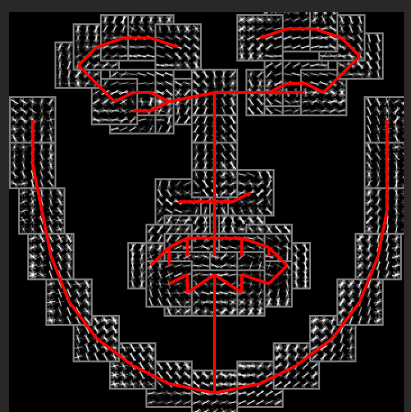
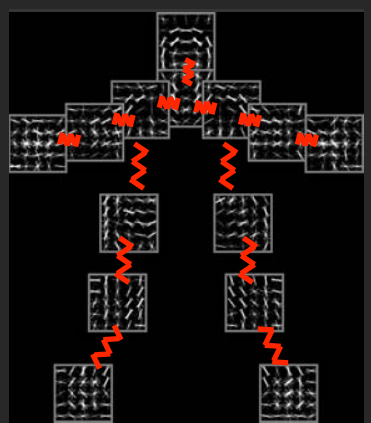
Pictorial structures

Deva Ramanan

UC Irvine





Original plan

Inference of deformable part models

"Pictorial structures"

(This morning)

Learning for deformable part models

"Latent SVMs"

(This afternoon)

Revised plan

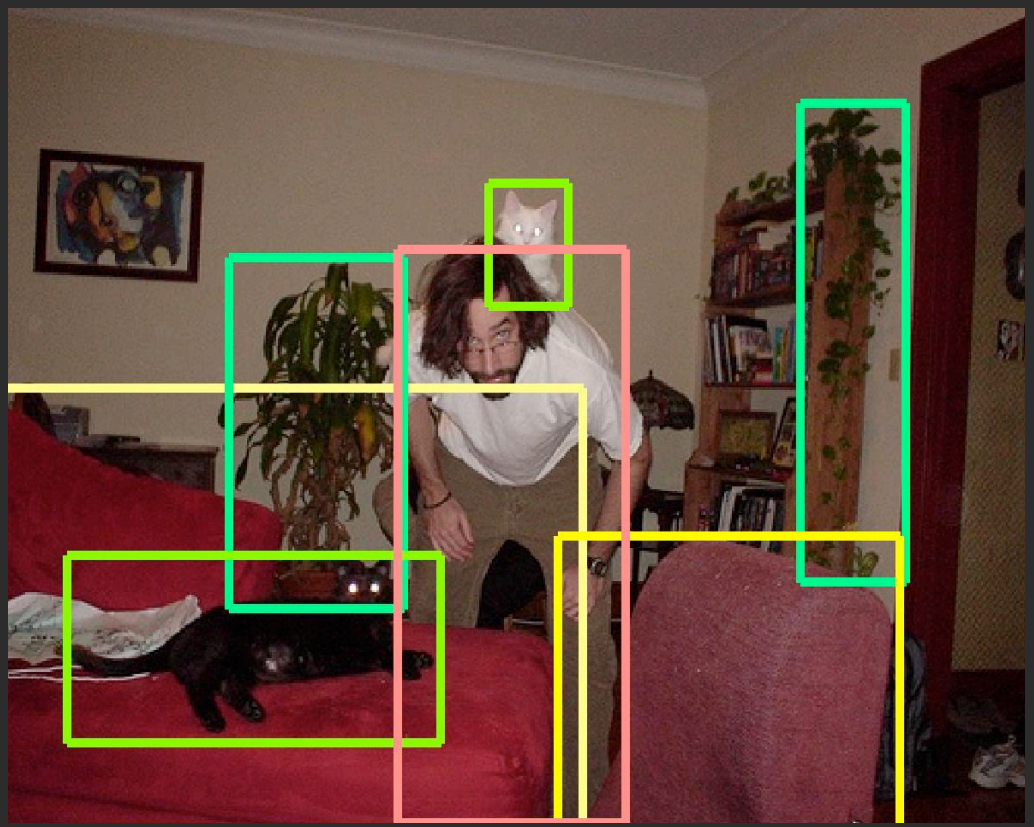
"Core" deformable part model system

(This morning)

"Extensions" of deformable part models

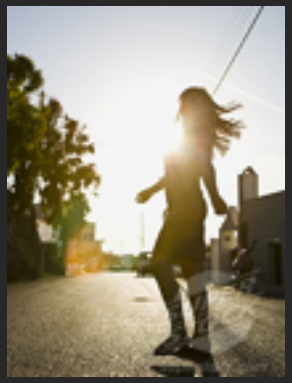
(This afternoon)

Goal: detect objects in cluttered images



person, plant, cat, dog, chair, sofa, car,
bicycle, motorbike, table, plane, ...

Why is finding objects (e.g. people) difficult?



variation in illumination

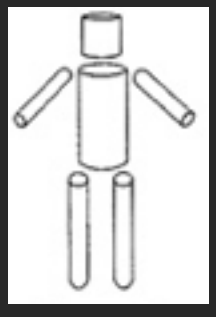
variation in appearance

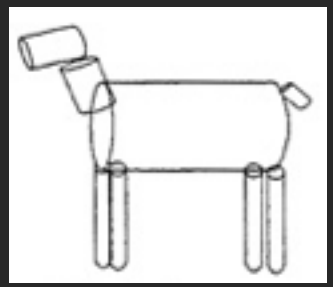
variation in pose, viewpoint

occlusion & clutter

Classic “nuisance factors” for general object recognition

Historical approaches

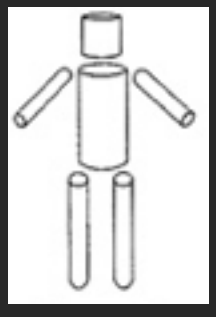


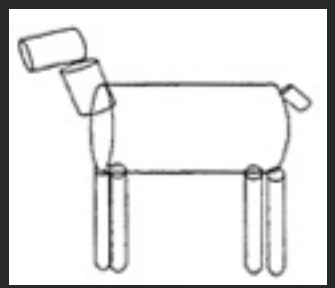


Geometric models
(1970s-1990s)

Hand-coded models

Historical approaches





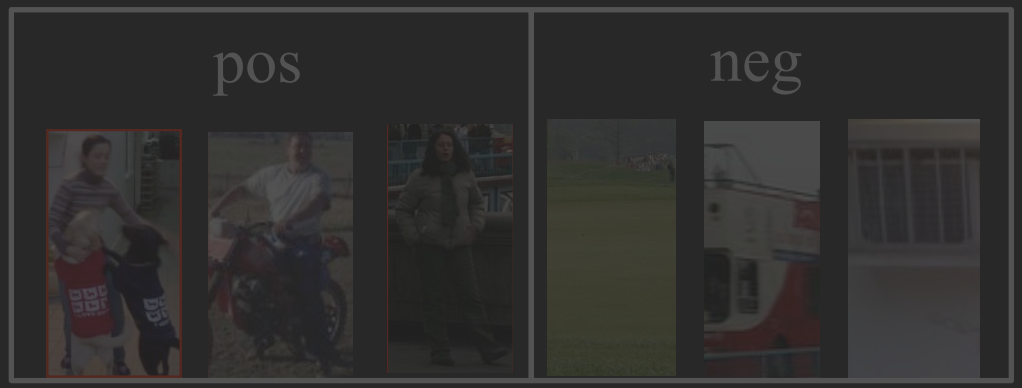
Geometric models
(1970s-1990s)

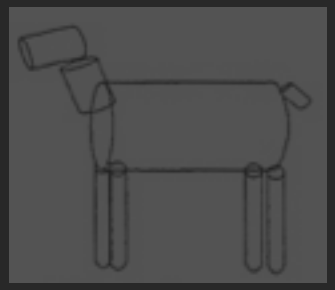
Hand-coded models

Statistical classifiers
(2000s-present)

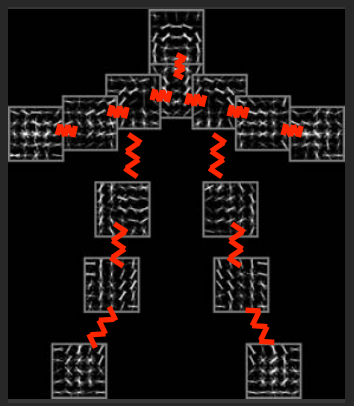
Large-scale training
Appearance-based representations

Historical approaches





geometric
statistical models



Geometric models
(1970s-1990s)

Hand-coded models

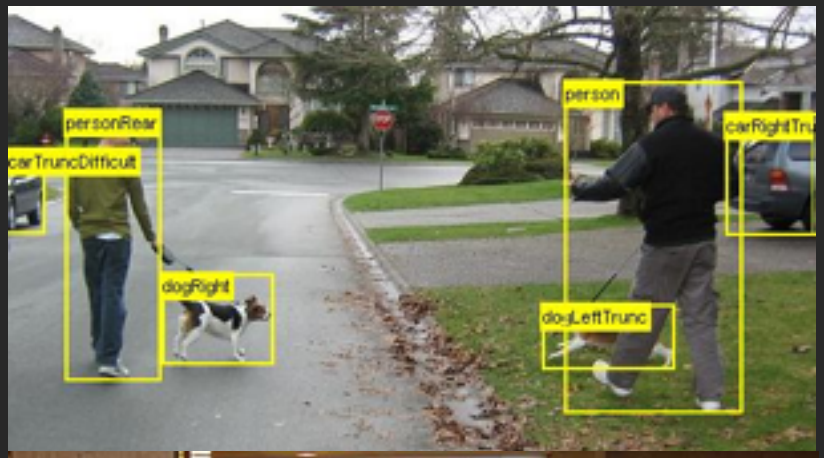
Statistical classifiers
(2000s-present)

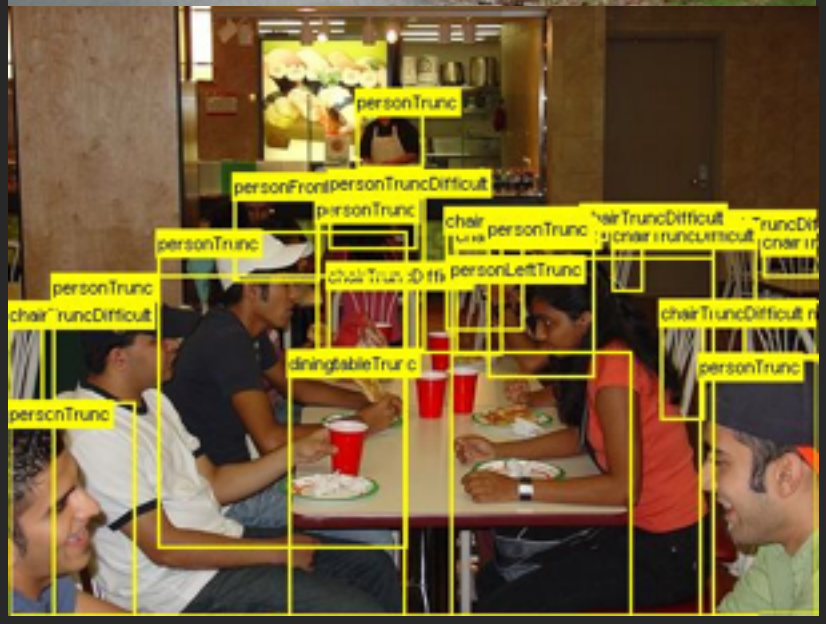
Large-scale training
Appearance-based representations

Evaluating performance

Columbia Dataset (1996)

Caltech 101/256
Image Net

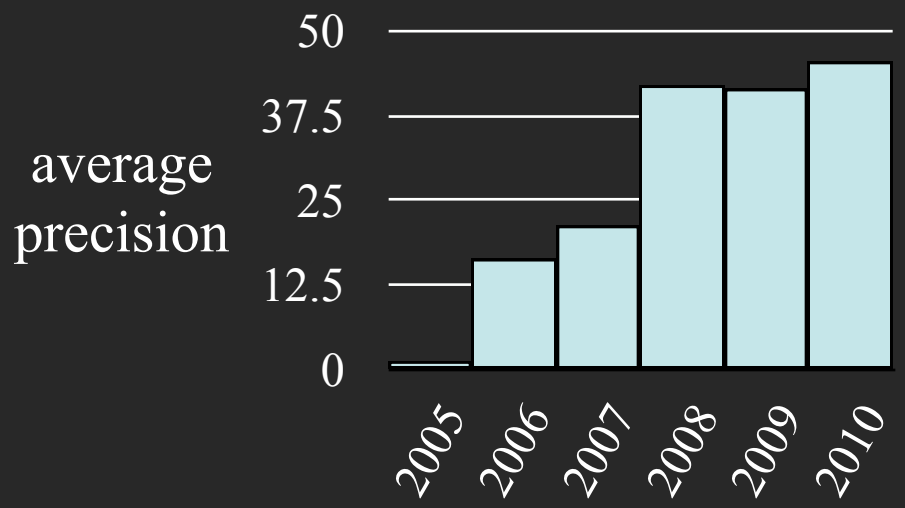


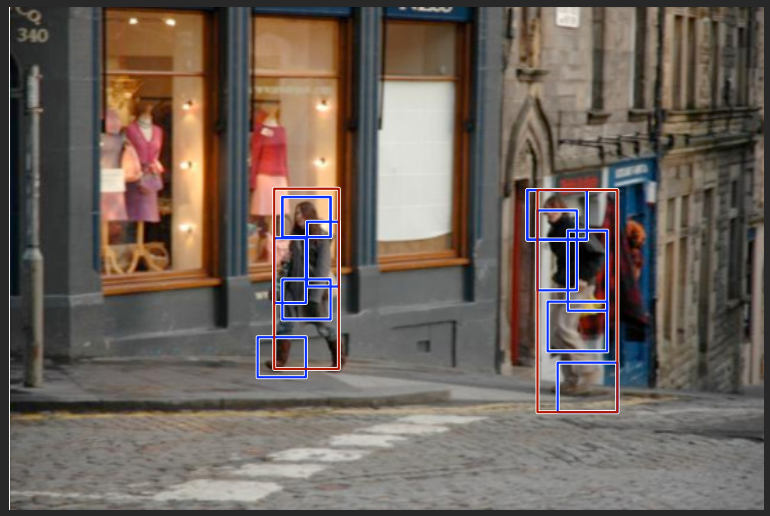


Flickr dataset (05-12)

"In-the-wild"

5 years of PASCAL people detection





1% to 45% in 5 years

Discriminative mixtures of star models 2007-2010

Felzenszwalb, McAllester, Ramanan *CVPR* 2008

Felzenszwalb, Girshick, McAllester, and Ramanan *PAMI* 2010

Benchmark evaluation

PASCAL VOC 2008 Average Precision Rankings

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	pers	plant	sheep	sofa	train	tv
CASIA_Det	25.2	14.6	9.8	10.5	6.3	23.2	17.6	9.0	9.6	10.0	13.0	5.5	14.0	24.1	11.2	3.0	2.8	3.0	28.2	14.6
Jena	4.8	1.4	0.3	0.2	0.1	1.0	1.3	-	0.1	4.7	0.4	1.9	0.3	3.1	2.0	0.3	0.4	2.2	6.4	13.7
_PlusClass	36.5	34.3	10.7	11.4	22.1	23.8	36.6	16.6	11.1	17.7	15.1	9.0	36.1	40.3	19.7	11.5	19.4	17.3	29.6	34.0
MPI_struct	25.9	8.0	10.1	5.6	0.1	11.3	10.6	21.3	0.3	4.5	10.1	14.9	16.6	20.0	2.5	0.2	9.3	12.3	23.6	1.5
Oxford	33.3	24.6	-	-	-	-	29.1	-	-	12.5	-	-	32.5	34.9	-	-	-	-	-	-
UoCTTIUCI	32.6	42.0	11.3	11.0	28.2	23.2	32.0	17.9	14.6	11.1	6.6	10.2	32.7	38.6	42.0	12.6	16.1	13.6	24.4	37.1
XRCE_Det	26.4	10.5	1.4	4.5	0.0	10.8	4.0	7.6	2.0	1.8	4.5	10.5	11.8	13.6	9.0	1.5	6.1	1.8	7.3	6.8

UoCTTIUCI 1rst on 7 classes, 2cnd on 8

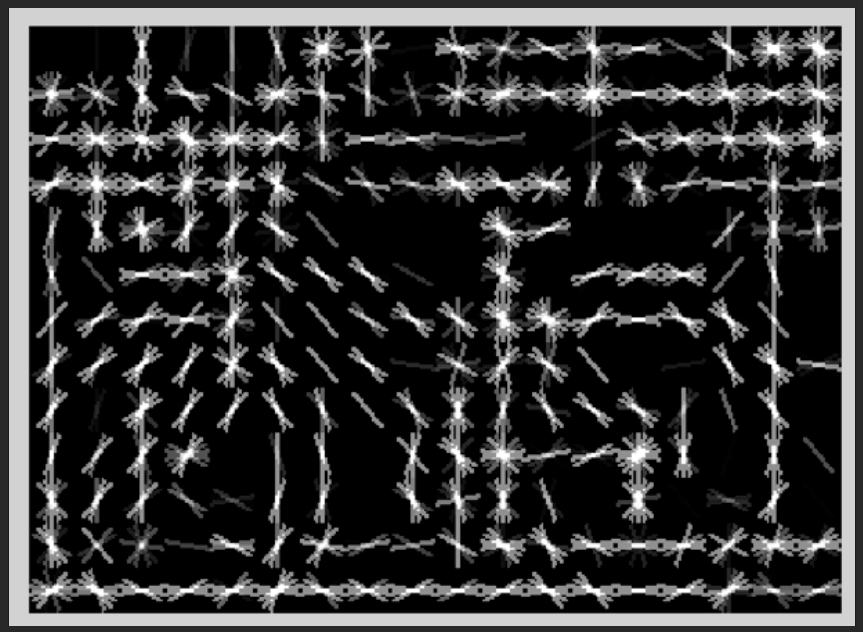
Test: ~ 2 second / image
Train: ~ 4 hours

All code online

PASCAL VOC Lifetime Achievement Award 2010
Invited application paper in *ICML* 2010
Invited article in *Communications of ACM* 2011

Image features:

Histograms of oriented gradients (HOG)



Bin gradients from 8x8 pixel neighborhoods into 9 orientations

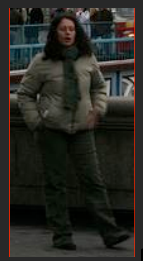
(Dalal & Triggs *CVPR* 05)

Scanning-window templates

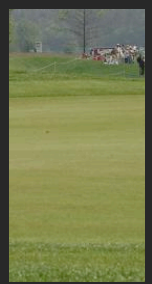
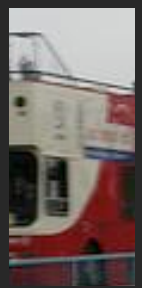
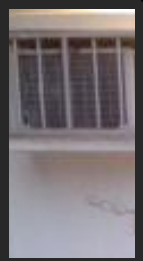
Dalal and Triggs CVPR05 (HOG)
Papageorgiou and Poggio ICIP99 (wavelets)

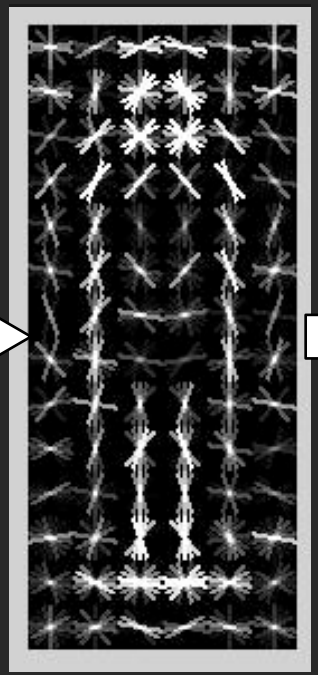
pos





neg



w

w = weights for orientation and spatial bins 

$w \cdot x > 0$

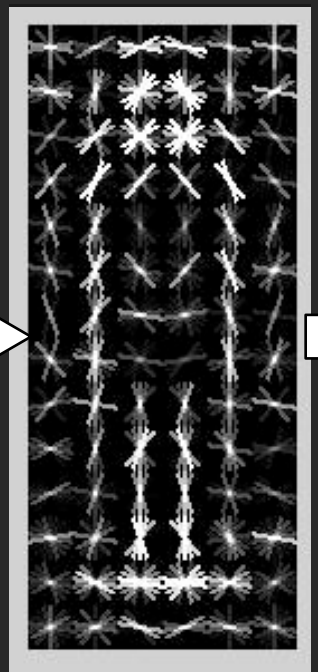
Train with a linear classifier (perceptron, logistic regression, SVMs...)

Scanning-window templates

Dalal and Triggs CVPR05 (HOG)
Papageorgiou and Poggio ICIP99 (wavelets)

pos

neg



w

w = weights for orientation and spatial bins 

$w \cdot x > 0$

Train with a linear classifier (perceptron, logistic regression, SVMs...)

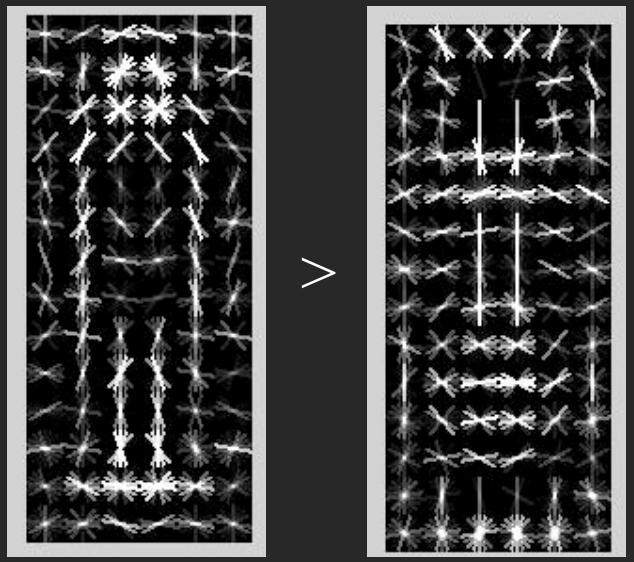
How to interpret positive and negative weights?

$$w \cdot x > 0$$

$$(w_{pos} - w_{neg}) \cdot x > 0$$

$$w_{pos} \cdot x > w_{neg} \cdot x$$

Pedestrian template



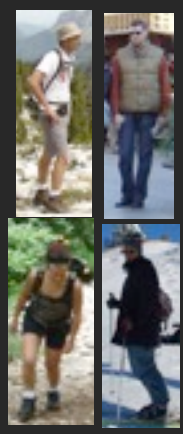
Pedestrian background template

Right approach is to compete pedestrian, pillar, doorway... models

Background class is hard to model - easier to penalize particular vertical edges

Out-of-core learning

pos



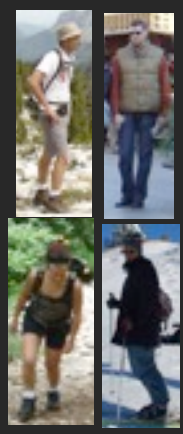
neg

Our test set distribution is highly imbalanced; so should be the training set

(hundreds of positives, hundreds of millions of negatives)

Out-of-core learning

pos



neg

Our test set distribution is highly imbalanced; so should be the training set

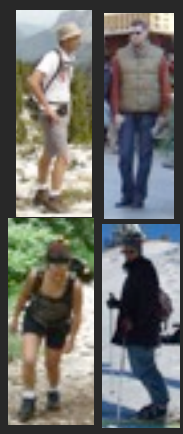
(hundreds of positives, hundreds of millions of negatives)

SVMs are attractive because they generate sparse learning problems

(One can solve problems that are too big to fit in memory)

Large-scale learning

pos

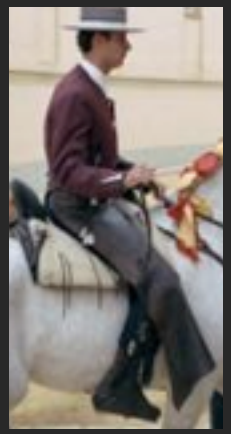


neg

1. Train SVM with subset of training data
2. Use model to find margin violations on all training data
3. If no new violations are found, model is optimal!

(More in afternoon's talk)

How to model large variations in appearance?



Mixtures of templates

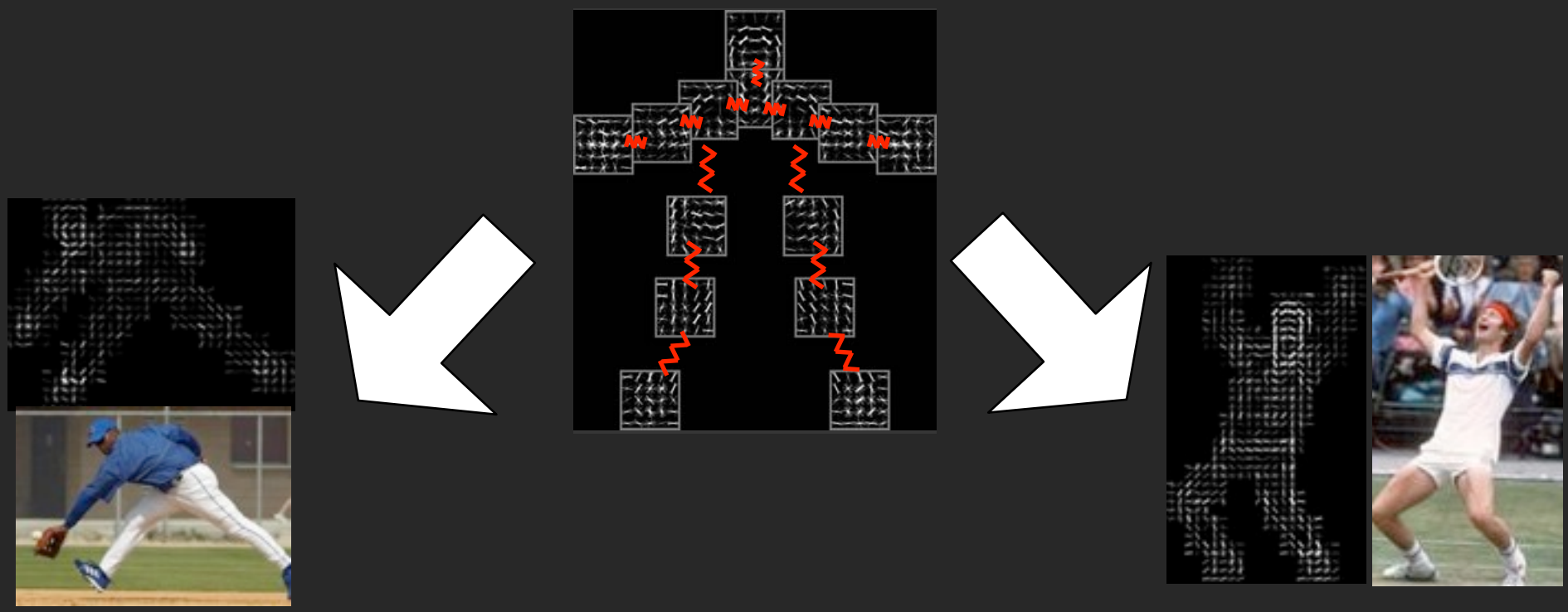
Train “sub-category” templates for each type of pose, body-shape, etc.

But how to handle...

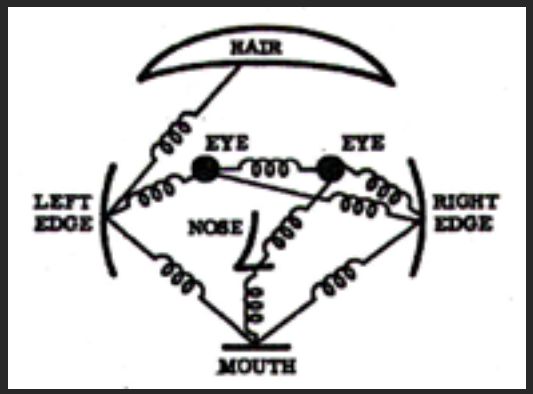
Long-tail distribution of poses

We need lots of templates, and will likely have little data of ‘yoga twist’ poses

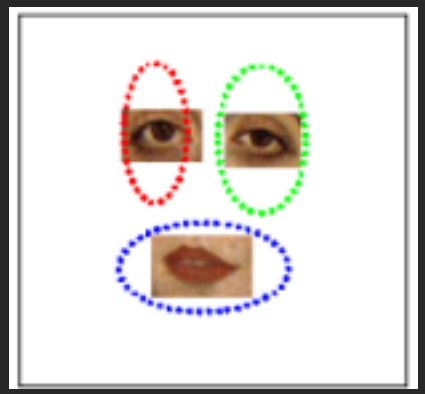
Deformable part models



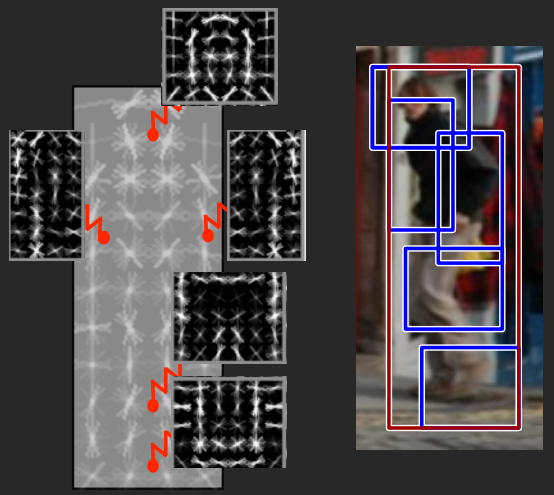
History over 40 years



Pictorial structures



Constellation models

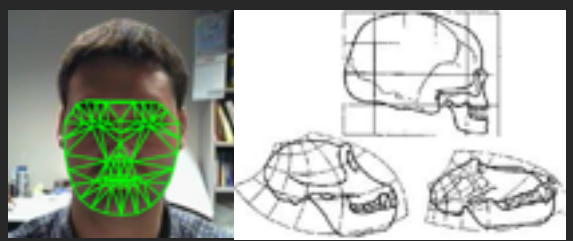


Deformable part models

Model encodes local appearance + pairwise geometry

Pictorial Structures (Fischler & Elschlager 73, Felzenswalb and Huttenlocher 00)
Cardboard People (Yu et al 96)
Body Plans (Forsyth & Fleck 97)
Active Appearance Models (Cootes & Taylor 98)
Constellation Models (Burl et all 98, Fergus et al 03)

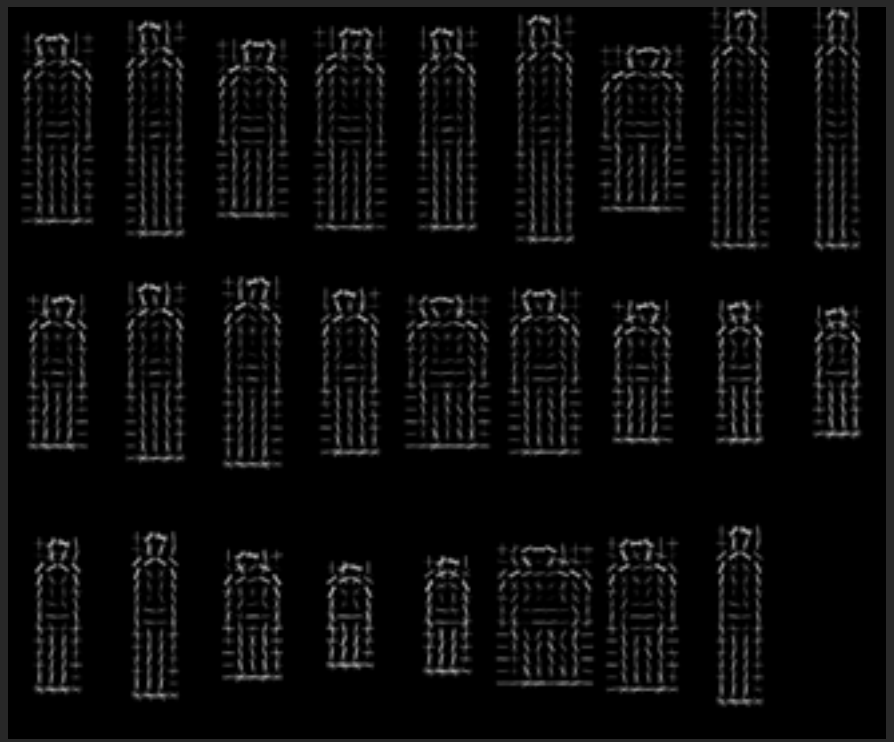
Relationship to other deformable models



Active appearance models

Continuous parameterization of shape
Continuous matching algs

"Local search"

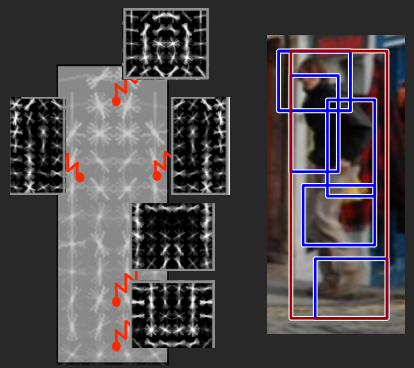


Deformable parts

Discrete parameterization of shape
Combinatorial matching algs

"Brute-force search"

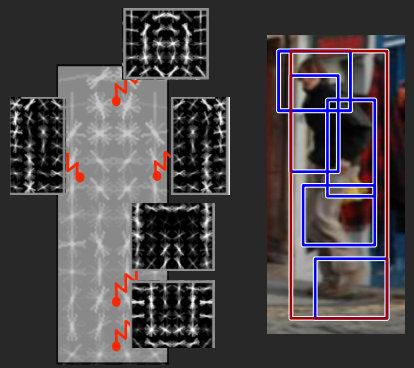
Scoring function



$$S(x, z) =$$

x = image
$z_i = (x_i, y_i)$
$z = \{z_1, z_2 \ldots\}$

Scoring function

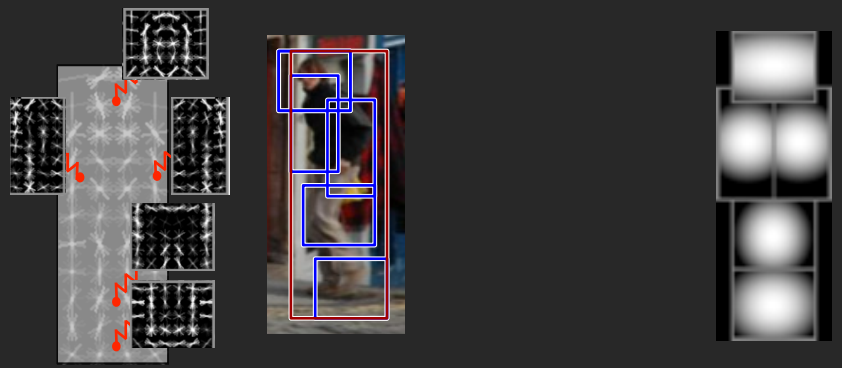


$$S(x, z) = \sum_i w_i \cdot \phi(x, z_i) +$$

x = image
$z_i = (x_i, y_i)$
$z = \{z_1, z_2 \ldots\}$

part template scores

Scoring function



$$S(x, z) = \sum_i w_i \cdot \phi(x, z_i) + \sum_{i,j \in E} w_{ij} \cdot \psi(z_i, z_j)$$

x = image
$z_i = (x_i, y_i)$
$z = \{z_1, z_2 \ldots\}$

part template scores

spring deformation model

$$\psi(z_i, z_j) = \begin{bmatrix} dx & dx^2 & dy & dy^2 \end{bmatrix}^T$$

E = relational graph

Deformation modes

$$\sum_{ij \in E} w_{ij} \cdot \psi(z_i, z_j) = (z - \mu)^T \Lambda (z - \mu)$$

where (μ, Λ) are functions/reparameterizations of $\{w_{ij}\}$

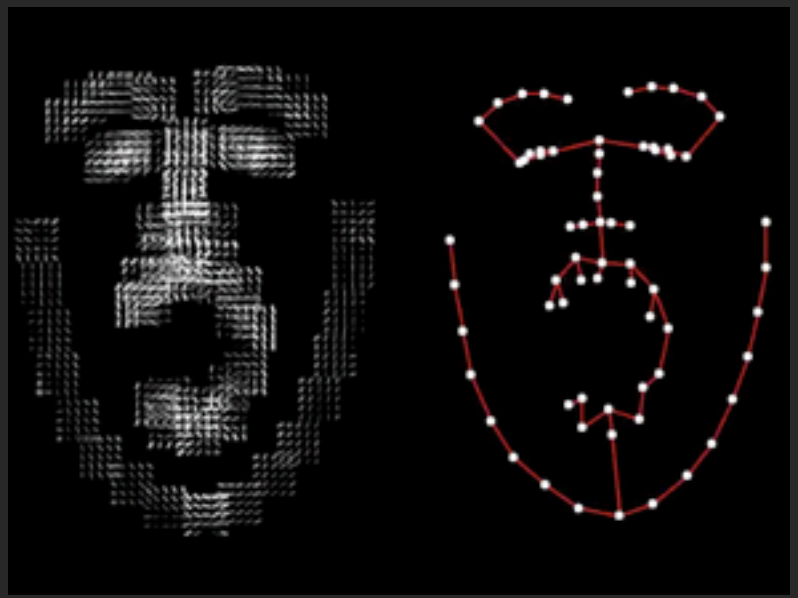
and Λ is the block-sparse inverse of a shape "covariance" matrix

Deformation modes

$$\sum_{ij \in E} w_{ij} \cdot \psi(z_i, z_j) = (z - \mu)^T \Lambda (z - \mu)$$

where (μ, Λ) are functions/reparameterizations of $\{w_{ij}\}$

and Λ is the block-sparse inverse of a shape "covariance" matrix



Scoring function

$$S(x,z) = \sum_i w_i \cdot \phi(x, z_i) + \sum_{ij \in E} w_{ij} \cdot \psi(z_i, z_j)$$

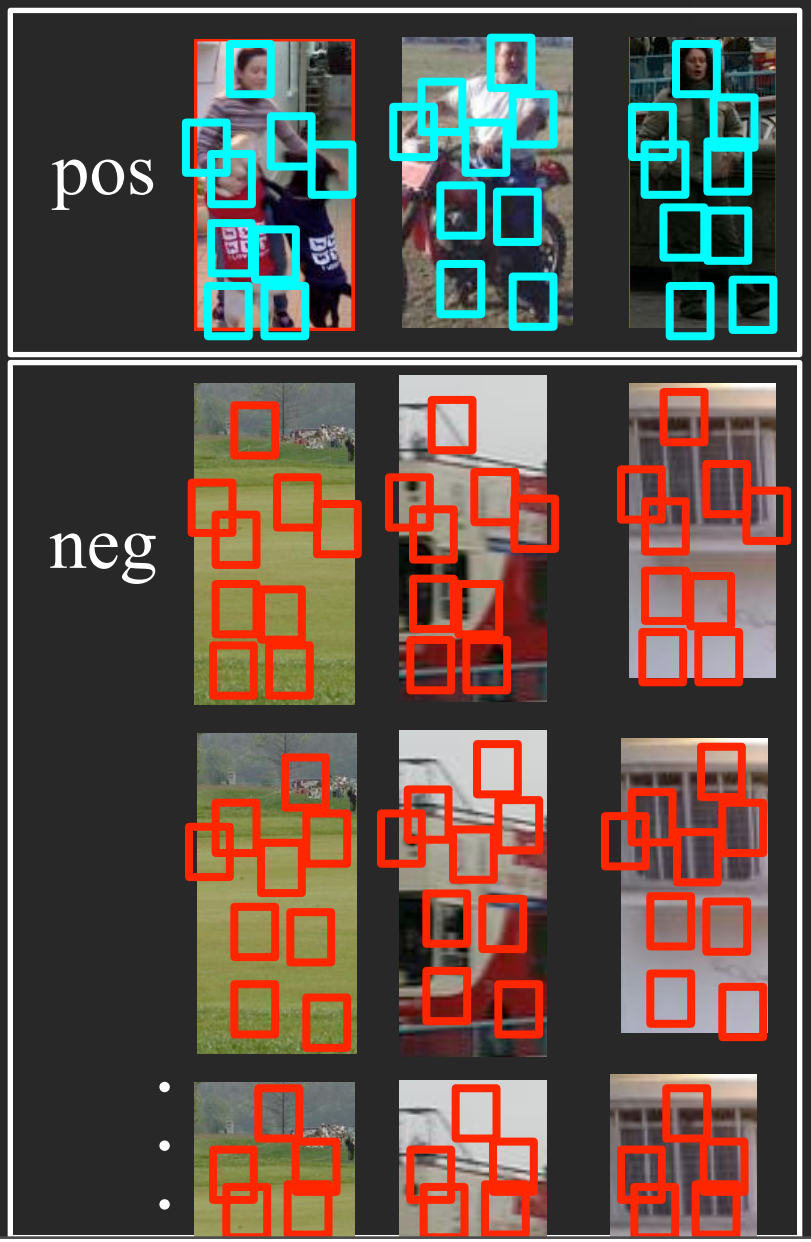
$$w = \left[\begin{array}{c} \\ \\ \\ \\ \end{array}\right] \qquad \Phi(x,z) = \left[\begin{array}{c} \\ \\ \\ \\ \end{array}\right]$$

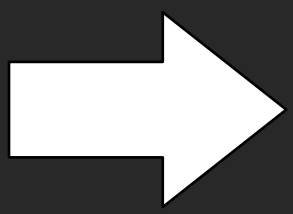
Score is linear in local templates w_i and spring parameters w_{ij}

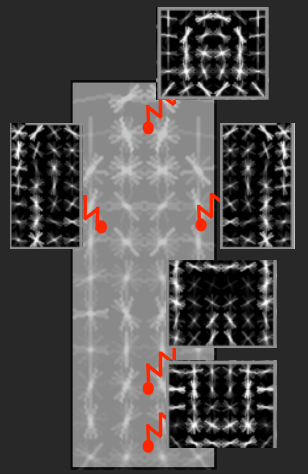
$$S(x,z) = w \cdot \Phi(x,z)$$

Learning structured linear parameters

$$S(x, p) = w \cdot \Phi(x, p)$$



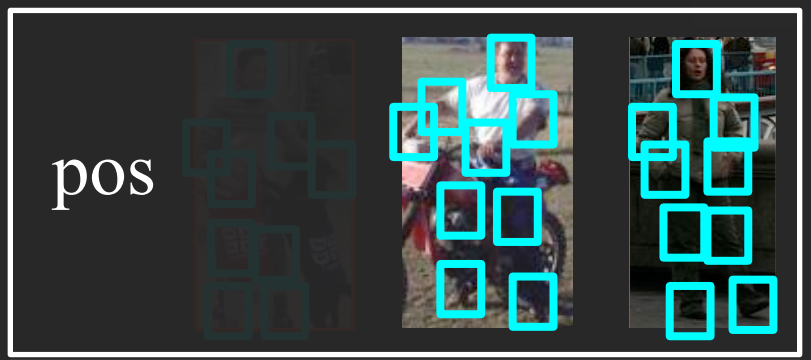


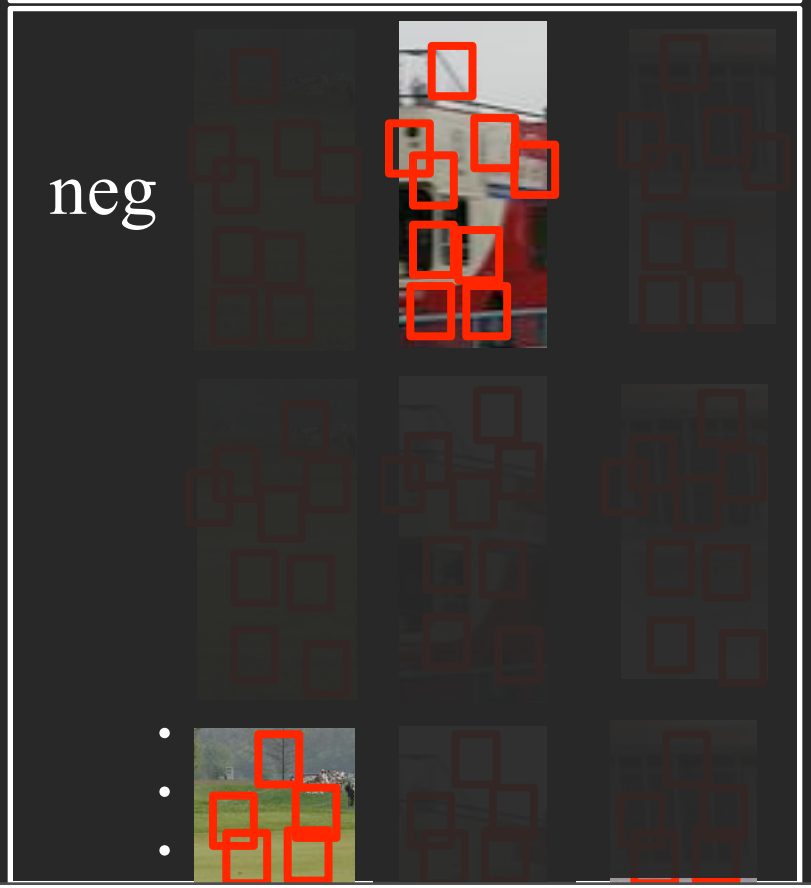


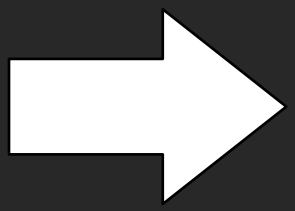
(Apply same sparse learning tricks to deal with exponential set of negatives!)

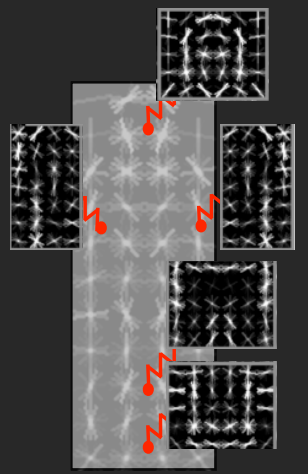
Learning structured linear parameters

$$S(x, p) = w \cdot \Phi(x, p)$$



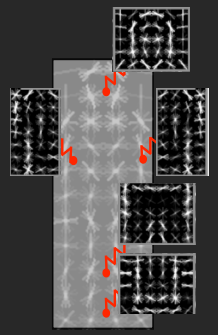




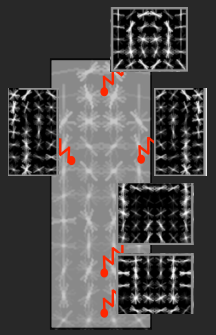


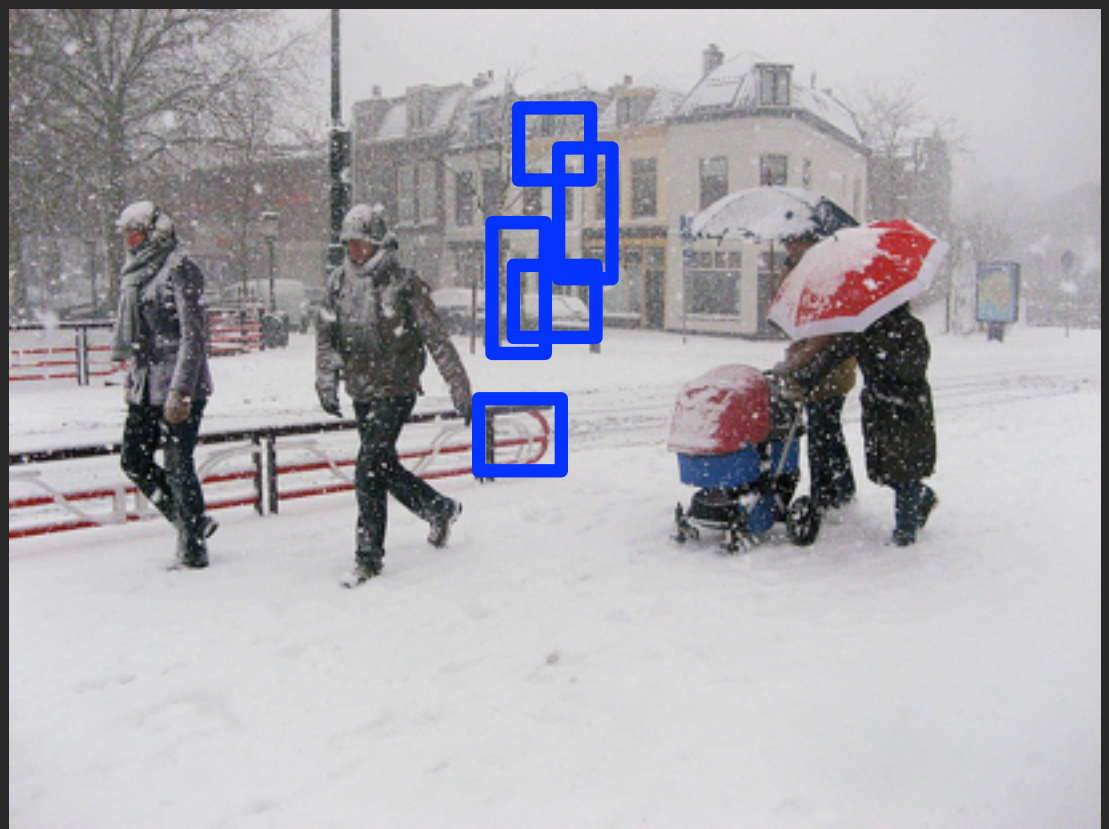
(Apply same sparse learning tricks to deal with exponential set of negatives!)

Inference

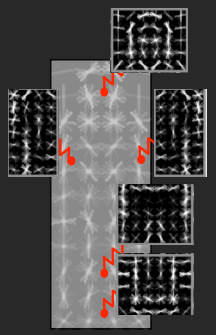


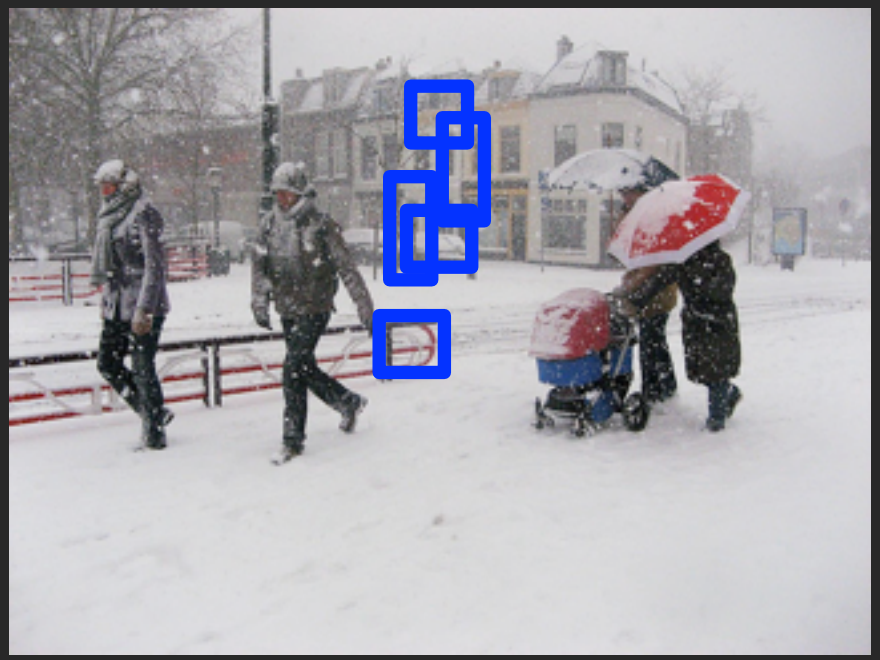
Inference

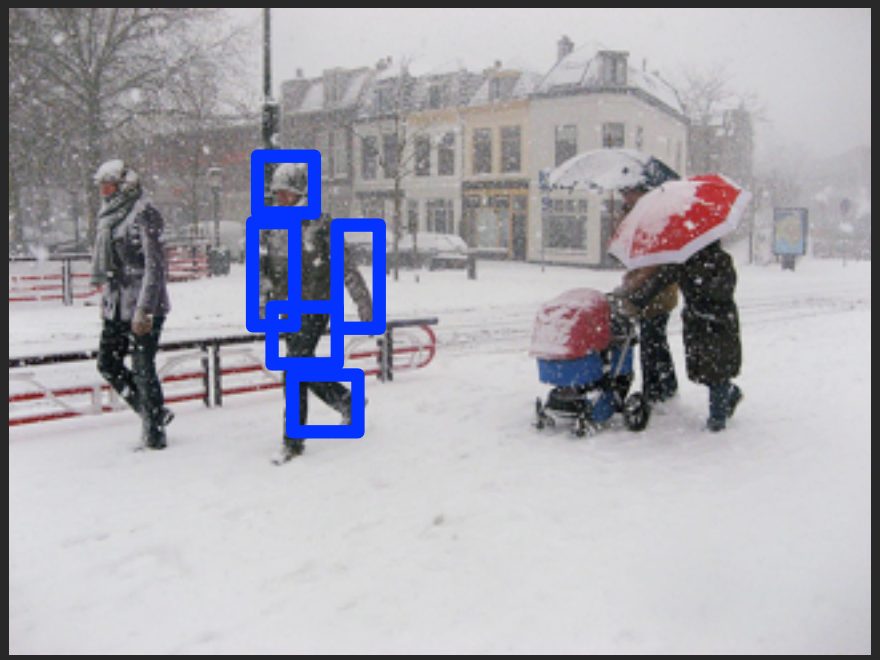




Inference



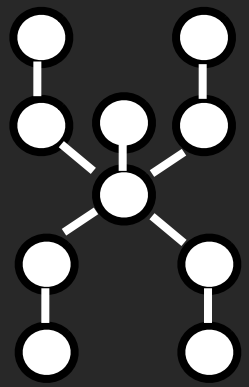




K parts with L possible positions: score all L^K configurations

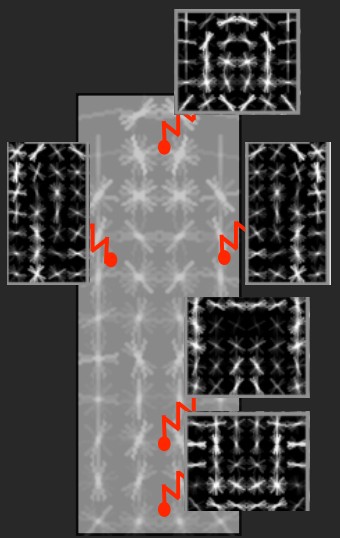
Inference: $\max_z S(x,z)$

Felzenszwalb & Huttenlocher 05

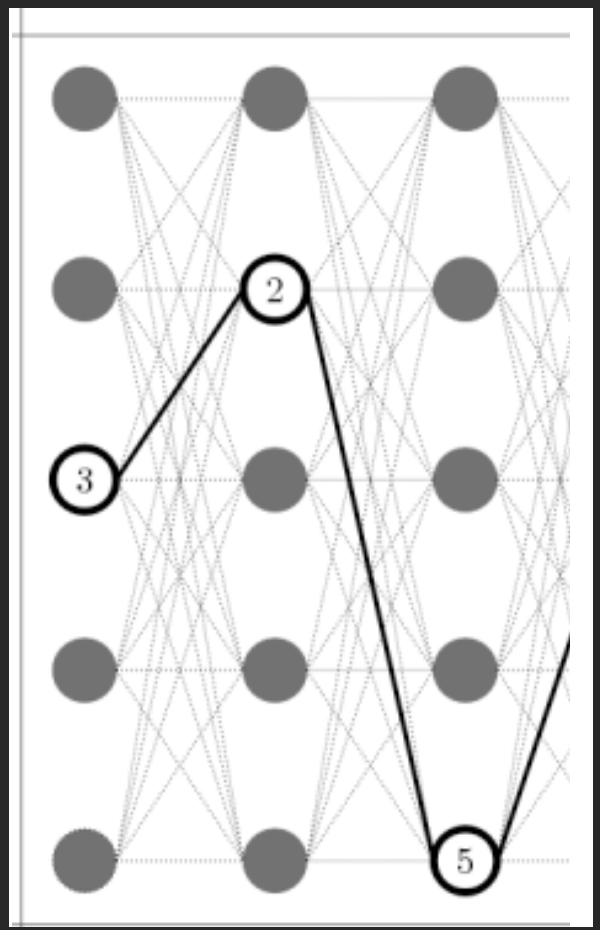


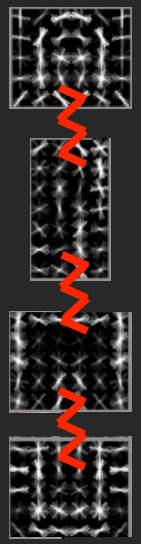
Markov model

- L candidate locations, K parts
- Dynamic programming reduces search from $O(L^k)$ to $O(KL^2)$ for trees
- For each candidate torso, independently estimate best arm and leg
- In practice, no more expensive than scoring each part independently



Inference: max S(x,z)





1) Initialize nodes with match score

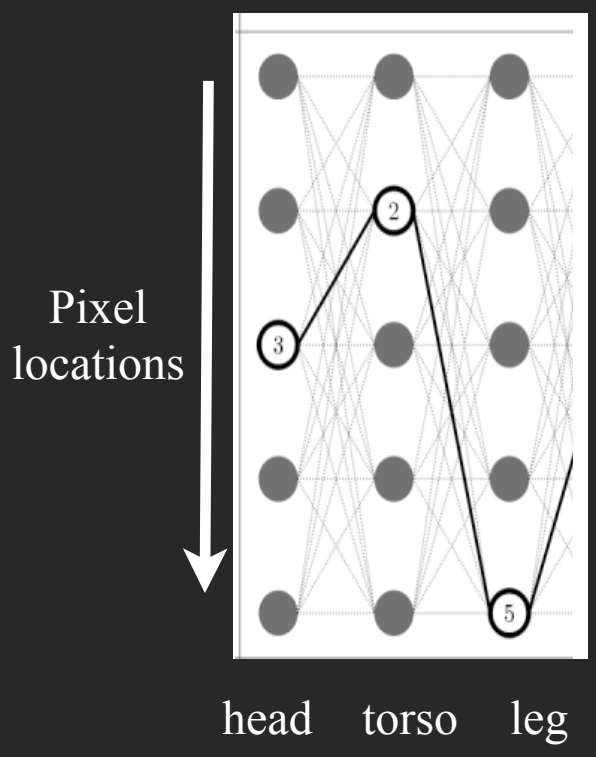
2) Initialize edges with spring score

3) Find best path from left to right

In practice, (1) is bottleneck

General formulation

$$S(x, z) = \sum_i \phi_i(z_i, x) + \sum_{ij \in E} \psi_{ij}(z_i, z_j, x)$$

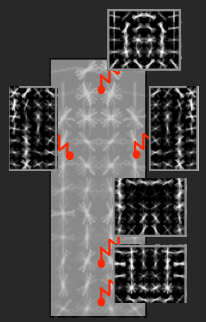


Local and pairwise potentials can be arbitrary nonlinear functions of image and position

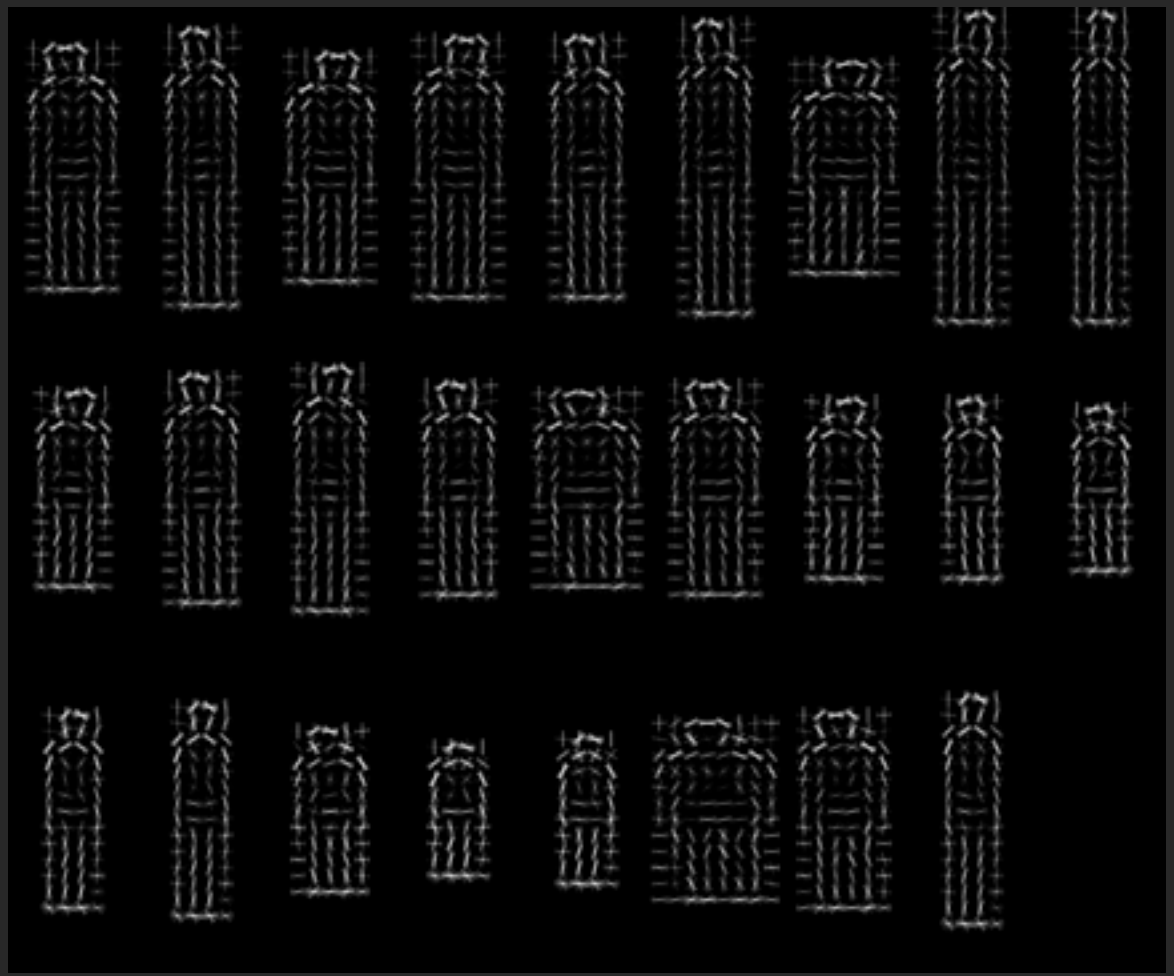
(e.g., neural net part model)

(e.g., intervening contour cue on part pairs)

Inference



≈



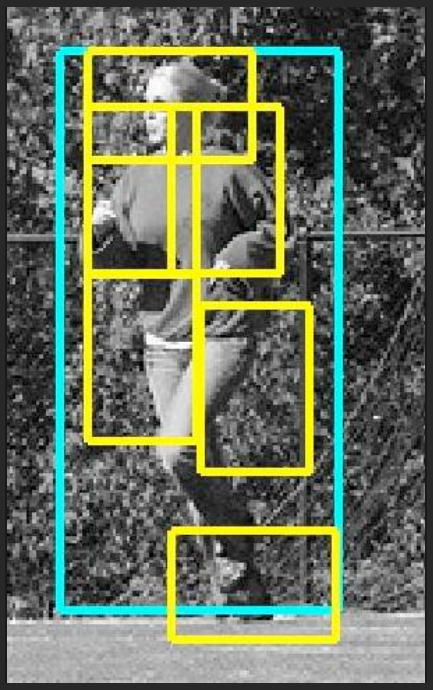
Classification

$$f_w(x) > 0$$

$$f_w(x) = w \cdot \Phi(x)$$

Latent-variable classification

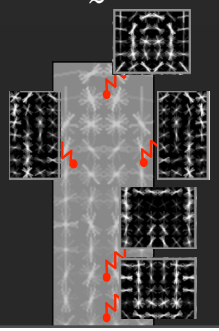
$$f_w(x) > 0$$



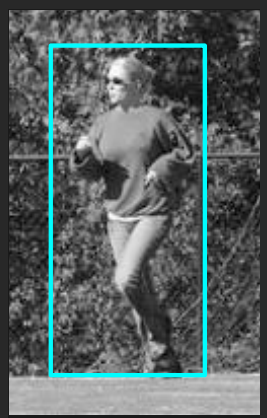
$$f_w(x) = w \cdot \Phi(x)$$

$$f_w(x) = \max_z S(x, z)$$

$$= \max_z w \cdot \Phi(x, z)$$

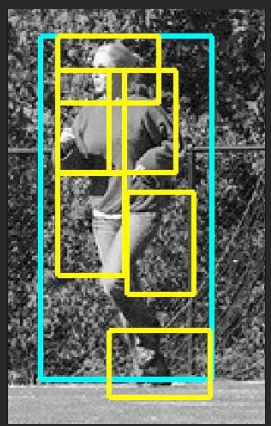


Comparison



$$f_w(x) = w \cdot \Phi(x)$$

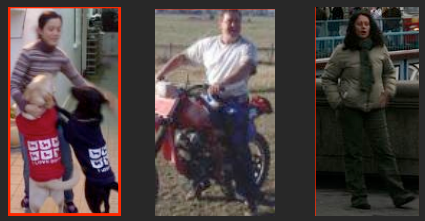
Score $f_w(x)$ is linear in w



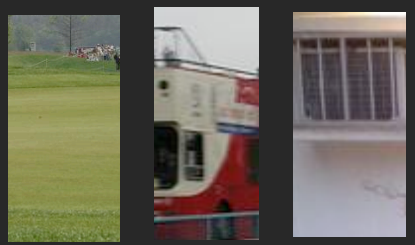
$$f_w(x) = \max_z w \cdot \Phi(x, z)$$

?

SVMs



pos



neg

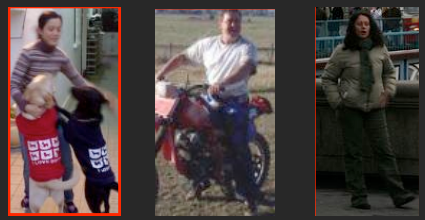
Given positive and negative training windows $\{x_n\}$

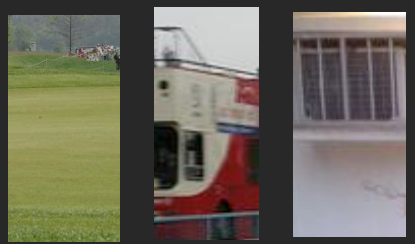
$$L(w) = ||w||^2 + \sum_{n \in \text{pos}} \max(0, 1 - f_w(x_n)) + \sum_{n \in \text{neg}} \max(0, 1 + f_w(x_n))$$

$$f_w(x) = w \cdot \Phi(x)$$

L(w) is convex (Quadratic Program)

Latent SVMs


pos


neg

Given positive and negative training windows $\{x_n\}$

$$L(w) = ||w||^2 + \sum_{n \in \text{pos}} \max(0, 1 - f_w(x_n)) + \sum_{n \in \text{neg}} \max(0, 1 + f_w(x_n))$$

$$f_w(x) = \max_z w \cdot \Phi(x, z)$$

L(w) is "almost" convex

Latent SVMs

$$f_w(x) = \max_z w \cdot \Phi(x, z)$$

Given positive and negative training windows $\{x_n\}$

$$L(w) = ||w||^2 + \sum_{n \in \text{pos}} \max(0, 1 - f_w(x_n)) + \sum_{n \in \text{neg}} \max(0, 1 + f_w(x_n))$$

$$w \cdot \Phi(x_n, z_n)$$

"almost-convex" - L(w) is convex if we fix latent values for positives

1) Given positive part locations, learn w with a convex program

$$w = \underset{w}{\operatorname{argmin}}\, L(w) \quad \text{with fixed} \quad \{z_n : n \in \text{pos}\}$$

Coordinate descent

1) Given positive part locations, learn w with a convex program

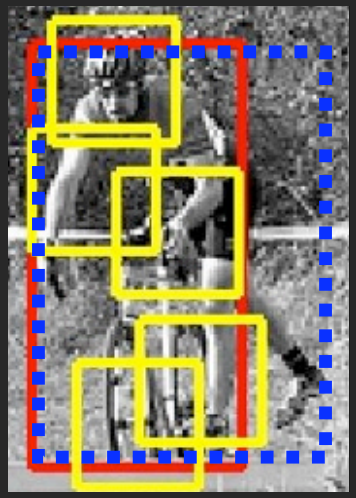
$$w = \underset{w}{\operatorname{argmin}}\, L(w) \quad \text{with fixed} \quad \{z_n : n \in \text{pos}\}$$

2) Given w, estimate part locations on positives

$$z_n = \underset{z}{\operatorname{argmax}}\, w \cdot \Phi(x_n, z) \quad \forall n \in \text{pos}$$

The above steps perform coordinate descent on a joint loss
Can be seen as an instance of the CCCP algorithm (Yuille)

Treat ground-truth labels as partially latent

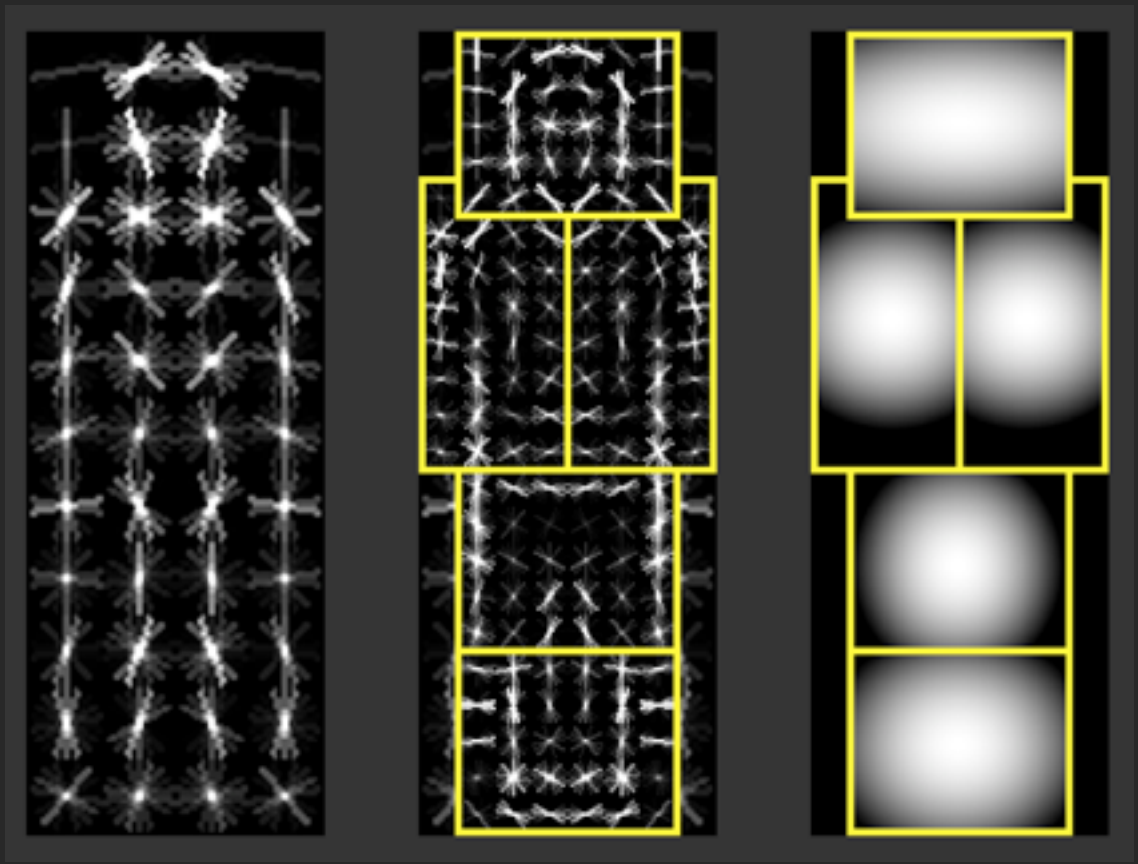


Allows for “cleaning up” of noisy labels (in blue) during iterative learning

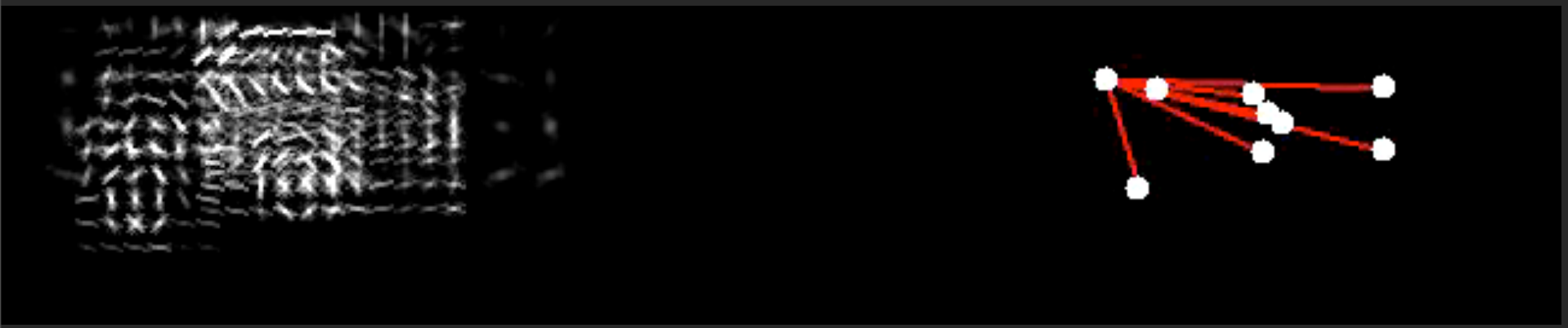
Initialization

Learn root filter with SVM

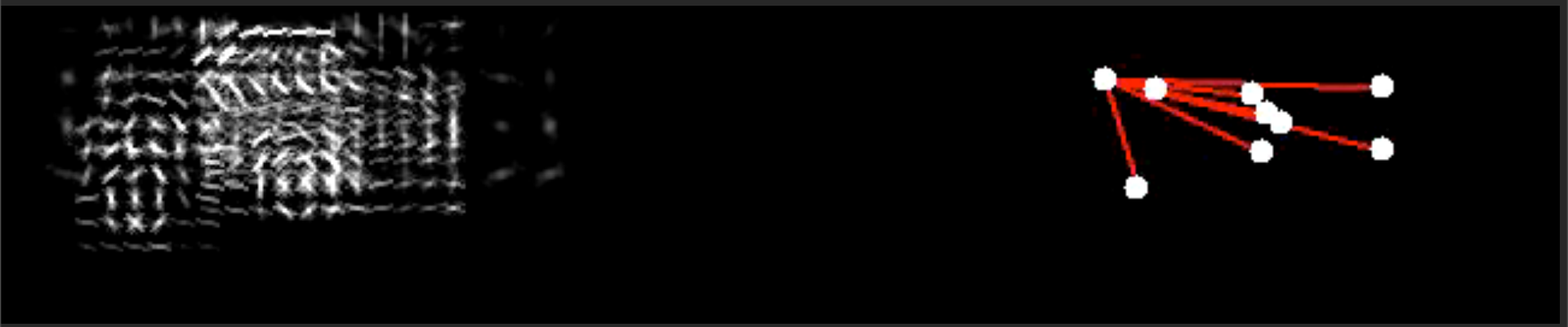
Initialize part filters to regions in root filter with lots of energy

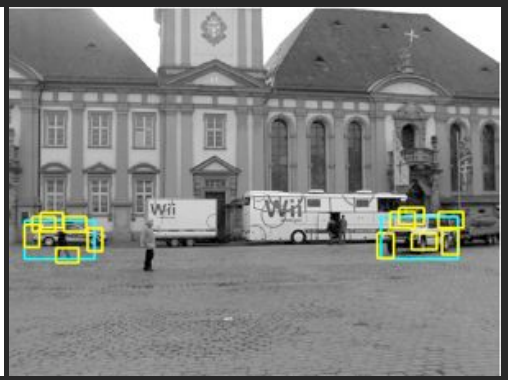


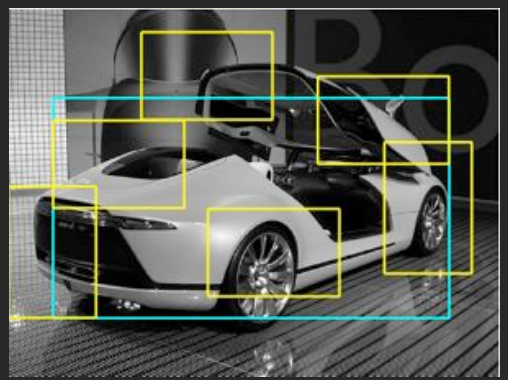
Example models

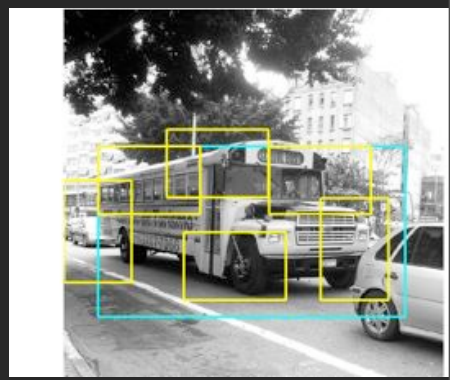


Example models

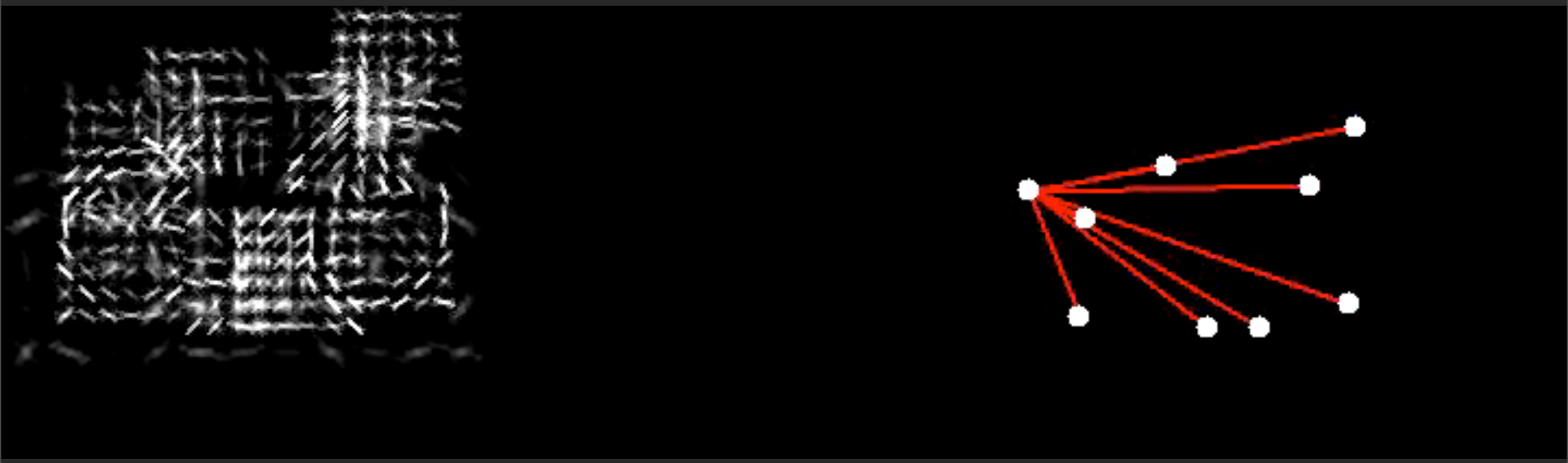




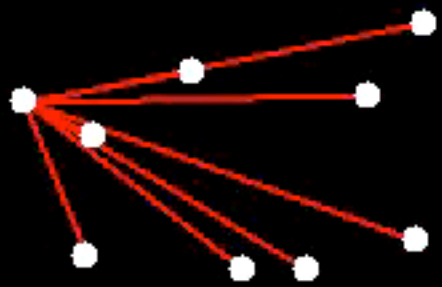


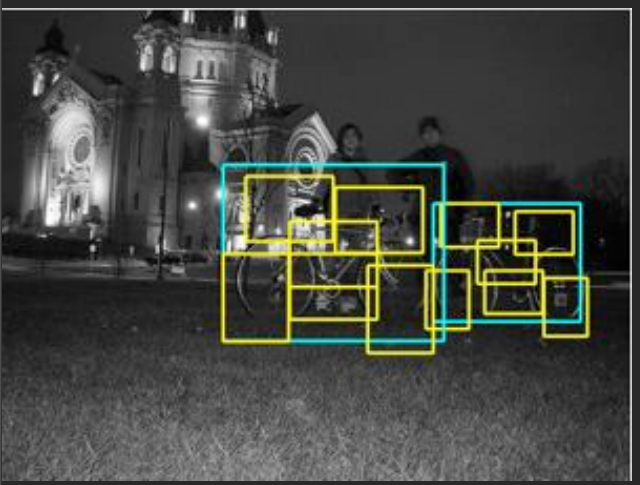


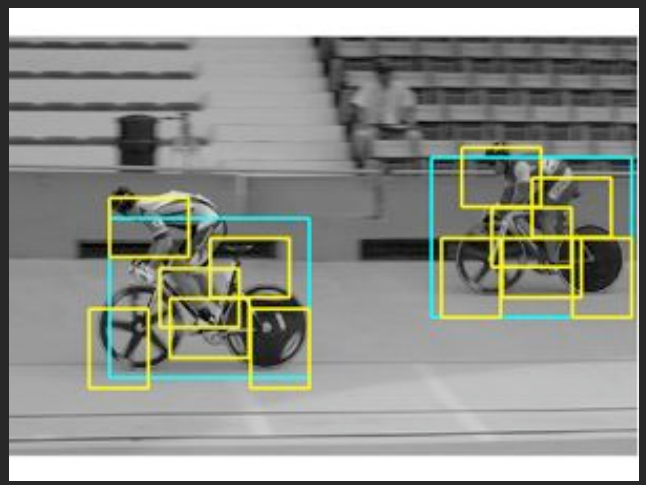
Example models



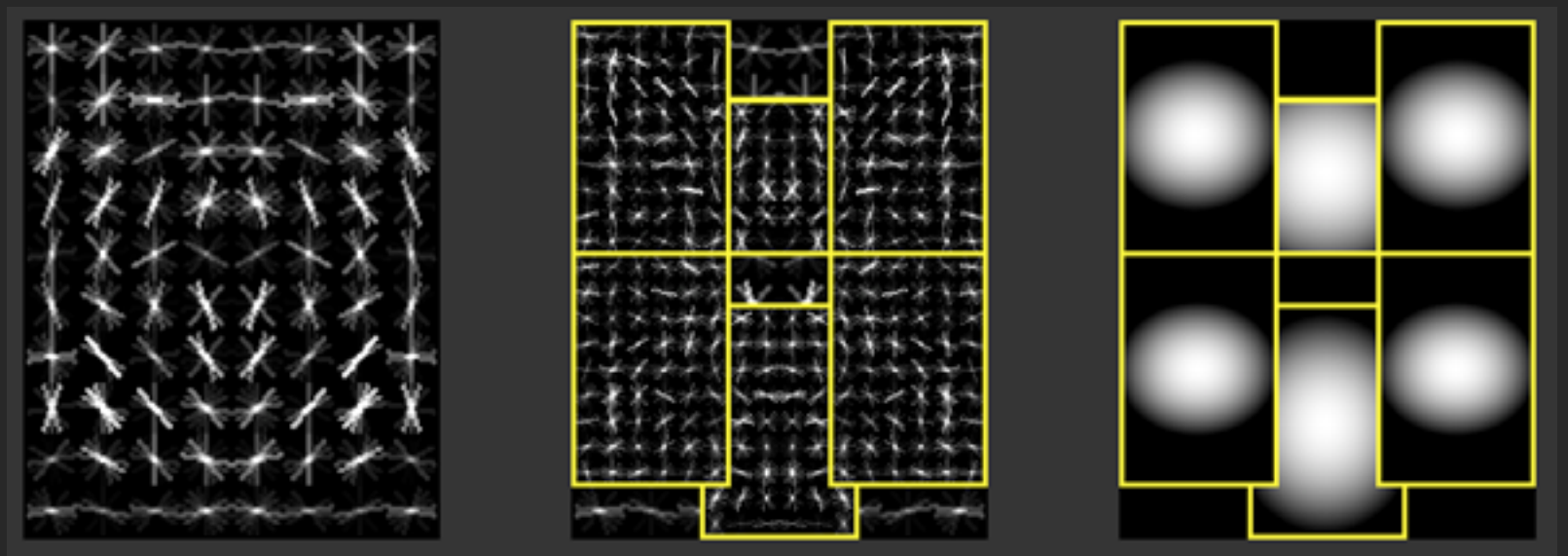
Example models



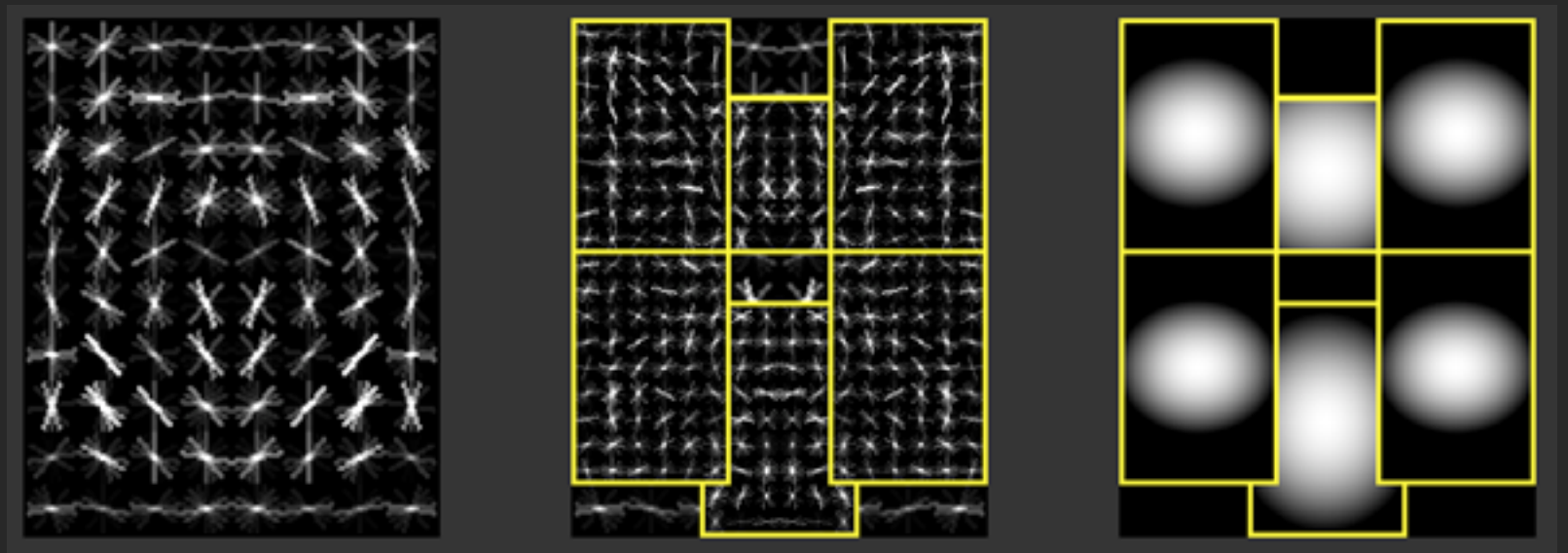




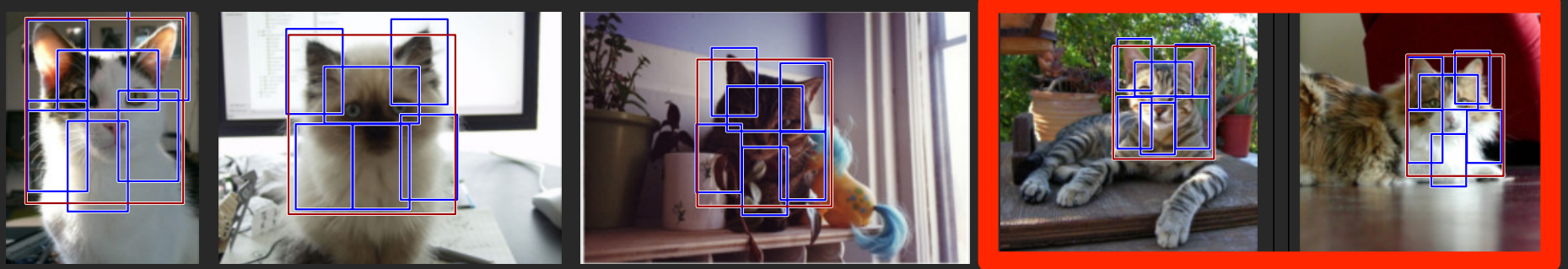
Example models



Example models

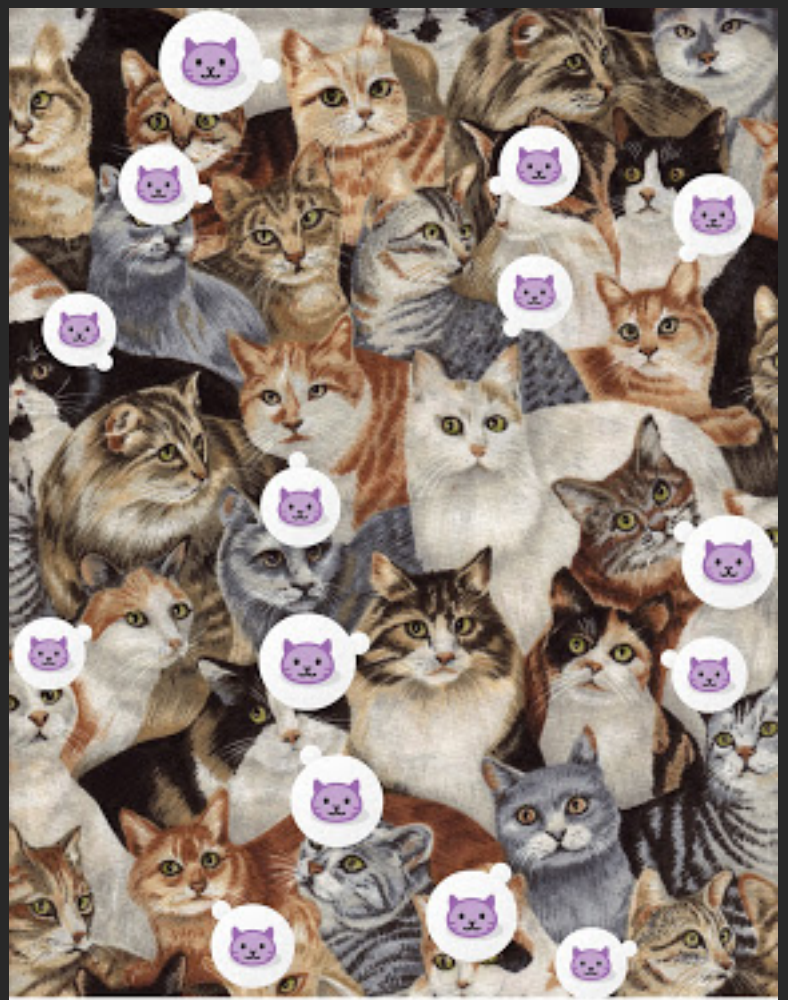


False positive due to imprecise bounding box

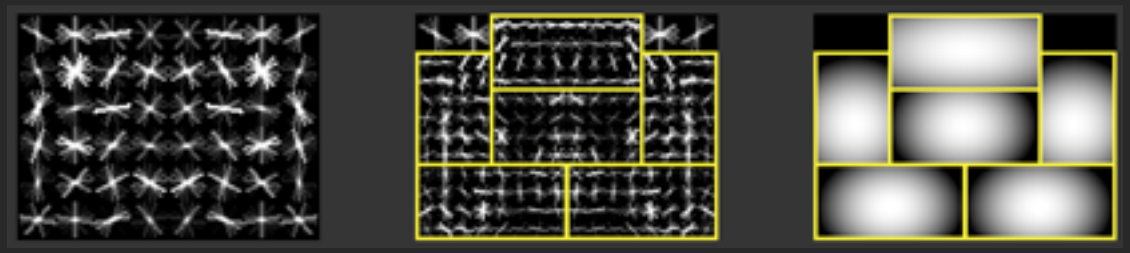


Google's Pet Emoticon Detector

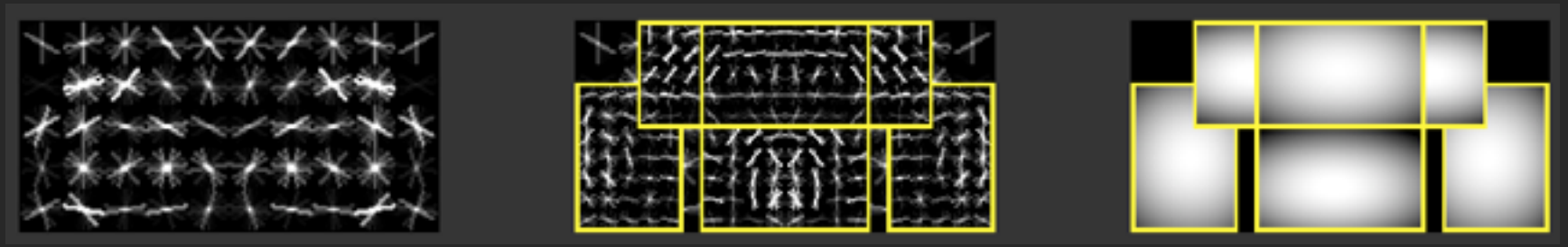
Our system!



Extensions: latent sub-categories



Frontal cars



Side / three-quarters view cars

Felzenswalb, Girshick, McAllester, and Ramanan *PAMI* 2010

Extensions: how do we find multiple objects?

Apply NMS to root scores after dynamic programming

Extensions: how do we find multiple objects?

Apply NMS to root scores after dynamic programming

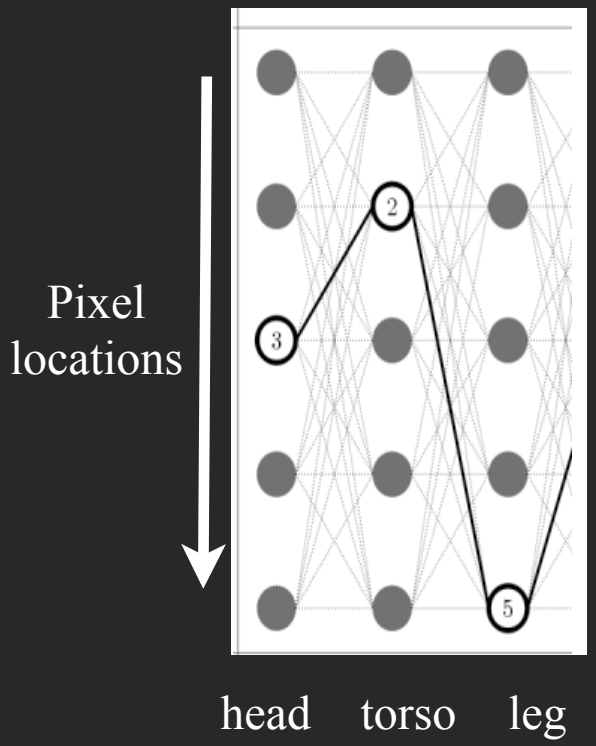
But will it work for ...?

Perhaps we want to use additional contextual information to resolve (global depth ordering, temporal info, etc...)

N-best decoding

Generate N high-scoring candidates with simple (tree) model, and evaluate with complex model

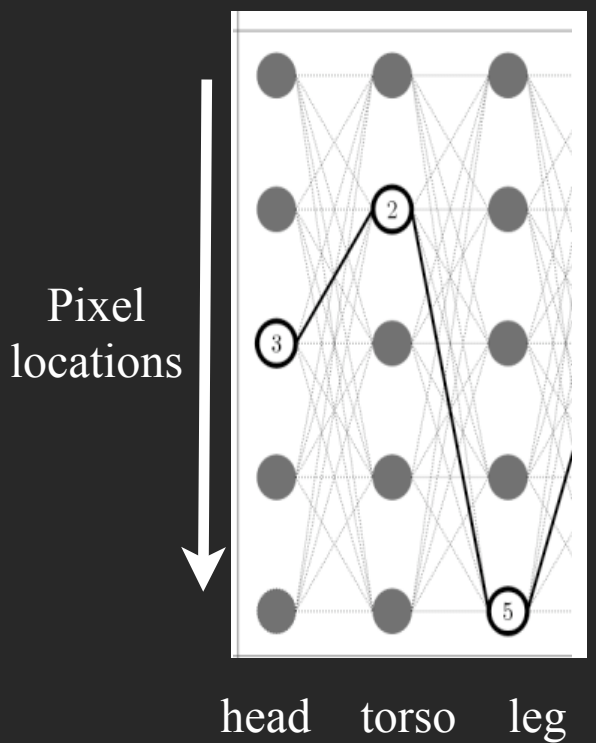
Popular in speech, but why not vision?



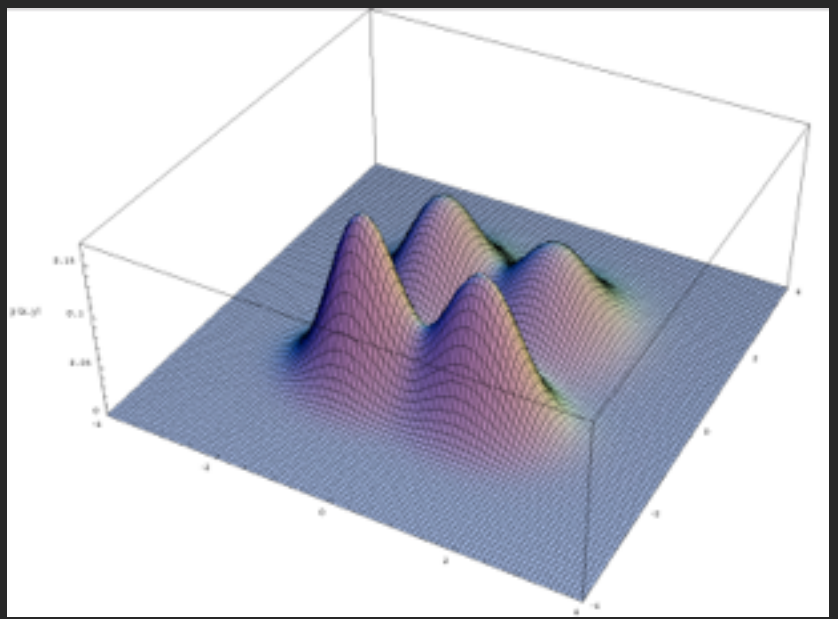
N-best decoding

Generate N high-scoring candidates with simple (tree) model, and evaluate with complex model

Popular in speech, but why not vision?



N-best maximal decoding

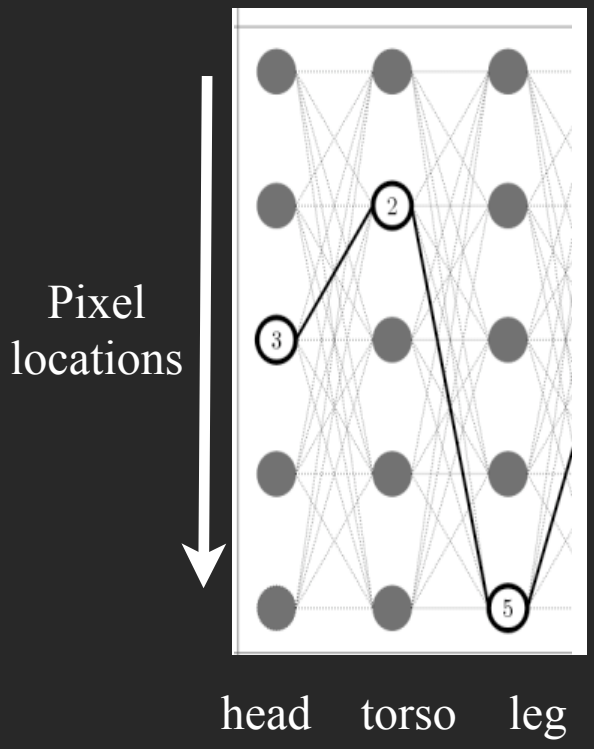


N-best with “NMS” or “mode-finding”

Park and Ramanan, ICCV11

Yadollahpour et al. ECCV12

N-best maximal decoding

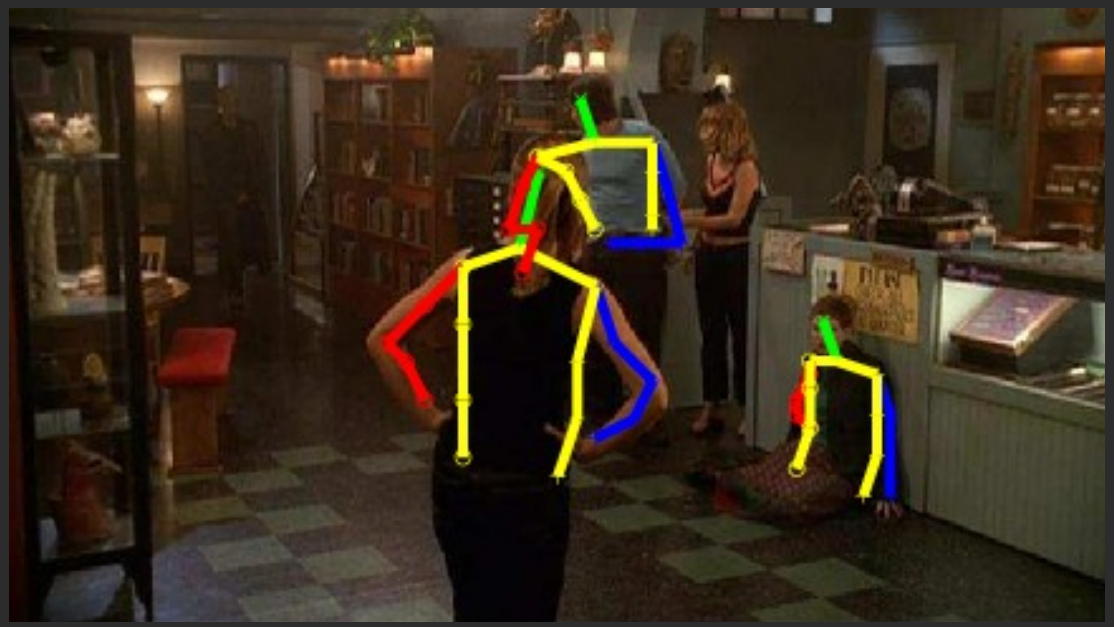


Intuition: backtrack from all part "max-marginals", not just root

(can we done without any noticeable increase in computation)

Park and Ramanan, ICCV 2011

N-best maximal decoding



Philosophy: Delay hard decisions as much as possible

Candidate interest points

Candidate parts

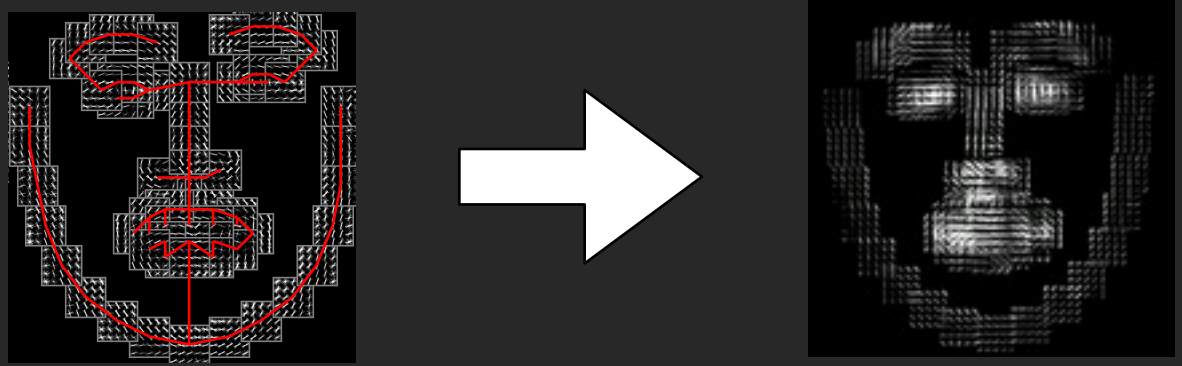
Candidate poses

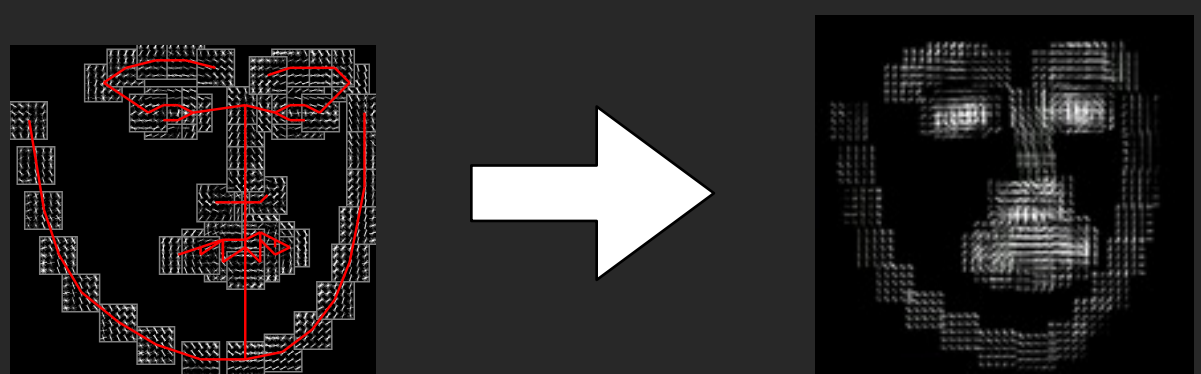
A look back: part models as mixture models

$$S(x,z) = \sum_i w_i \cdot \phi(x, z_i) + \sum_{ij \in E} w_{ij} \cdot \psi(z_i, z_j)$$

Each distinct placement of parts yields a unique global template

$$S(x,z) = w_z \cdot x + b_z$$

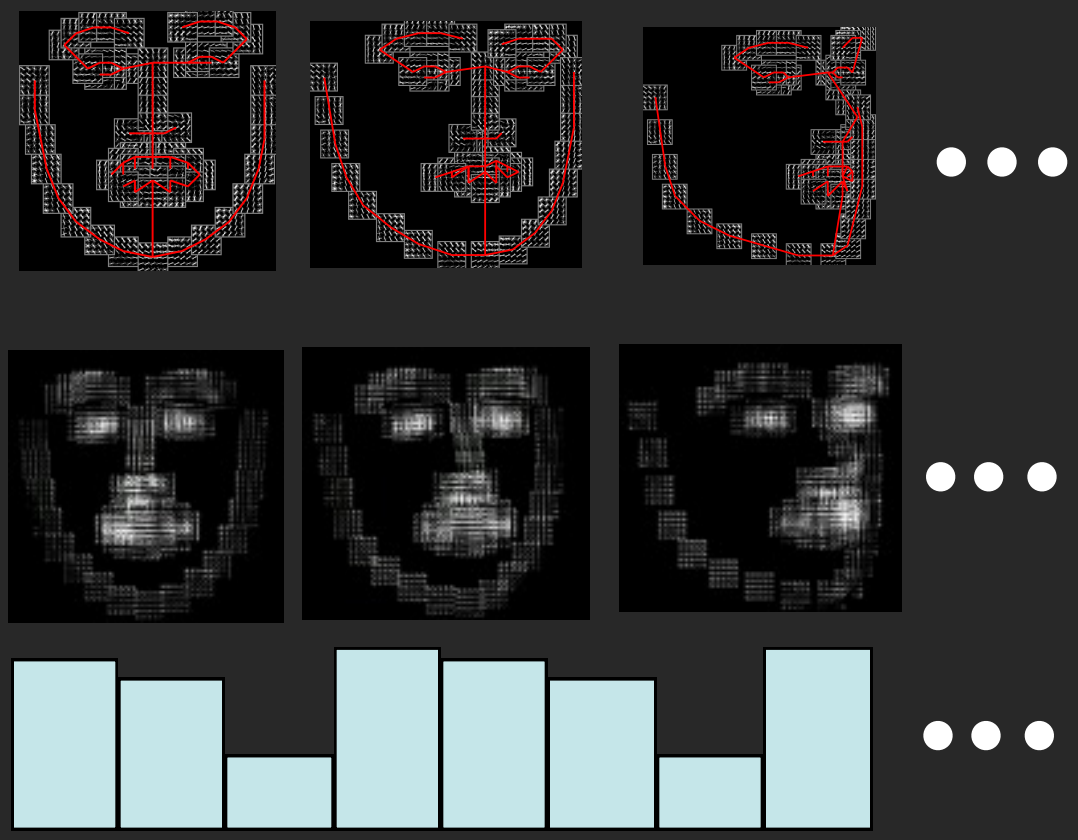




Parts as mixture models

Spatial model defines bias or “prior”

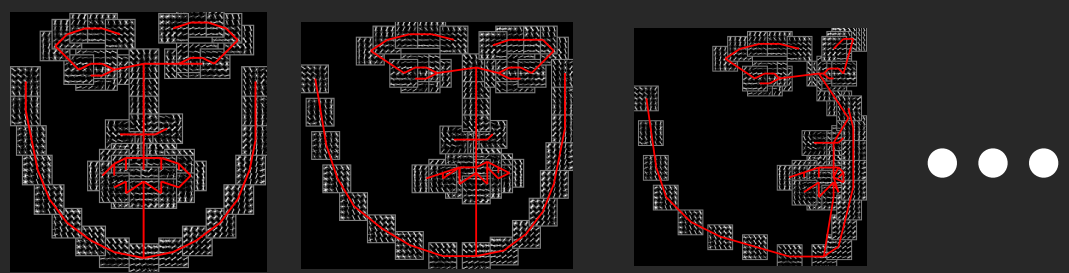
$$f(x) = \max_{z \in Z} w_z \cdot x + b_z$$

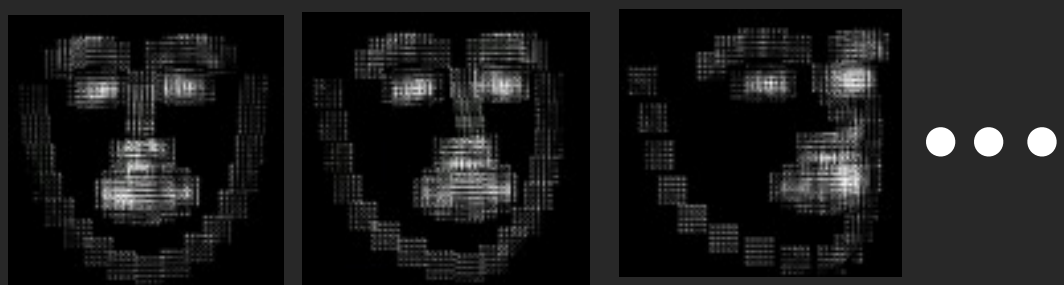


Parts as mixture models

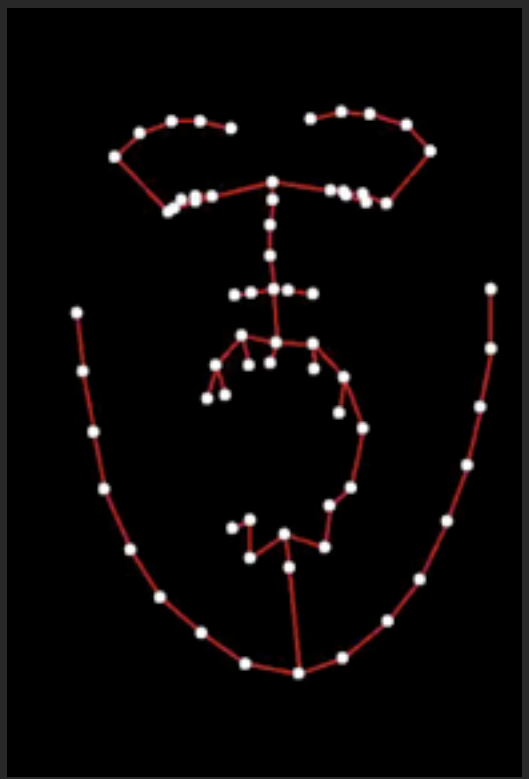
Part models allow us to represent an exponentially-large family of global templates

$$f(x) = \max_{z \in Z} w_z \cdot x + b_z$$

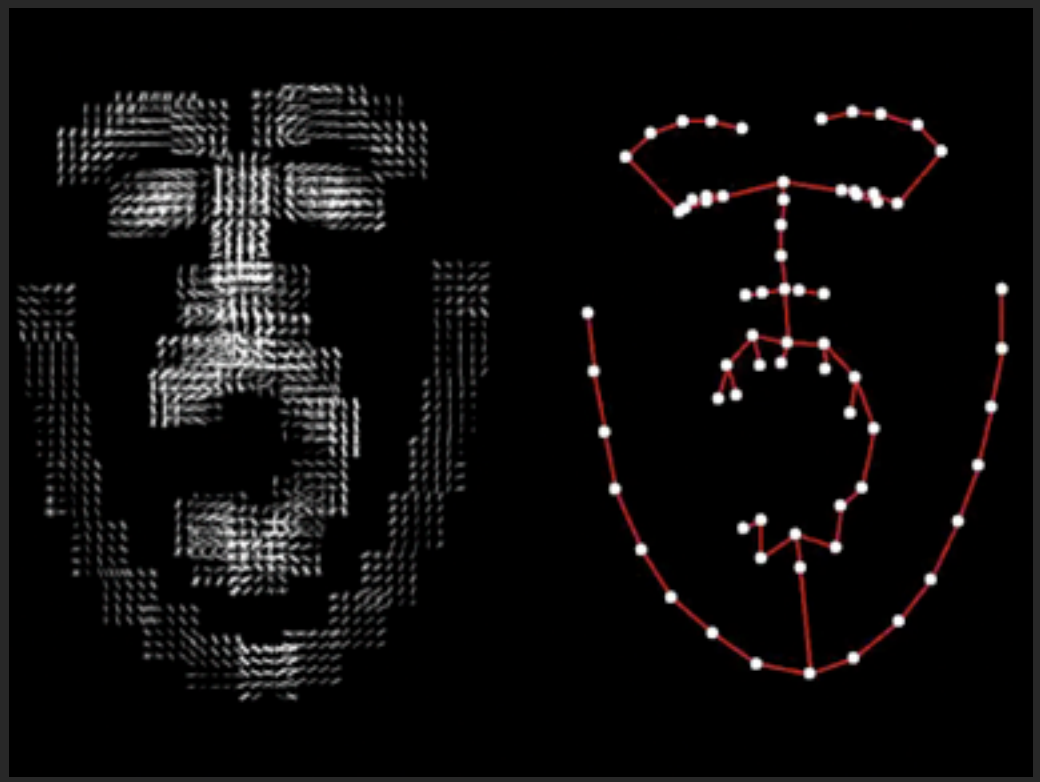




Deformation modes



Deformation modes



DPMs as large-mixture models

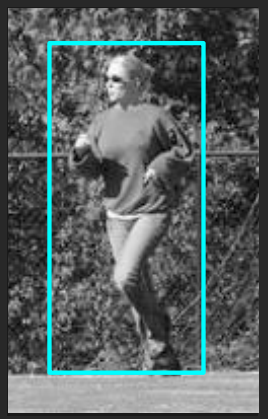
$$f(x) = \max_{z \in Z} w_z \cdot x + b_z$$

- "Double-counting" manifests simply as too strong of a weight

- Suggests jointly learning parts is crucial

(more on that this afternoon)

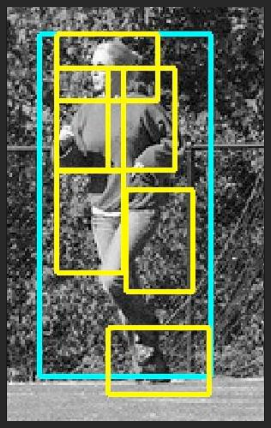
Revisit latent (vs linear) classification



$$f_w(x) = w \cdot \Phi(x)$$

Score is linear in x

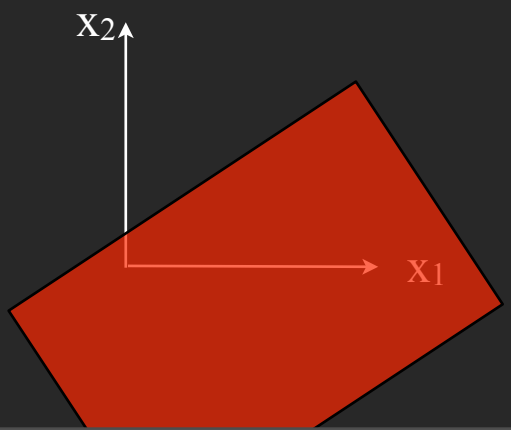
Positive set $\{x: f_w(x) > 0\}$ is half-space

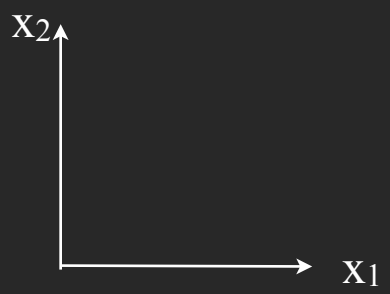


$$f_w(x) = \max_z w_z \cdot x$$

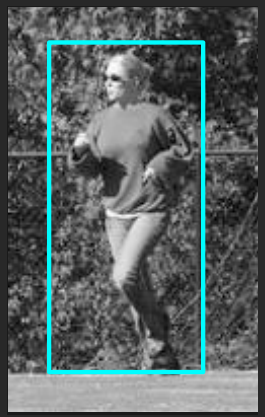
Score is ?

Positive set is ?





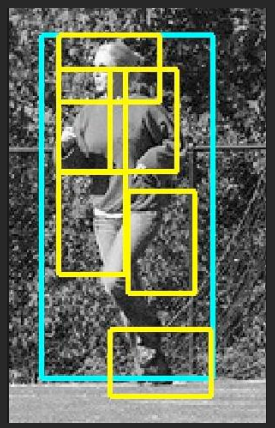
Revisit latent (vs linear) classification



$$f_w(x) = w \cdot \Phi(x)$$

Score $f_w(x)$ is linear in x

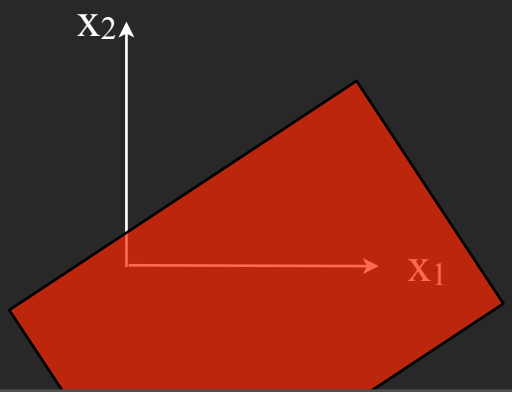
Positive set $\{x: f_w(x) > 0\}$ is half-space

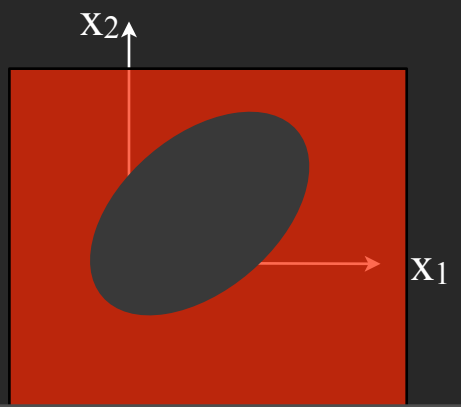


$$f_w(x) = \max_z w_z \cdot x$$

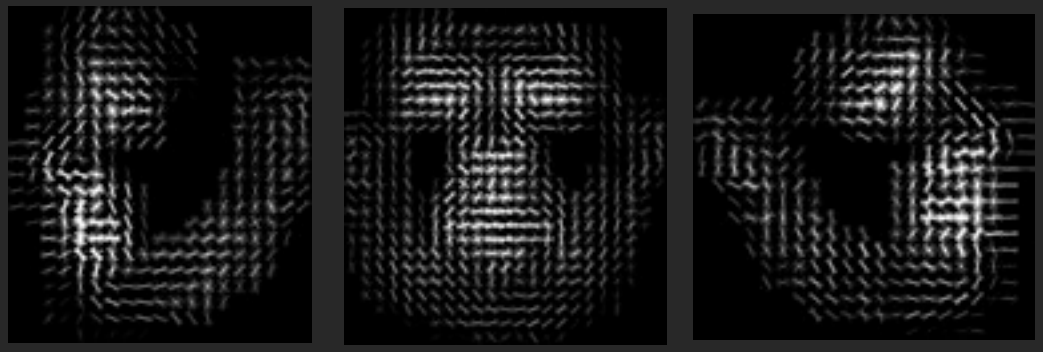
Score $f_w(x)$ is convex in x

Positive set $\{x: f_w(x) > 0\}$ is concave

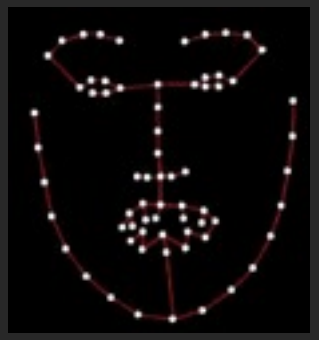




DPMs vs explicit mixtures



Mixtures of rigid templates

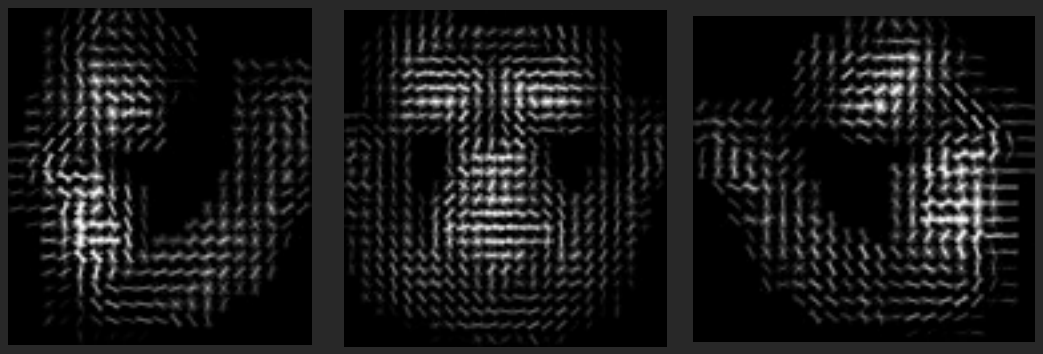


Part model

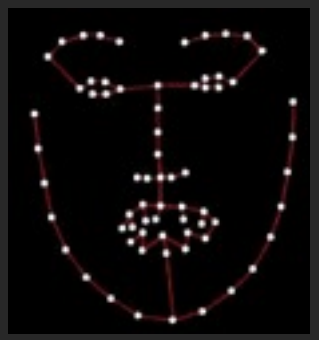
"Exemplar SVMs"

Malisiewicz et al ICCV 11

DPMs vs explicit mixtures



Mixtures of rigid templates



Part model

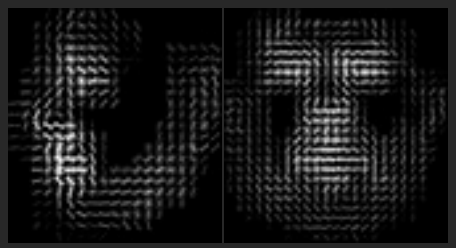
"Exemplar SVMs"

Malisiewicz et al ICCV 11

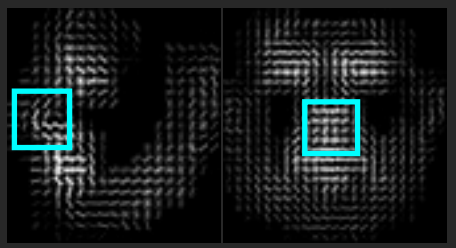
Compared to a mixture of exemplars, part models...

1) Share parameters across templates
2) Synthesize new templates not seen during training
3) Efficiently search over templates using dynamic programming

DPMs vs explicit mixtures



Mixtures of rigid templates



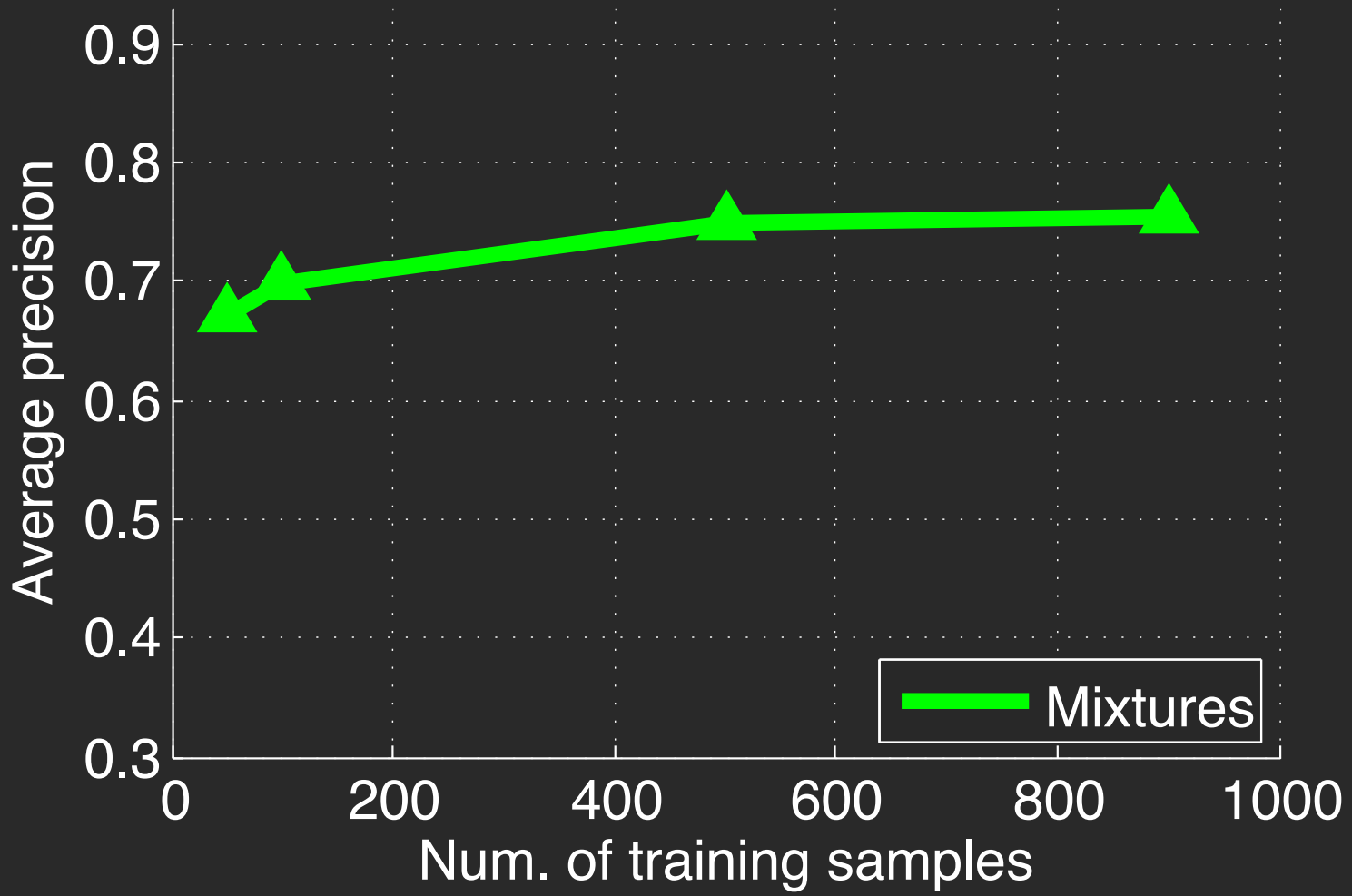
Mixtures of rigid templates with tied parameters (given by parts)

Part model

1) Share parameters across mixtures
2) "Synthesize" new rigid templates not seen during training

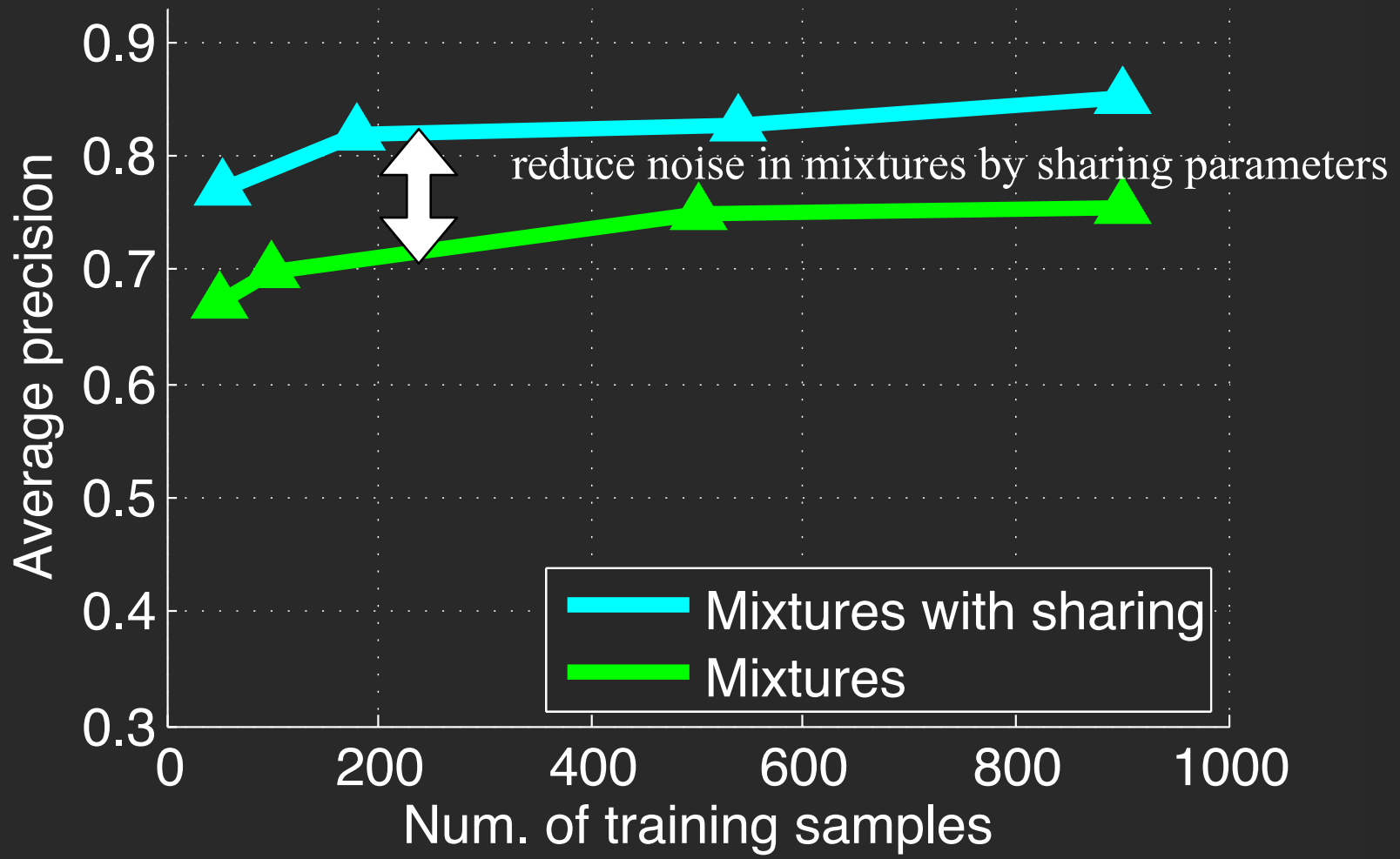
To examine (1) vs (2), lets define mixture of exemplars with sharing

An analysis of part models



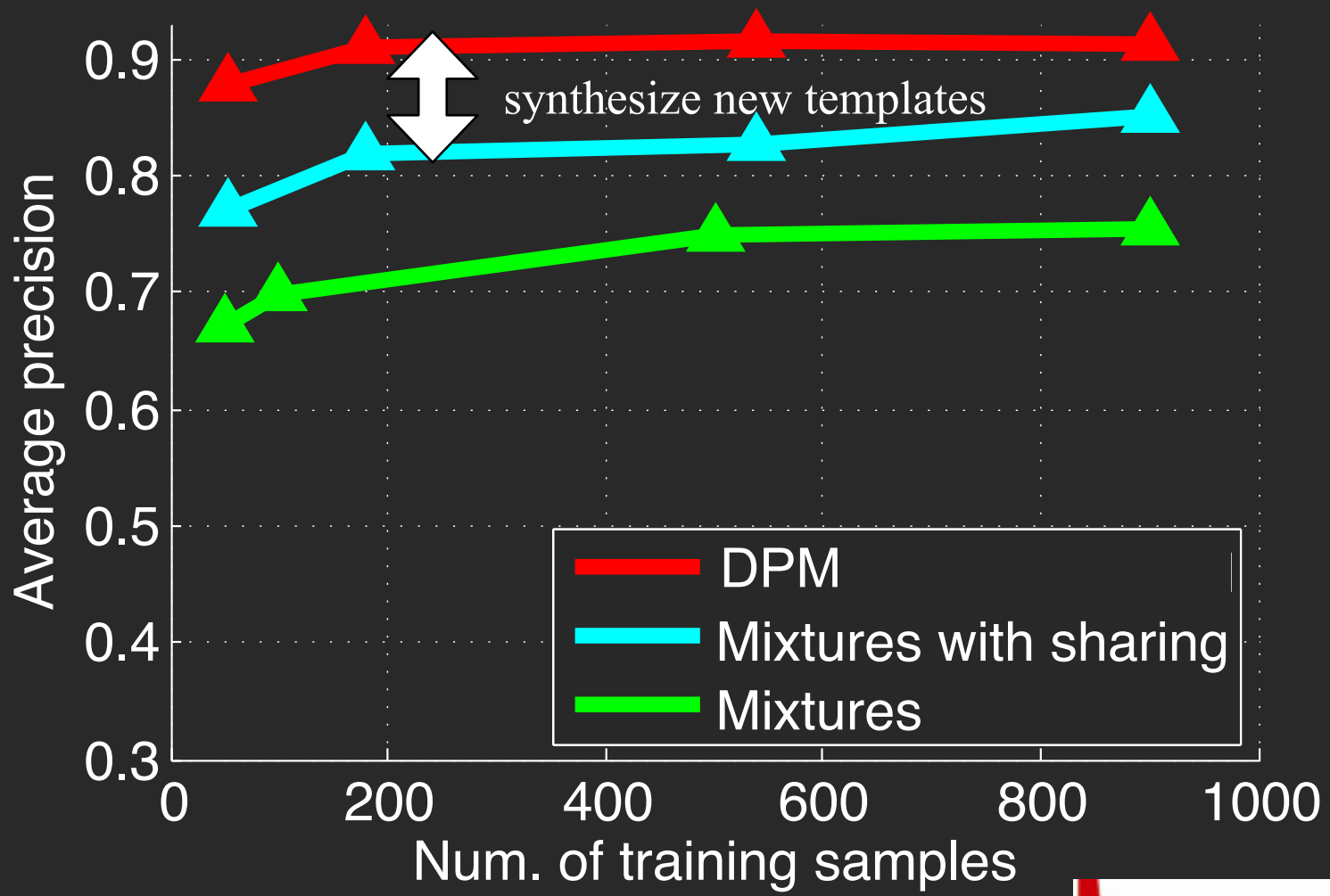
Zhu, Vondrick, Ramanan & Fowlkes,
"Do we need more training data or better models?"
BMVC 2012

An analysis of part models



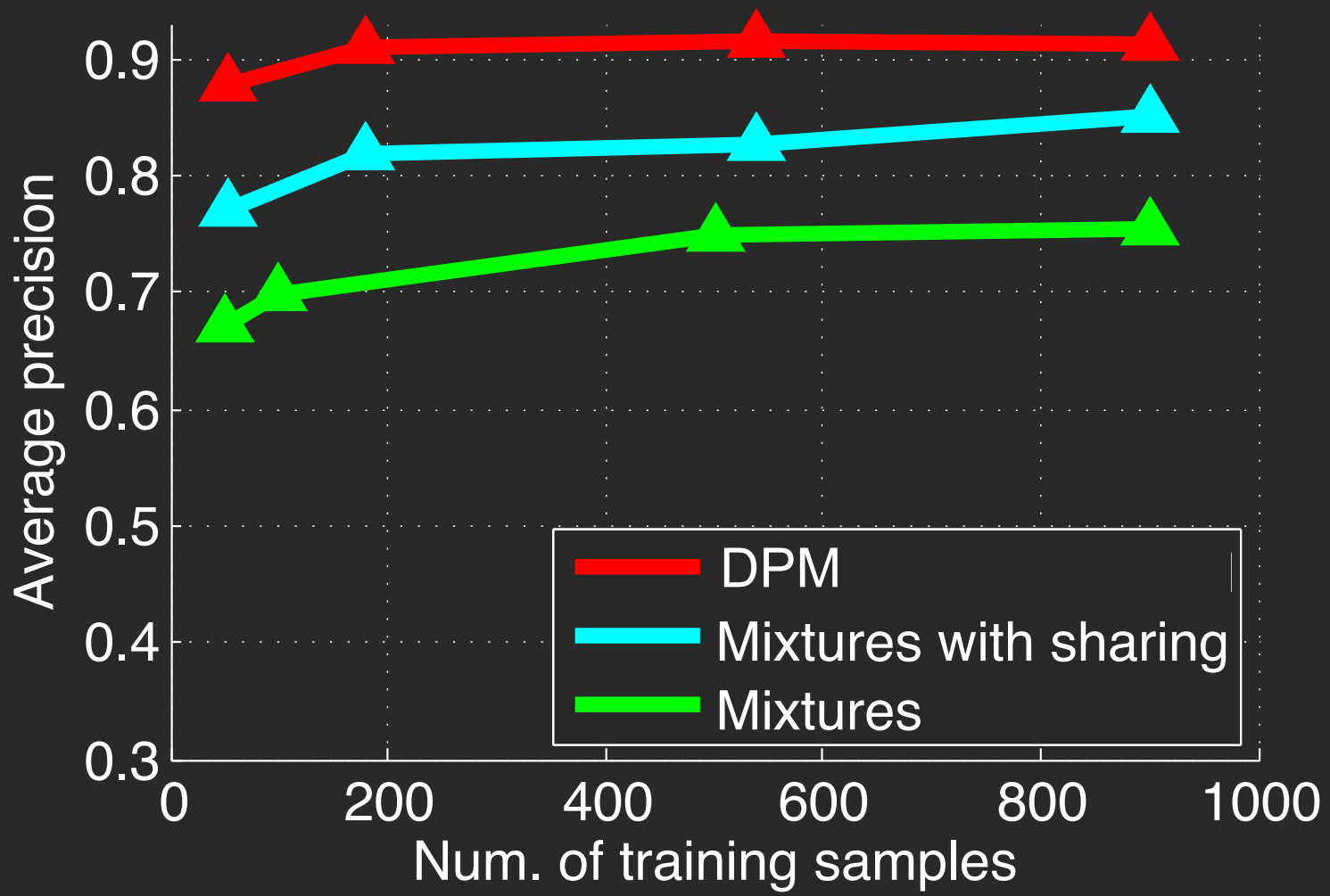
Zhu, Vondrick, Ramanan & Fowlkes,
"Do we need more training data or better models?"
BMVC 2012

An analysis of part models



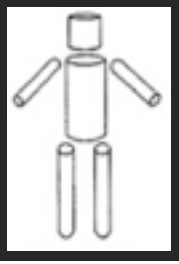
"Synthesis" of unseen (rare) templates is even more beneficial than sharing

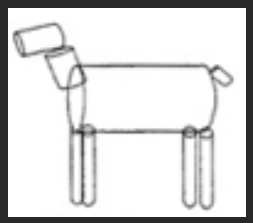
An argument against "big-data"

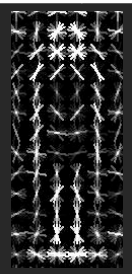


One can train a state-of-art face detector (*c.f.* Google Picassa & Facebook's face.com) with 100 faces!

A look back







Geometric statistical models