Pictorial structures for object recognition

Josef Sivic

http://www.di.ens.fr/~josef

Equipe-projet WILLOW, ENS/INRIA/CNRS UMR 8548

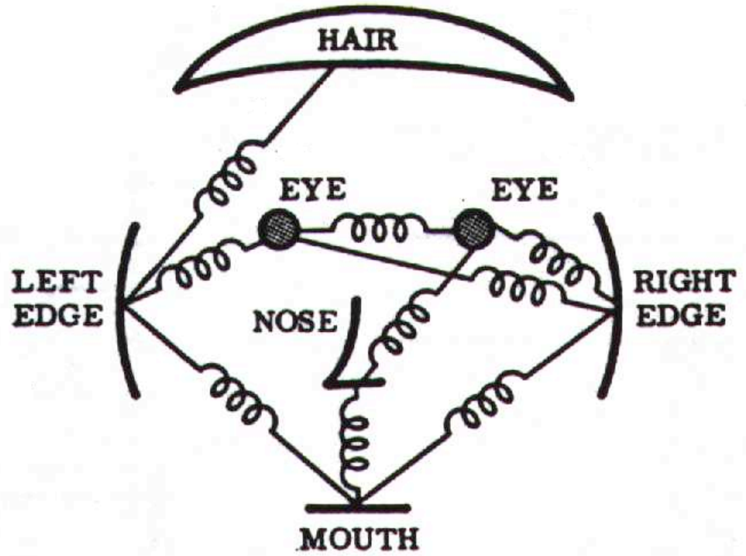
Laboratoire d'Informatique, Ecole Normale Supérieure, Paris

With slides from: A. Zisserman,

M. Everingham and P. Felzenszwalb

Pictorial Structure

- Intuitive model of an object
- Model has two components
  1. parts (2D image fragments)
  2. structure (configuration of parts)
- Dates back to Fischler & Elschlager 1973

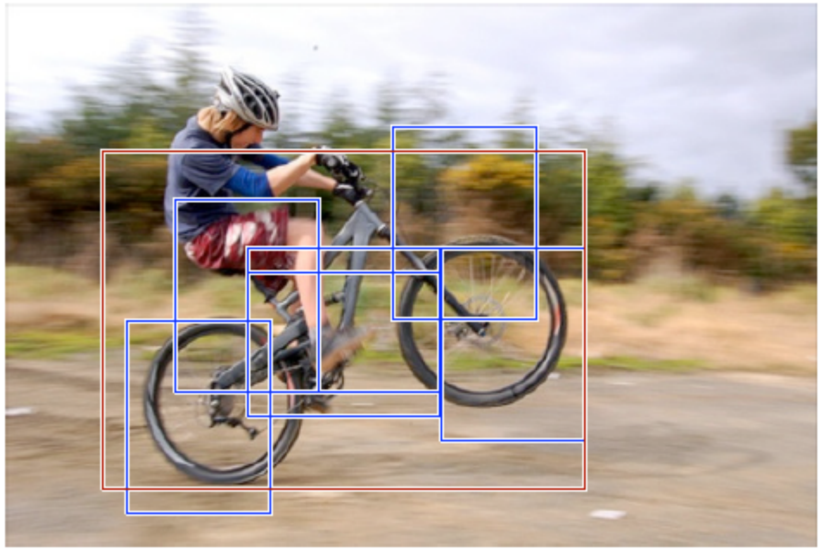


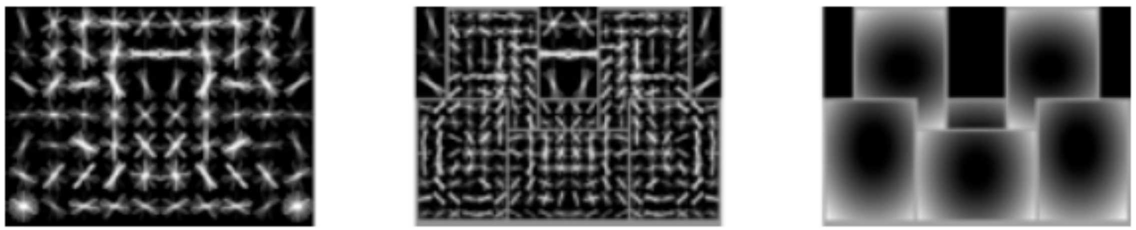
Recall : Generative part-based models (Lecture 7)

R. Fergus, P. Perona and A. Zisserman,
Object Class Recognition by Unsupervised Scale-Invariant Learning, CVPR 2003

Recall: Discriminative part-based model (Lecture 9)

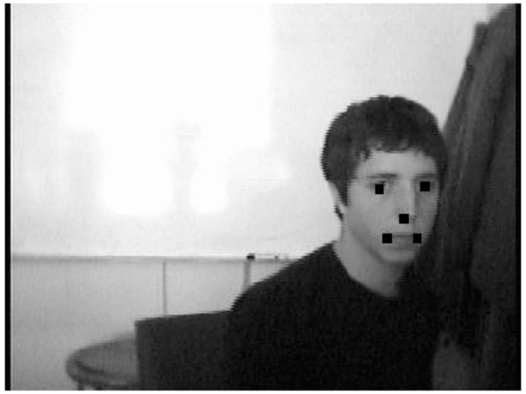
[Felsenszwalb et al. 2009]





Localize multi-part objects at arbitrary locations in an image

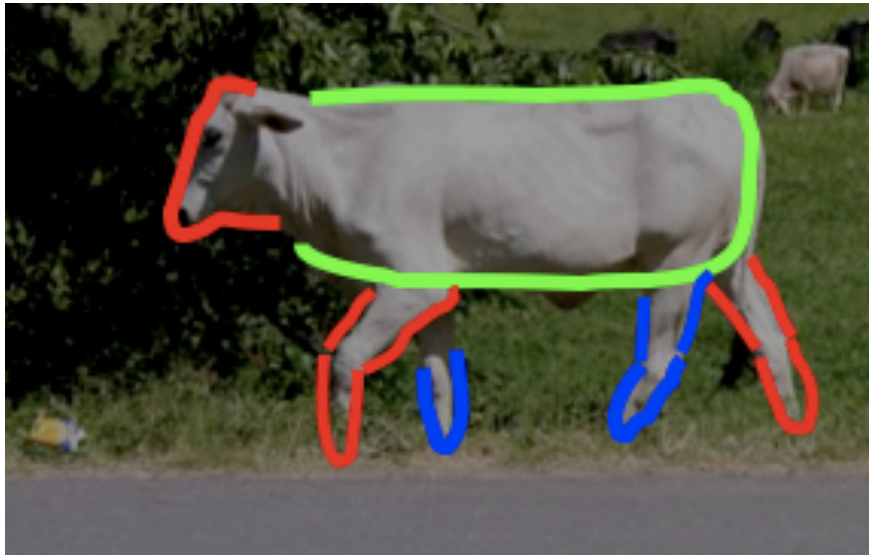
- Generic object models such as person or car
- Allow for articulated objects
- Simultaneous use of appearance and spatial information
- Provide efficient and practical algorithms



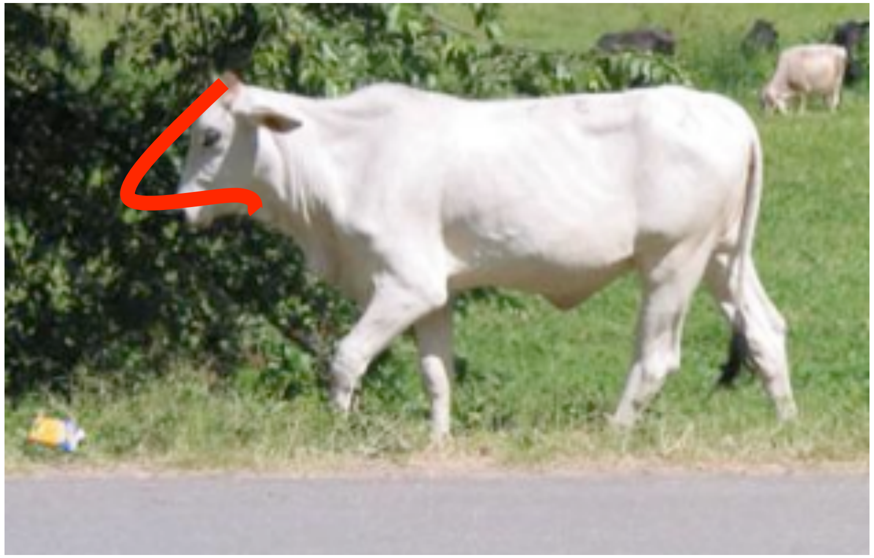
To fit model to image: minimize an energy (or cost) function that reflects both

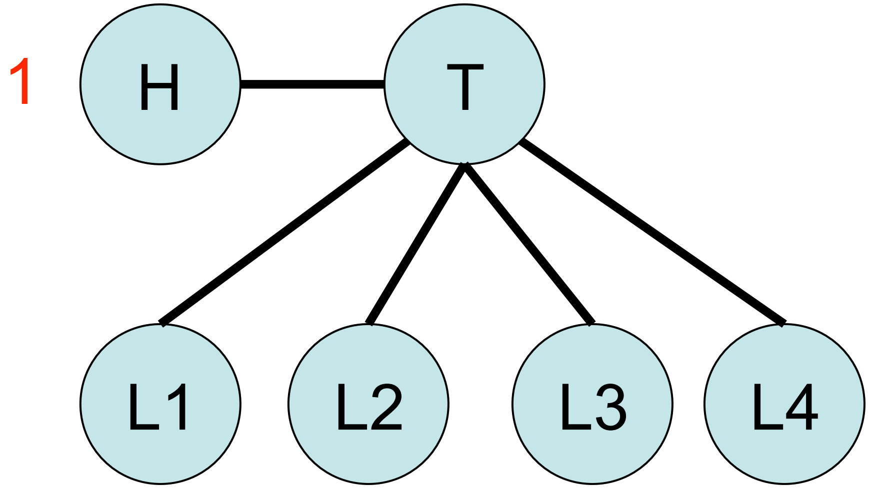
- Appearance: how well each part matches at given location
- Configuration: degree to which parts match 2D spatial layout

Example: cow layout



Example: cow layout





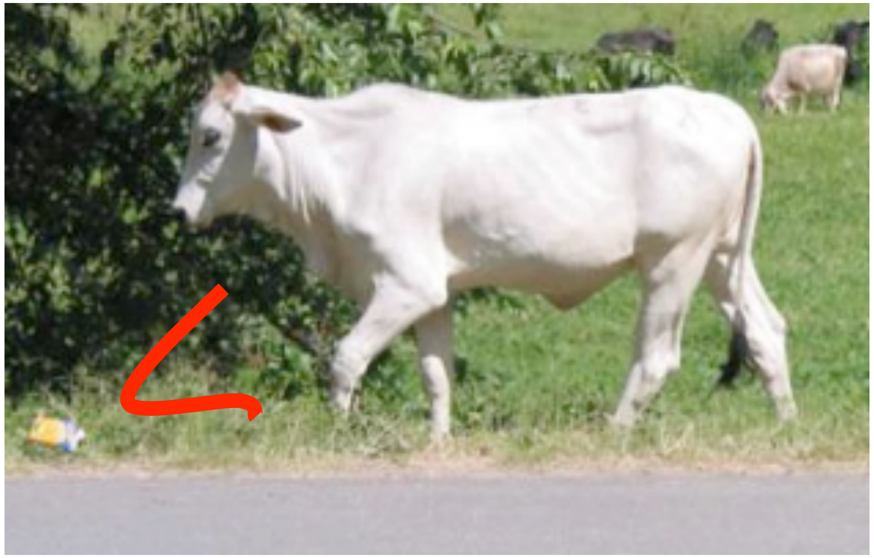
Graph G = (V,E)

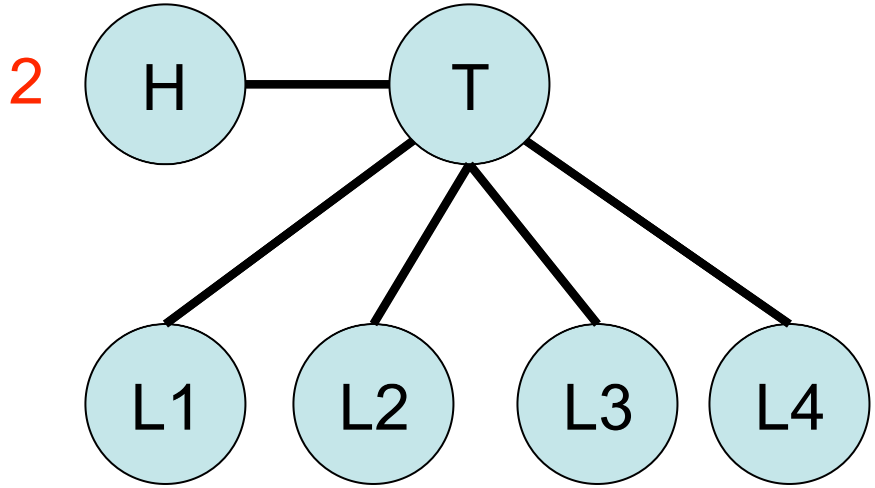
Each vertex corresponds to a part - ‘Head’, ‘Torso’, ‘Legs’

Edges define a TREE

Assign a label to each vertex from H = {positions}

Example: cow layout





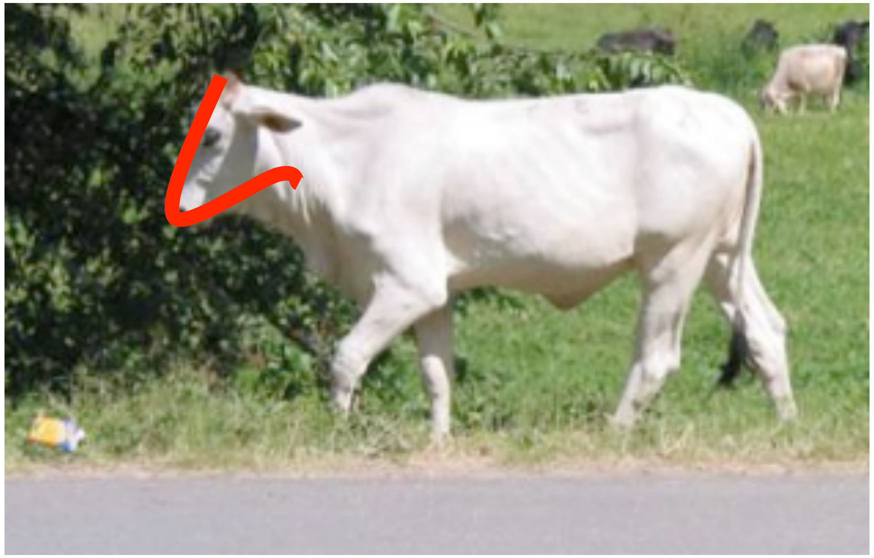
Graph G = (V,E)

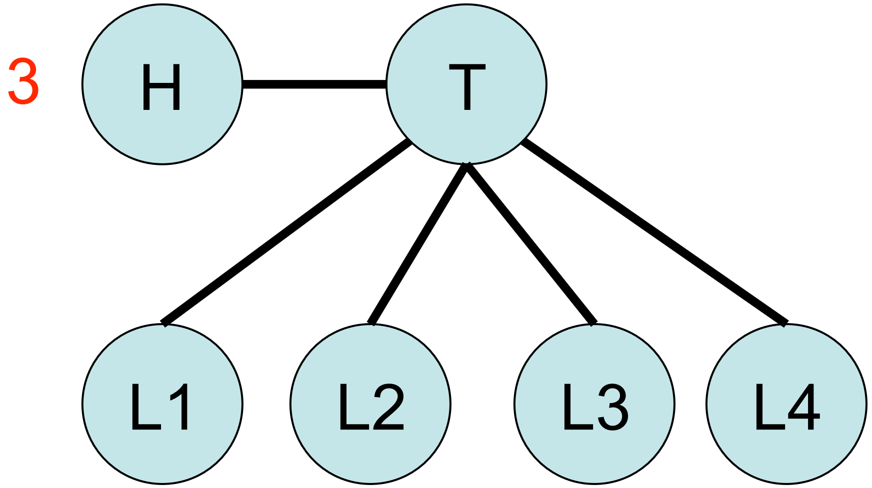
Each vertex corresponds to a part - ‘Head’, ‘Torso’, ‘Legs’

Edges define a TREE

Assign a label to each vertex from H = {positions}

Example: cow layout





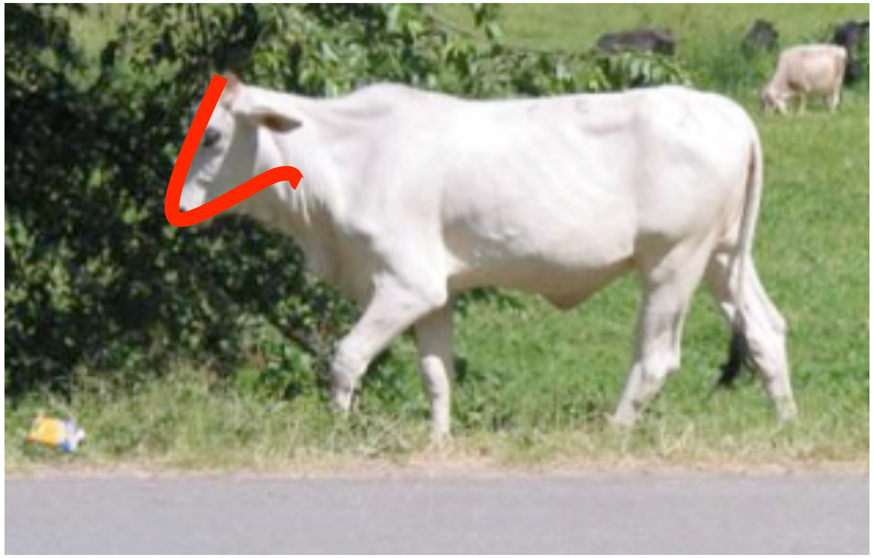
Graph G = (V,E)

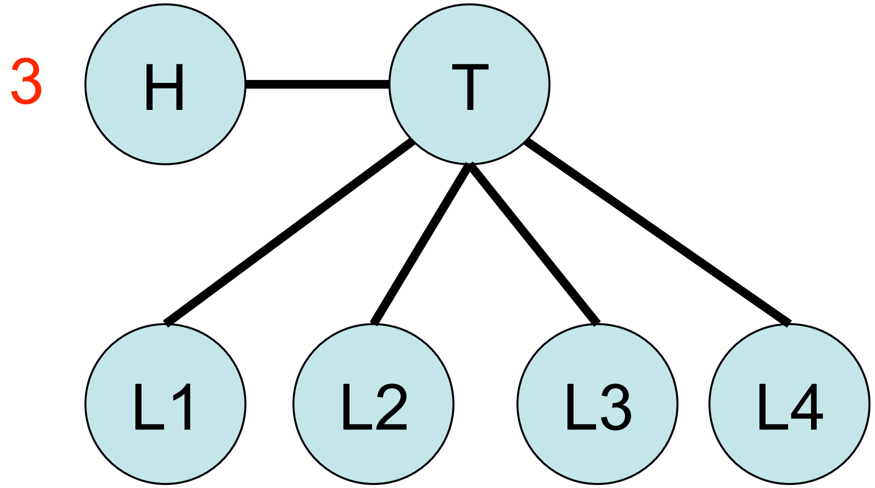
Each vertex corresponds to a part - ‘Head’, ‘Torso’, ‘Legs’

Edges define a TREE

Assign a label to each vertex from H = {positions}

Example: cow layout





Graph G = (V,E)

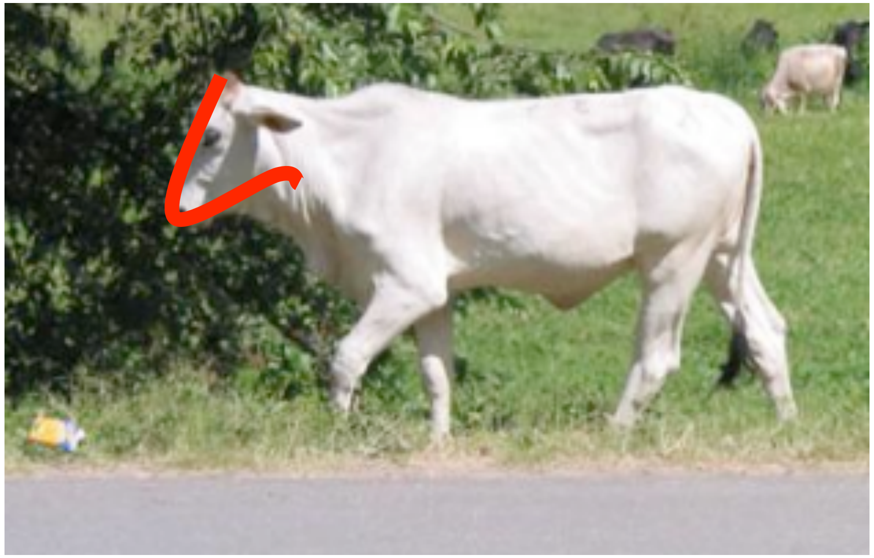
Cost of a labelling L : V → H

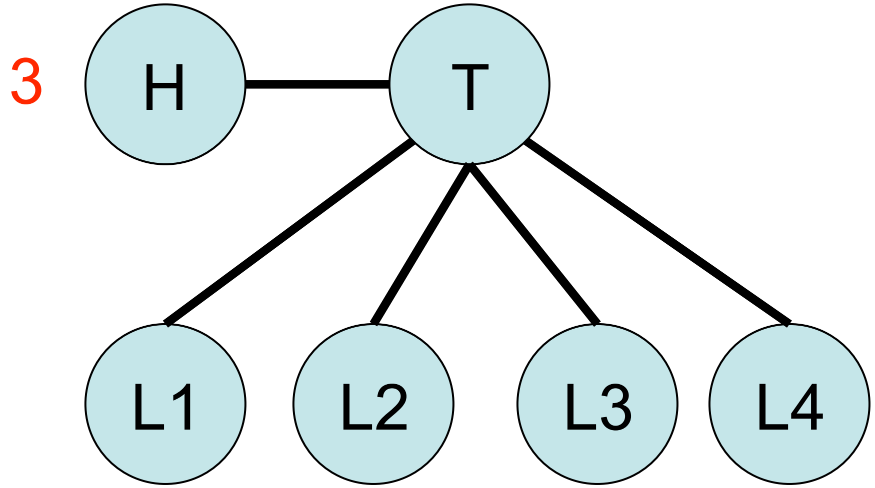
Unary cost : How well does part match image patch?

Pairwise cost : Encourages *valid configurations*

Find best labelling L*

Example: cow layout



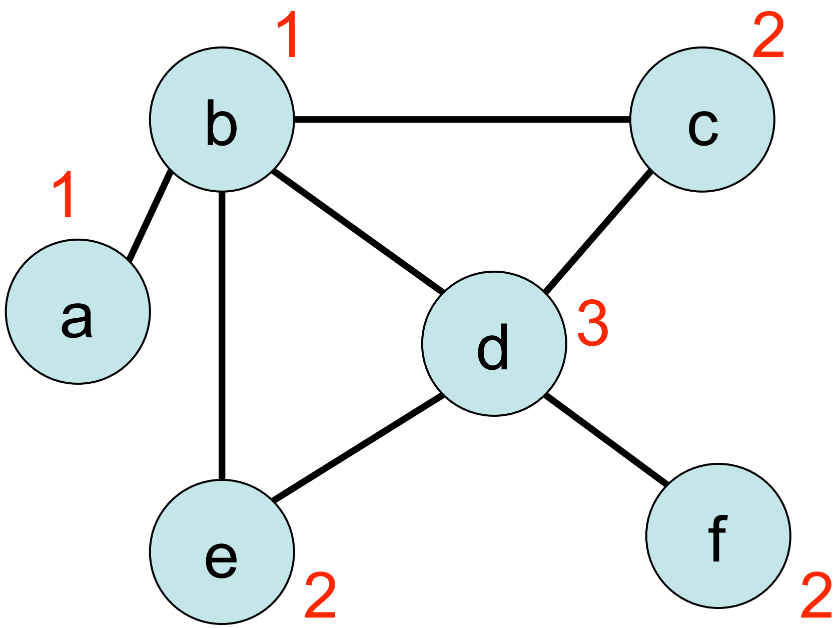


Graph G = (V,E)

Find best labelling L* by minimizing energy:

$$L^* = \arg\min_L \left(\sum_{i=1}^{n} m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j) \right)$$

The General Problem



Graph G = (V, E)

Discrete label set H = {1,2,…,h}

Assign a label to each vertex
L: V ➜ H

Cost of a labelling E(L)

Unary Cost + n-nary cost (depends on the size of maximal cliques of the graph)

Find L* = arg min E(L) [Bishop, 2006]

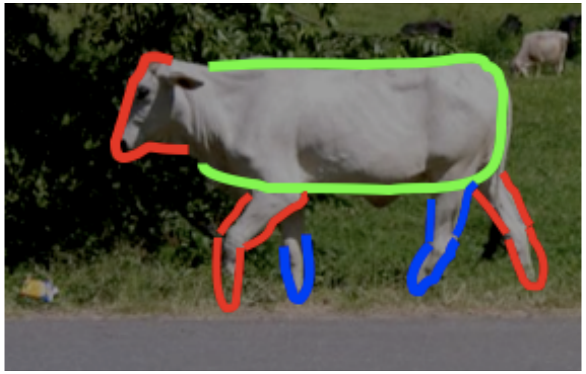
Computational Complexity

Fitting

$$|H|^{|V|} = h^n$$

n parts

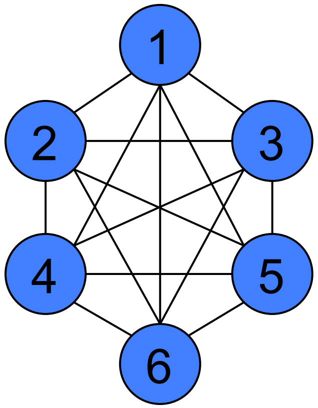
h positions



e.g. h = number of pixels (512x300) ≈ 153600

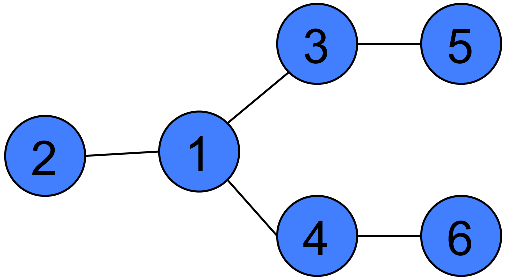
Different graph structures

Can use dynamic programming



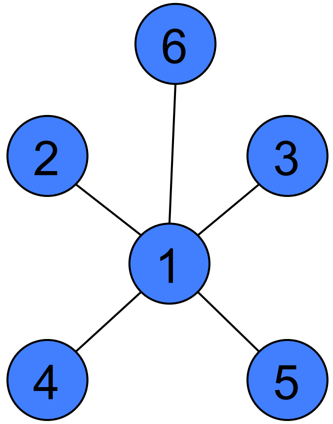
Fully connected

$O(h^n)$



Tree structure

$O(nh^2)$



Star structure

$O(nh^2)$

n parts

h positions (e.g. every pixel for translation)

Brute force solutions intractable

- With n parts and h possible discrete locations per part, $O(h^n)$
- For a tree, using dynamic programming this reduces to $O(nh^2)$

If model is a tree and has quadratic edge costs then complexity reduces to O(nh) (using a distance transform)

Felzenszwalb & Huttenlocher, *IJCV, 2004*

Distance transforms for DP

Special case of DP cost function

Distance transforms

- $O(nh^2) \rightarrow O(nh)$ for DP cost functions
- Assume model is quadratic, i.e. $\phi(x_{k-1}, x_k) = \lambda^2 (x_{k-1} - x_k)^2$

Recall that we need to compute

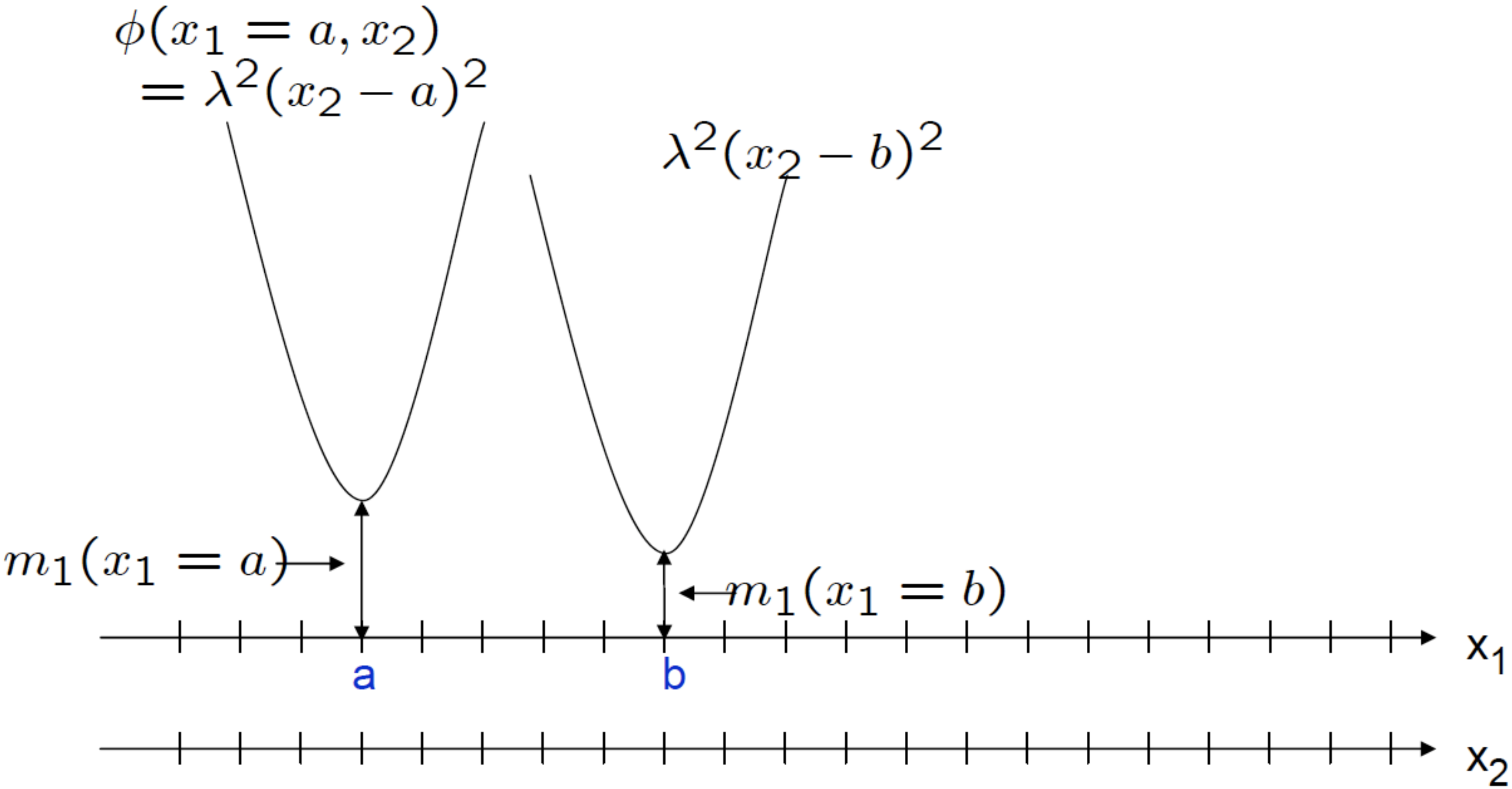
$$\min_{x_{k-1}} \{S_{k-1}(x_{k-1}) + \phi(x_{k-1}, x_k)\}$$

e.g. for $k = 2$, compute for each value of x_2

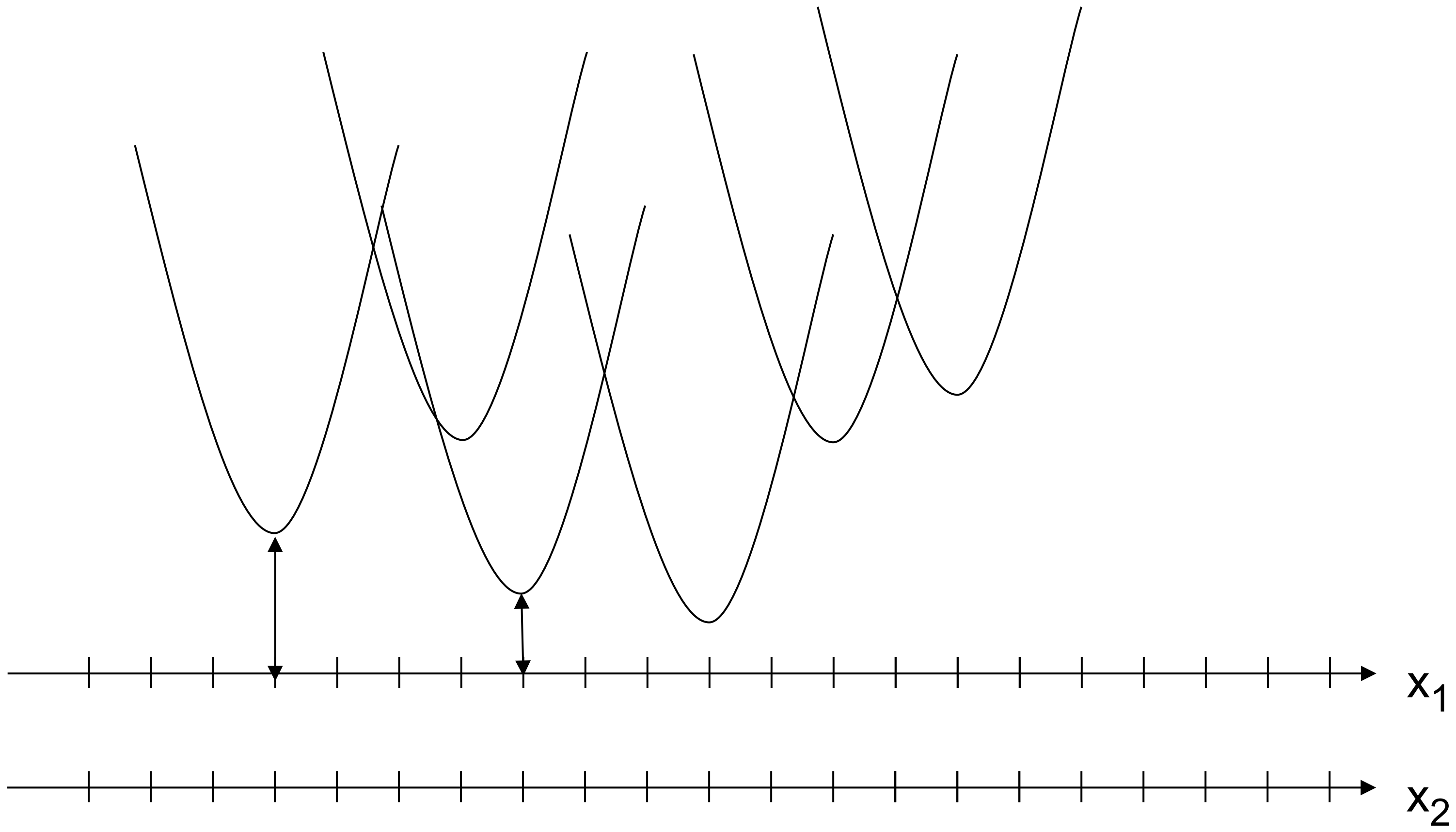
$$\min_{x_1} \{m_1(x_1) + \phi(x_1, x_2)\}$$

Plot $\min_{x_1} \{m_1(x_1) + \phi(x_1, x_2)\}$ as function of x_2

Plot $\min_{x_1}\{m_1(x_1) + \phi(x_1, x_2)\}$ as function of x_2



Plot $\min_{x_1}\{m_1(x_1) + \phi(x_1, x_2)\}$ as function of x_2

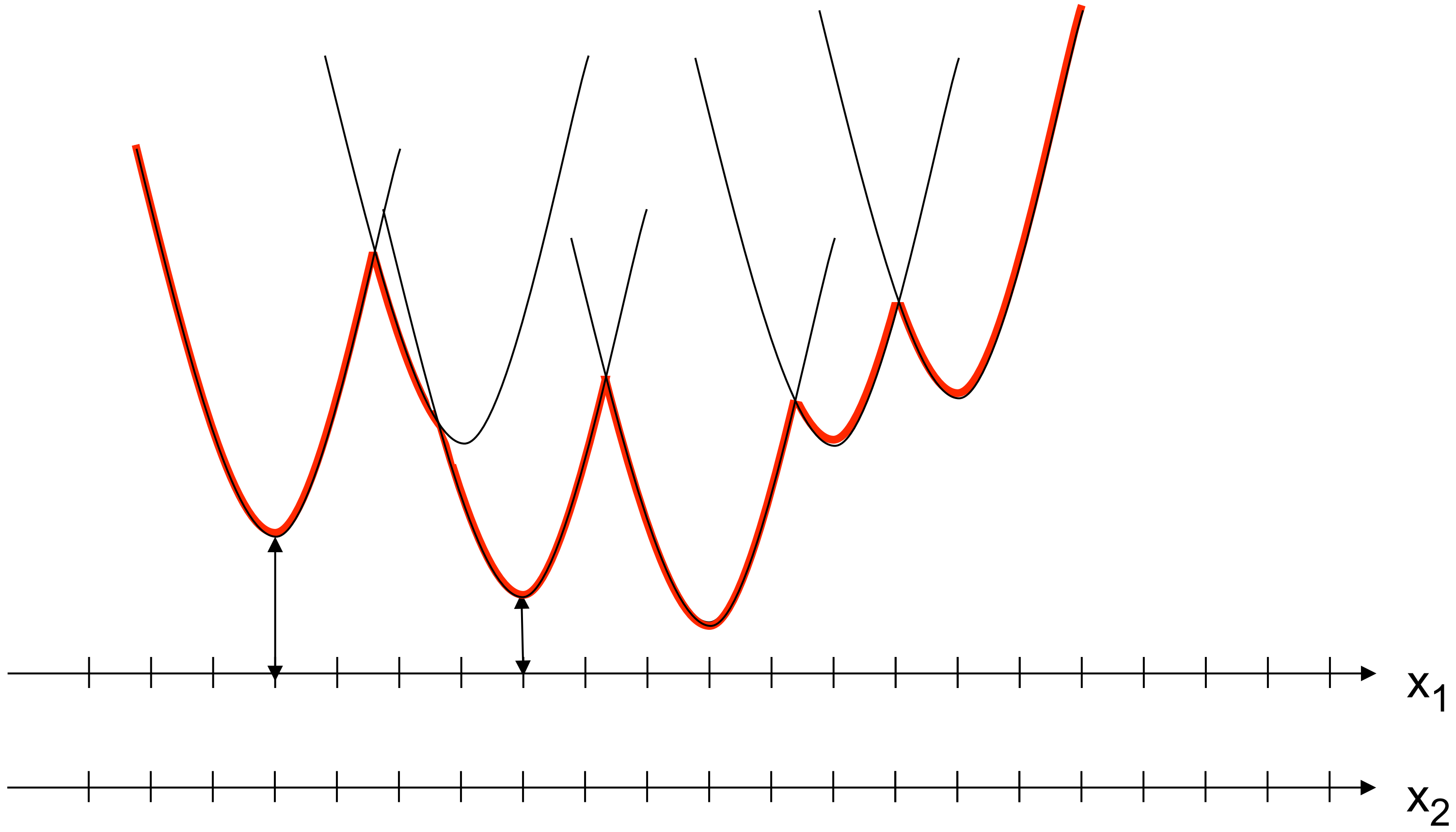


For each x_2

- Finding min over x_1 is equivalent finding minimum over set of offset parabolas
- Lower envelope computed in O(h) rather than O(h^2) via distance transform

Felzenszwalb and Huttenlocher '05

Plot $\min_{x_1}\{m_1(x_1) + \phi(x_1, x_2)\}$ as function of x_2

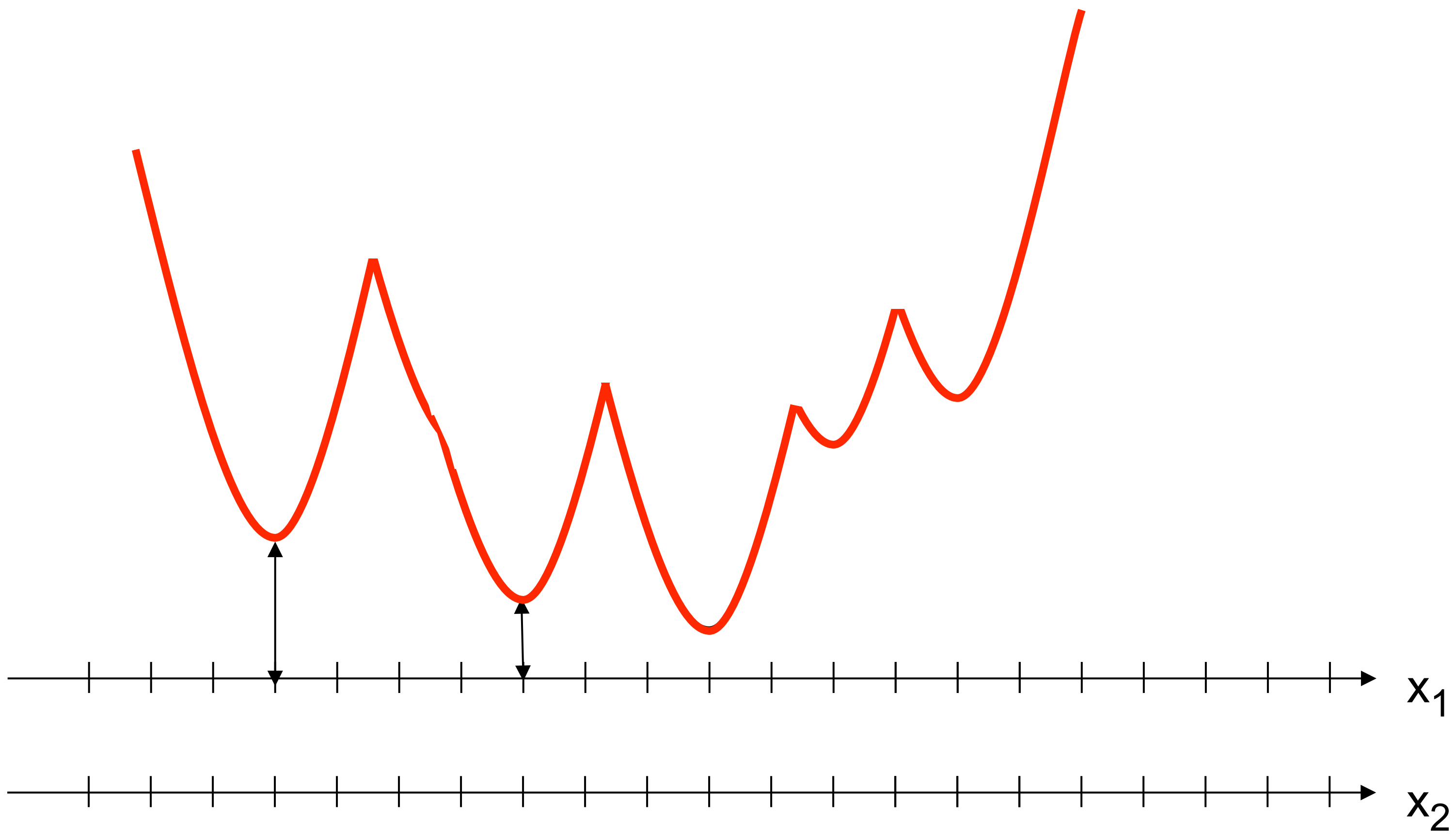


For each x_2

- Finding min over x_1 is equivalent finding minimum over set of offset parabolas
- Lower envelope computed in $O(h)$ rather than $O(h^2)$ via distance transform

Felzenszwalb and Huttenlocher ’05

Plot $\min_{x_1}\{m_1(x_1) + \phi(x_1, x_2)\}$ as function of x_2



For each x_2

- Finding min over x_1 is equivalent finding minimum over set of offset parabolas
- Lower envelope computed in O(h) rather than O(h^2) via distance transform

Felzenszwalb and Huttenlocher ’05

Generalized distance transform

Given a function $f : \mathcal{G} \to \mathbb{R}$,

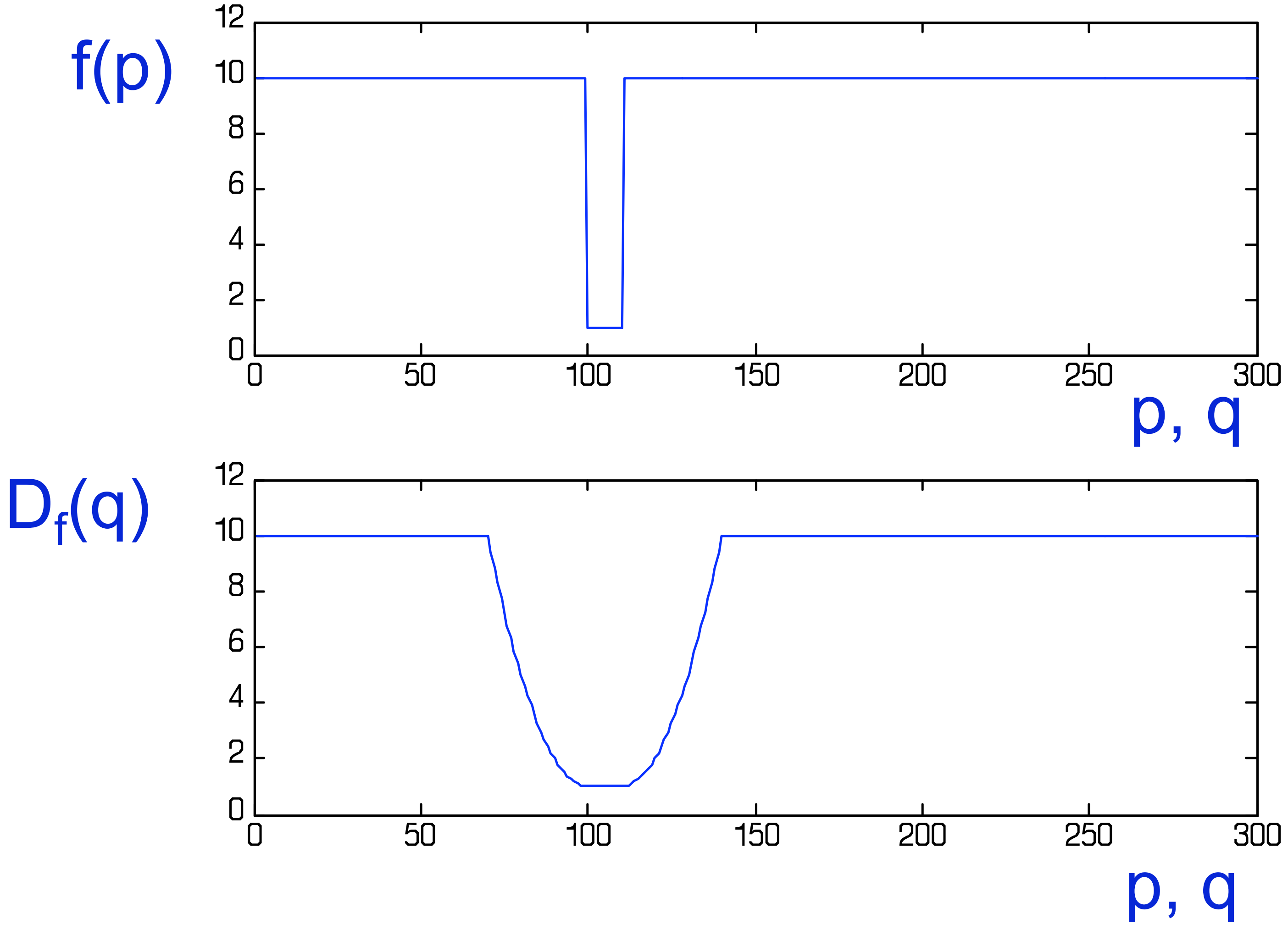
$$\mathcal{D}_f(q) = \min_{p \in \mathcal{G}} \left(||q - p||^2 + f(p) \right)$$

– for each location q, find nearby location p with $f(p)$ small.

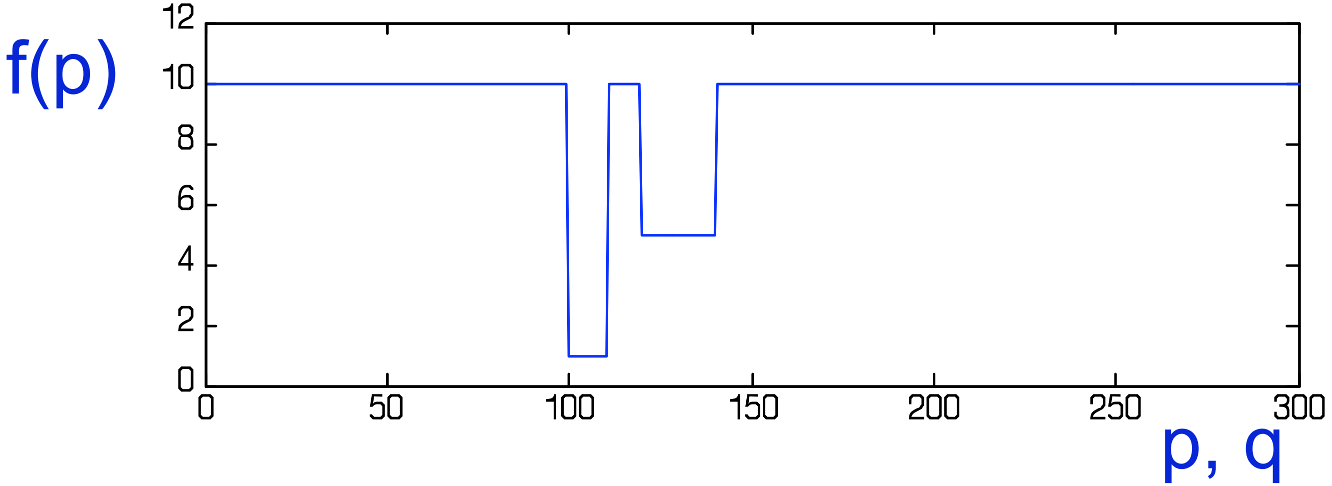
– equals DT of points P if f is an indicator function.

$$f(p) = \begin{cases} 0 & \text{if } p \in P \\ \infty & \text{otherwise} \end{cases}.$$

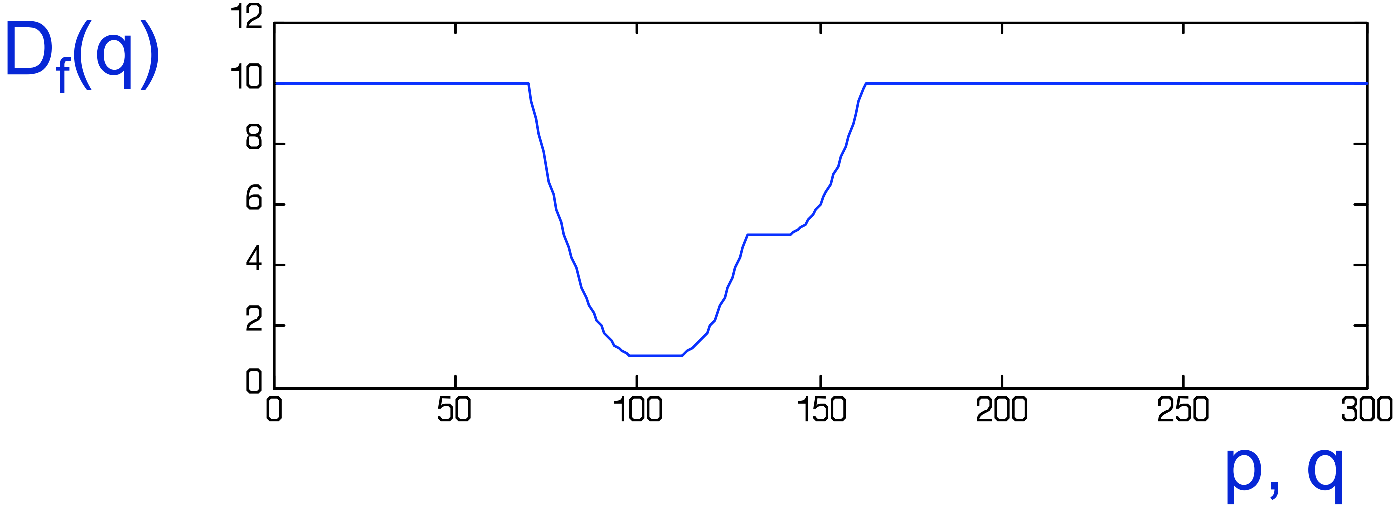
1D Examples


f(p)
12
10
8
6
4
2
0
0
50
100
150
200
250
300
p, q
$D_f(q)$
12
10
8
6
4
2
0
0
50
100
150
200
250
300
p, q

1D Examples



f(p)
12
10
8
6
4
2
0
0
50
100
150
200
250
300
p, q



$D_f(q)$
12
10
8
6
4
2
0
0
50
100
150
200
250
300
p, q

There is a simple geometric algorithm that computes $\mathcal{D}_f(p)$ in $O(h)$ time for the 1D case.

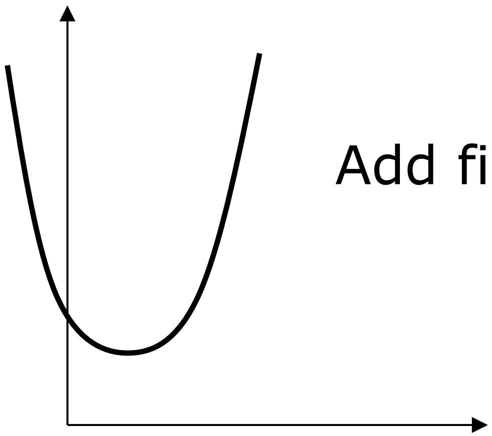
- similar to Graham's scan convex hull algorithm.
- about 20 lines of C code.

The 2D case is "separable", it can be solved by sequential 1D transformations along rows and columns of the grid.

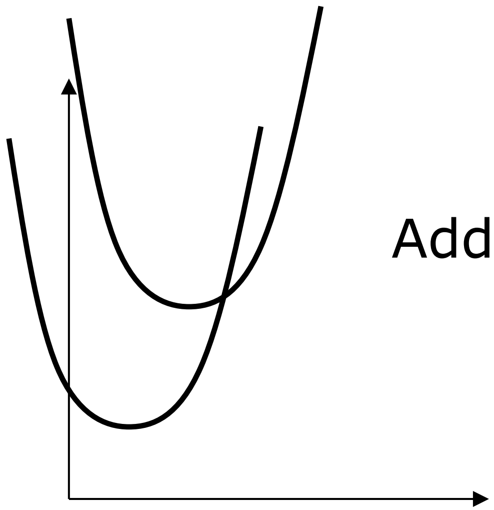
See **Distance Transforms of Sampled Functions**, Felzenszwalb and Huttenlocher.

Algorithm is non-examinable

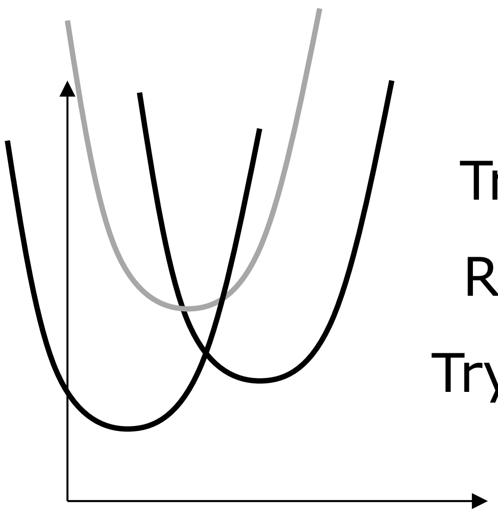
“Lower Envelope” Algorithm



Add first



Add second



Try adding third

Remove second

Try again and add

...

Algorithm for Lower Envelope

- Quadratics ordered left to right
- At step j consider adding j-th quadratic to LE of first j-1 quadratics

  - Maintain two ordered lists
    > Quadratics currently visible on LE
    > Intersections currently visible on LE

  - Compute intersection of j-th quadratic and rightmost quadratic visible on LE
    > If to right of rightmost visible intersection, add quadratic and intersection to lists
    > If not, this quadratic hides at least rightmost quadratic, remove it and try again

Code available online: http://people.cs.uchicago.edu/~pff/dt/

Running Time of LE Algorithm

Considers adding each of h quadratics just once

- Intersection and comparison constant time
- Adding to lists constant time
- Removing from lists constant time
  - > But then need to try again

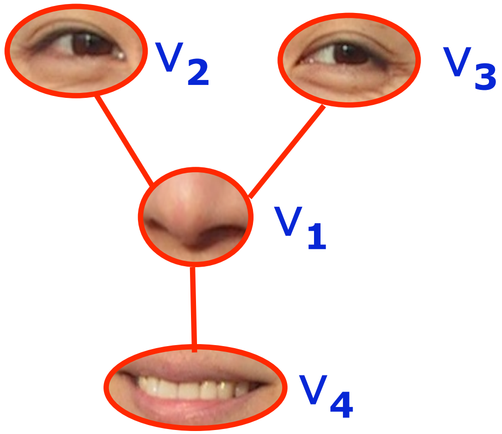
Simple amortized analysis

- Total number of removals O(h)
  - > Each quadratic once removed never considered for removal again

Thus overall running time O(h)

Example: facial feature detection in images

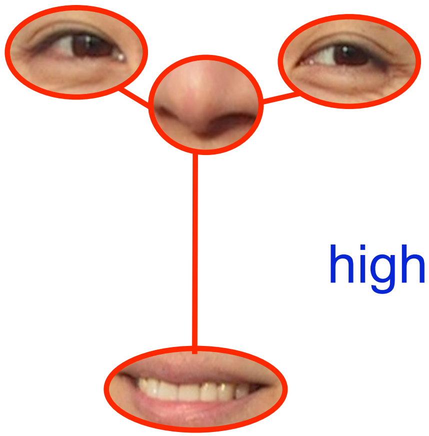
Model



- Parts V= $\{v_1, \ldots v_n\}$
- Connected by springs in star configuration to nose
- Quadratic cost for spring

$$E(\mathbf{x}) = \sum_{v_i \in V} m_i(v_i) + \sum_{e_{ij} \in E} d_{ij}(v_i, v_j)$$

$$= \sum_{v_i \in V} m_i(v_i) + \sum_{j} d_{1,j}(v_1, v_j)$$



high spring cost

1 - NCC with appearance template

Spring extension from v_1 to v_j

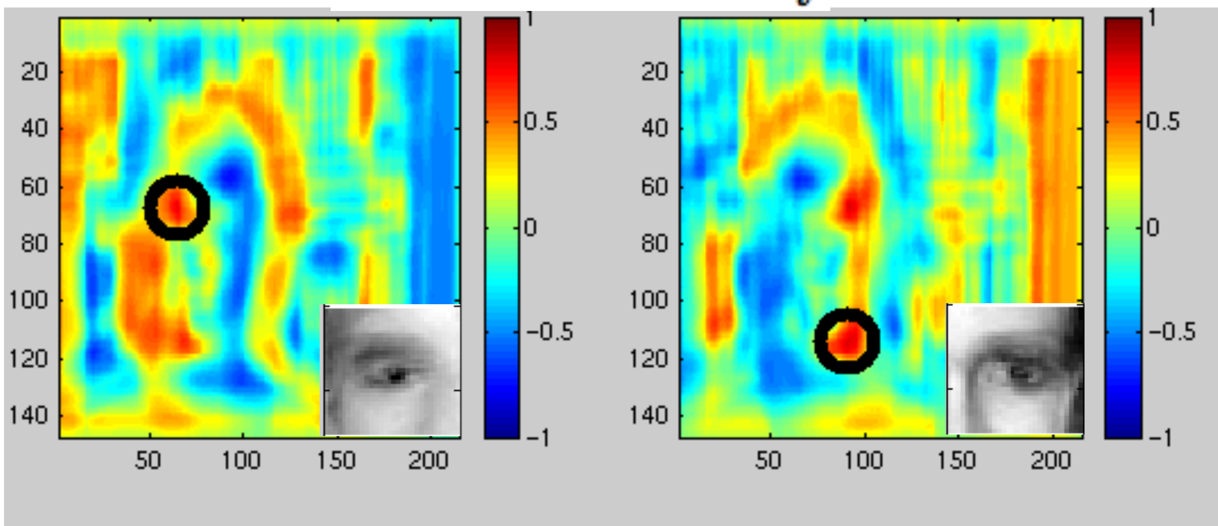
Appearance templates and springs

$$E(\mathbf{x}) \;=\; \sum_{v_i \in V} m_i(v_i) + \sum_j d_{1,j}(v_1, v_j)$$

$$\mathbf{x} = (x_1, y_1, \ldots, x_4, y_4)^\top$$

Each $l_i=(x_i, y_i)$ ranges over h (x,y) positions in the image

$$NCC = 1 - m_i$$



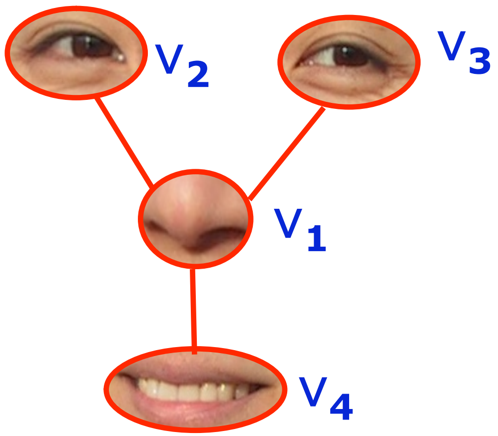
Requires pair wise terms for correct detection

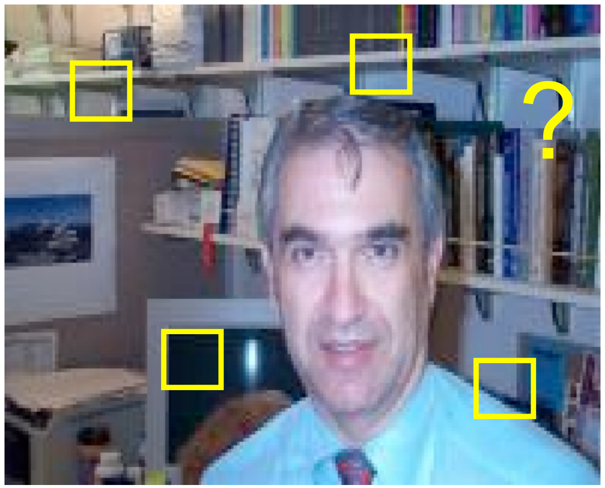
Fitting the model to an image

Find the configuration with the lowest energy

$$E(\mathbf{x}) = \sum_{v_i \in V} m_i(v_i) + \sum_j d_{1,j}(v_1, v_j)$$

Model



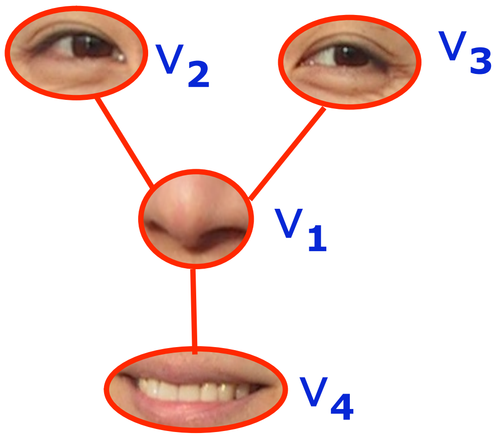


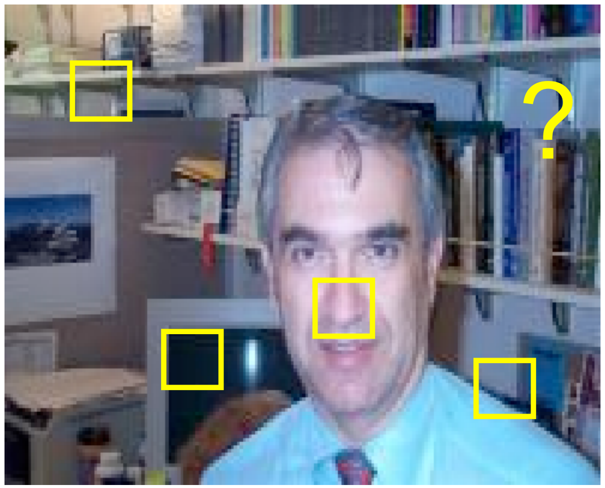
Fitting the model to an image

Find the configuration with the lowest energy

$$E(\mathbf{x}) = \sum_{v_i \in V} m_i(v_i) + \sum_j d_{1,j}(v_1, v_j)$$

Model



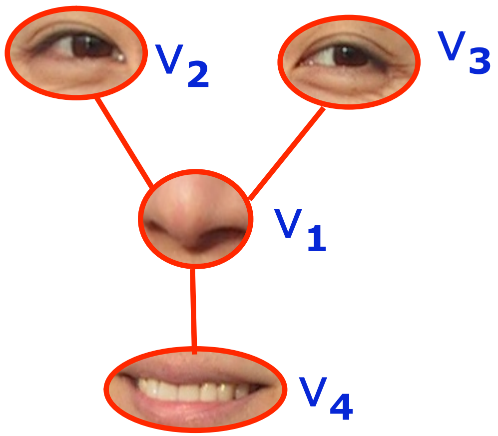


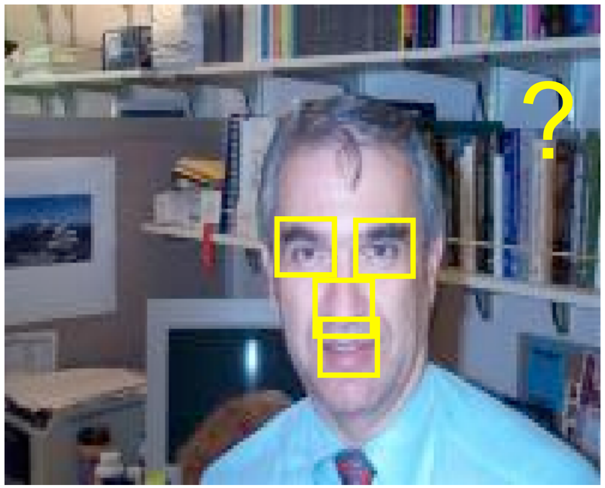
Fitting the model to an image

Find the configuration with the lowest energy

$$E(\mathbf{x}) = \sum_{v_i \in V} m_i(v_i) + \sum_j d_{1,j}(v_1, v_j)$$

Model



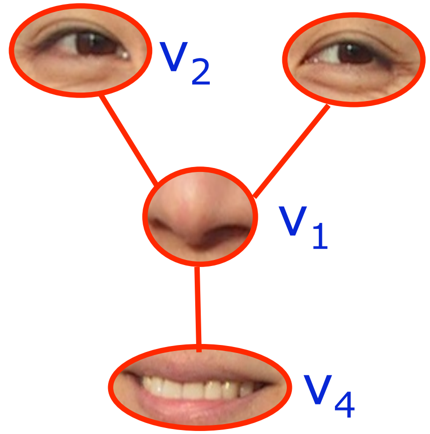


Notation

- Model is represented by a graph $G = (V, E)$.
  - $V = \{v_1, \ldots, v_n\}$ are the parts.
  - $(v_i, v_j) \in E$ indicates a connection between parts.
- $m_i(l_i)$ is the cost of placing part i at location l_i.
- $d_{ij}(l_i, l_j)$ is a deformation cost.
- Optimal location for object is given by $L^* = (l_1^*, \ldots, l_n^*)$,

$$L^* = \underset{L}{\operatorname{argmin}} \left(\sum_{i=1}^{n} m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j) \right)$$

Simple face model



- Locations are positions in the image grid.
- Match cost $m_i(l_i)$ for placing part i at l_i.
- Central part v_1 - the nose.
- Each part has an ideal position p_i relative to nose.
  - Let $T_{1i}(l_1) = l_1 + p_i$,

$$E(l_1, \ldots, l_n) = \sum_{i=1}^{n} m_i(l_i) + \sum_{i=2}^{n} ||l_i - T_{1i}(l_1)||^2$$

Efficient minimization

$$L^* = \underset{L}{\text{argmin}} \left(\sum_{i=1}^{n} m_i(l_i) + \sum_{i=2}^{n} ||l_i - T_{1i}(l_1)||^2 \right)$$

$$L^* = \underset{L}{\text{argmin}} \left(m_1(l_1) + \sum_{i=2}^{n} m_i(l_i) + ||l_i - T_{1i}(l_1)||^2 \right)$$

$$l_1^* = \underset{l_1}{\text{argmin}} \left(m_1(l_1) + \sum_{i=2}^{n} \min_{l_i}(m_i(l_i) + ||l_i - T_{1i}(l_1)||^2) \right)$$

$$l_1^* = \underset{l_1}{\text{argmin}} \left(m_1(l_1) + \sum_{i=2}^{n} \mathcal{D}_{m_i}(T_{1i}(l_1)) \right)$$

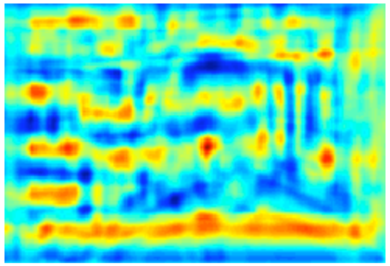
where $\mathcal{D}_f(q) = \min_{p \in \mathcal{G}} \left(||q - p||^2 + f(p) \right)$

Visualization: Compute part matching cost (dense)

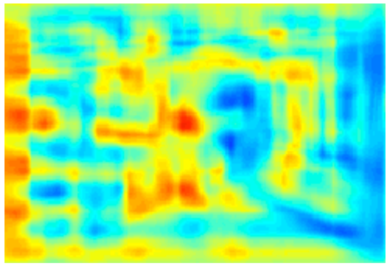
Input image

Compute matching cost $m_i(l_i)$ for each pixel

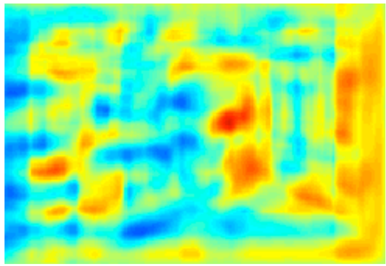
Nose



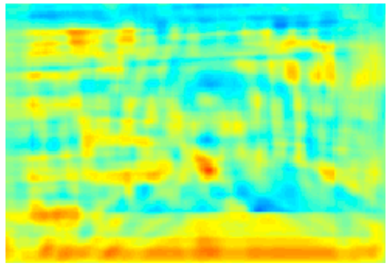
Left eye



Right eye

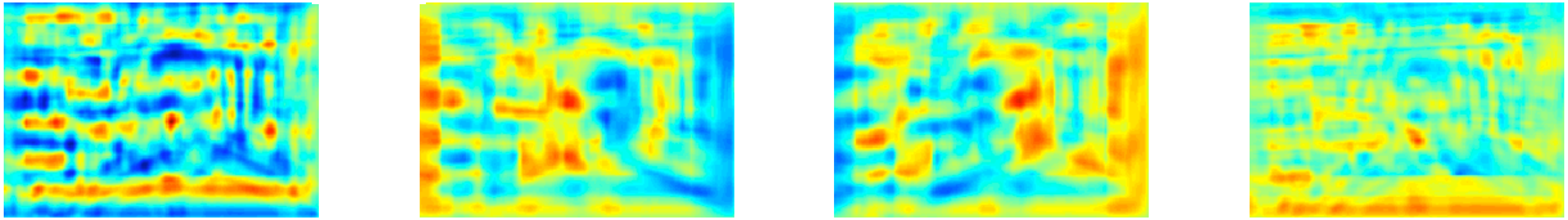


Mouth



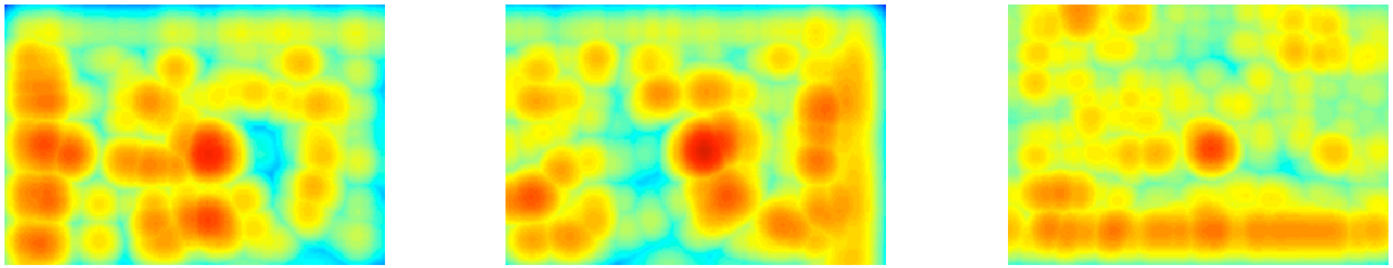
Visualization: Combine appearance with relative shape

Part matching cost $m_i(l_i)$

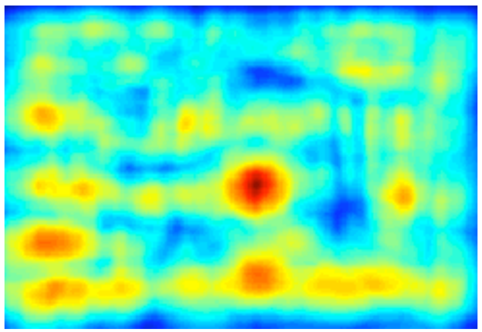


1. Nose 2. Left eye 3. Right eye 4. Mouth

(Shifted) distance transform of $m_i(l_i) = \mathcal{D}_{m_i}(T_{1i}(l_1))$



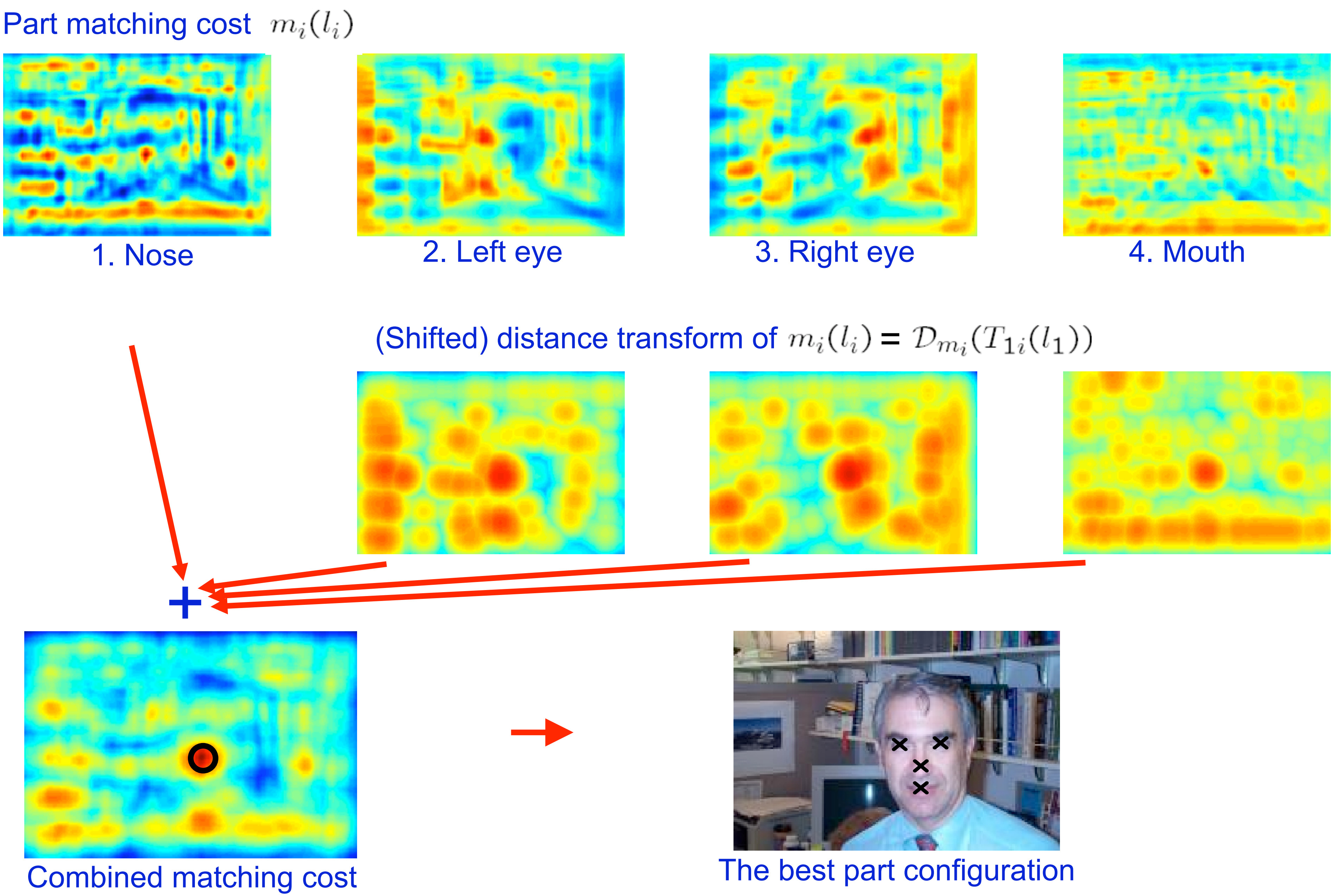
+



Combined matching cost

$$l_1^* = \underset{l_1}{\operatorname{argmin}} \left(m_1(l_1) + \sum_{i=2}^{n} \mathcal{D}_{m_i}(T_{1i}(l_1)) \right)$$

Visualization: Combine appearance with relative shape



Combine appearance with relative shape

The distance transform can be computed separately for rows and columns of the image (i.e. is “separable”), which results in the O(hn) running time

Given the best location of the reference location (root), locations of leafs can be found by “back-tracking” (here only one level).

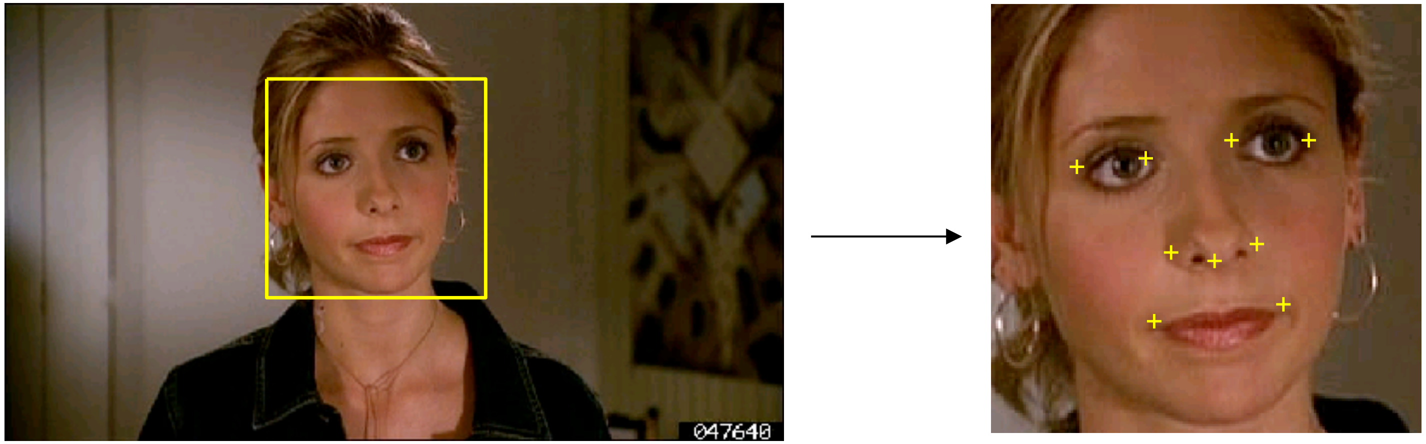
Simple part based face model demo code [Fei Fei, Fergus, Torralba]:
http://people.csail.mit.edu/torralba/shortCourseRLOC/

Example

Example of a model with 9 parts

The goal:

Localize facial features in faces output by face detector



Support parts-based face descriptors

Provide initialization for global face descriptors

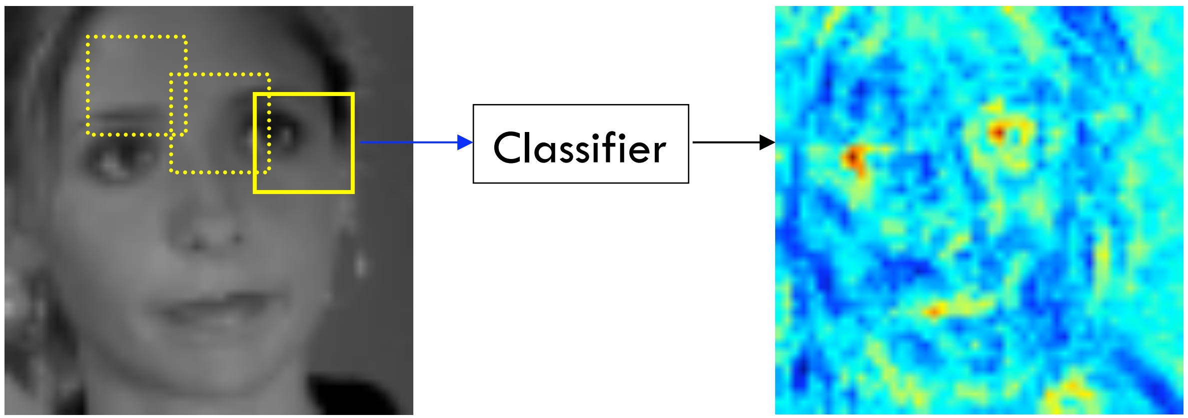
Code available online: http://www.robots.ox.ac.uk/~vgg/research/nface/index.html

Example of a model with 9 parts

Classifier for each facial feature

- Linear combination of thresholded simple image filters (Viola/Jones) trained discriminatively using AdaBoost
- Applied in "sliding window" fashion to patch around every pixel
- Similar to Viola&Jones face detector – see lecture 6

Ambiguity e.g. due to facial symmetry

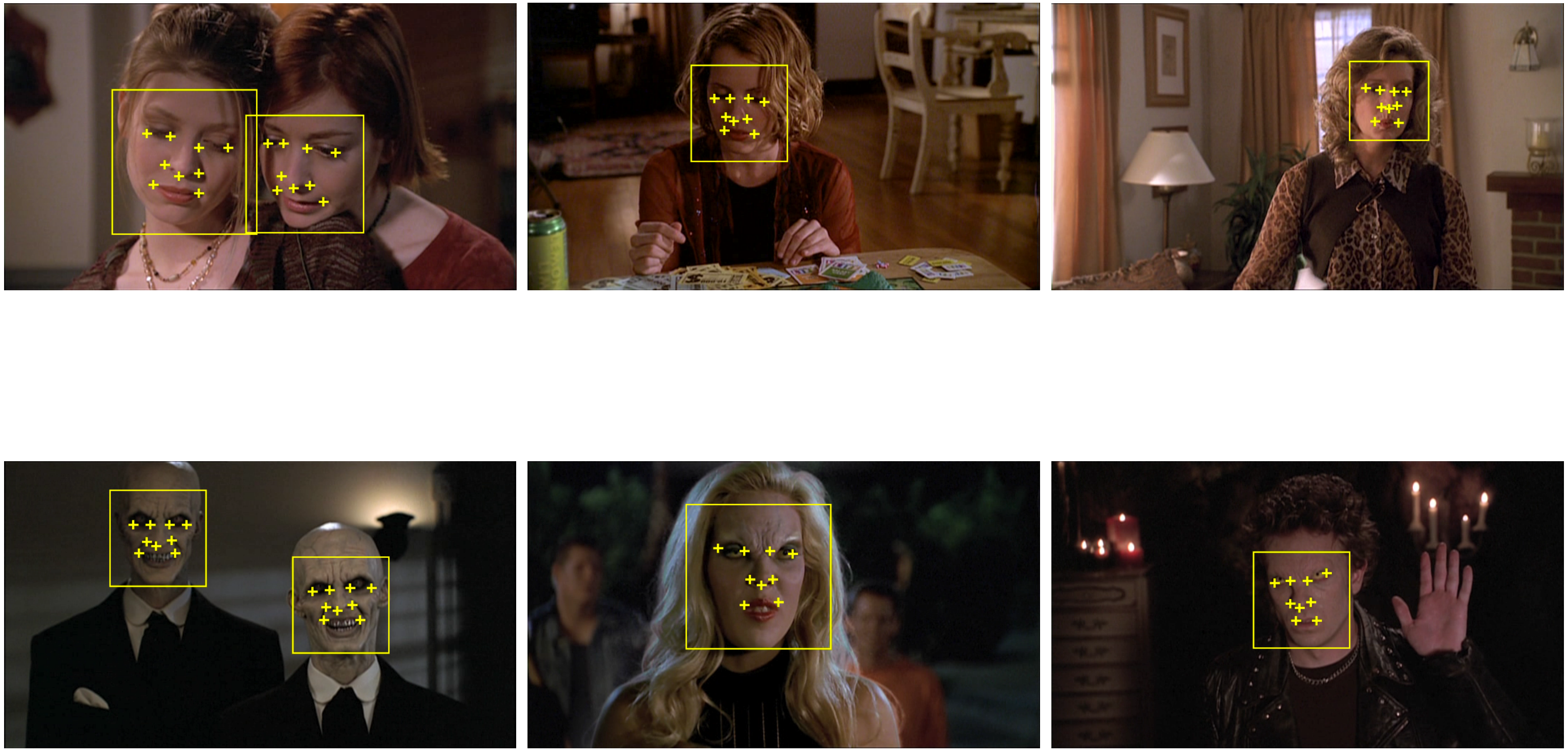


Resolve ambiguity using spatial model.

Results

Nine facial features, ~90% predicted positions within 2 pixels in 100×100 face image

Results



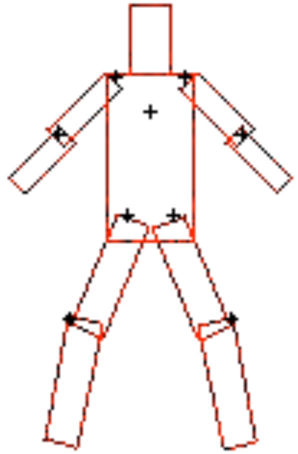
Example II: Generic Person Model

Each part represented as rectangle

- Fixed width, varying length, uniform colour
- Learn average and variation
  > Connections approximate revolute joints
- Joint location, relative part position, orientation, foreshortening - Gaussian
- Estimate average and variation

Learned 10 part model

- All parameters learned
  > Including “joint locations”
- Shown at ideal configuration (mean locations)



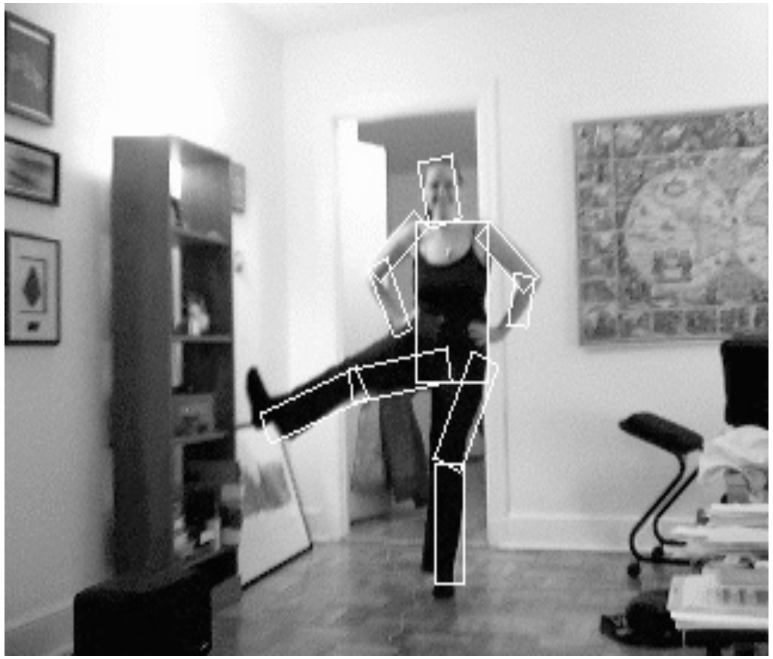
Learning

Manual identification of

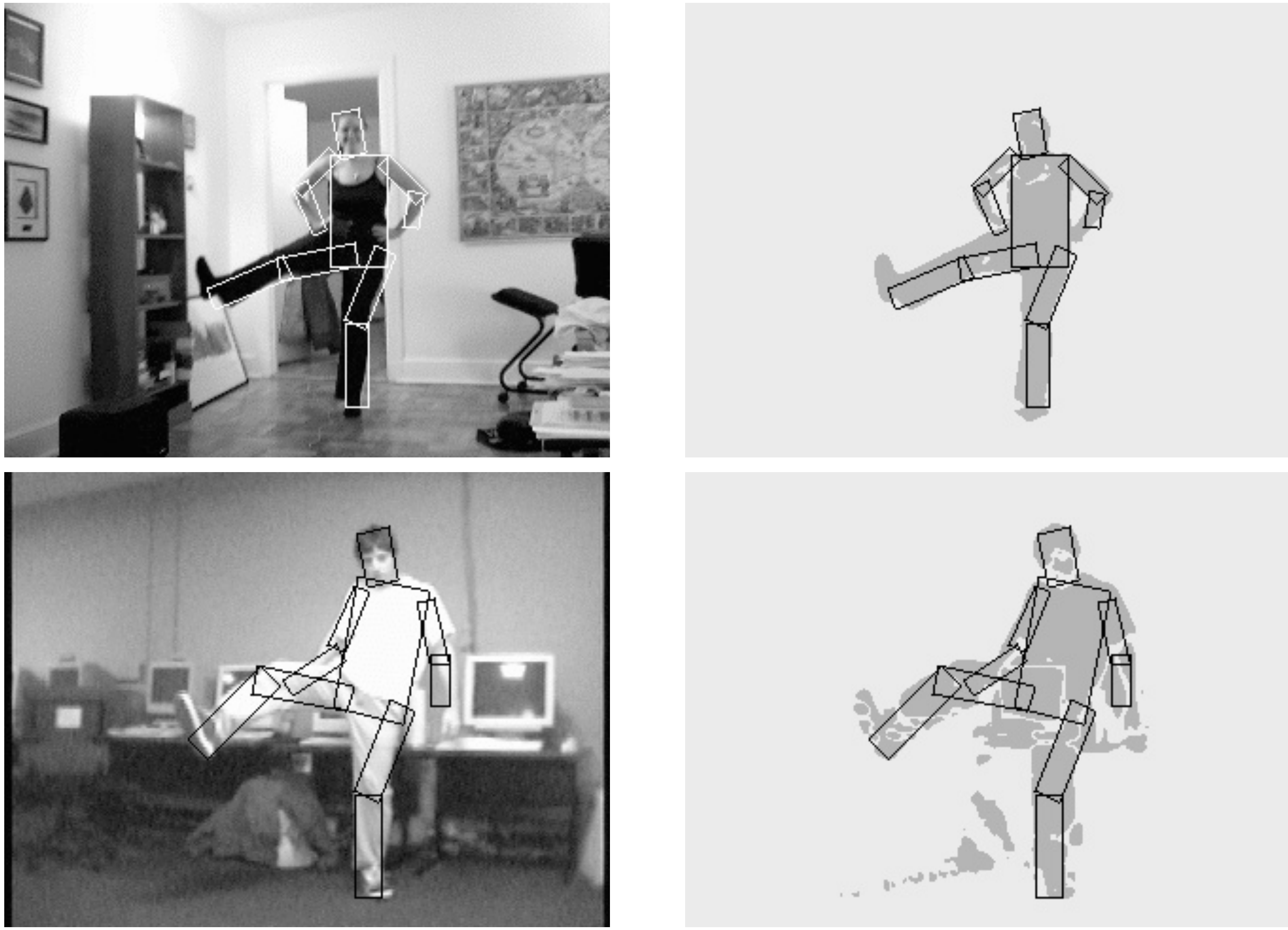
- rectangular parts in a set of
- training images hypotheses

Learn

- relative position (x & y),
- relative angle,
- relative foreshortening

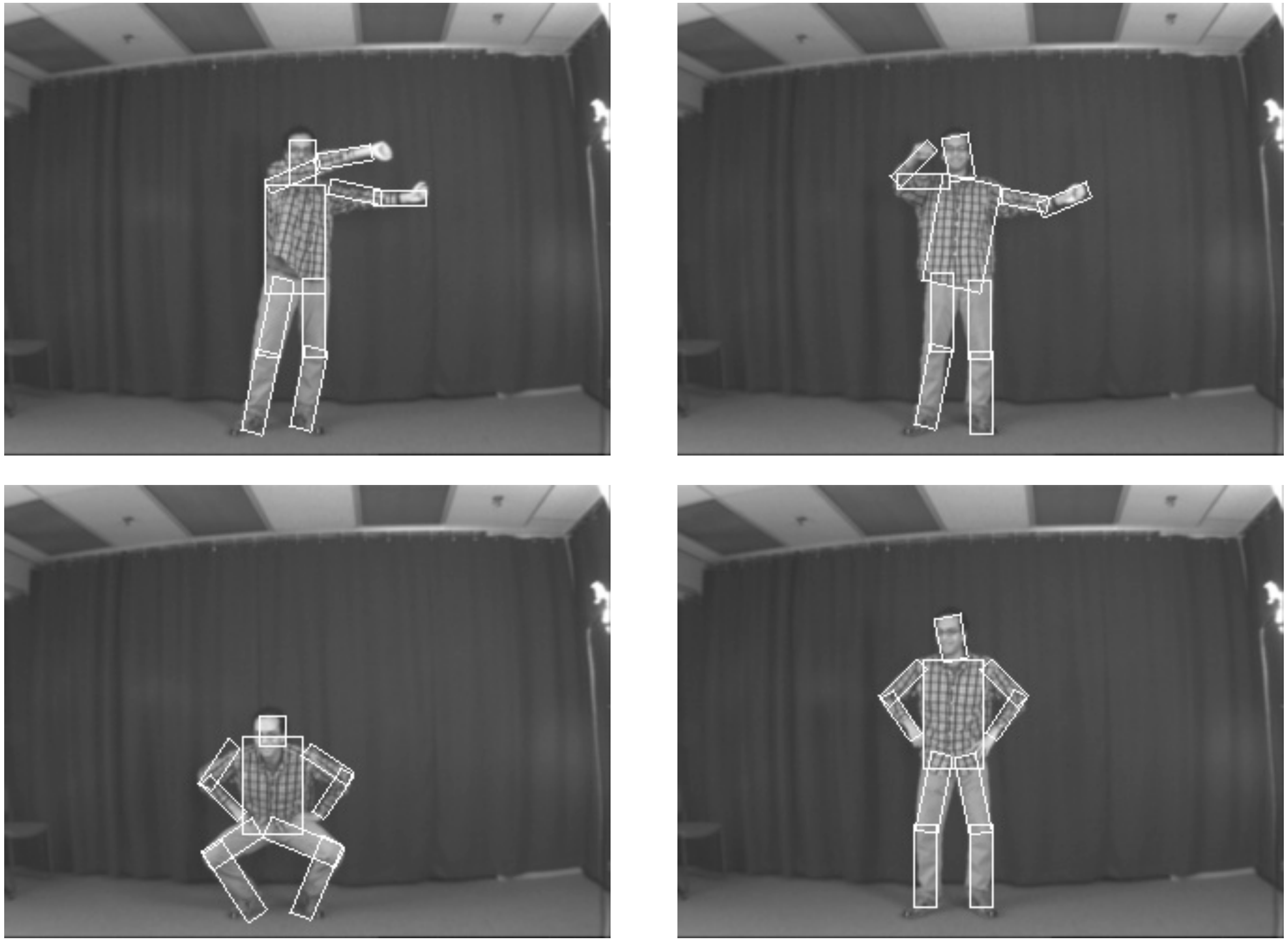


Example: Recognizing People

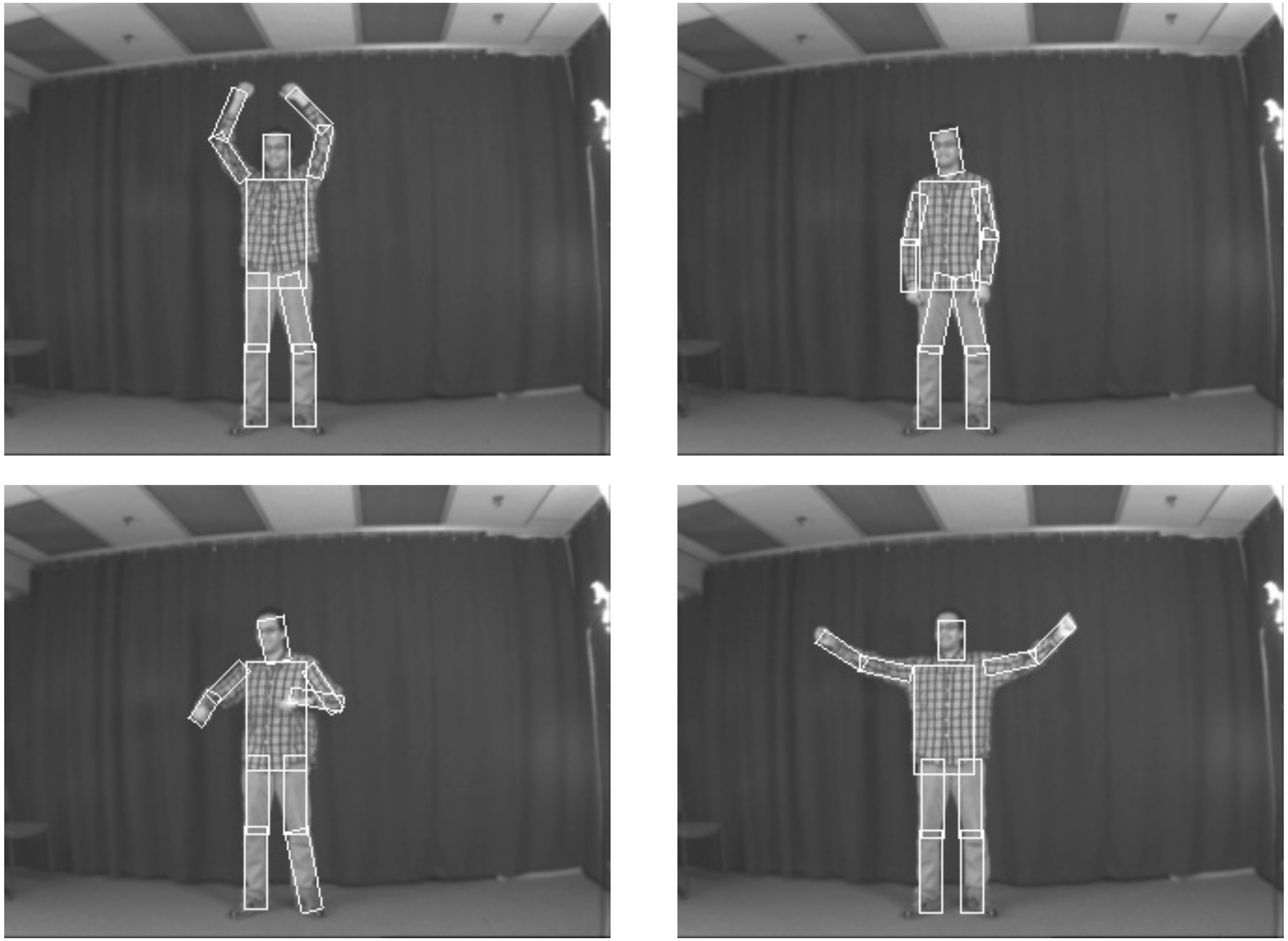


NB: requires background subtraction

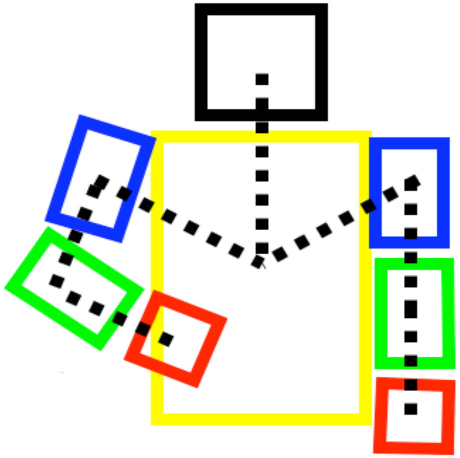
Variety of Poses

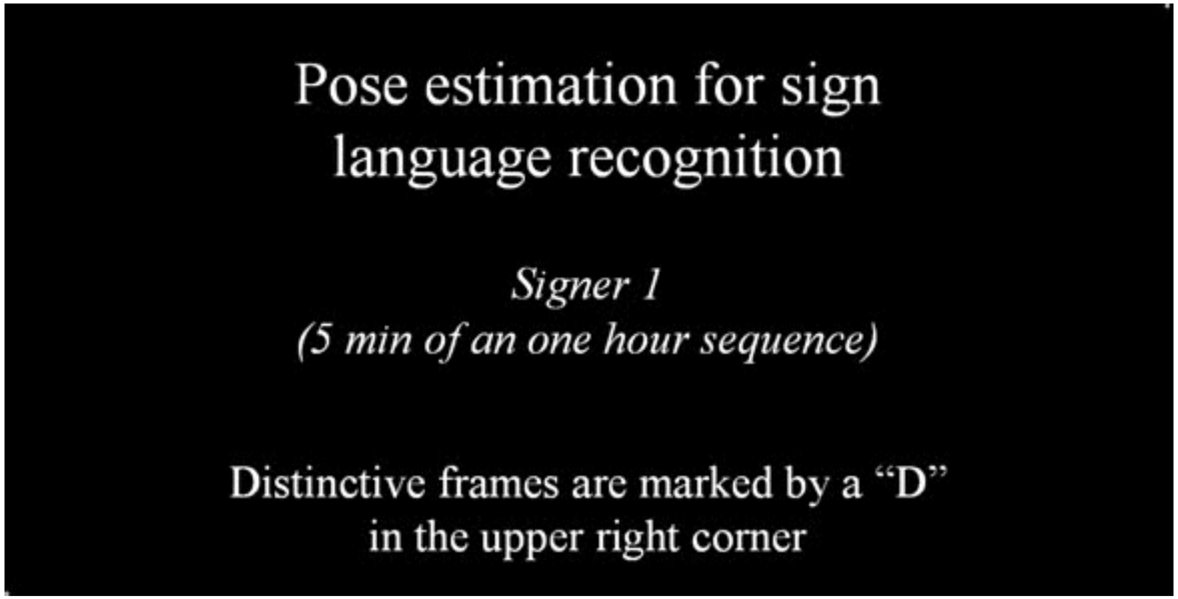


Variety of Poses



Example III: Hand tracking for sign language interpretation



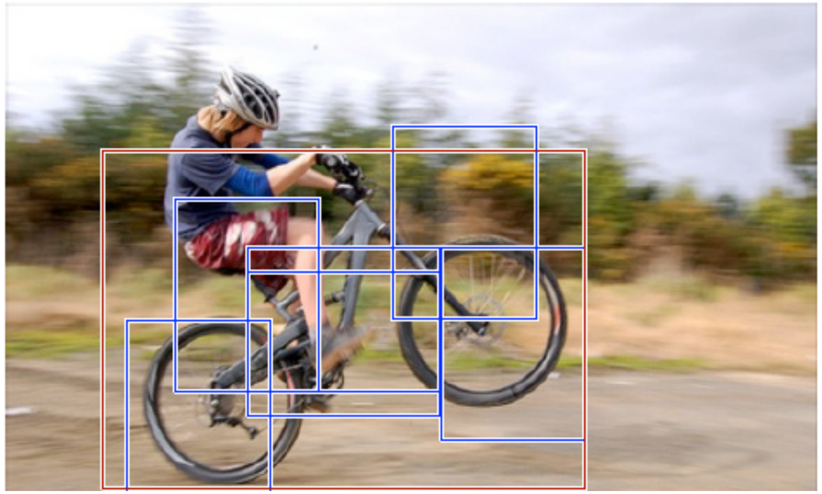


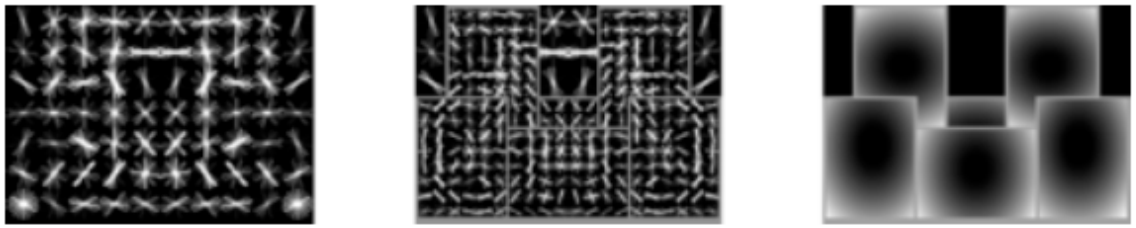
Buehler et al. BMVC'2008

Example results

Example IV: Part based models for object detection (Recall from Lecture 9)

[Felsenszwalb et al. 2009]

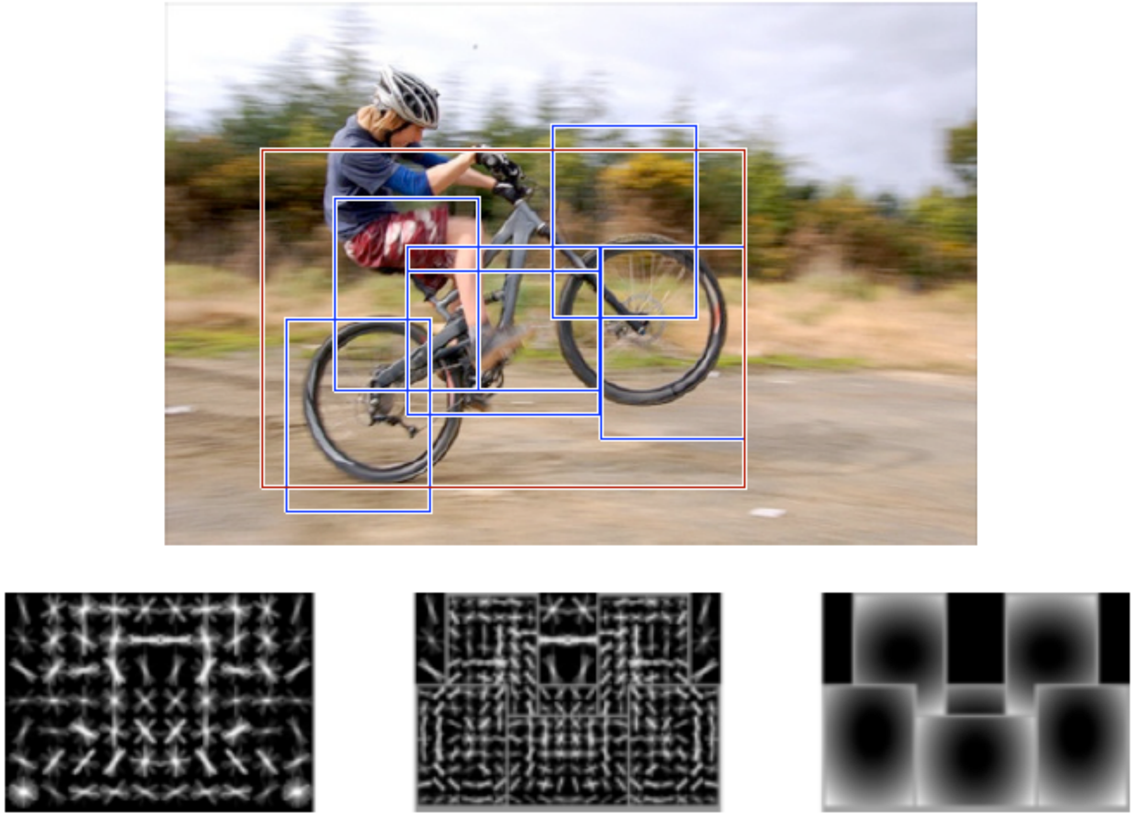




- Each component has global template + deformable parts
- Fully trained from bounding boxes alone

Code available online: http://people.cs.uchicago.edu/~pff/latent/

Bicycle model



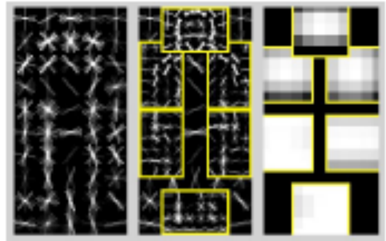
root filters
coarse resolution

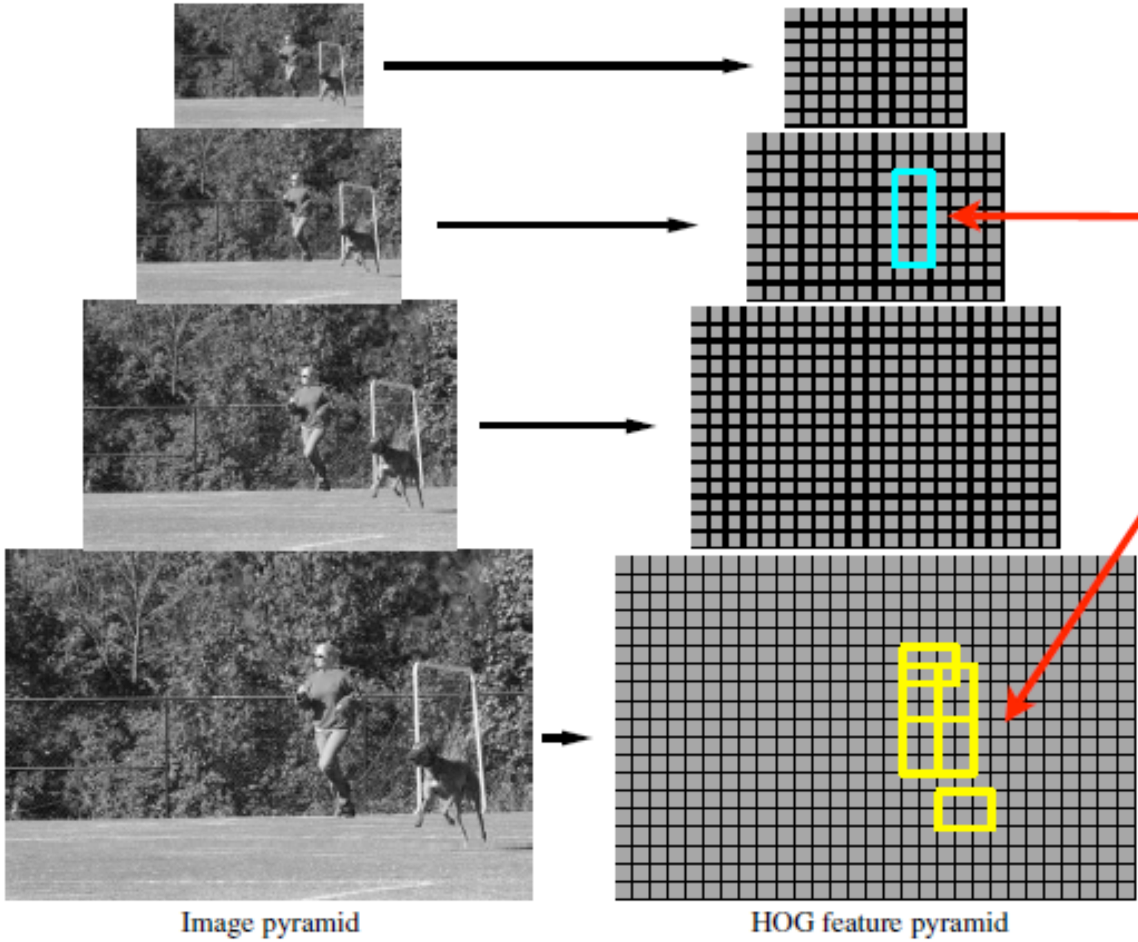
part filters
finer resolution

deformation
models

Each component has a root filter F_0
and n part models (F_i, v_i, d_i)

Object hypothesis





$z = (p_0, \ldots, p_n)$

p_0 : location of root

$p_1, \ldots, p_n$: location of parts

Score is sum of filter scores minus deformation costs

Multiscale model captures features at two-resolutions

Score of a hypothesis

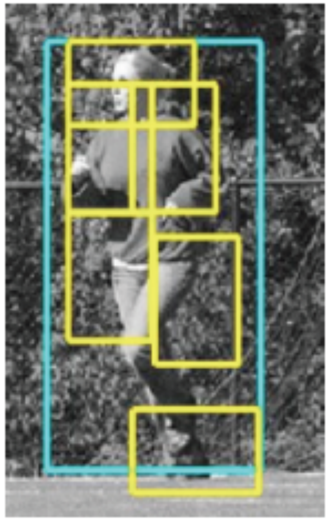
"data term" "spatial prior"

$$\text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} F_i \cdot \phi(H, p_i) - \sum_{i=1}^{n} d_i \cdot (dx_i^2, dy_i^2)$$

filters

deformation parameters

displacements



$$\text{score}(z) = \beta \cdot \Psi(H, z)$$

concatenation filters and deformation parameters

concatenation of HOG features and part displacement features

Matching

- Define an overall score for each root location
  - Based on best placement of parts

$$\text{score}(p_0) = \max_{p_1,\ldots,p_n} \text{score}(p_0, \ldots, p_n).$$

- High scoring root locations define detections
  - "sliding window approach"