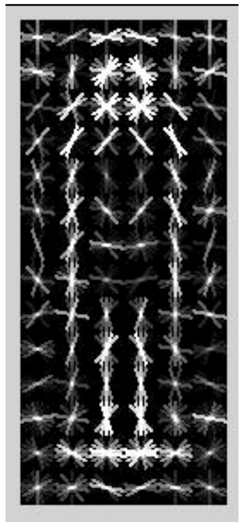
Parts

Limits of templates



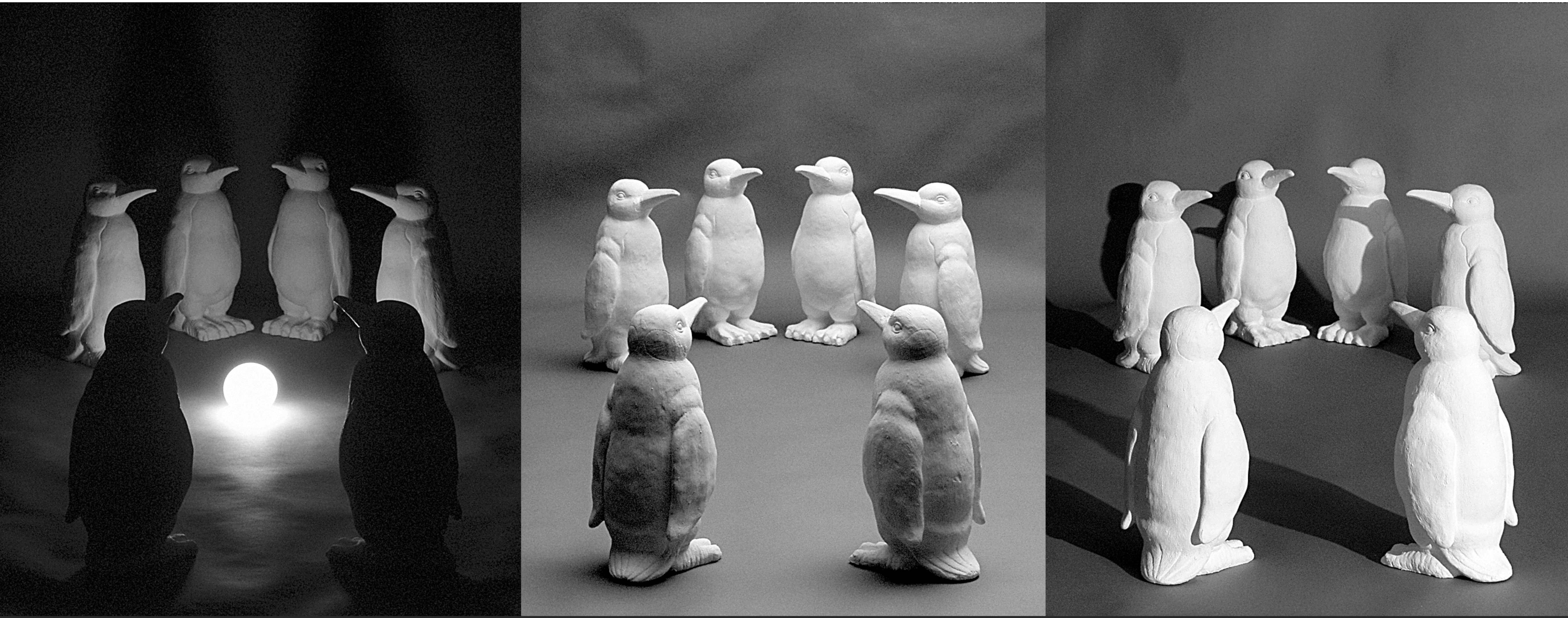
How to model large variations in appearance?

This is generally regarded as a "central challenge" for recognition

Challenges: viewpoint variation

Michelangelo 1475-1564

Challenges: illumination



Challenges: scale

Challenges: background clutter

Kilmeny Niland. 1995

Challenges: intra-class variation






Why is finding people difficult?

variation in illumination

variation in appearance

variation in pose, viewpoint

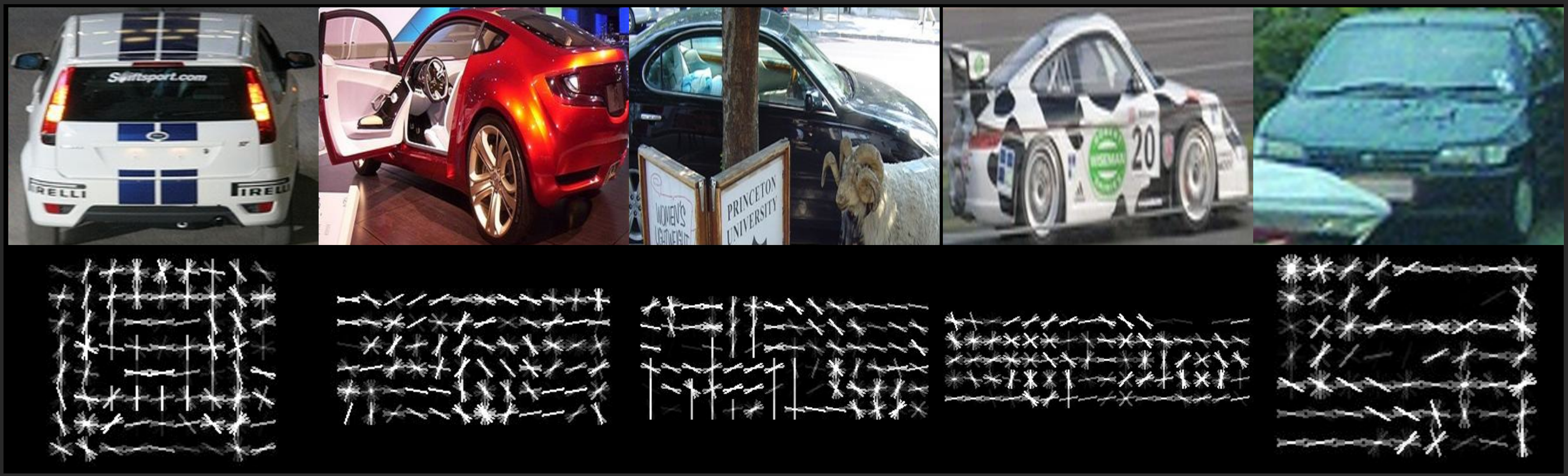
occlusion & clutter

Classic “nuisance factors” for general object recognition

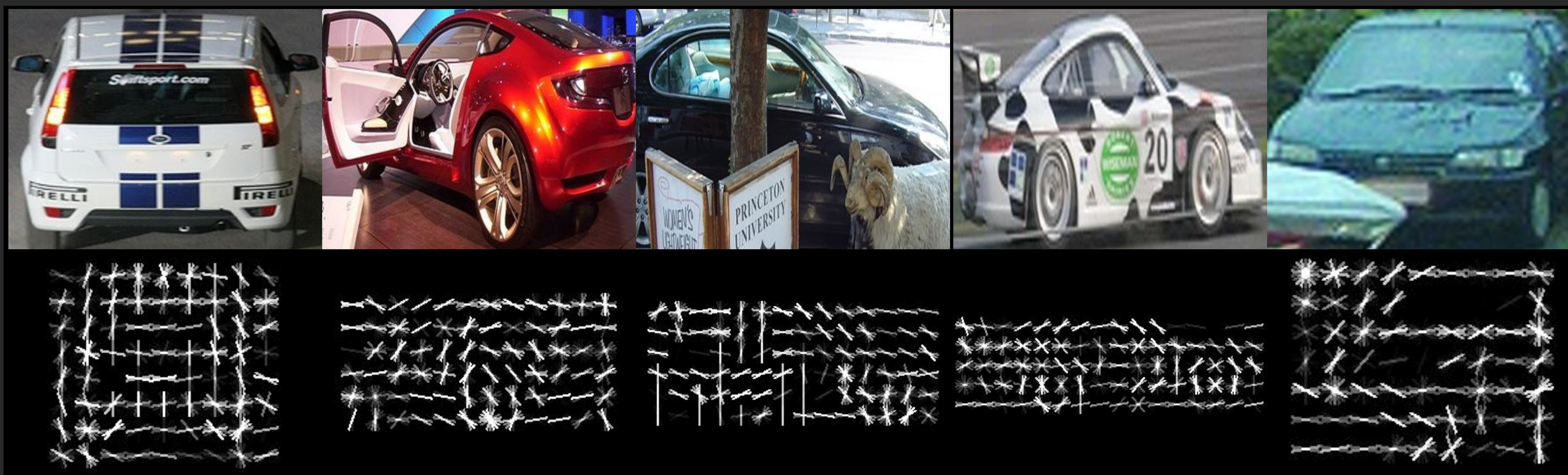
“Sub”categories

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

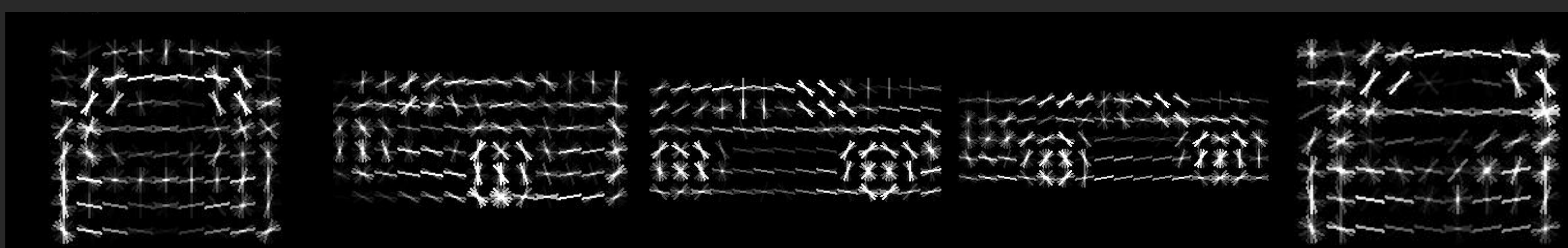
Why not treat each positive example as a unique subcategory?



Why not treat each positive example as a unique subcategory?

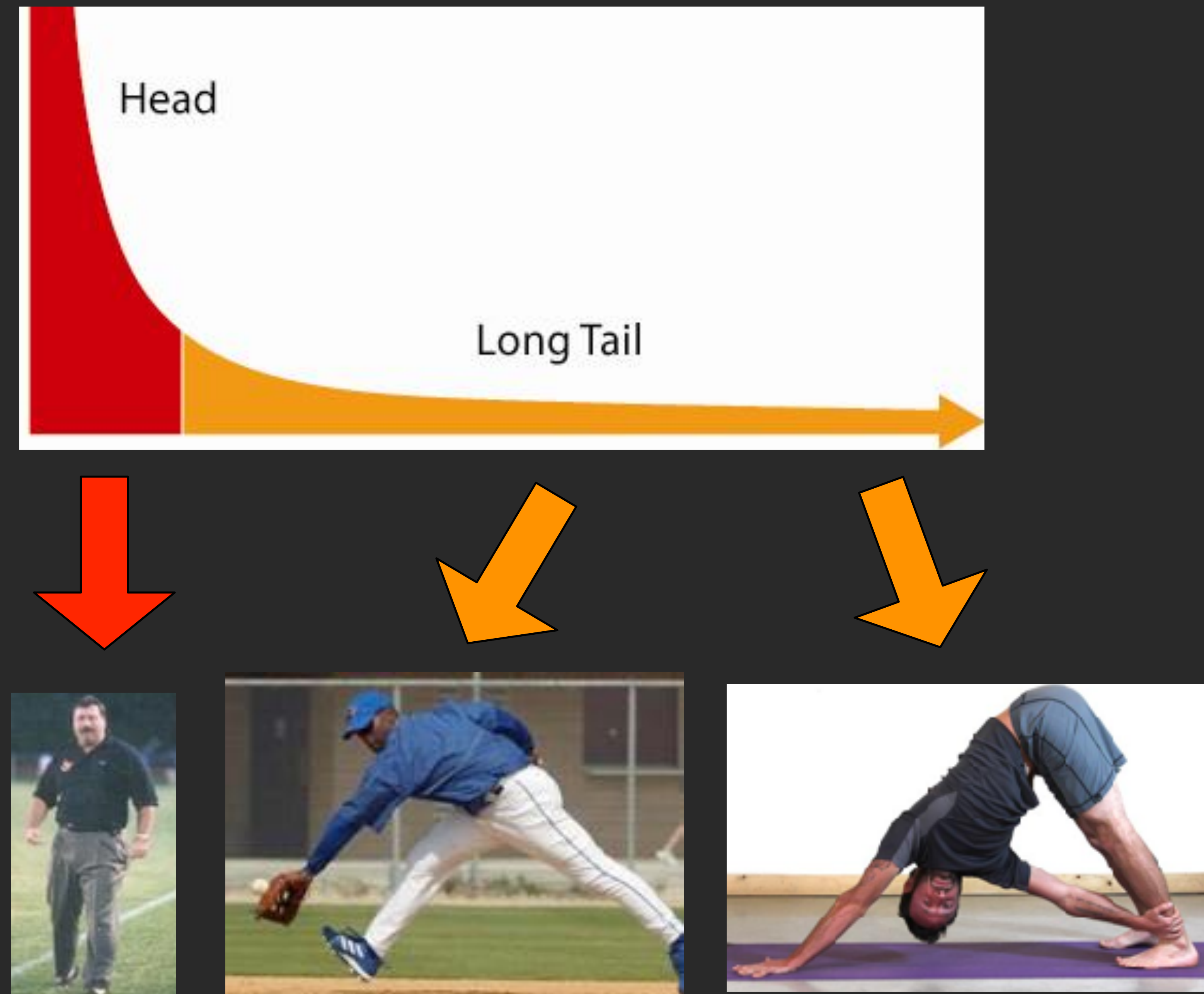


Single positive example



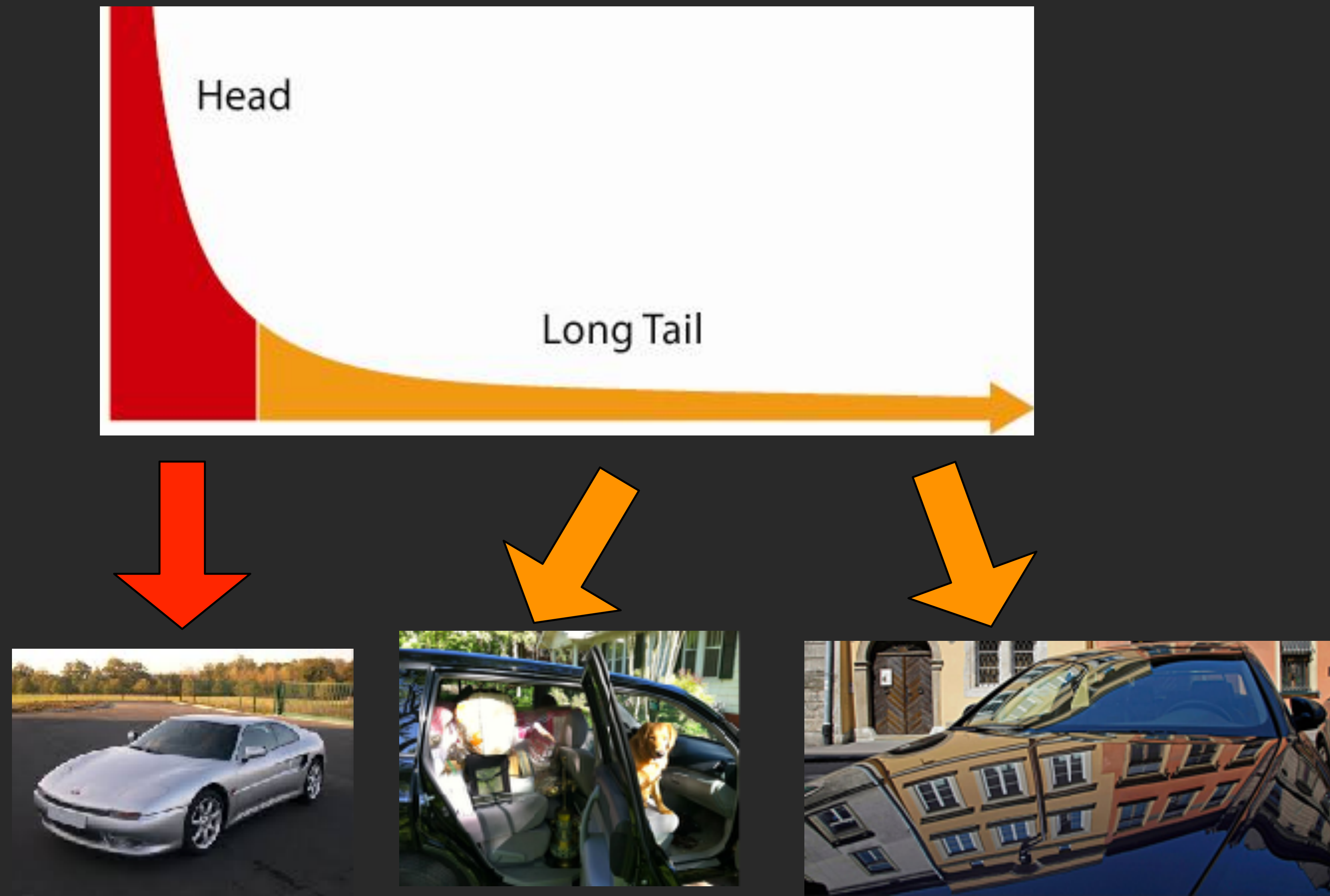
Average of 50 closest

But how to handle...



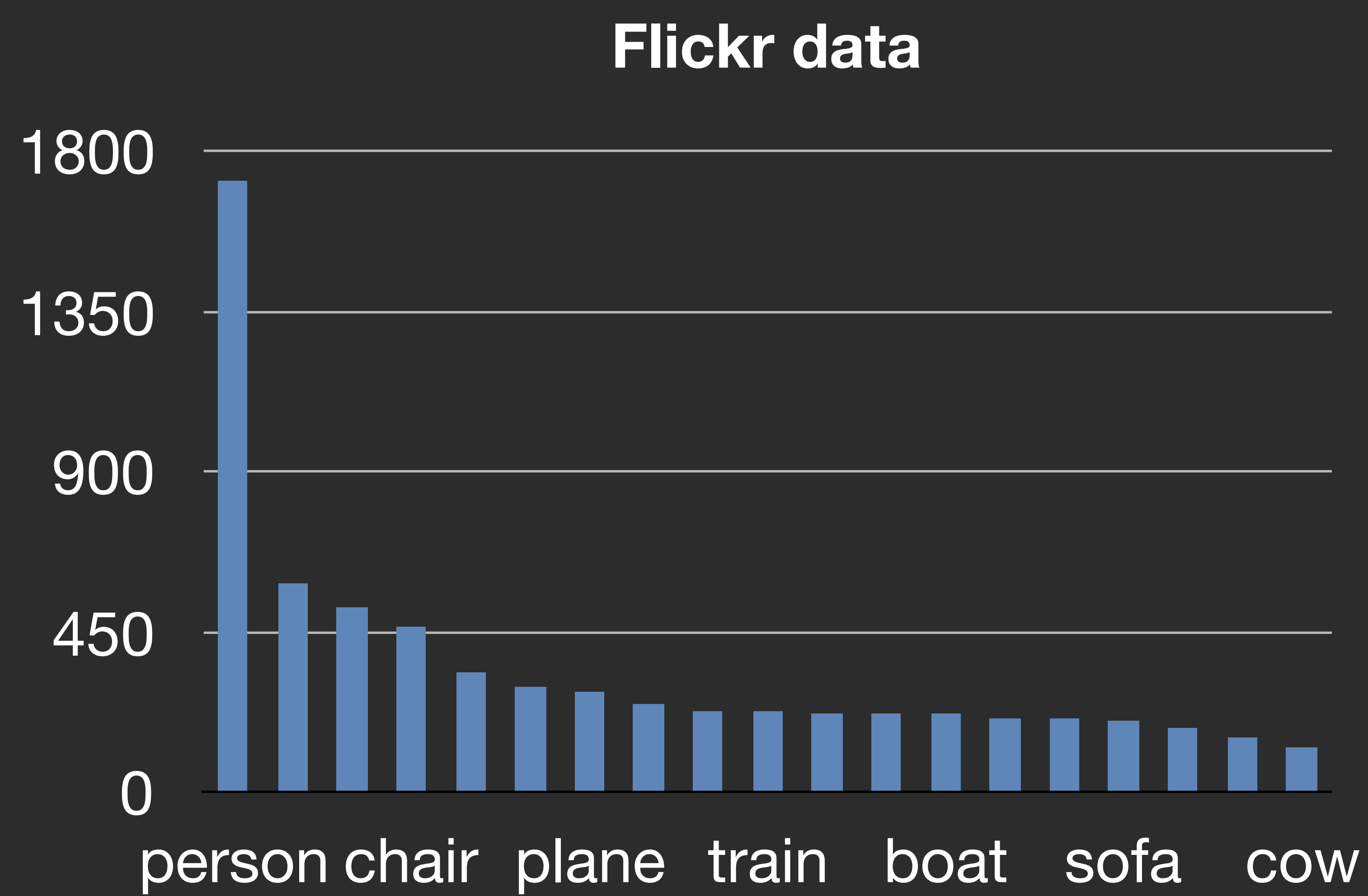
We need lots of templates, but will likely have little data of ‘twisted’ poses

But how to handle...

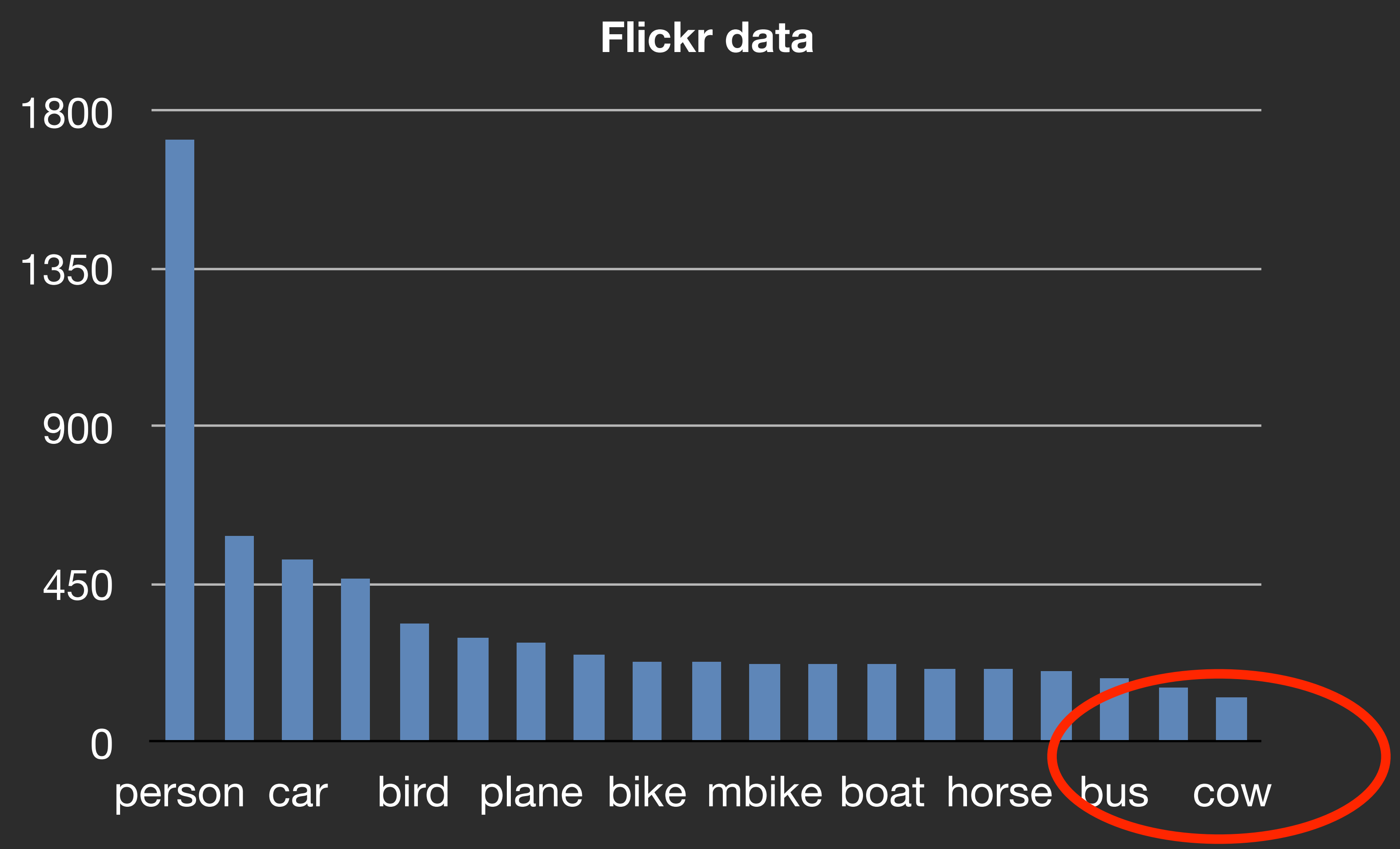


We need lots of templates, but will likely have little data of ‘rare’ car-appearances

Difficulties: long tails

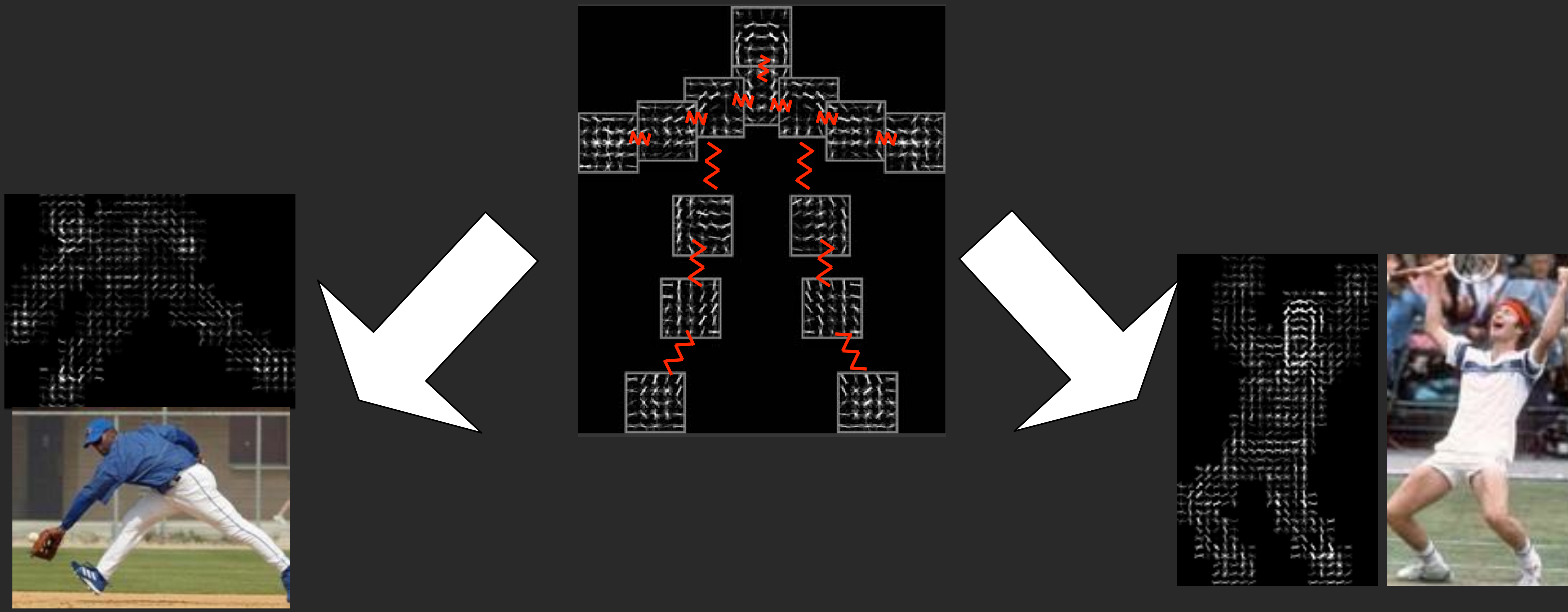


Difficulties: long tails

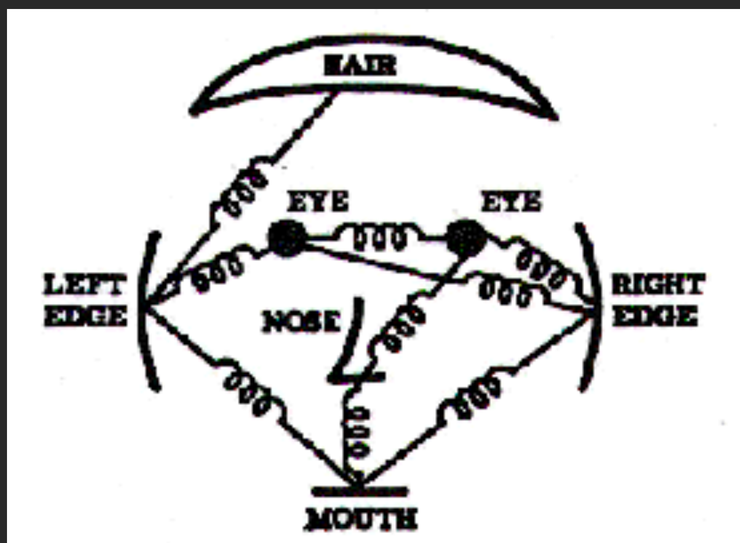


“One-shot learning”: sharing

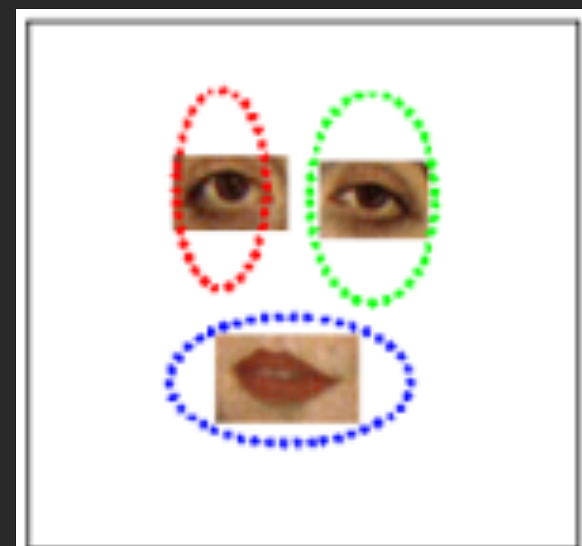
Parts to the rescue!



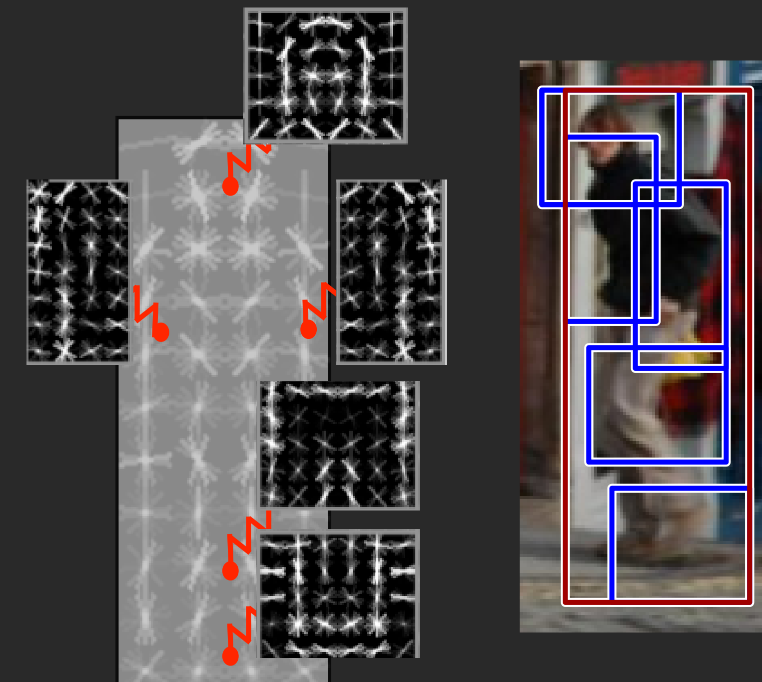
History over 40 years



Pictorial structures



Constellation models

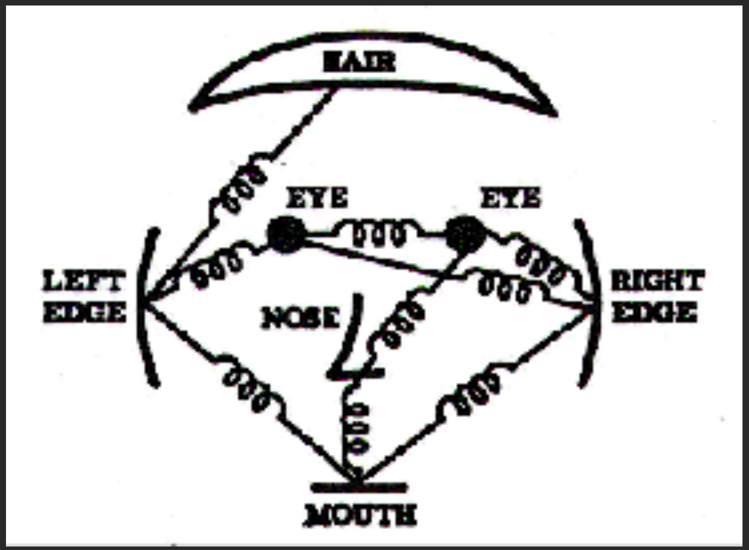


Deformable part models

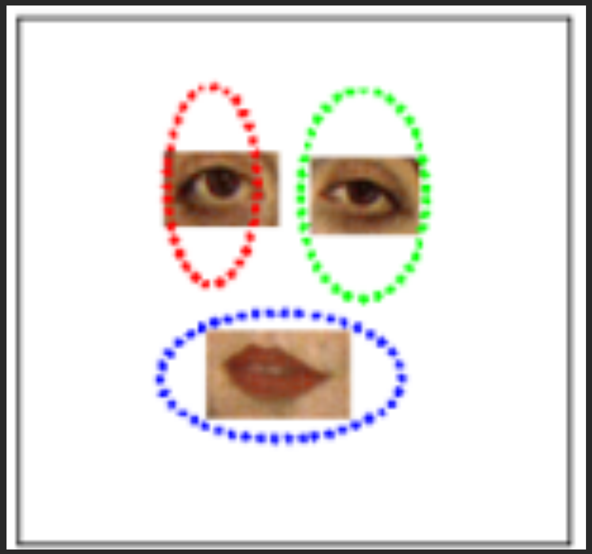
Model encodes local appearance + pairwise geometry

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)

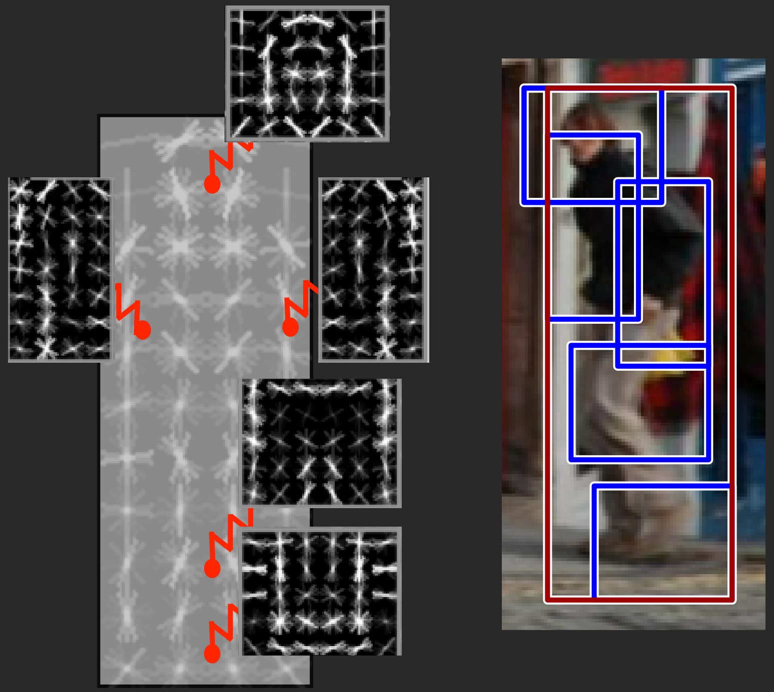
Part models



Pictorial structures



Constellation models

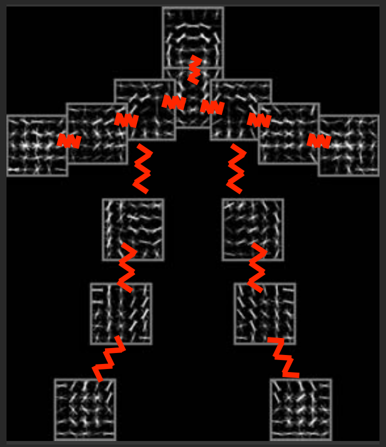


Deformable part models

I'll talk about DPMs, but give an alternate "long tail" perspective

Felzenszwalb, Girshick, McAllester, Ramanan CACM 2013

Scoring function



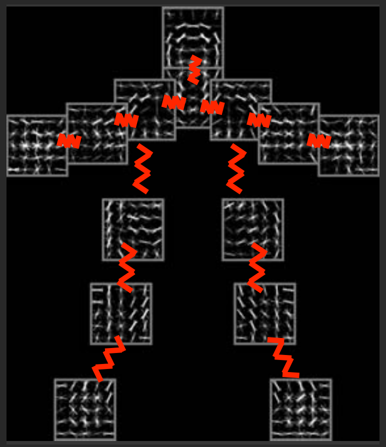
$$S(x, z) =$$

$$x = \text{image}$$

$$z_i = (x_i, y_i)$$

$$z = \{z_1, z_2, \ldots\}$$

Scoring function



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

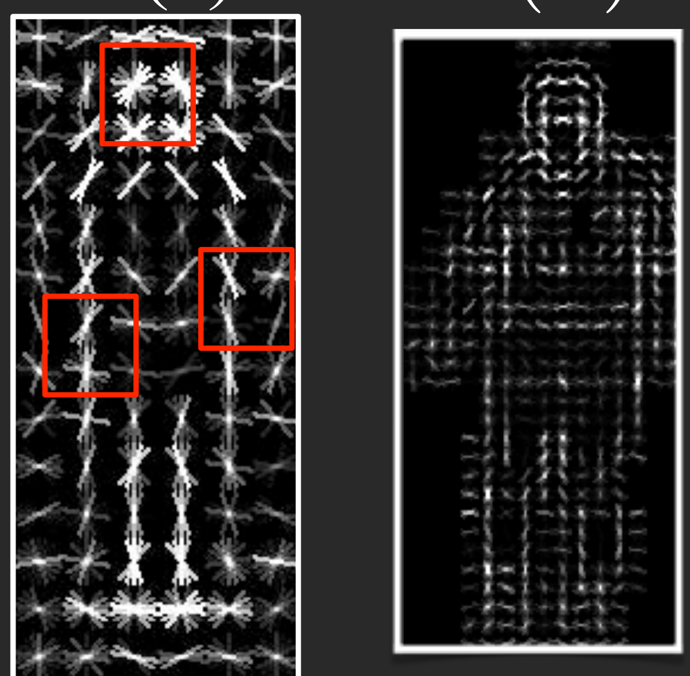
$$x = \text{image}$$
$$z_i = (x_i, y_i), \quad i \in \{\text{head,elbow}, \ldots\}$$
$$z = \{z_1, z_2, \ldots\}$$

(often the scoring function includes an additional “spring term”; let’s ignore for now)

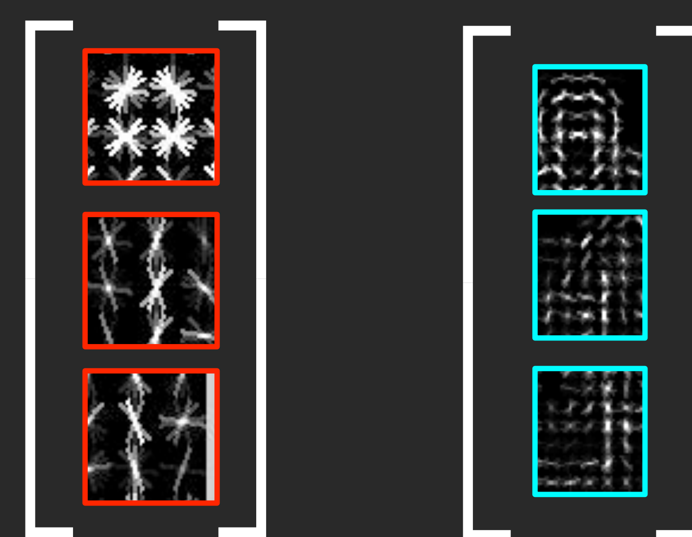
Alternative formulations

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

w(z) $\quad\Phi(x)$



w $\quad\Phi(x,z)$



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

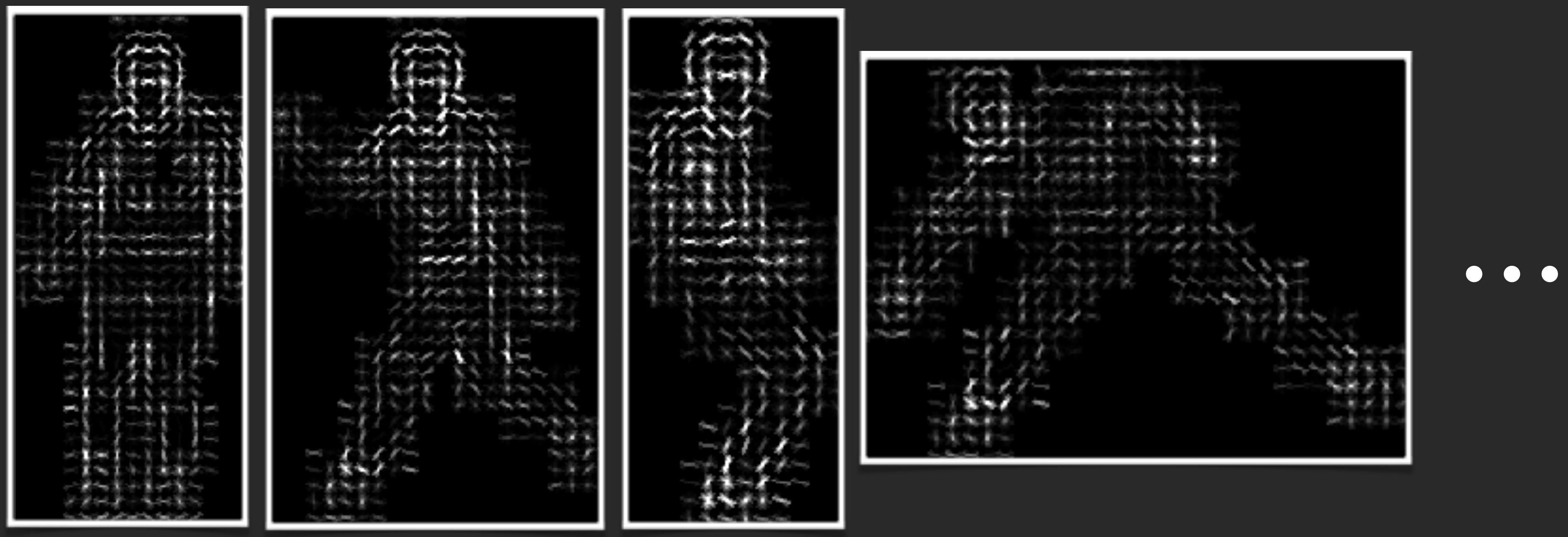
$$S(x,z) = w \cdot \Phi(x,z)$$

[Useful for visualizing model]

[Useful for learning model parameters]

Visualizing family of classifiers

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

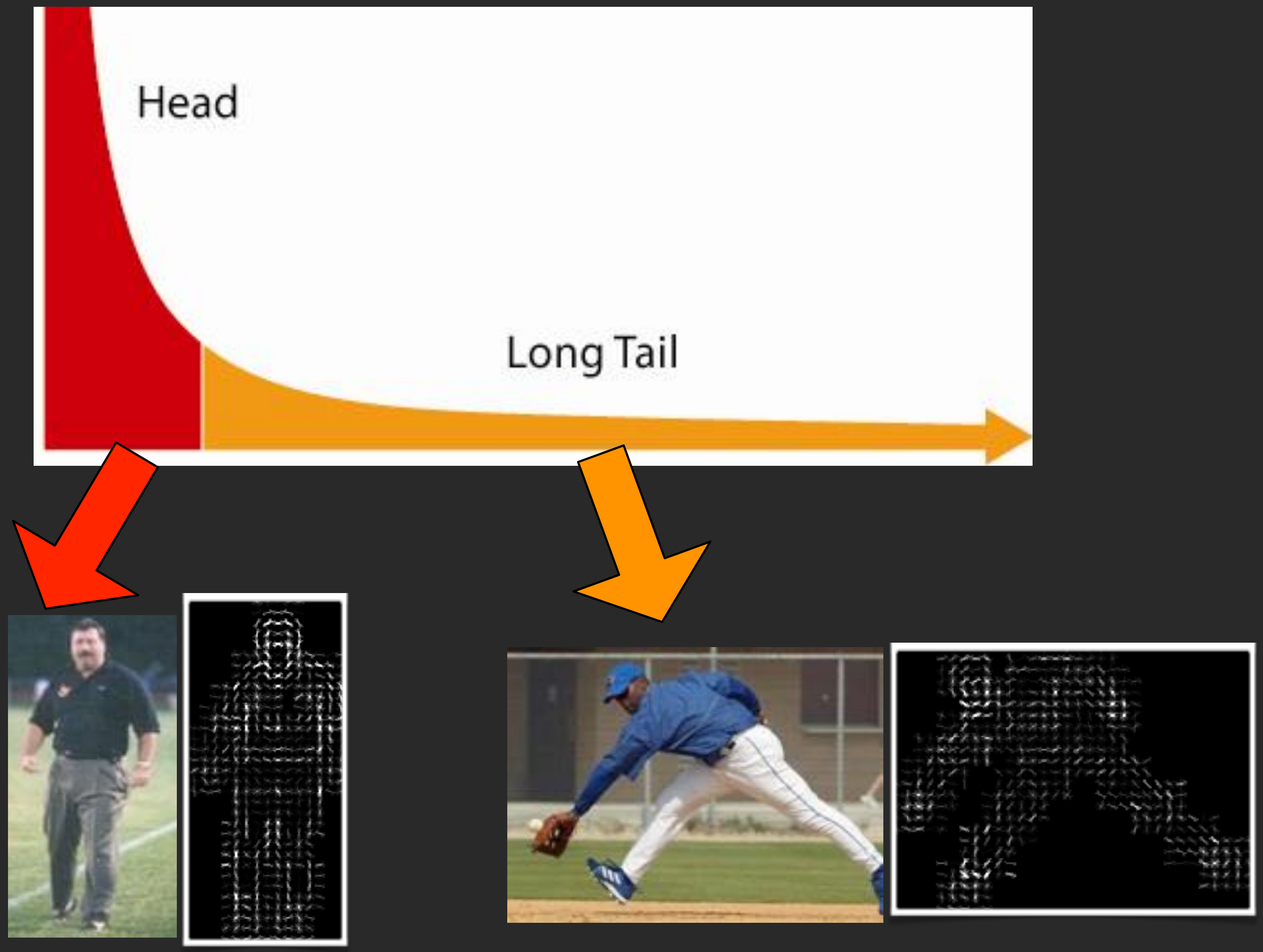


How do we define set of valid $z \in \Omega$?

One option: just use set of poses observed in training set

Sharing

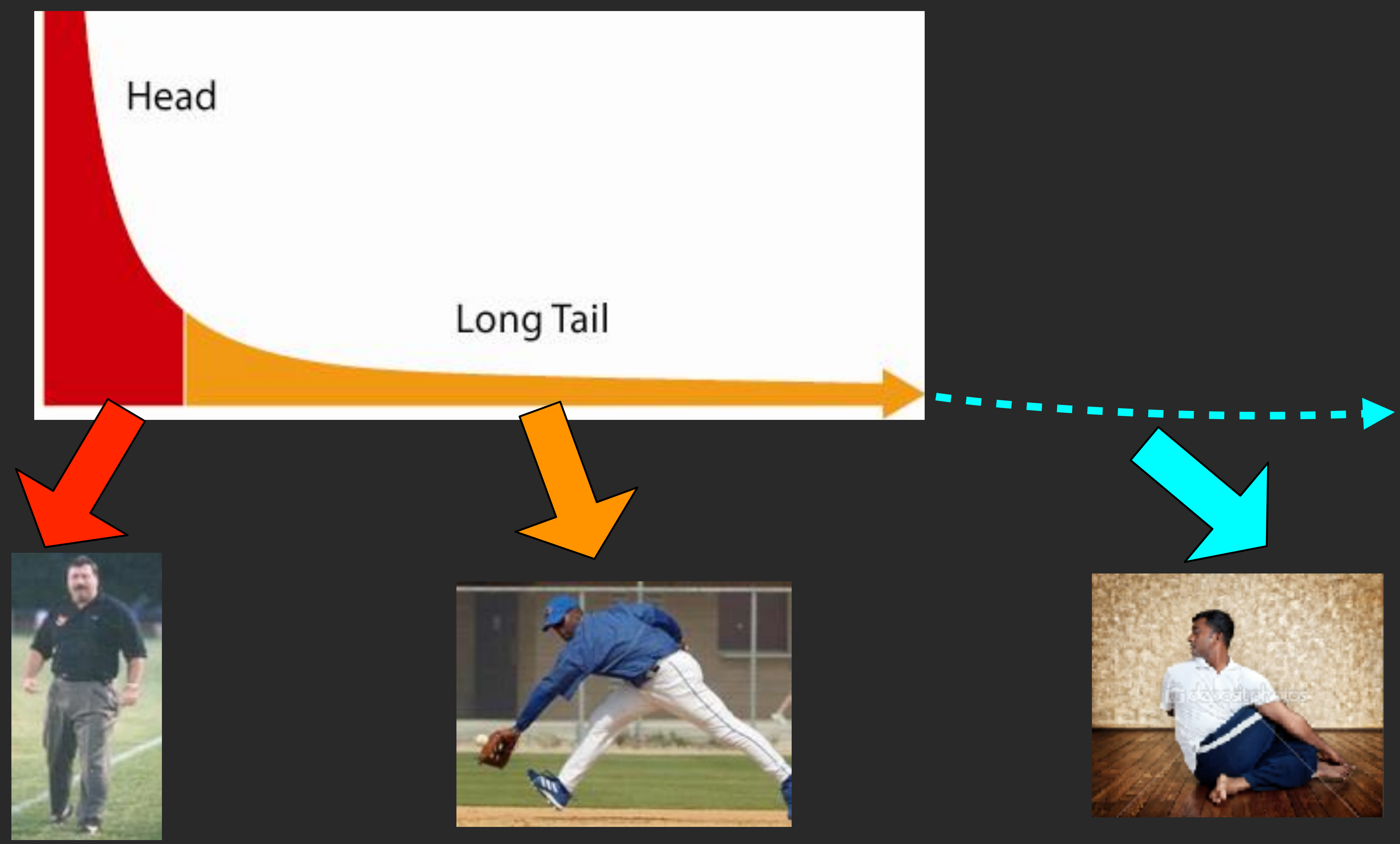
Helps address “one-shot” learning (subcategory seen at least once)



Use parts from common poses to help model rare poses

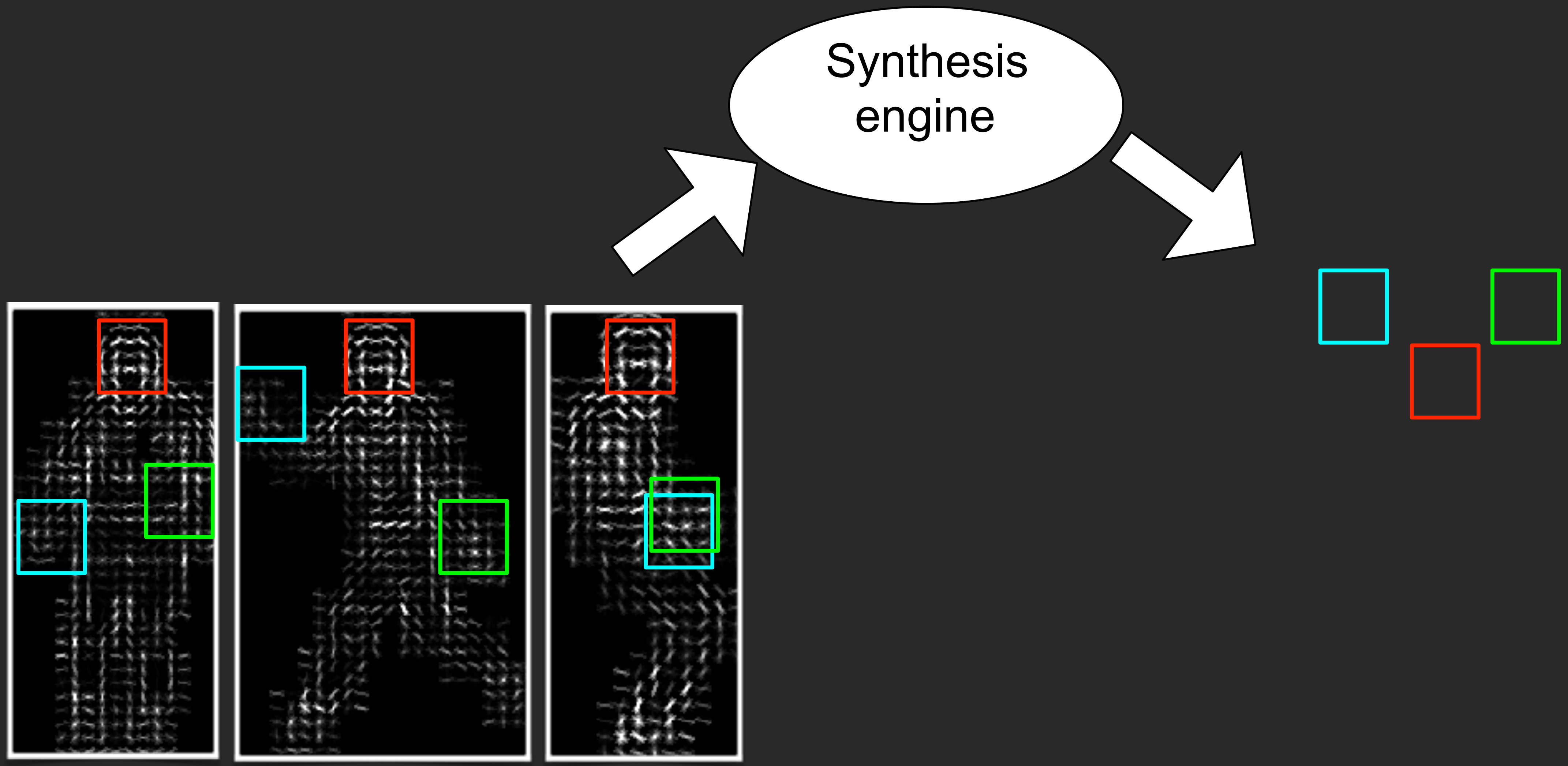
Sharing

Helps address "one-shot" learning (subcategory seen at least once)

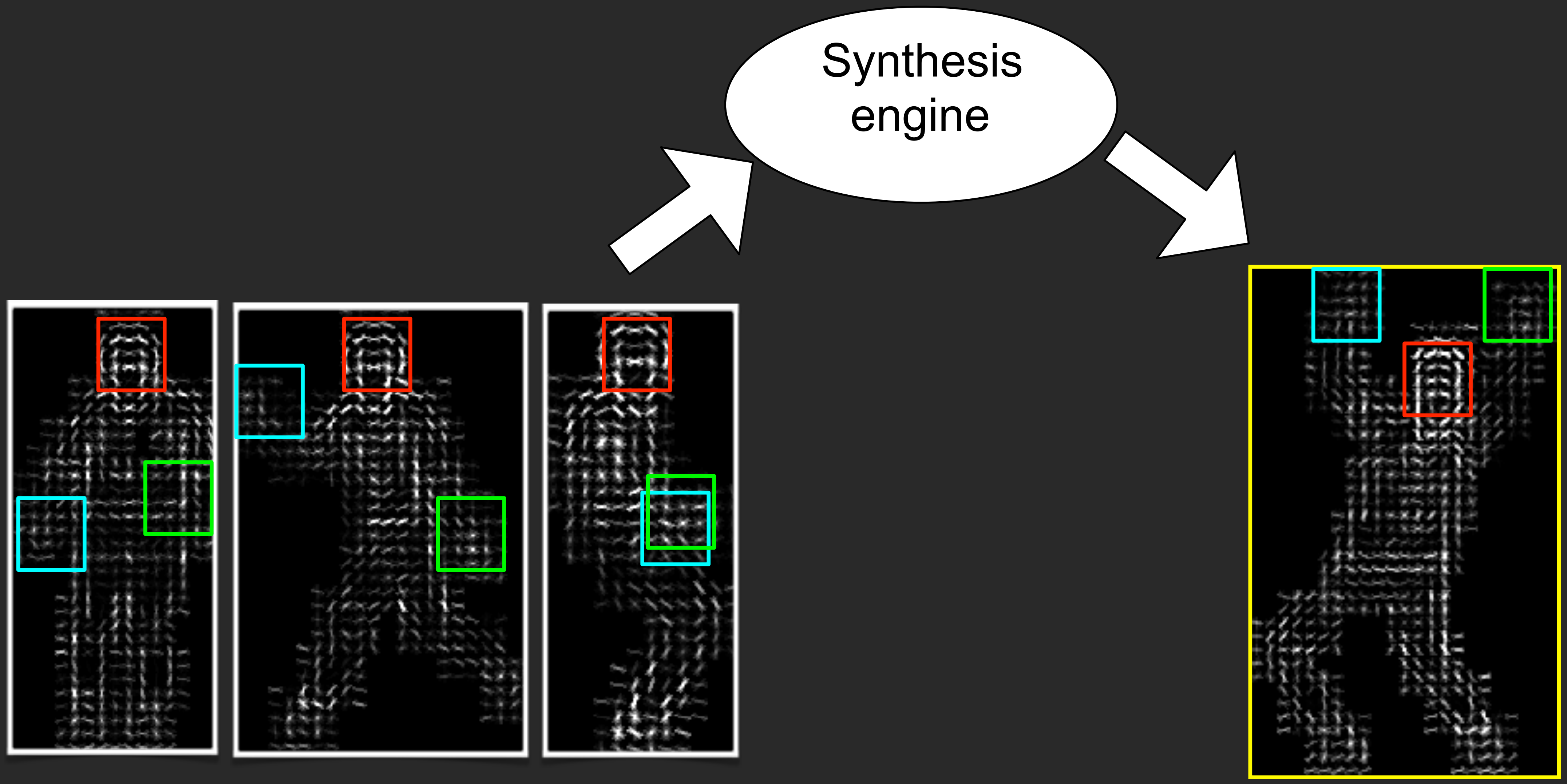


What about poses that are never seen ("zero-shot" learning)?

Shape synthesis

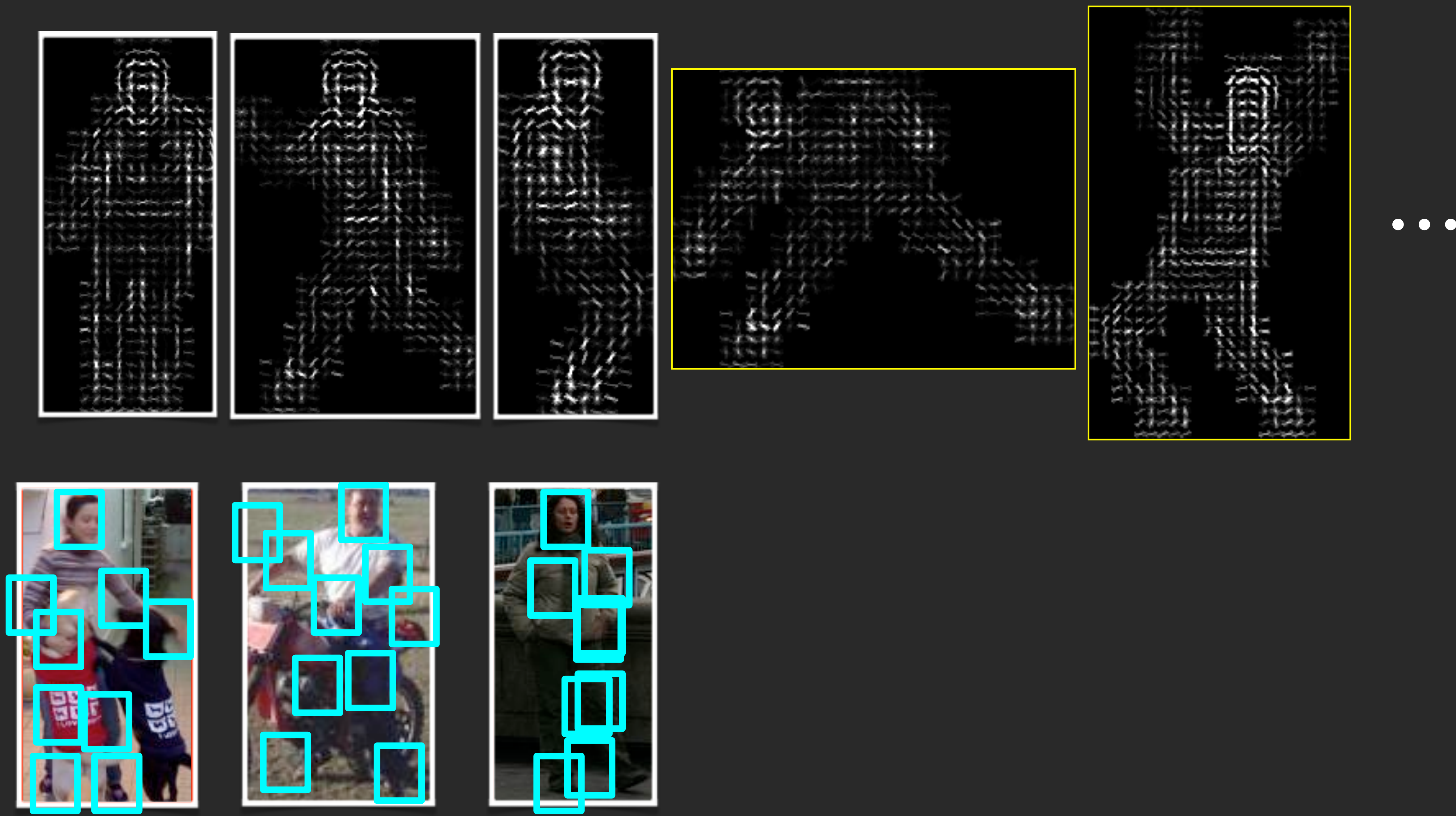


Shape synthesis



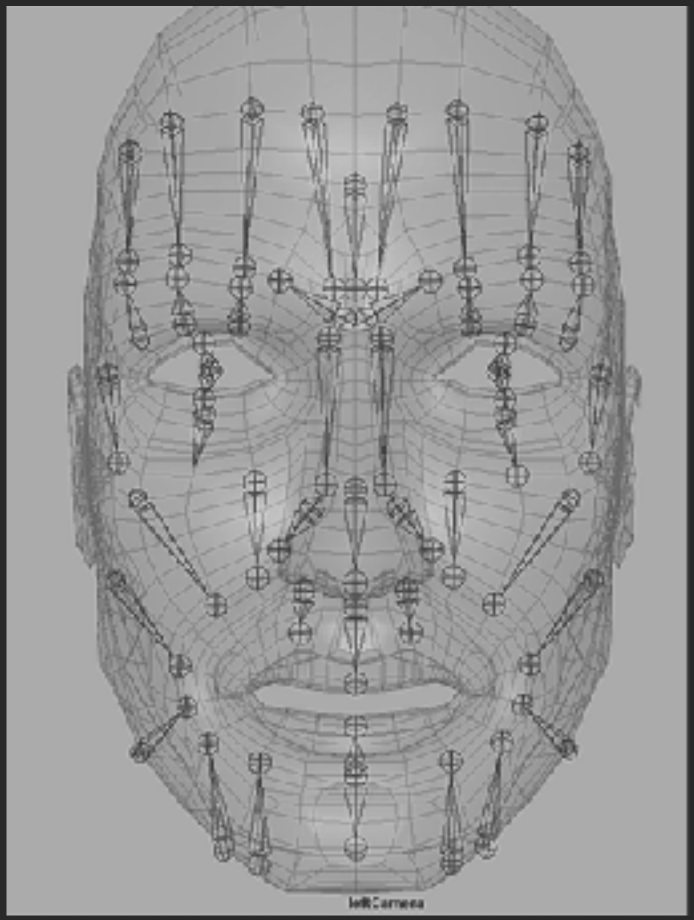
Algorithmic synthesis

Ω : set of observed + synthesized part locations

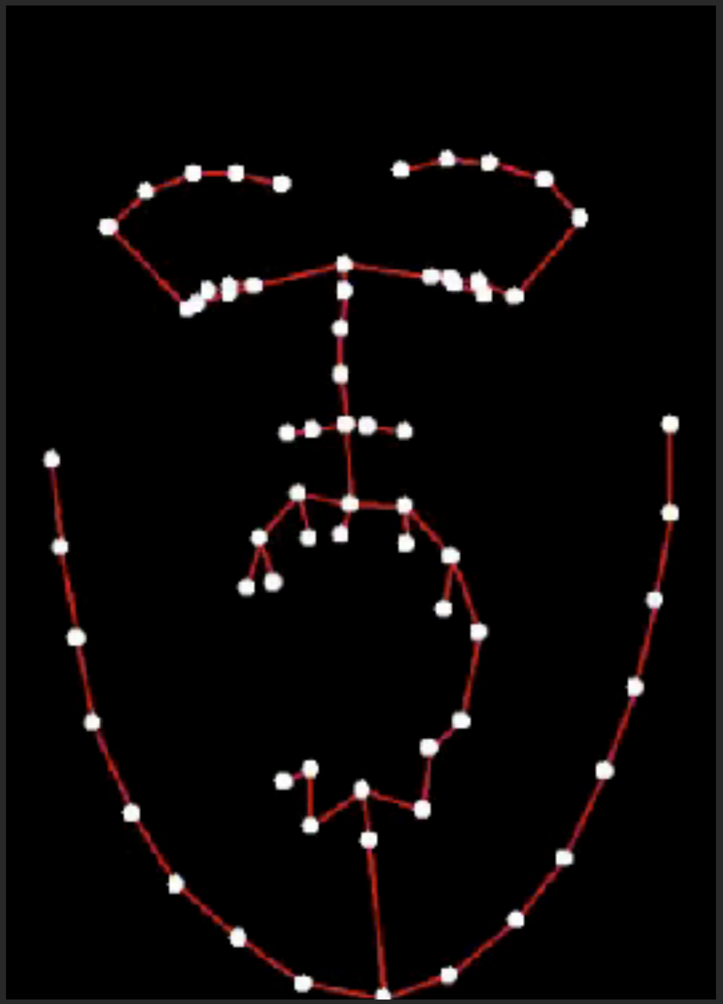


Shape synthesis

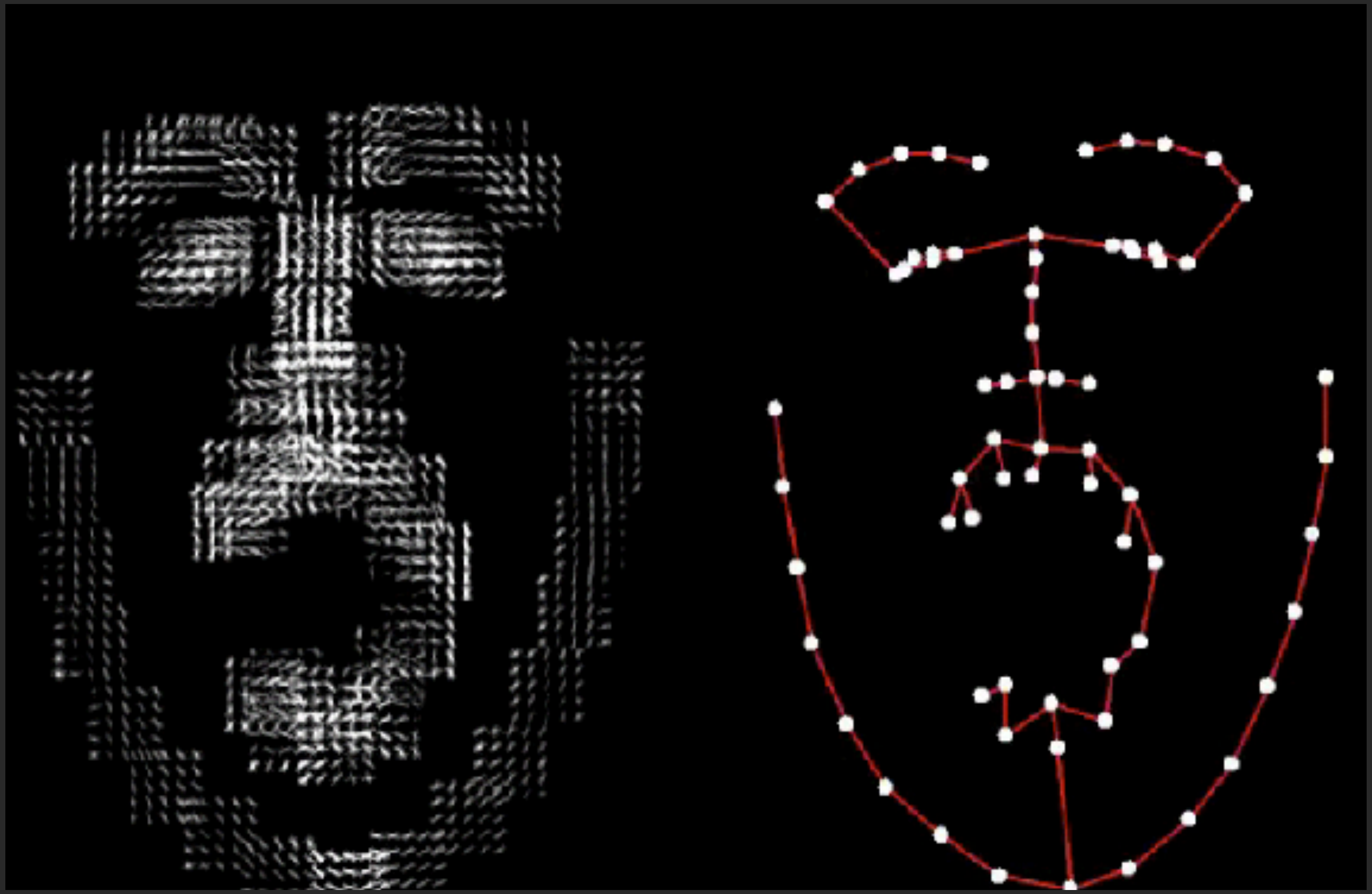
$$S(z) = (z - \mu)\Sigma^{-1}(z - \mu)$$



Graphics engine



Parametric family of classifiers

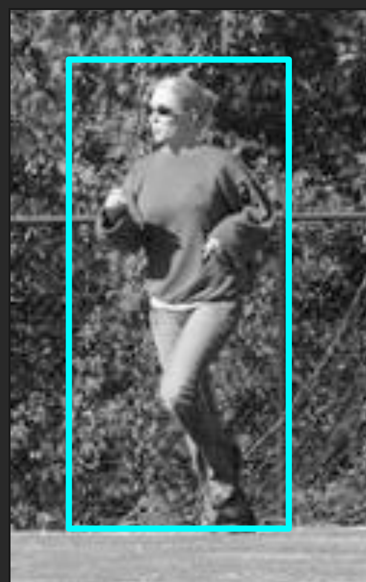


Recognition

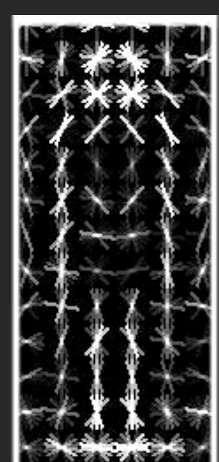
$$f(x) > 0$$

$$f(x) = w \cdot x$$

Recognition as reconstruction

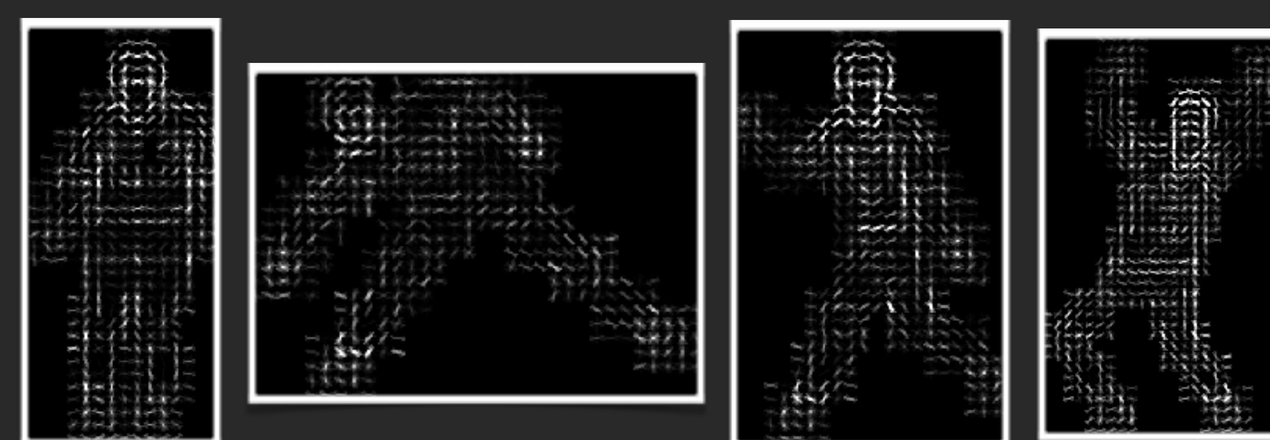


$$f(x) = w \cdot x$$



$$f(x) > 0$$

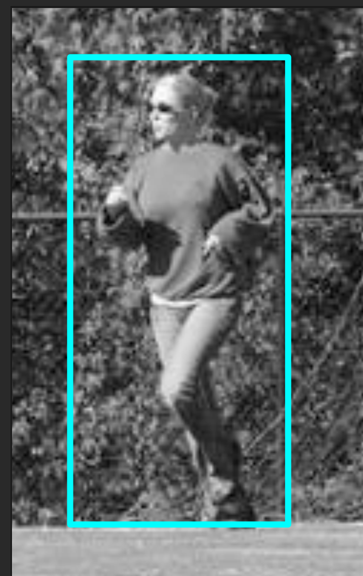
$$f(x) = \max_{z \in \Omega} w(z) \cdot x$$



Ω : set of observed + synthesized part locations

Argmax (z*) reveals pose

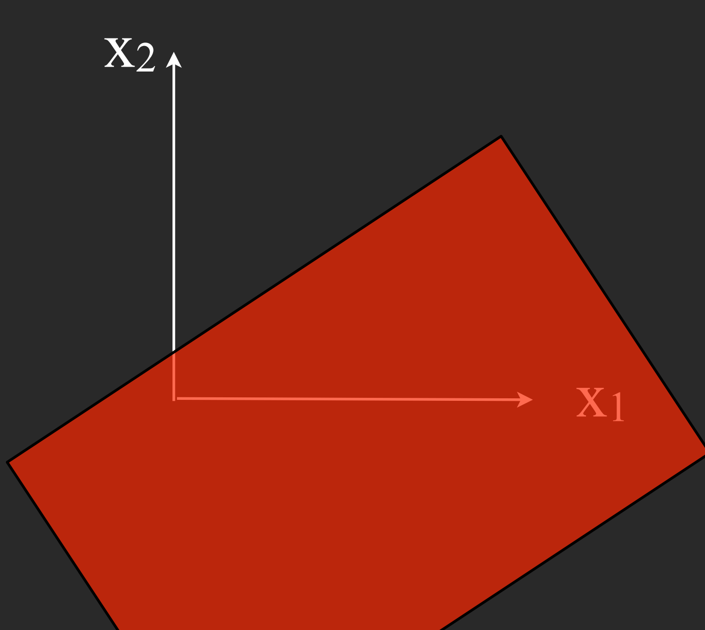
Revisit latent (vs linear) classification

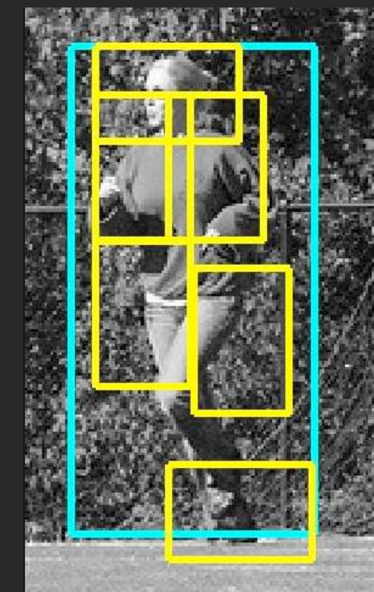


$$f(x) = w \cdot x$$

Score is linear in x

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

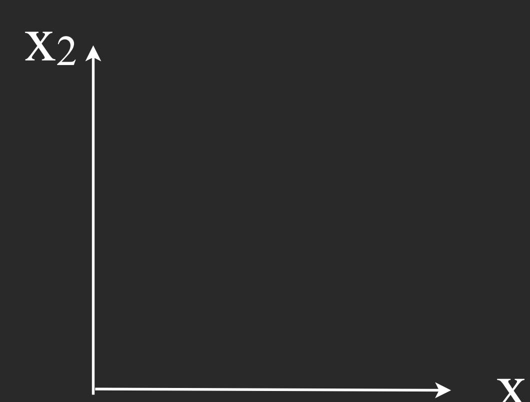




$$f(x) = \max_{z \in \Omega} w(z) \cdot x$$

Score is ?

Positive set is ?

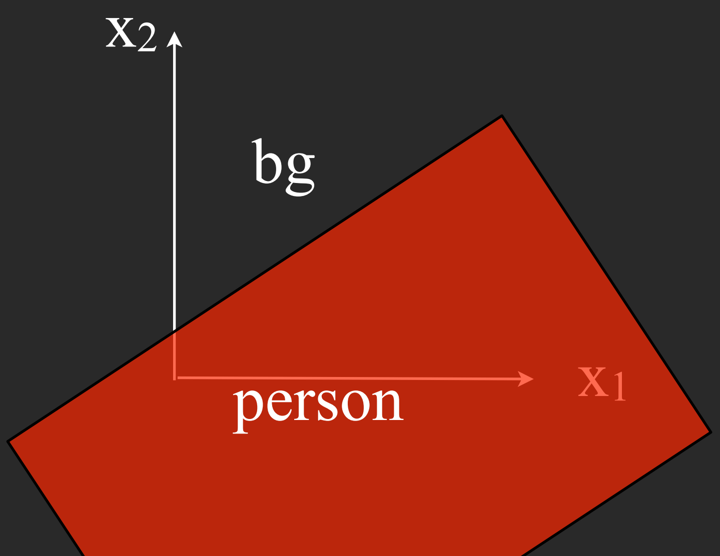


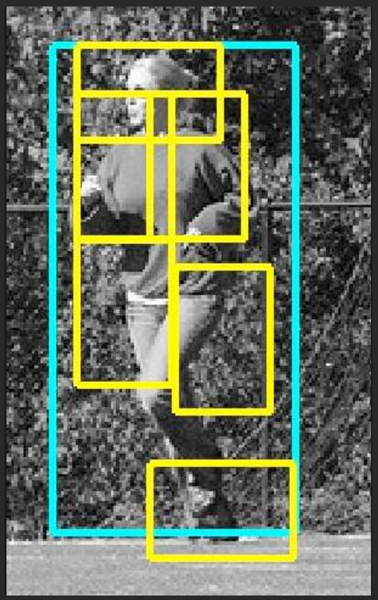
Revisit latent (vs linear) classification

$$f(x) = w \cdot x$$

Score $f_w(x)$ is linear in x

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

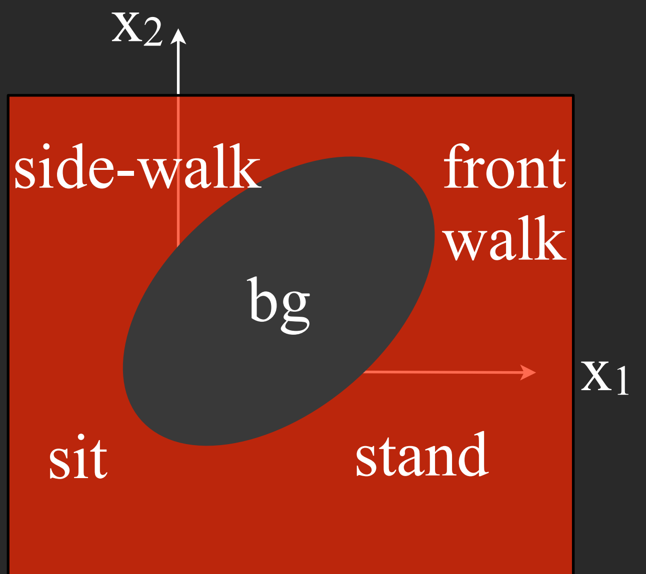




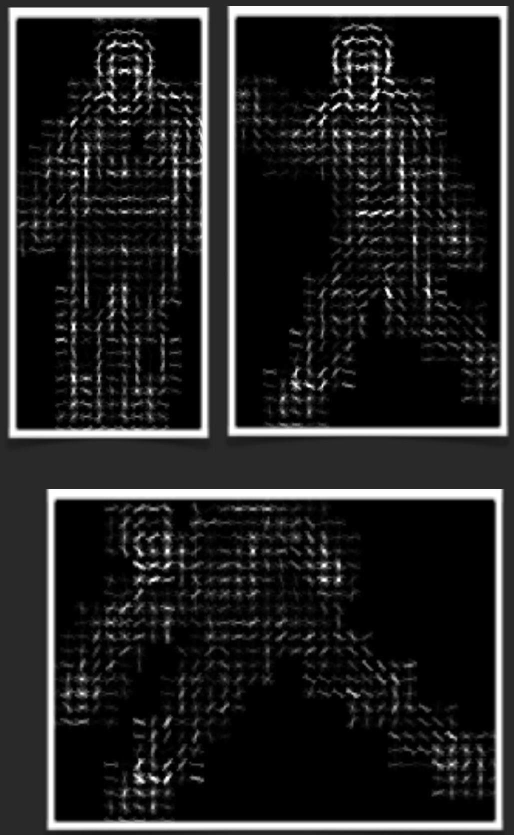
$$f(x) = \max_{z \in \Omega} w(z) \cdot x$$

Score $f_w(x)$ is convex in x

Negative set $\{x : f_w(x) <= 0\}$ is convex

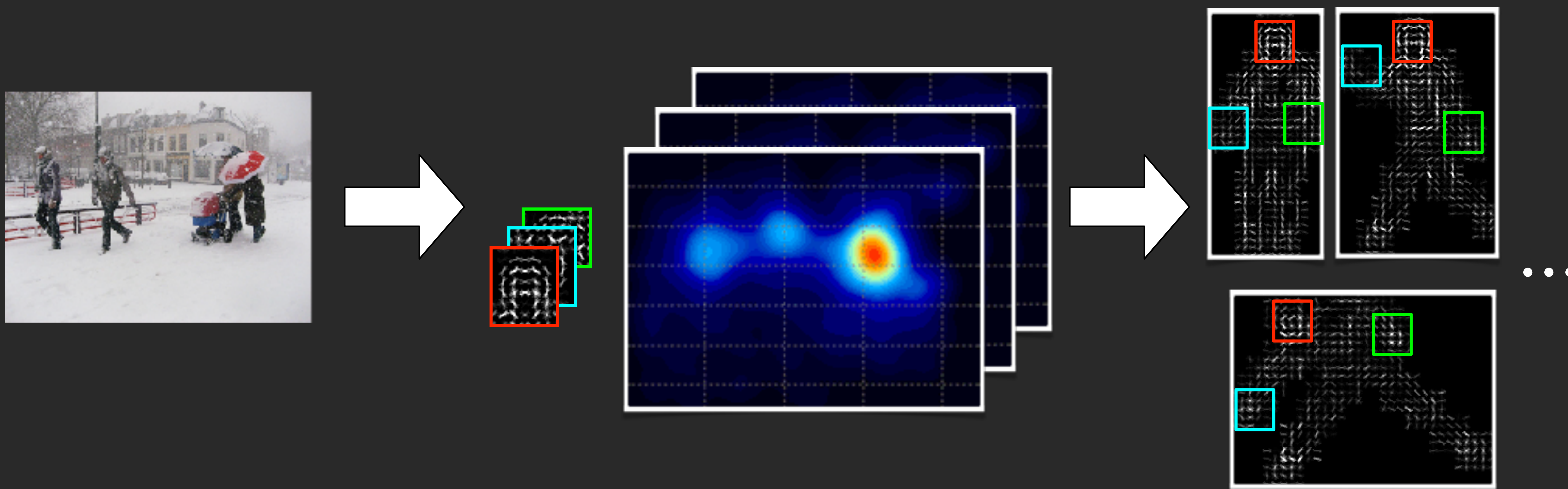


Inference



...

Inference



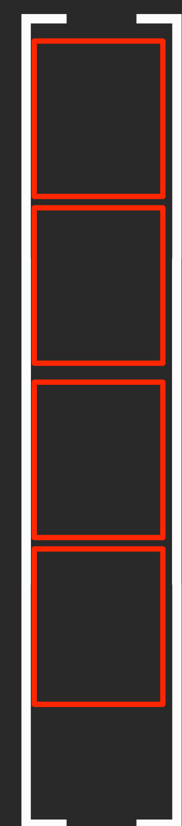
(1) Pre-compute tables of part responses

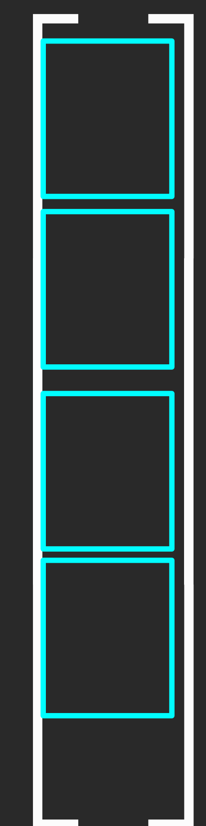
(2) Score each template with lookup table (LUT) queries

Can be implemented as a two-layer convolution

Learning

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

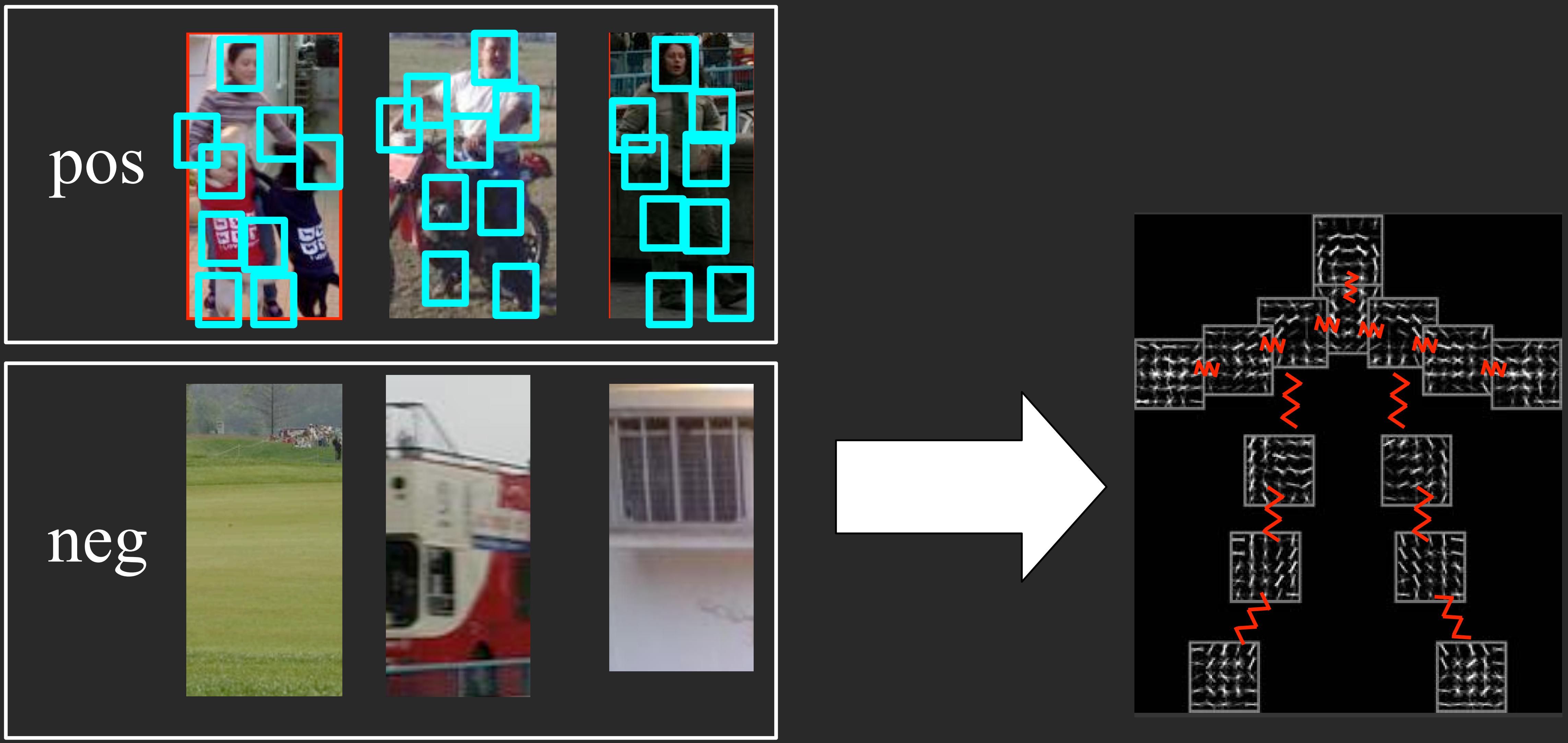
$w =$ 

$\Phi(x, z) =$ 

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

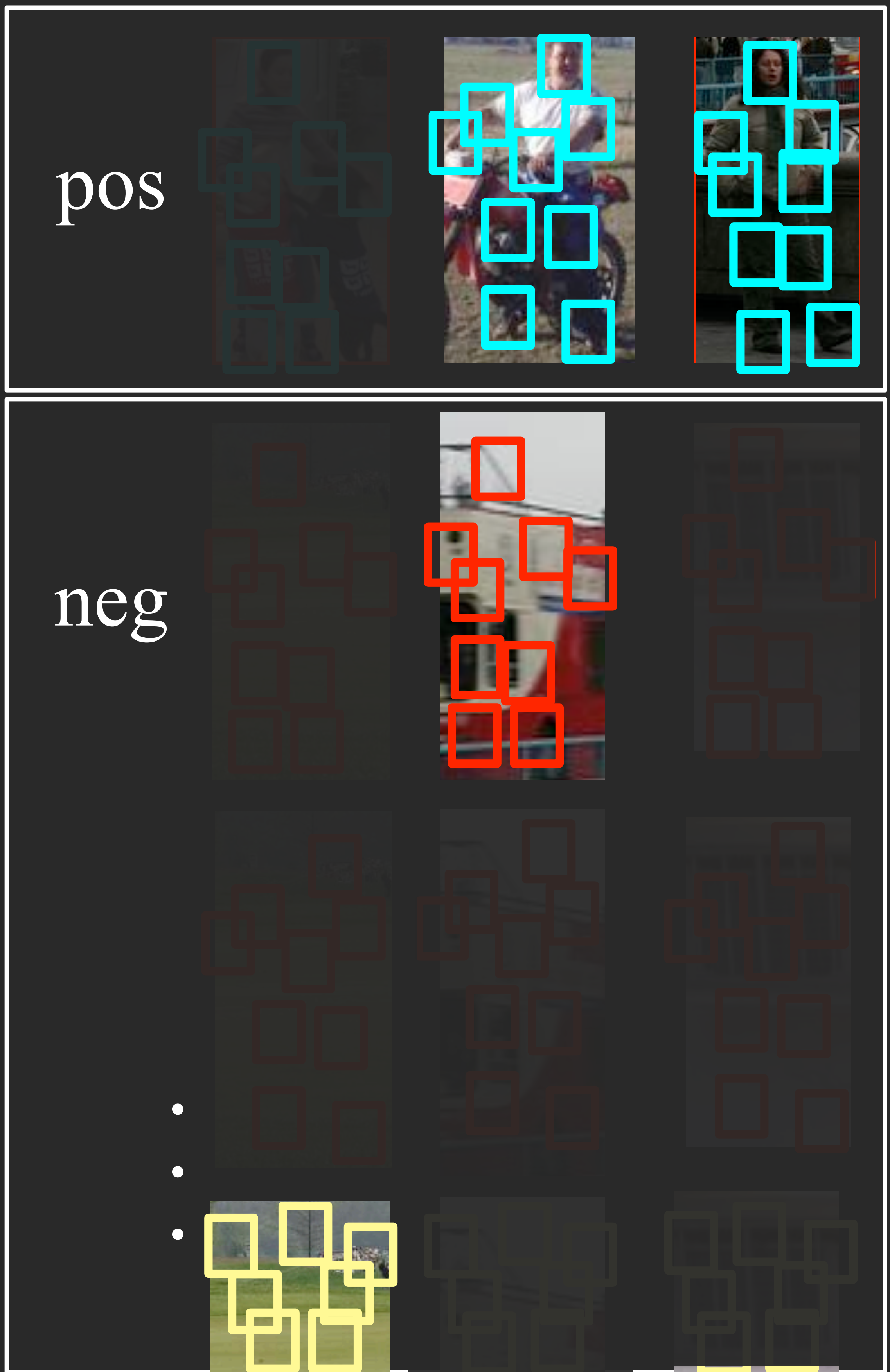
Supervised learning

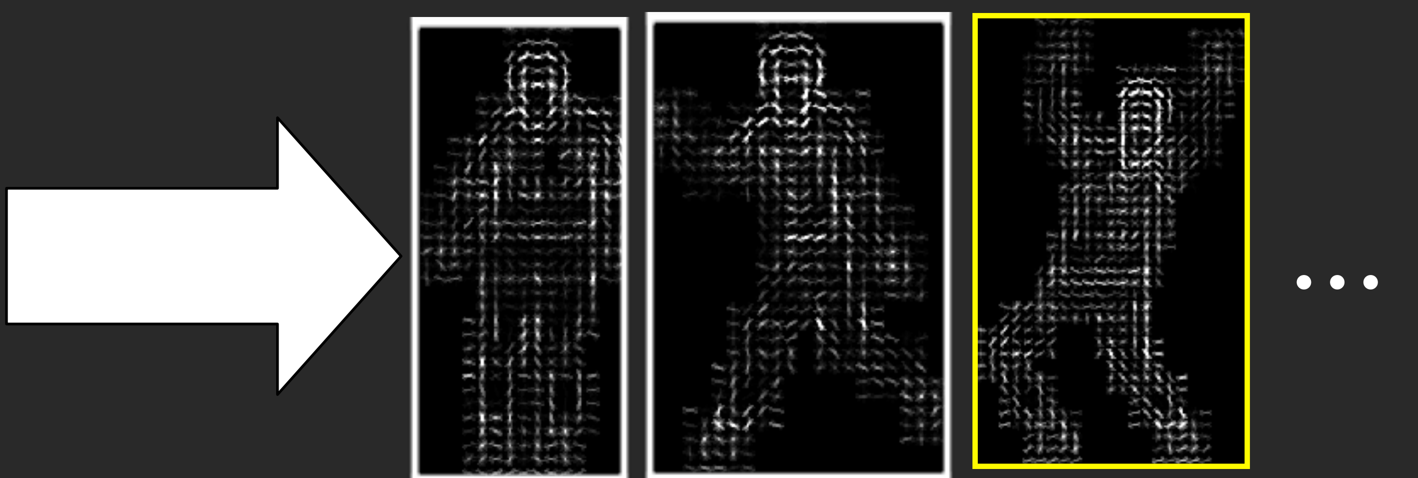
$$S(x, z) = w \cdot \Phi(x, z), \quad z \in \Omega$$



Supervised learning

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

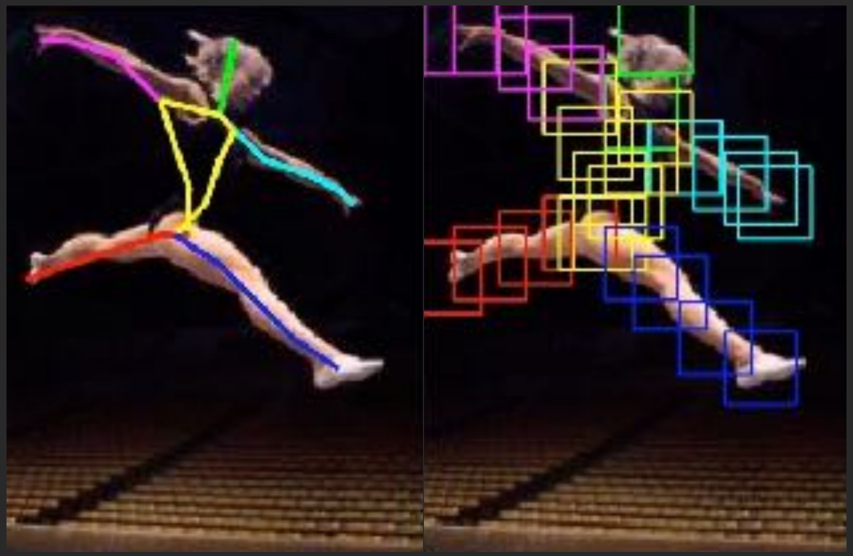


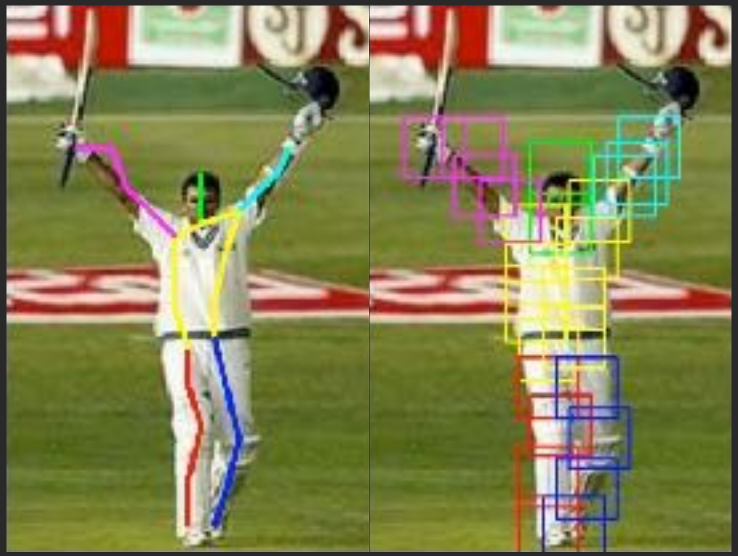


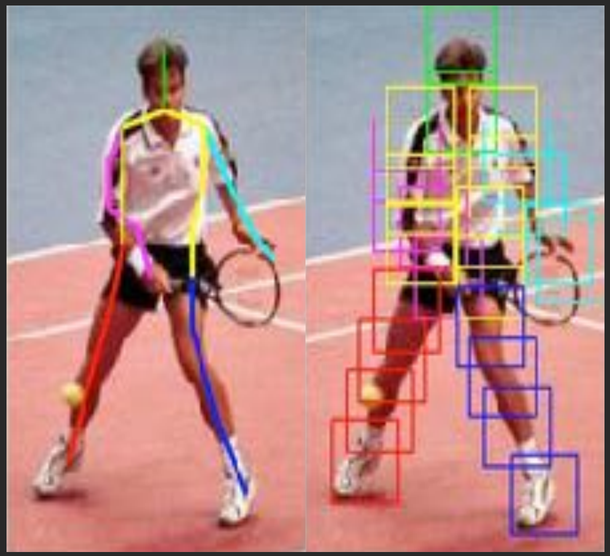
Learn classifiers for never-before-seen templates with synthesis

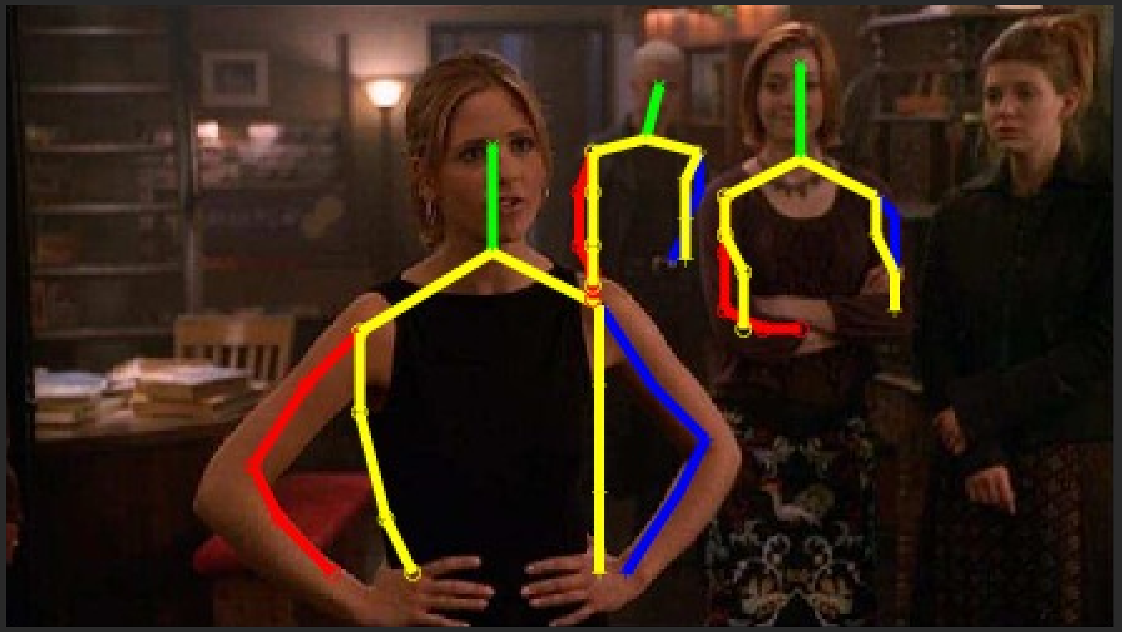
(Apply same sparse learning tricks to deal with large set of negatives!)

Joint recognition + (2D) reconstruction



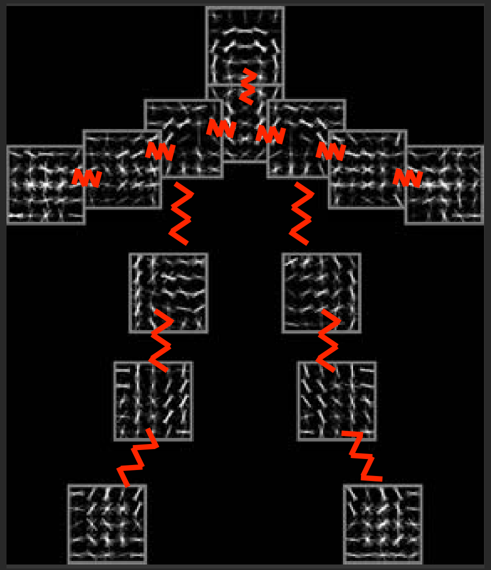






Implicit synthesis of Ω

Assume K parts and L candidate locations: $|\Omega| = L^K$

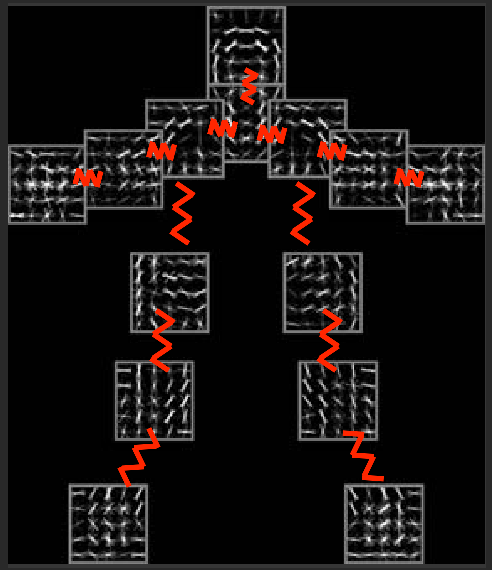


$$\max_{z \in \Omega} \sum_i w_i \cdot \phi(x, z_i)$$

Do we really need to search over an exponential number of templates to compute maximum?

Implicit synthesis of Ω

Assume K parts and L candidate locations: $|\Omega| = L^K$



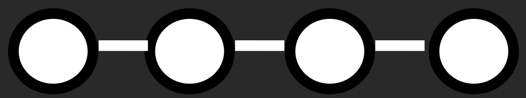
$$\max_{z \in \Omega} \sum_i w_i \cdot \phi(x, z_i)$$

Do we really need to search over an exponential number of templates to compute maximum?

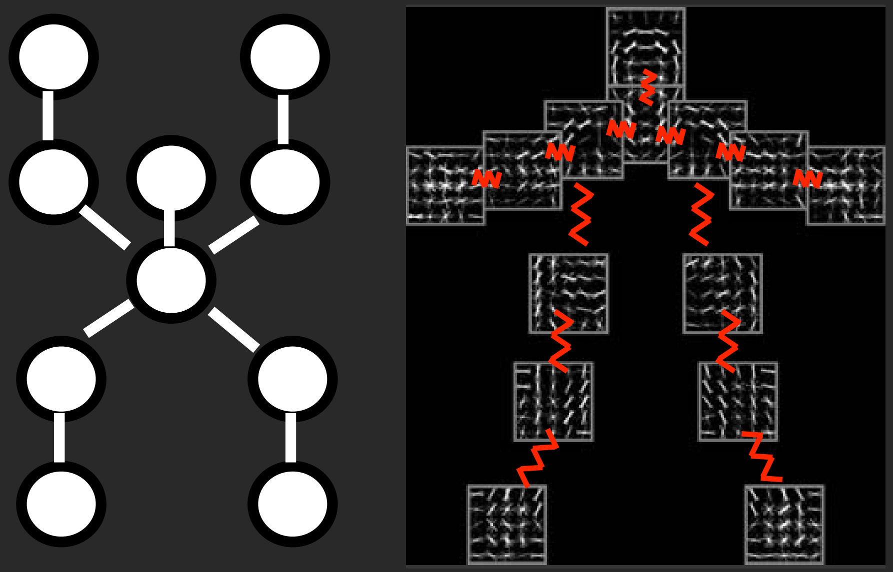
$$= \sum_i \max_{z_i} w_i \cdot \phi(x, z_i)$$

No! Independantly find best location of each part. Allows us to *implicitly* synthesize Ω

Generalize approach to markov models



temporal markov model

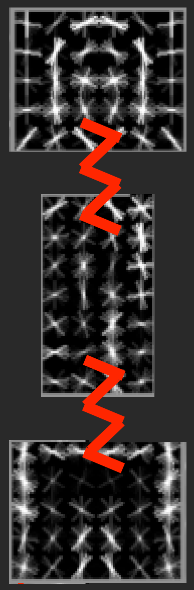


spatial markov model

- For each candidate torso, independently estimate best arm and leg
- Allows us to model (and learn) priors over exponentially-large shape space

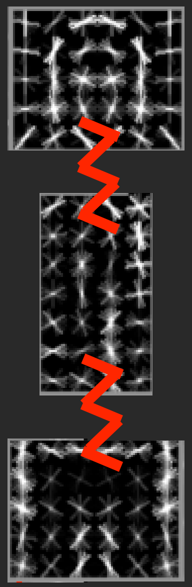
$$S(x,z) = \sum_{i \in V} w_i \cdot \phi(x, z_i) + \sum_{ij \in E} w_{ij} \cdot \psi(z_i, z_j)$$

General case



$$\max_{z_1,z_2,z_3} \left[\phi(z_1) + \phi(z_2) + \phi(z_3) + \psi(z_1, z_2) + \psi(z_2, z_3)\right]$$

General case



$$\max_{z_1,z_2,z_3} \left[\phi(z_1) + \phi(z_2) + \phi(z_3) + \psi(z_1, z_2) + \psi(z_2, z_3)\right]$$

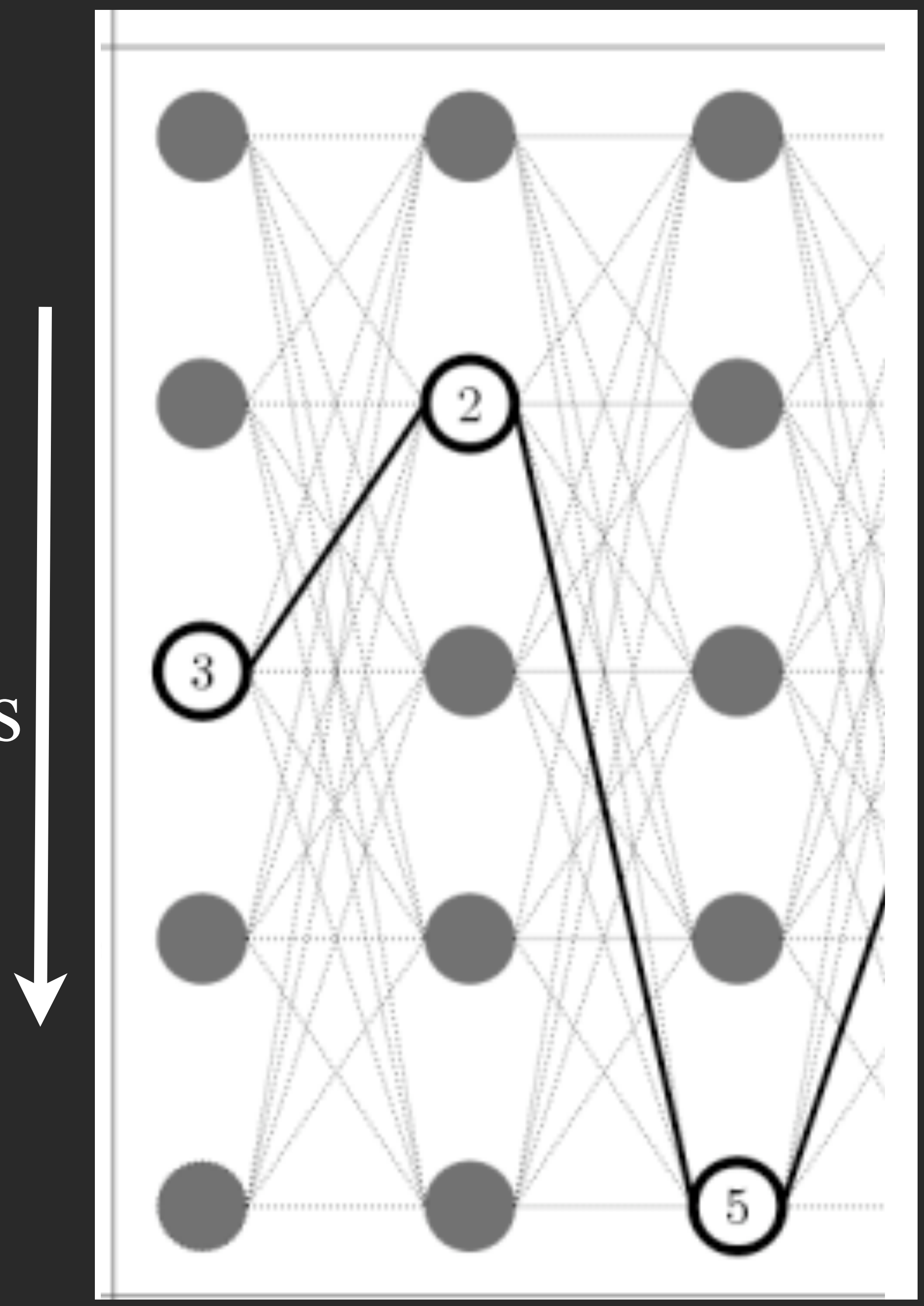
$$= \max_{z_2,z_3} \left(\psi(z_2) + \psi(z_2, z_3) + \max_{z_1} \left[\phi(z_1) + \psi(z_1, z_2)\right]\right)$$

$$= \max_{z_2,z_3} \left(\psi(z_2) + \psi(z_2, z_3) + m(z_2)\right)$$

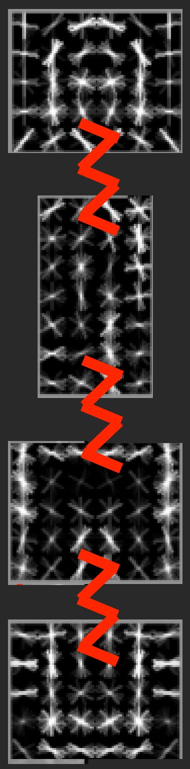
Use simple variatble elimination to reduce inference to $O(KL^2)$

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

Pixel locations



head torso leg



1) Initialize nodes with match score

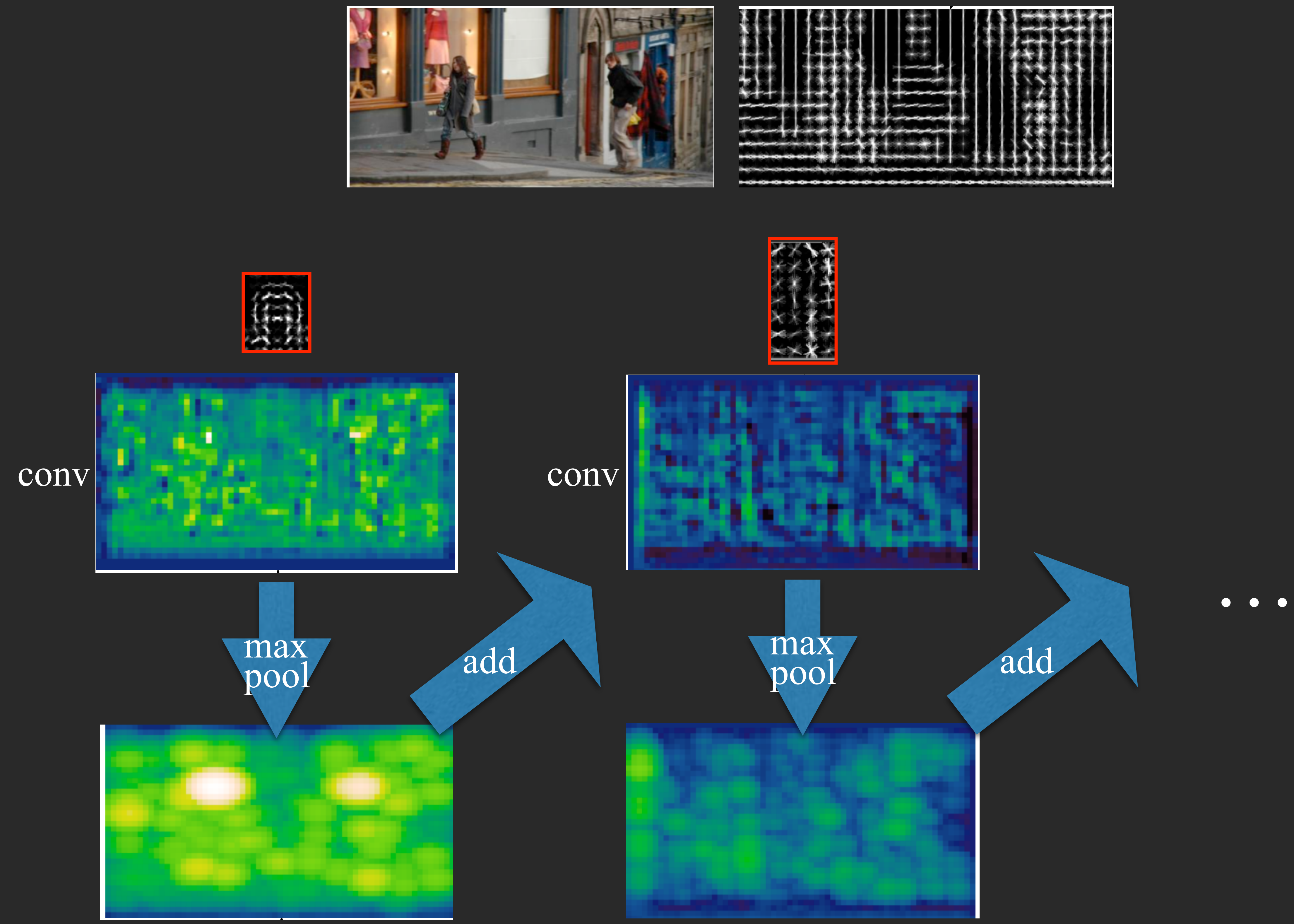
2) Initialize edges with spring score

3) Find best path from left to right

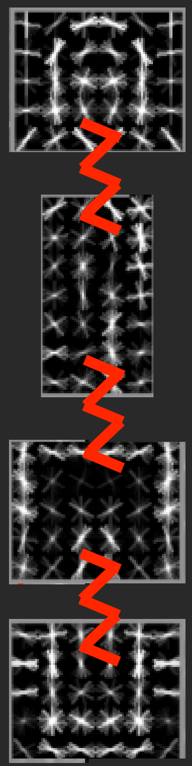
In practice, (1) is bottleneck

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

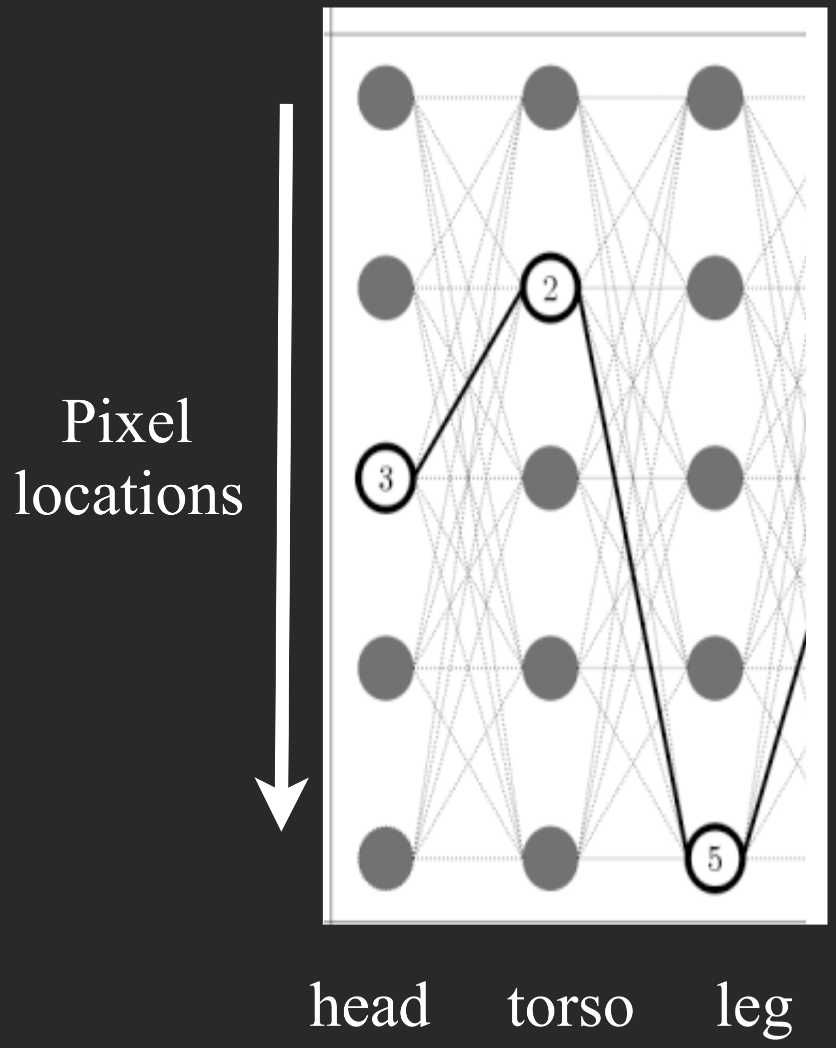
implement with convolutions + max pooling



General formulation: inference



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

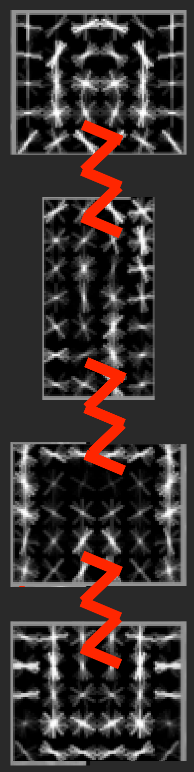


Local and pairwise potentials can be arbitrary nonlinear functions of image and position

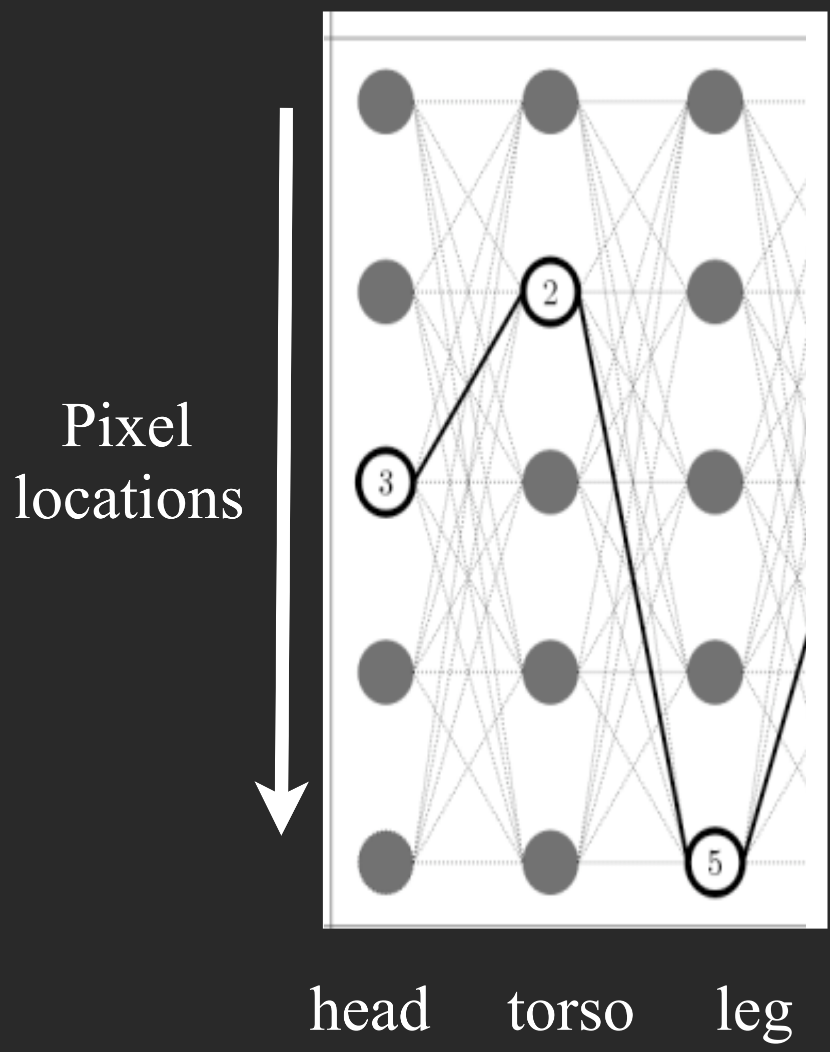
(e.g., neural net part model)

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

General formulation: inference



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



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

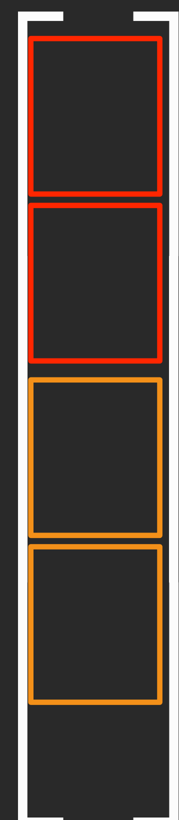
(e.g., neural net part model)

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

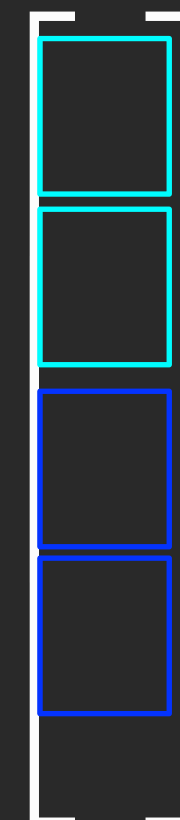
General formulation: learning

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

$w =$



$\Phi(x, z) =$



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

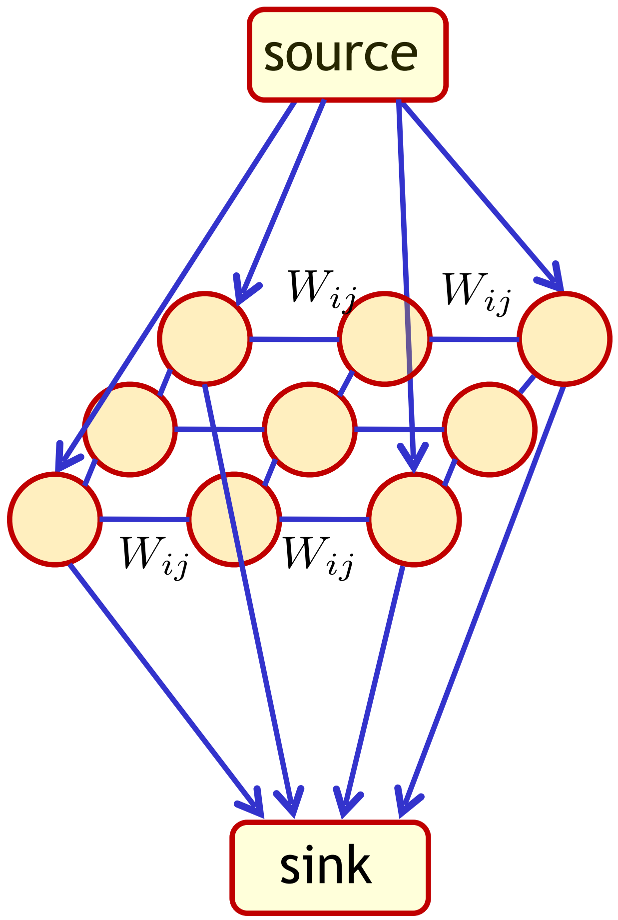
Jointly learn appearance and geometry with linear classification!

Aside: structural SVM learning

$$\{(x_n, y_n)\}$$

$$x_n \in R^{H \times W}$$

$$y_n \in \{0, 1, \ldots K\}^{H \times W}$$



$$E(x, y) = w \cdot \Phi(x, y)$$

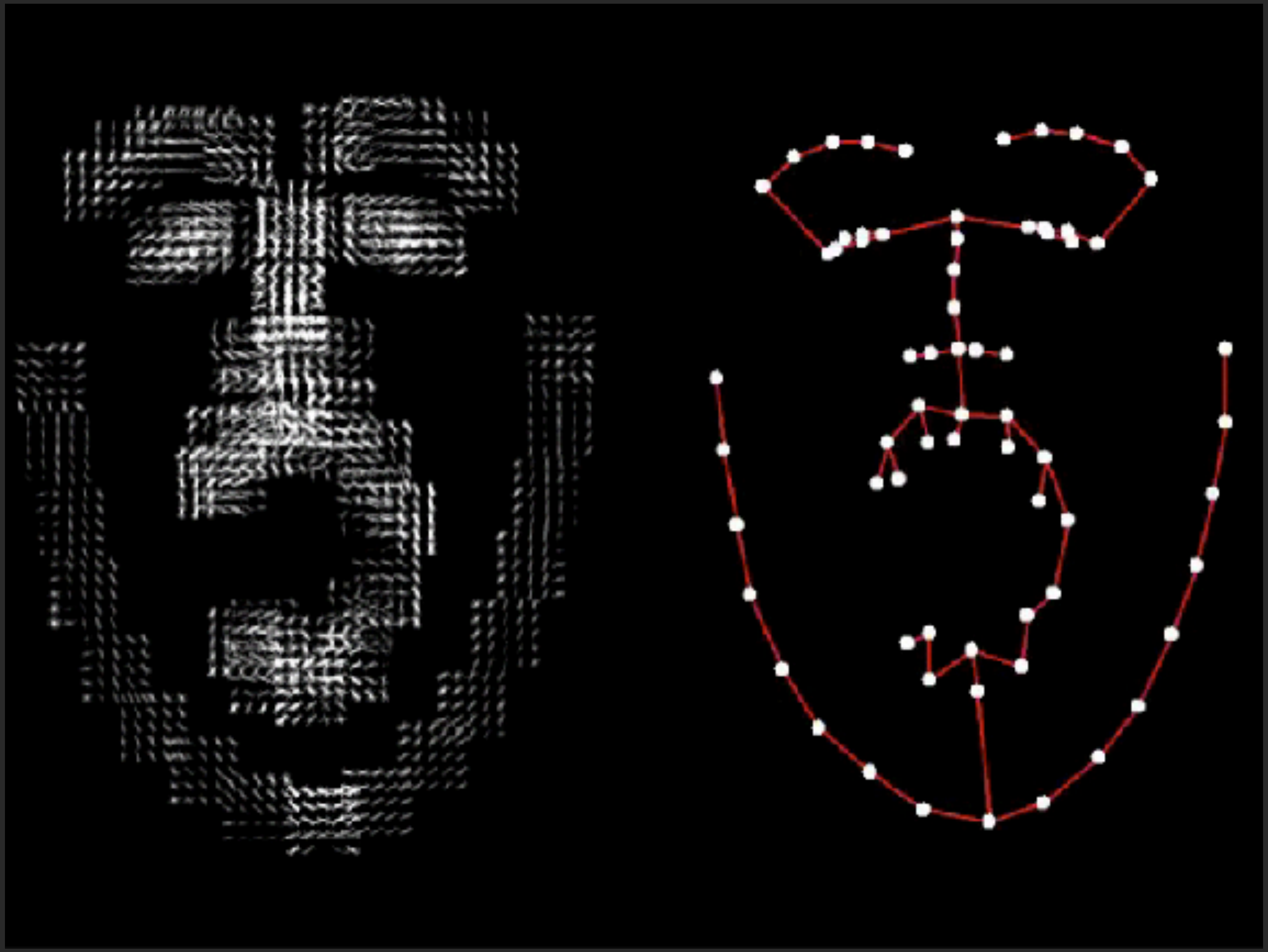
$$E(x, y) = \sum_{i \in V} w_{local} \cdot \phi(x, y_i) + \sum_{ij \in \mathcal{E}} w_{pair} \cdot \psi(y_i, y_j, x)$$

Appears in European Conference on Computer Vision (ECCV) 2008

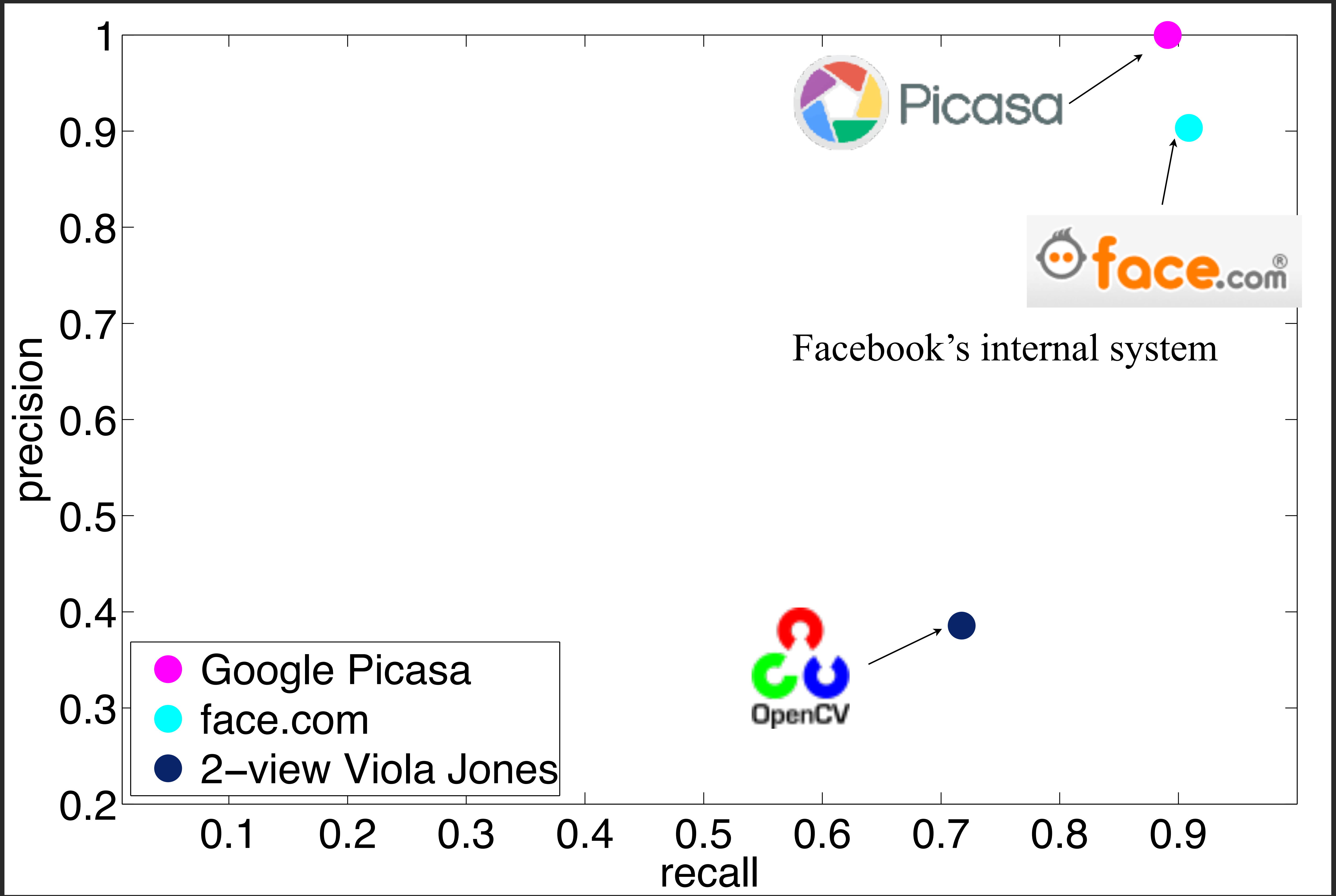
Learning CRFs using Graph Cuts

Martin Szummer[1], Pushmeet Kohli[1], and Derek Hoiem[2]

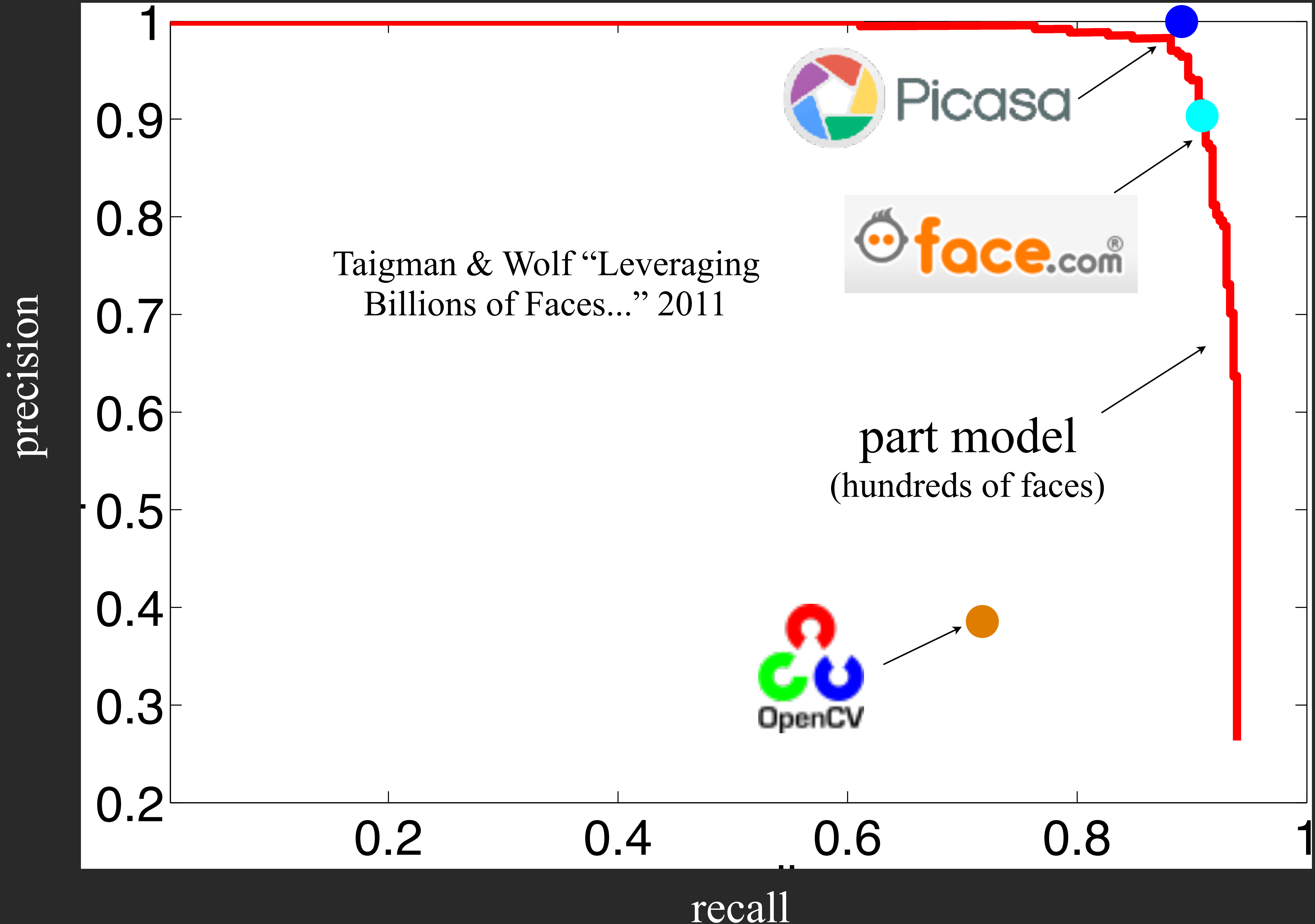
Modeling faces



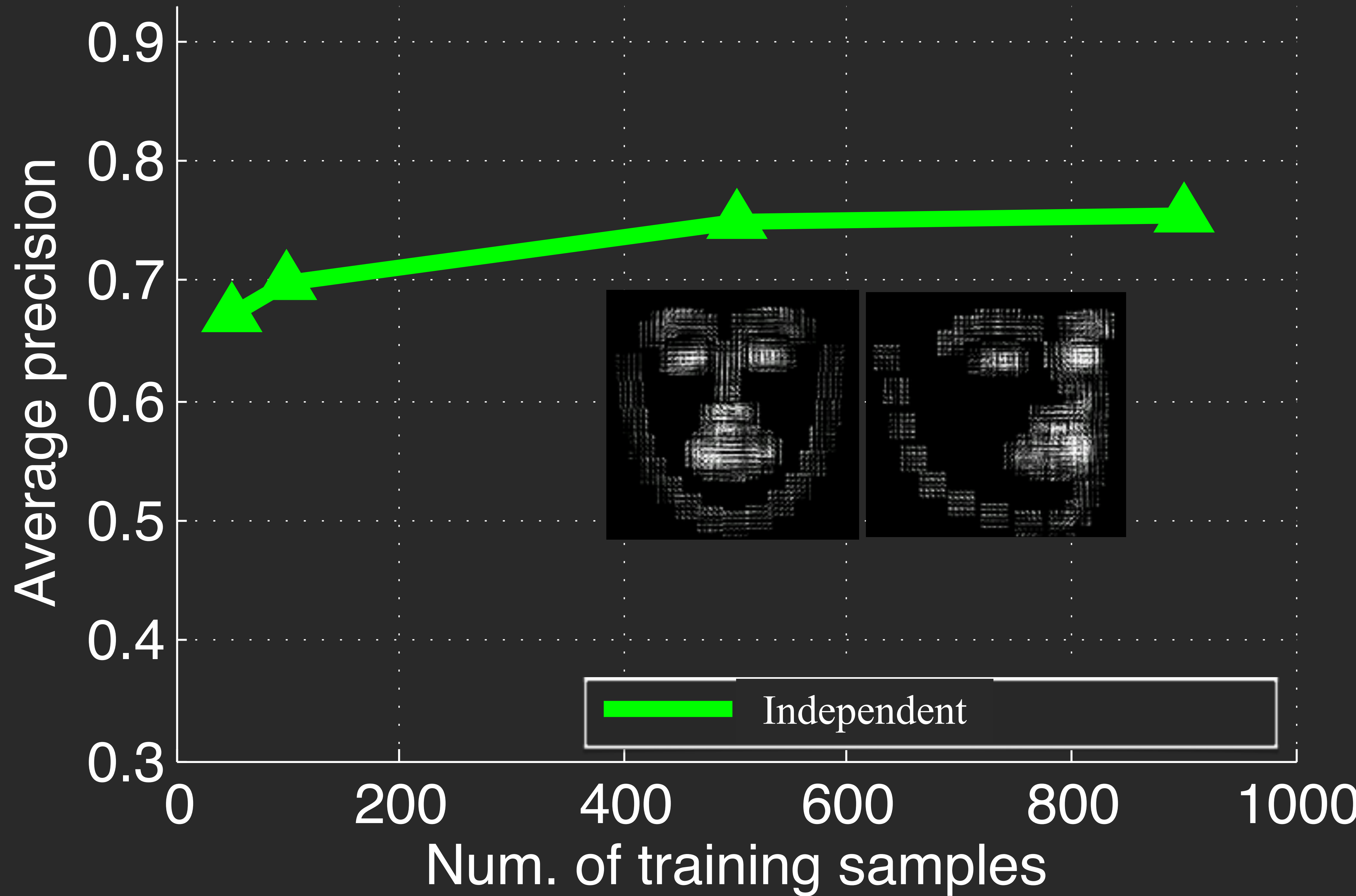
Face detection



Face detection

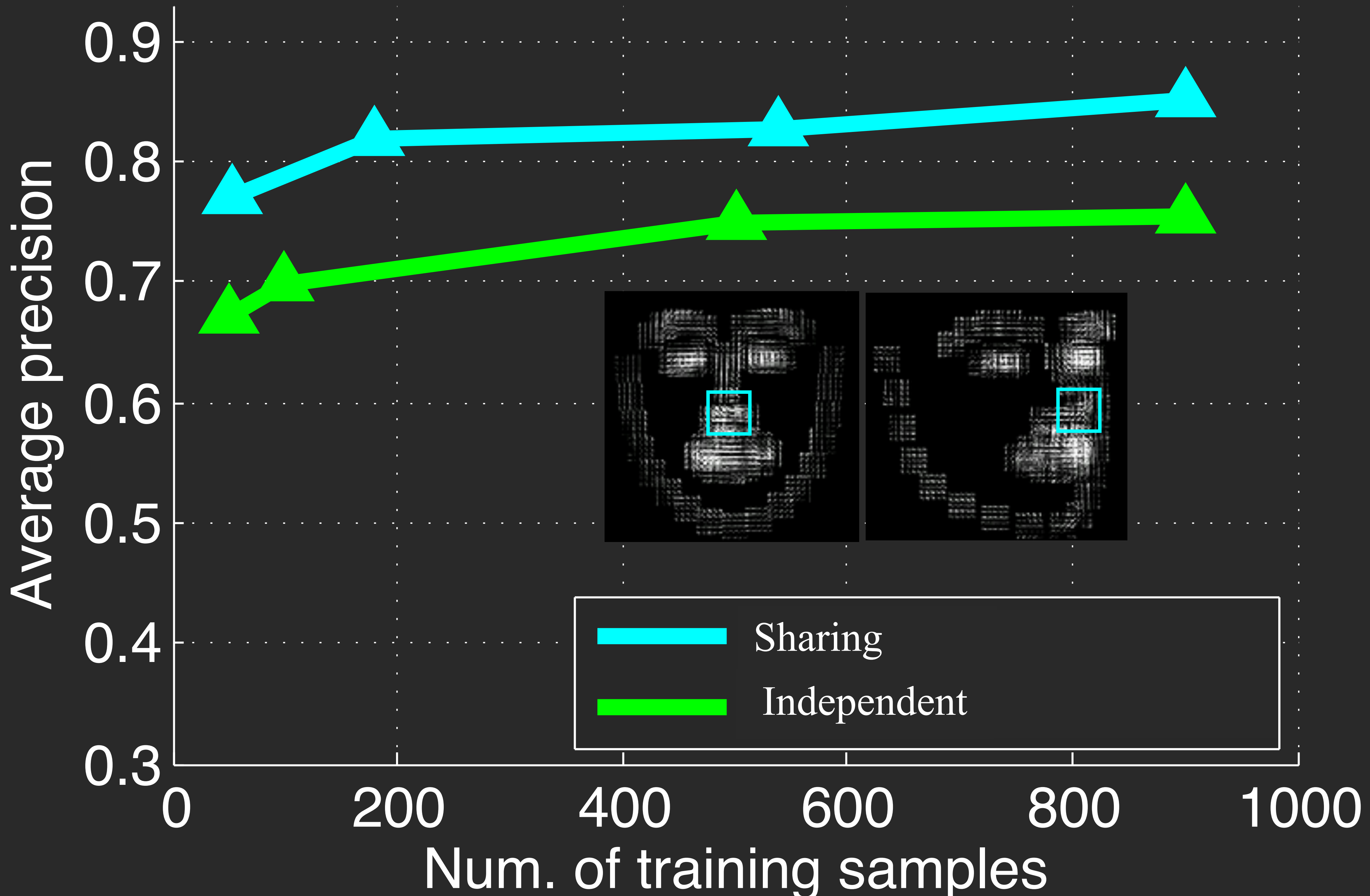


Could we do better with more data?



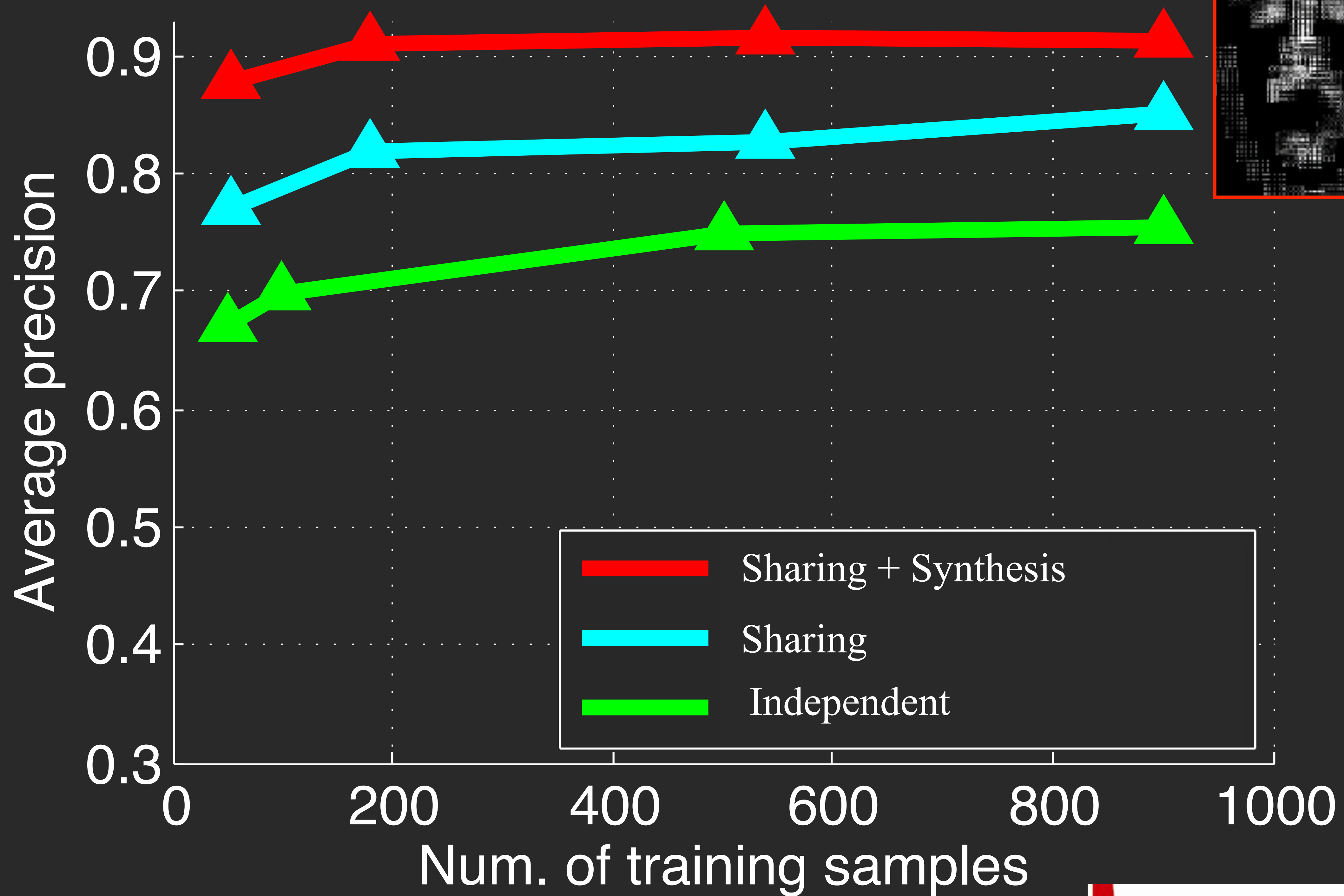
Zhu, Vondrick, Ramanan & Fowlkes,
"Do we need more training data or better models?"
IJCV 2015 (accepted)

Indepedant subcategories vs sharing



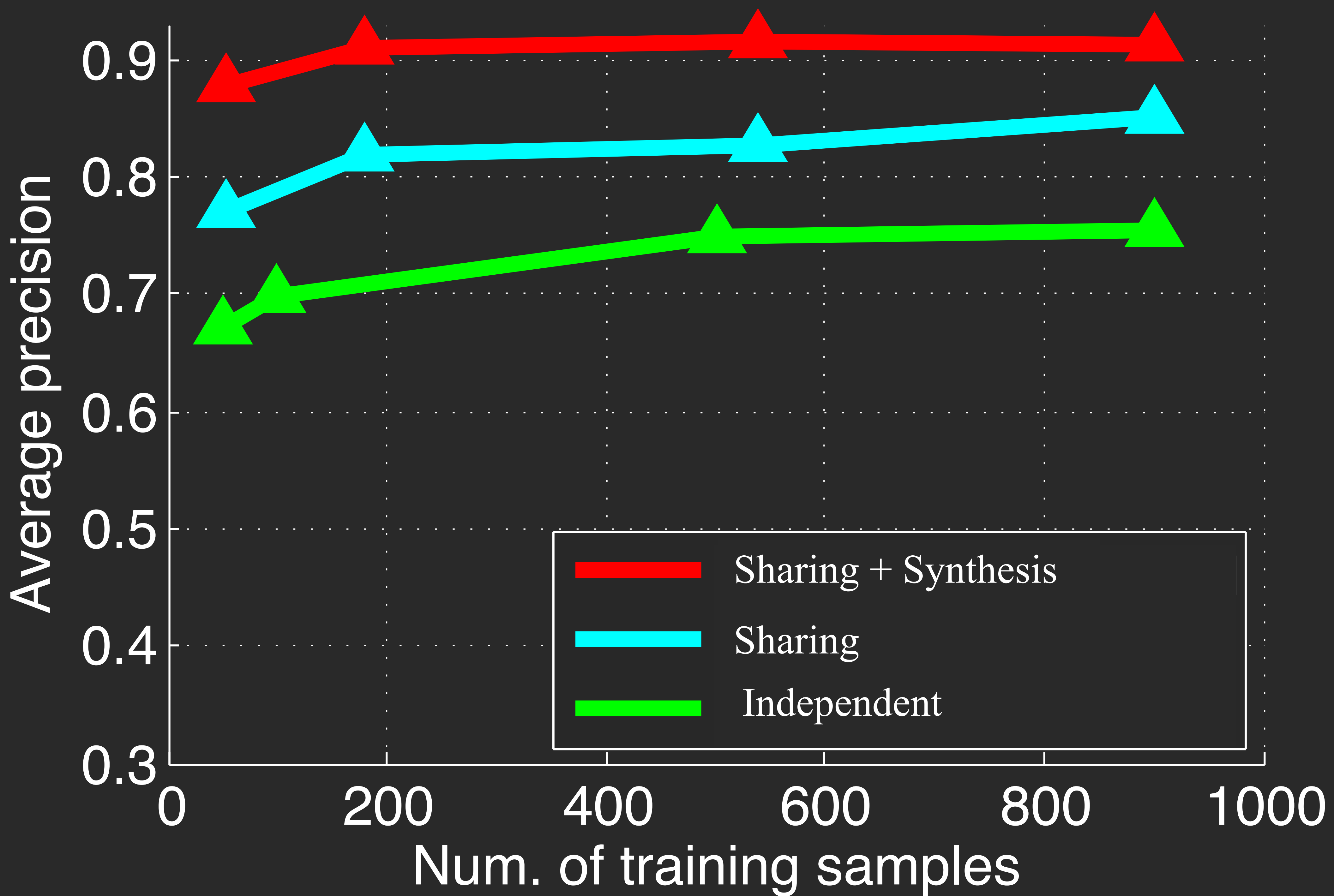
Zhu, Vondrick, Ramanan & Fowlkes,
"Do we need more training data or better models?"
IJCV 2015 (accepted)

Sharing versus synthesis



Synthesis even more beneficial than sharing

Sharing versus synthesis



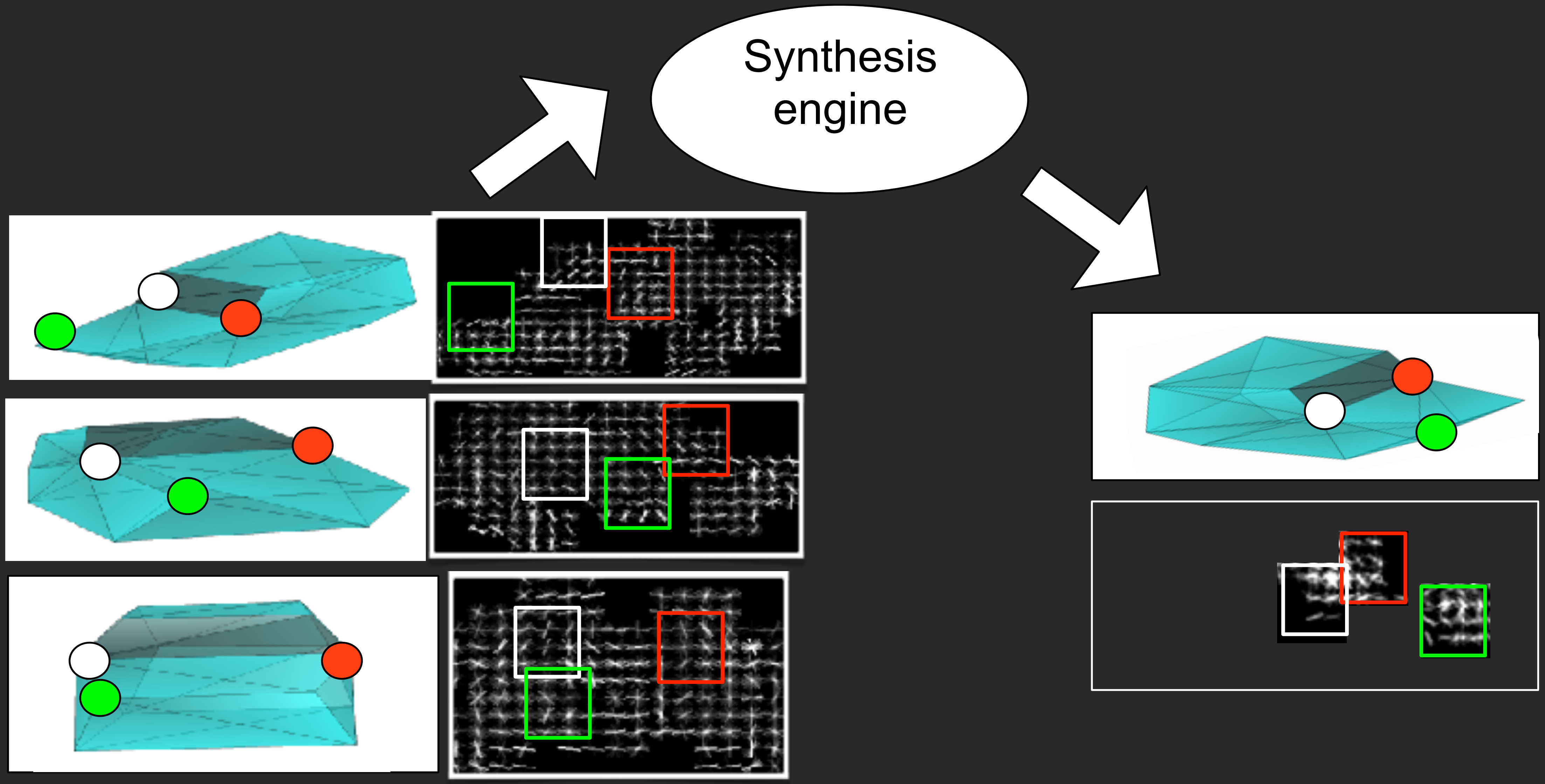
One can train a state-of-art face detector (*c.f.* Google Picassa & Facebook's face.com) with 100 faces!

What if we want to recognize 3D shapes?

Input:
2D image

Output:
3D shape
camera viewpoint

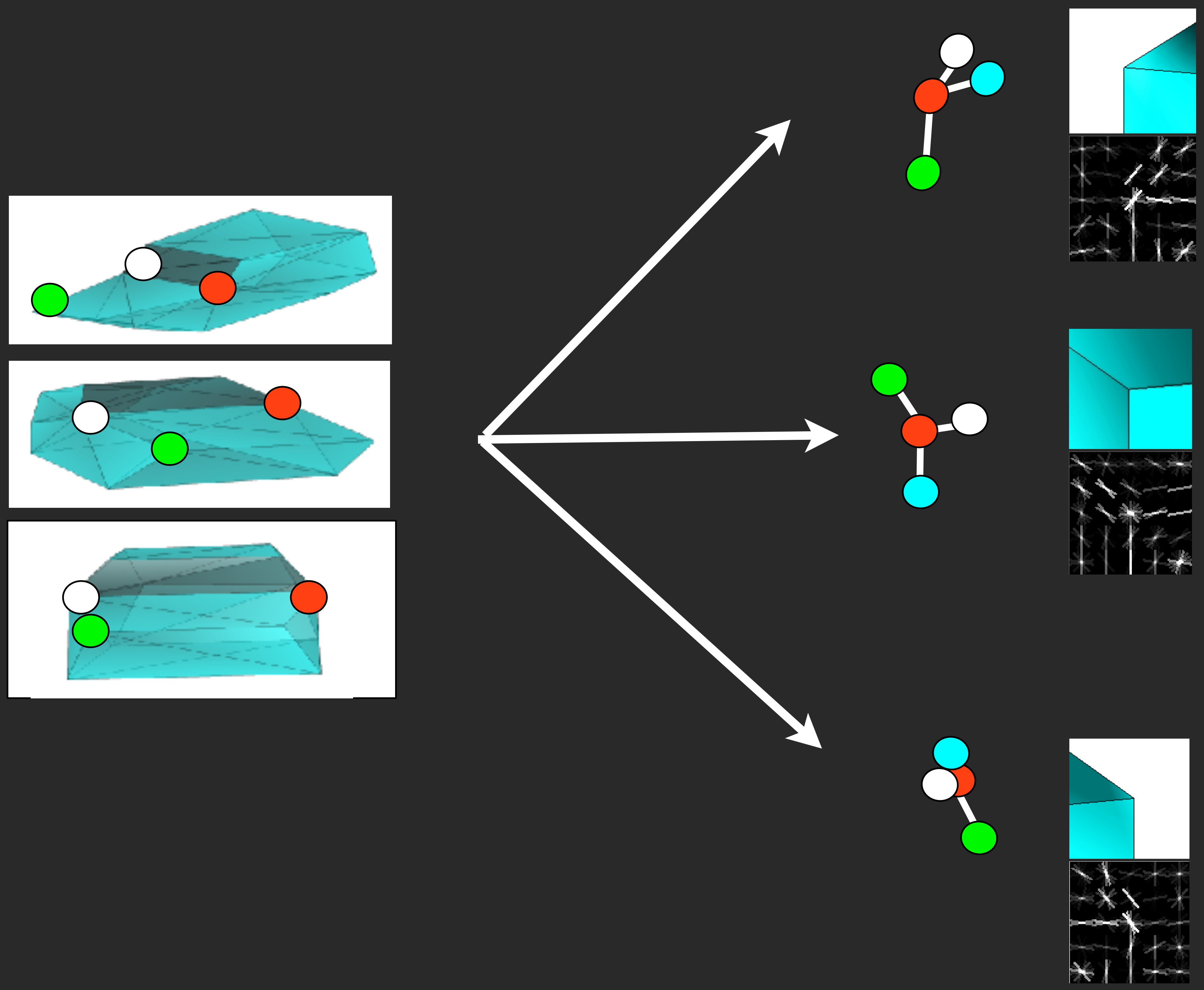
Shape synthesis



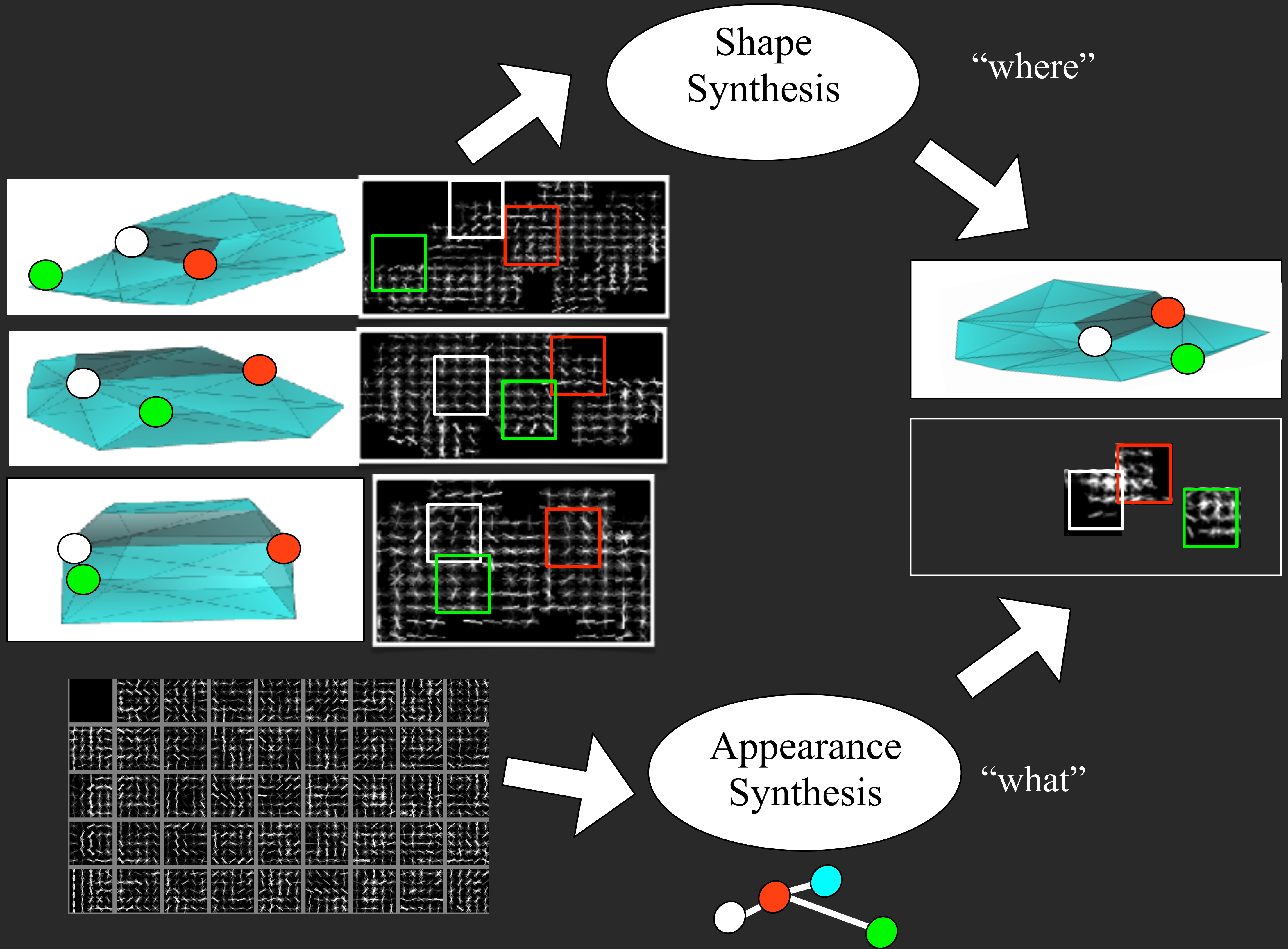
Problem... local appearances depend on global geometry
(foreshortened or occluded wheels look very different)

Geometry-conditioned appearance

Enlarge part dictionary to include parts with different local geometries
(learned by clustering)



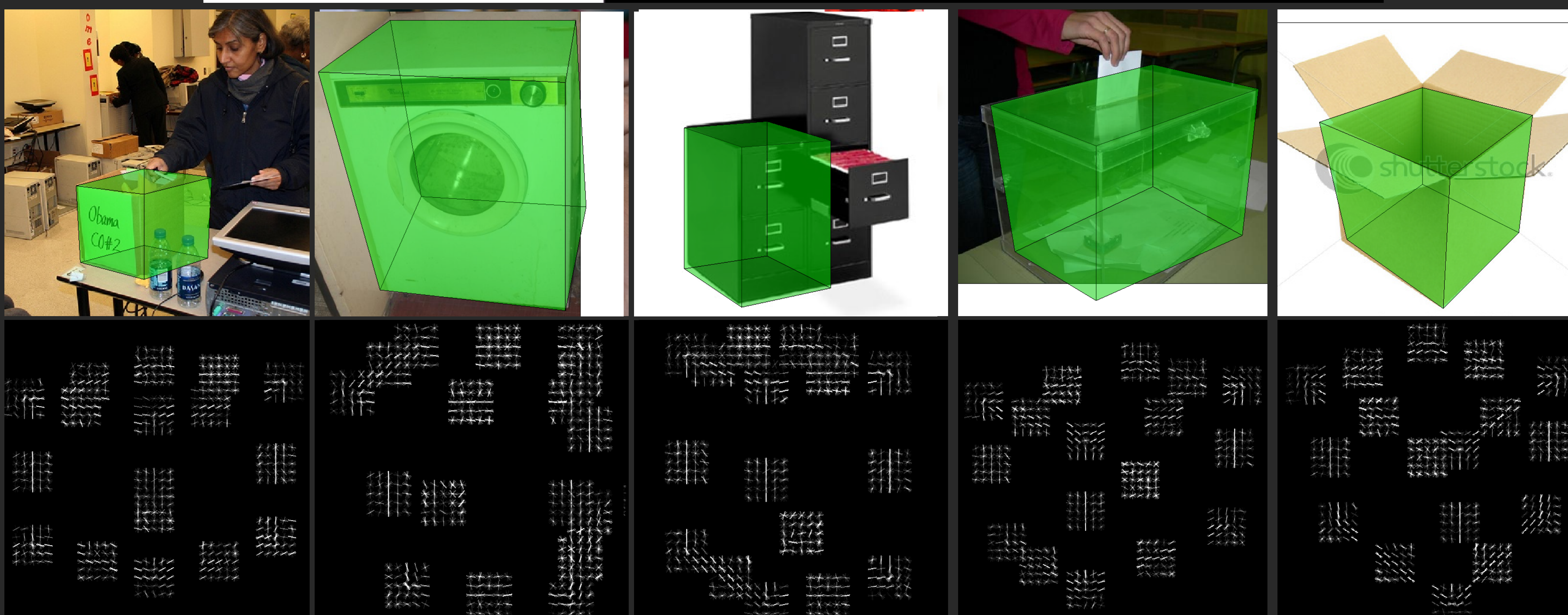
Shape and appearance synthesis



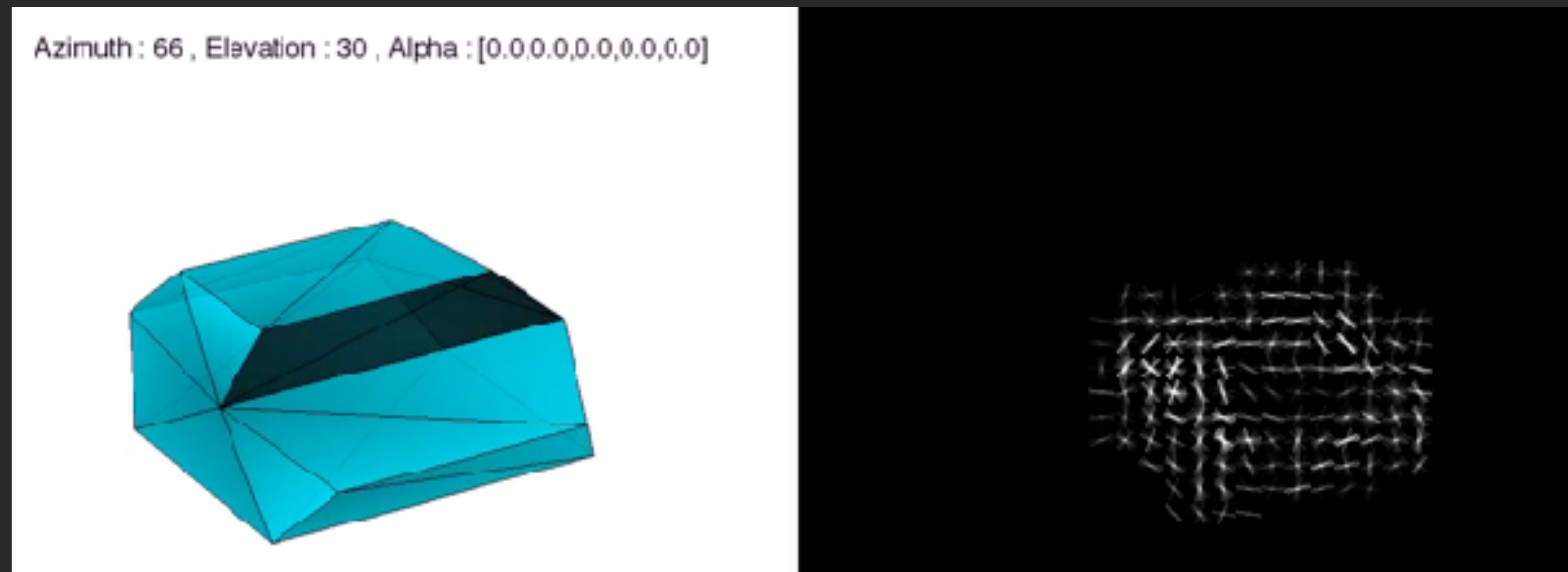
Explicit set of synthesized templates

(Most viewpoints never seen during training)

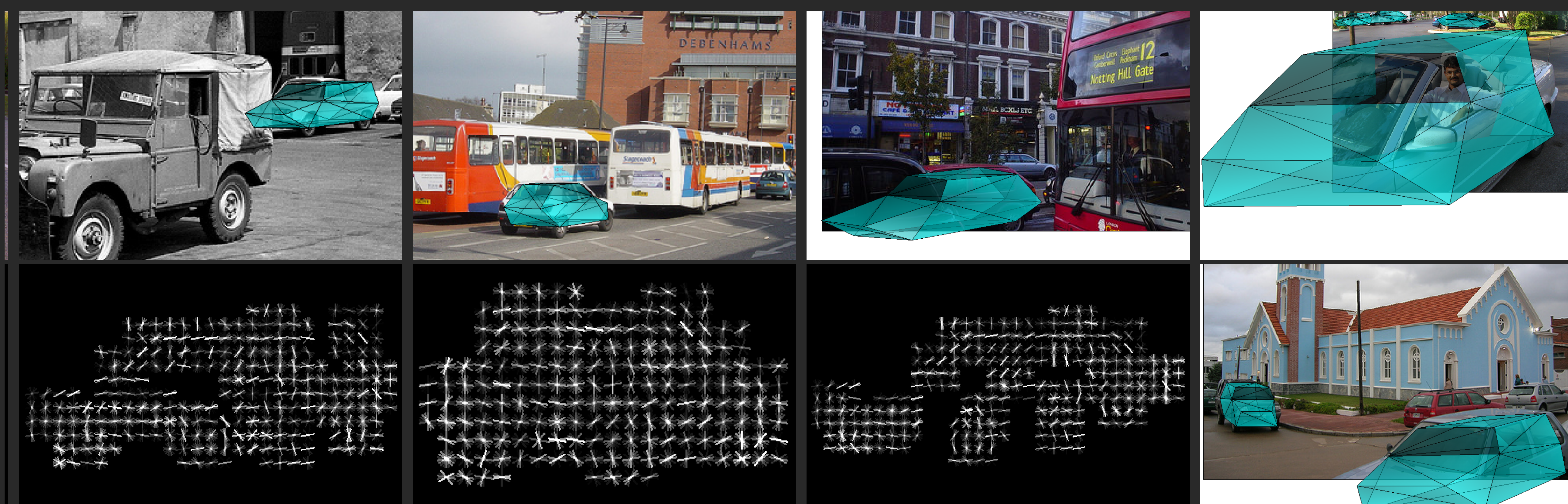
Example detections



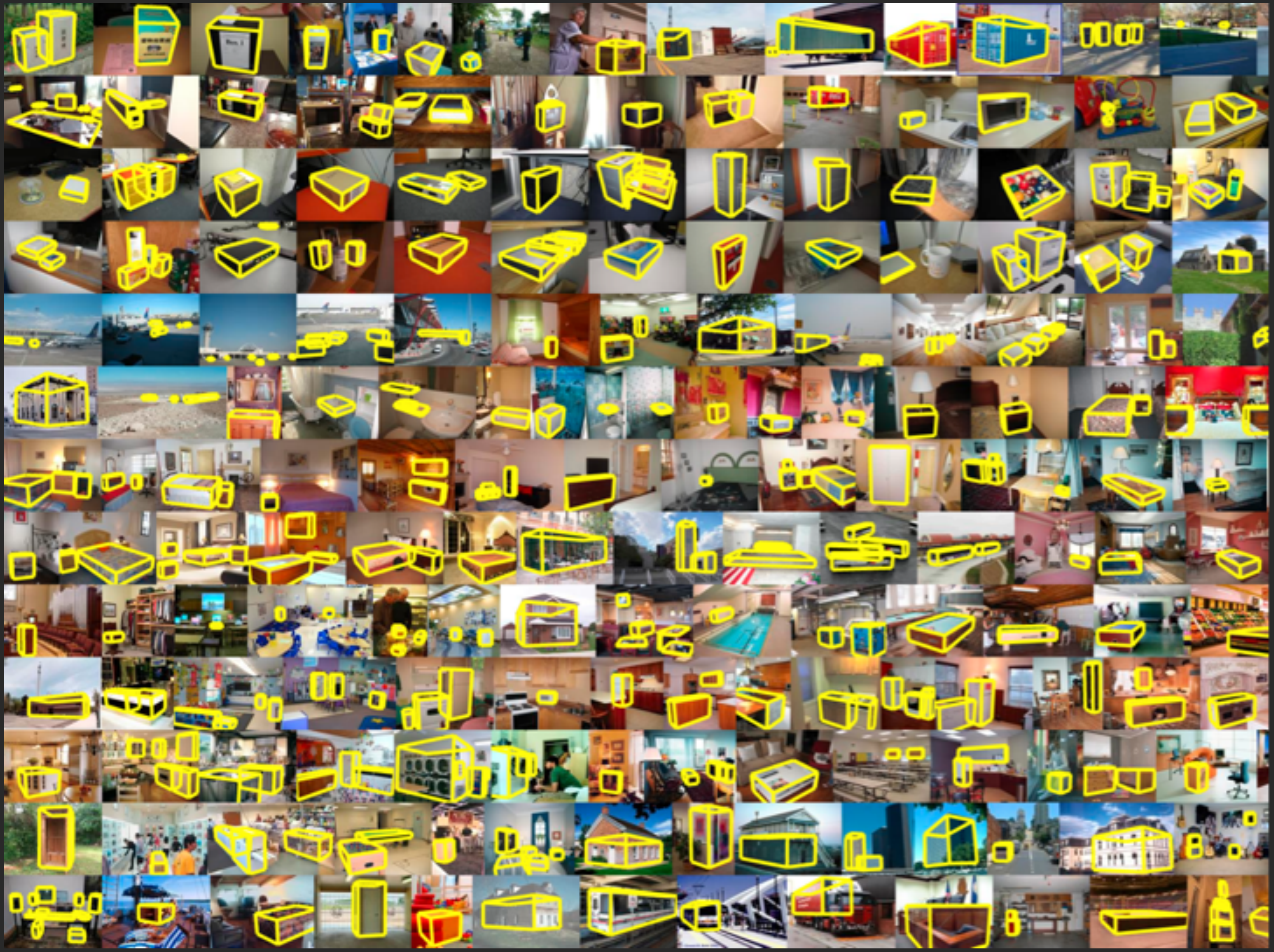
Car detection + reconstruction



parts are beign spatially moved *and* swapped out

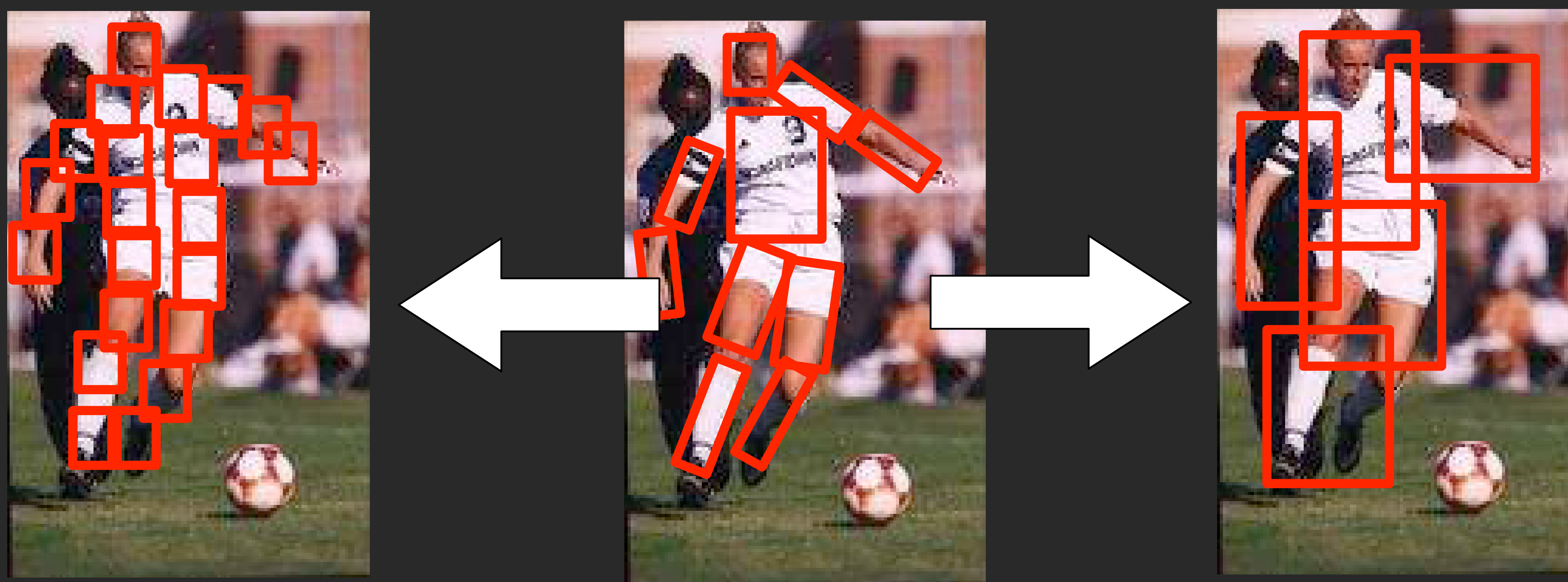


Evaluation: MIT Geometric Primitive Dataset



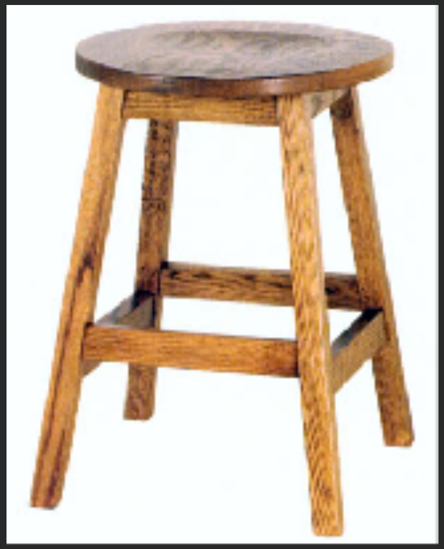
Semantic vs learned representations

What are the right units for sharing and synthesis?



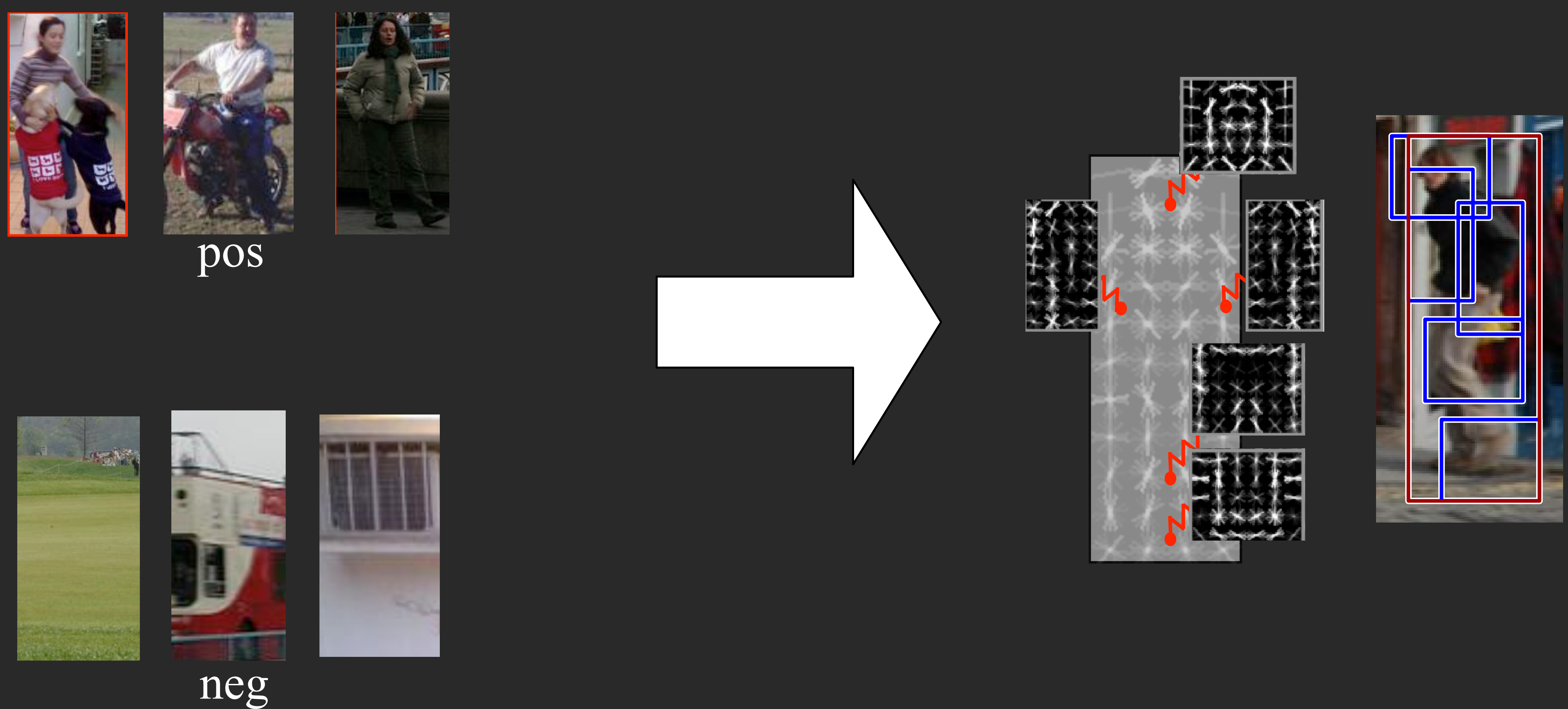
Patches Skeleton “Poselets”

For general objects?



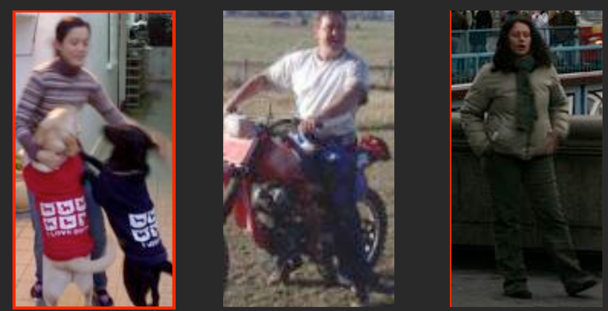

Data-driven parts

Learn parts that allow for accurate recognition

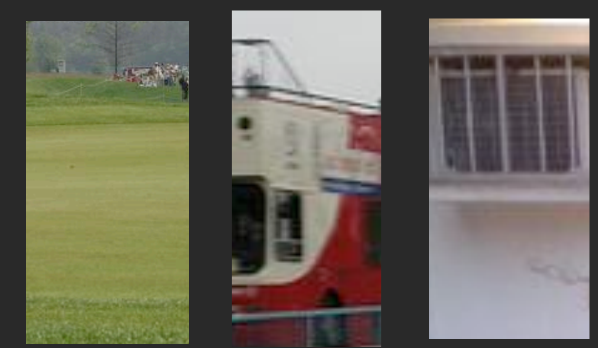


Felzenszwalb, Girshick, McAllester, and Ramanan *PAMI* 2010

SVMs



pos



neg

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

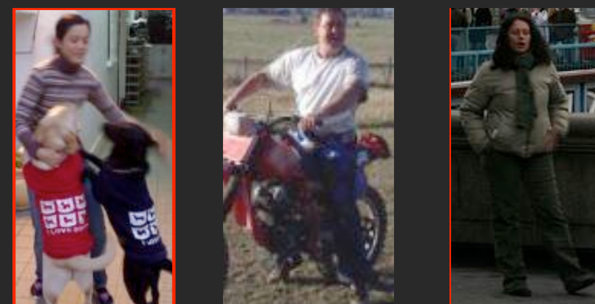
$$L(w) = ||w||^2 + \sum_{n \in \text{pos}} \max(0, 1 - f_w(x_n)) + \sum_{n \in \text{neg}} \max(0, 1 + f_w(x_n))$$

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

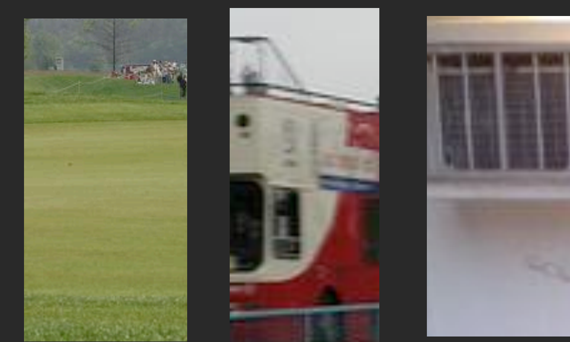
L(w) is convex (Quadratic Program)

Latent SVMs

Felzenszwalb, McAllester, Ramanan *CVPR* 2008



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

Coordinate descent

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

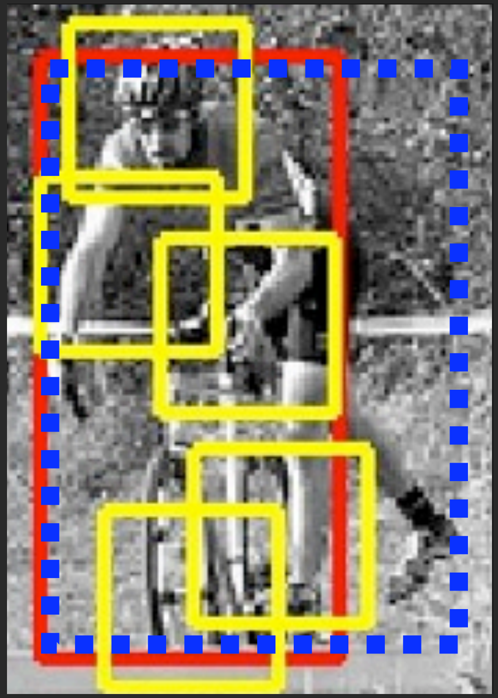
$$w = \underset{w}{\operatorname{argmin}}\, L(w) \quad \text{with fixed} \quad \{z_n : n \in \text{pos}\}$$

2) Given w, estimate part locations on positives

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

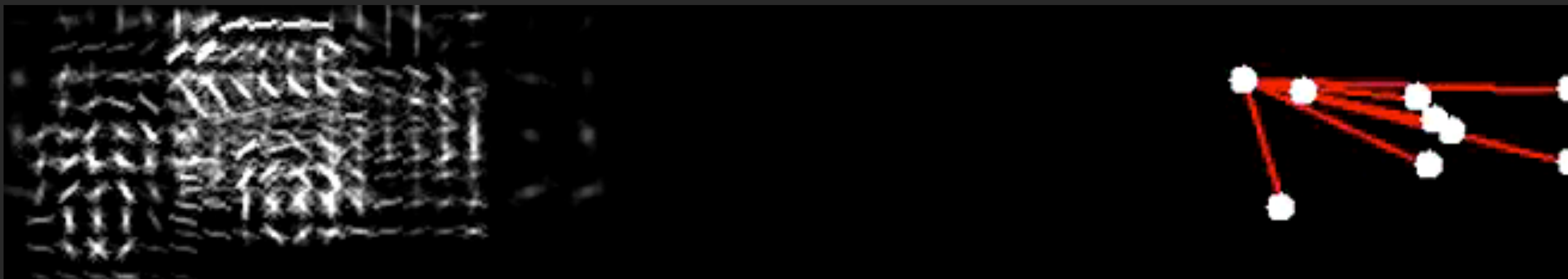
The above steps perform coordinate descent on a joint loss
Can be seen as an instance of the CCCP algorithm (Yuille)

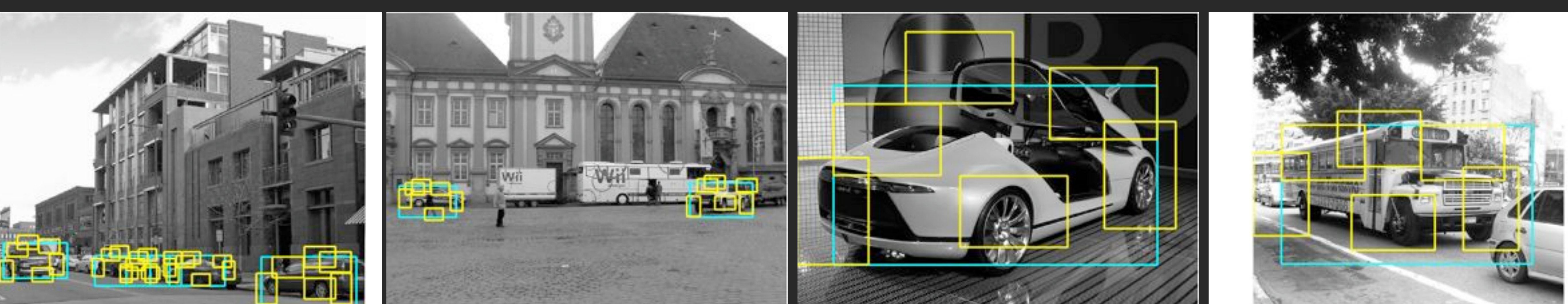
Treat ground-truth labels as partially latent



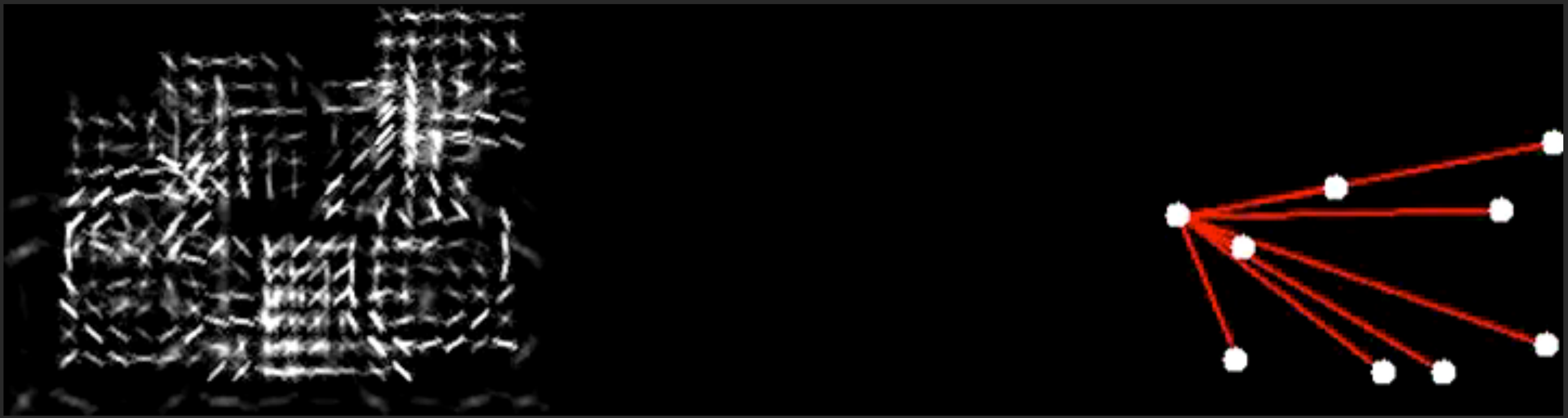
Allows for "cleaning up" of noisy labels (in blue) during iterative learning

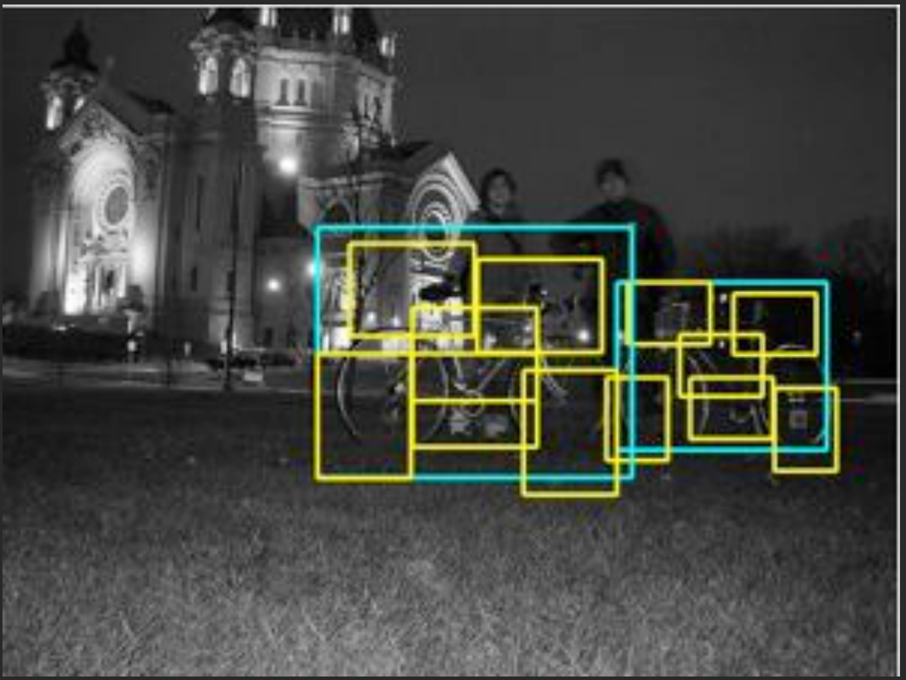
Example models

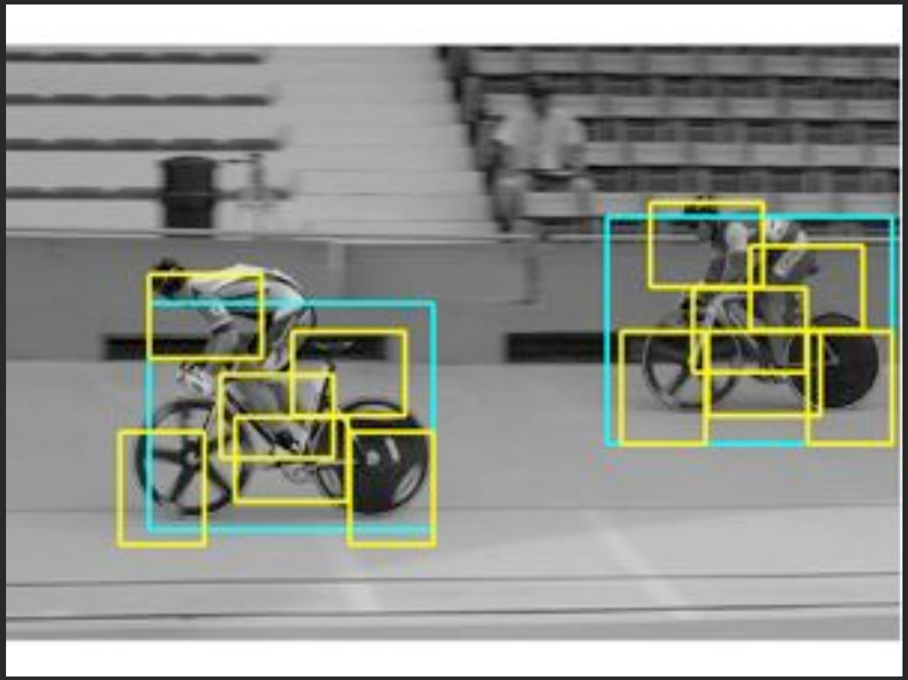




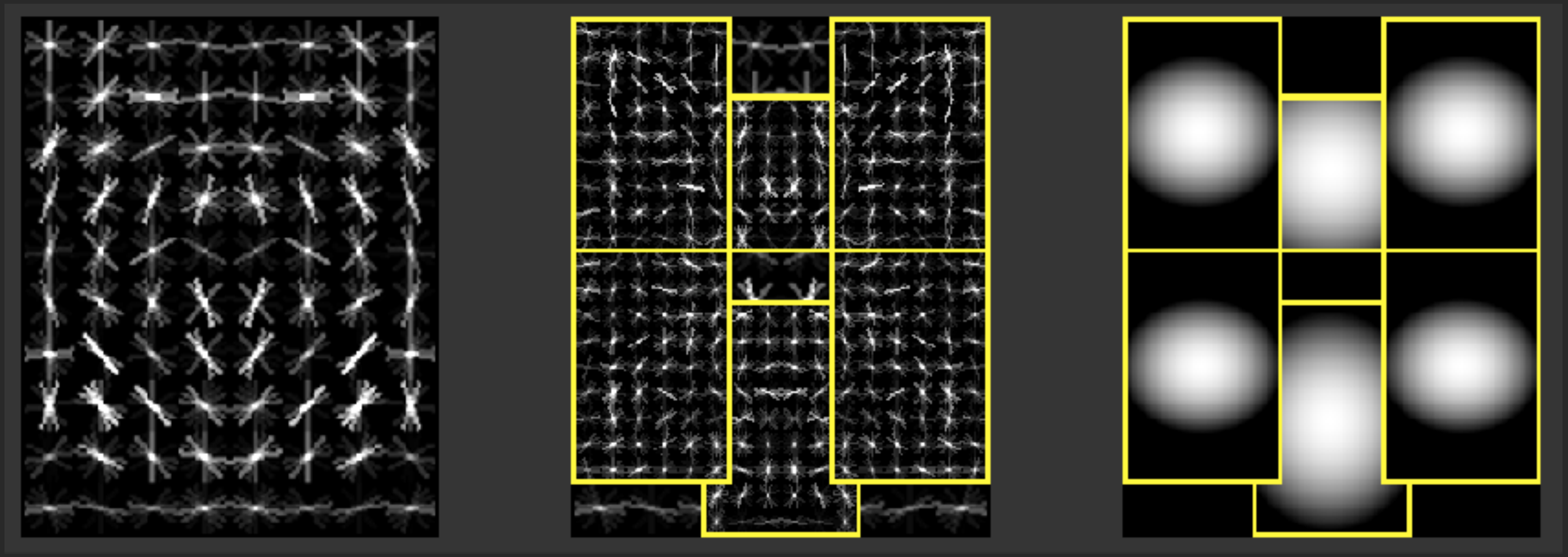
Example models



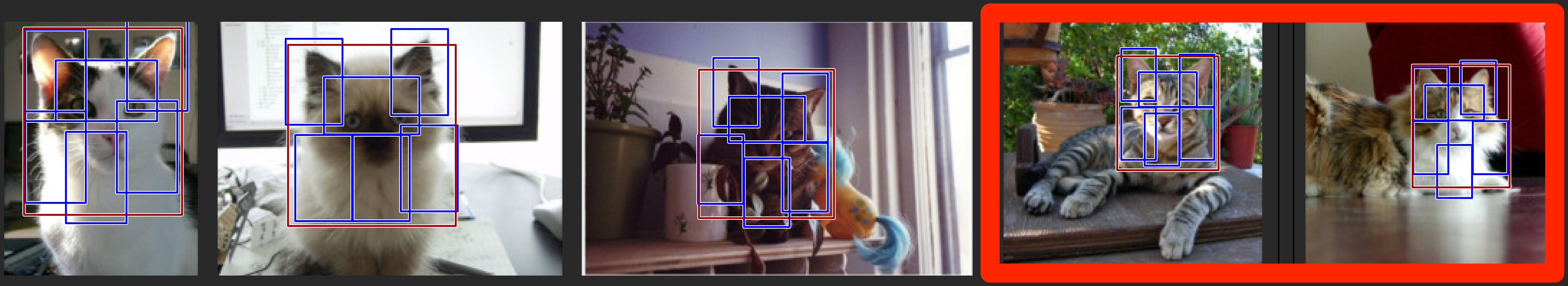




Example models

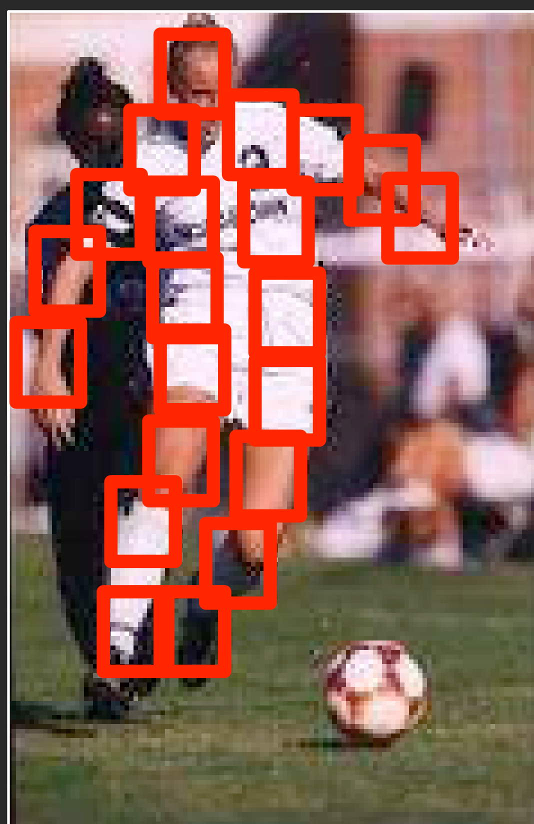


False positive due to imprecise bounding box

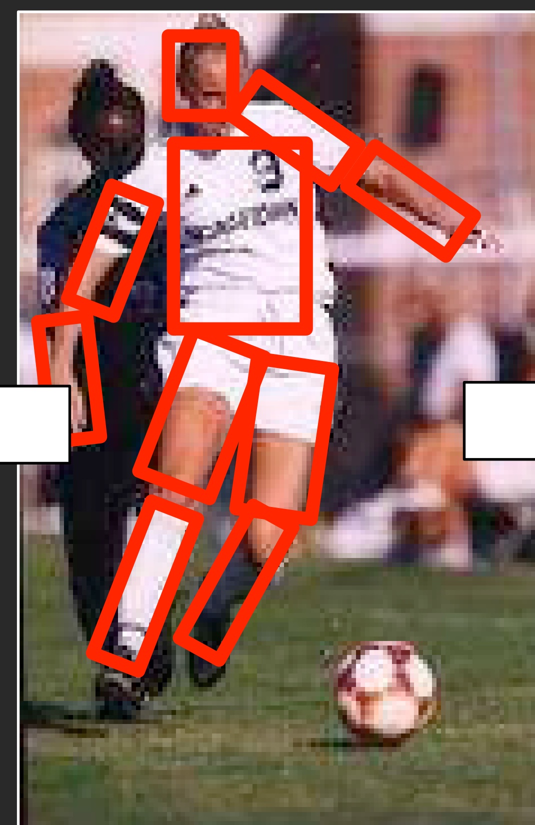


Challenge 1:

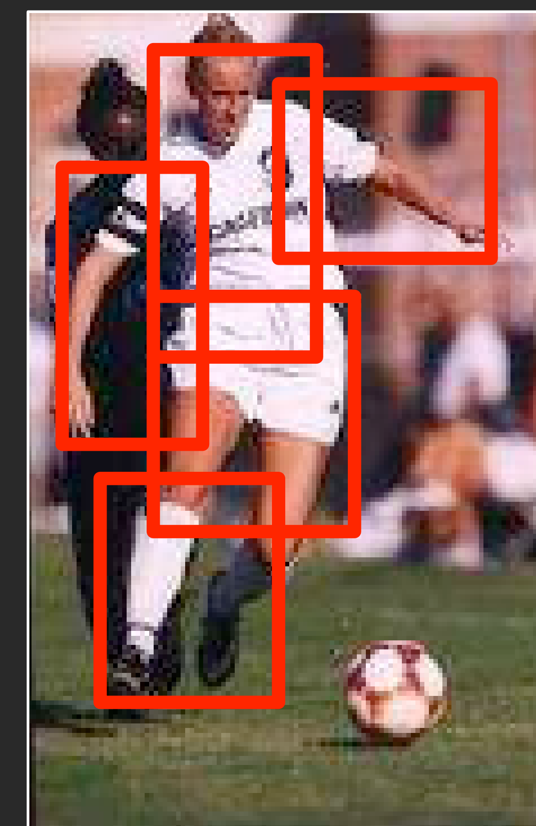
What are the right parts?



Patches



Skeleton

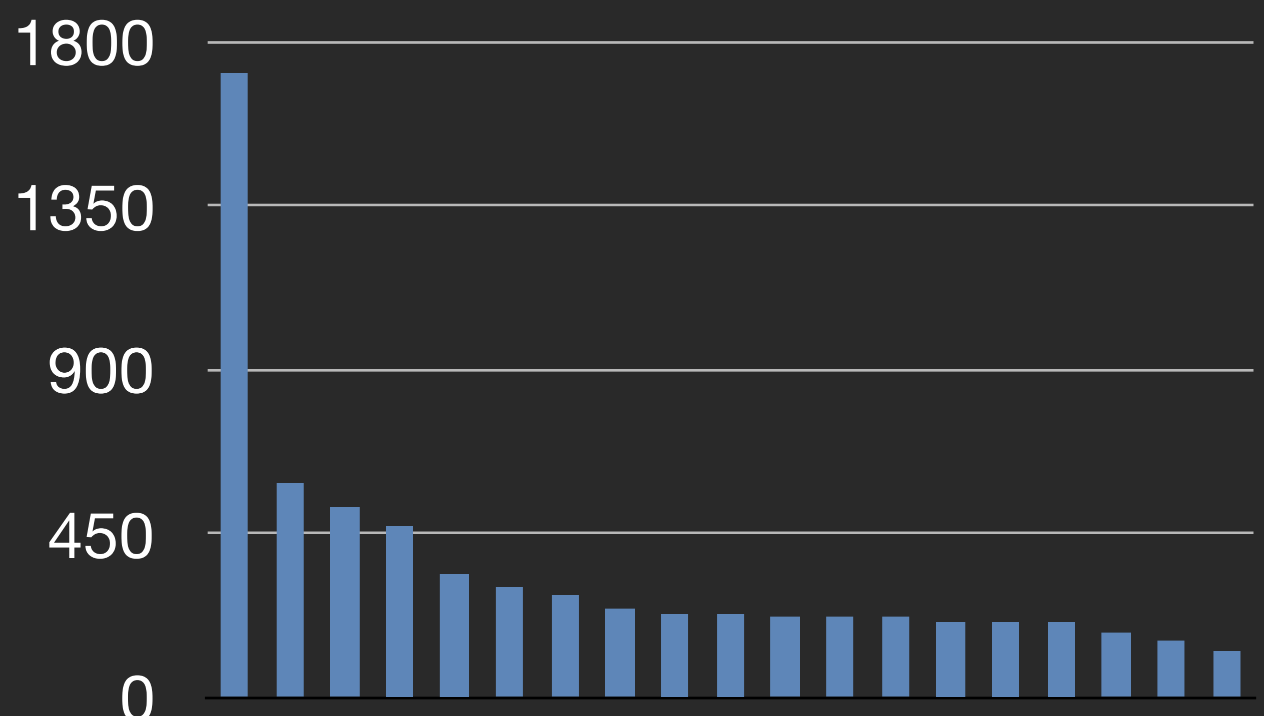


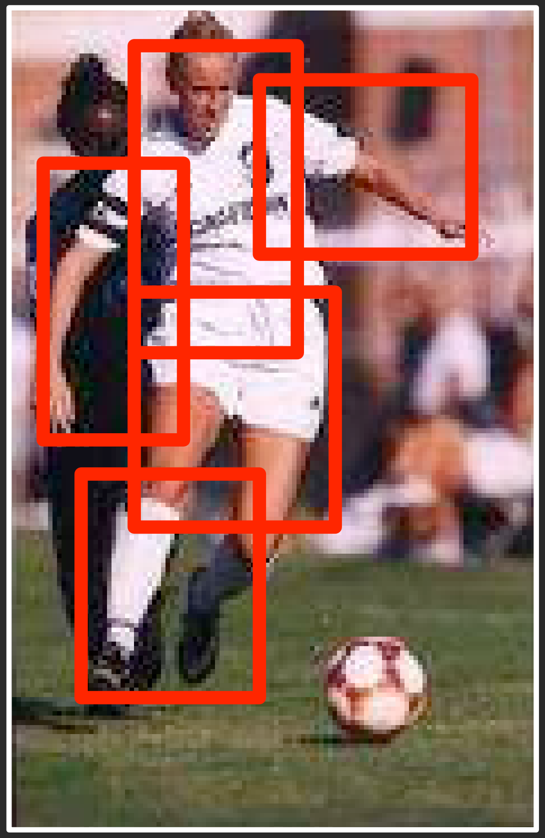
"Poselets"

Are "computer graphics" primitives the right choice?

Challenge 2:

How to deal with long-tail distribution of part types?



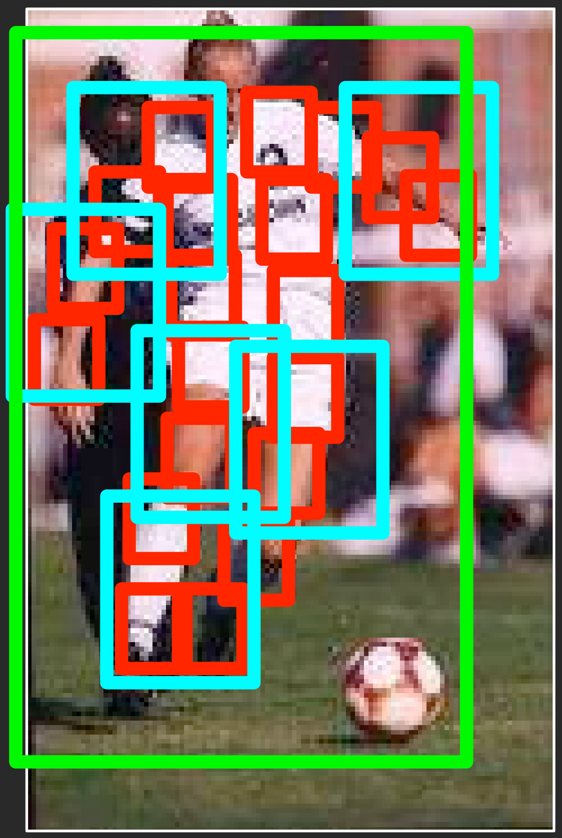


Solution 1+2: latent hierarchical (or "deep") models

Inference on such models requires layers of convolution and max-pooling

(we've *almost* derived a convolutional neural net)

Latent hierarchical models

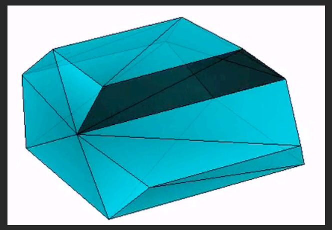


$$S(x,z) = \sum_{i \in V} w_i \cdot \phi(x, z_i) + \sum_{ij \in E} w_{ij} \cdot \psi(z_i, z_j)$$

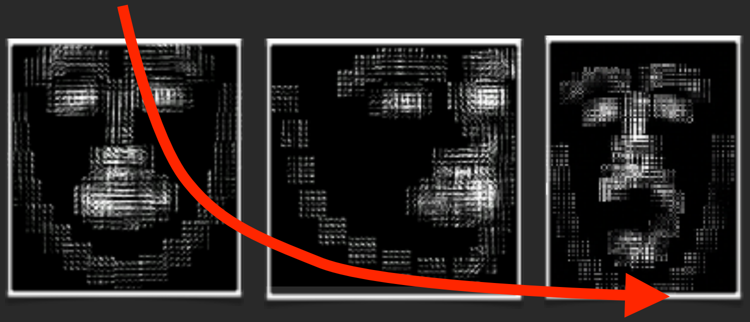
Next lecture (deep models): define z_{in} to be binary variable that specify if (sub)part i is found at location n

Parts: a look back

Recognition through reconstruction: latent-variable classification



Sharing + synthesis: zero & one-shot learning for tails



Representation learning: part discovery