Pictorial Structures and Distance Transforms

Computer Vision

CS 543 / ECE 549

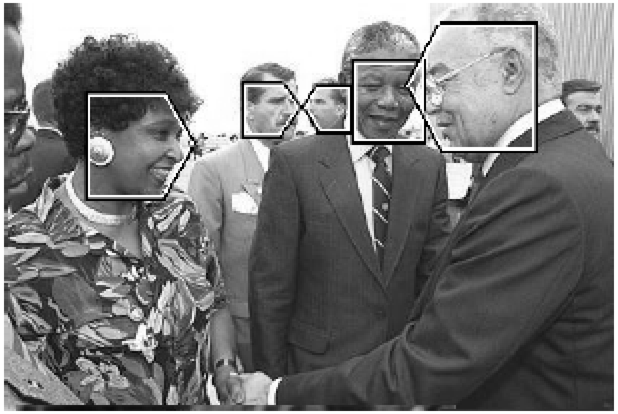
University of Illinois

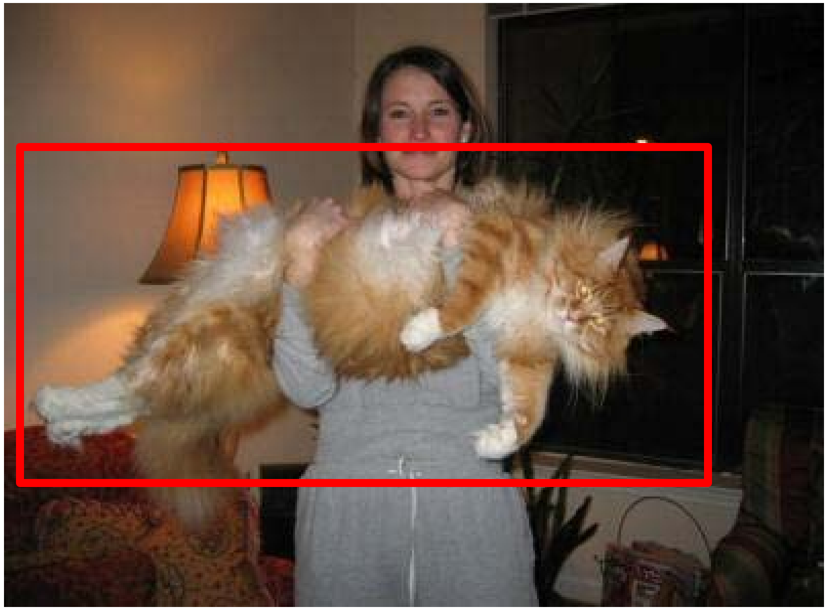
Ian Endres

Goal: Detect all instances of objects

Cars

Faces



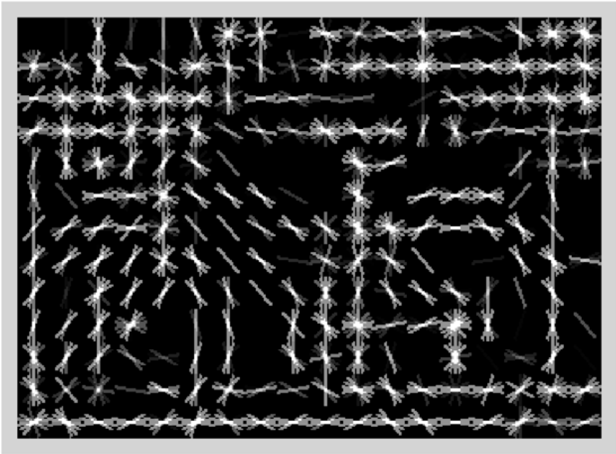


Cats

Object model: last class

- Statistical Template in Bounding Box
  - Object is some (x,y,w,h) in image
  - Features defined wrt bounding box coordinates

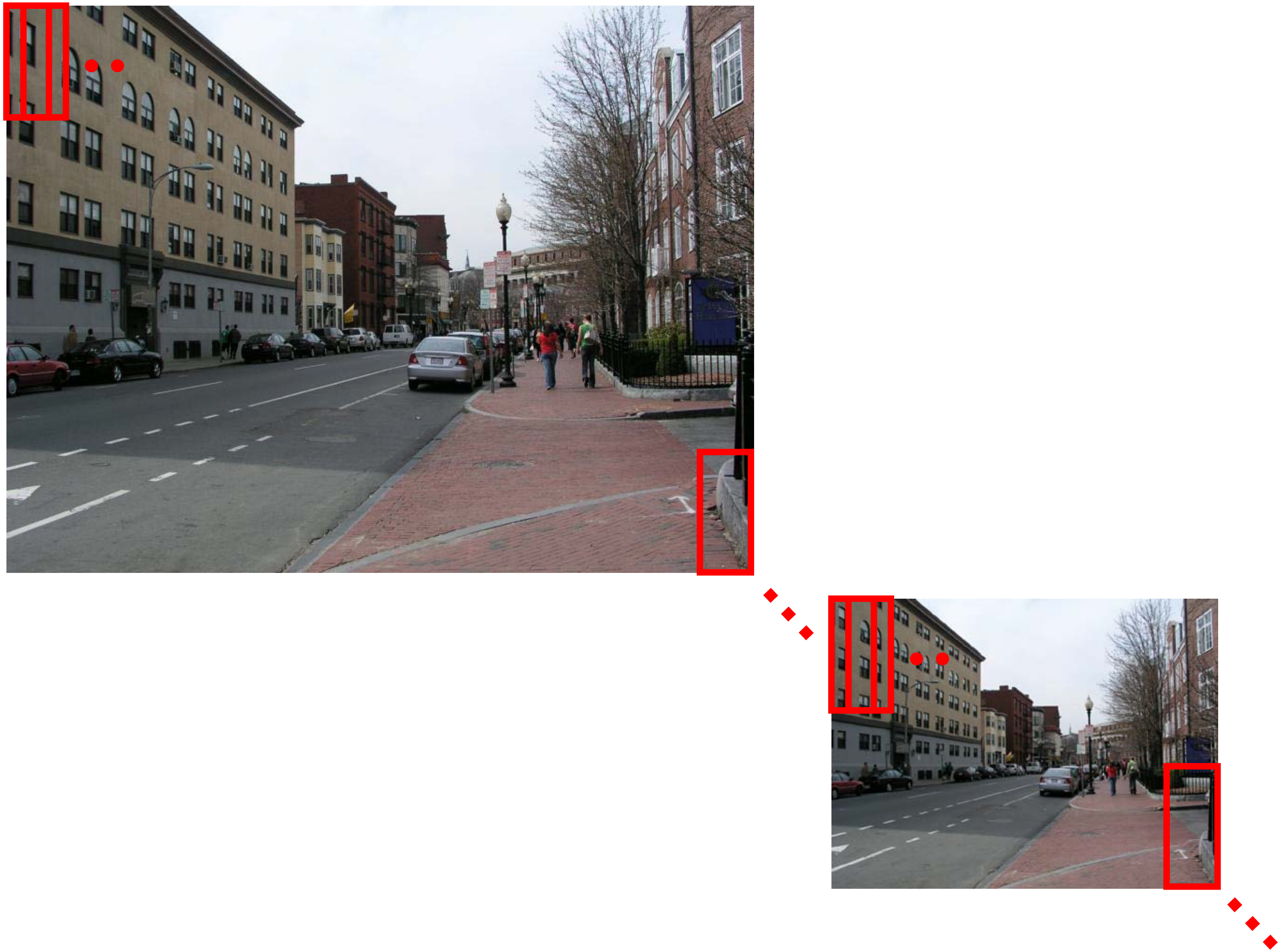
Image



Template Visualization

Images from Felzenszwalb

Last class: sliding window detection

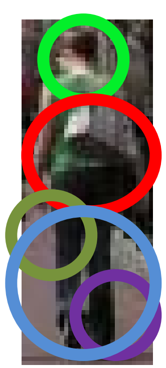


Last class: statistical template

- Object model = log linear model of parts at fixed positions

$+3 +2 -2 -1 -2.5 = -0.5 \overset{?}{>} 7.5$

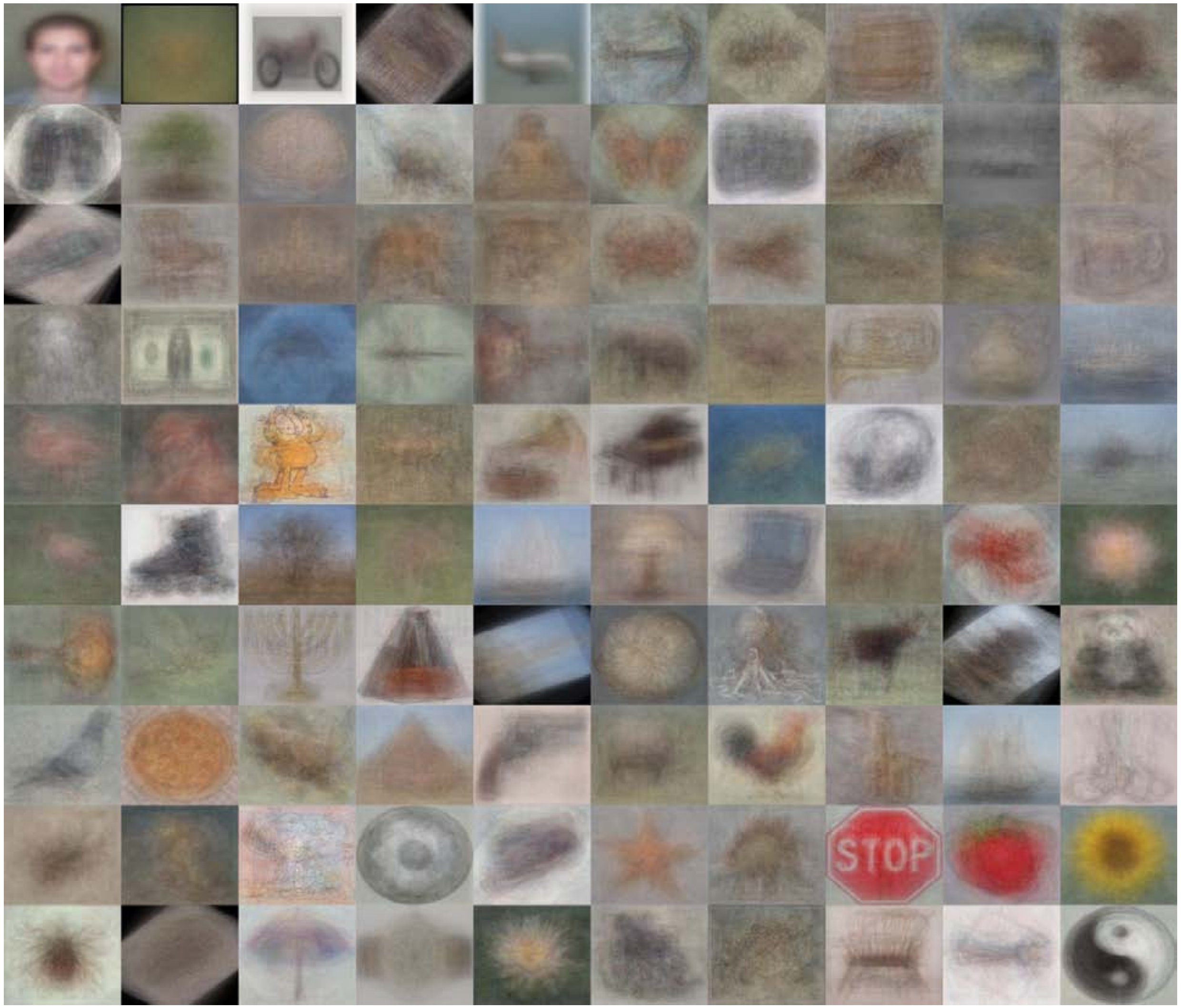
Non-object



$+4 +1 +0.5 +3 +0.5 = 10.5 \overset{?}{>} 7.5$

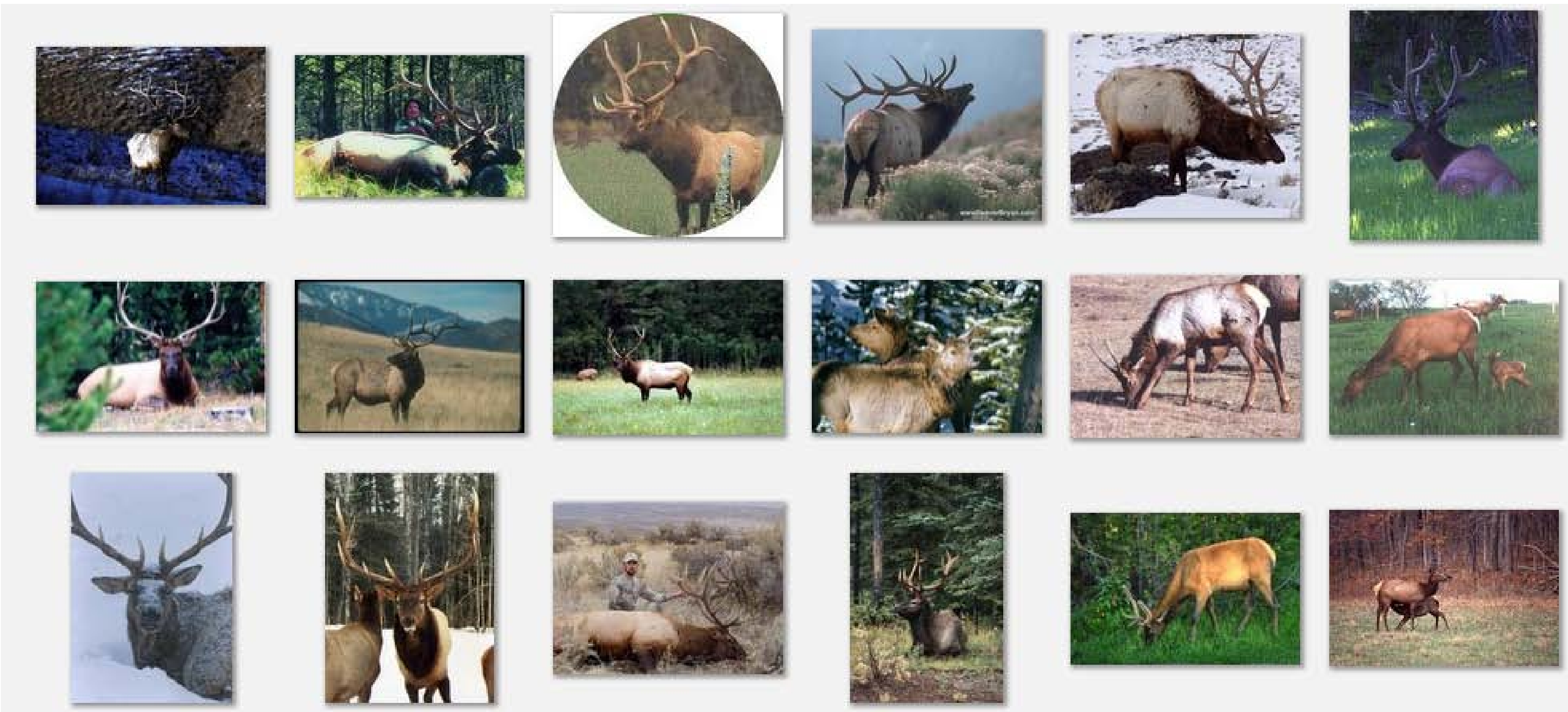
Object

When are statistical templates useful?



Caltech 101 Average Object Images

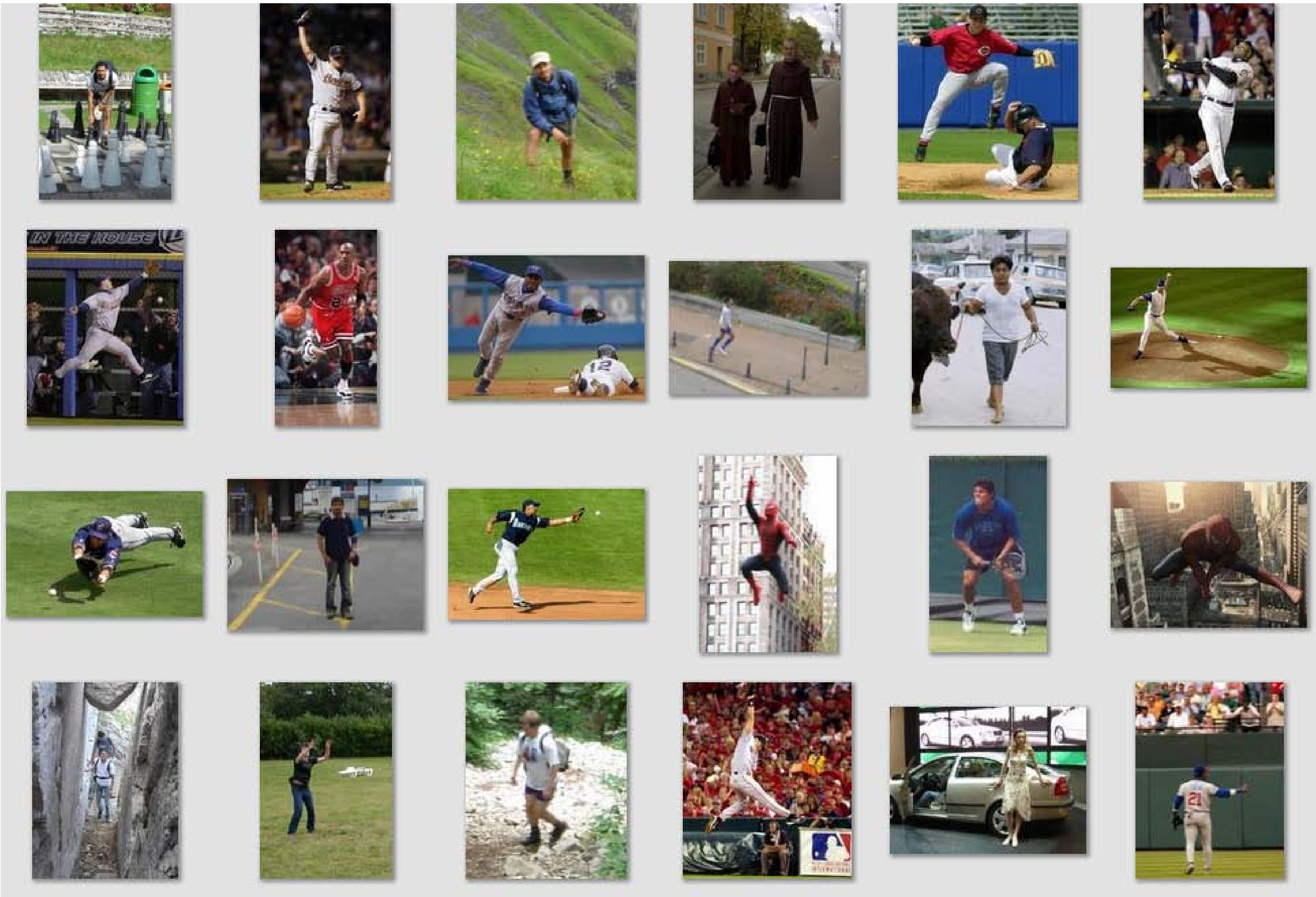
Deformable objects



Images from Caltech-256

Slide Credit: Duan Tran

Deformable objects

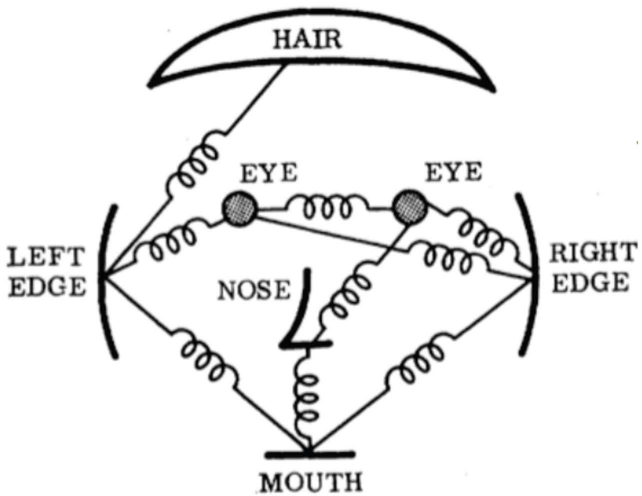


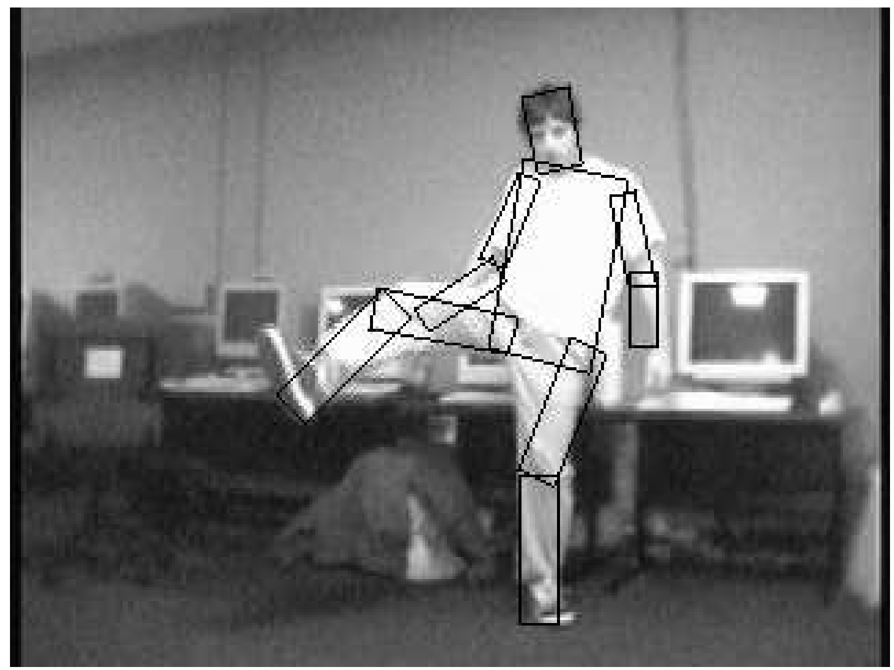
Images from D. Ramanan's dataset

Slide Credit: Duan Tran

Object models: this class

- Articulated parts model
  - Object is configuration of parts
  - Each part is detectable





Images from Felzenszwalb

Parts-based Models

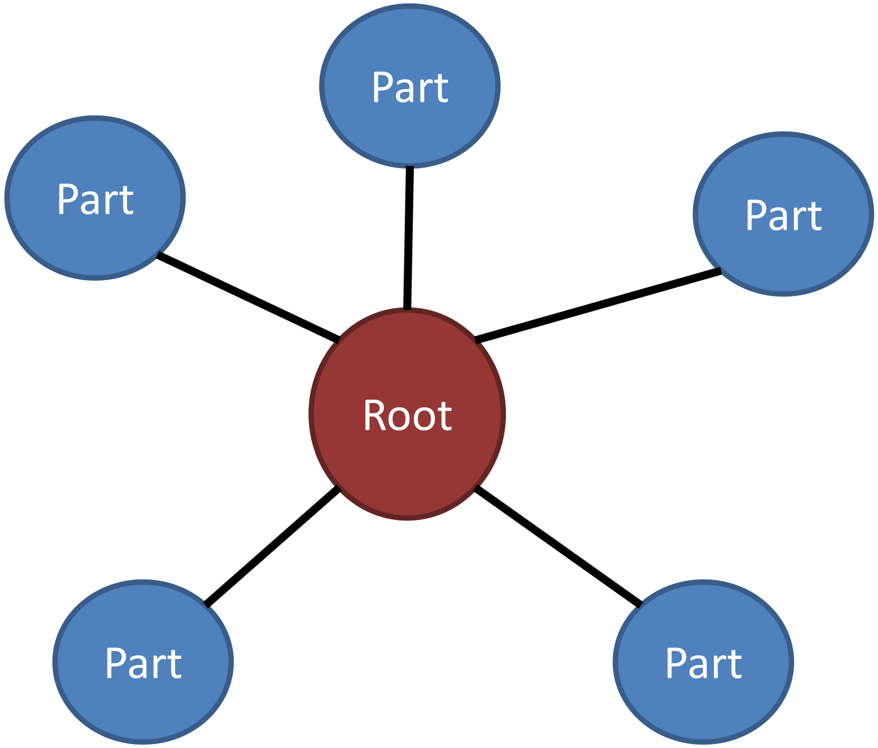
Define object by collection of parts modeled by

1. Appearance
2. Spatial configuration

Slide credit: Rob Fergus

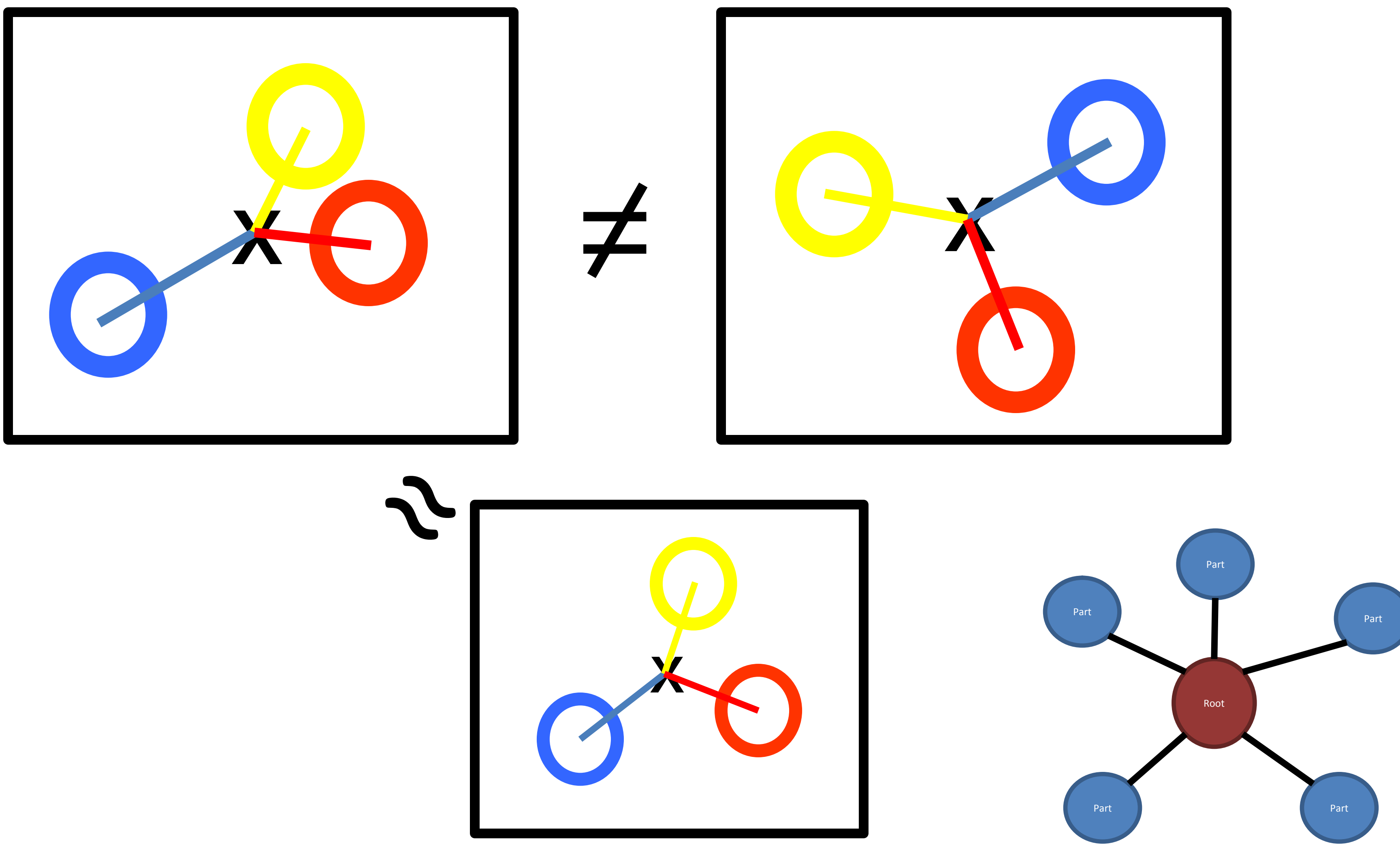
How to model spatial relations?

- Star-shaped model



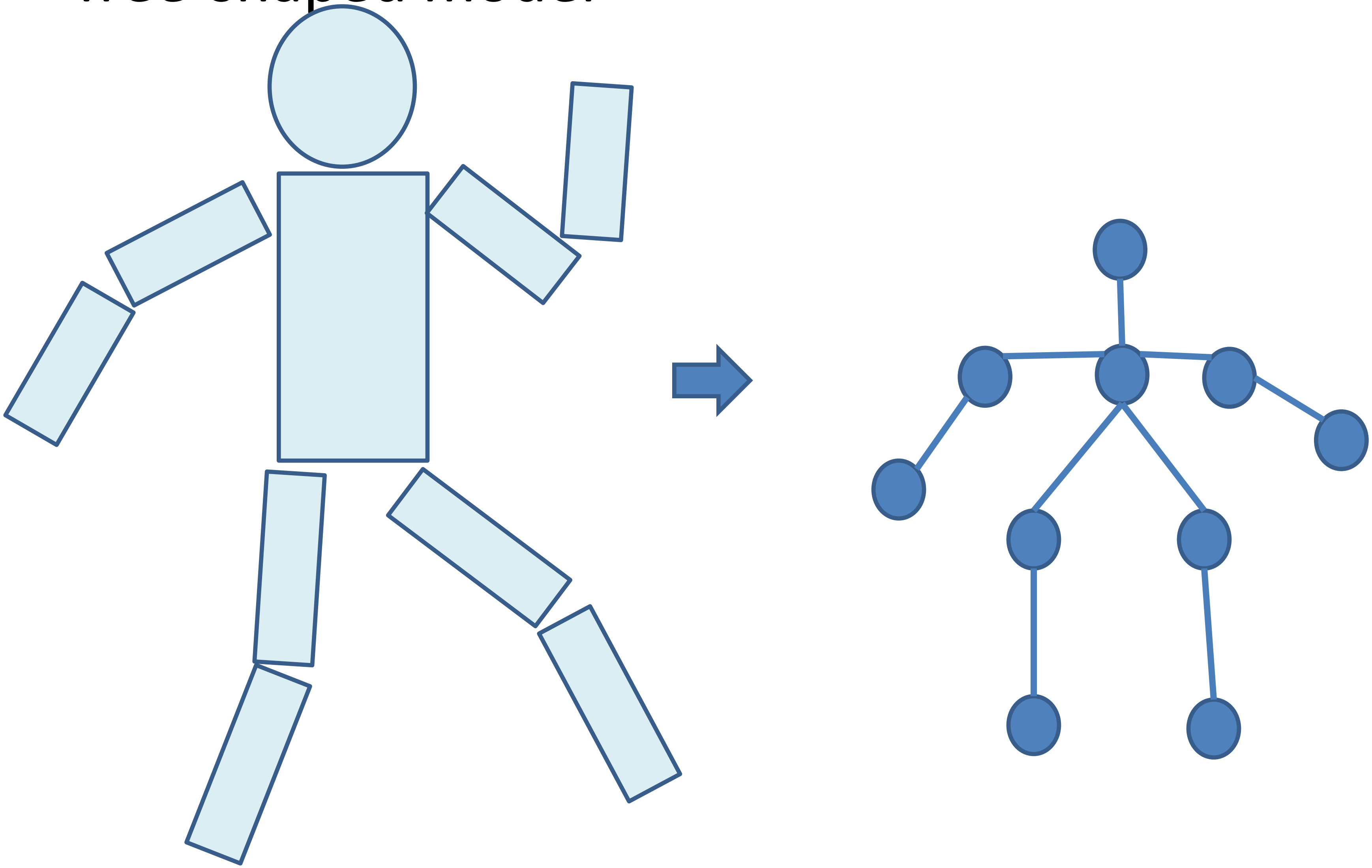
How to model spatial relations?

- Star-shaped model



How to model spatial relations?

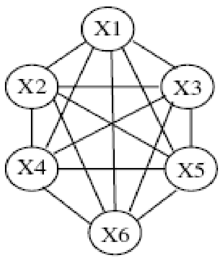
- Tree-shaped model



How to model spatial relations?

- Many others...

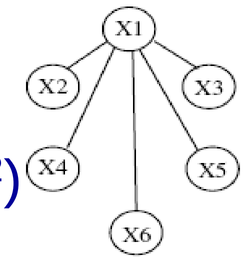
O(N^6)



a) Constellation

Fergus et al. ’03
Fei-Fei et al. ‘03

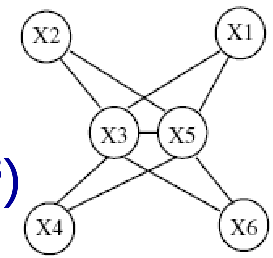
O(N^2)



b) Star shape

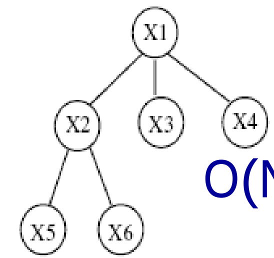
Leibe et al. ’04, ‘08
Crandall et al. ‘05
Fergus et al. ’05

O(N^3)



c) k-fan ($k = 2$)

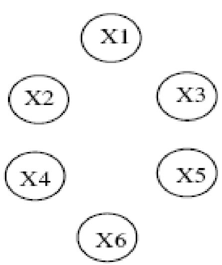
Crandall et al. ‘05



O(N^2)

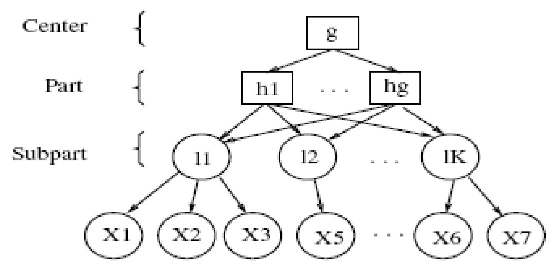
d) Tree

Felzenszwalb &
Huttenlocher ‘05



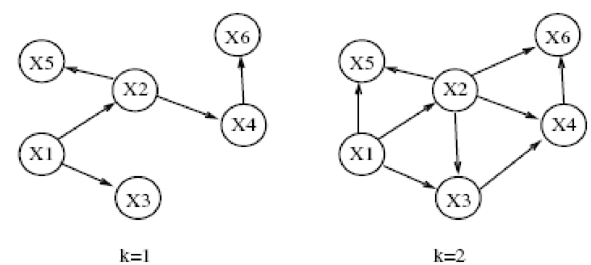
e) Bag of features

Csurka ’04
Vasconcelos ‘00



f) Hierarchy

Bouchard & Triggs ‘05



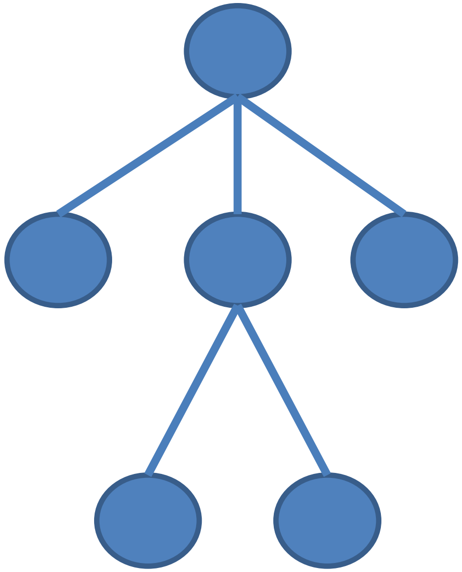
g) Sparse flexible model

Carneiro & Lowe ‘06

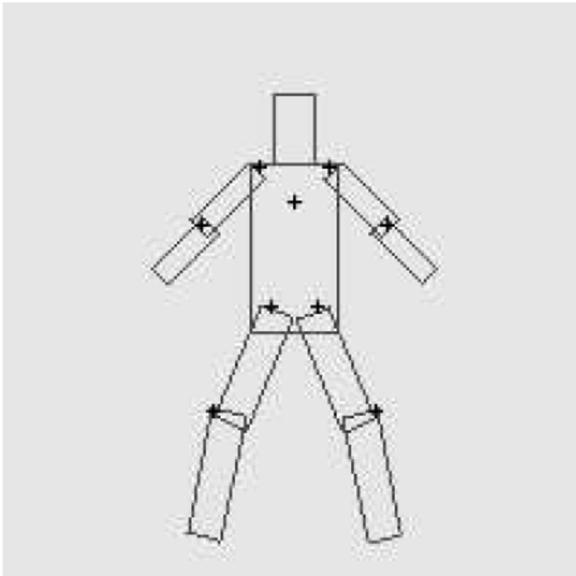
from [Carneiro & Lowe, ECCV’06]

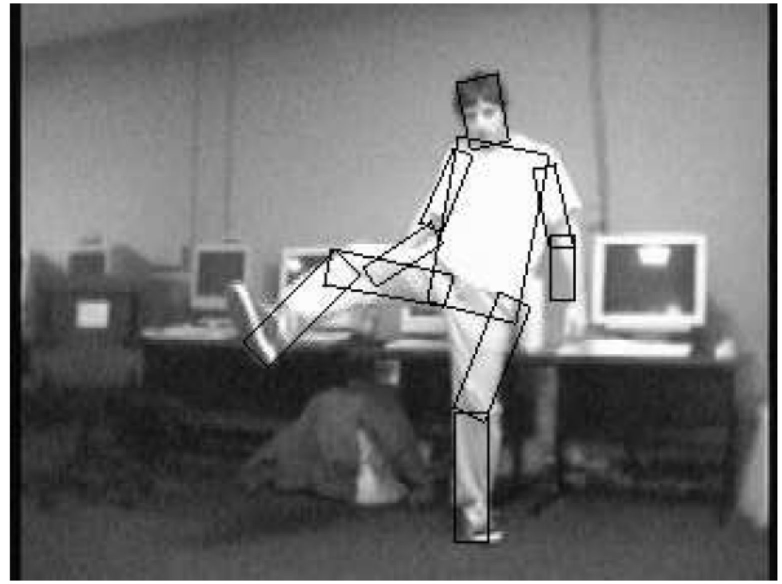
Today’s class

1. Tree-shaped model
   - Example: Pictorial structures
     - [Felzenszwalb Huttenlocher 2005]
2. Optimization with Dynamic Programming
3. Distance Transforms



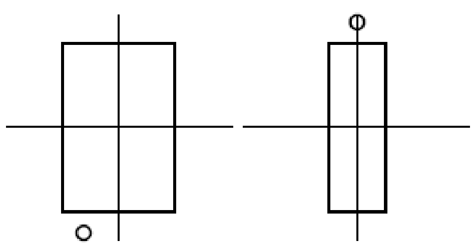
Pictorial Structures Model



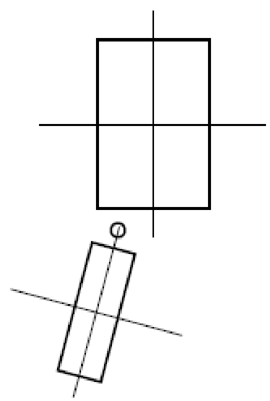


Part = oriented rectangle

Spatial model = relative size/orientation



a

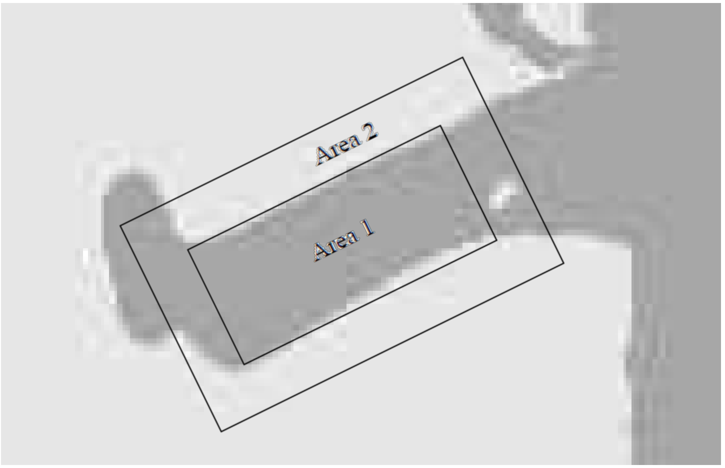


b

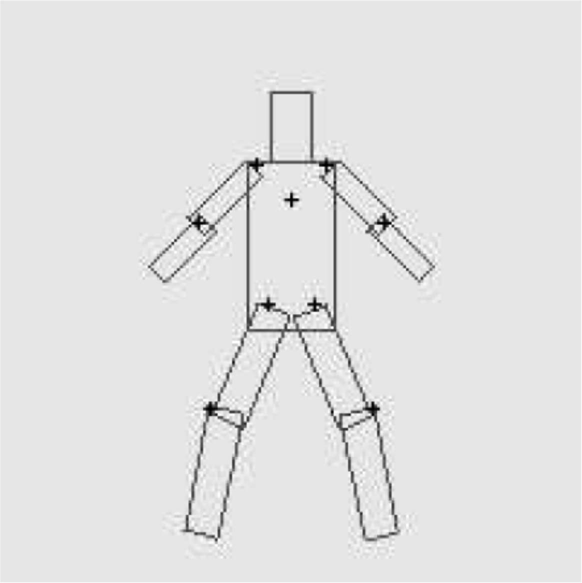
Felzenszwalb and Huttenlocher 2005

Part representation

- Background subtraction



Pictorial Structures Model



$$P(L|I,\theta) \propto \left(\prod_{i=1}^{n} p(I|l_i,u_i) \prod_{(v_i,v_j)\in E} p(l_i,l_j|c_{ij}) \right)$$

Appearance likelihood

Geometry likelihood

Modeling the Appearance

- Any appearance model could be used
  - HOG Templates, etc.
  - Here: rectangles fit to background subtracted binary map
- Train a detector for each part independently

$$P(L|I,\theta) \propto \left(\prod_{i=1}^{n} p(I|l_i, u_i) \prod_{(v_i,v_j)\in E} p(l_i, l_j|c_{ij}) \right)$$

Appearance likelihood

Geometry likelihood

Pictorial Structures Model

$$P(L|I,\theta) \propto \left(\prod_{i=1}^{n} p(I|l_i, u_i) \prod_{(v_i,v_j)\in E} p(l_i, l_j|c_{ij}) \right)$$

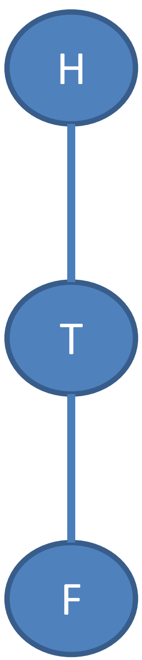
Minimize Energy (-log of the likelihood):

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

Optimization – Brute Force

$$E = \min_{l_H \in [1,N]} \min_{l_T \in [1,N]} \min_{l_F \in [1,N]}$$

$$\underbrace{m_H(l_H) + m_T(l_T) + m_F(l_F)}_{\text{Appearance Cost}} + \underbrace{d_{H,T}(l_H, l_T) + d_{T,F}(l_T, l_F)}_{\text{Deformation Cost}}$$

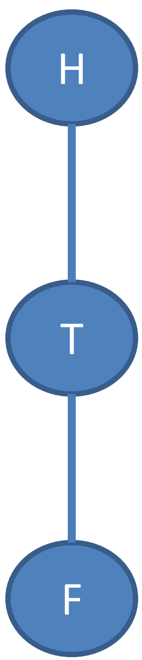


$$L^* = \arg\min_L \left(\sum_{i=1}^{n} m_i(l_i) + \sum_{(v_i,v_j) \in E} d_{ij}(l_i, l_j) \right)$$

Optimization

$$E = \min_{l_H} \min_{l_T} \min_{l_F} \; m_H(l_H) + m_T(l_T) + m_F(l_F) + d_{H,T}(l_H, l_T) + d_{T,F}(l_T, l_F)$$

$$E = \min_{l_H} \; m_H(l_H) + \left(\min_{l_T} \; m_T(l_T) + d_{H,T}(l_H, l_T) + \left(\min_{l_F} \; m_F(l_F) + d_{T,F}(l_T, l_F)\right)\right)$$

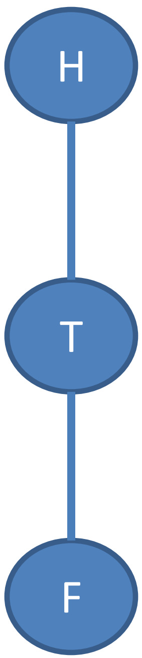


$$L^* = \arg\min_L \left(\sum_{i=1}^{n} m_i(l_i) + \sum_{(v_i, v_j) \in E} d_{ij}(l_i, l_j) \right)$$

Optimization - Dynamic Programming

$$E = \min_{l_H} \; m_H(l_H) + (\min_{l_T} \; m_T(l_T) + d_{H,T}(l_H, l_T) + (\min_{l_F} \; m_F(l_F) + d_{T,F}(l_T, l_F)))$$

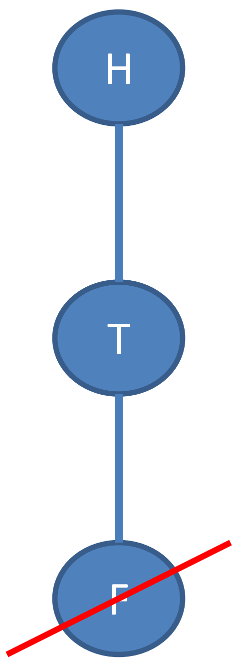
$$s_F(l_T) = \min_{l_F} m_F(l_F) + d_{T,F}(l_T, l_F)$$



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

Optimization - Dynamic Programming

$$E = \min_{l_H} \; m_H(l_H) + (\min_{l_T} \; m_T(l_T) + d_{H,T}(l_H, l_T) + s_F(l_T))$$

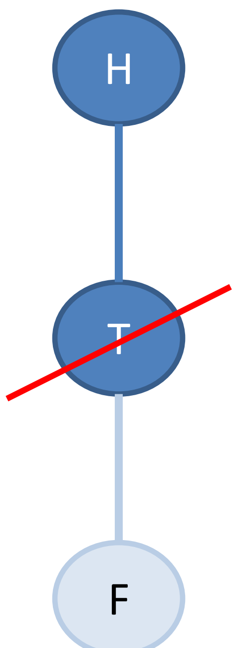


$$s_F(l_T) = \min_{l_F} m_F(l_F) + d_{T,F}(l_T, l_F)$$

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

Optimization - Dynamic Programming

$$E = \min_{l_H} \; m_H(l_H) + (\min_{l_T} \; m_T(l_T) + d_{H,T}(l_H, l_T) + s_F(l_T))$$



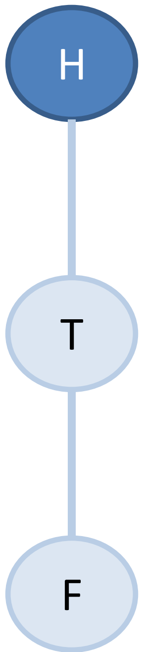
$$s_T(l_H) = \min_{l_T} m_T(l_T) + d_{H,T}(l_H, l_T) + s_F(l_T)$$

$$s_F(l_T) = \min_{l_F} m_F(l_F) + d_{T,F}(l_T, l_F)$$

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

Optimization - Dynamic Programming

$$E = \min_{l_H} \; m_H(l_H) + S_T(l_H)$$



$$s_T(l_H) = \min_{l_T} m_T(l_T) + d_{H,T}(l_H, l_T) + s_F(l_T)$$

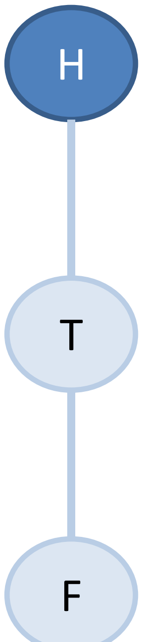
$$s_F(l_T) = \min_{l_F} m_F(l_F) + d_{T,F}(l_T, l_F)$$

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

Optimization - Complexity

Each part can take N locations

$$E = \min_{l_H} \; m_H(l_H) + S_T(l_H)$$ ← ? $O(N)$



For all l_T?

$$s_T(l_H) = \min_{l_T} m_T(l_T) + d_{H,T}(l_H, l_T) + s_F(l_T)$$

→ $$s_F(l_T) = \min_{l_F} m_F(l_F) + d_{T,F}(l_T, l_F)$$

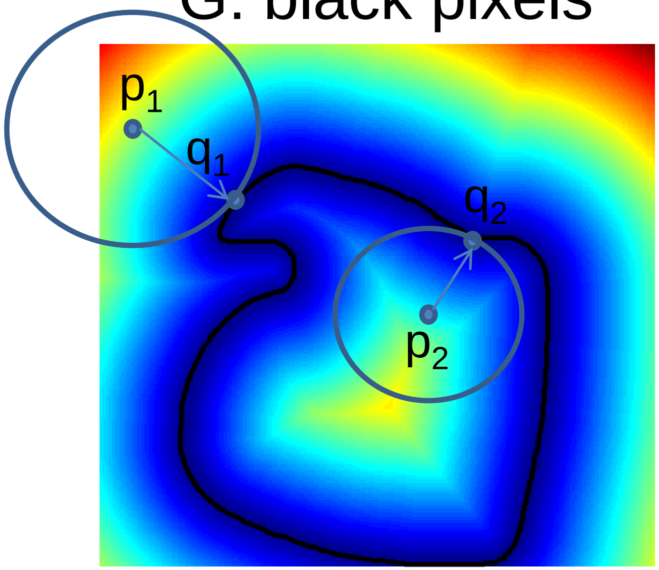
$O(N^2)$

Overall? (for K parts) $O(k \cdot N^2)$

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

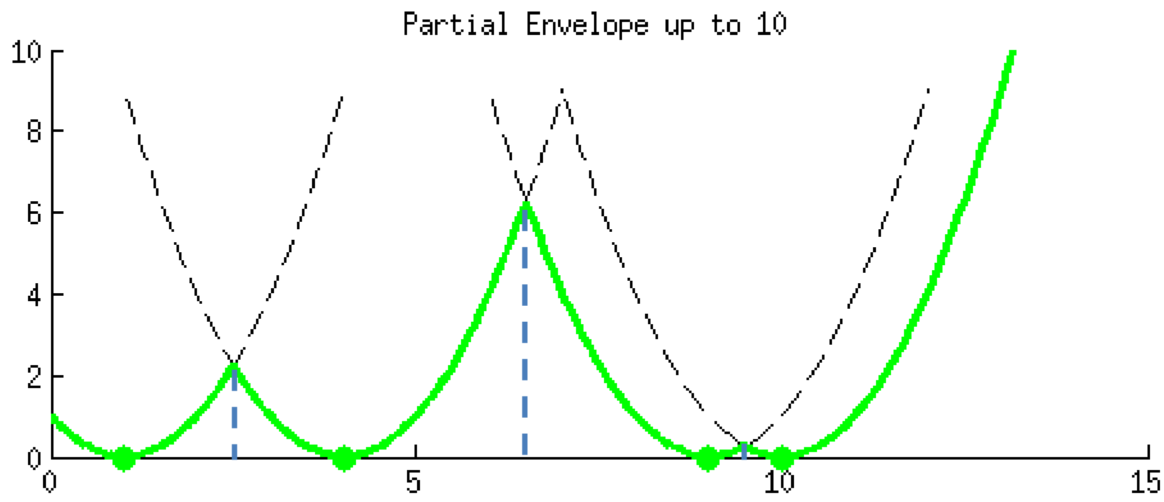
Distance Transform

- For each pixel p, how far away is the nearest pixel q of set S
  - $f(p) = \min_{q \in G} \; d(p,q)$
  - G is often the set of edge pixels



Computing the Distance Transform

- 1-Dimension



$$f(p) = \min_{q \in G} \; d(p, q)$$

$$G = \{1, 4, 9, 10\}$$

Computing the Distance Transform

- Extending to N-Dimensions
  - Savings can be extended from 1-D to N-D cases if deformation cost is separable:

$$d(p_x, q_x, p_y, q_y) = d(p_x, q_x) + d(p_y, q_y)$$

Distance Transform - Applications

- Set distances – e.g. Hausdorff Distance
- Image processing – e.g. Blurring
- Robotics – Motion Planning
- Alignment
  - Edge images
  - Motion tracks
  - Audio warping
- Deformable Part Models (of course!)

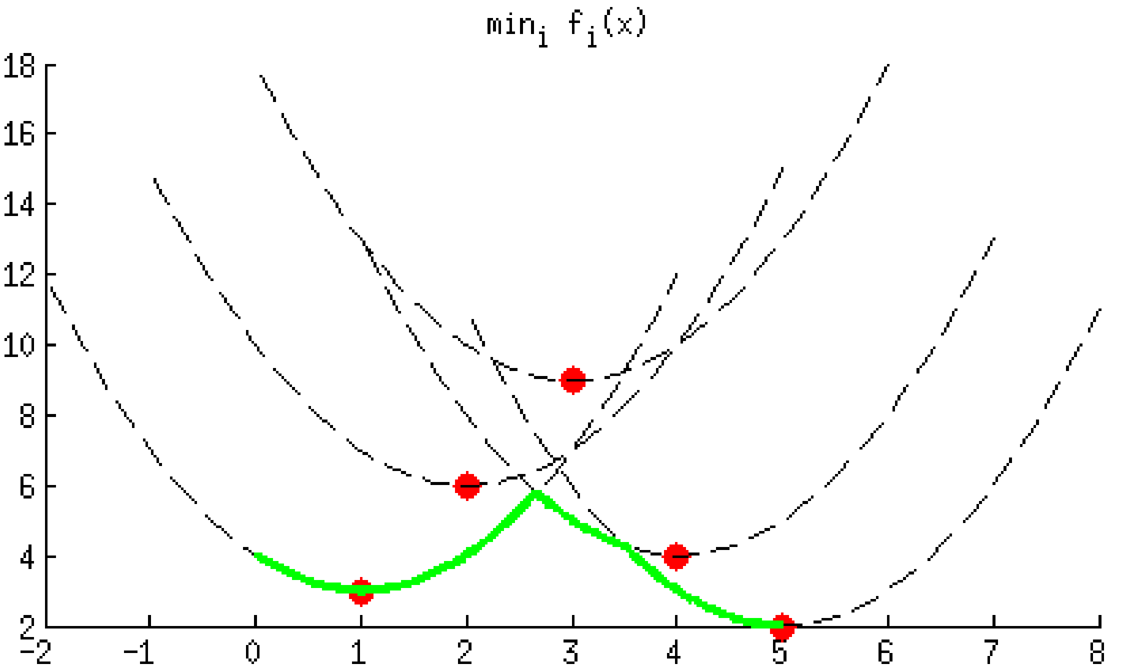
Generalized Distance Transform

- Original form: $f(p) = \min_{q \in G} \ d(p,q)$
- General form: $f(p) = \min_{q \in [1,N]} \ m(q) + d(p,q)$
  - m(q): arbitrary function sampled at discrete points
  - For each p, find a nearby q with small m(q)
  - For original DT: $m(q) = \begin{cases} 0 & : q \in G \\ \infty & : q \notin G \end{cases}$
  - For part models: $s_F(l_T) = \min_{l_F} m_F(l_F) + d_{T,F}(l_T, l_F)$

- For some deformation costs, $O(N^2) \rightarrow O(N)$

How do we do it?

- Key idea: Construct lower “envelope” of data



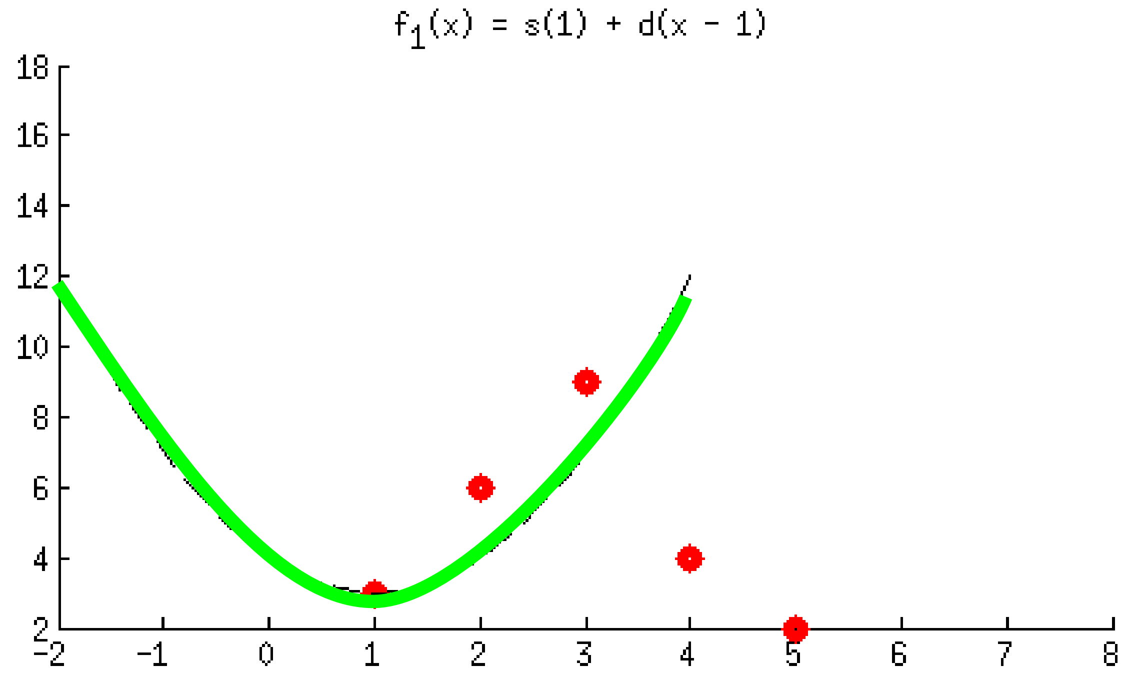
$$f(p) = \min_{q \in [1,N]} \; f_q(p)$$

- Given envelope, compute f(p) in O(N) time
- Goal: Compute envelope in O(N) time

Computing the Lower Envelope

- Key idea: Keep track of intersection points
  - $f_i(x)$, $f_j(x)$ only intersect once (let i<j)

Start(1) = -Inf
End(1) = Inf

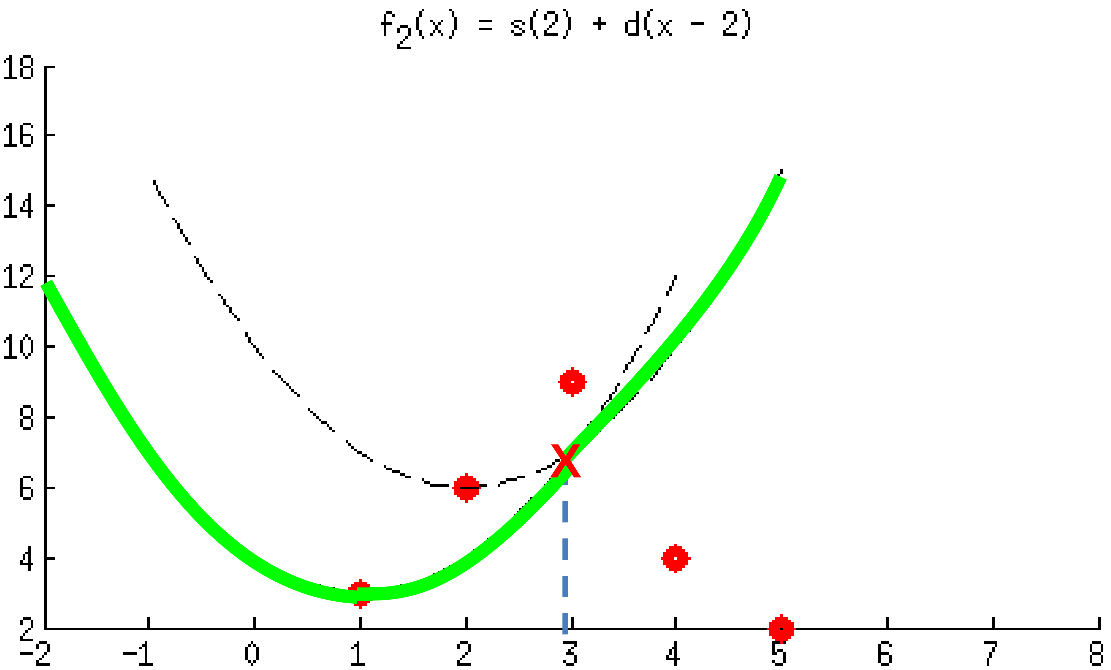


Computing the Lower Envelope

- Key idea: Keep track of intersection points
  - $f_i(x)$, $f_j(x)$ only intersect once (let i<j)

Start(1) = -Inf
End(1) = 3

Start(2) = 3
End(2) = Inf



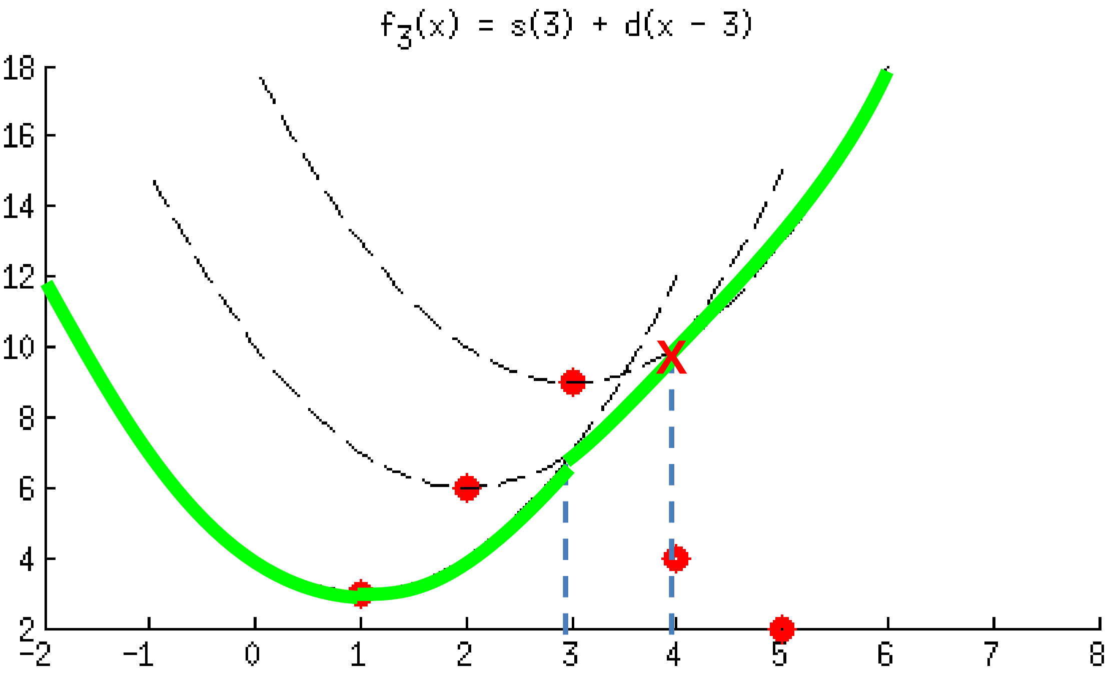
Computing the Lower Envelope

- Key idea: Keep track of intersection points
  - $f_i(x)$, $f_j(x)$ only intersect once (let $i<j$)

Start(1) = -Inf
End(1) = 3

Start(2) = 3
End(2) = 4

Start(3) = 4
End(3) = Inf



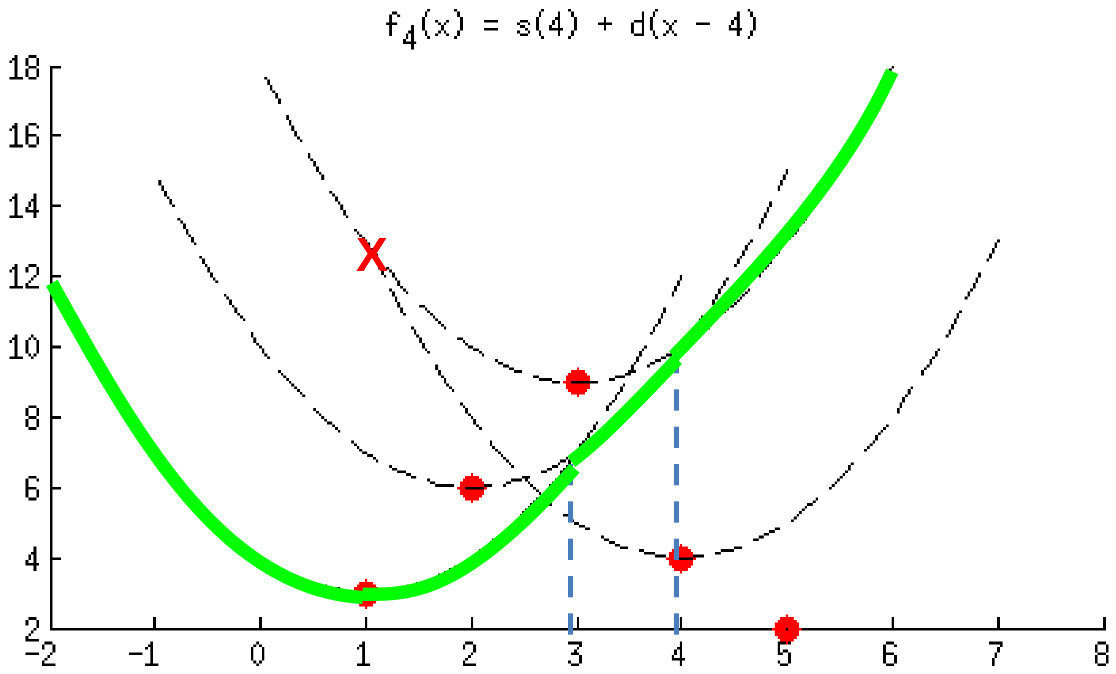
Computing the Lower Envelope

- Key idea: Keep track of intersection points
  - $f_i(x)$, $f_j(x)$ only intersect once (let i<j)

Start(1) = -Inf
End(1) = 3

Start(2) = 3
End(2) = 4

Start(3) = 4
End(3) = Inf



Computing the Lower Envelope

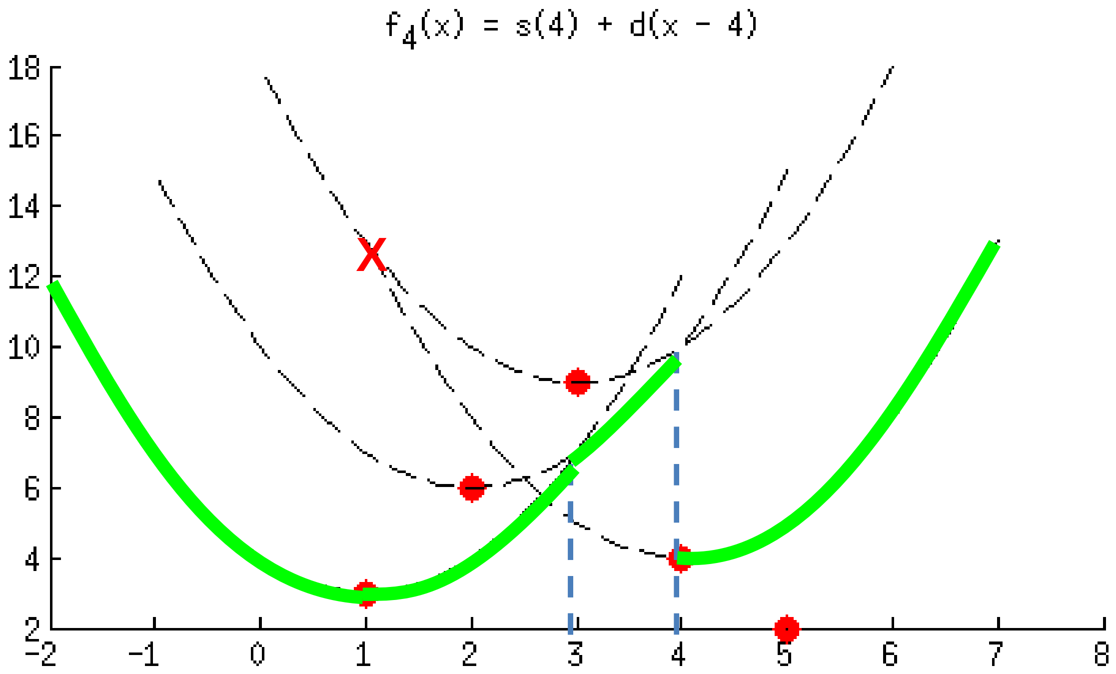
- Key idea: Keep track of intersection points
  - $f_i(x)$, $f_j(x)$ only intersect once (let $i<j$)

Start(1) = -Inf
End(1) = 3

Start(2) = 3
End(2) = 4

Start(3) = []
End(3) = []

Start(4) = 4
End(4) = Inf



Computing the Lower Envelope

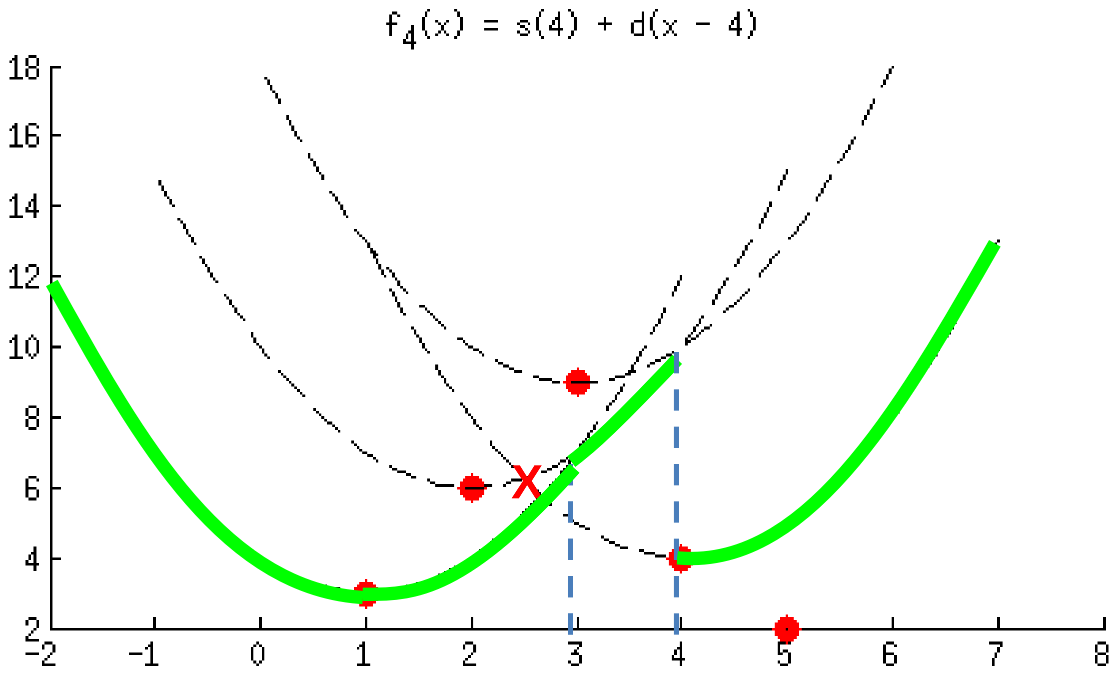
- Key idea: Keep track of intersection points
  - $f_i(x)$, $f_j(x)$ only intersect once (let $i<j$)

Start(1) = -Inf
End(1) = 3

Start(2) = 3
End(2) = 4

Start(3) = []
End(3) = []

Start(4) = 4
End(4) = Inf



Computing the Lower Envelope

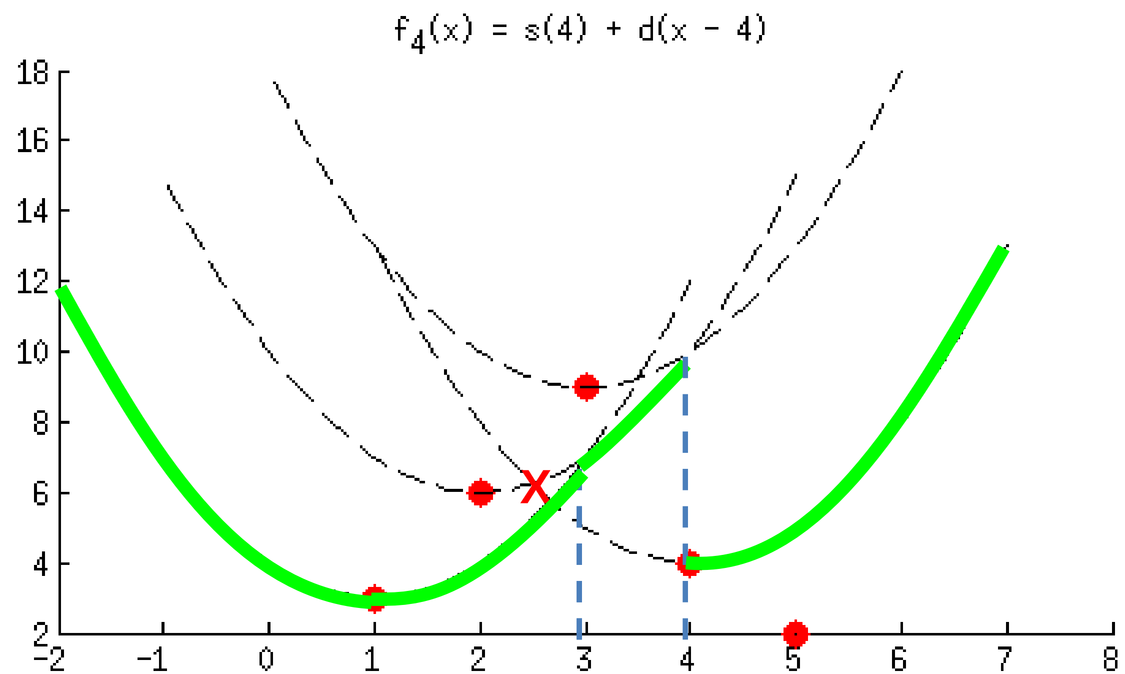
- Key idea: Keep track of intersection points
  - $f_i(x)$, $f_j(x)$ only intersect once (let i<j)

Start(1) = -Inf
End(1) = 3

Start(2) = 3
End(2) = 4

Start(3) = []
End(3) = []

Start(4) = 4
End(4) = Inf



Computing the Lower Envelope

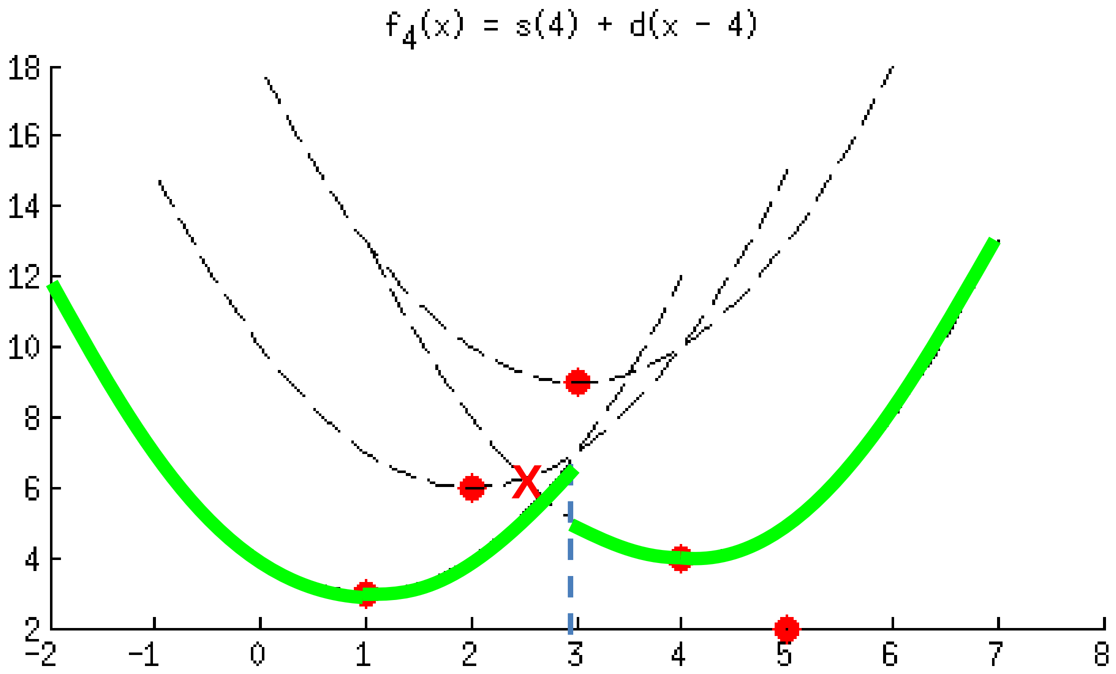
- Key idea: Keep track of intersection points
  - $f_i(x)$, $f_j(x)$ only intersect once (let i<j)

Start(1) = -Inf
End(1) = 3

Start(2) = []
End(2) = []

Start(3) = []
End(3) = []

Start(4) = 3
End(4) = Inf



Computing the Lower Envelope

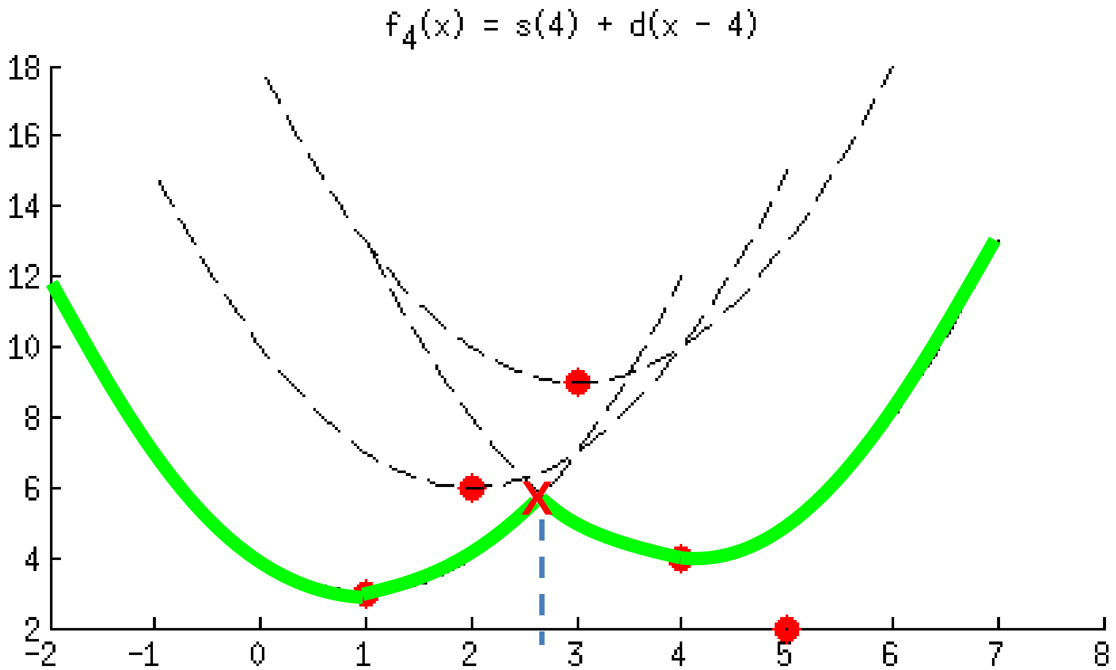
- Key idea: Keep track of intersection points
  - $f_i(x)$, $f_j(x)$ only intersect once (let i<j)

Start(1) = -Inf
End(1) = 2.6

Start(2) = []
End(2) = []

Start(3) = []
End(3) = []

Start(4) = 2.6
End(4) = Inf



Computing the Lower Envelope

- Key idea: Keep track of intersection points
  - $f_i(x)$, $f_j(x)$ only intersect once (let $i<j$)

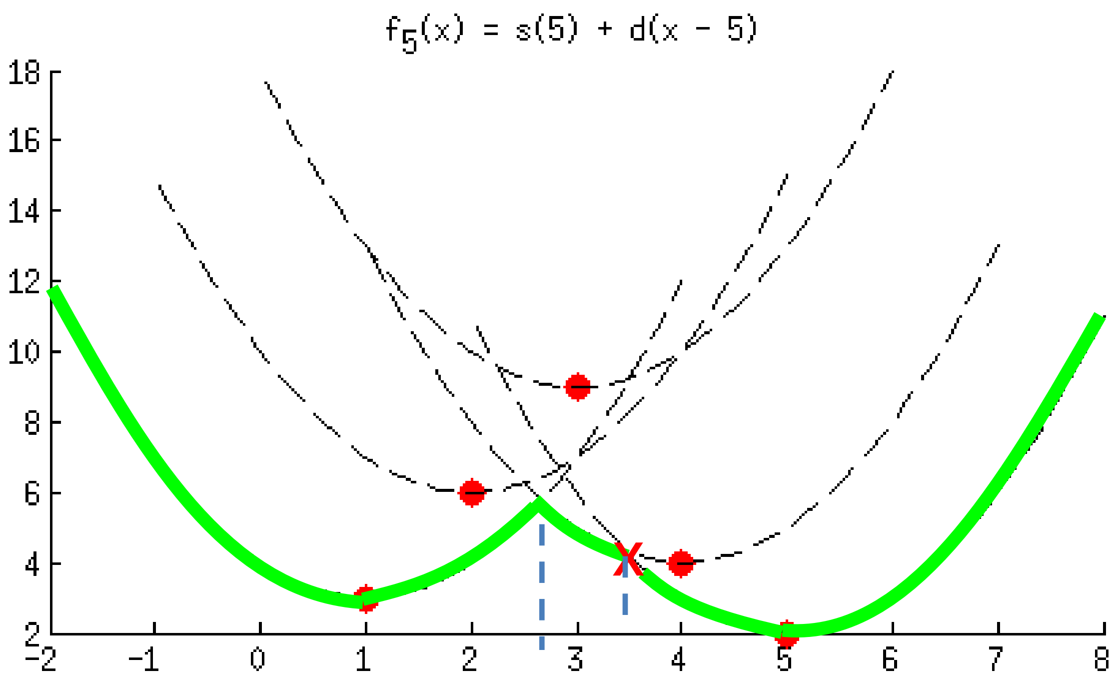
Start(1) = -Inf
End(1) = 2.6

Start(2) = []
End(2) = []

Start(3) = []
End(3) = []

Start(4) = 2.6
End(4) = 3.5

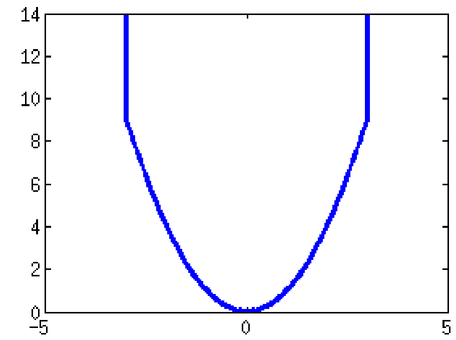
Start(5) = 3.5
End(5) = Inf



Is This O(N)?

- Yes!
- N parabolas are added
  - For each addition, may remove up to O(N) parabolas
- But, each parabola can only be deleted once
- Thus, O(N) additions, and at most O(N) deletions

What distances can we use?



- Quadratic (with linear shift)

$$d(p,q) = \alpha(p-q)^2 + \beta(p-q)$$

- Abs. diff

$$d(p,q) = \alpha|p-q|$$

- Min-composition

$$d(p,q) = \min(d_1(p,q), d_2(p,q))$$

- Bounded (Requires extra bookkeeping)

$$d_\tau(p,q) = \begin{cases} d(p,q) & : |p-q| < \tau \\ \infty & : |p-q| \geq \tau \end{cases}$$

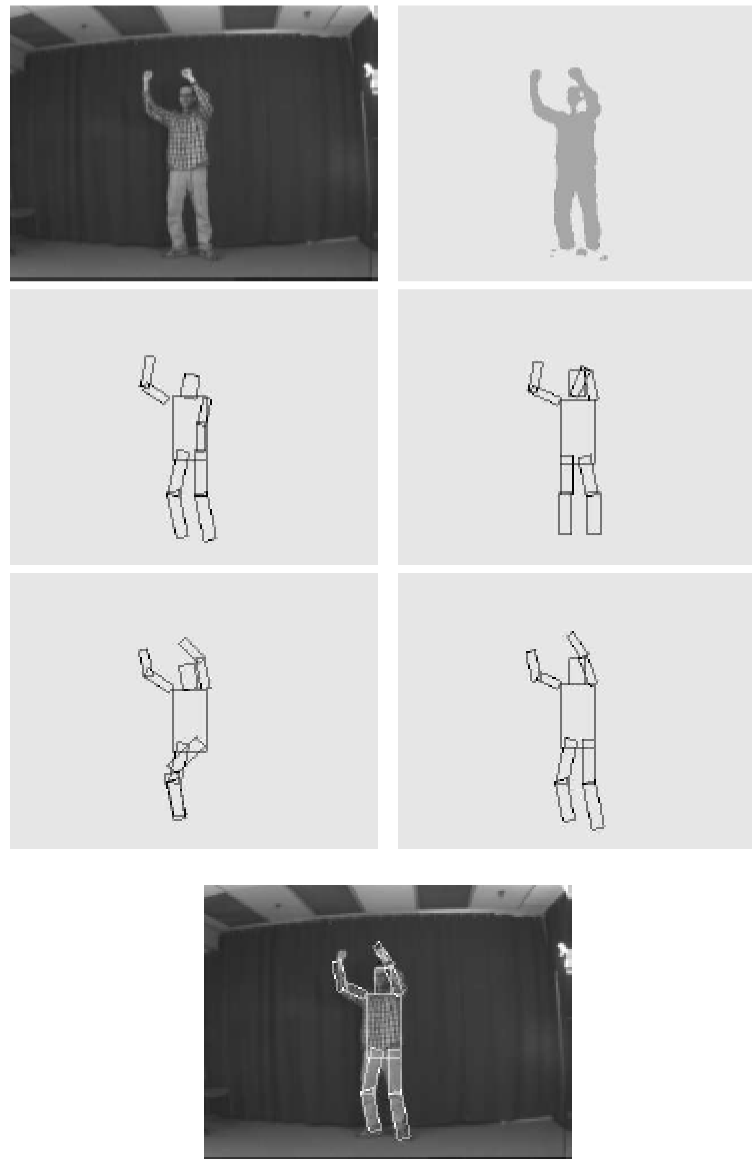
Back to the Pictorial structures model

Problem: May need to infer more than one configuration

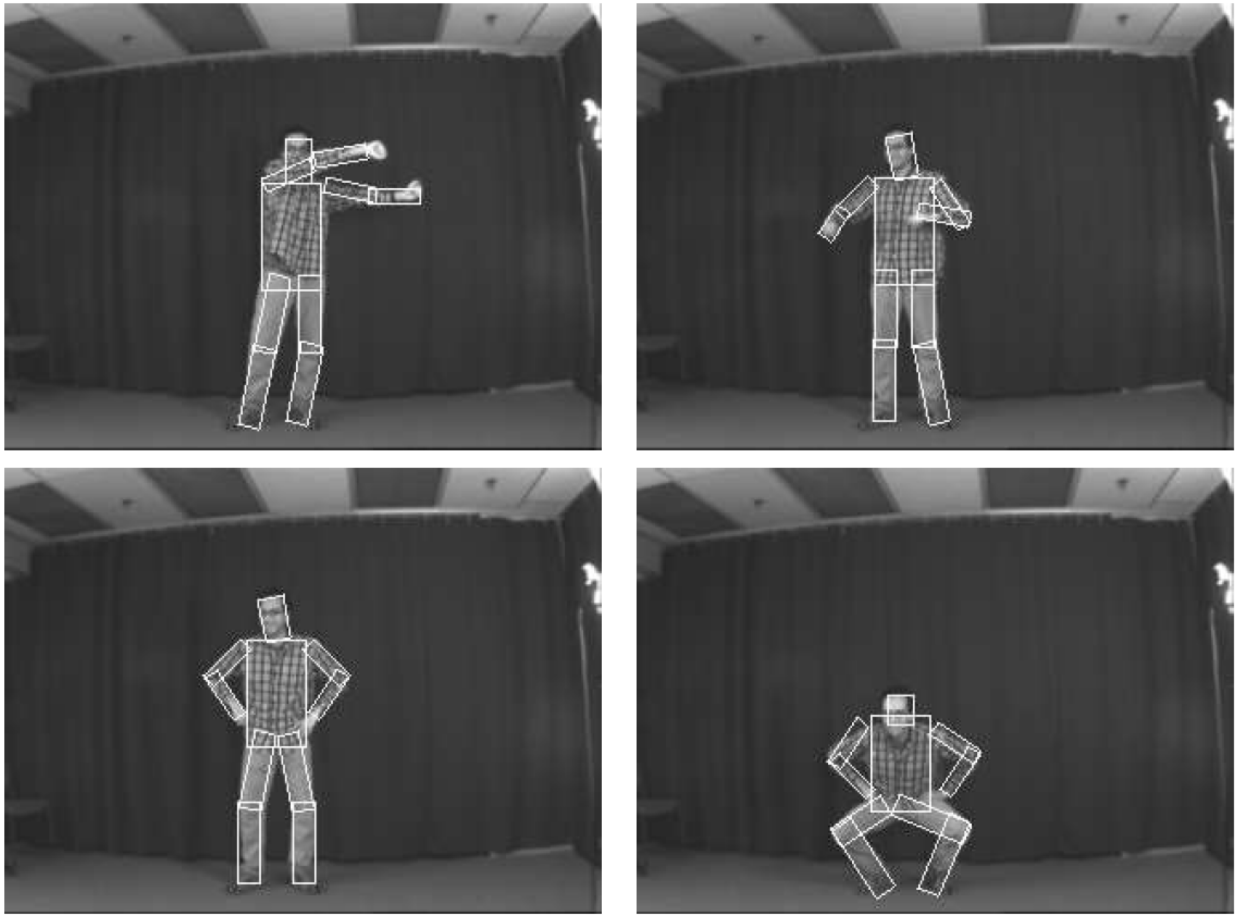
a) May be more than one object
   - Report scores for every root position, apply NMS

b) Optimal solution may be incorrect
   - Sampling
     - Sample root node, then each node given parent, until all parts are sampled

Sample poses from likelihood and choose best match with Chamfer distance to map

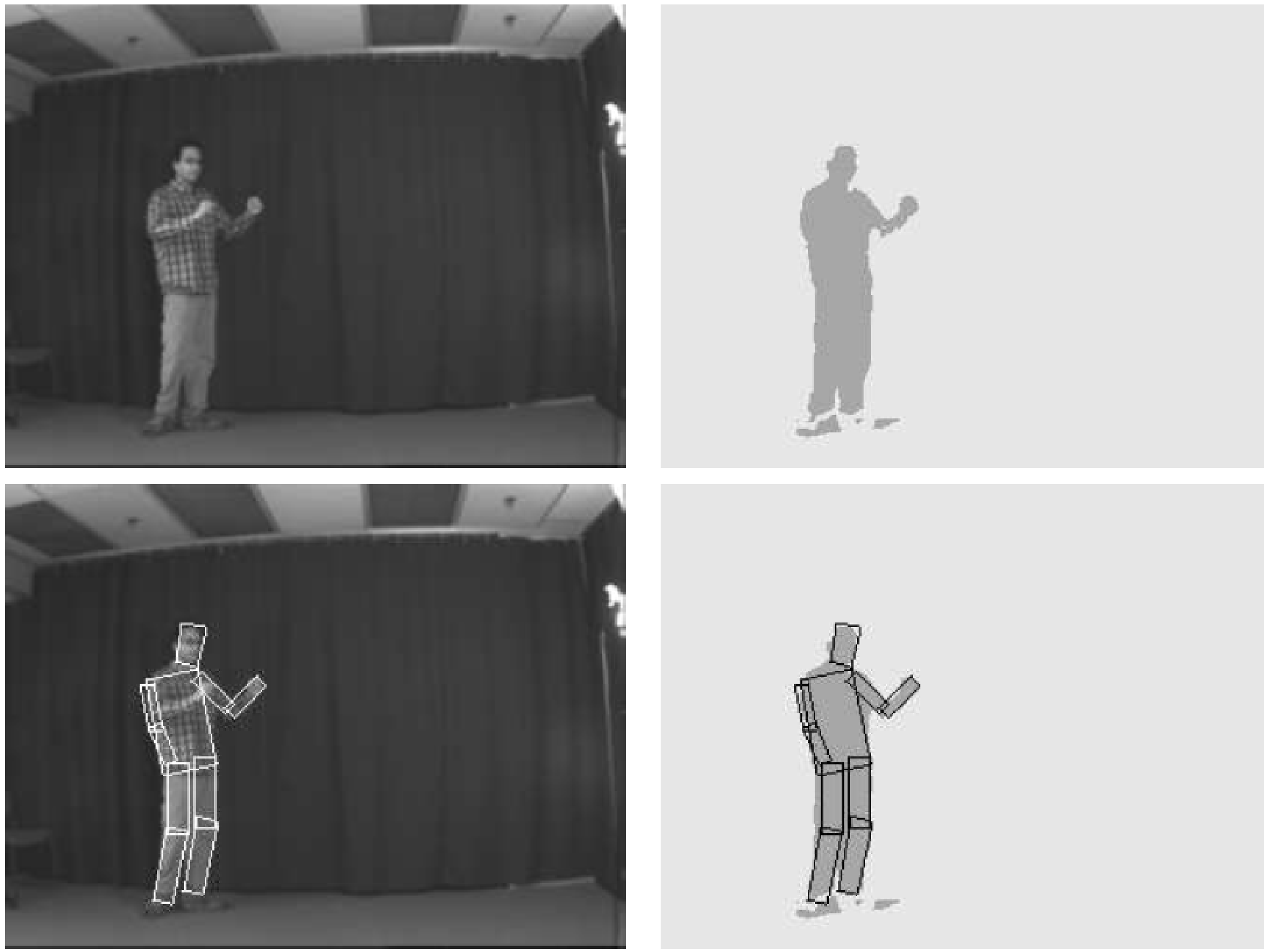


Results for person matching



Results for person matching - Mistakes

- Ambiguous Background subtracted image



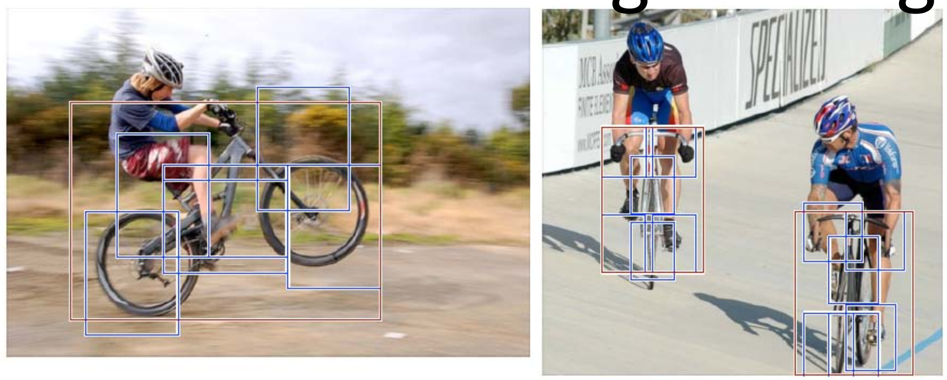
Recently enhanced pictorial structures

BMVC 2009

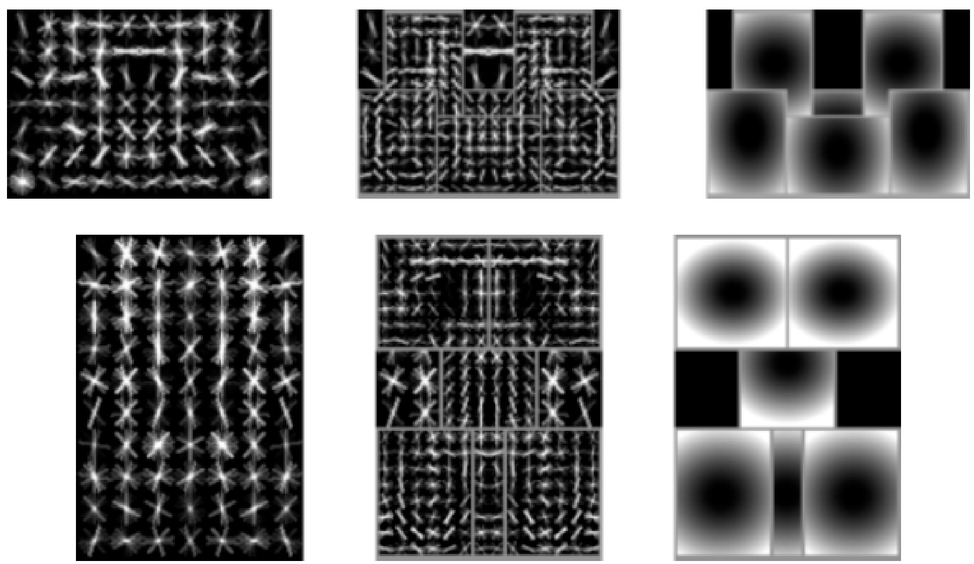
Deformable Latent Parts Model

Useful parts discovered during training

Detections



Template Visualization



root filters
coarse resolution

part filters
finer resolution

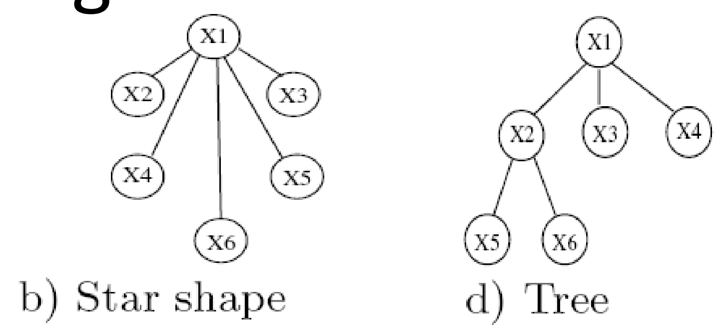
deformation
models

Felzenszwalb et al. 2008

Things to Remember

- Rather than searching for whole object, can locate parts that compose the object
  - Better encoding of spatial variation

- Models can be broken down into part appearance and spatial configuration
  - Wide variety of models



- Efficient optimization is often tricky, but many tricks available
  - Dynamic programming, Distance transforms