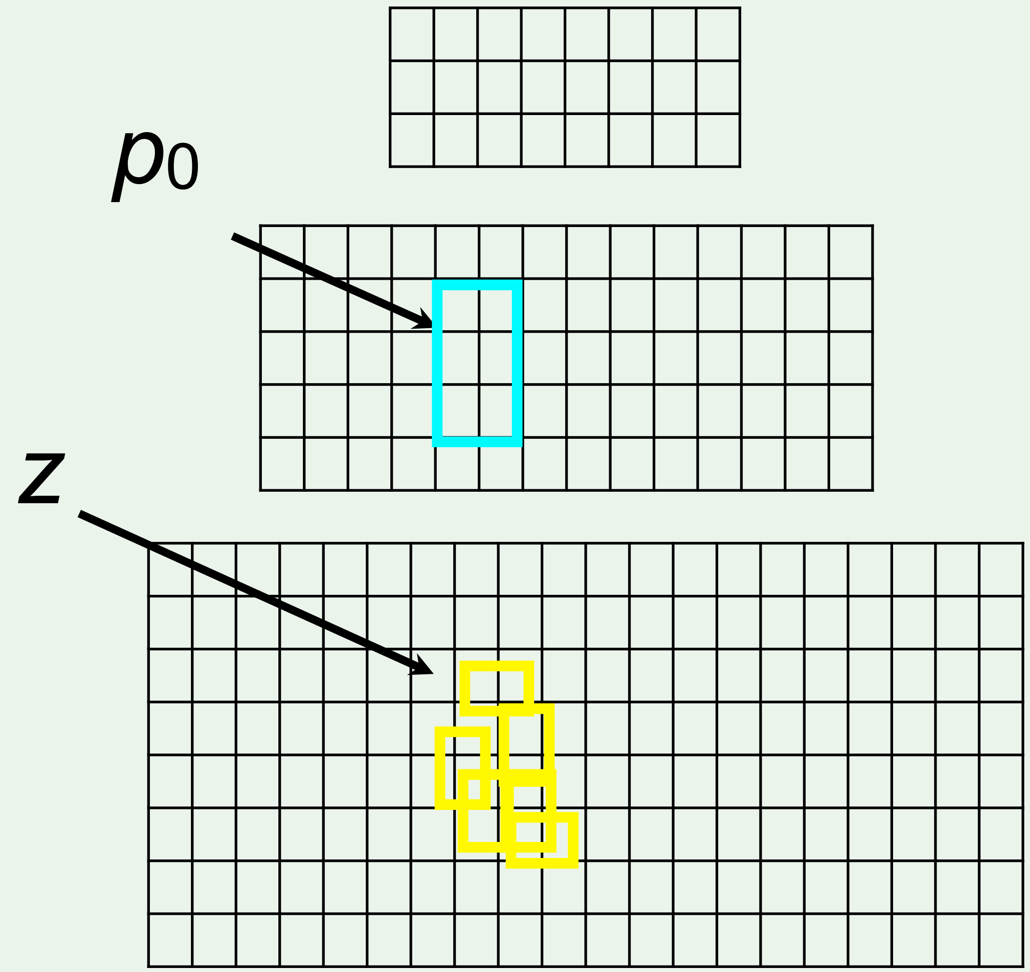
to score >= +1

At least one configuration scores high

$Z(x)$ includes all z with more than 70% overlap with ground truth

Negative examples (y = -1)

x specifies an image and a HOG pyramid location p_0



We want

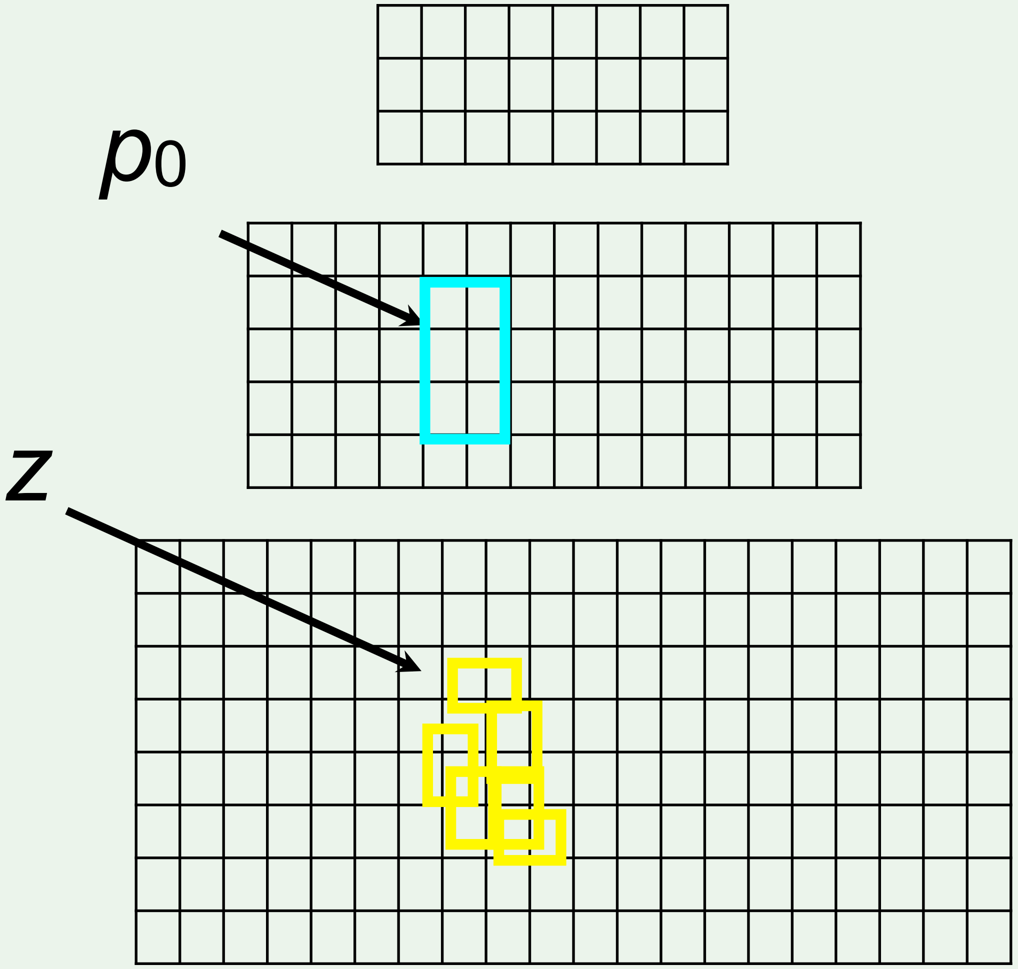
$$f_{\mathbf{w}}(x) = \max_{z \in Z(x)} \mathbf{w} \cdot \Phi(x, z)$$

to score <= -1

$Z(x)$ restricts the root to p_0 and allows *any* placement of the other filters

Negative examples (y = -1)

x specifies an image and a HOG pyramid location p_0



We want

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

to score <= -1

All configurations score low

$Z(x)$ restricts the root to p_0 and allows *any* placement of the other filters

Typical dataset

300 – 8,000 positive examples

500 million to 1 billion negative examples
(not including latent configurations!)

Large-scale optimization!

How we learn parameters: latent SVM

$$E(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \max\{0, 1 - y_i f_{\mathbf{w}}(x_i)\}$$

How we learn parameters: latent SVM

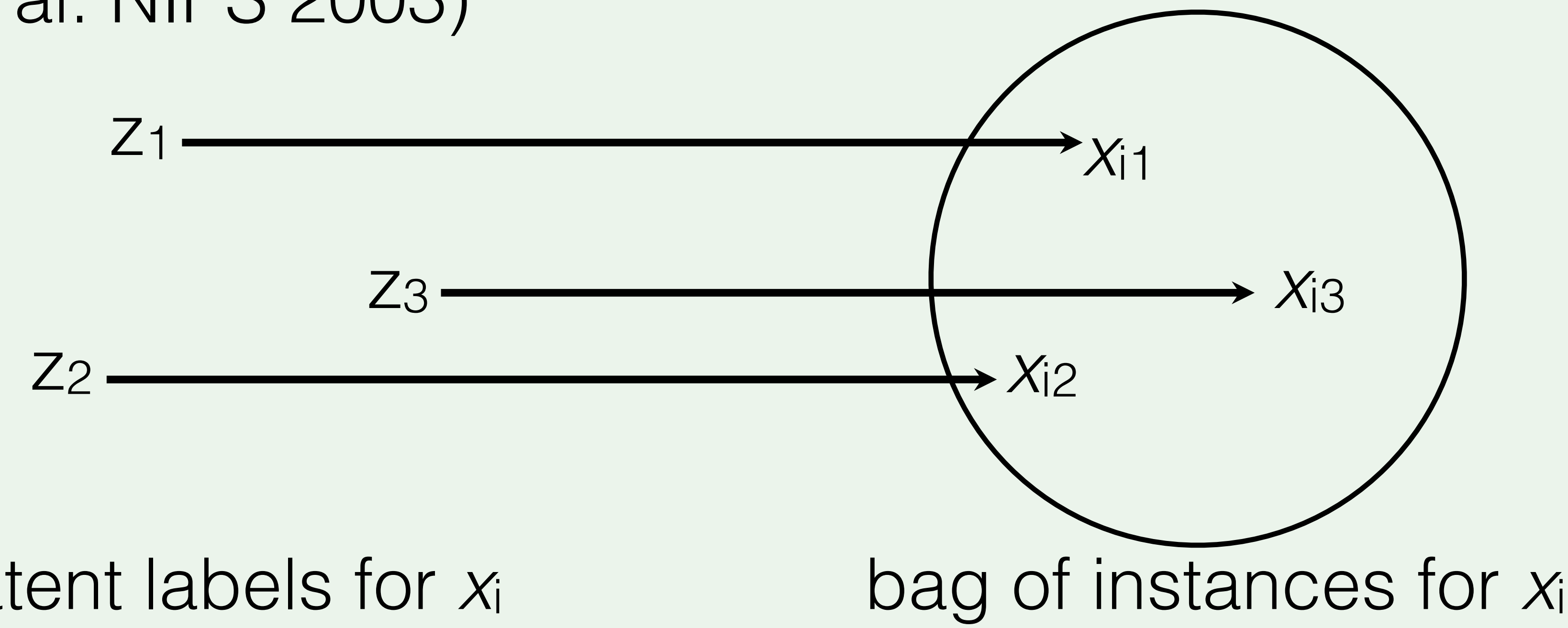
$$E(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \max\{0, 1 - y_i f_{\mathbf{w}}(x_i)\}$$

$$E(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i\in P} \max\{0, 1 - \max_{z\in Z(x)} \mathbf{w}\cdot\Phi(x_i, z)\} + C\sum_{i\in N} \max\{0, 1 + \max_{z\in Z(x)} \mathbf{w}\cdot\Phi(x_i, z)\}$$

P: set of positive examples
N: set of negative examples

Latent SVM and Multiple Instance Learning via MI-SVM

Latent SVM is mathematically equivalent to MI-SVM
(Andrews et al. NIPS 2003)



Latent SVM can be written as a latent structural SVM
(Yu and Joachims ICML 2009)

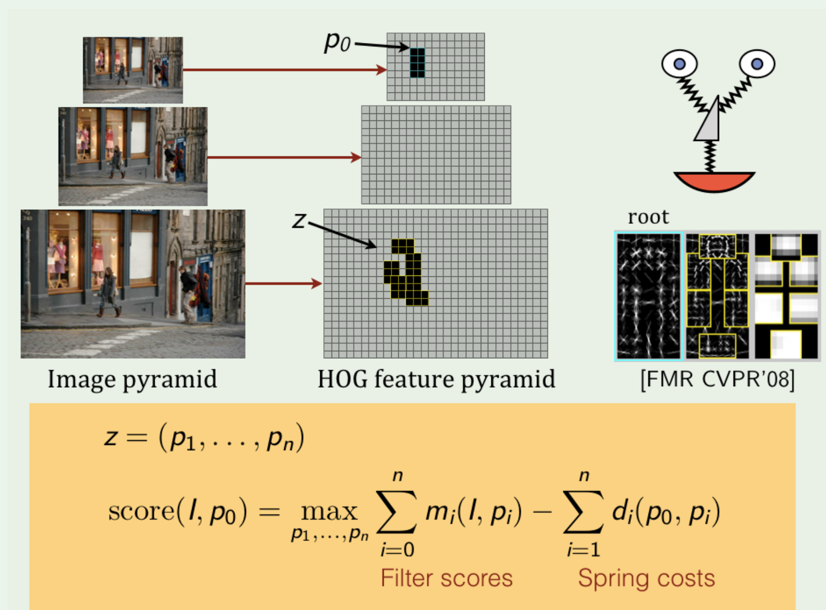
– natural optimization algorithm is concave-convex procedure

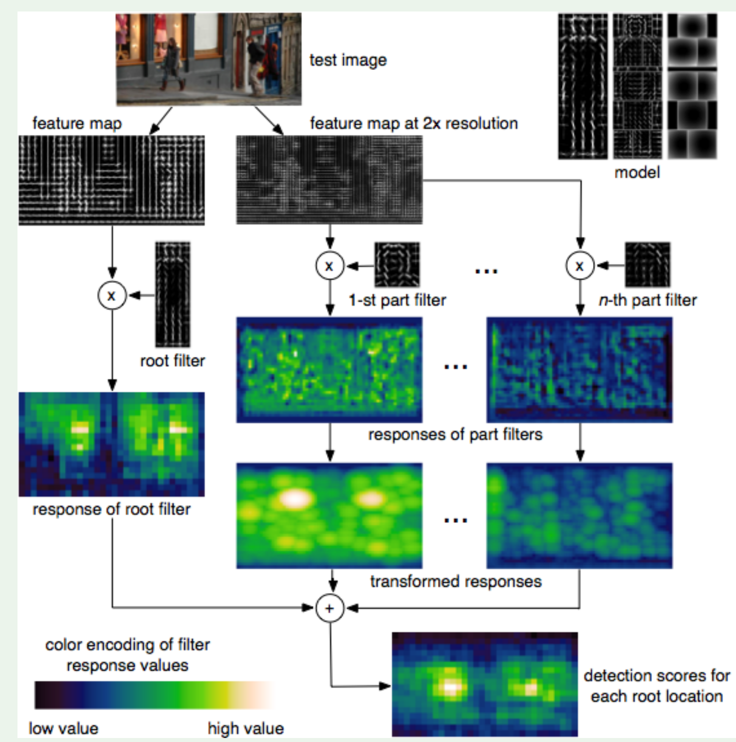
– similar to, but not exactly the same as, coordinate descent

Step 1

$$Z_{P_i} = \underset{z \in Z(x_i)}{\operatorname{argmax}} \, \mathbf{w}_{(t)} \cdot \Phi(x_i, z) \quad \forall i \in P$$

This is just detection:





We know how to do this!

Step 2

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i \in P} \max\{0, 1 - \mathbf{w} \cdot \Phi(x_i, Z_{Pi})\} + C\sum_{i \in N} \max\{0, 1 + \max_{z \in Z(x)} \mathbf{w} \cdot \Phi(x_i, z)\}$$

Convex!

Step 2

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i\in P} \max\{0, 1-\mathbf{w}\cdot\Phi(x_i, Z_{Pi})\}$$
$$+ C\sum_{i\in N} \max\{0, 1+\max_{z\in Z(x)} \mathbf{w}\cdot\Phi(x_i, z)\}$$

Convex!

Similar to a structural SVM

Step 2

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i\in P} \max\{0, 1 - \mathbf{w}\cdot\Phi(x_i, Z_{Pi})\}$$
$$+ C\sum_{i\in N} \max\{0, 1 + \max_{z\in Z(x)} \mathbf{w}\cdot\Phi(x_i, z)\}$$

Convex!

Similar to a structural SVM

But, recall 500 million to 1 billion negative examples!

Step 2

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i\in P}\max\{0, 1-\mathbf{w}\cdot\Phi(x_i, Z_{Pi})\}$$
$$+ C\sum_{i\in N}\max\{0, 1+\max_{z\in Z(x)}\mathbf{w}\cdot\Phi(x_i, z)\}$$

Convex!

Similar to a structural SVM

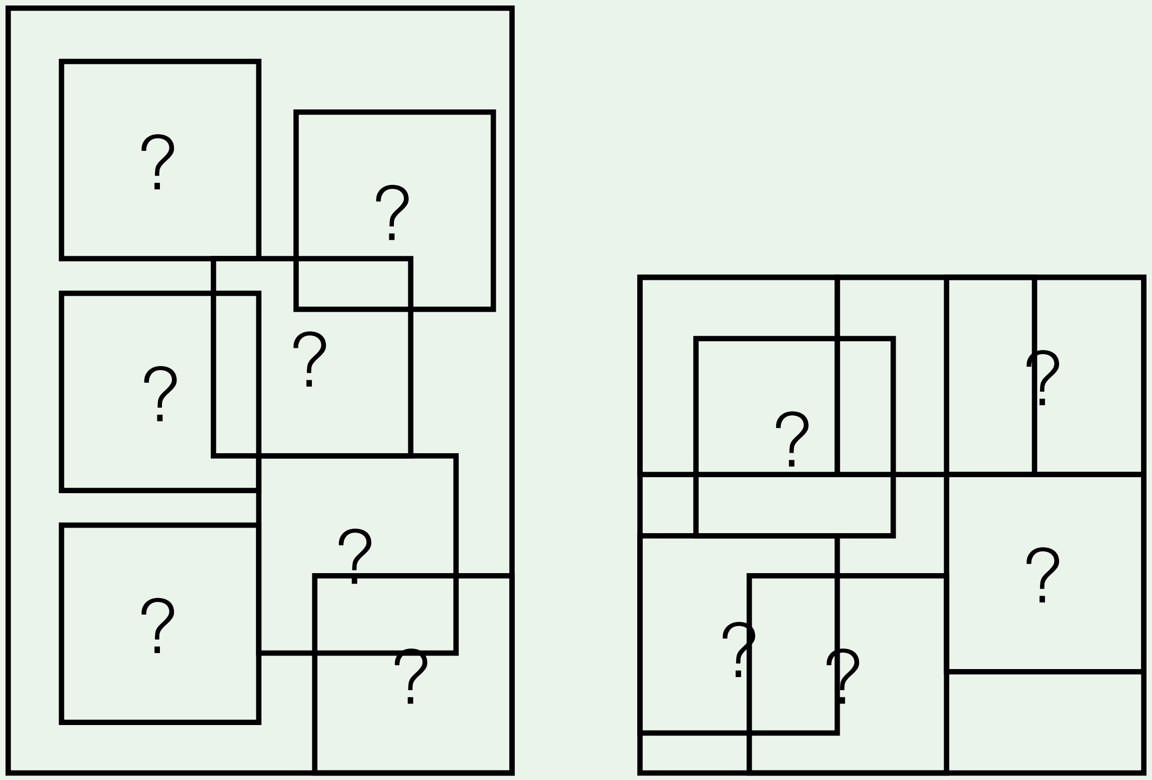
But, recall 500 million to 1 billion negative examples!

Can be solved by a working set method

- "bootstrapping"
- "data mining" / "hard negative mining"
- "constraint generation"
- requires a bit of engineering to make this fast

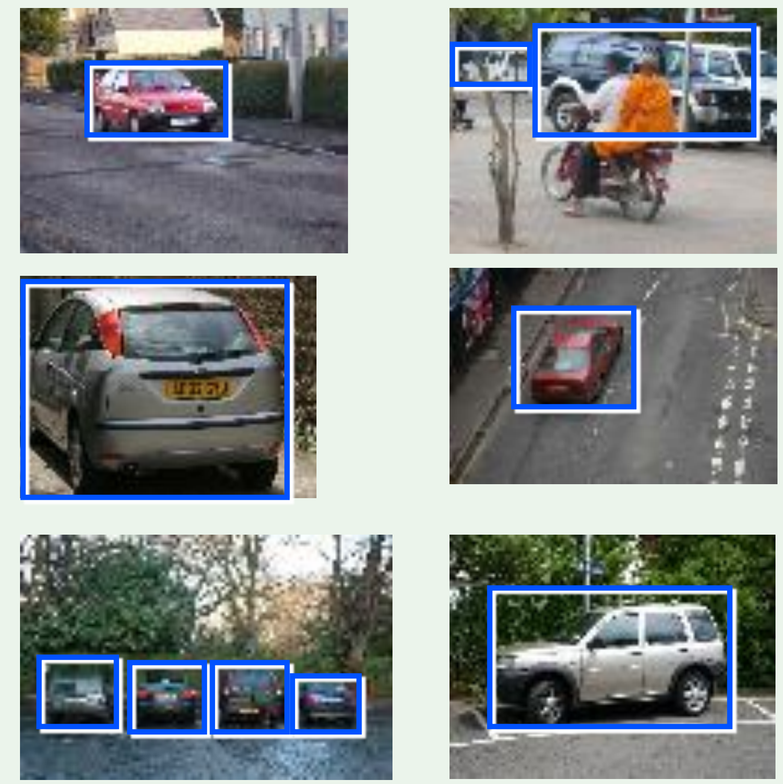
What about the model structure?

<u>Given fixed model *structure*</u>



component 1 component 2

training images y



+1

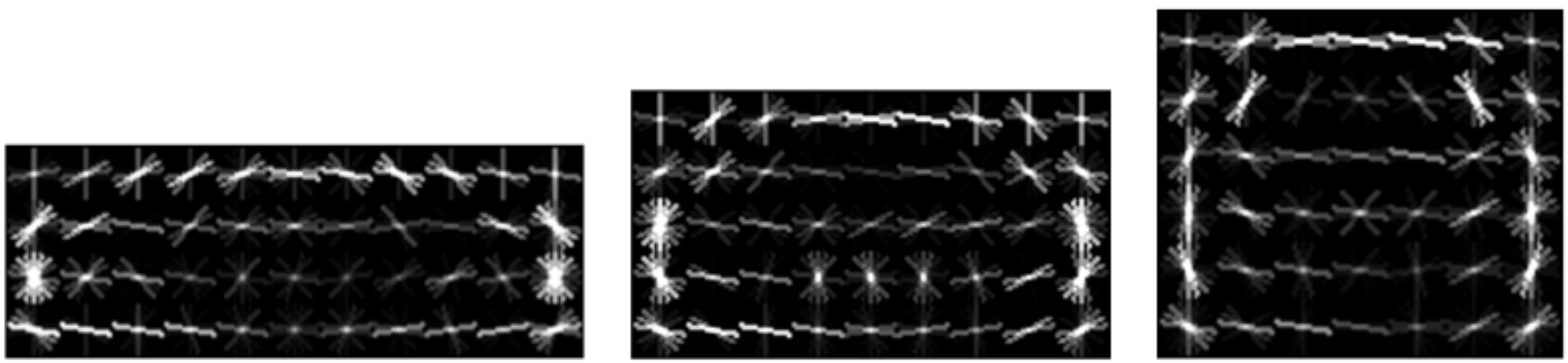
-1

Model structure

– # components
– # parts per component
– root and part filter shapes
– part anchor locations

Learning model structure

1a. Split positives by aspect ratio

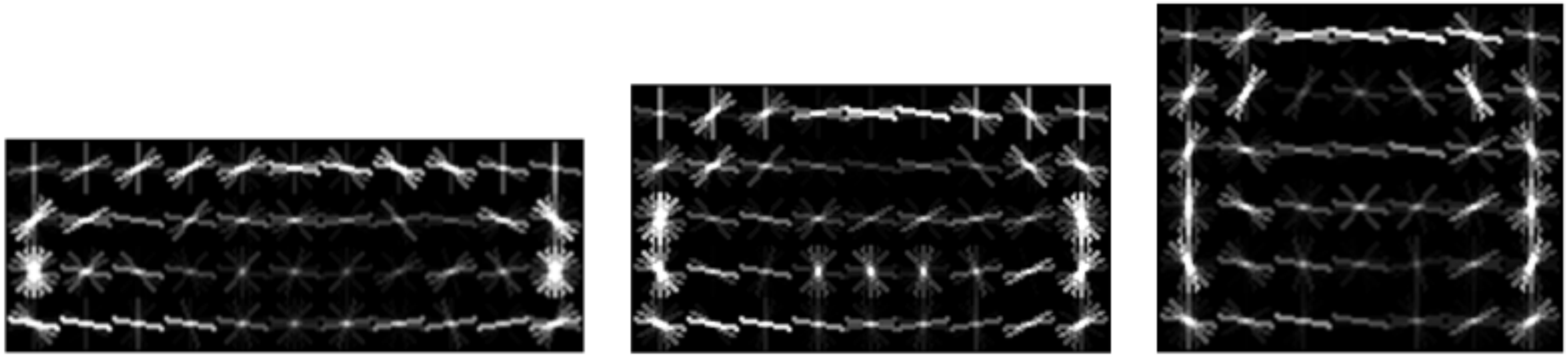


(a) Car component 1 (Phase 1) (b) Car component 2 (Phase 1) (c) Car comp. 3 (Phase 1)

1b. Warp to common size

1c. Train Dalal & Triggs model for each aspect ratio on its own

Learning model structure



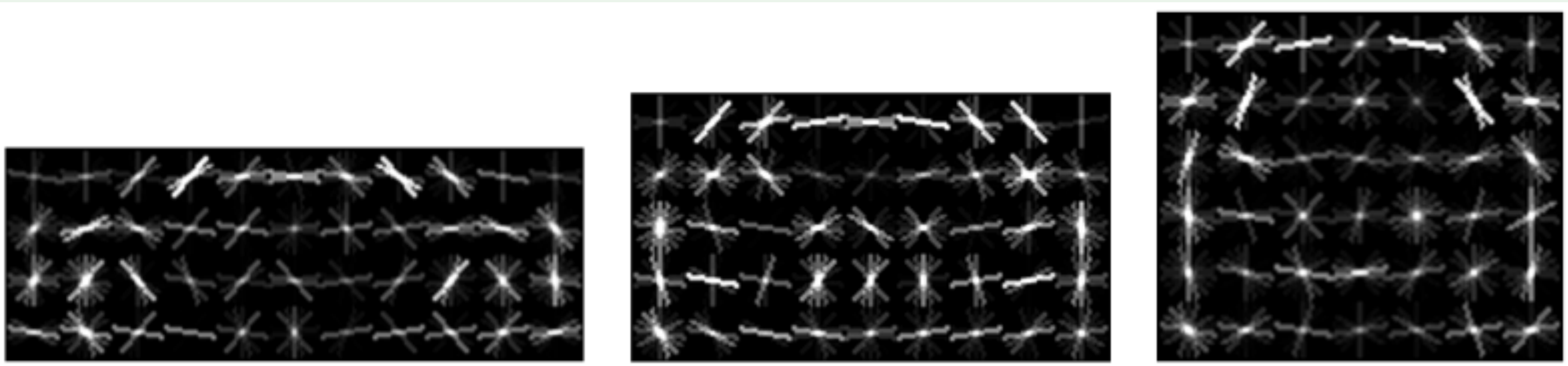
(a) Car component 1 (Phase 1) (b) Car component 2 (Phase 1) (c) Car comp. 3 (Phase 1)

2a. Use D&T filters as initial **w** for LSVM training

Merge components

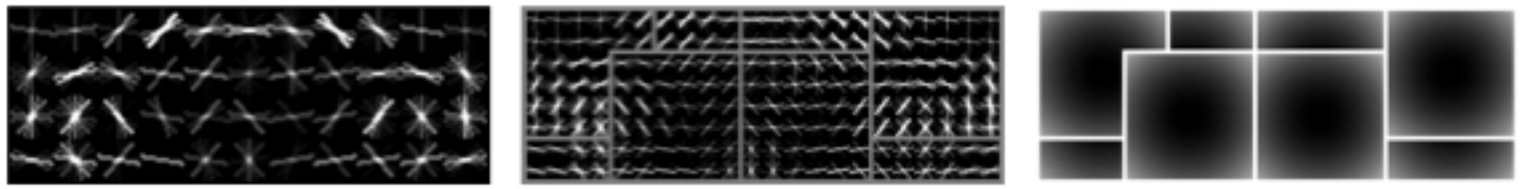
2b. Train with latent SVM

Root filter placement and component choice are latent

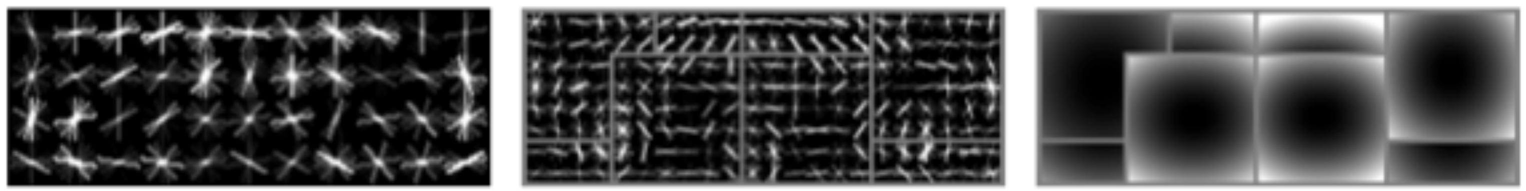


(d) Car component 1 (Phase 2) (e) Car component 2 (Phase 2) (f) Car comp. 3 (Phase 2)

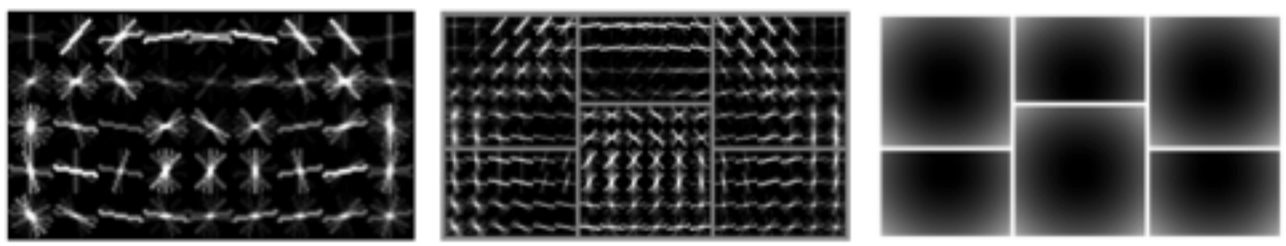
Learning model structure



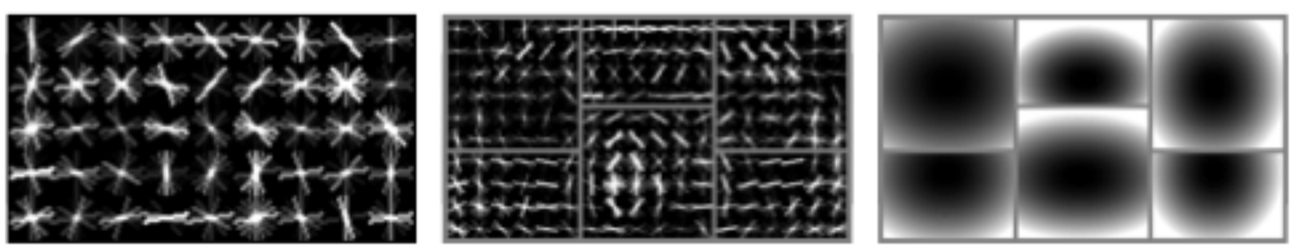
(a) Car component 1 (initial parts)



(b) Car component 1 (trained parts)



(c) Car component 2 (initial parts)

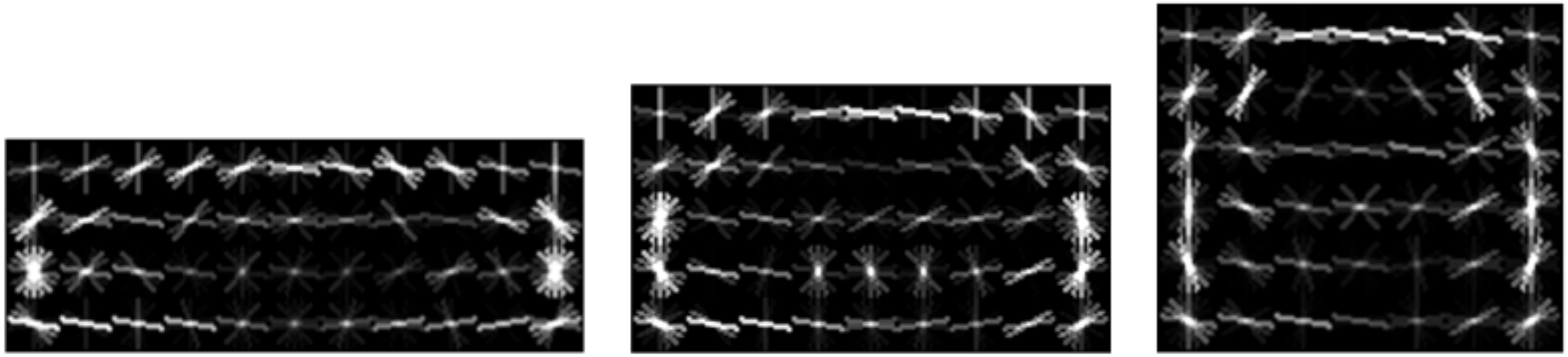


(d) Car component 2 (trained parts)

3a. Add parts to cover high-energy areas of root filters

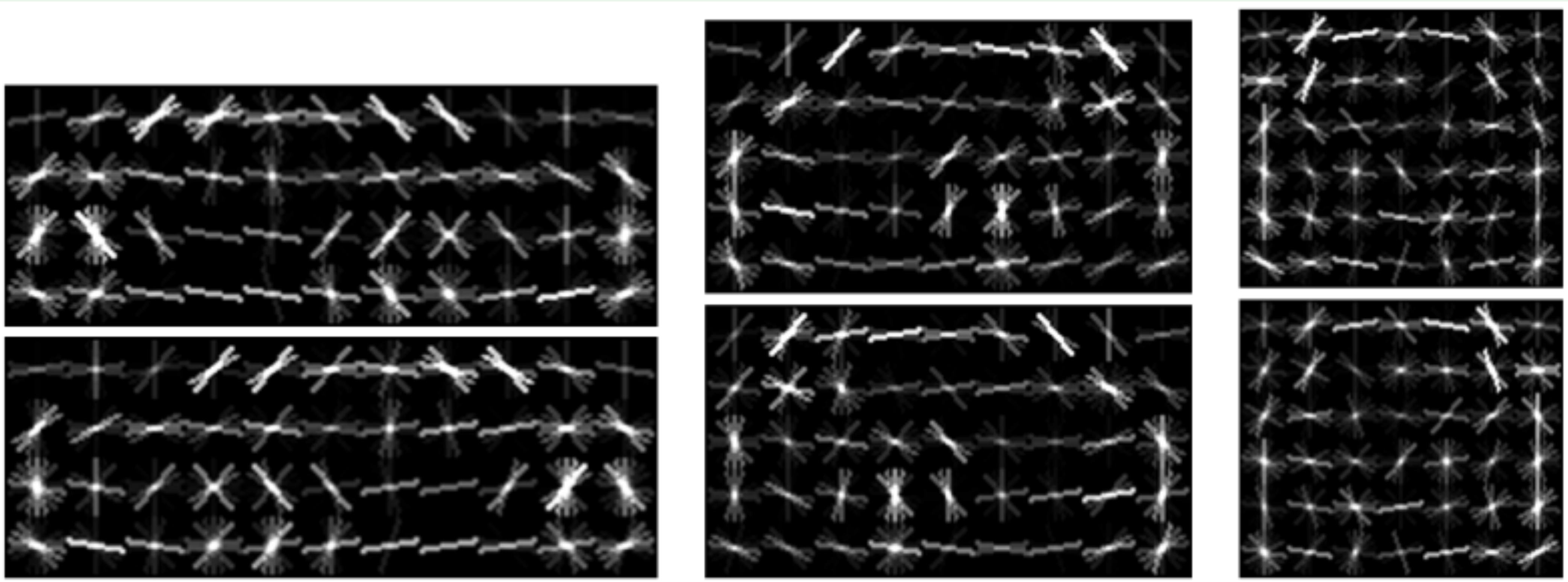
3b. Continue training model with LSVM

Learning model structure



(a) Car component 1 (Phase 1) (b) Car component 2 (Phase 1) (c) Car comp. 3 (Phase 1)

without orientation clustering



(a) Car component 1 (b) Car component 2 (c) Car component 3

with orientation clustering

Learning model structure

In summary

– repeated application of LSVM training to models of increasing complexity

– structure learning involves many heuristics — still a wide open problem!