

Deformable part models

Ross Girshick
UC Berkeley

CS231B Stanford University
Guest Lecture April 16, 2013

Image understanding

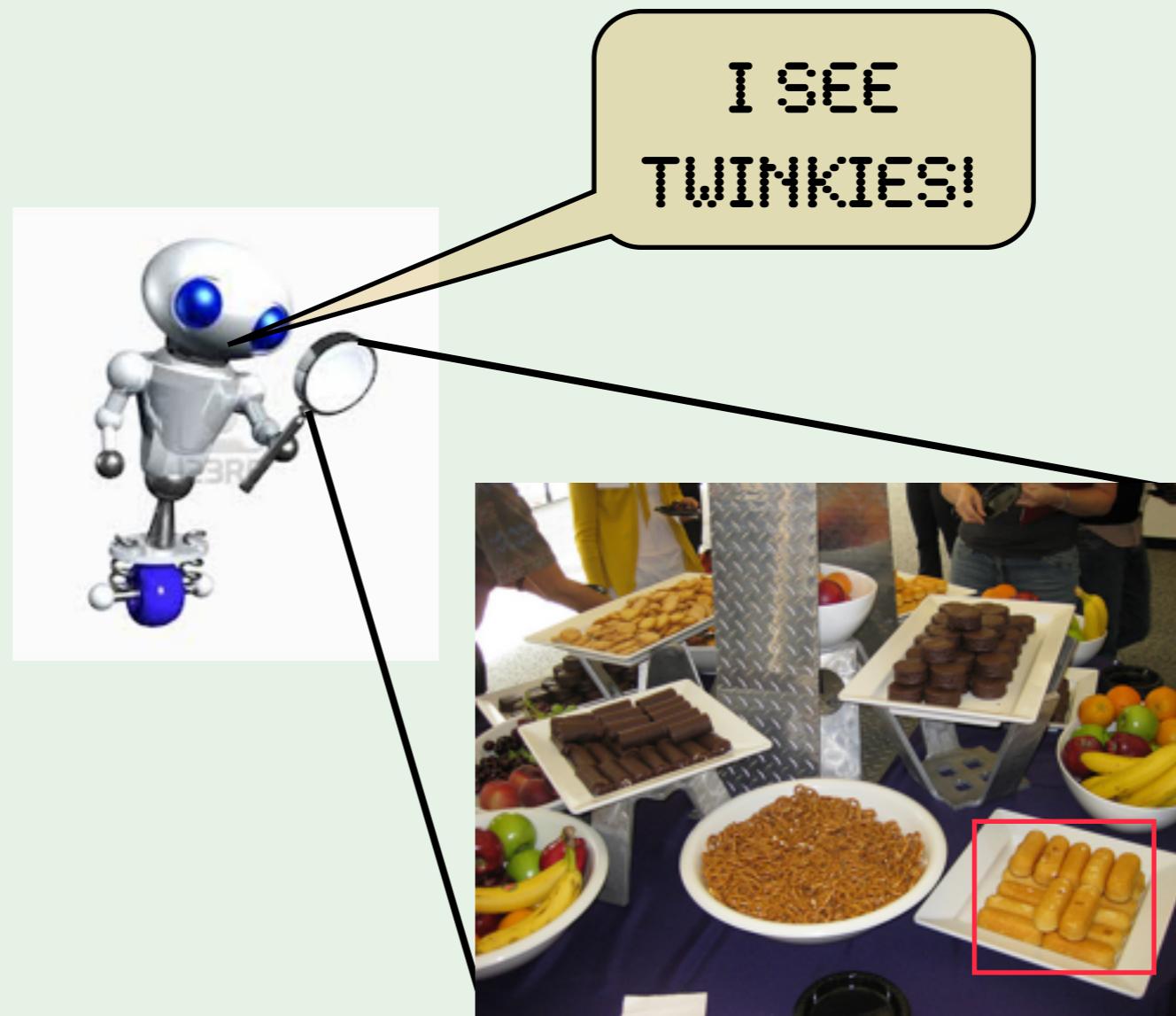
Snack time in the lab



What objects are where?



•
•
•

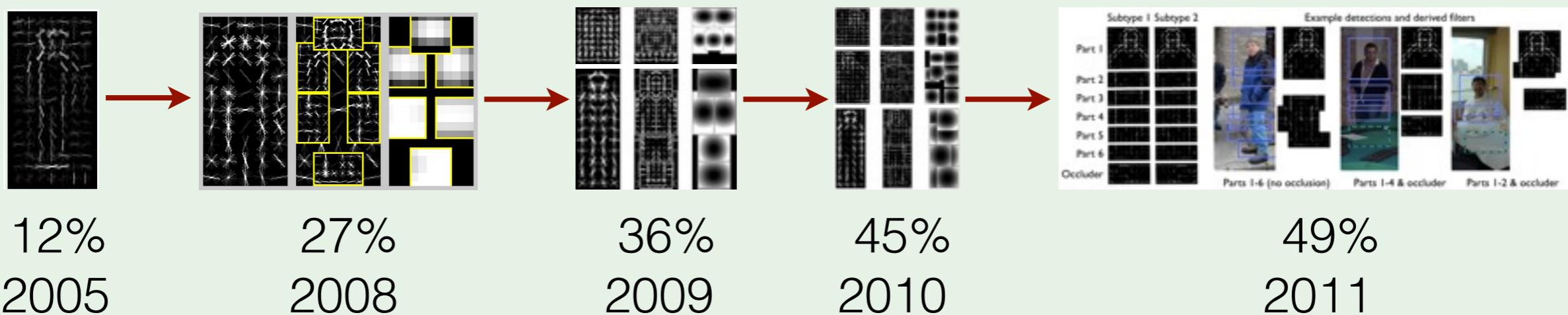


robot: “I see a table with twinkies,
pretzels, fruit, and some mysterious
chocolate things...”

DPM lecture overview

Part 1: modeling

Part 2: learning

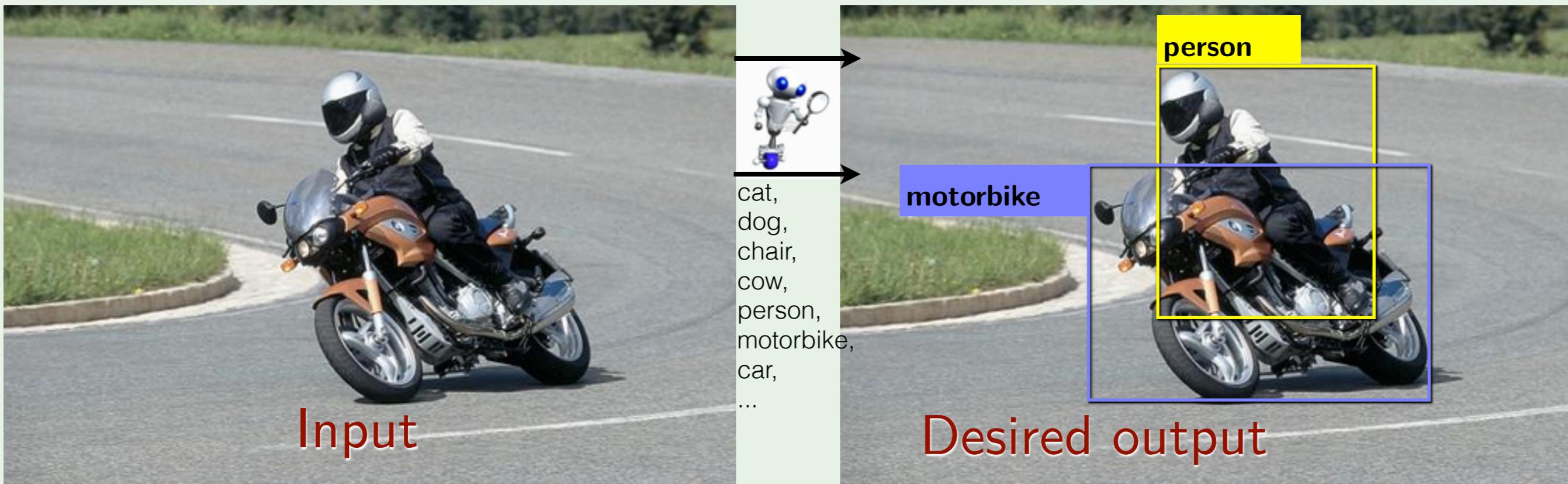


Formalizing the object detection task

Many possible ways

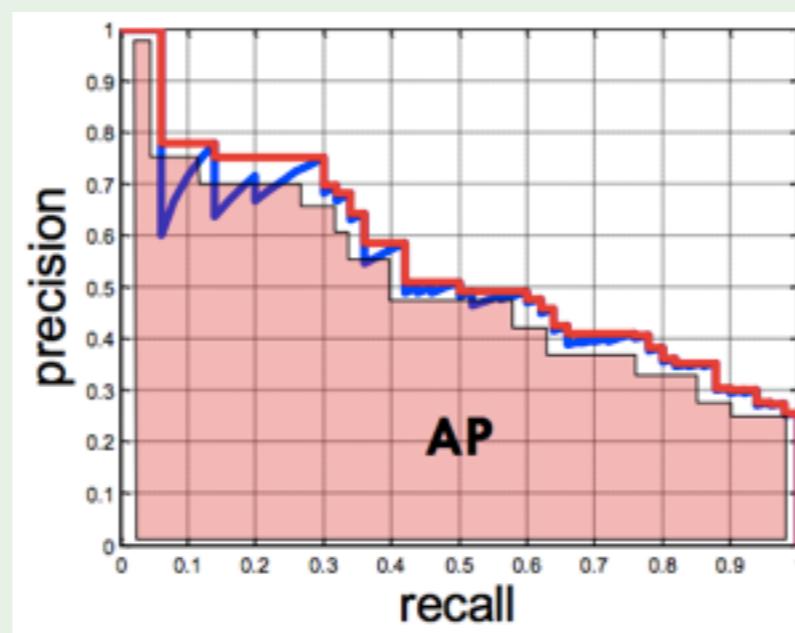
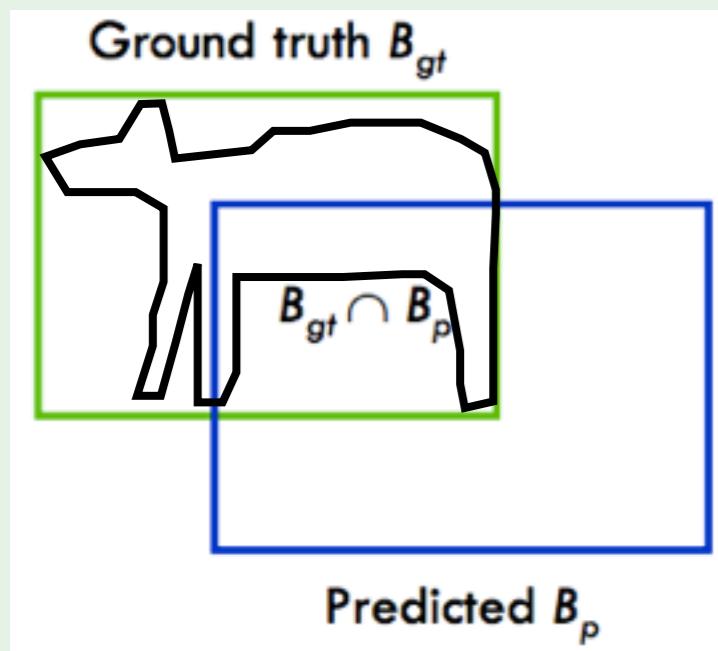
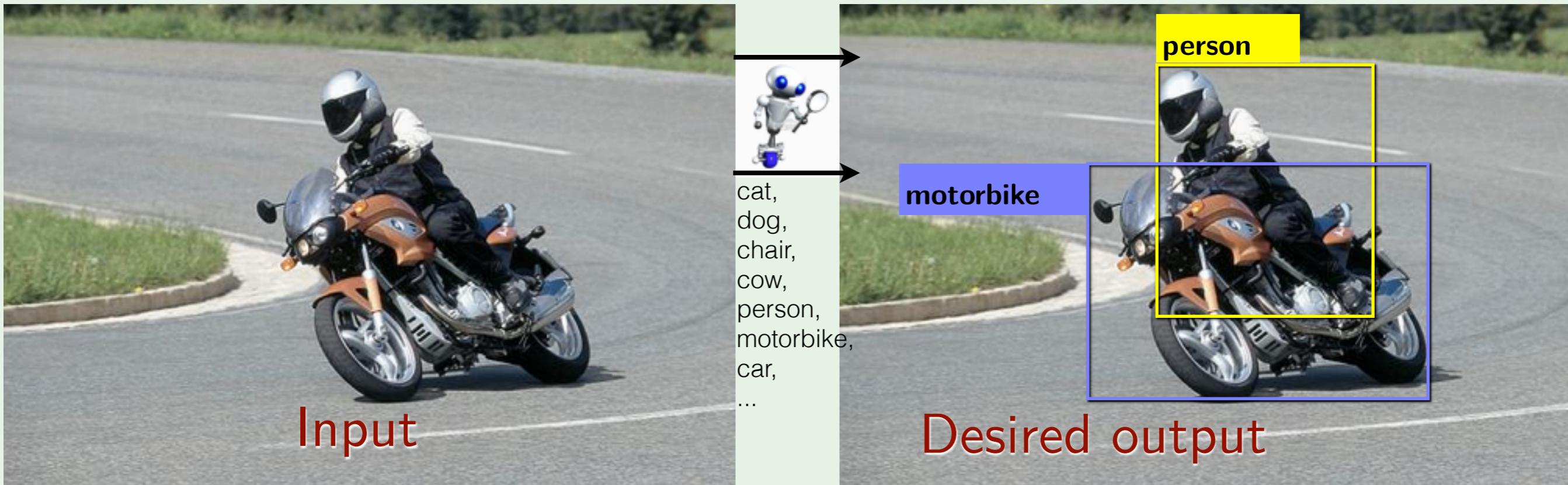
Formalizing the object detection task

Many possible ways, this one is popular:



Formalizing the object detection task

Many possible ways, this one is popular:



Performance summary:
Average Precision (AP)
0 is worst 1 is perfect

Benchmark datasets

PASCAL VOC 2005 – 2012

- 54k objects in 22k images
- 20 object classes
- annual competition

Benchmark datasets

PASCAL VOC 2005 – 2012

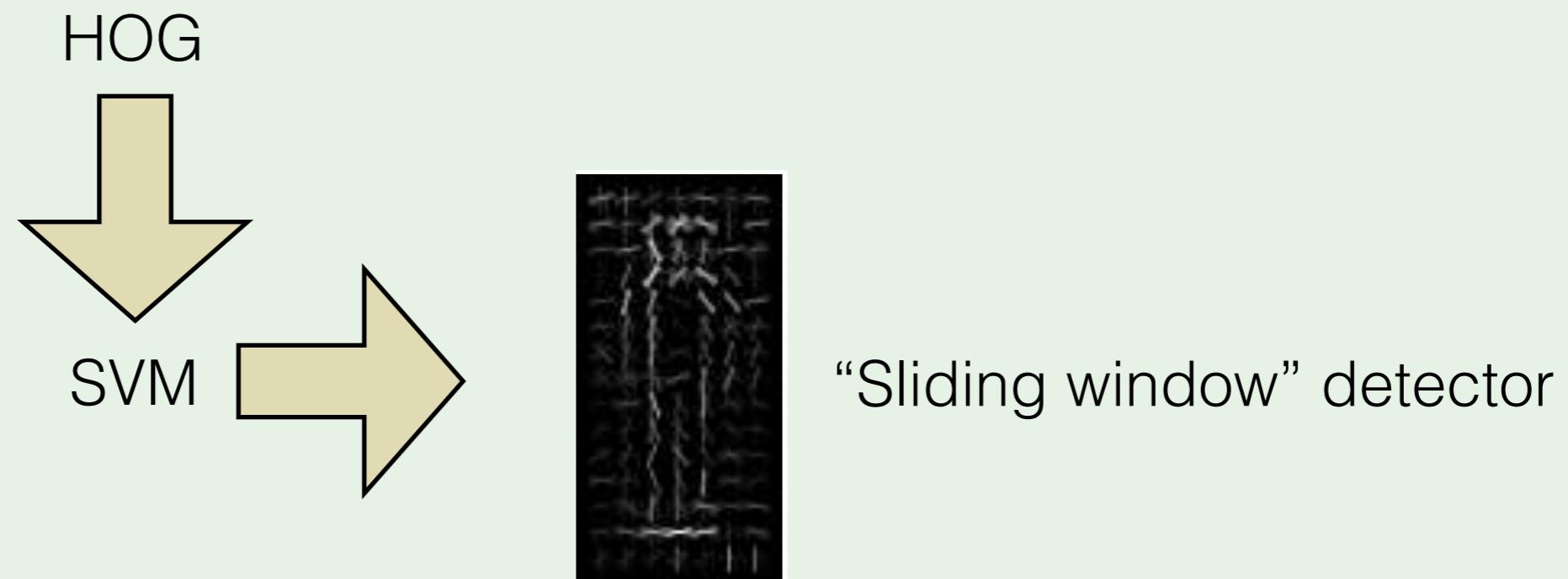
- 54k objects in 22k images
- 20 object classes
- annual competition



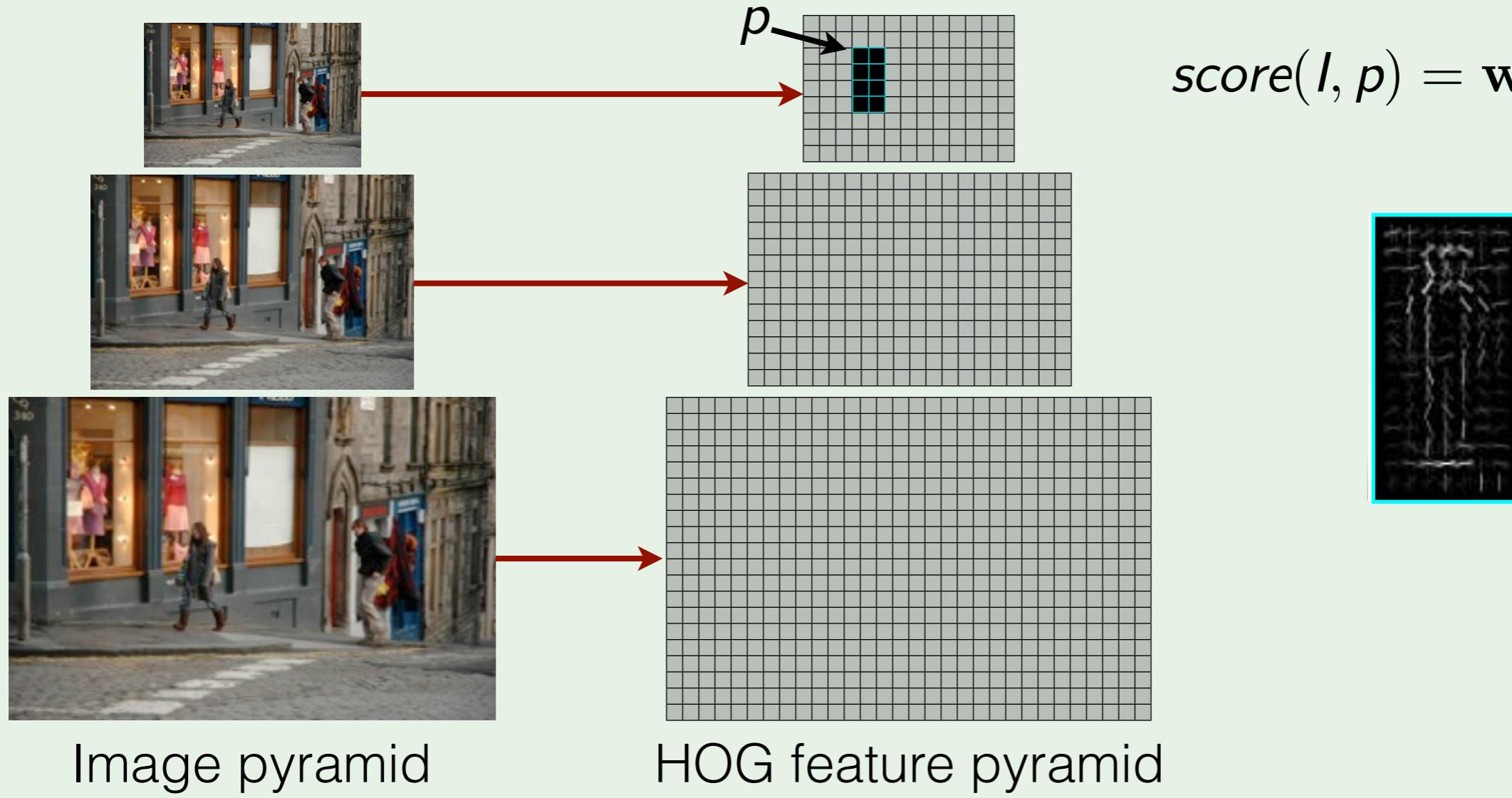
Reduction to binary classification



neg = { ... background patches ... }

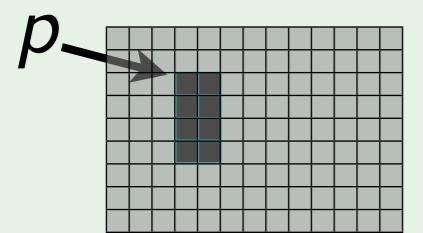


Sliding window detection

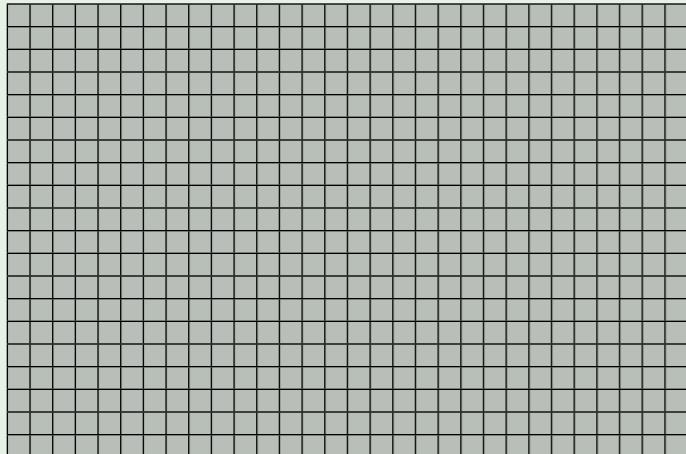
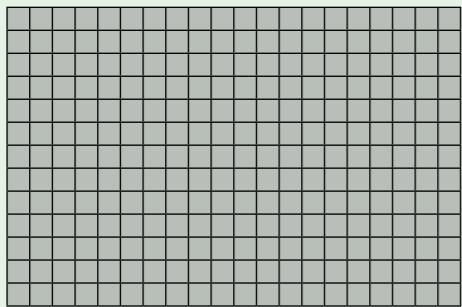


- Compute HOG of the whole image at multiple resolutions
- Score every subwindow of the feature pyramid
- Apply non-maxima suppression

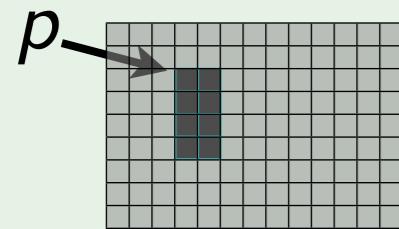
Detection



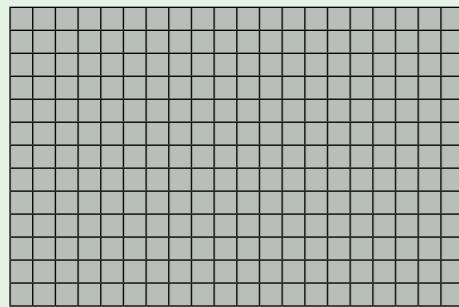
number of locations $p \sim 250,000$ per image



Detection

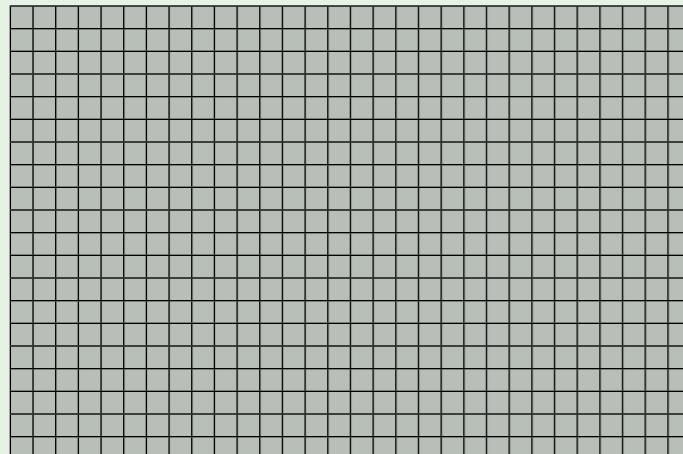


number of locations $p \sim 250,000$ per image

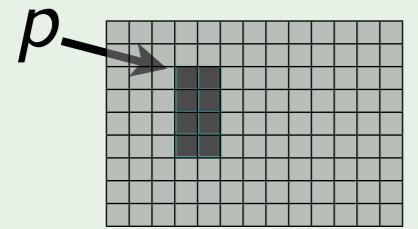


test set has ~ 5000 images

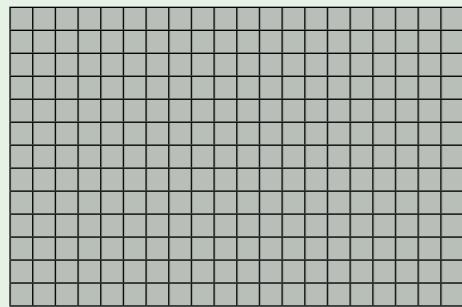
$>> 1.3 \times 10^9$ windows to classify



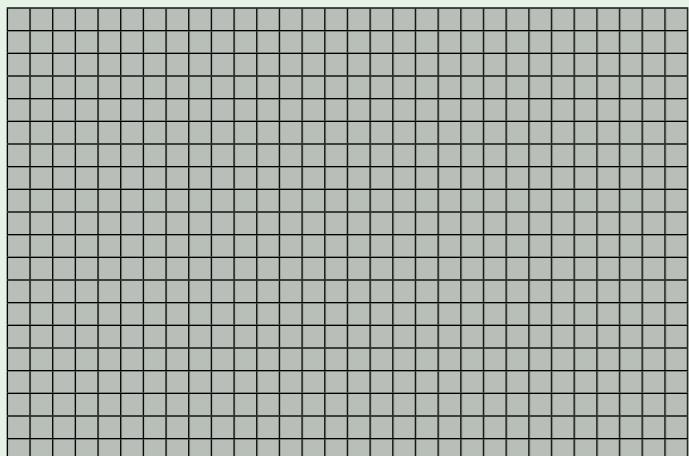
Detection



number of locations $p \sim 250,000$ per image



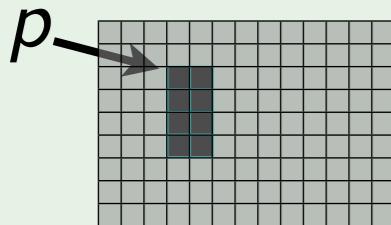
test set has ~ 5000 images



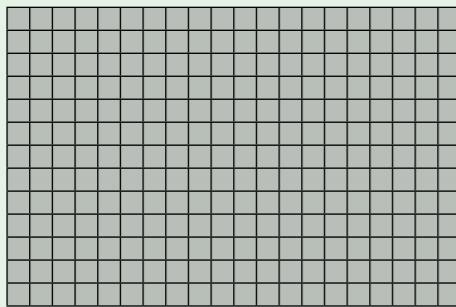
$>> 1.3 \times 10^9$ windows to classify

typically only $\sim 1,000$ true positive locations

Detection

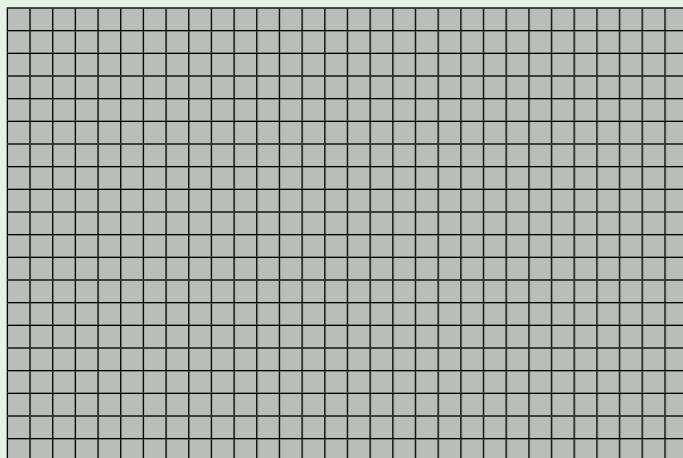


number of locations $p \sim 250,000$ per image



test set has ~ 5000 images

$>> 1.3 \times 10^9$ windows to classify

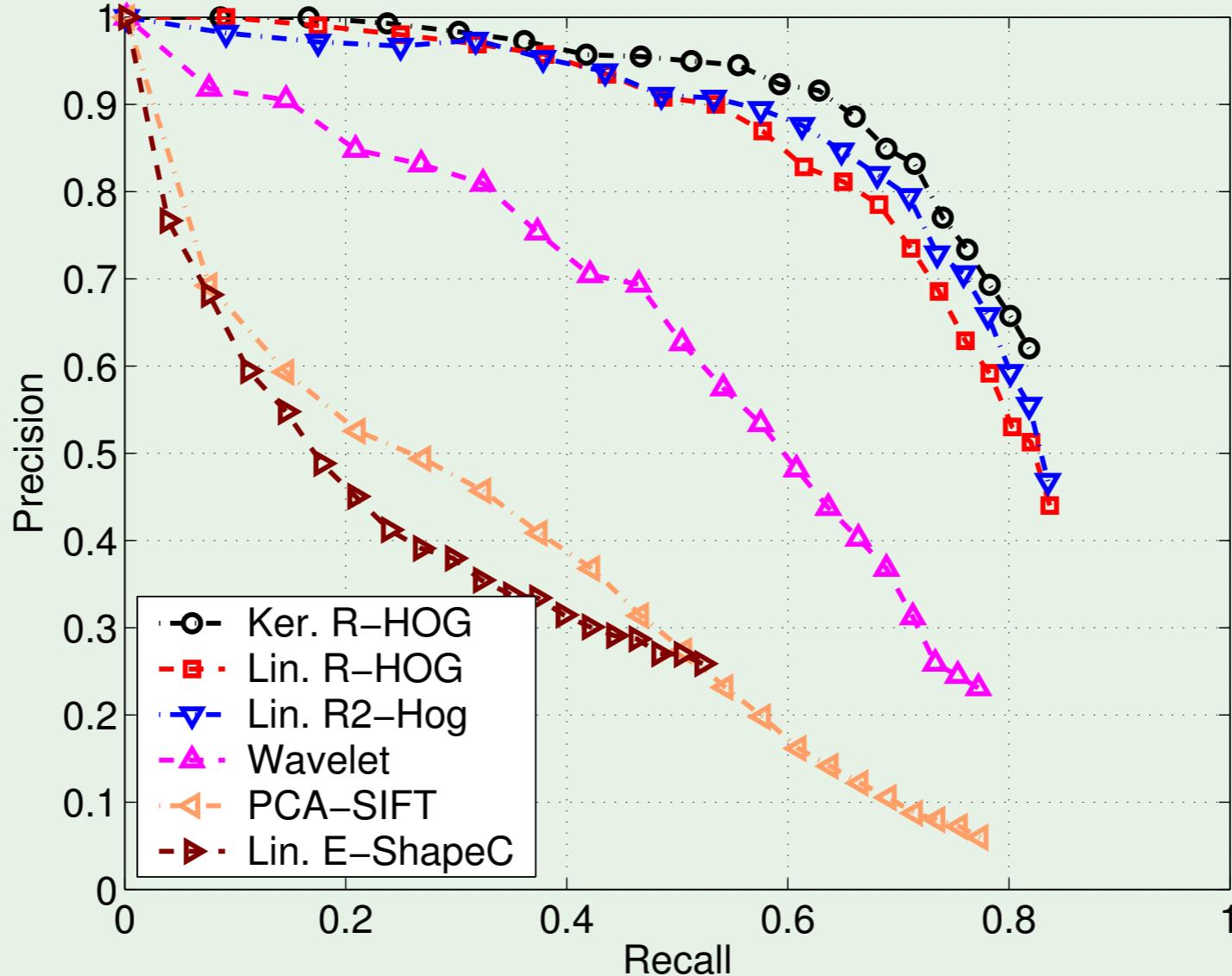


typically only $\sim 1,000$ true positive locations

Extremely unbalanced binary classification

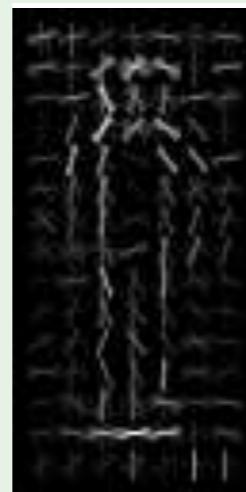
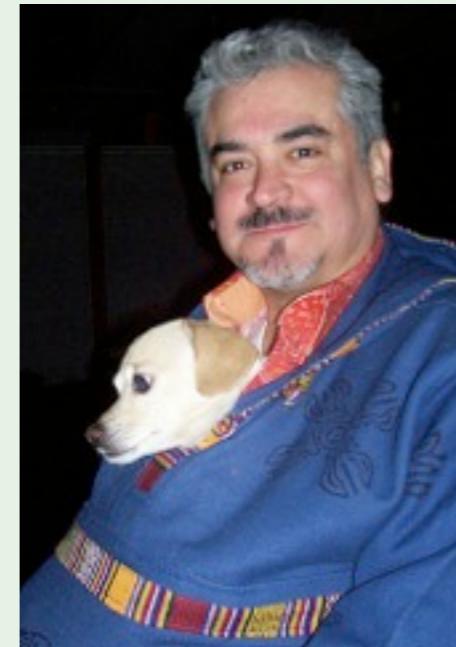
Dalal & Triggs detector on INRIA

Recall–Precision -- different descriptors on INRIA static person database



- AP = 75%
 - (79% in my implementation)
- 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

Combine with modern features and machine learning

Part-based models

- Parts — local appearance templates
- “Springs” — spatial connections between parts (geom. prior)

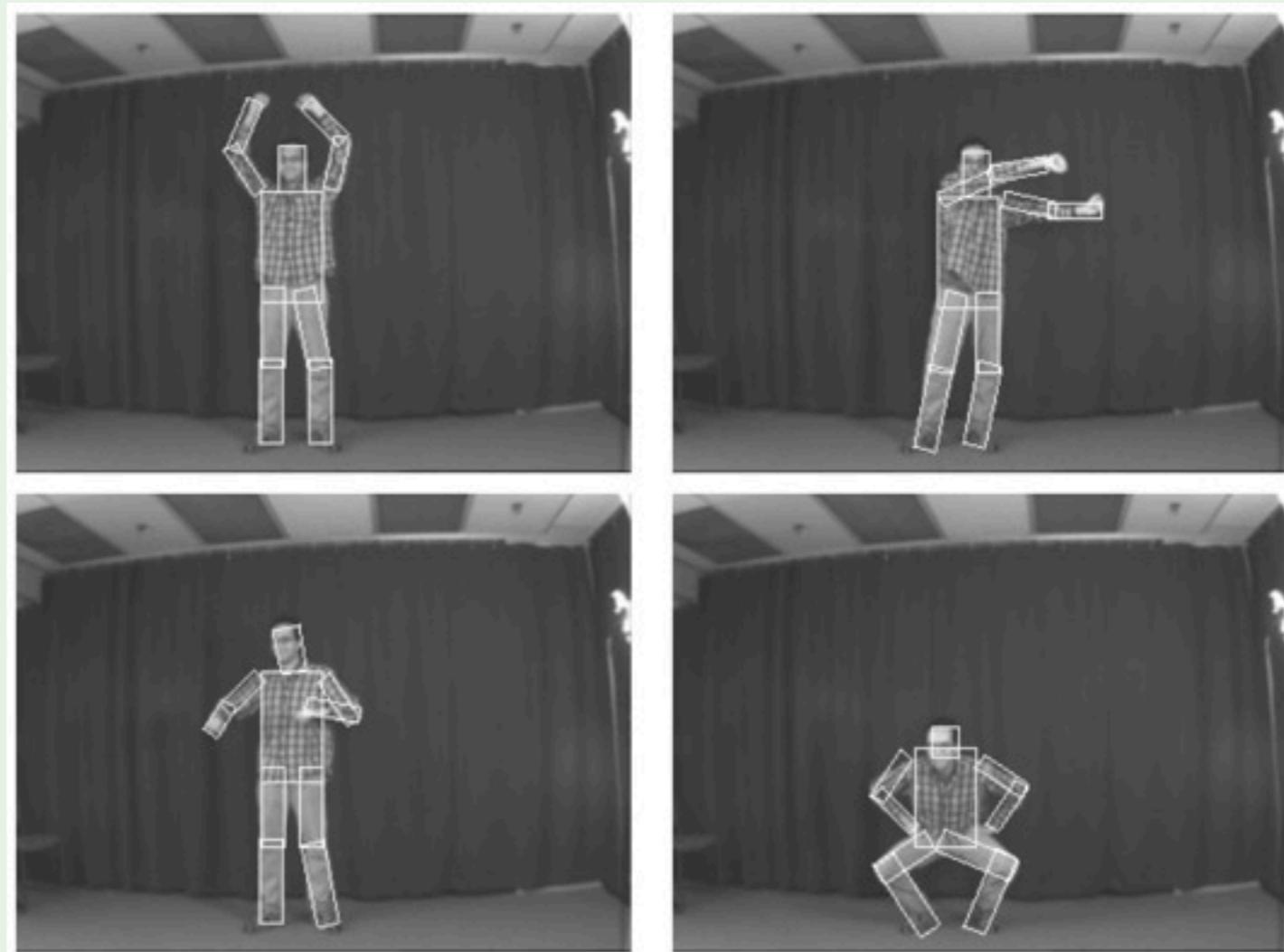
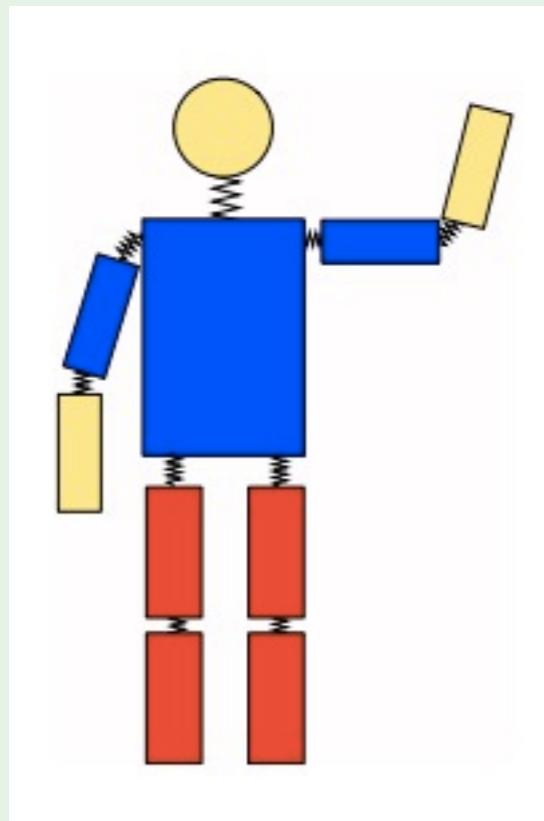
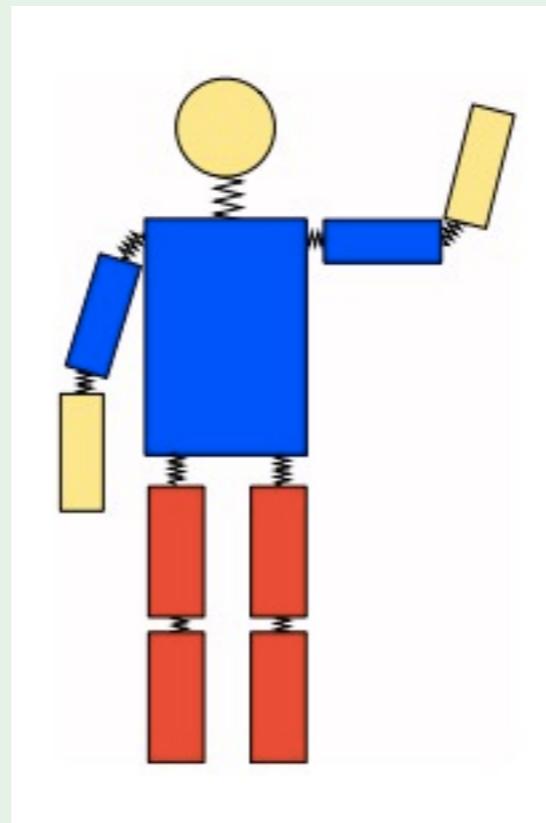


Image: [Felzenszwalb and Huttenlocher 05]

Part-based models

- Local appearance is easier to model than the global appearance
 - Training data shared across deformations
 - “part” can be local or global depending on resolution
- Generalizes to previously unseen configurations

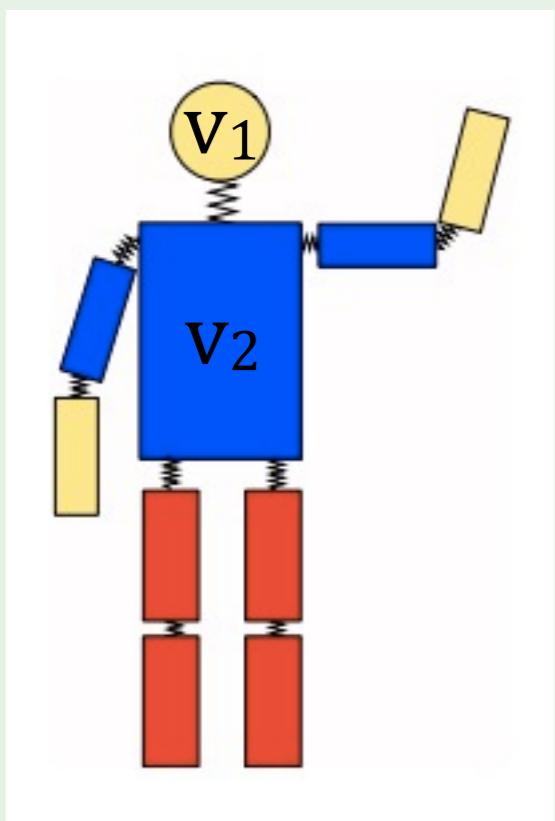


General formulation

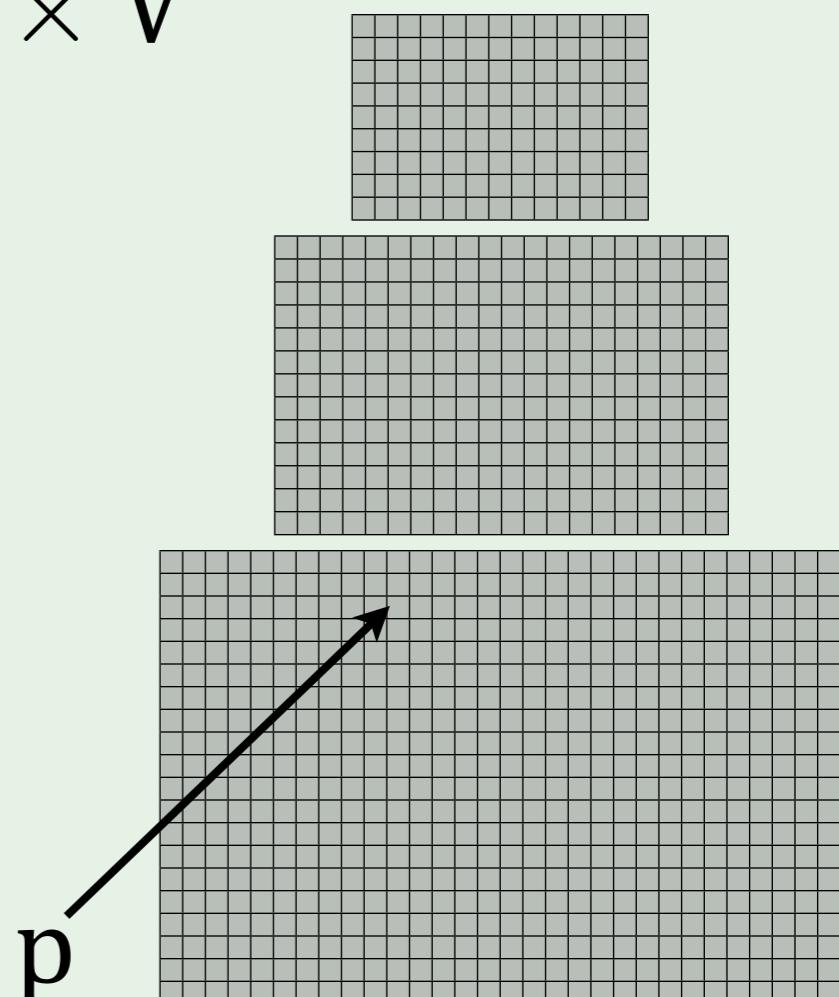
$$G = (V, E)$$

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

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



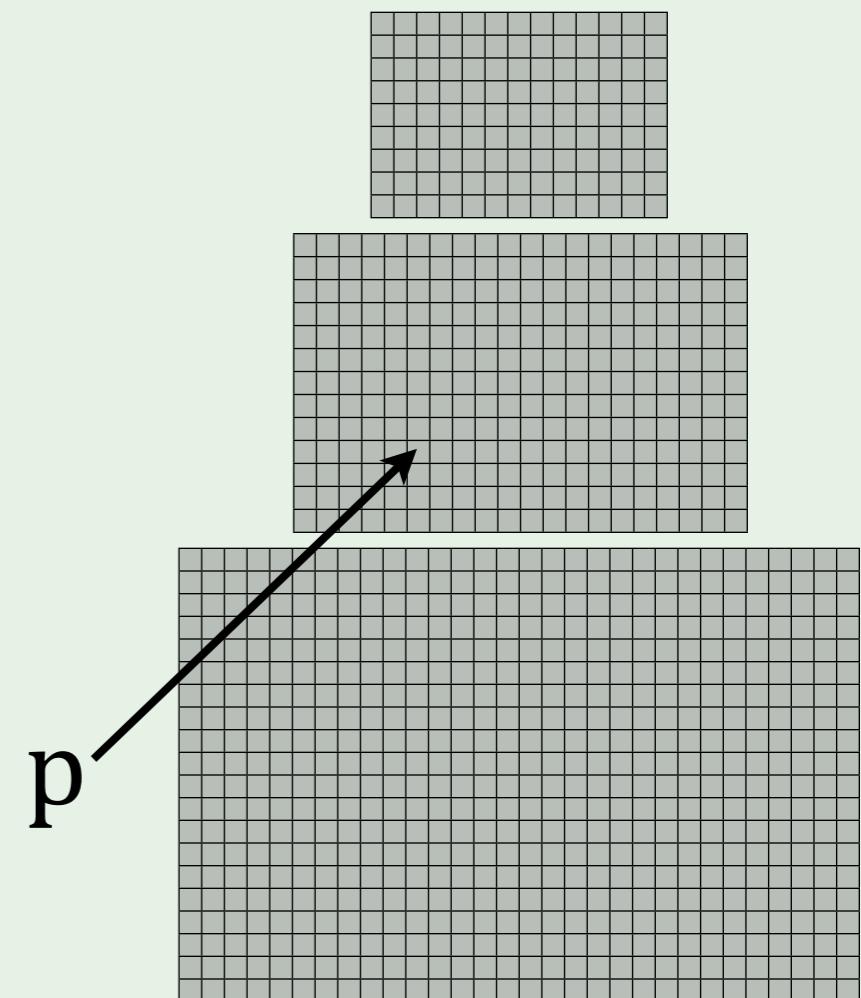
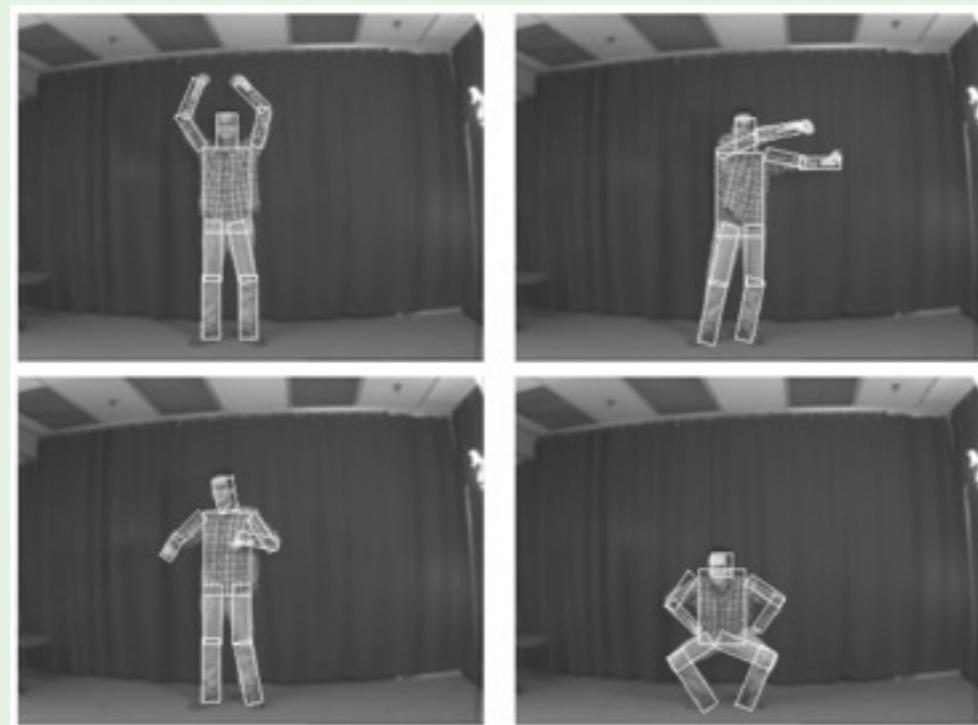
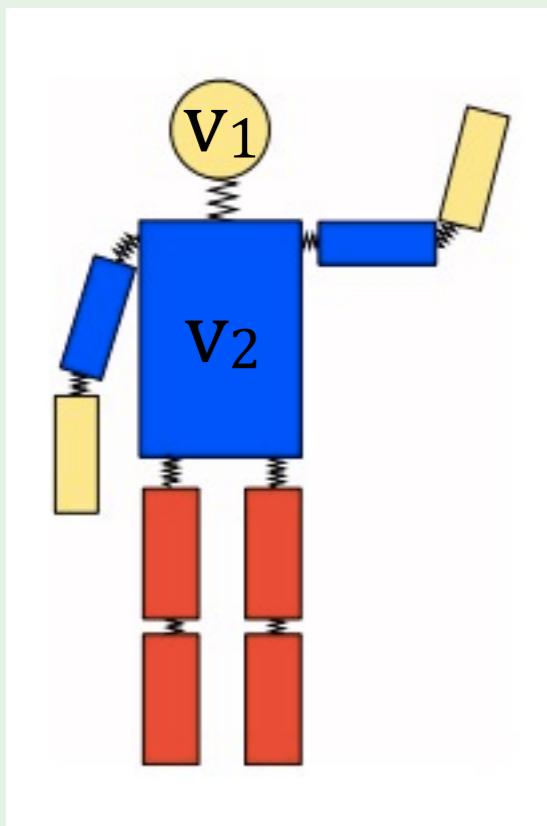
part locations in the image
(or feature pyramid)



Part configuration score function

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

spring costs
Part match scores



Highest scoring configurations

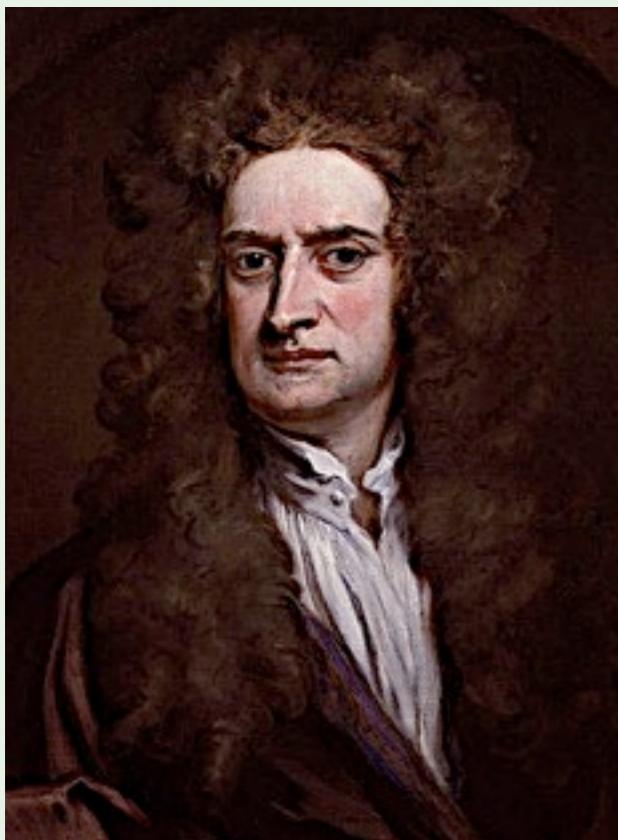
Part configuration score function

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

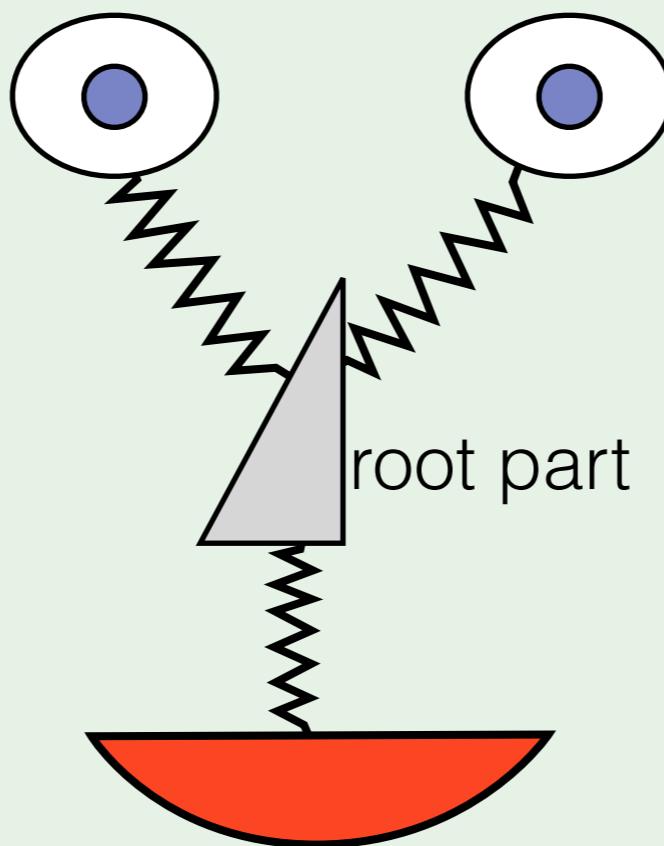
spring costs
Part match scores

- Objective: maximize score over p_1, \dots, p_n
- h^n configurations! ($h = |P|$, about 250,000)
- Dynamic programming
 - If $G = (V, E)$ is a tree, $O(nh^2)$ general algorithm
 - ▶ $O(nh)$ with some restrictions on d_{ij}

Star-structured deformable part models



test image

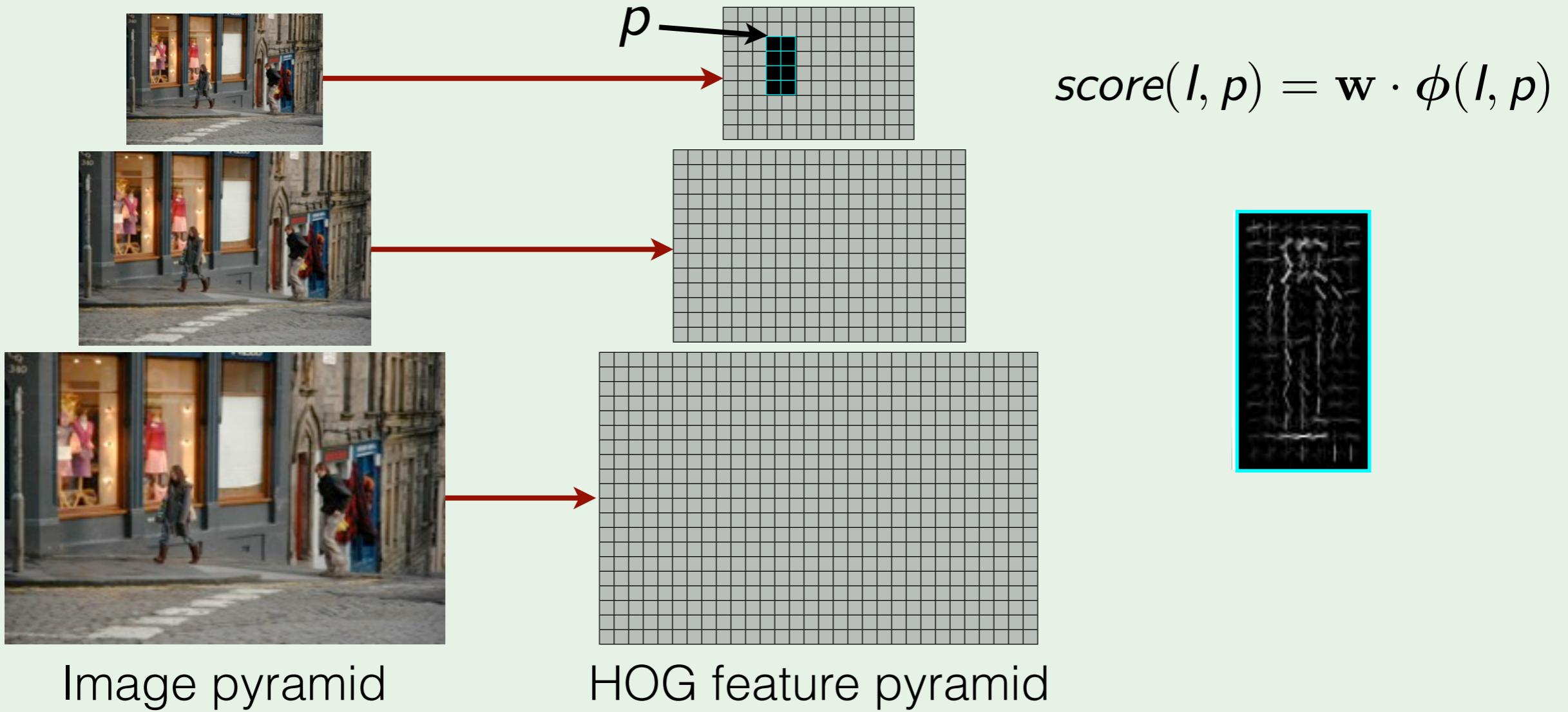


“star” model



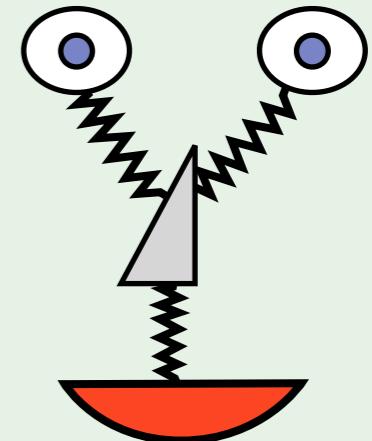
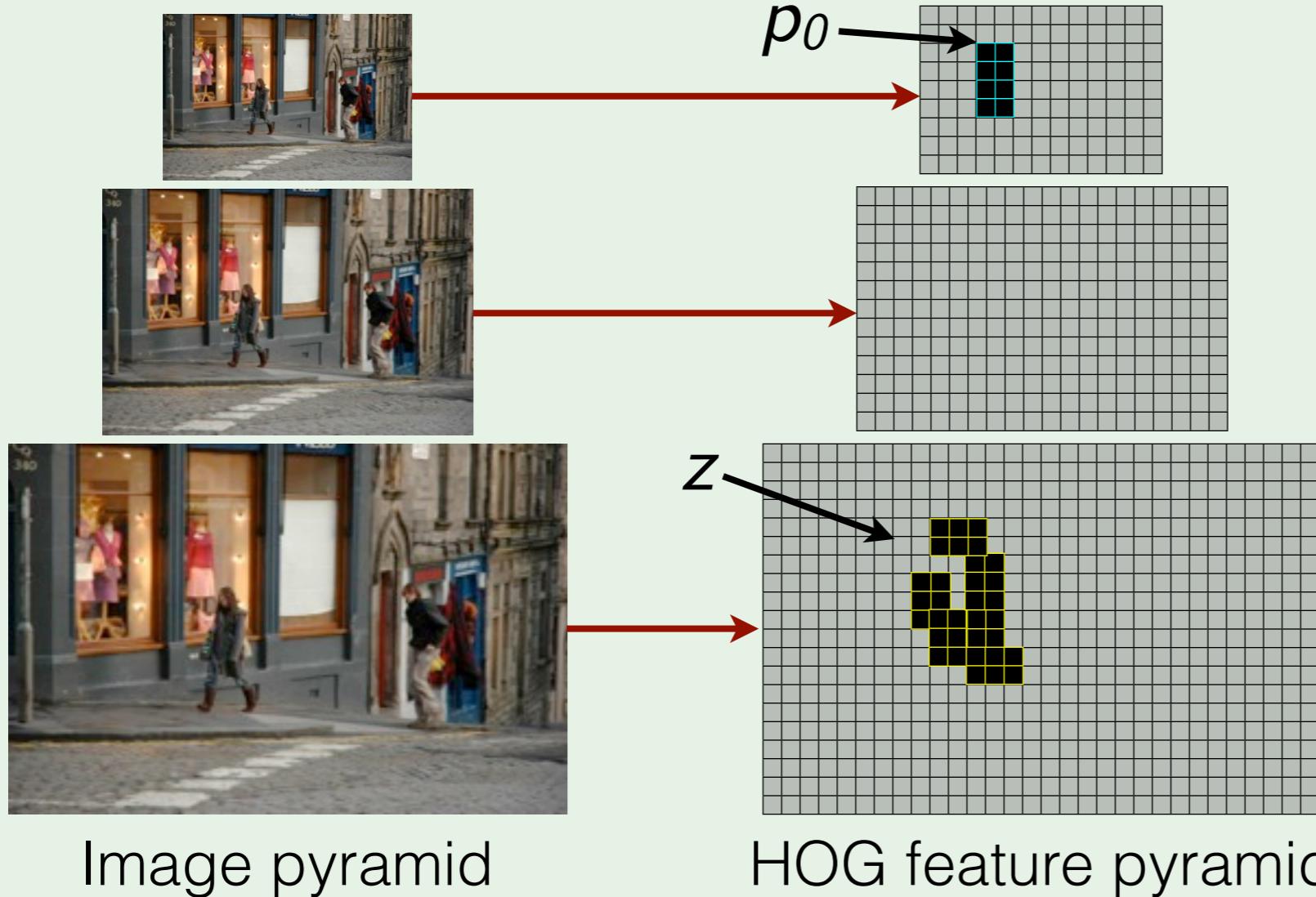
detection

Recall the Dalal & Triggs detector

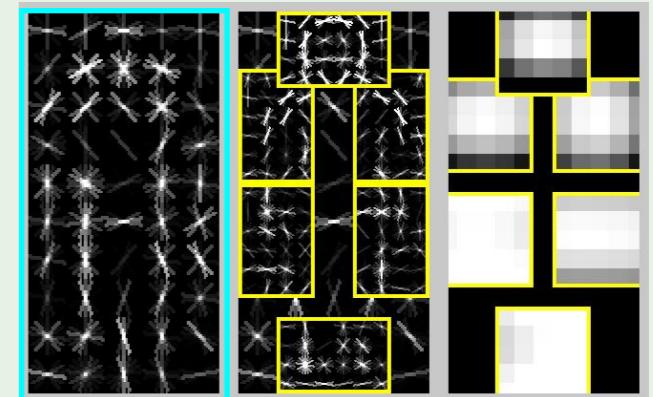


- HOG feature pyramid
- Linear filter / sliding-window detector
- SVM training to learn parameters w

D&T + parts



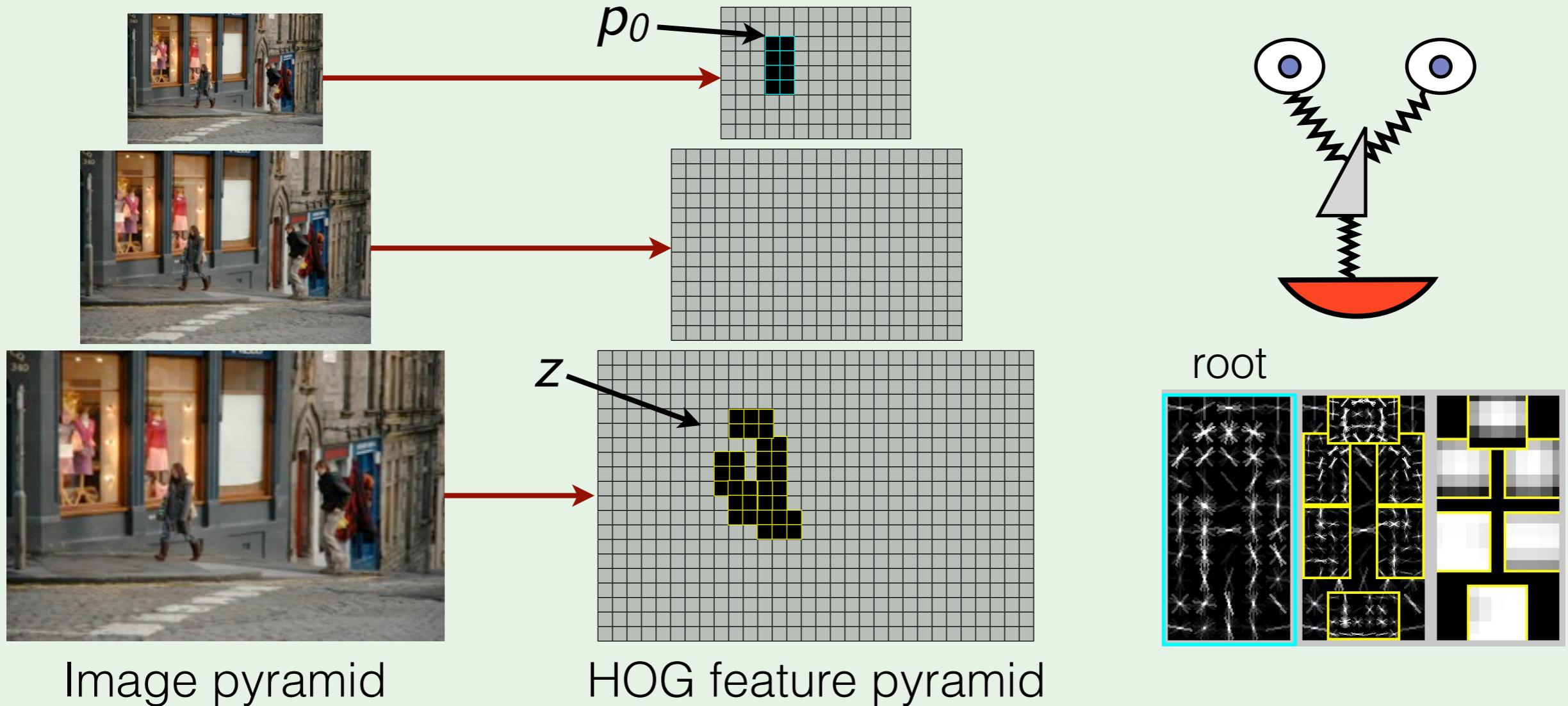
root



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

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

Sliding window DPM score function

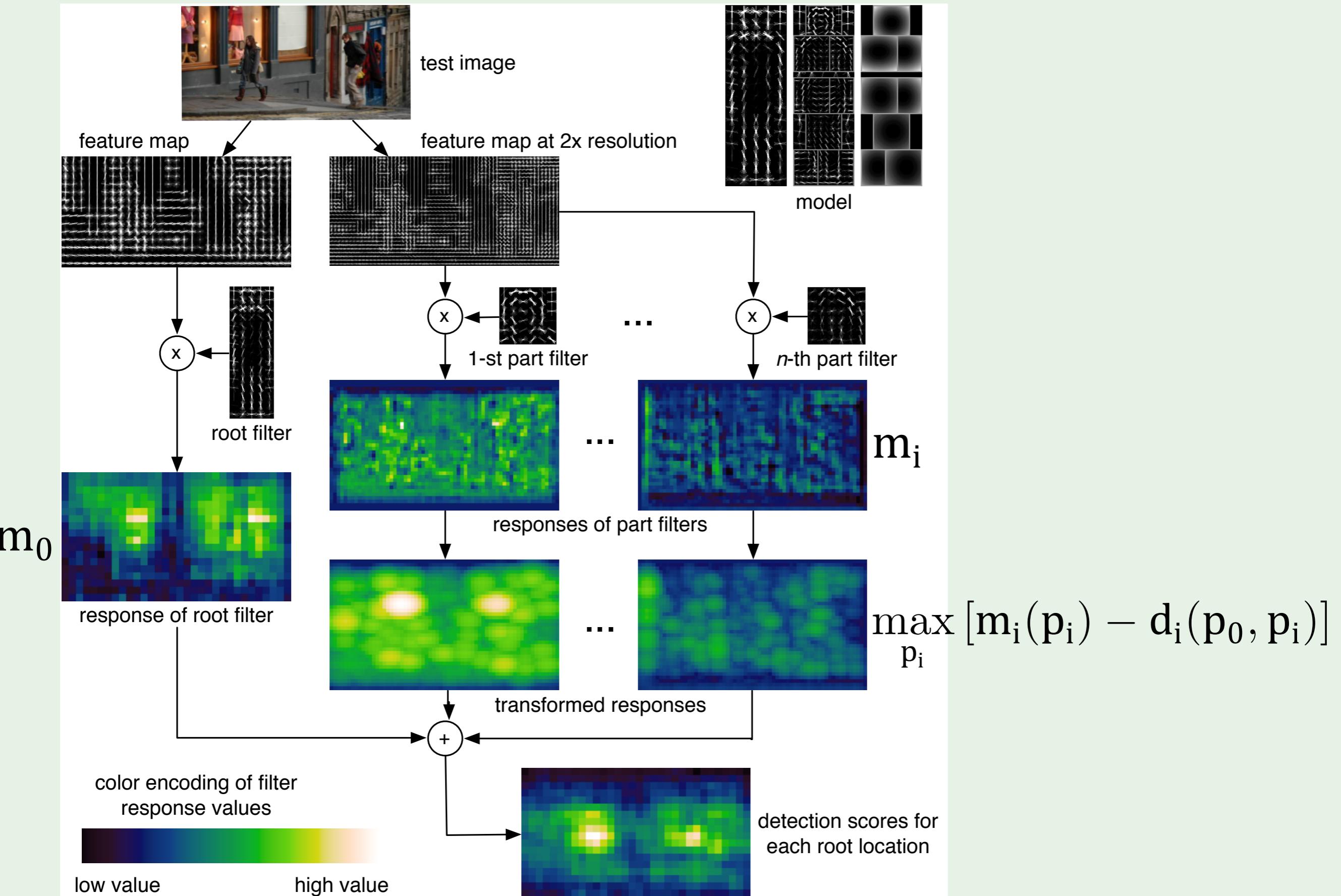


$$z = (p_1, \dots, p_n)$$

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

Filter scores Spring costs

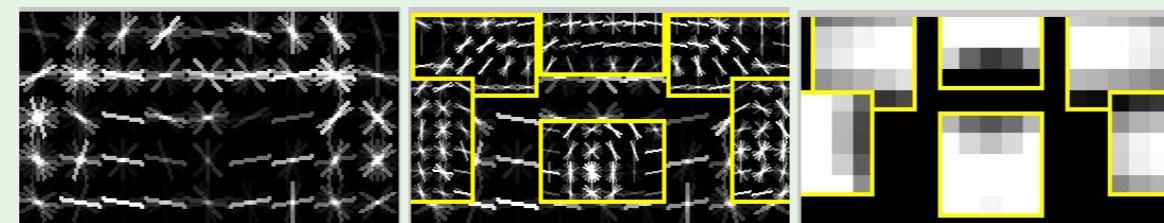
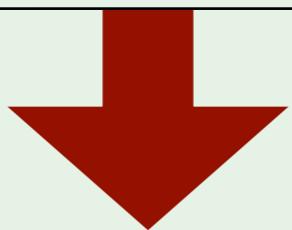
Detection in a slide



What are the parts?



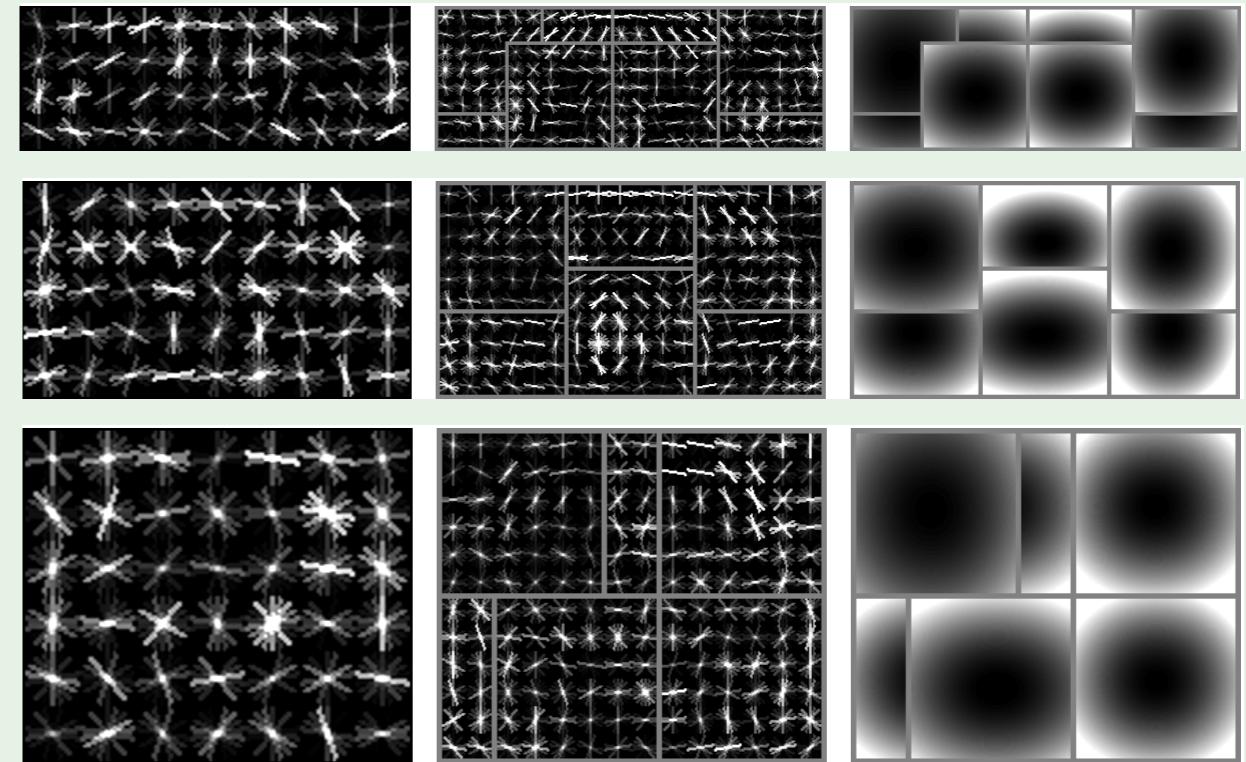
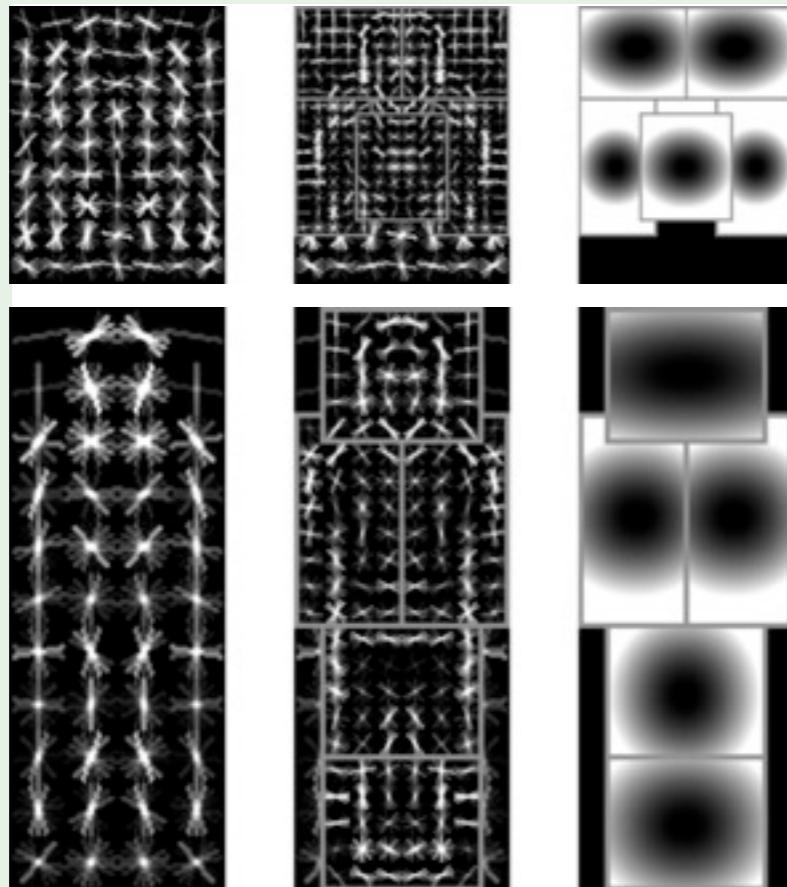
Aspect soup



General philosophy: enrich models to better represent the data

Mixture models

Data driven: aspect, occlusion modes, subclasses

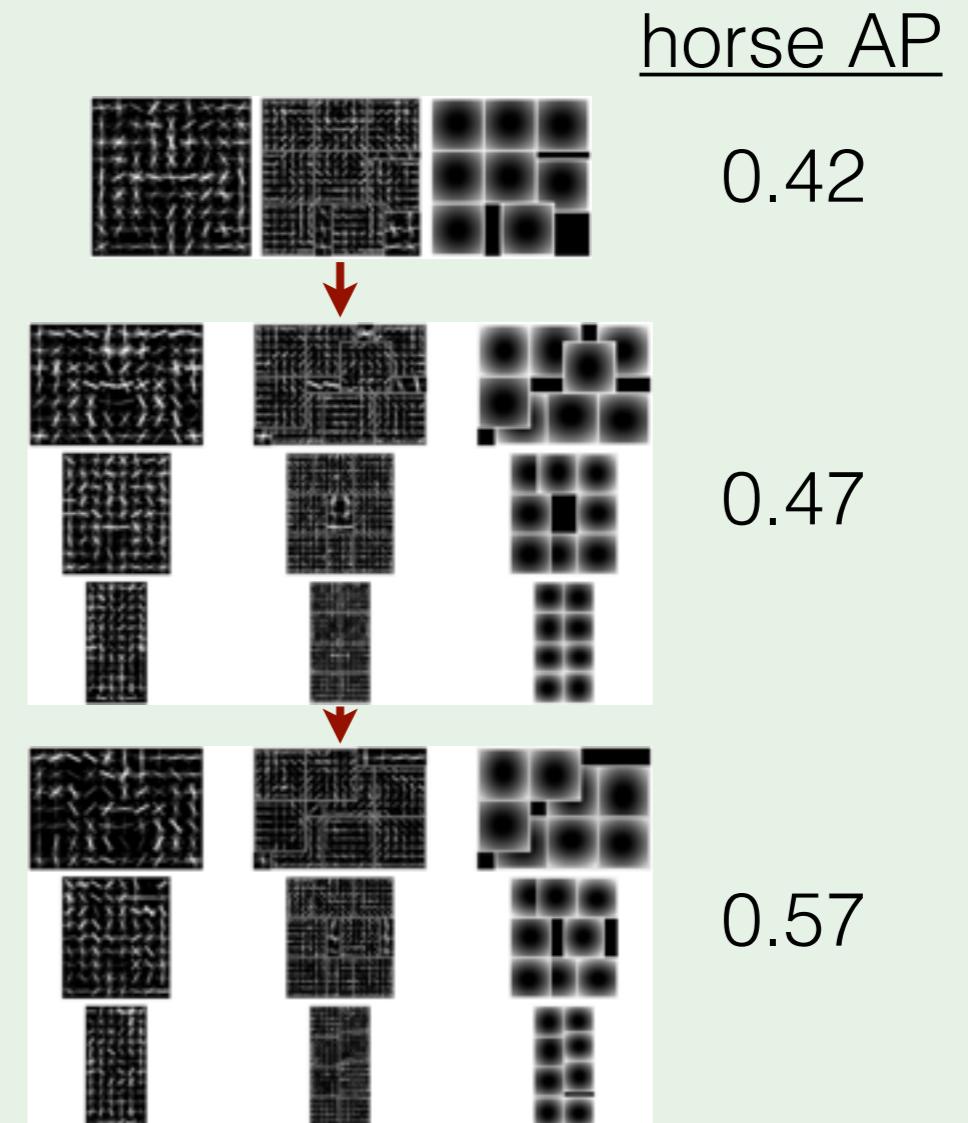
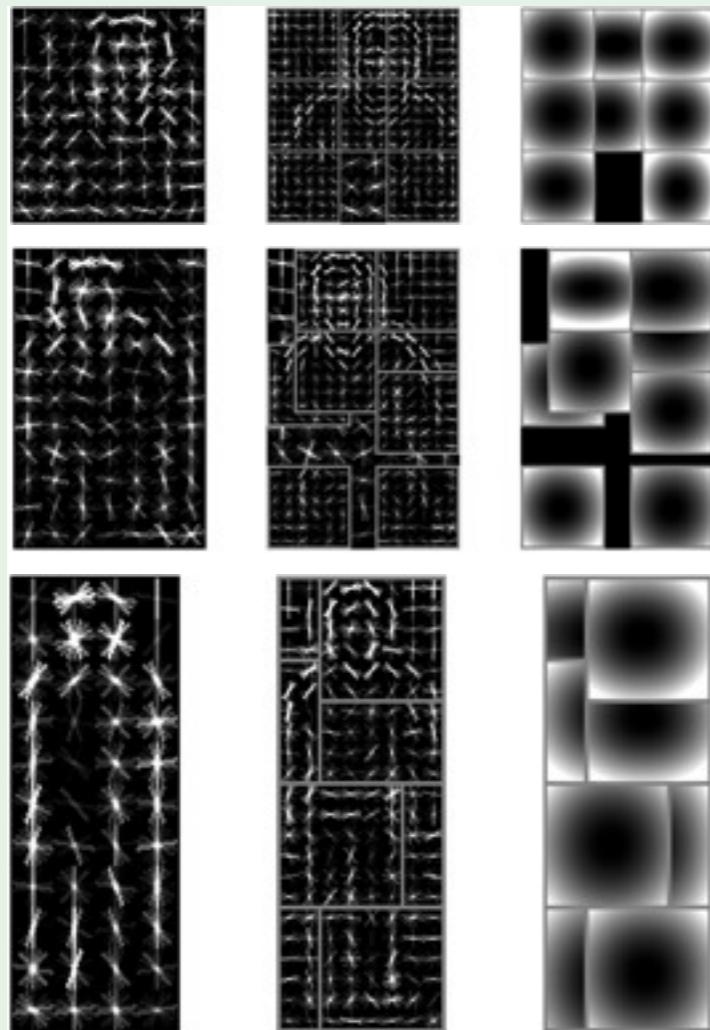


FMR CVPR '08: AP = 0.27 (person)

FGMR PAMI '10: AP = 0.36 (person)

Latent orientation

Unsupervised left/right orientation discovery

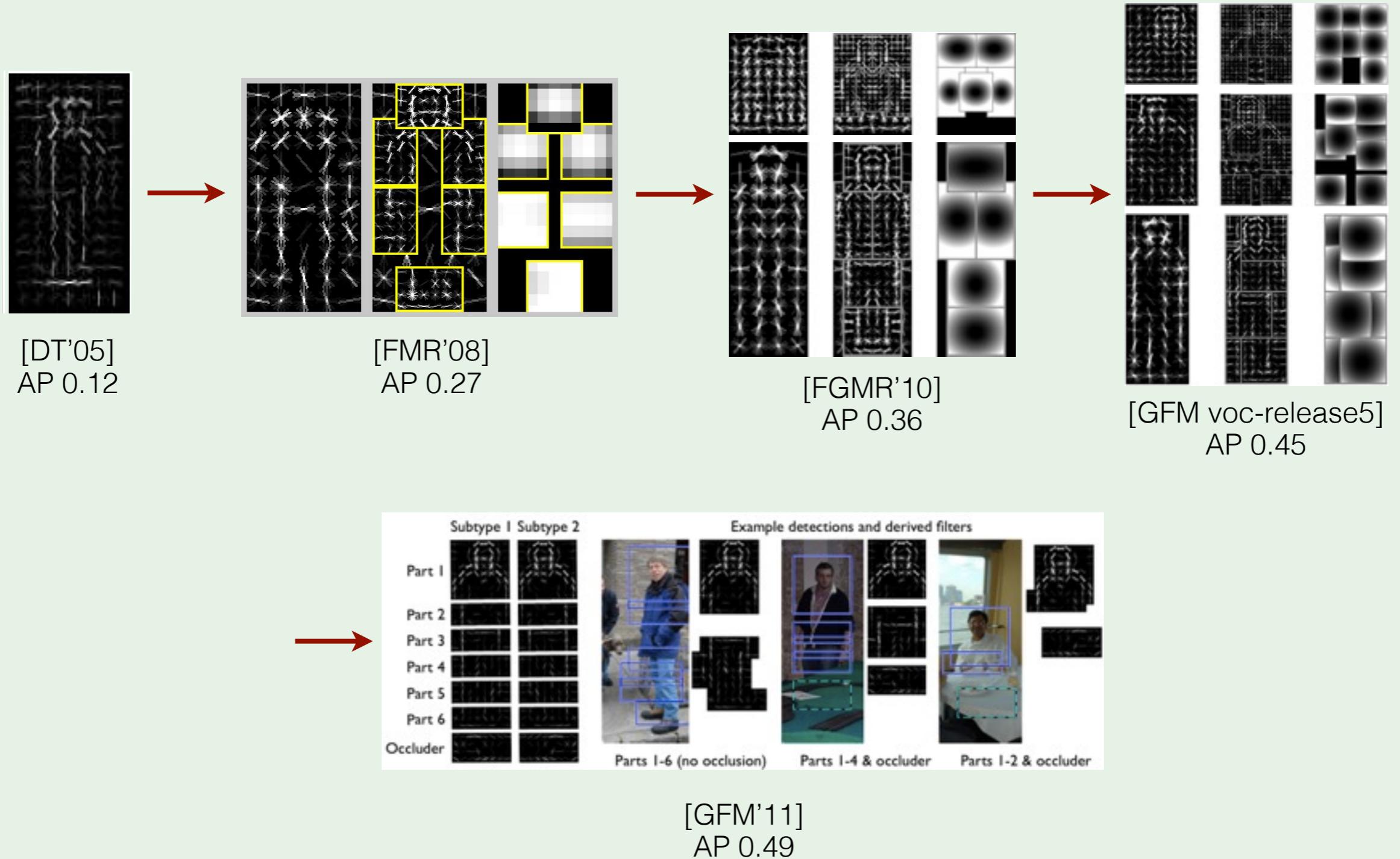


FGMR PAMI '10: AP = 0.36 (person)

voc-release5: AP = 0.45 (person)

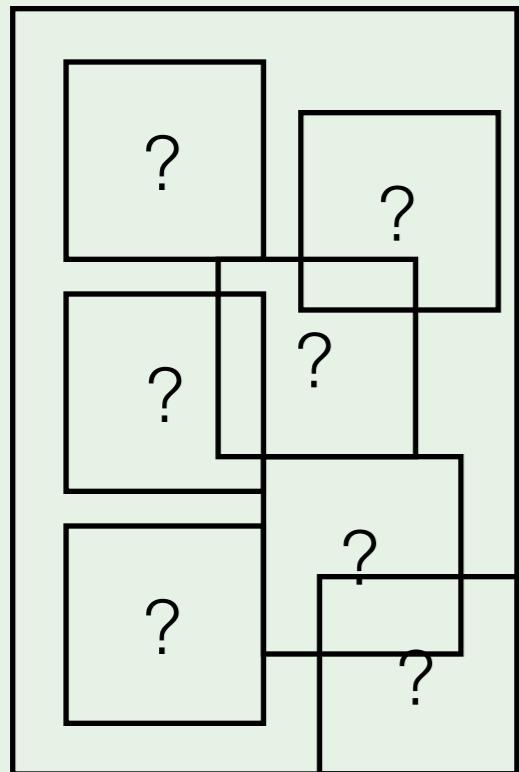
Publicly available code for the whole system: current voc-release5

Summary of results

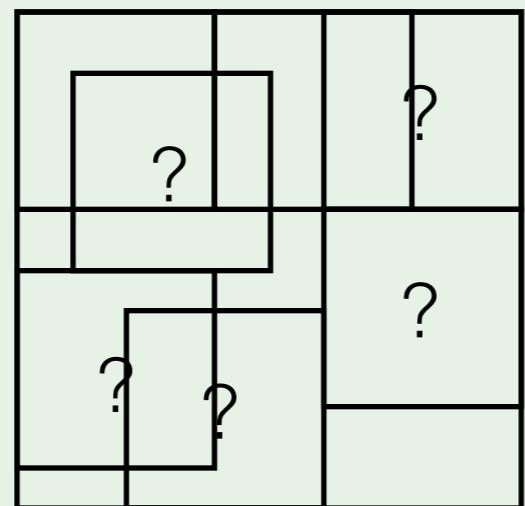


Part 2: DPM parameter learning

fixed model *structure*



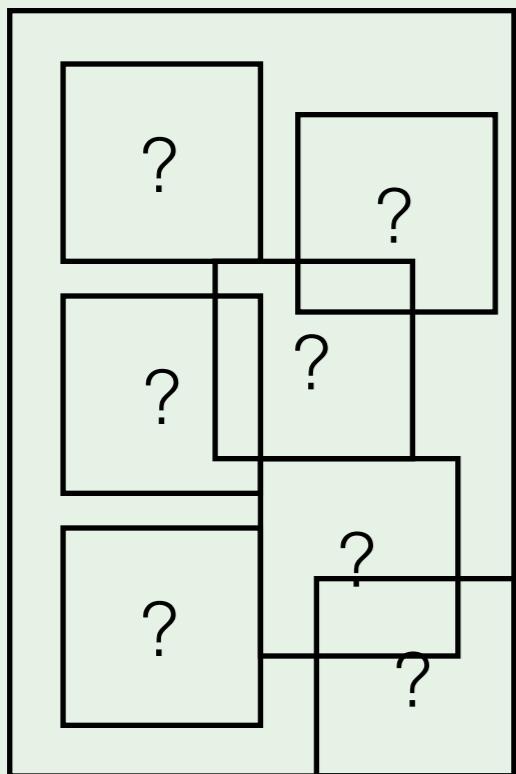
component 1



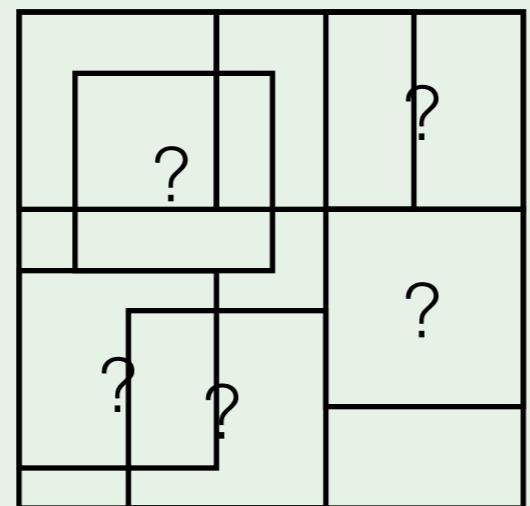
component 2

Part 2: DPM parameter learning

fixed model *structure*



component 1



component 2

training images

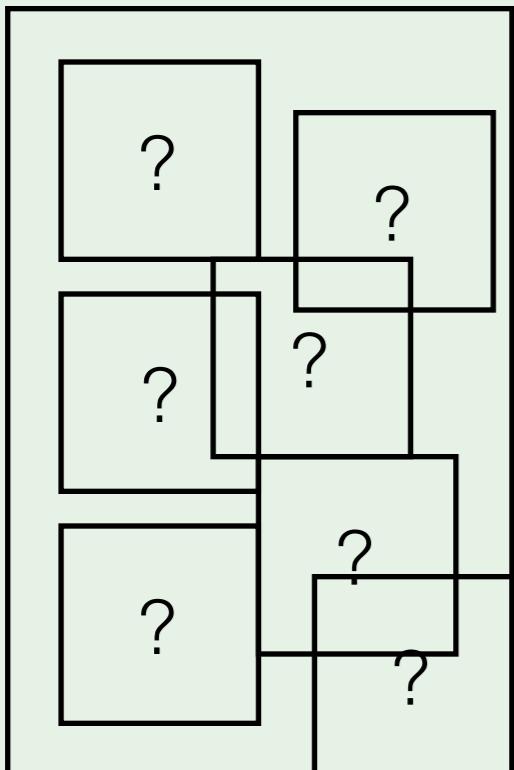


y

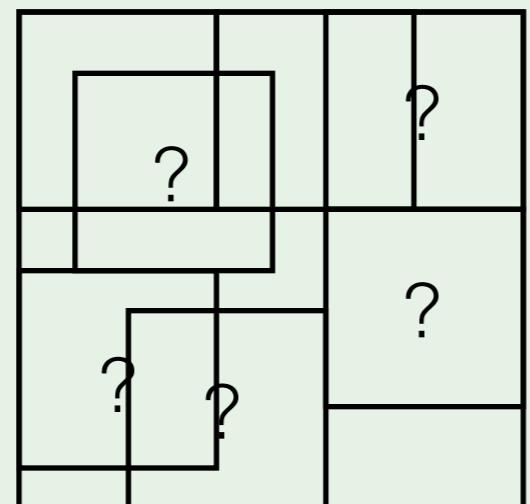
+1

Part 2: DPM parameter learning

fixed model *structure*



component 1



component 2

training images



y

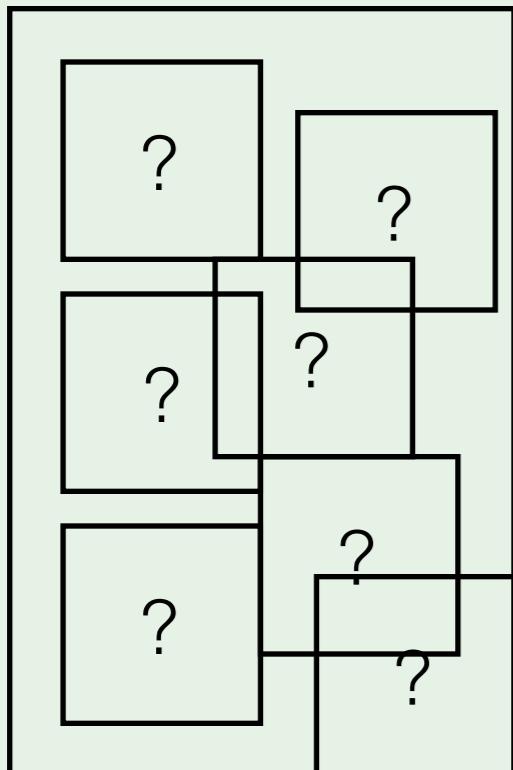
+1



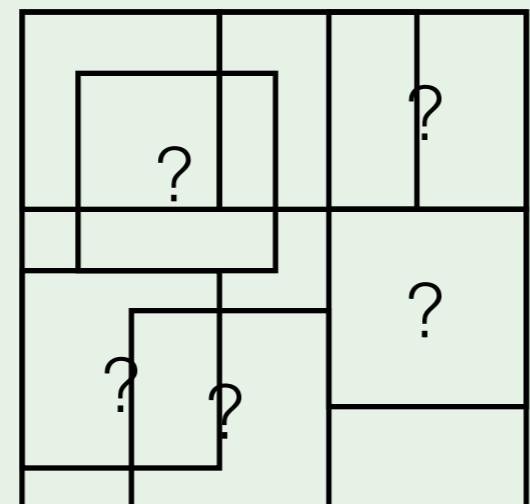
-1

Part 2: DPM parameter learning

fixed model *structure*



component 1



component 2

training images

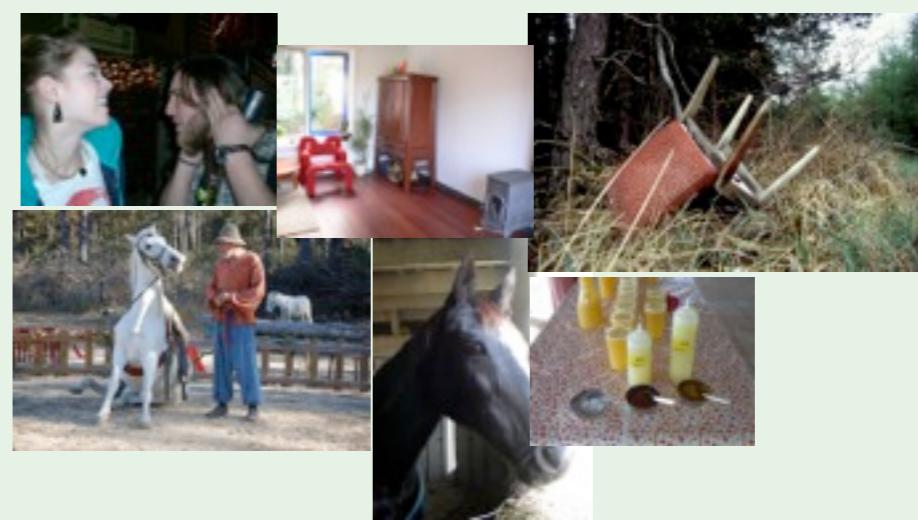


y

+1

Parameters to learn:

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



-1

Linear parameterization

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

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

Filter scores Spring costs

Filter scores

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

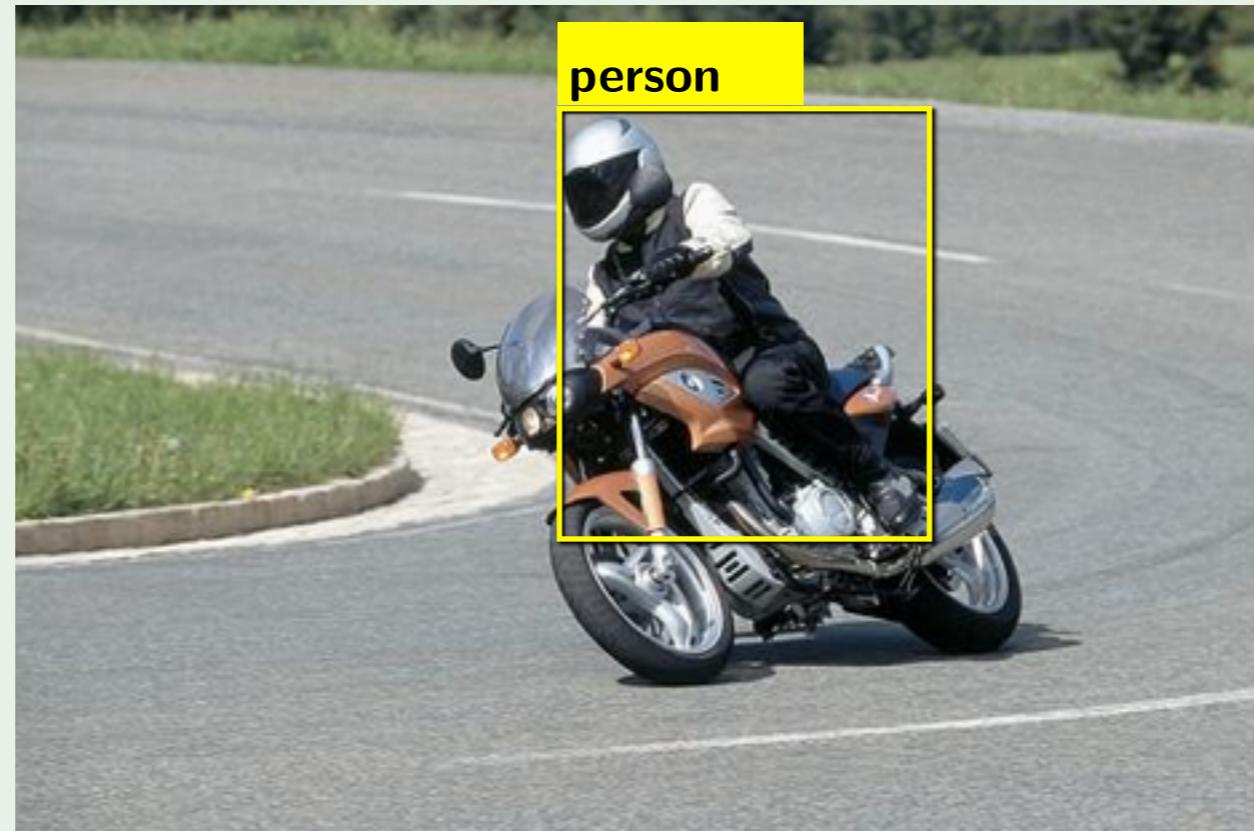
Spring costs

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

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

Positive examples ($y = +1$)

x specifies an image and bounding box



We want

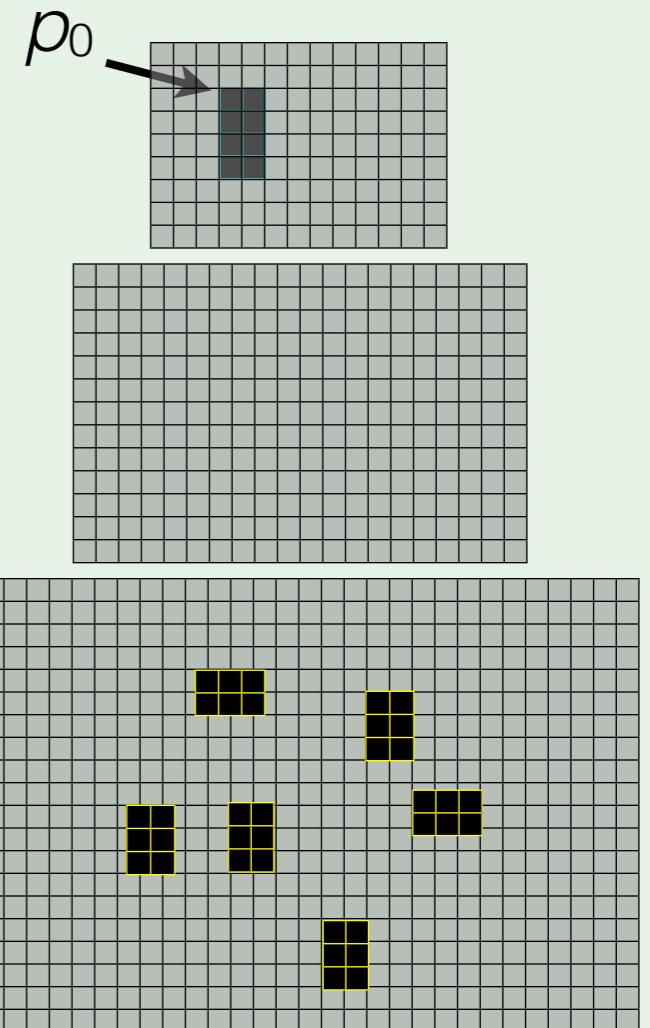
$$f_{\mathbf{w}}(x) = \max_{z \in Z(x)} \mathbf{w} \cdot \Phi(x, z)$$

to score $\geq +1$

$Z(x)$ 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

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

to score ≤ -1

$Z(x)$ 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*

*unless someone from google is here

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

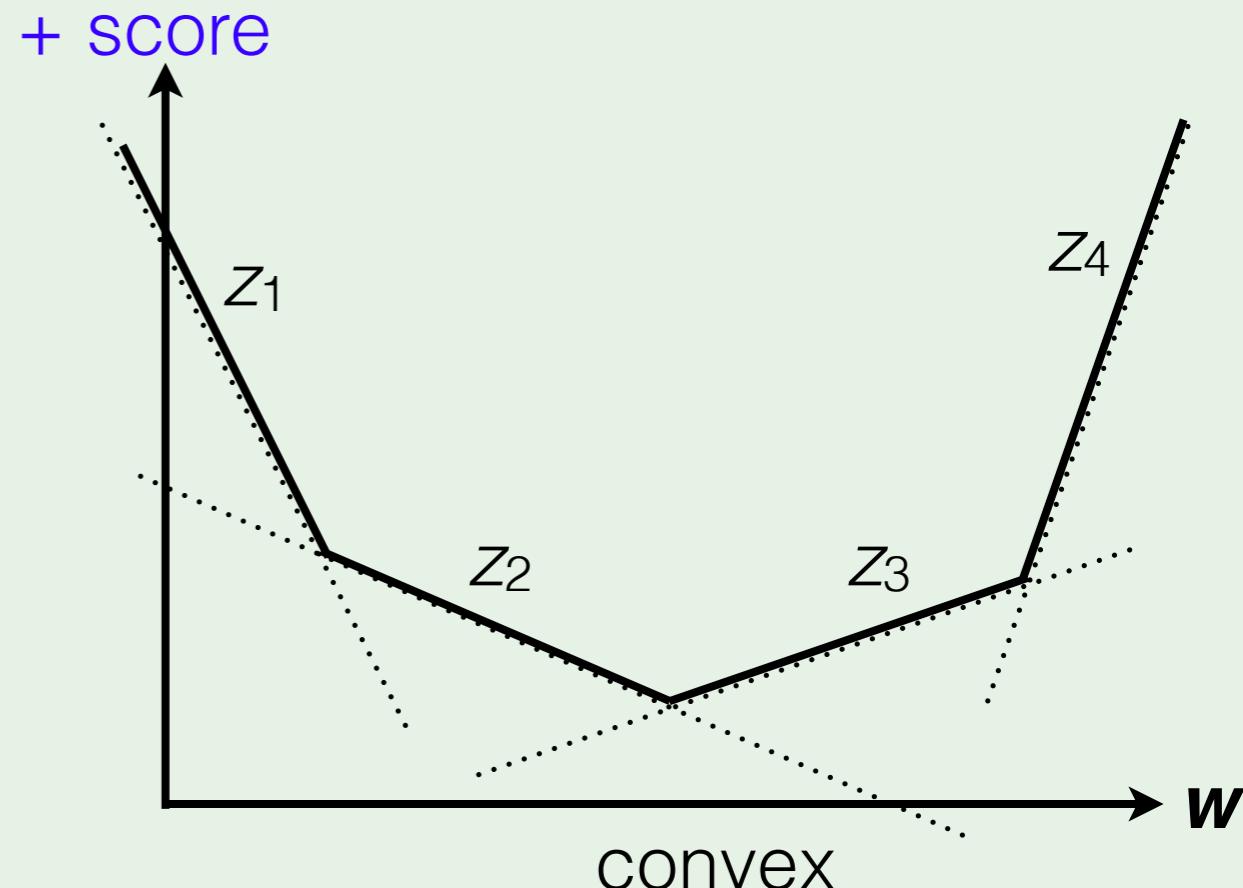
$$E(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \max\{0, 1 - y_i f_{\mathbf{w}}(x_i)\}$$

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

How we learn parameters: latent SVM

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

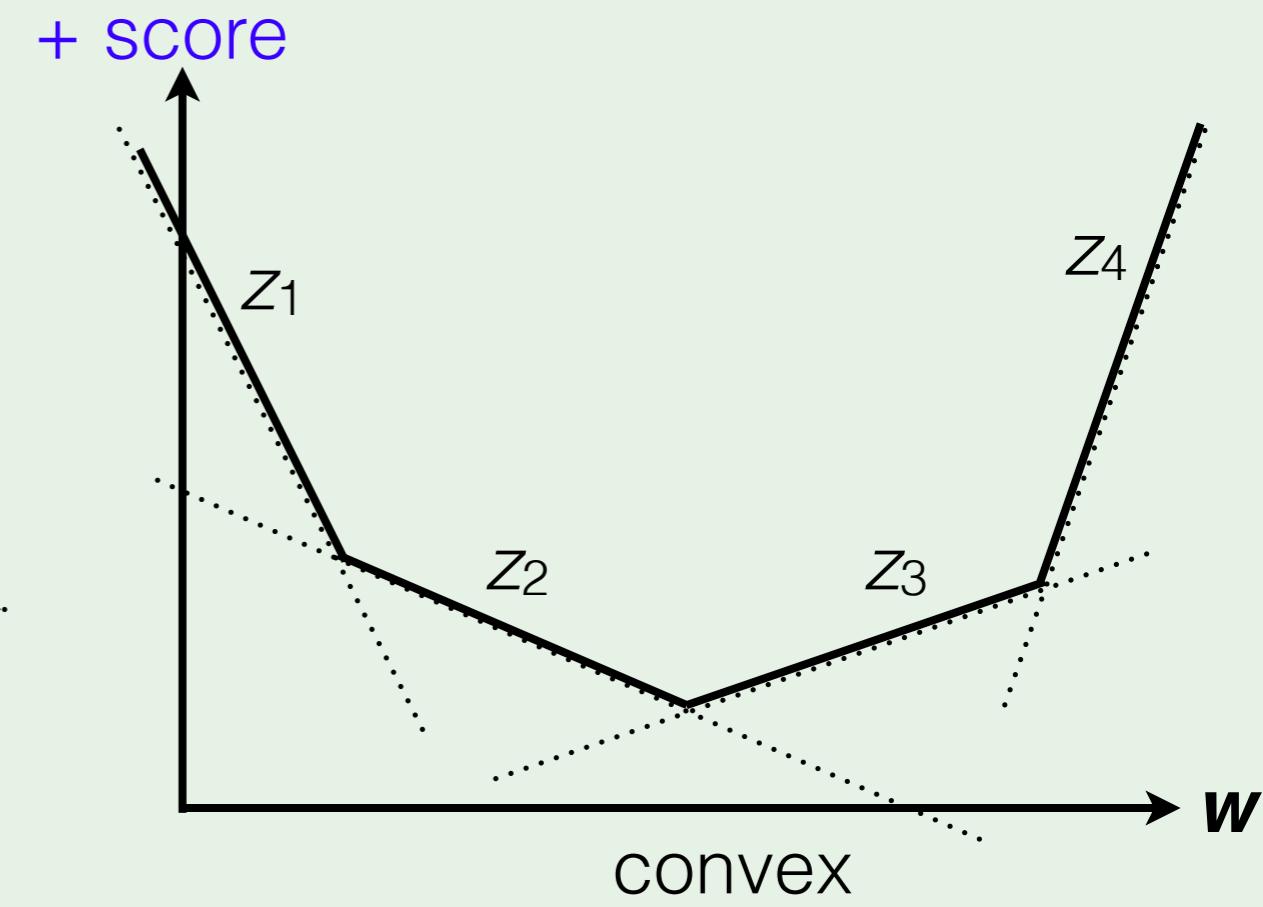
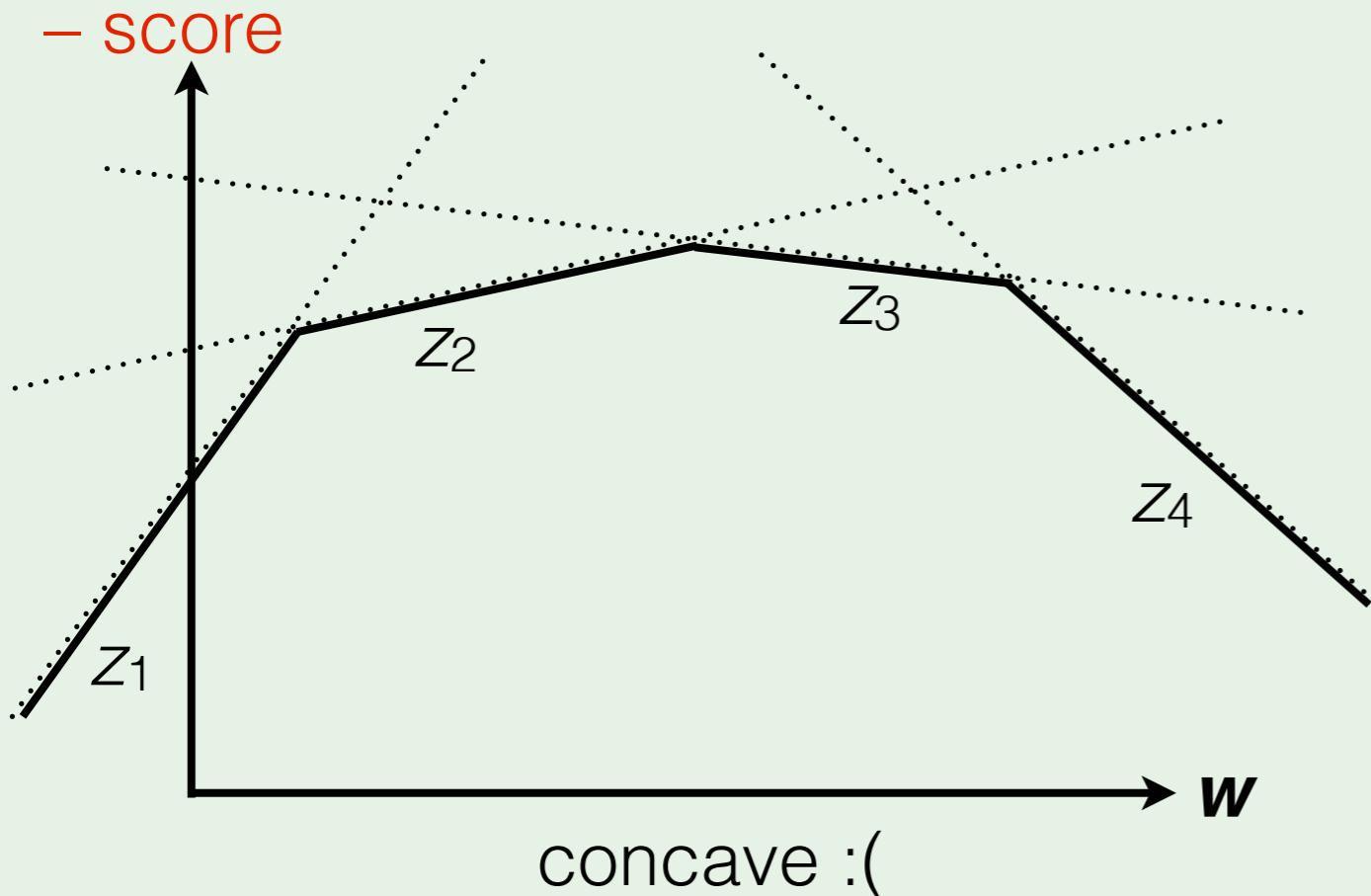
$$\begin{aligned} E(\mathbf{w}) = & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i \in P} \max\{0, 1 - \max_{z \in Z(x)} \mathbf{w} \cdot \Phi(x_i, z)\} \\ & + C \sum_{i \in N} \max\{0, 1 + \max_{z \in Z(x)} \mathbf{w} \cdot \Phi(x_i, z)\} \end{aligned}$$



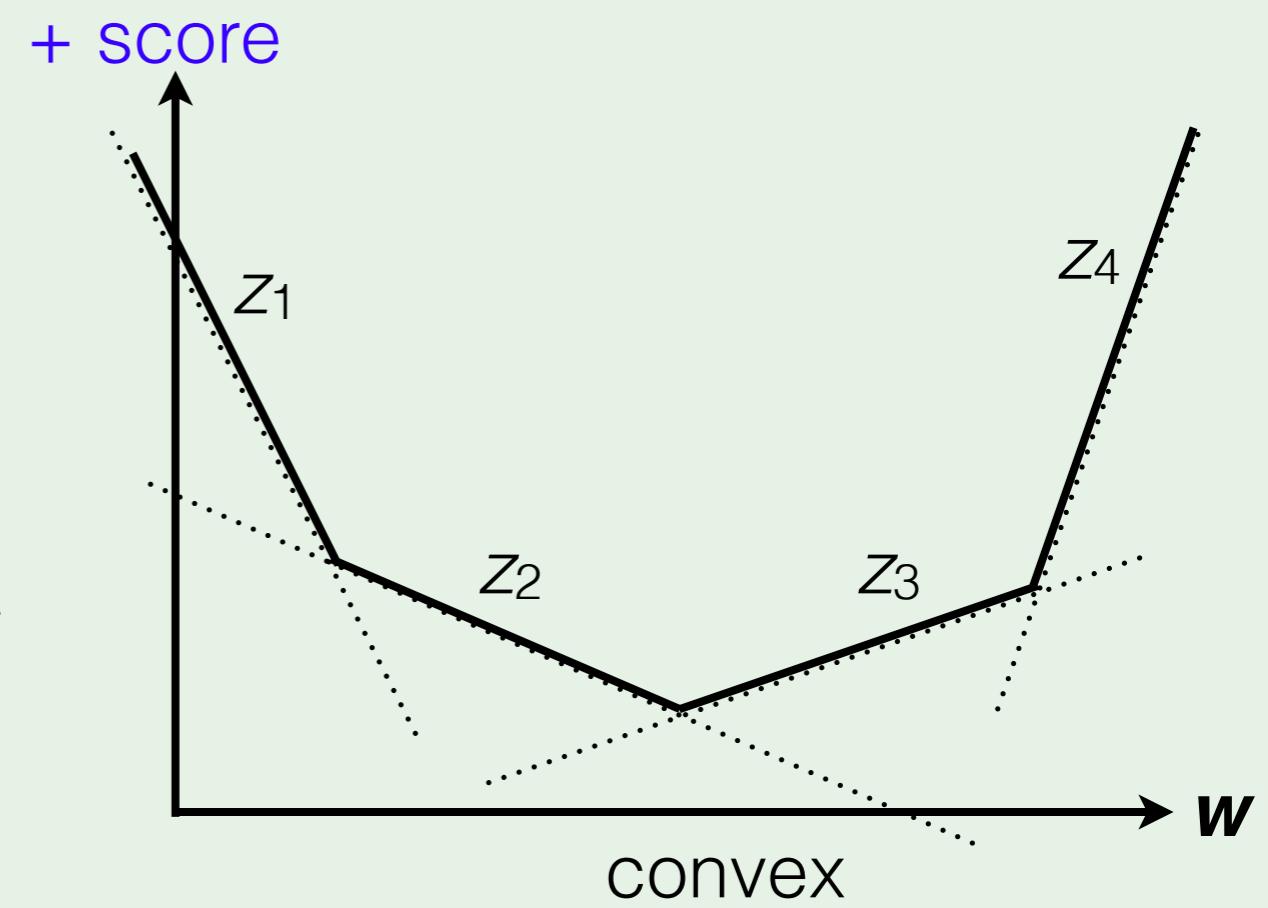
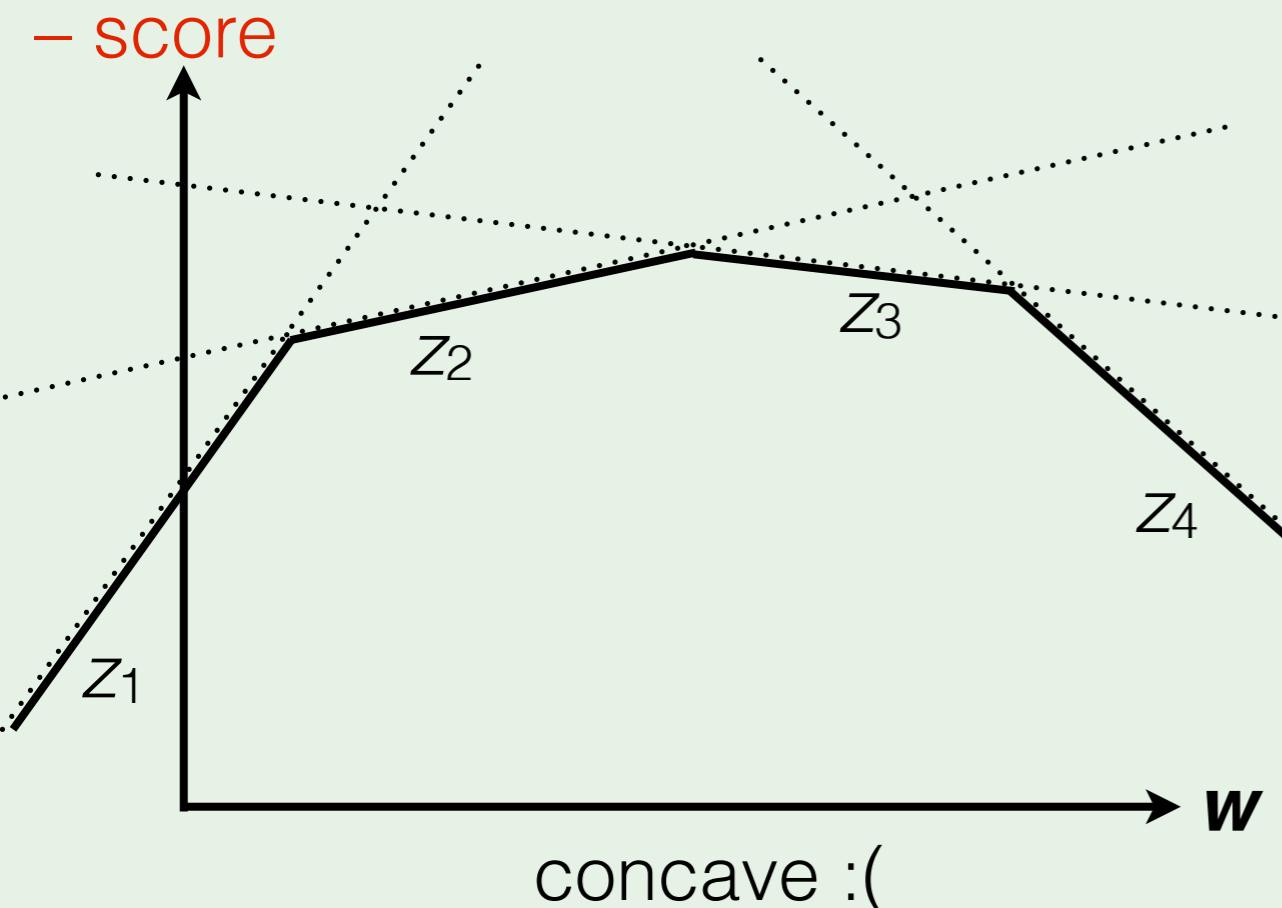
How we learn parameters: latent SVM

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

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



Observations

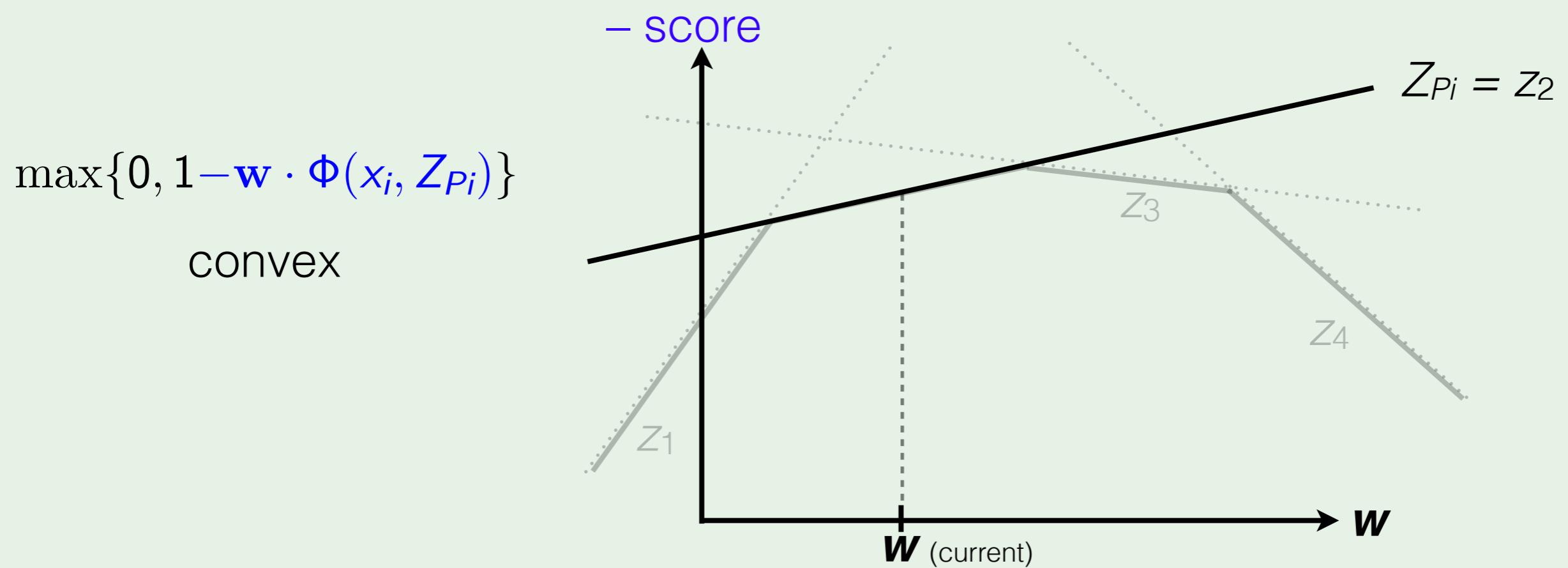
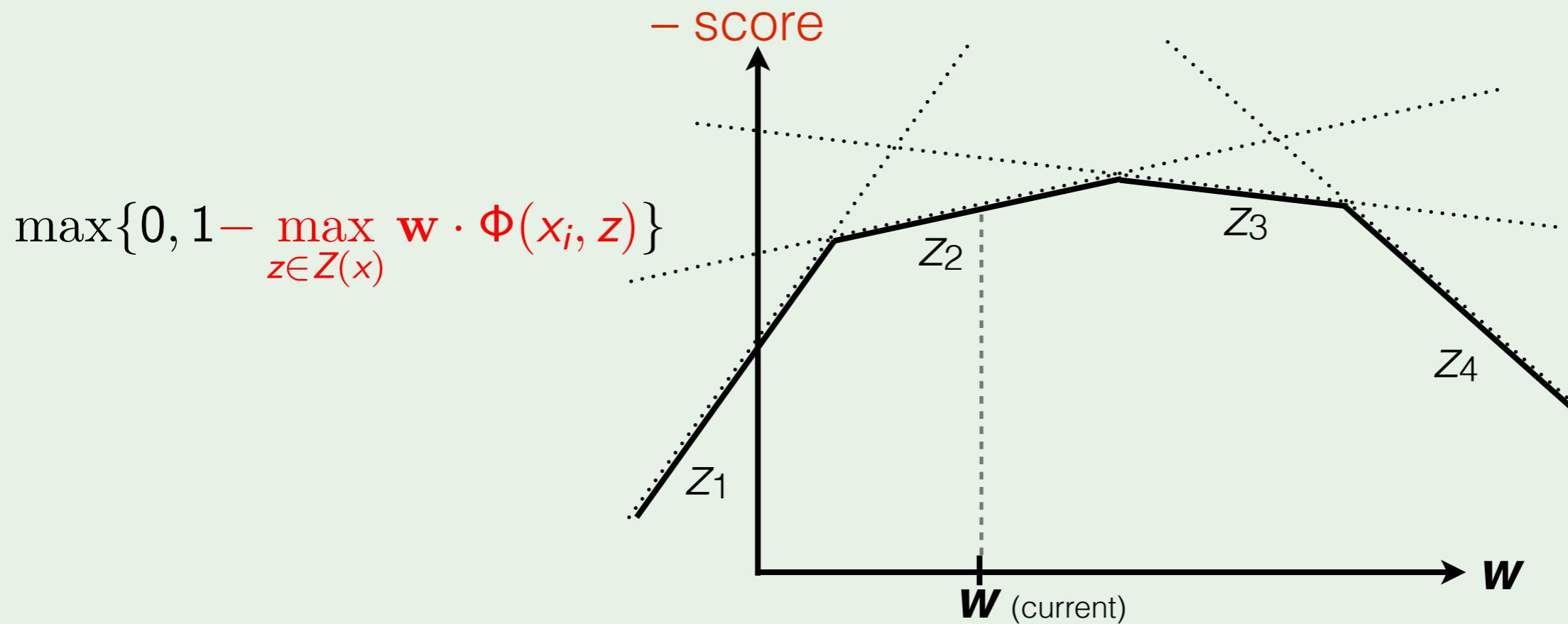


Latent SVM objective is convex in the negatives

but not in the positives

>> “semi-convex”

Convex upper bound on loss



Auxiliary objective

Let $Z_P = \{Z_{P1}, Z_{P2}, \dots\}$

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

Auxiliary objective

Let $Z_P = \{Z_{P1}, Z_{P2}, \dots\}$

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

Note that $E(\mathbf{w}, Z_p) \geq \min_{Z_P} E(\mathbf{w}, Z_P) = E(\mathbf{w})$

Auxiliary objective

Let $Z_P = \{Z_{P1}, Z_{P2}, \dots\}$

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

Note that $E(\mathbf{w}, Z_p) \geq \min_{Z_P} E(\mathbf{w}, Z_P) = E(\mathbf{w})$

and $\mathbf{w}^* = \min_{\mathbf{w}, Z_P} E(\mathbf{w}, Z_P) = \min_{\mathbf{w}} E(\mathbf{w})$

Auxiliary objective

$$\mathbf{w}^* = \min_{\mathbf{w}, Z_P} E(\mathbf{w}, Z_P) = \min_{\mathbf{w}} E(\mathbf{w})$$

This isn't any easier to optimize

Auxiliary objective

$$\mathbf{w}^* = \min_{\mathbf{w}, Z_P} E(\mathbf{w}, Z_P) = \min_{\mathbf{w}} E(\mathbf{w})$$

This isn't any easier to optimize

Find stationary point by coordinate descent on $E(\mathbf{w}, Z_P)$

Auxiliary objective

$$\mathbf{w}^* = \min_{\mathbf{w}, Z_P} E(\mathbf{w}, Z_P) = \min_{\mathbf{w}} E(\mathbf{w})$$

This isn't any easier to optimize

Find stationary point by coordinate descent on $E(\mathbf{w}, Z_P)$

Initialization: either by picking a $\mathbf{w}_{(0)}$ (or Z_P)

Auxiliary objective

$$\mathbf{w}^* = \min_{\mathbf{w}, Z_P} E(\mathbf{w}, Z_P) = \min_{\mathbf{w}} E(\mathbf{w})$$

This isn't any easier to optimize

Find stationary point by coordinate descent on $E(\mathbf{w}, Z_P)$

Initialization: either by picking a $\mathbf{w}_{(0)}$ (or Z_P)

Step 1:

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

Auxiliary objective

$$\mathbf{w}^* = \min_{\mathbf{w}, Z_P} E(\mathbf{w}, Z_P) = \min_{\mathbf{w}} E(\mathbf{w})$$

This isn't any easier to optimize

Find stationary point by coordinate descent on $E(\mathbf{w}, Z_P)$

Initialization: either by picking a $\mathbf{w}_{(0)}$ (or Z_P)

Step 1:

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

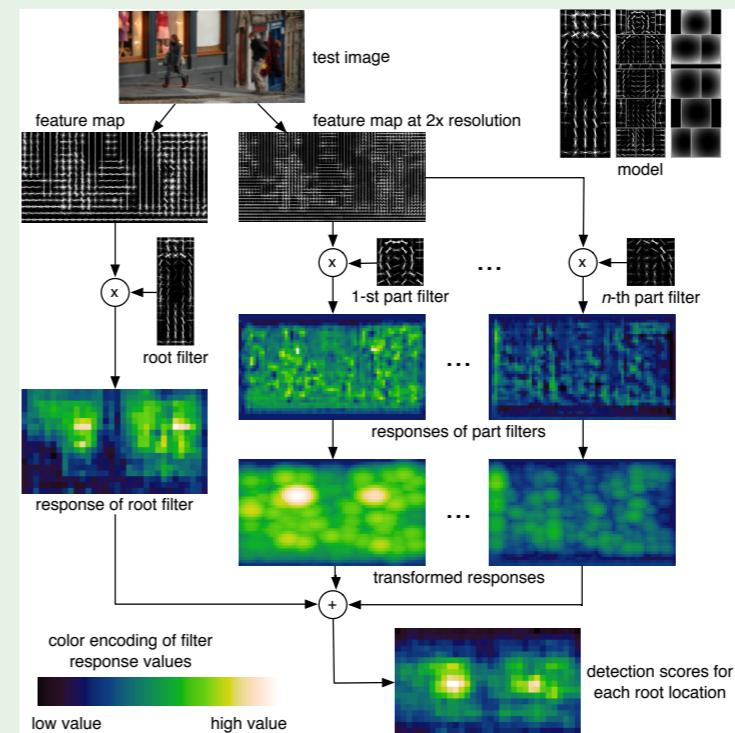
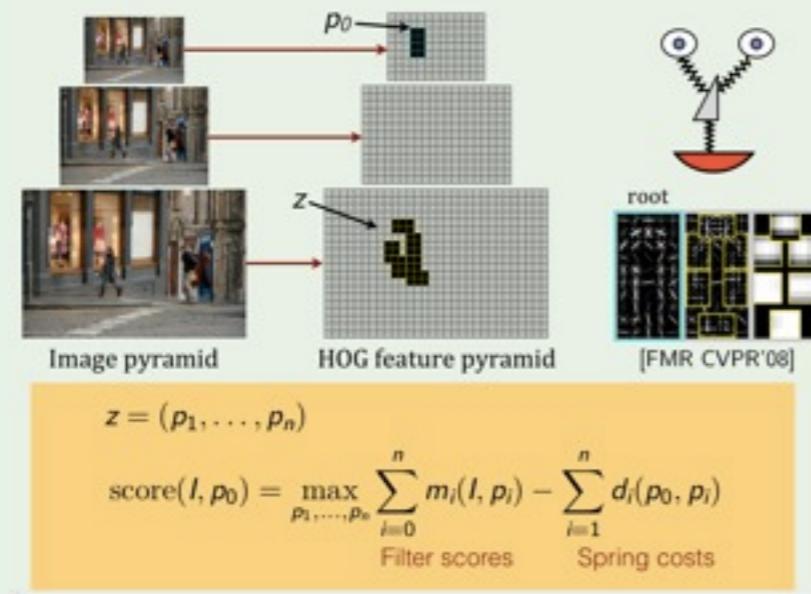
Step 2:

$$\mathbf{w}_{(t+1)} = \operatorname{argmin}_{\mathbf{w}} E(\mathbf{w}, Z_P)$$

Step 1

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

This is just detection:



Step 2

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

Convex

Step 2

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

Convex

Similar to a structural SVM

Step 2

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

Convex

Similar to a structural SVM

But, recall 500 million to 1 billion negative examples!

Step 2

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

Convex

Similar to a structural SVM

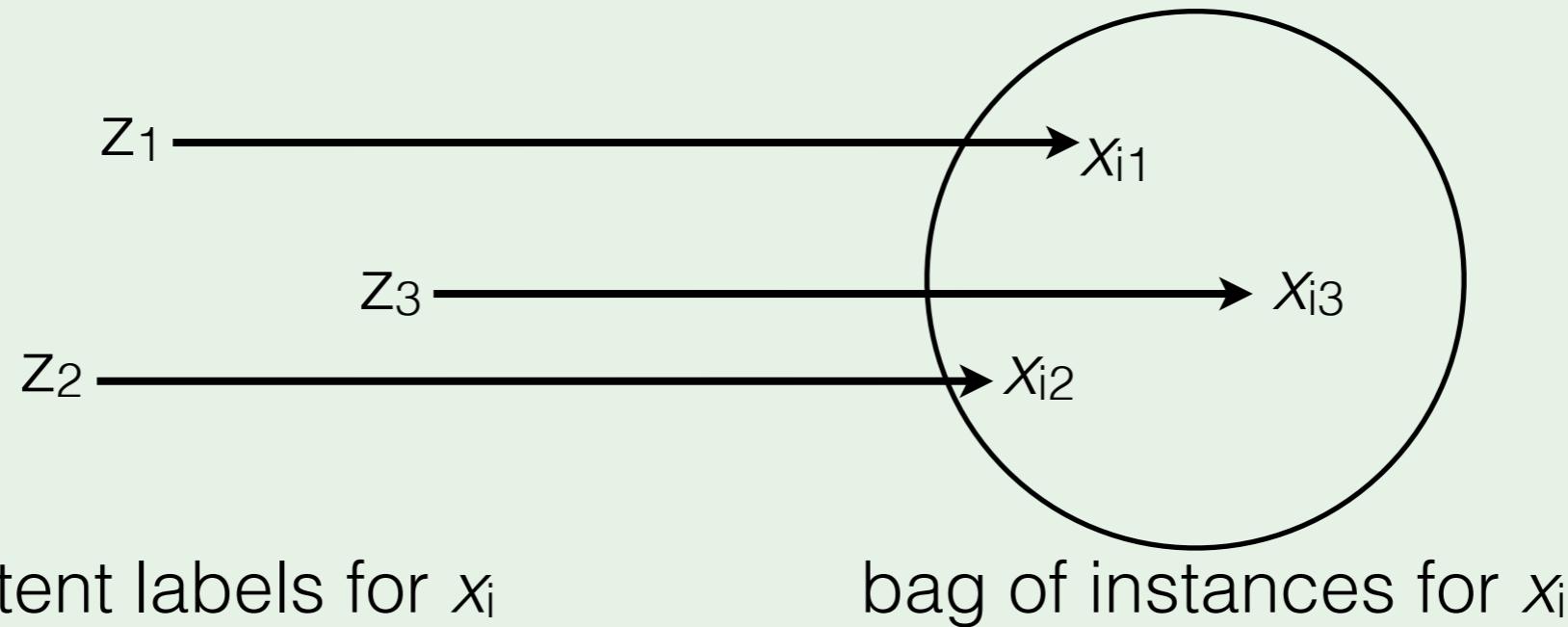
But, recall 500 million to 1 billion negative examples!

Can be solved by a working set method

- “bootstrapping”
- “data mining”
- “constraint generation”
- requires a bit of engineering to make this fast

Comments

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

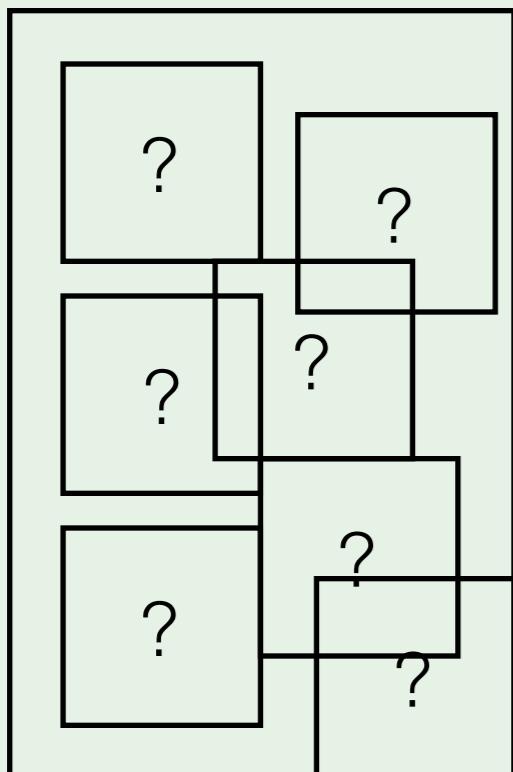


Latent SVM can be written as a latent structural SVM (Yu and Joachims ICML 2009)

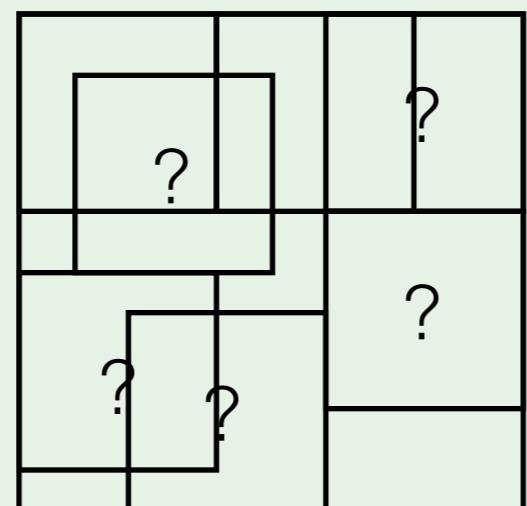
- natural optimization algorithm is concave-convex procedure
- similar to, but not exactly the same as, coordinate descent

What about the model structure?

fixed model *structure*



component 1



component 2

training images



y

+1

Model structure

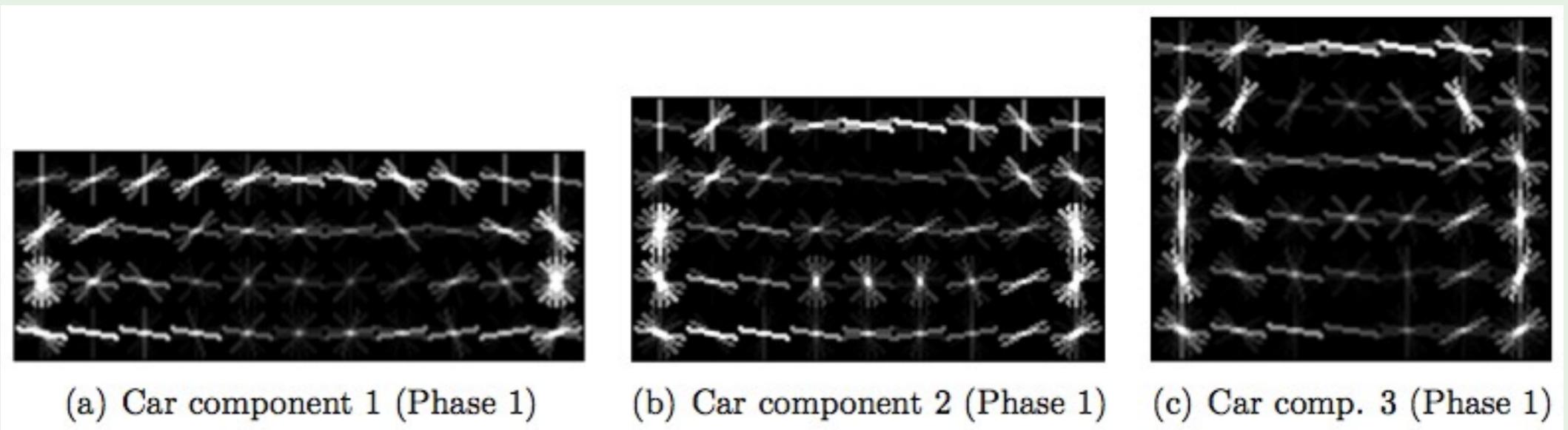
- # components
- # parts per component
- root and part filter shapes
- part anchor locations



-1

Learning model structure

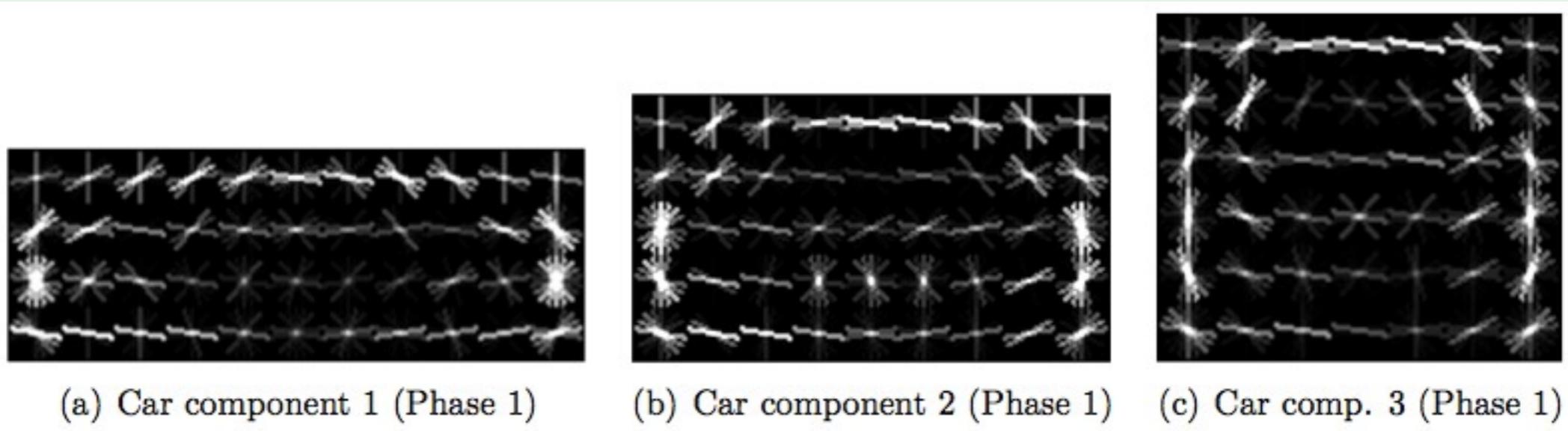
Split positives by aspect ratio



Warp to common size

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

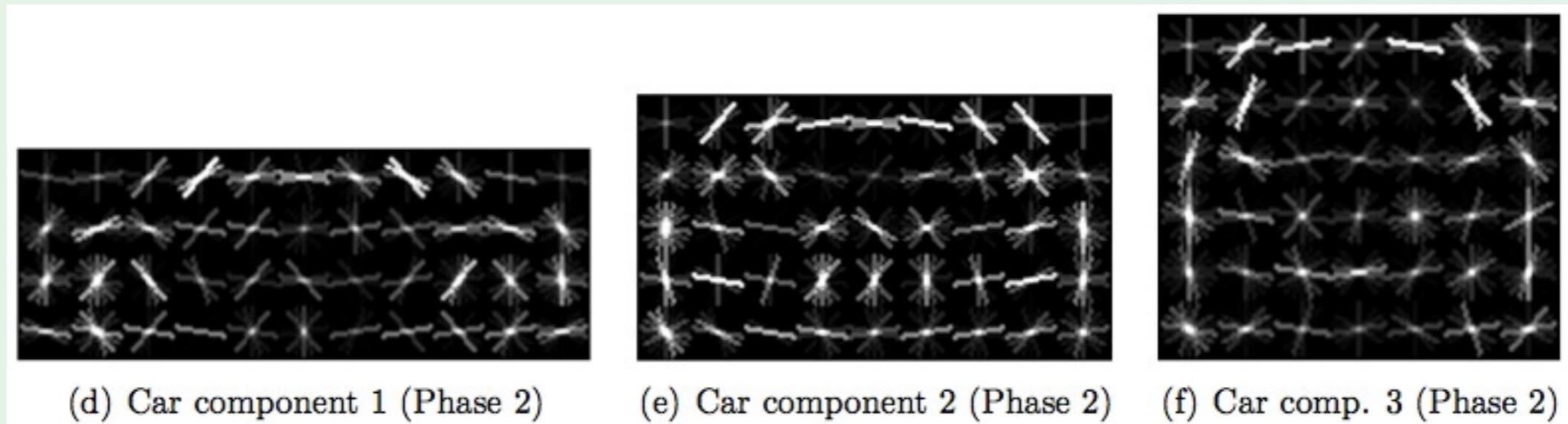
Learning model structure



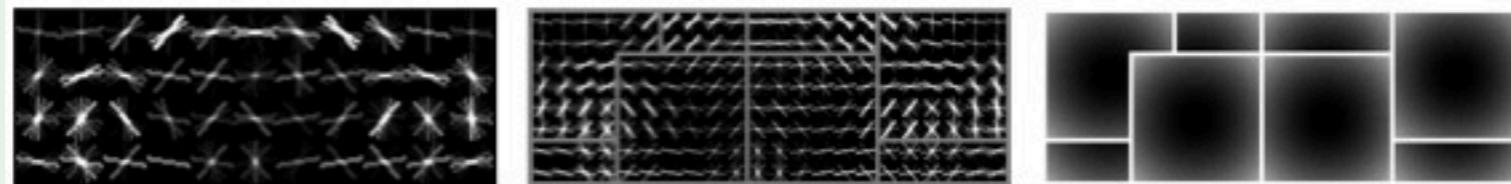
Use D&T filters as initial \mathbf{w} for LSVM training

Merge components

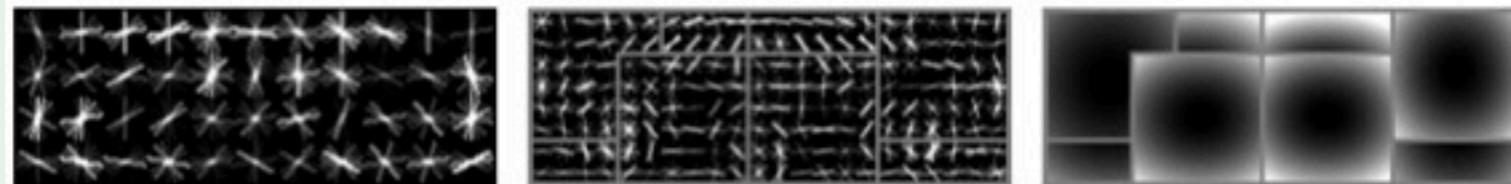
Root filter placement and component choice are latent



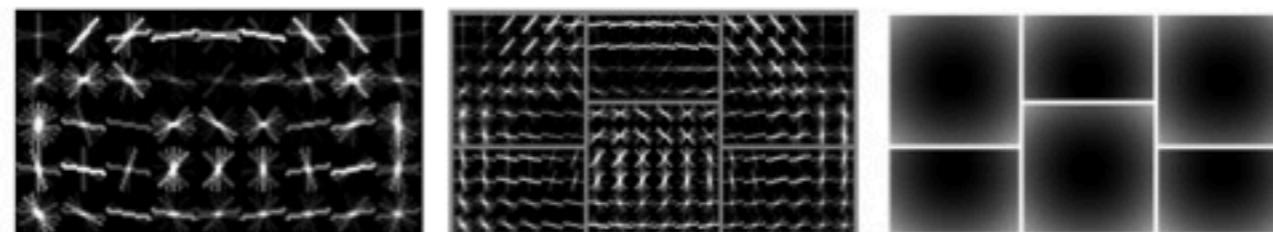
Learning model structure



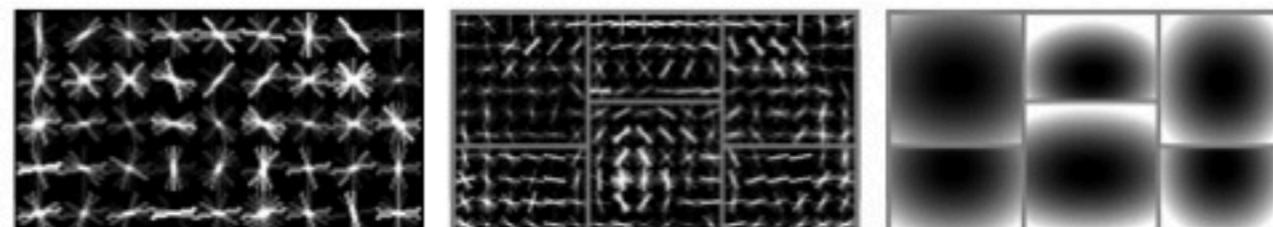
(a) Car component 1 (initial parts)



(b) Car component 1 (trained parts)



(c) Car component 2 (initial parts)

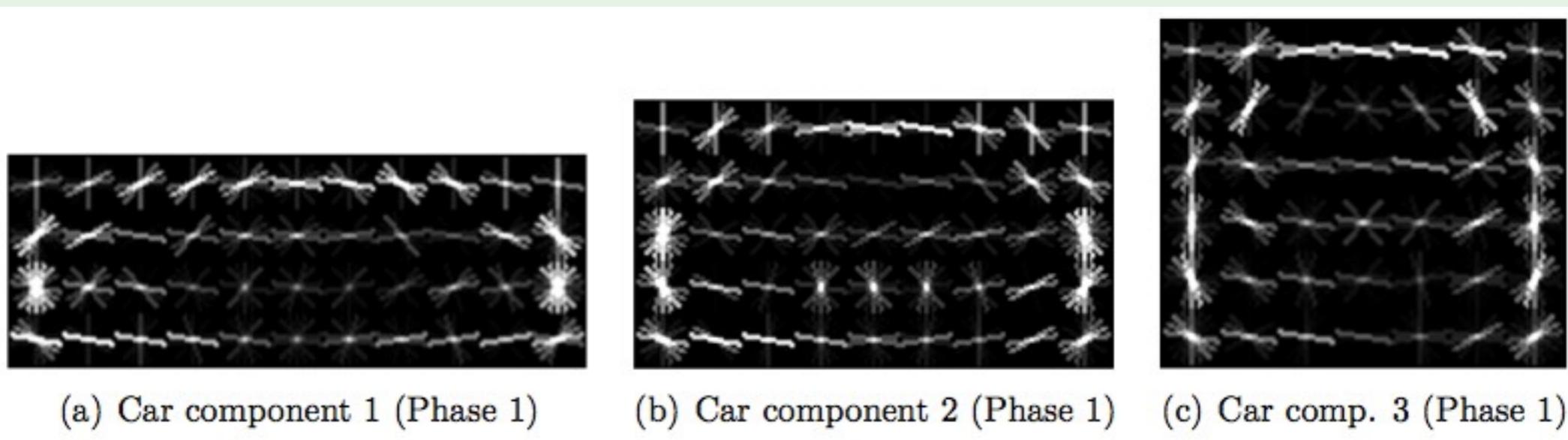


(d) Car component 2 (trained parts)

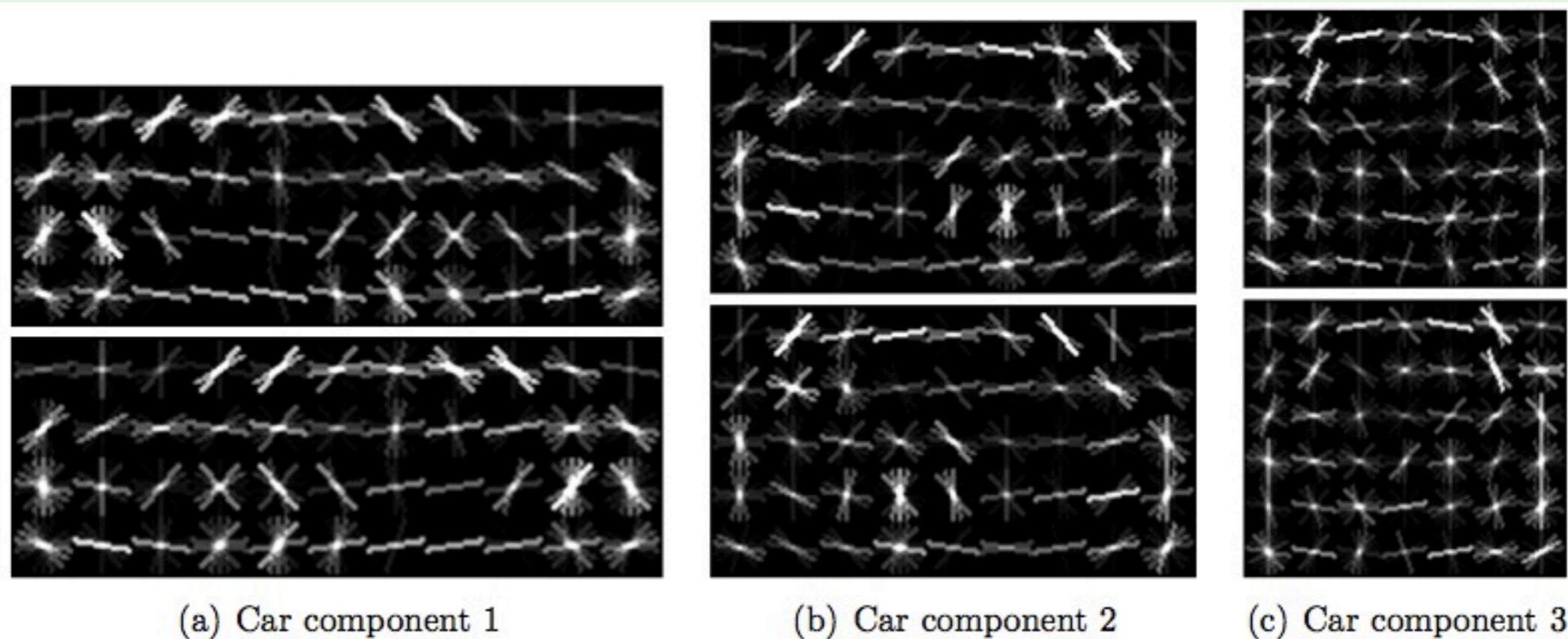
Add parts to cover high-energy areas of root filters

Continue training model with LSVM

Learning model structure



without orientation clustering



with orientation clustering

Learning model structure

In summary

- repeated application of LSVM training to models of increasing complexity
- structure learning involves many heuristics (and vision insight!)