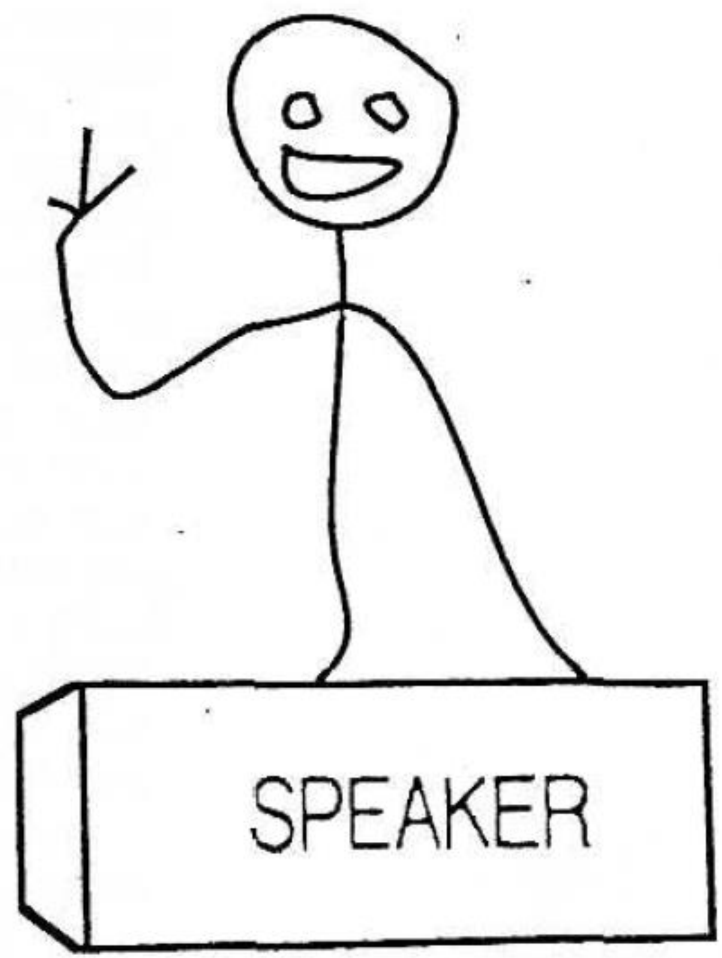
Putting it together

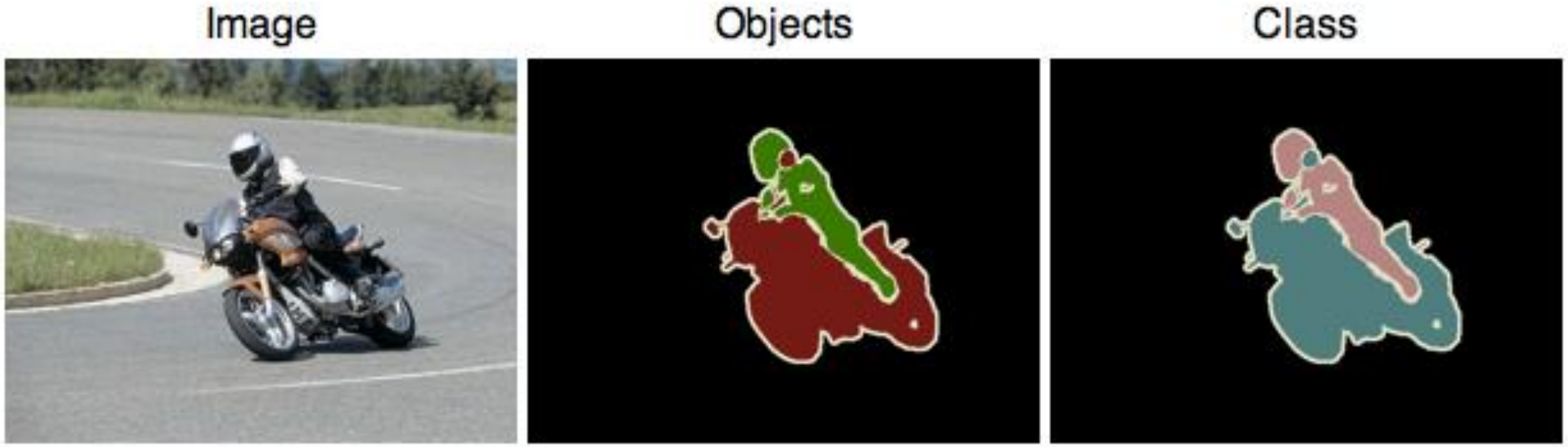
What we see

What we really see



Object Detection

Object Segmentation

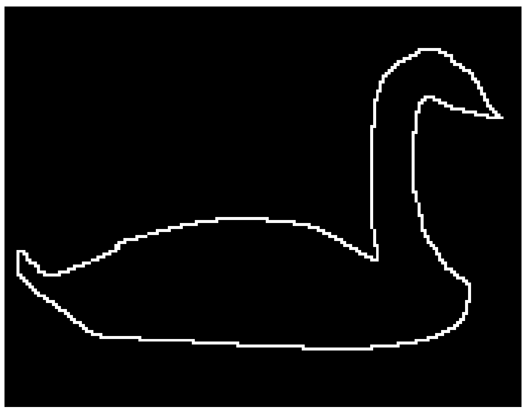


Image

Person Layout

Basic Shape Comparison

Contour based shape matching

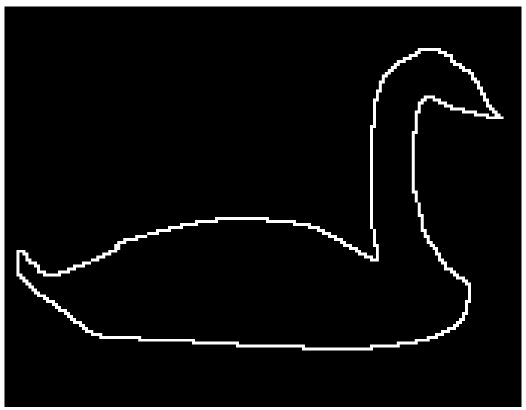


template shape

query image

How to find the template shape in the query image?

Contour based shape matching

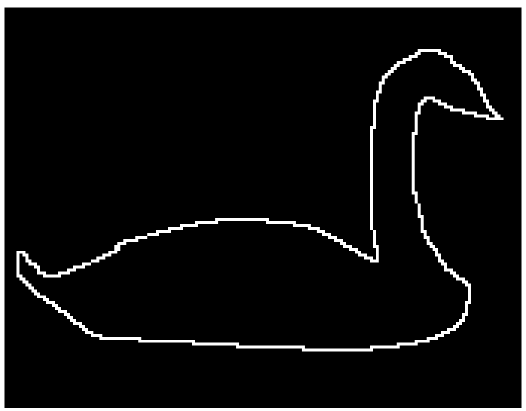


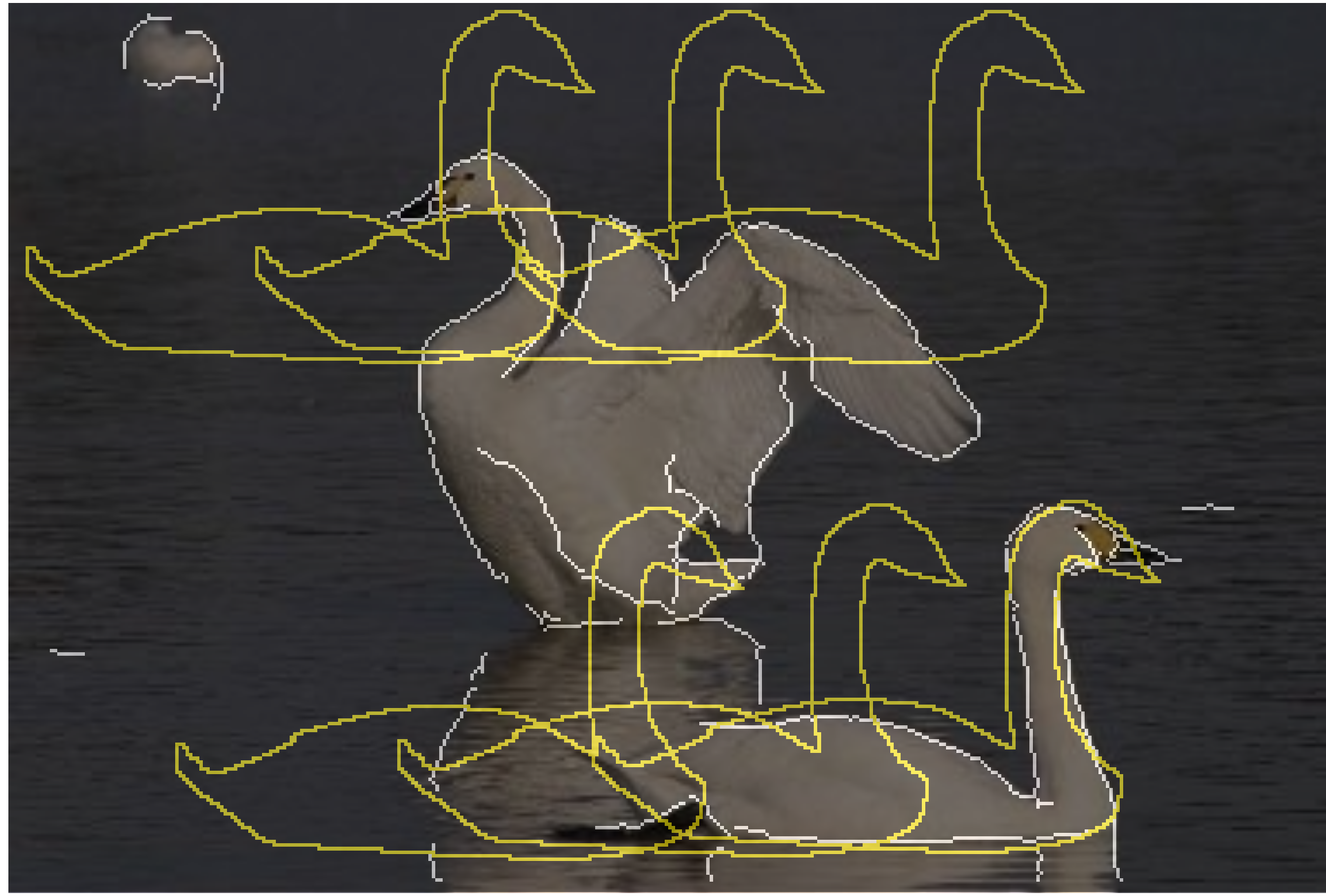
template shape

query image

Detect edges in query image, binary edge or edge with soft-magnitude

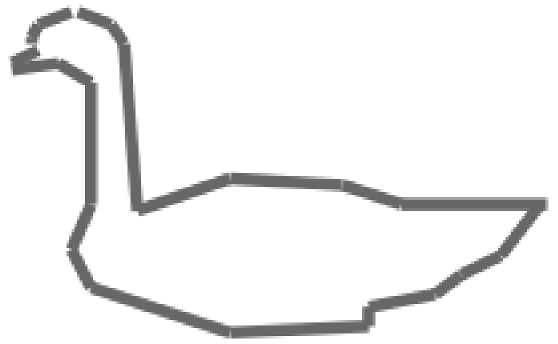
Contour based shape matching

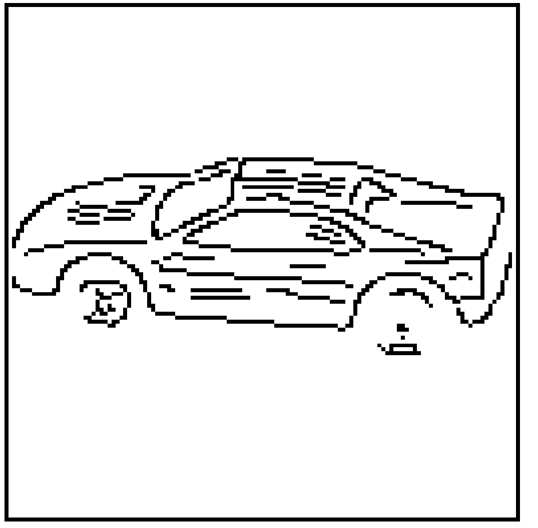




template shape query image

Slide template over query image edge map





Let p,q be two edge sets to be compared

$$ShapeDiff(p,q) = \sum_{x \in p} min_{y \in q} ||x - y||^2$$

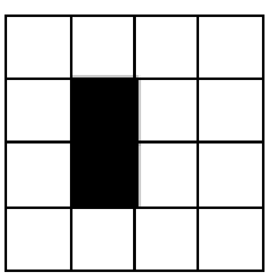
Distance transform: $D_q(x)$

Distance Transform Definition

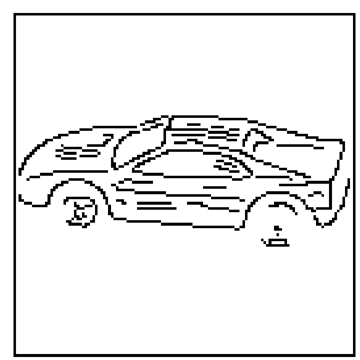
Set of points, P, some distance $\| \bullet \|$

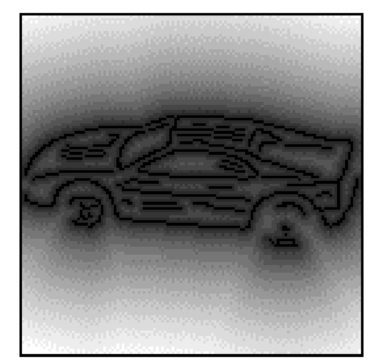
$$D_P(x) = \min_{y \in P} \| x - y \|$$

- For each location x distance to nearest y in P
- Think of as cones rooted at each point of P



2	1	2	3
1	0	1	2
1	0	1	2
2	1	2	3



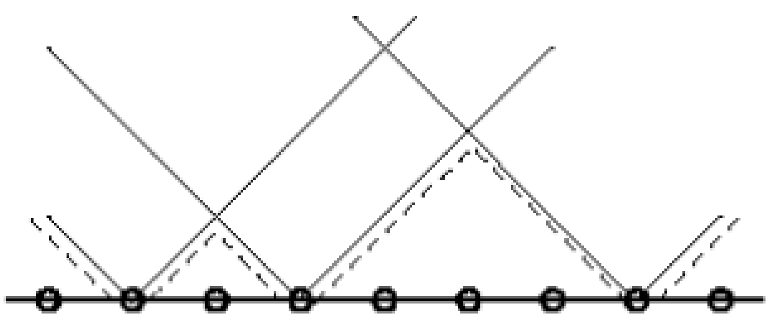


- Two pass O(n) algorithm for 1D L_1 norm (for simplicity just distance)
  1. Initialize: For all j
     $D[j] \leftarrow 1_{\mathbf{P}}[j]$
  2. Forward: For j from 1 up to n-1
     $D[j] \leftarrow \min(D[j], D[j-1]+1)$

1	0

  3. Backward: For j from n-2 down to 0
     $D[j] \leftarrow \min(D[j], D[j+1]+1)$

0	1

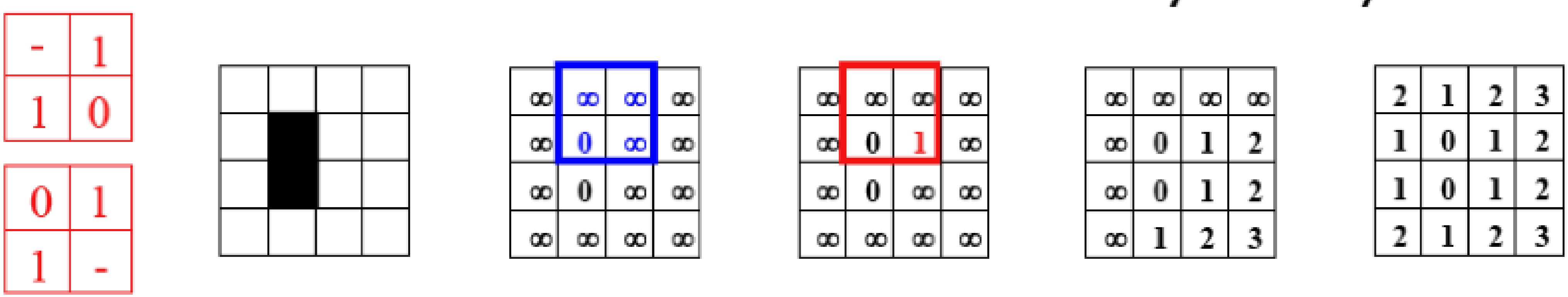


∞	0	∞	0	∞	∞	∞	0	∞

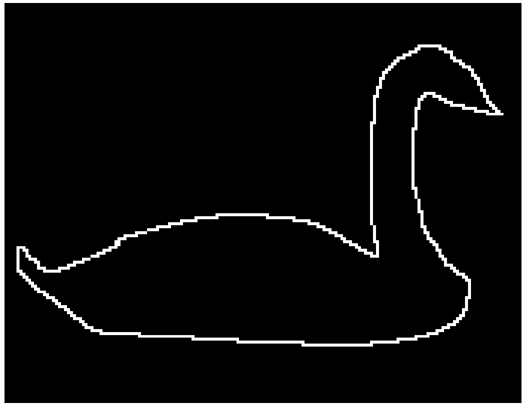
∞	0	1	0	1	2	3	0	1

1	0	1	0	1	2	1	0	1

- 2D case analogous to 1D
  - Initialization
  - Forward and backward pass
    - Fwd pass finds closest above and to left
    - Bwd pass finds closest below and to right
- Note nothing depends on 0,∞ form of initialization
  - Can "distance transform" arbitrary array



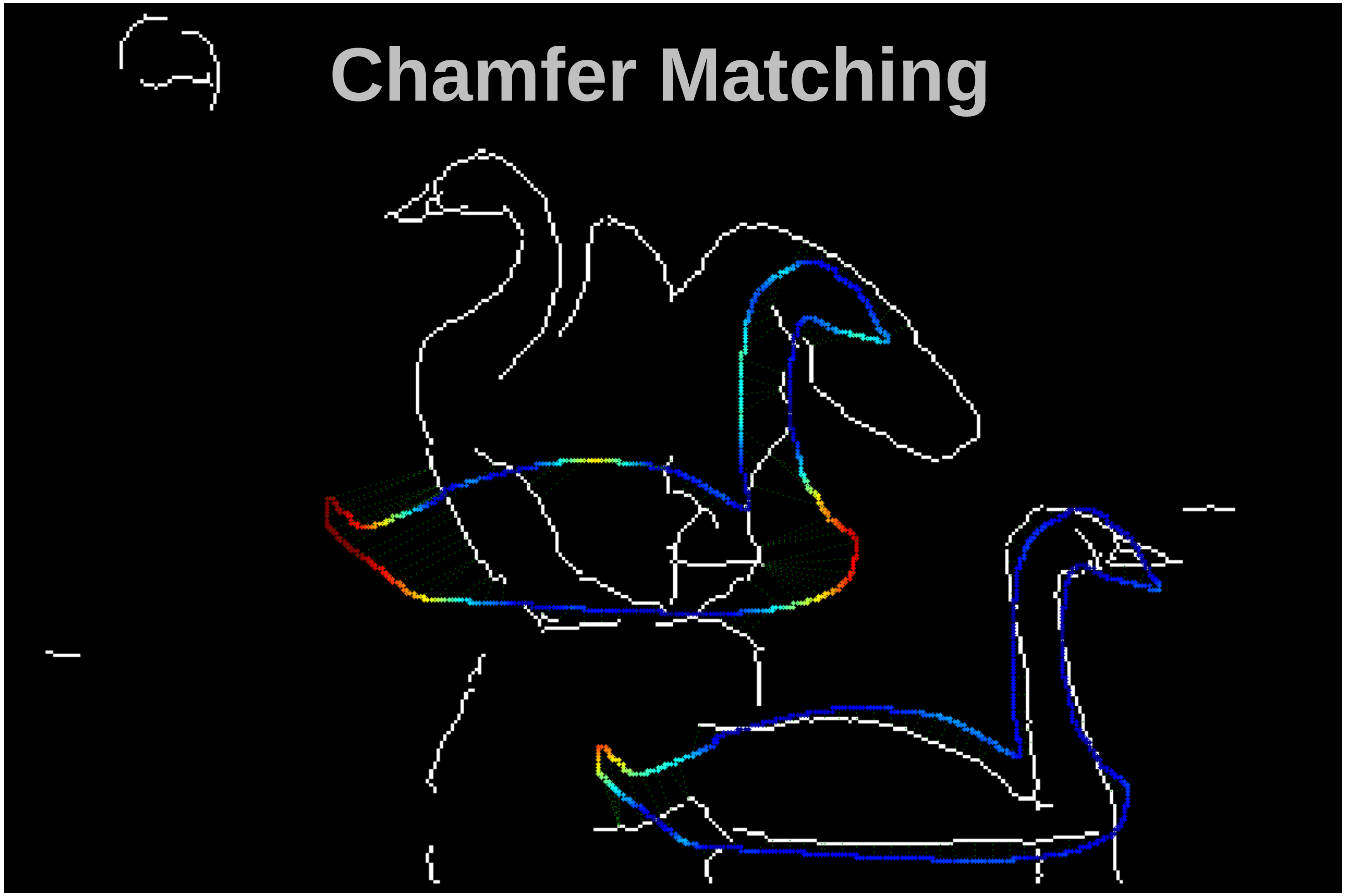
Contour based shape matching



template shape

query image

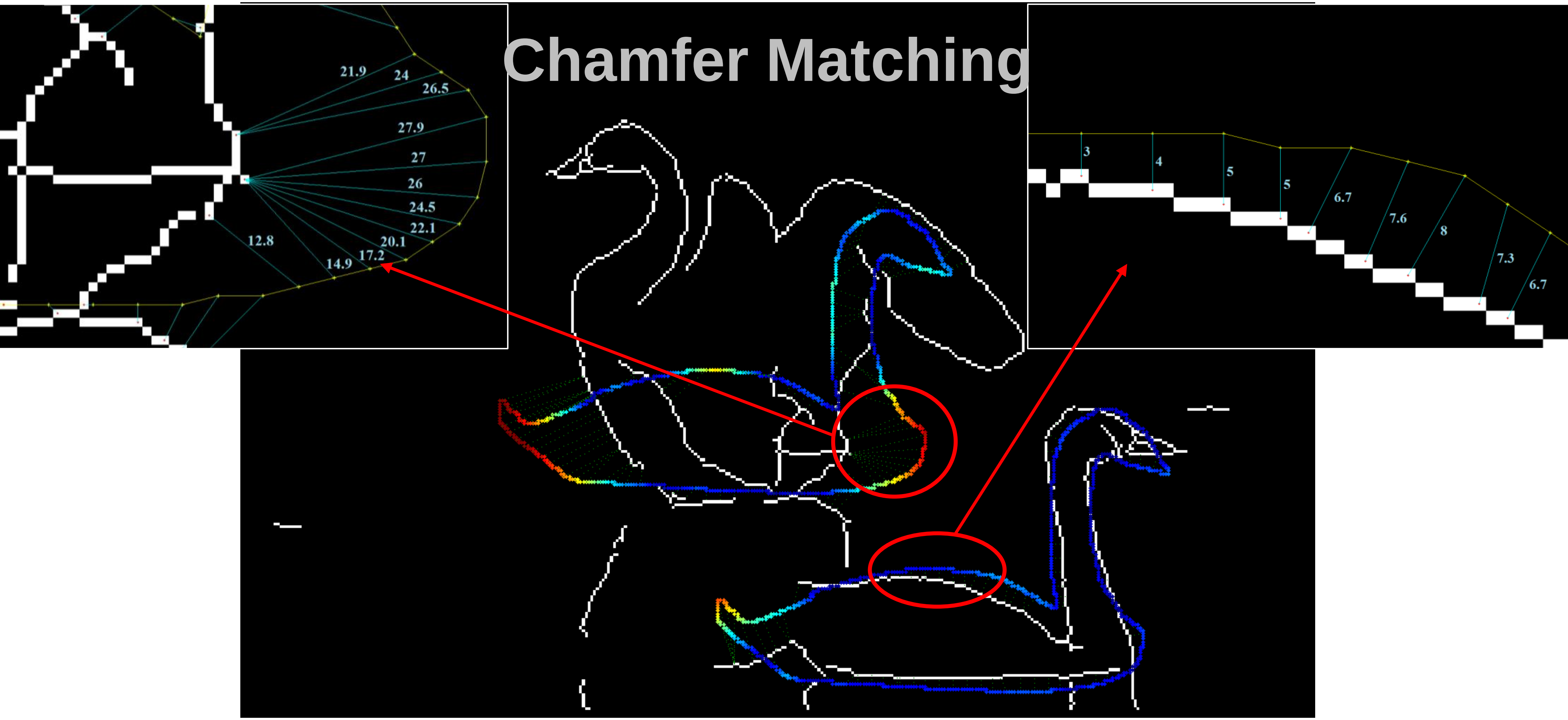
At each location, compute distance from each pixel *p* in template to closest edge in image *q*. (red is large distance, blue is low)
Location with the lowest *average* cost match wins (over template pixels)



1. Slide template *T* by *(u,v)* over query image edge map *E*

$$T(u,v) = \{(x+u, y+v) \mid (x,y) \in T\}$$

Chamfer Matching



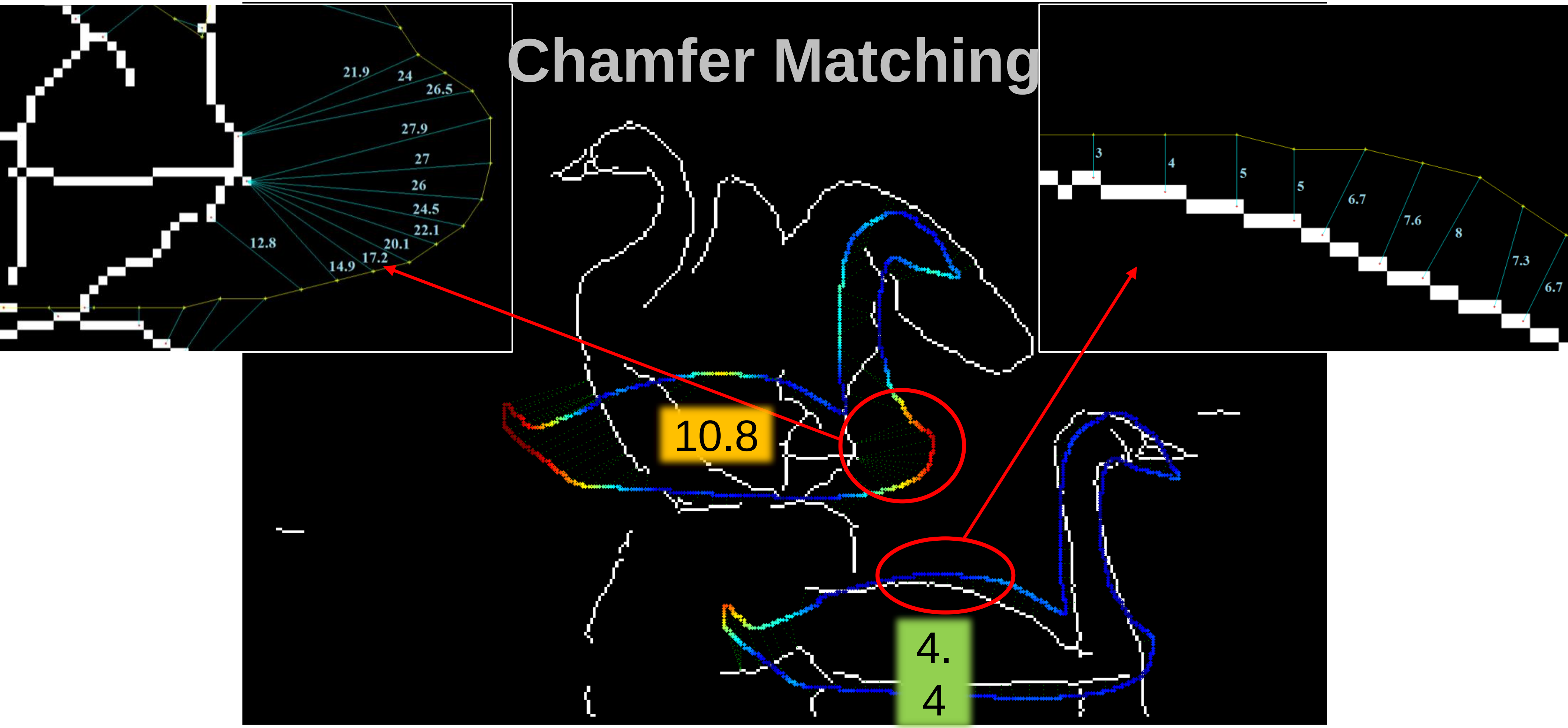
1. Slide template *T* by *(u,v)* over query image edge map *E*
2. **Matching cost of each pixel *p* in the shifted template *T(u,v)* is its shortest distance to any edge pixel *q* in the edge map *E***

$$c(p) = \min_{q \in E} \|p - q\|_2$$

Brute force computation takes $O(n \| T \|)$

Using distance transform, it takes $O(n) + O(\| T \|)$

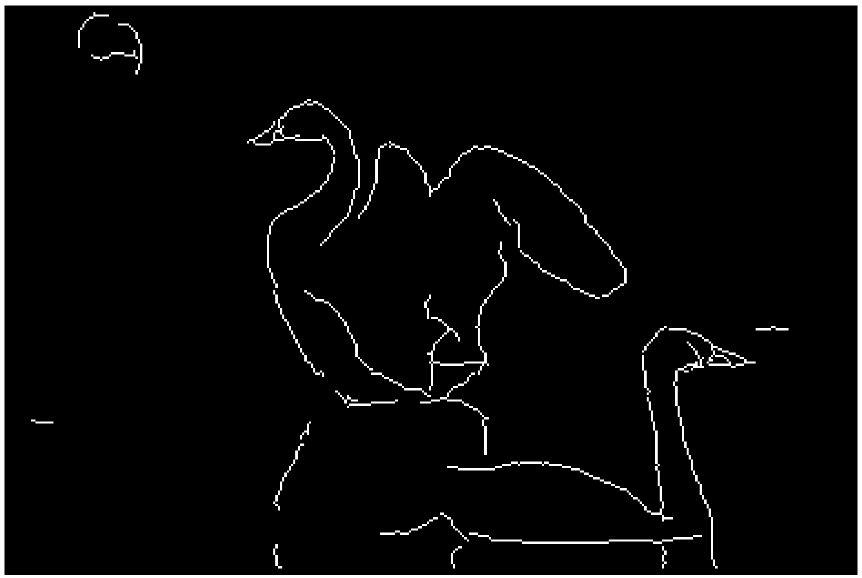
Chamfer Matching



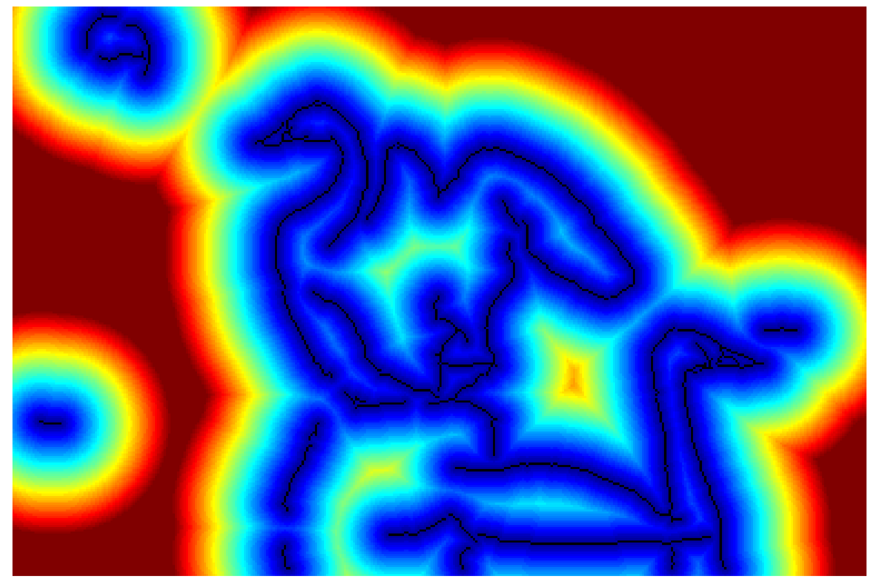
1. Slide template *T* by *(u,v)* over query image edge map *E*
2. Matching cost of each pixel *p* in the shifted template *T(u,v)*
   is its shortest distance to any edge pixel *q* in the edge map *E*
3. **Total cost of the shifted template is the average cost of each shifted template pixel**

$$Cost(u,v) = \frac{1}{\|T\|} \sum_{p \in T(u,v)} \min_{q \in E} \|p - q\|_2$$

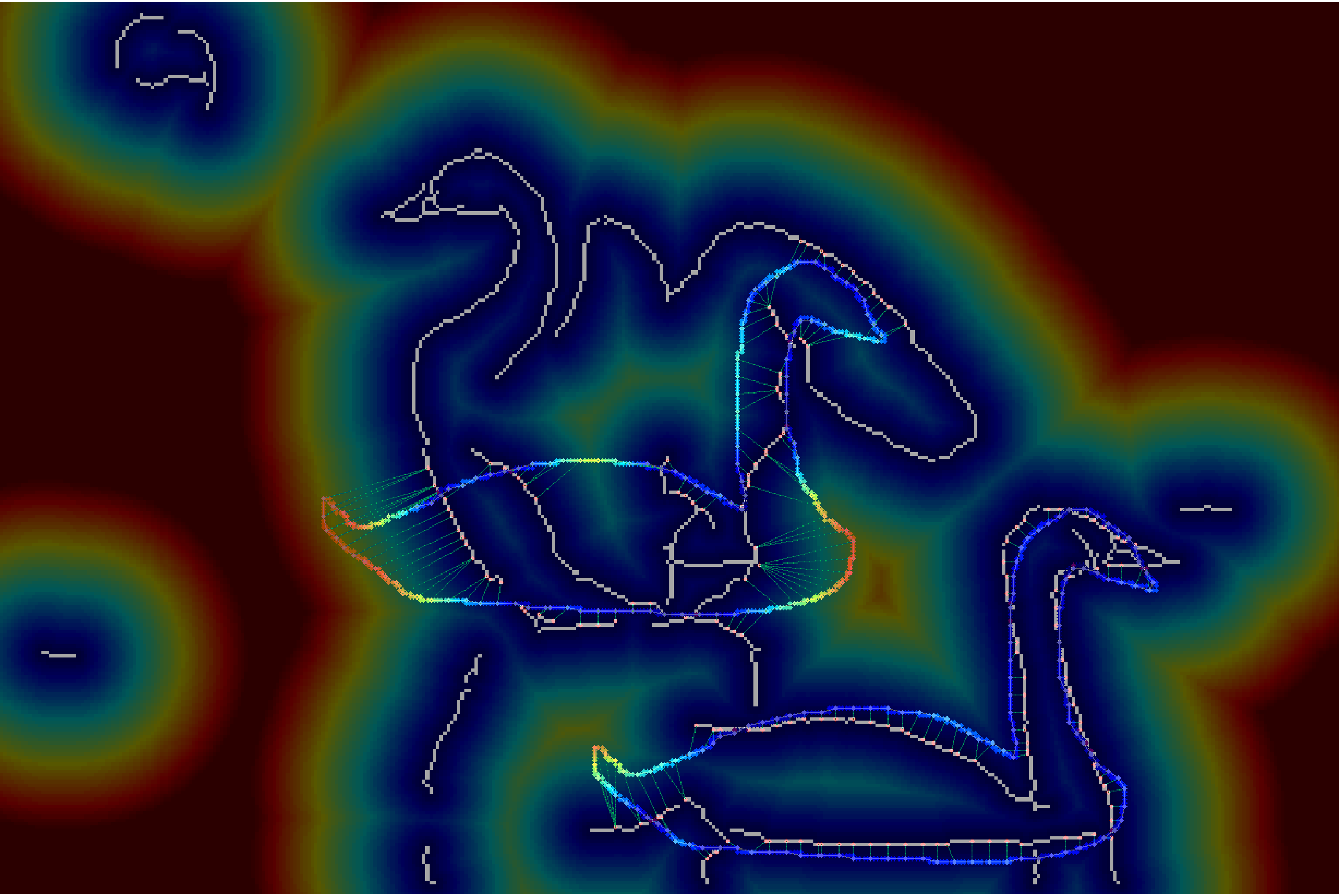
Recall: generalized distance transform

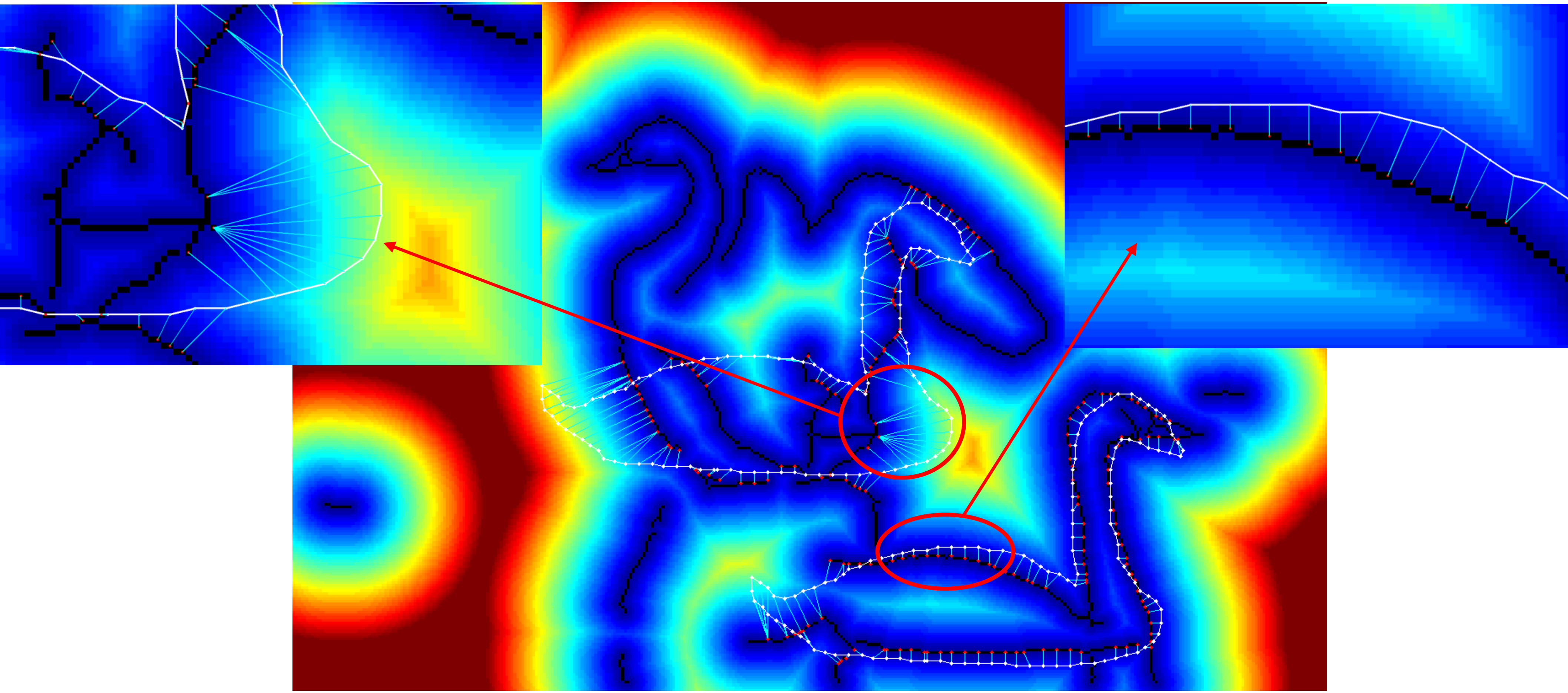


E



$$c(p) = \min_{q \in E} \|p - q\|_2$$





Now finding the cost of each point is just a look up!

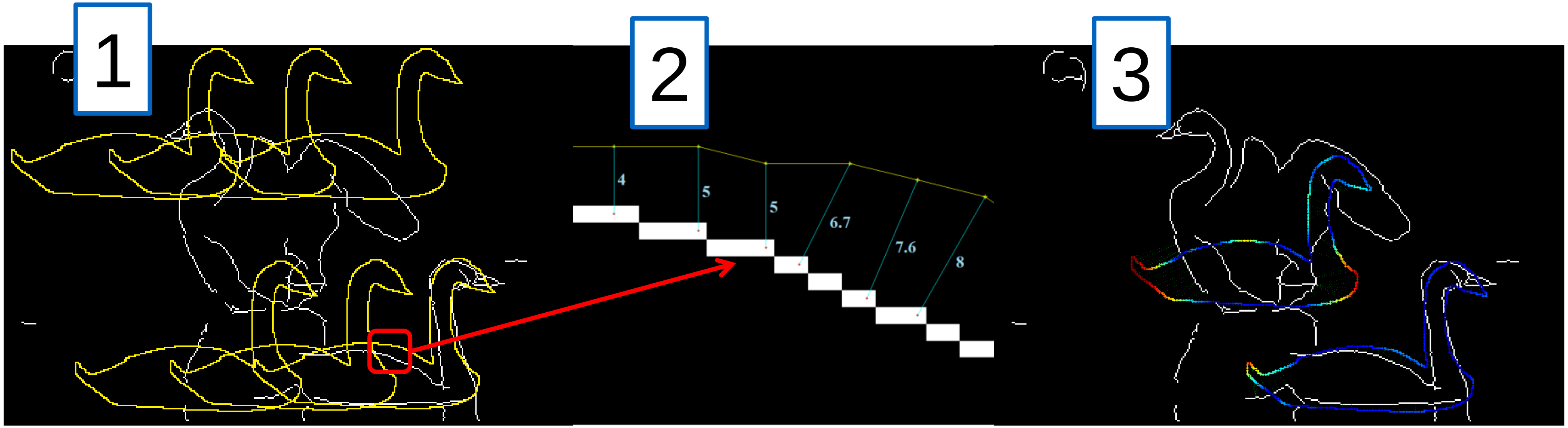
Evaluation time for each shift is just $O(\|T\| O(\min_{q\in E}\|p-q\|_2)) = O(\|T\|)$

Total running time for *m* shifts is: (typically, *m* = *n*)

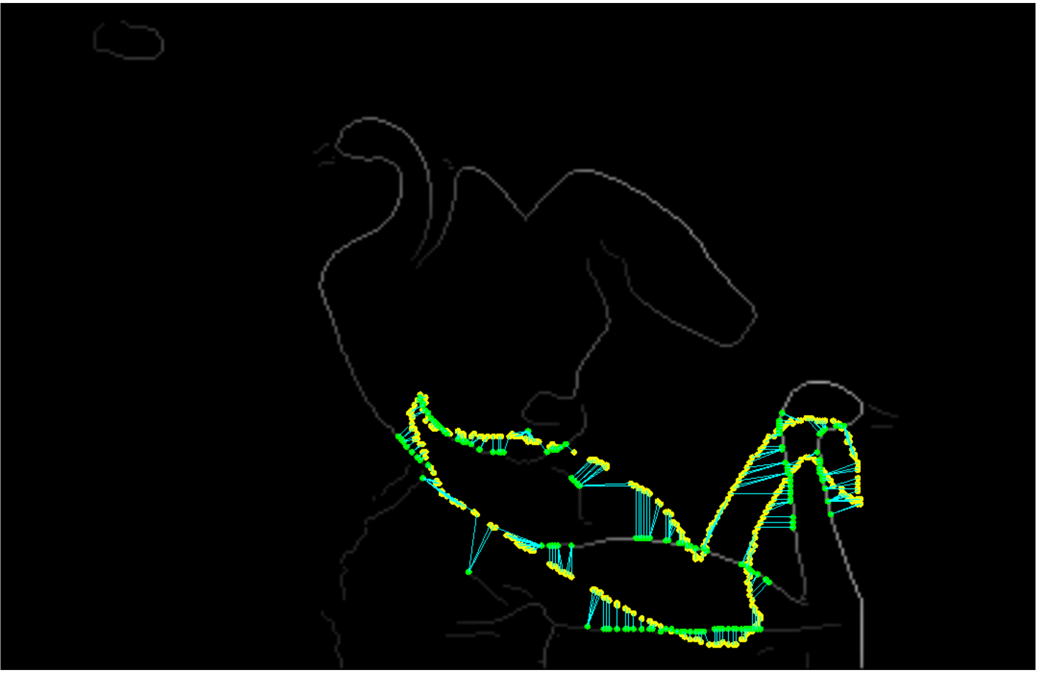
$$O(n + m\|T\| O(\min_{q\in E}\|p-q\|_2)) = O(n) + O(m\|T\|)$$

Chamfer Matching Review

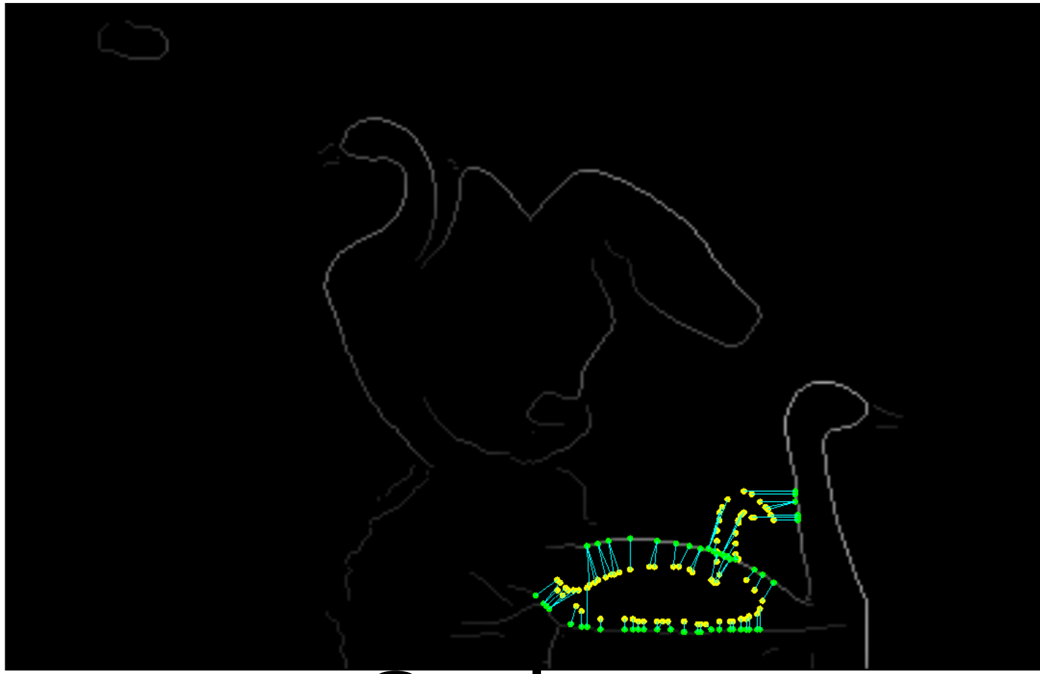
0. Detect edges in query image
1. Slide template over query image edge map
2. Find closest edge pixel in image for each shifted template pixel
3. At each location, compute average distance from each pixel in template to closest edge in image
4. Lowest cost match wins



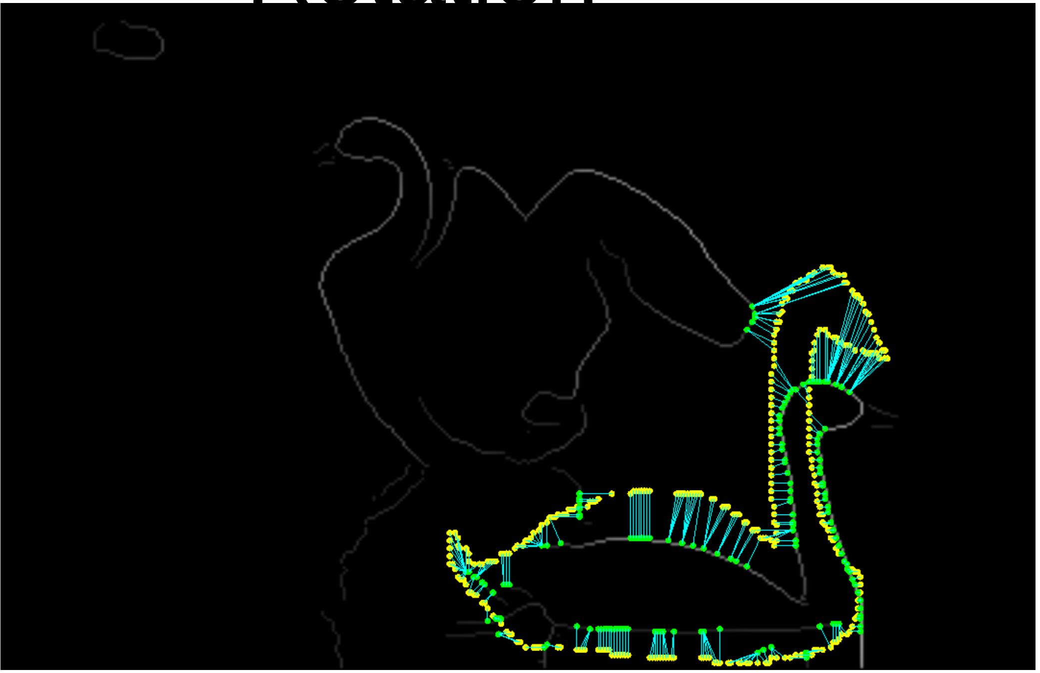
Weaknesses of Chamfer Matching?



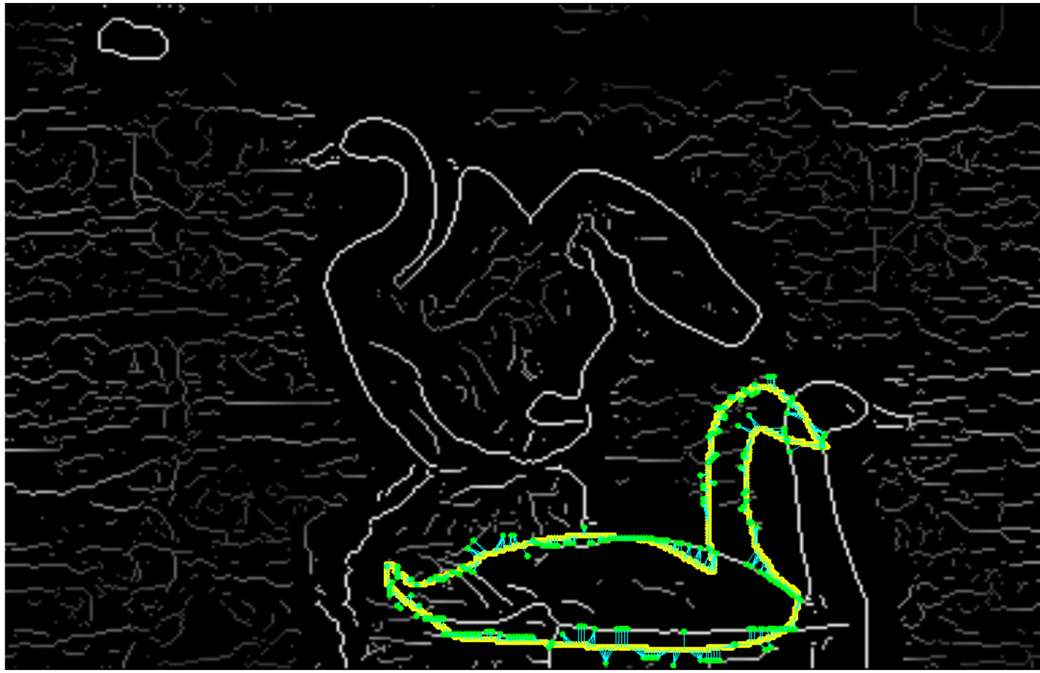
Rotation



Scale



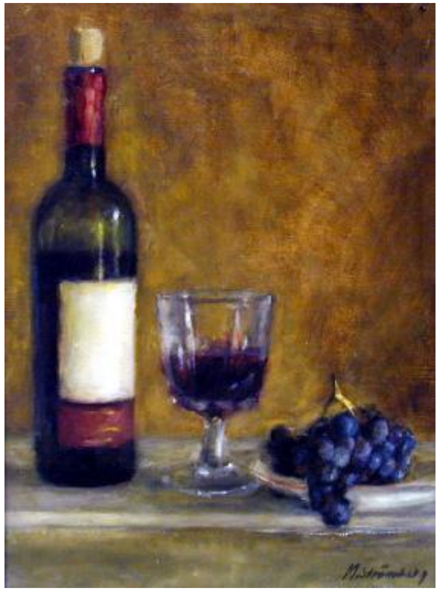
Aspect ratio



Bad edge map threshold / Clutter

Some Alternatives

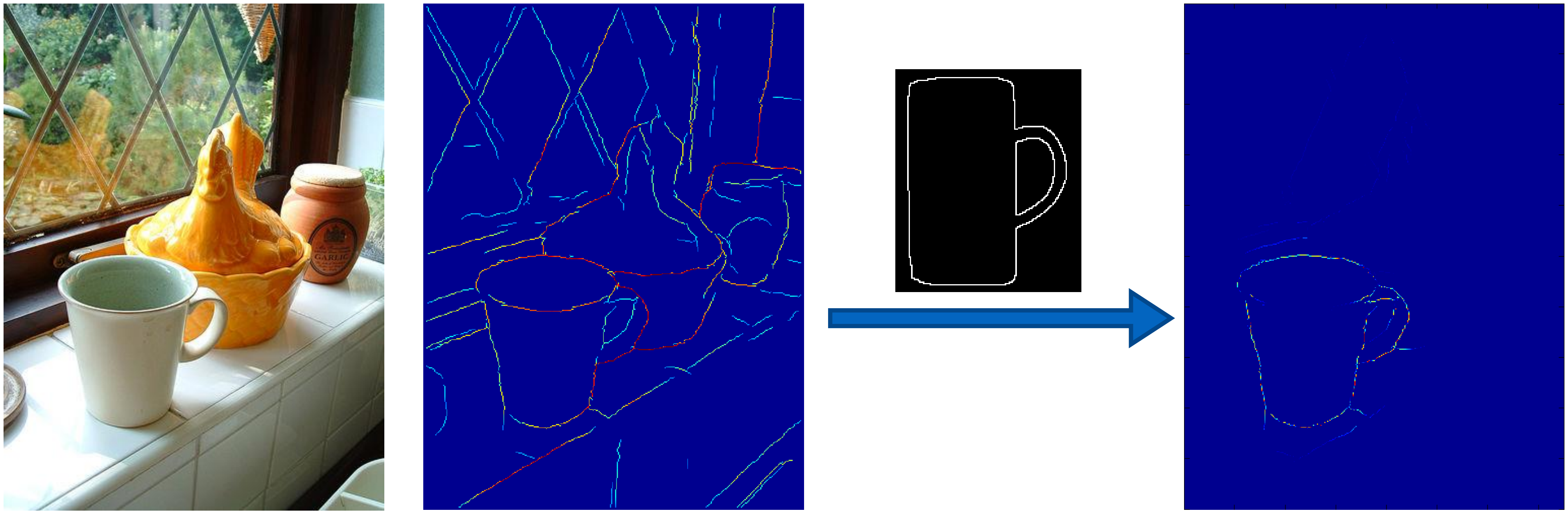
Each edge pixel may have an "edgeness" score instead of a binary value to avoid bad thresholding.

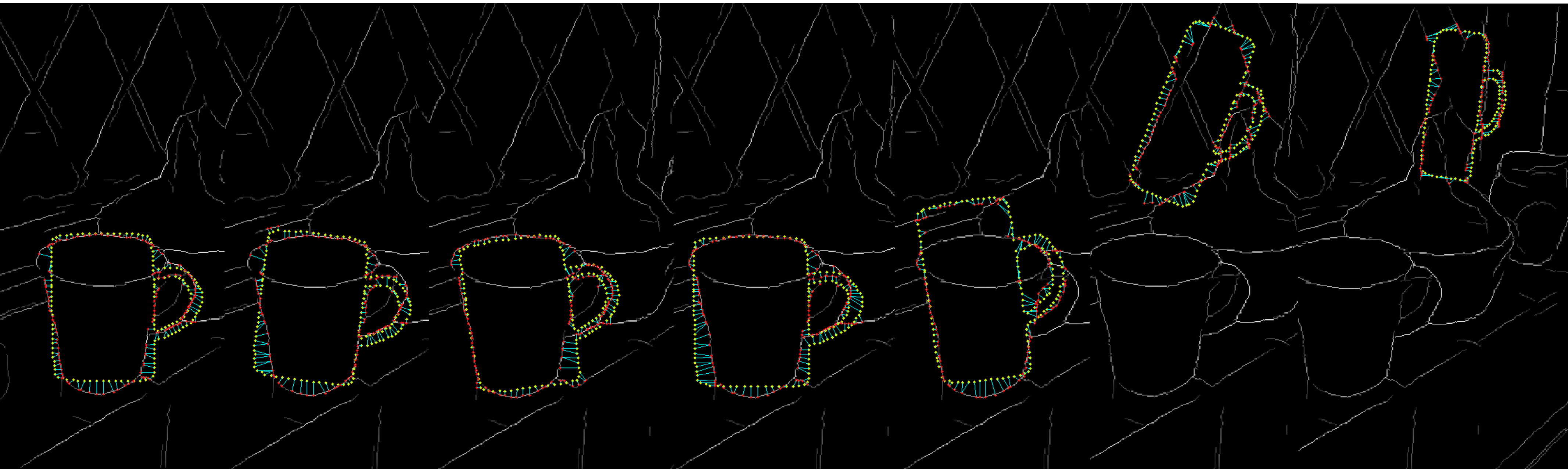


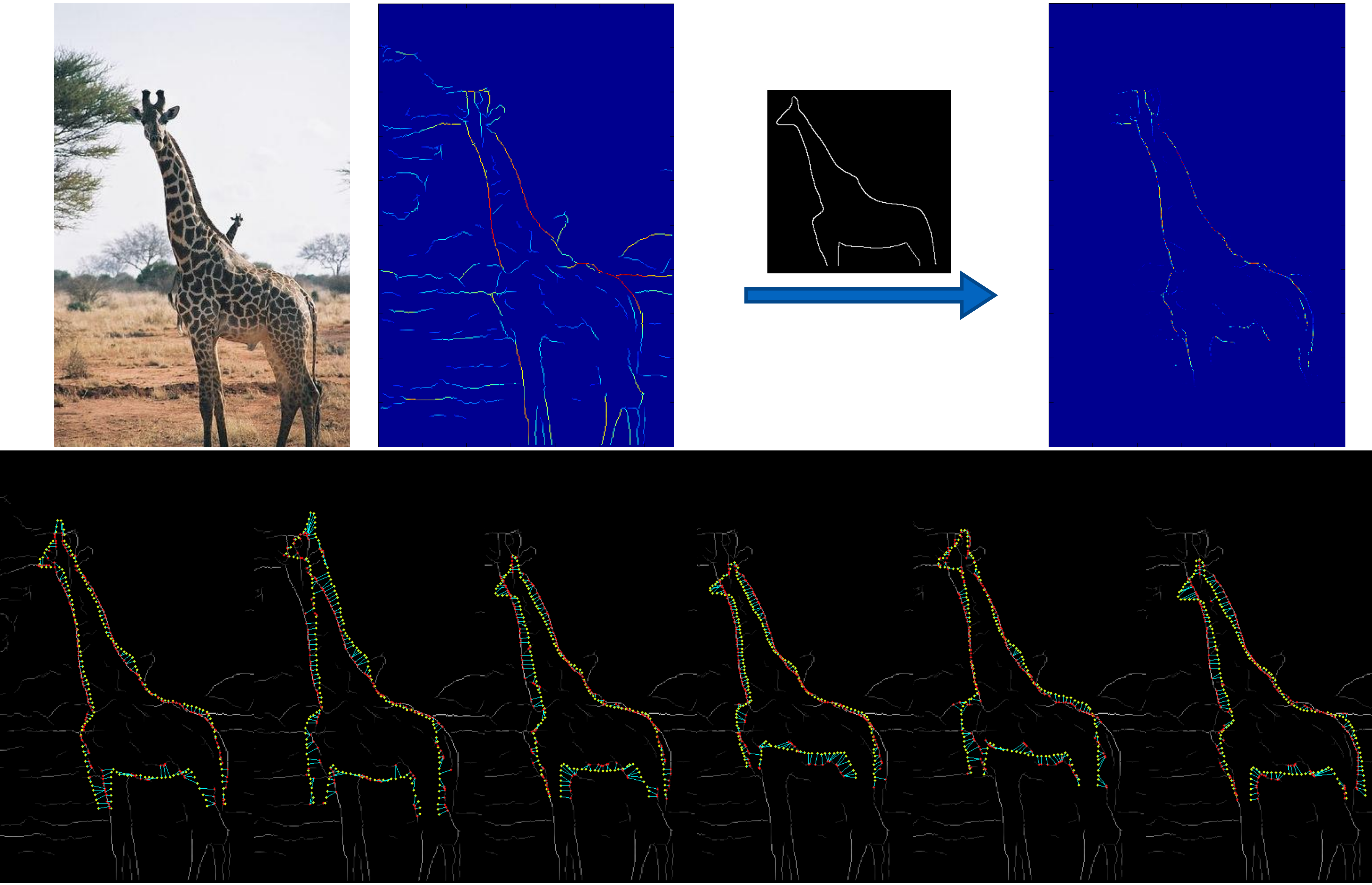
$$c(p) = \min_{q:f(q)>0}\left(\left(\frac{1}{f(q)^2} - 1\right) + \|p - q\| + \lambda|\varphi(p) - \varphi(q)|\right)$$

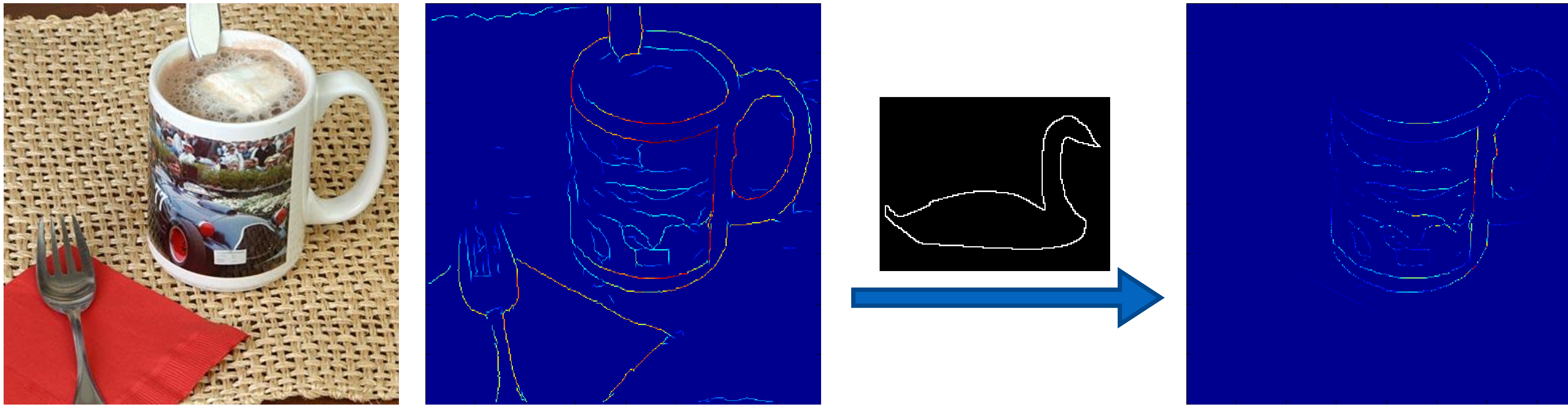
Where *f(q)* is the "edgeness" of pixel q, and *f(q)* is in [0,1]. Distance transform still applies.

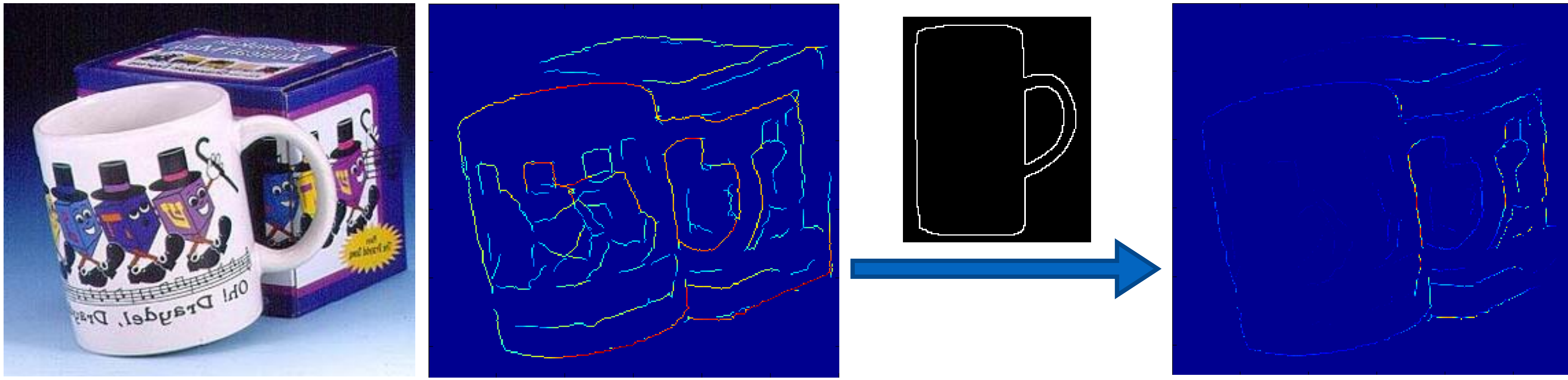
Voting from low cost matches:
Each hypothesis votes for edge pixels in the query image that participates in the match.









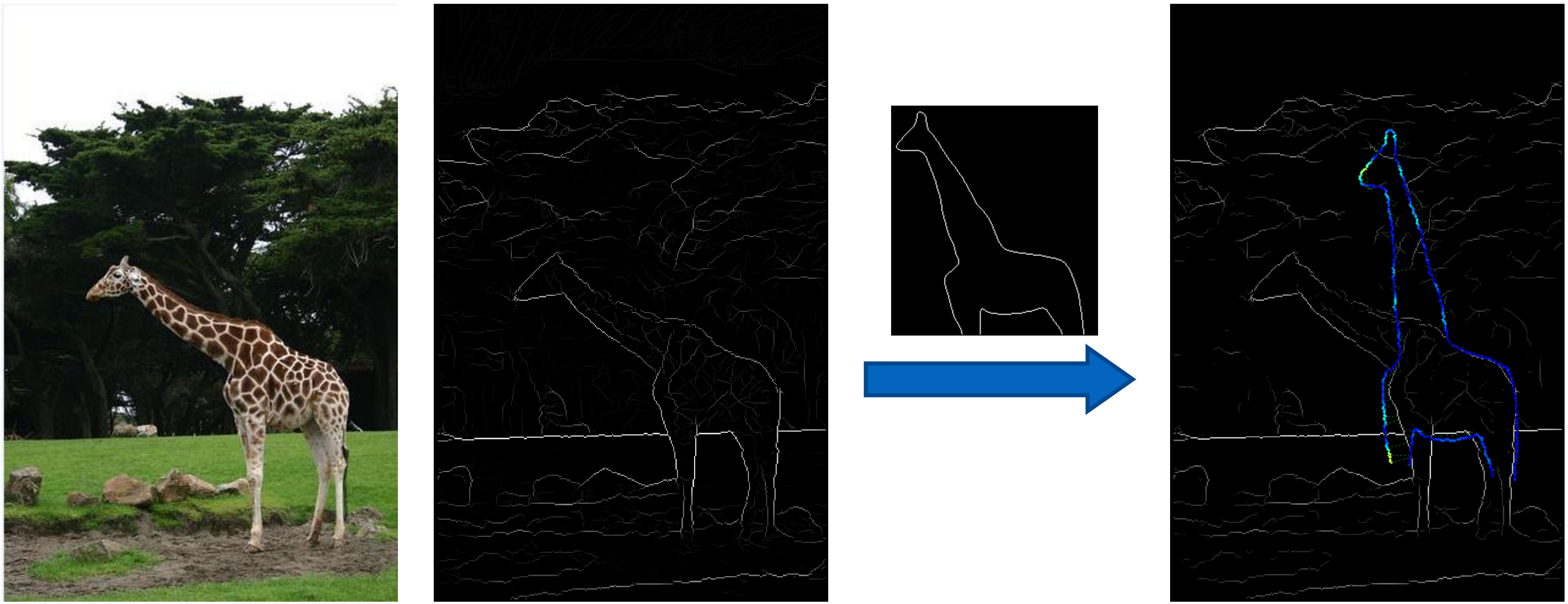


What results in high chance of accidental alignment?

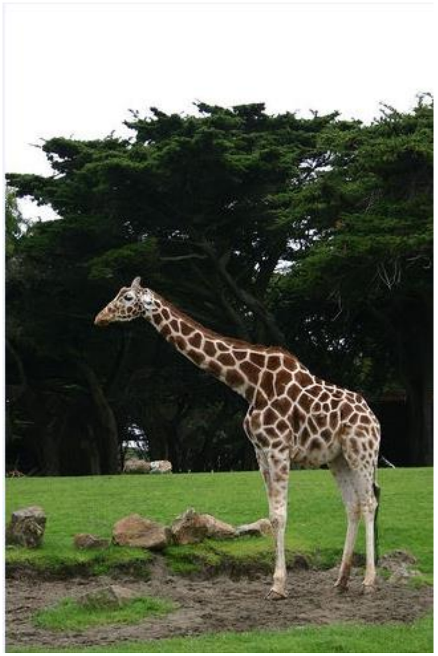
How is chamfer matching different from other shape methods we introduced?

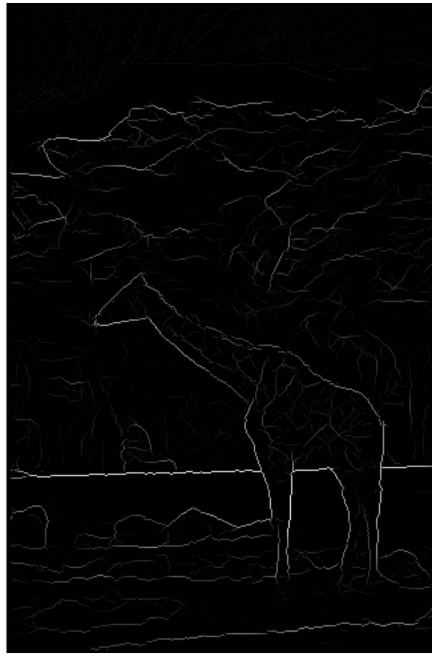
Deformable Shape

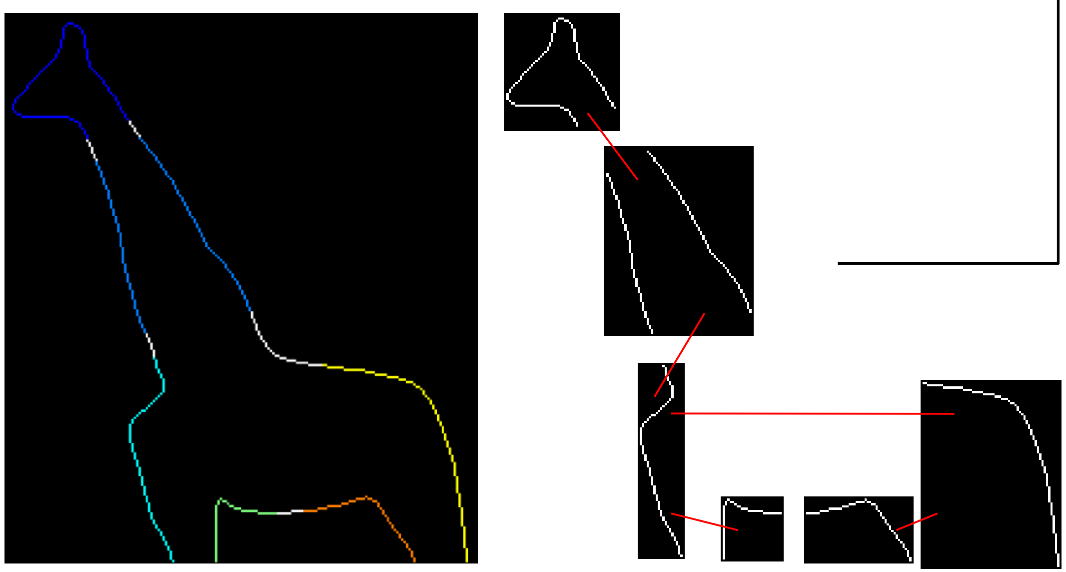
Applying chamfer matching directly

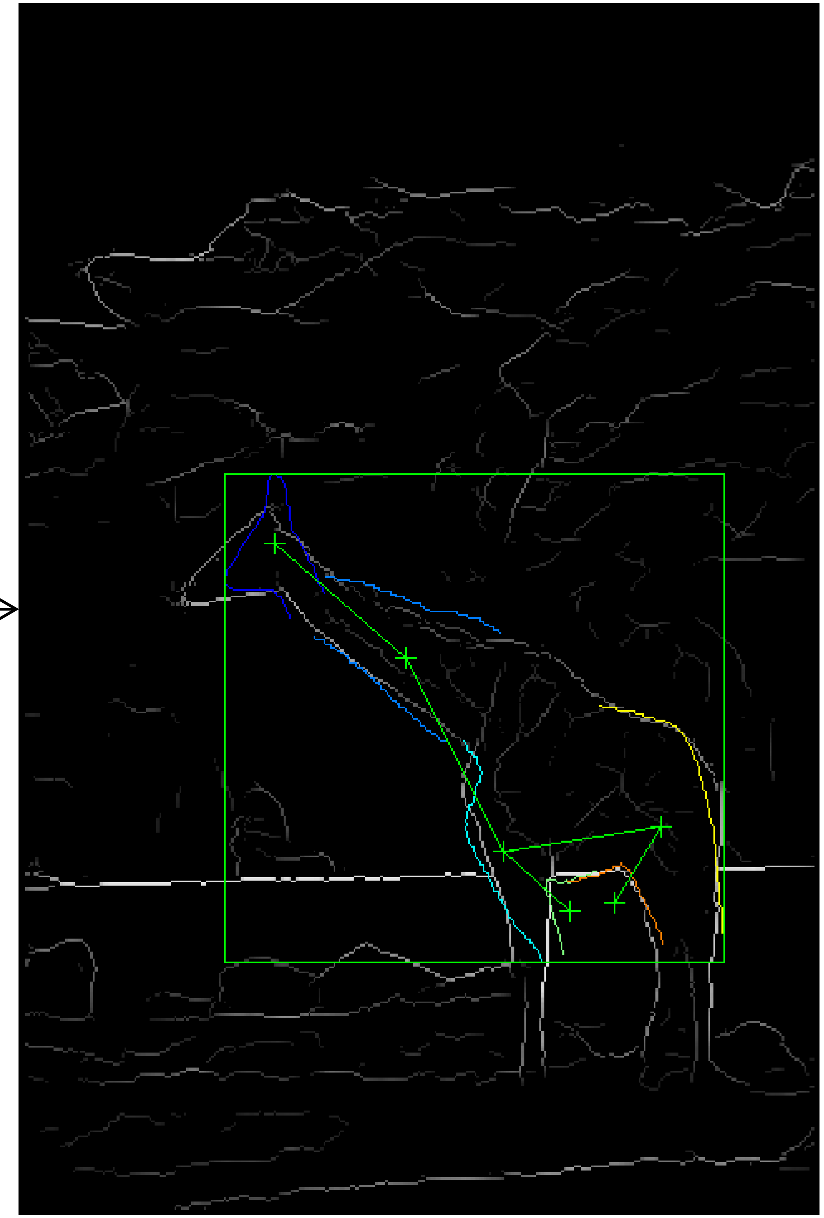


Deformable part model detection with 6
arts



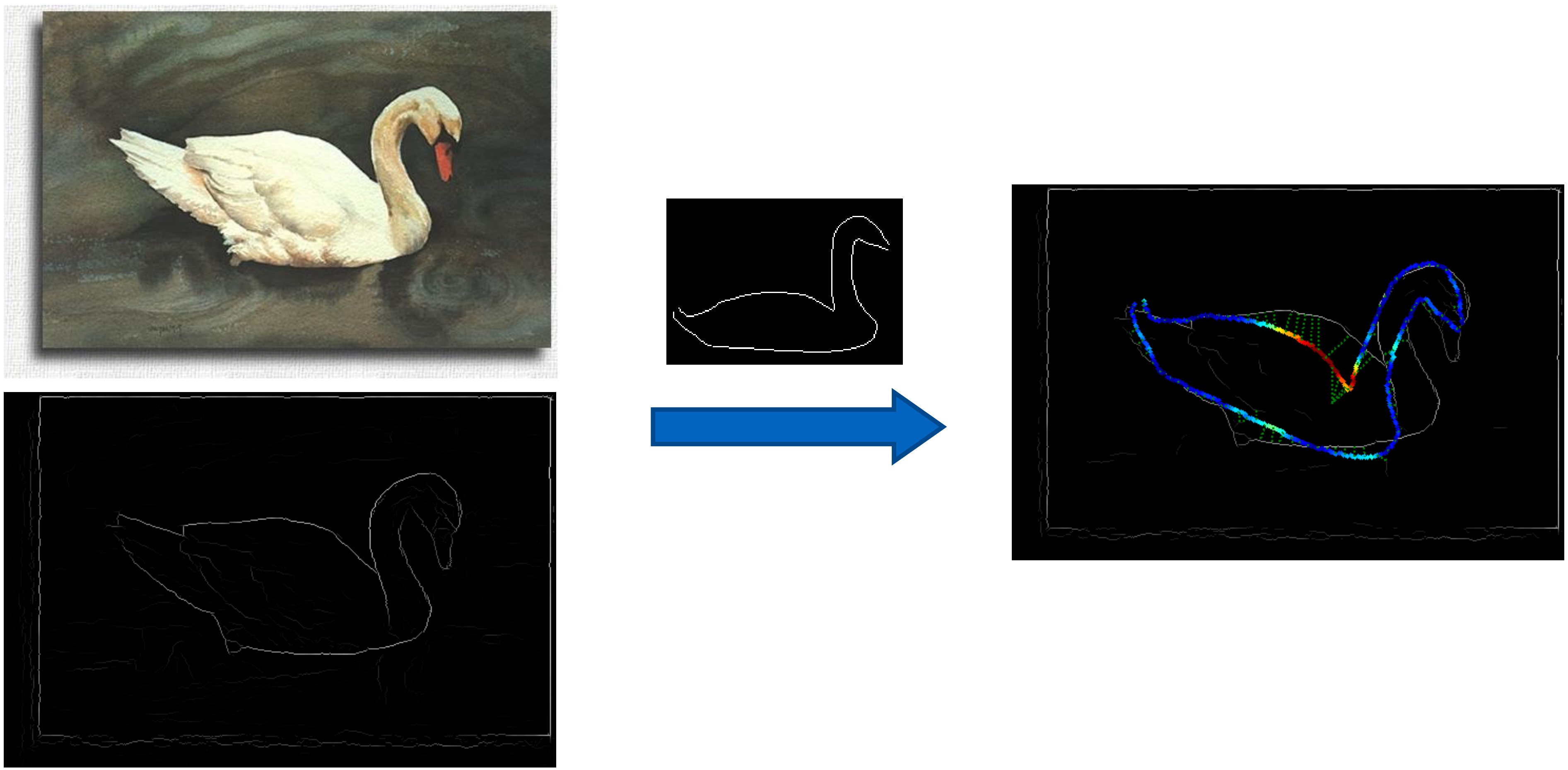




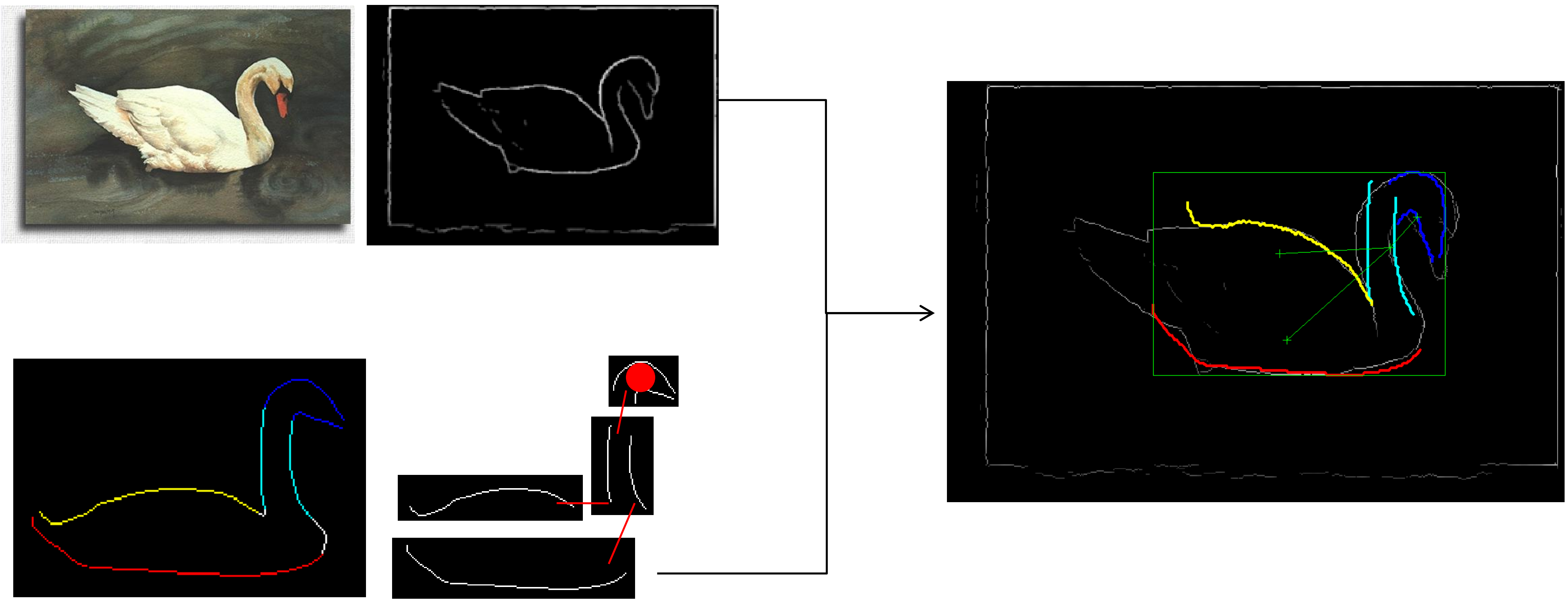


mod

Applying chamfer matching directly



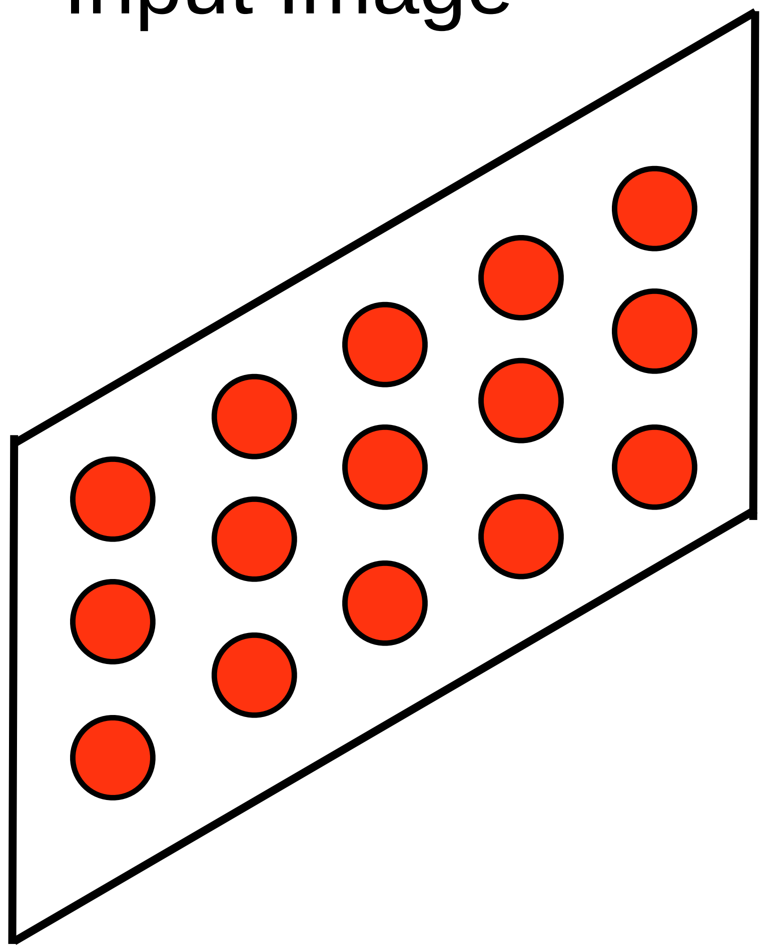
Deformable part model detection with 4 parts



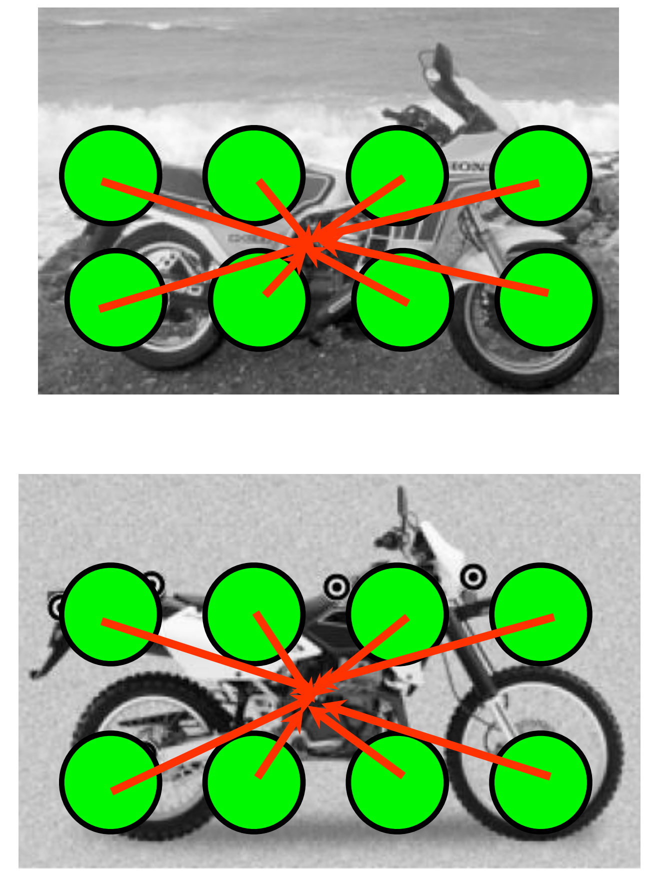
model

Voting based shape detection
Simplified

Input Image

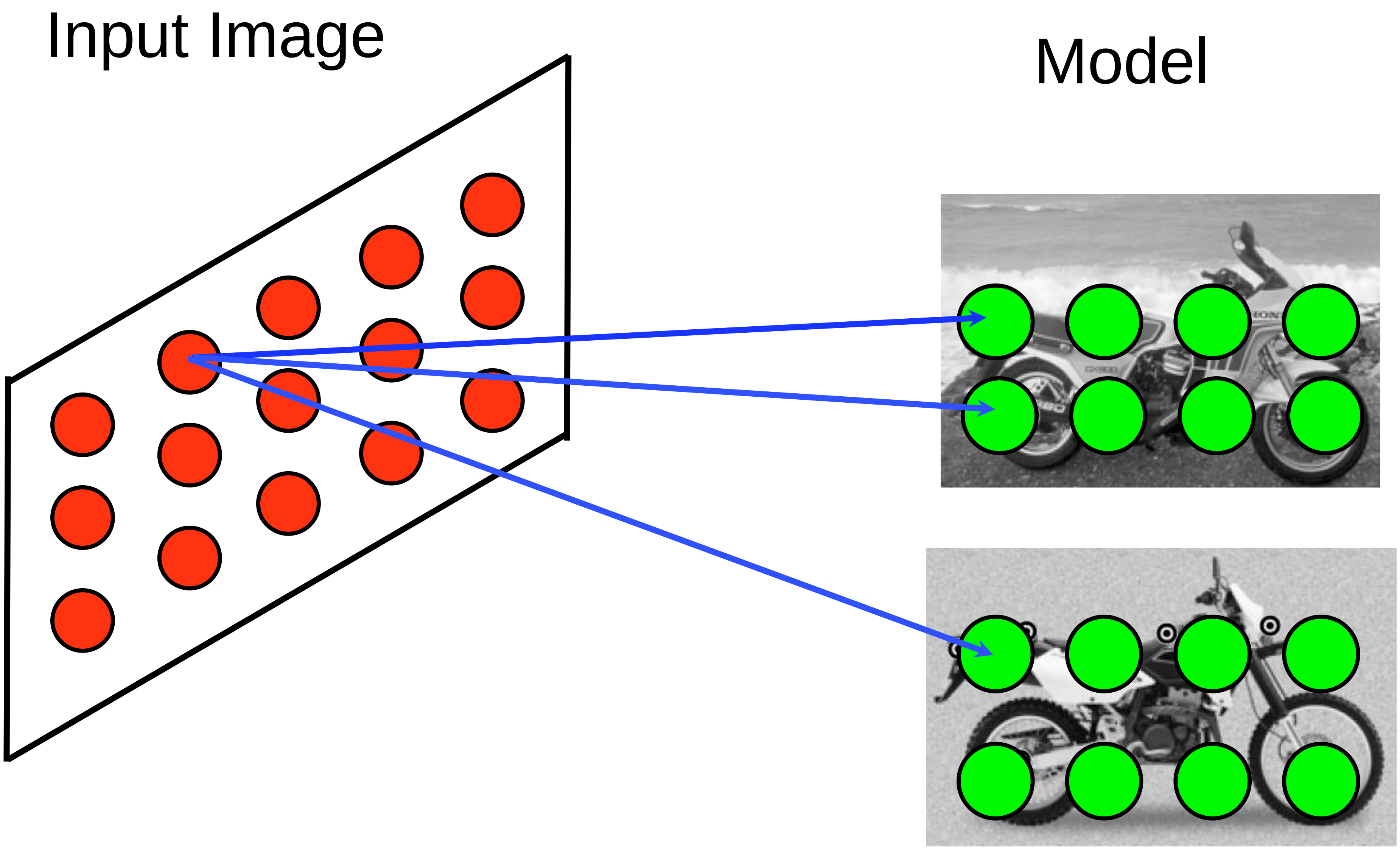


Model

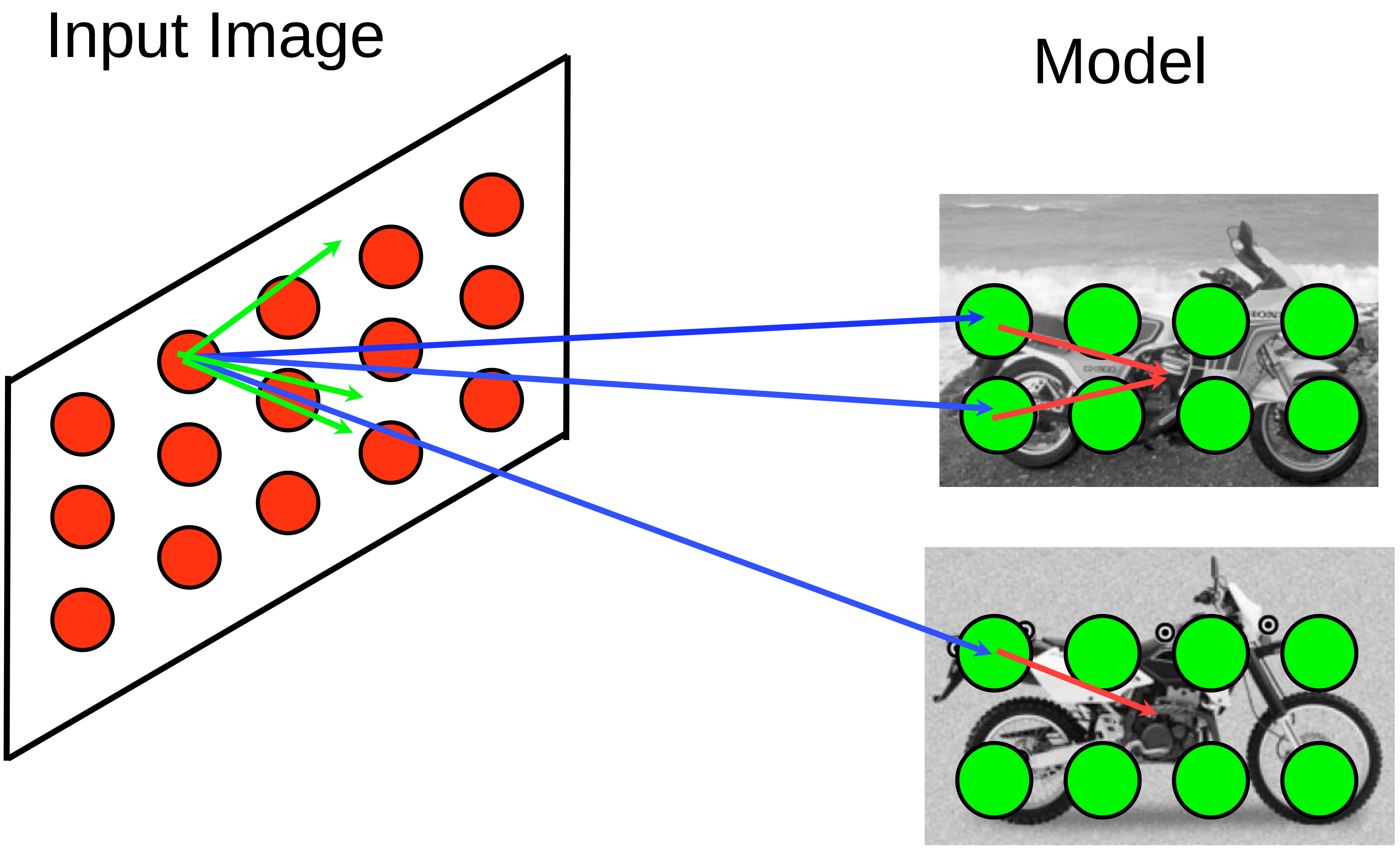


Construct a code book for each model points: (green) nodes

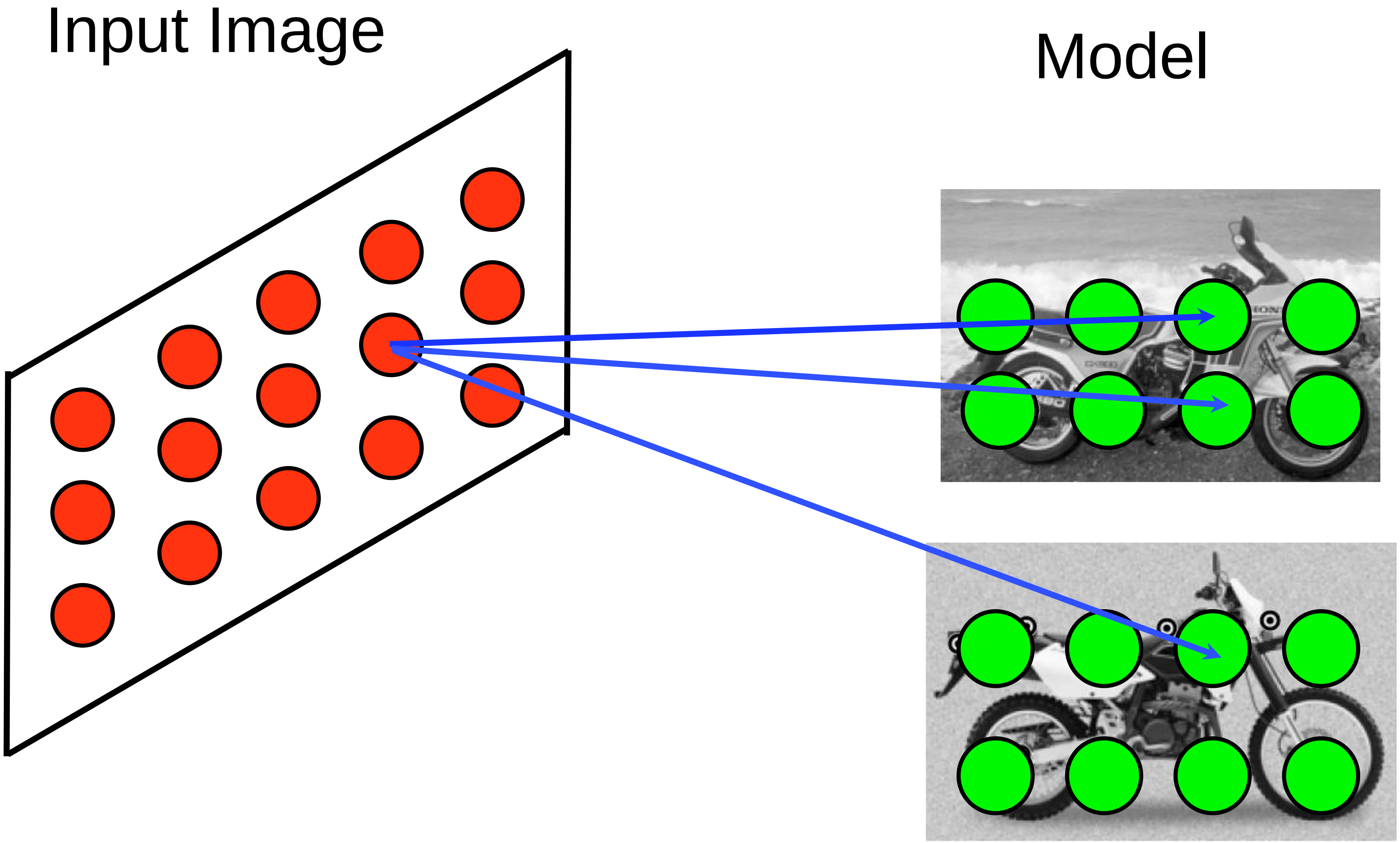
Code: = (Hog or Shape Context + offset to center)



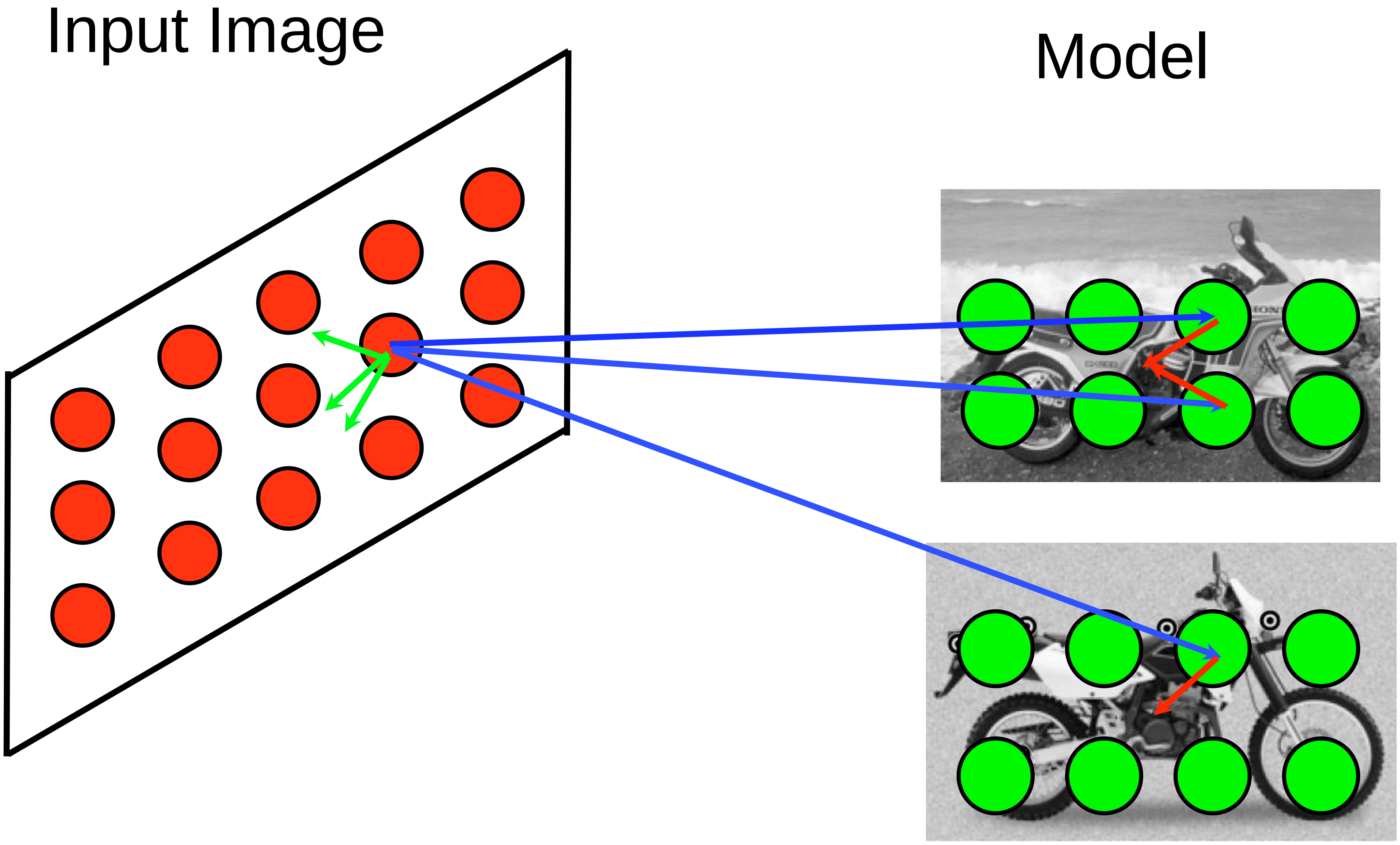
scan over image points, find the top k matches in model



Create vote map in Input image, based on the top k matches in model



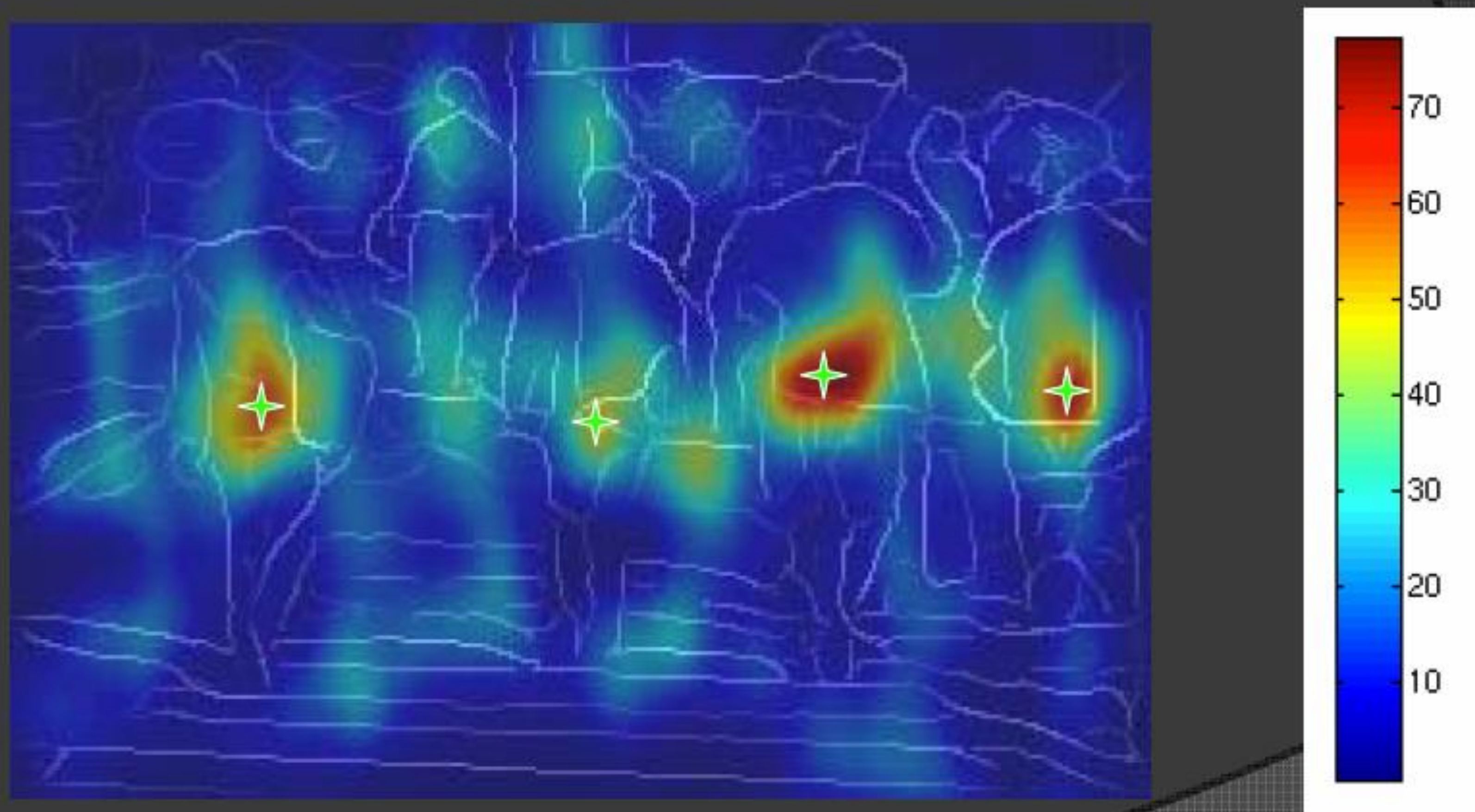
scan over image points, find the top k matches in model



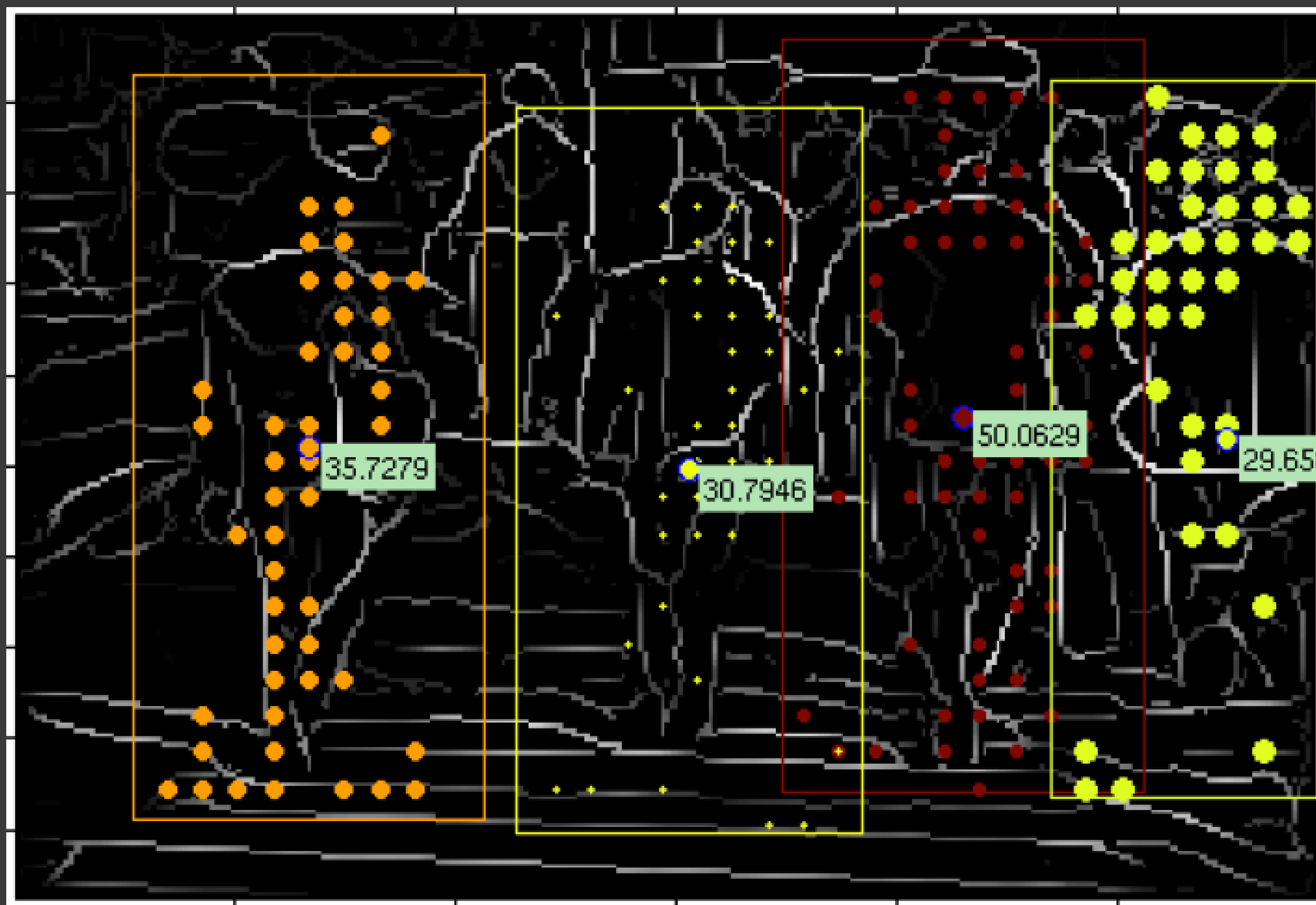
1) Create vote map in Input image, based on the top k matches in model

2) Summing up the map

Using this 'score map', we can choose hypotheses centers on it (green stars).

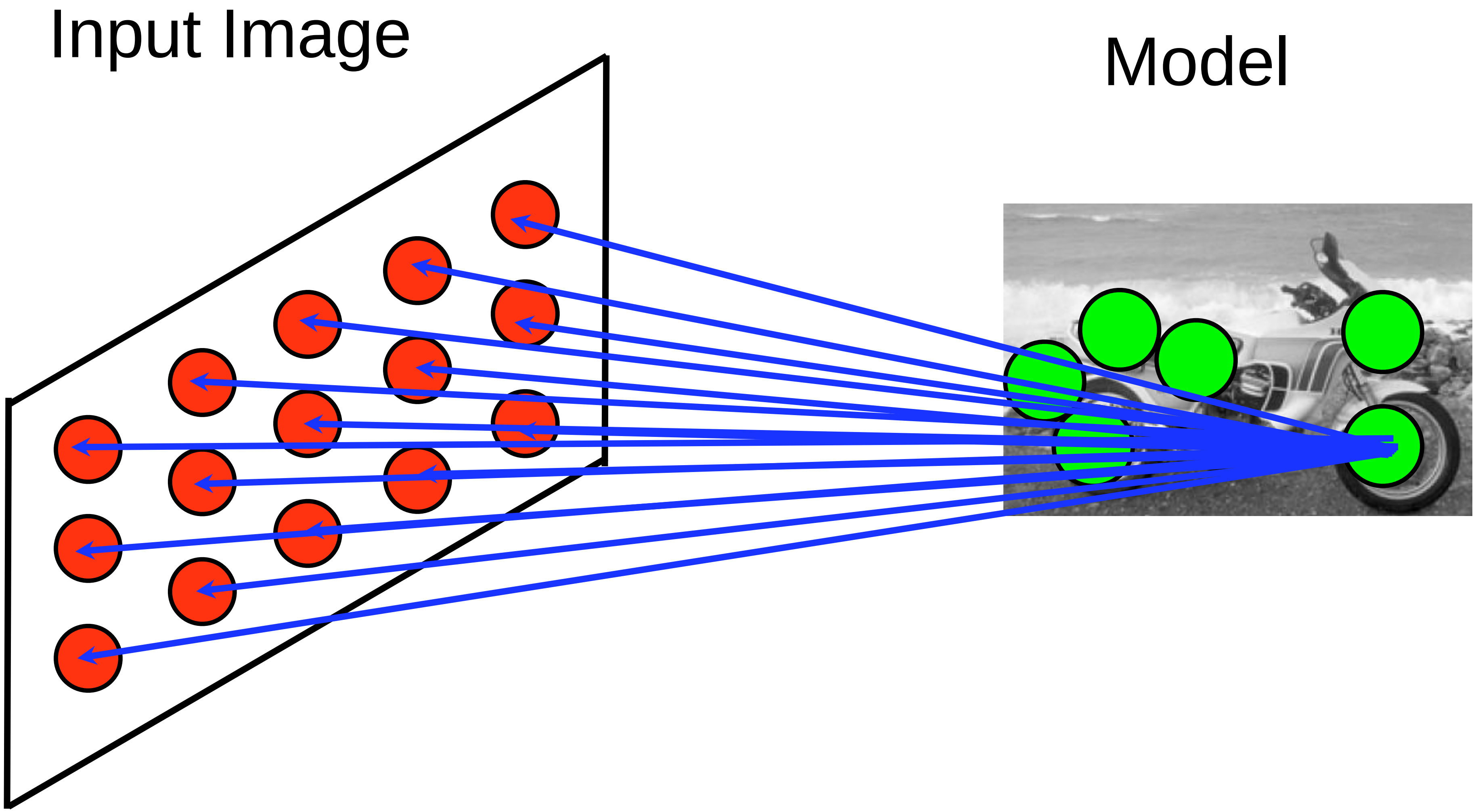


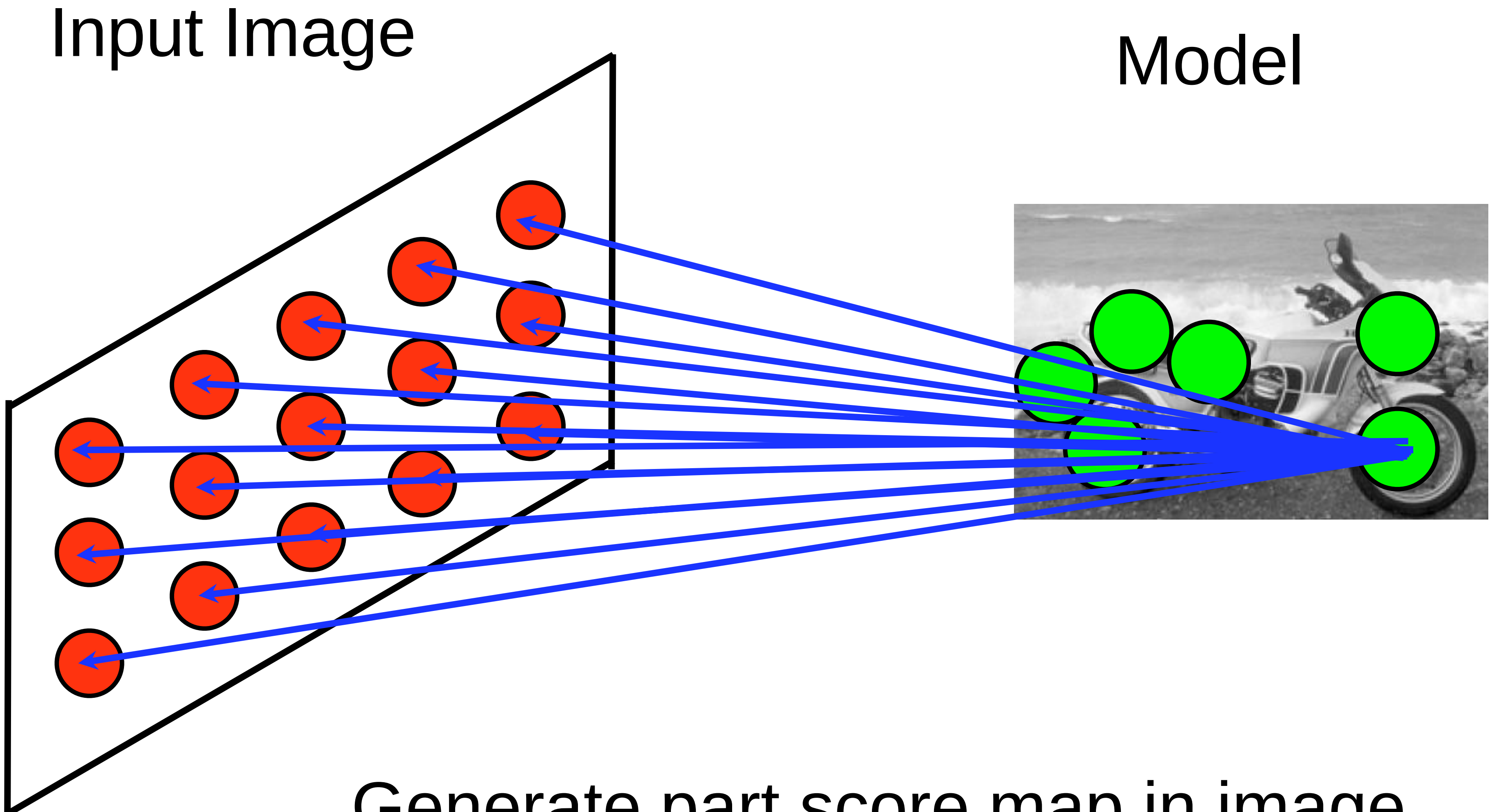
For each hypothesis position, trace back to find its voters.



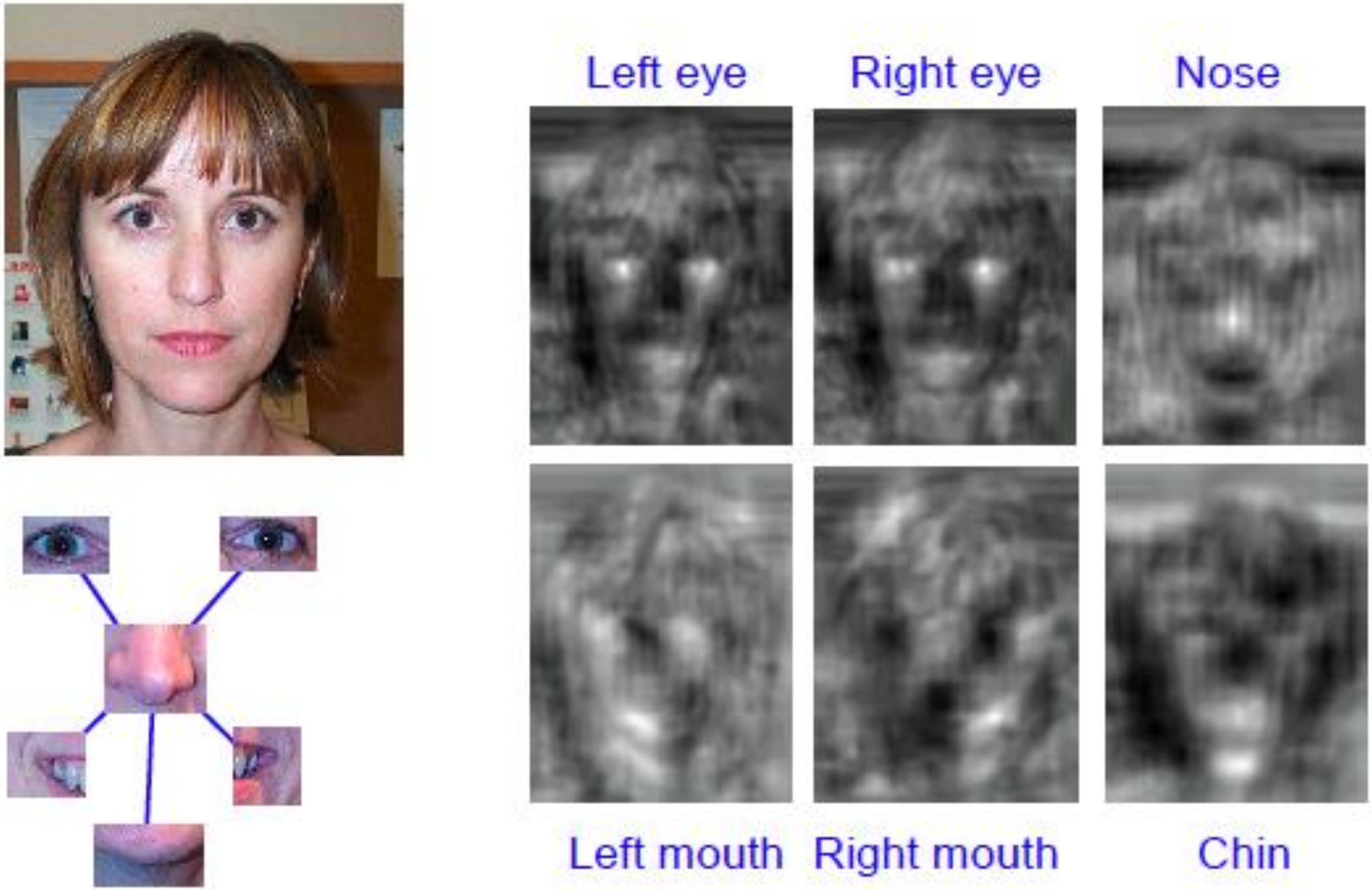
Note: the numbers inside the rectangles are scores for each hypothesis after enforce one-2-one match, so they are a bit lower than voting scores.

Pictorial Structure Simplified

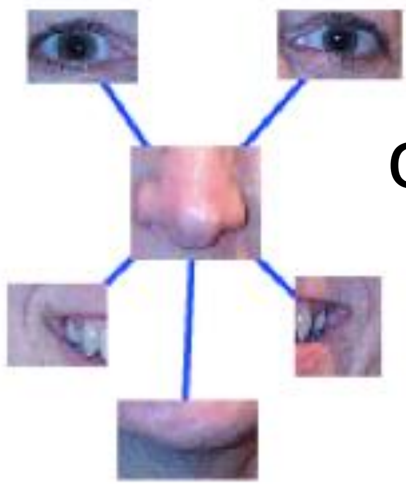

Input Image
Model



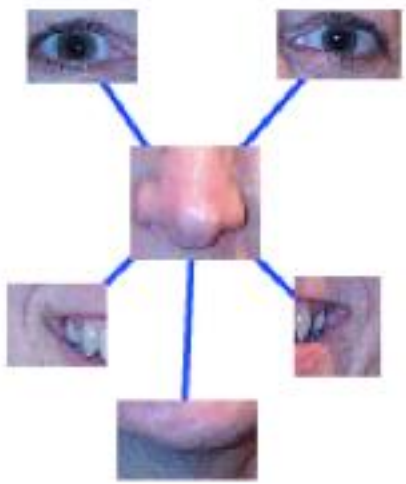
Generate part score map in image



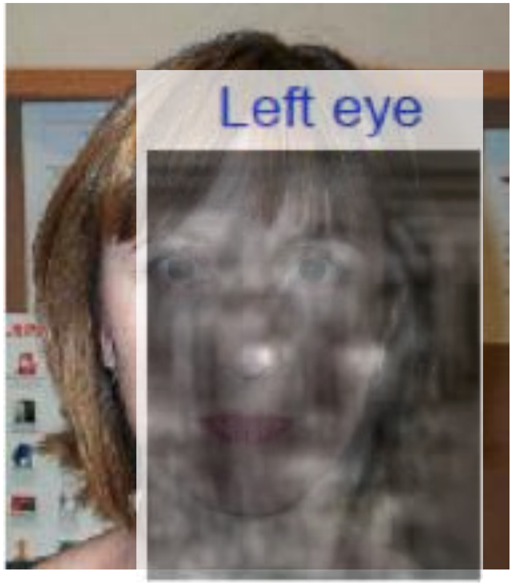
Combine multiple part score function into one score map



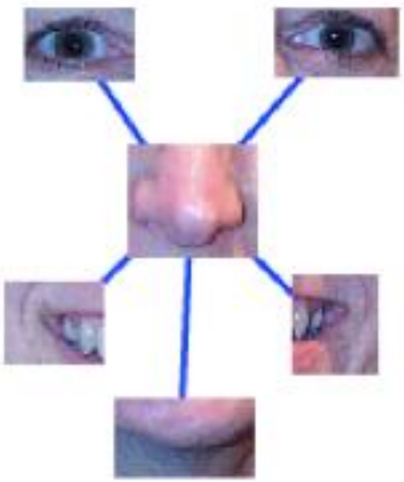
construct a ‘star’ graph, with parts as node
pick one node as “root”

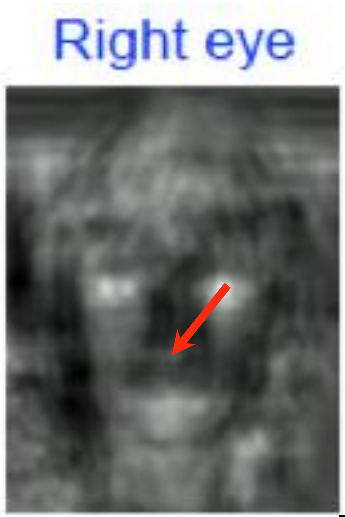


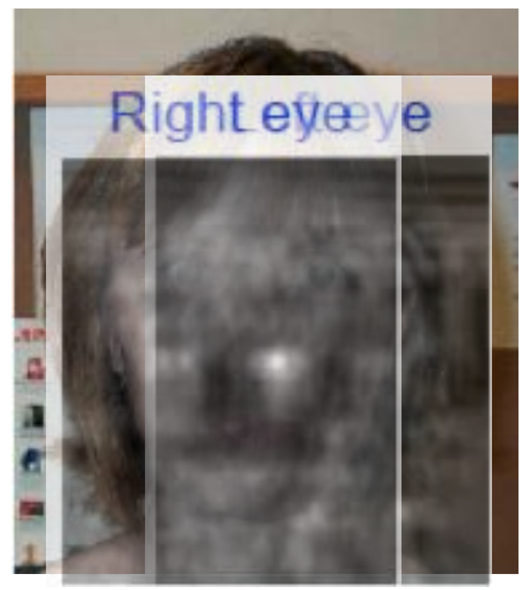
For each non-root node:



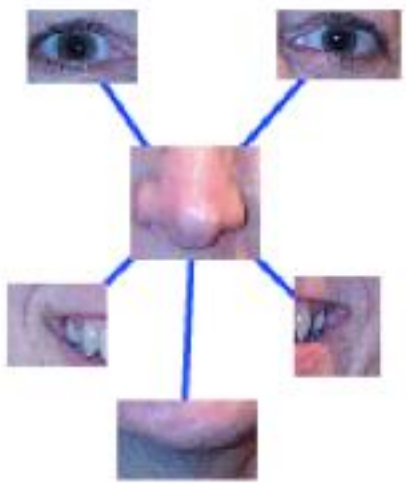
Shift score map for Left eye onto center(nose)

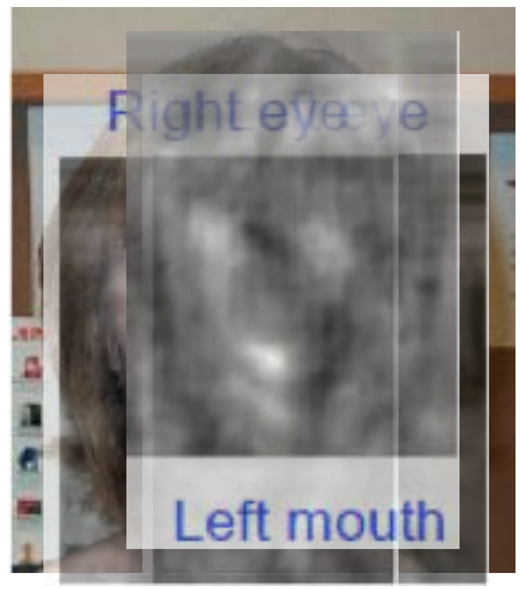




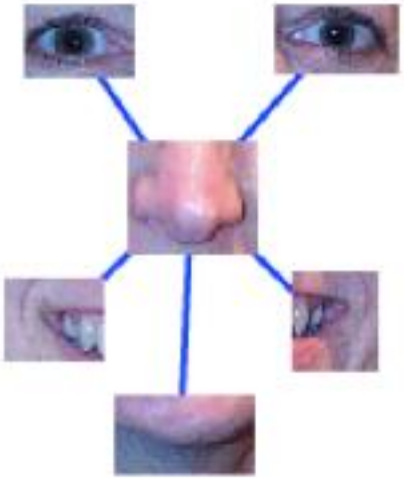


Shift score map for Right eye onto center(nose)

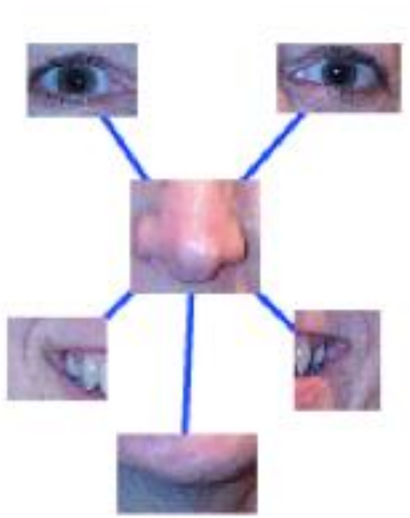


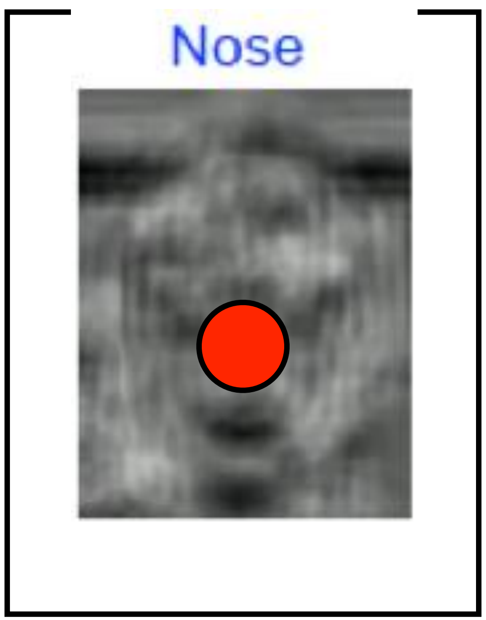


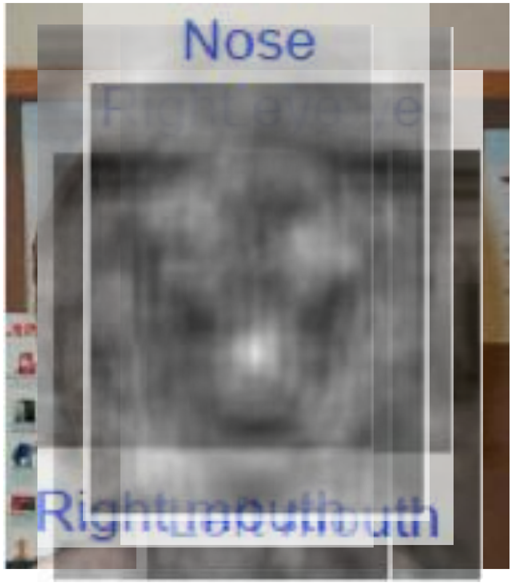
Shift score map for Left mouth onto center(nose)



Shift score map for Right mouth onto center(nose)







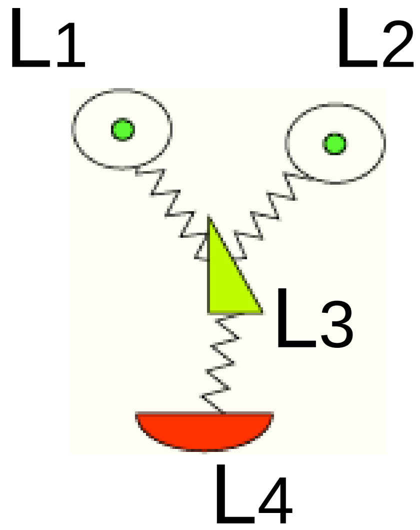
Add up all the part vote score maps

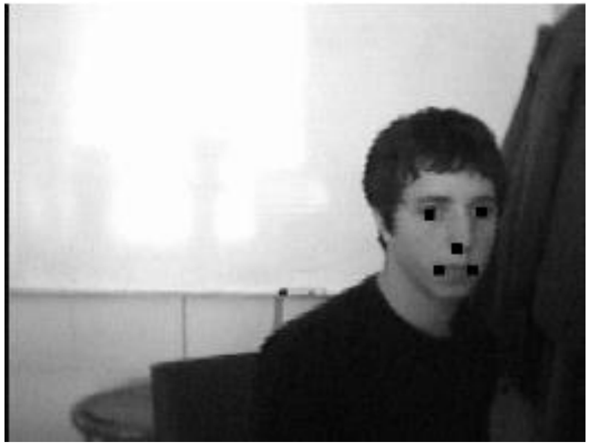
Object Representation

Pictorial Structure

Object Representation

- Object with n parts labeled 1 through n





- Object configuration given by: $L = (l_1, \ldots, l_n)$
  - Location of each part

    $(L_1, L_2, L_3, L_4) = ((300,200), (300,250), (330,230), (360,230))$

Find the most probable configuration of the object,

$$P(L|I) \propto \underbrace{P(I|L)}_{\text{Appearance}}\underbrace{P(L)}_{\text{Geometry}}$$

Geometrical model: $P(L)$

Appearance model: $P(I|L) \propto \prod g_i(I, l_i)$

Part-based Object Representation

Geometrical model: $P(L)$

measuring "goodness" of the part configuration

Appearance model: $P(I|L) \propto \prod g_i(I, l_i)$

image Label

measuring "goodness" of the part appearance

Part-based Object Representation

Find the most probable configuration of the object,

$$P(L|I) \propto \underbrace{P(I|L)}_{\text{Appearance}}\underbrace{P(L)}_{\text{Geometry}}$$

- Size of configuration space is exponential
  - n parts, m locations - $O(m^n)$ configurations
  - Use implicit search techniques

S o l u t i o n

1) Reduce number of possible feature locations, by feature detection.

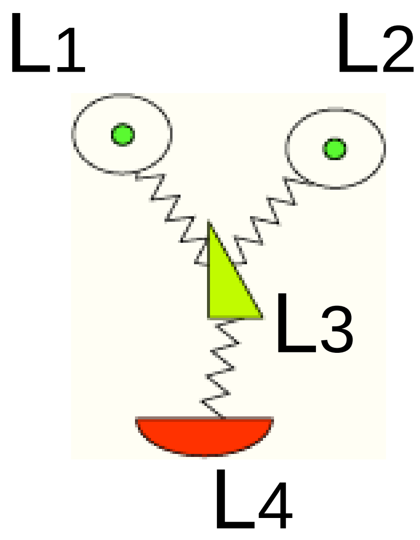
-- a possible solution is use shape context features

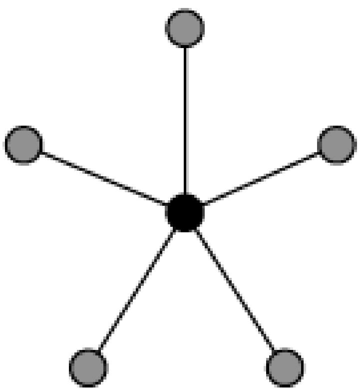
2) Find efficient way of dealing large number of features, each of which has a goodness measure

-- we will cover this story here...

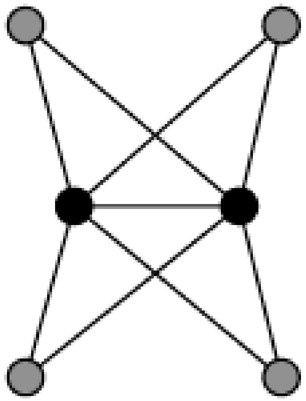
Geometrical model: $P(L)$

measuring “goodness” of the part configuration





1-fan



2-fan

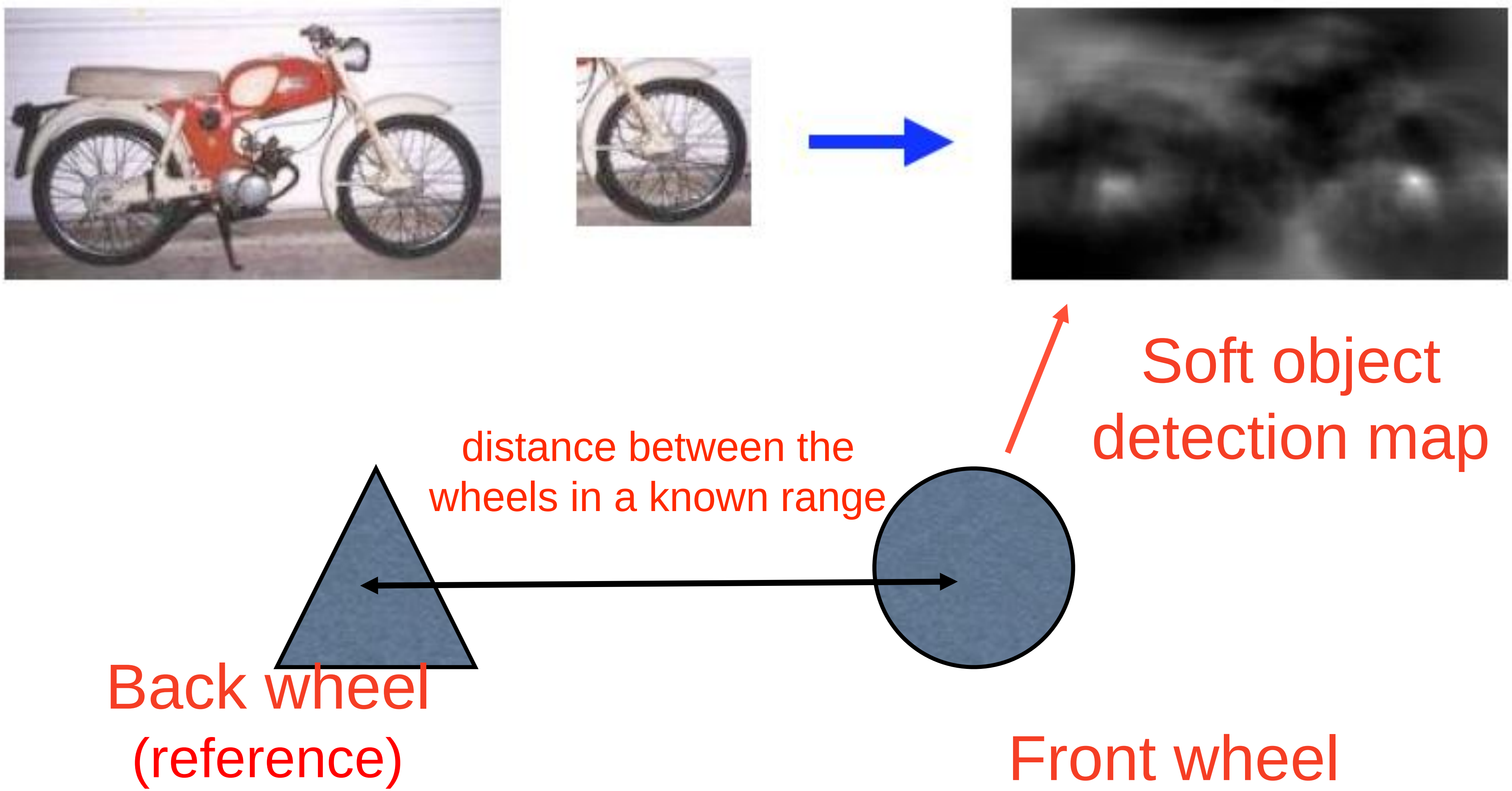
Simplifying “goodness” measure using k-fan model

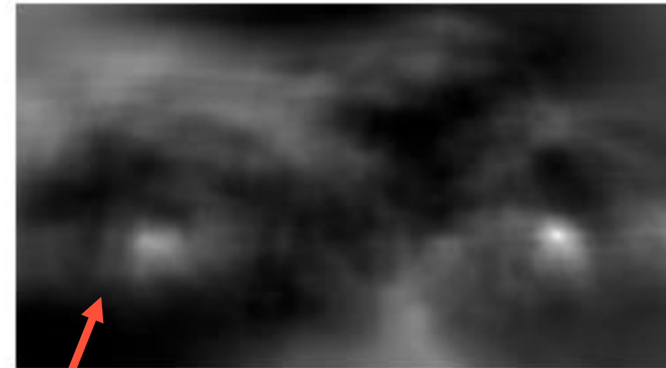
1) we only check if the parts configuration between the reference node(nose in this case), with all other nodes

Dealing with "soft" features

- Recognition without feature detection
  - Single overall inference problem
  - Parts have a match quality at each location

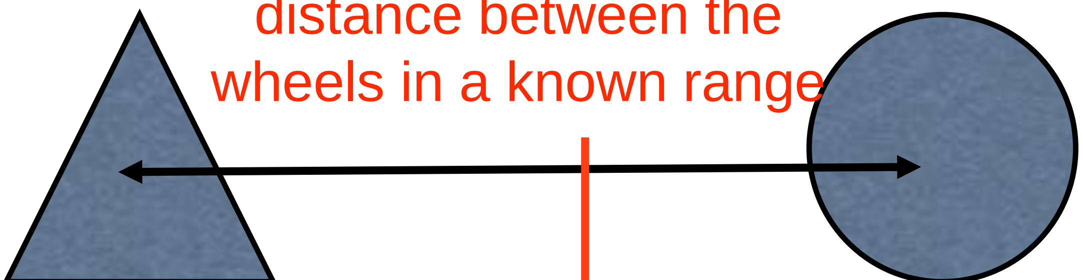
A simplified object of two parts (front & back wheel)





Soft object detection map

distance between the wheels in a known range



p: location of back wheel

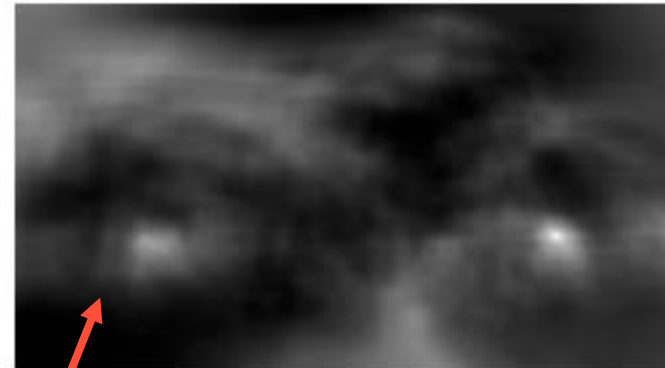
q: location of front wheel

$$L(p,q|I) = f(p) + h(p-q) + f(q)$$

Soft part detection measure for p

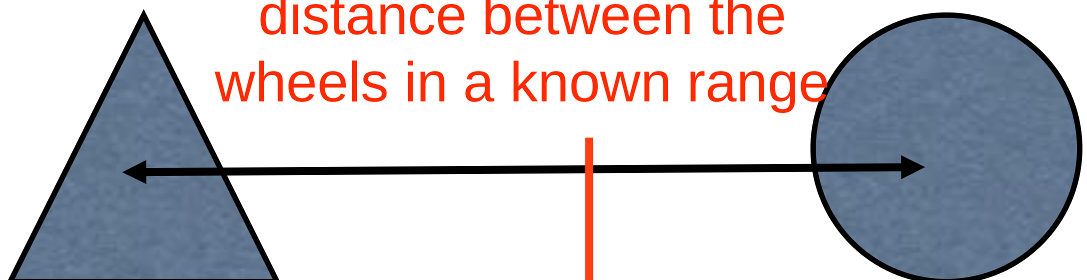
configuration goodness (p,q)

Soft part detection measure for q



Soft object detection map

distance between the wheels in a known range



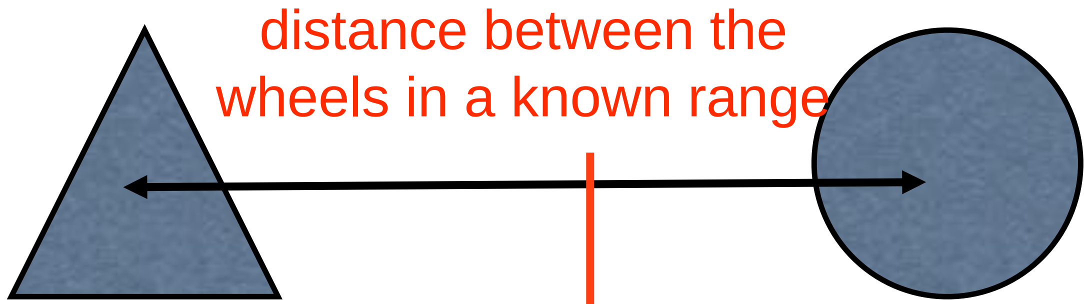
p: location of back wheel

q: location of front wheel

$$L(p,q|I) = f(p) + h(p-q) + f(q)$$

In this case p, q each has n (image size=1million) possible locations, L(p,q| I) has n^2 (Trillion) possible solutions

fast solution is needed!



p: location of back wheel

q: location of front wheel

$$L(p,q|I) = f(p) + h(p-q) + f(q)$$

$$\min_{(p,q)} L(p,q|I) = \min_{(p,q)} f(p) + h(p-q) + f(q)$$

$$= \min_{(p)} \left(f(p) + \underline{\min_{(q)} \left(f(q) + h(p-q) \right)} \right)$$

$D_q(p)$: generalized distance transform

This can be computed in linear time!

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.

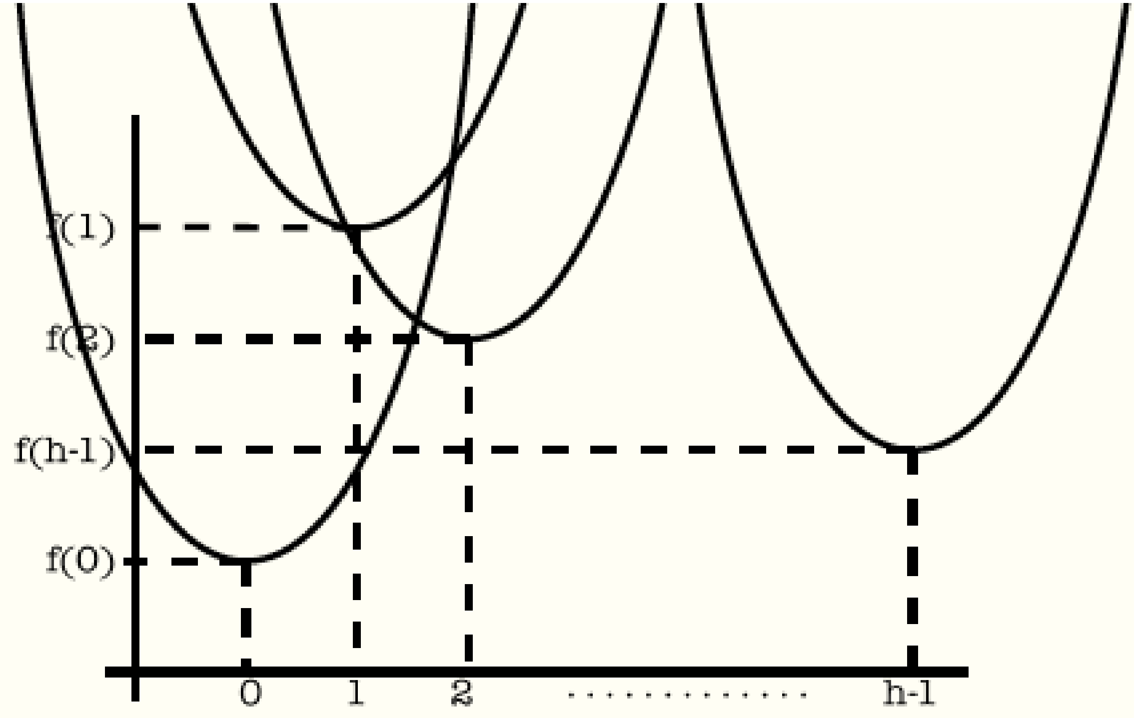
$$f(p) = \begin{cases} 0 & \text{if } p \in P \\ \infty & \text{otherwise} \end{cases}.$$

http://www.cs.cornell.edu/~dph/papers/dt.pdf

1D case: $\mathcal{D}_f(q) = \min_{p \in \mathcal{G}} \left((q-p)^2 + f(p) \right)$

For each p, $\mathcal{D}_f(q)$ is below the parabola rooted at $(p, f(p))$.

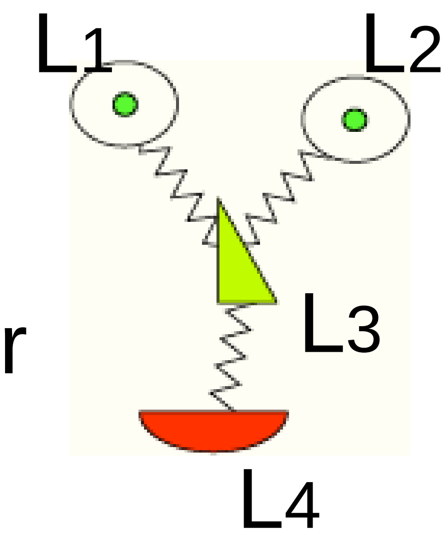
$\mathcal{D}_f(q)$ is defined by the lower envelope of h parabolas.



http://www.cs.cornell.edu/~dph/papers/dt.pdf

There is an efficient exact inference for graph without loop

Procedure:



Step 1, order tree

determine a root of the tree, and order the nodes according to its depth

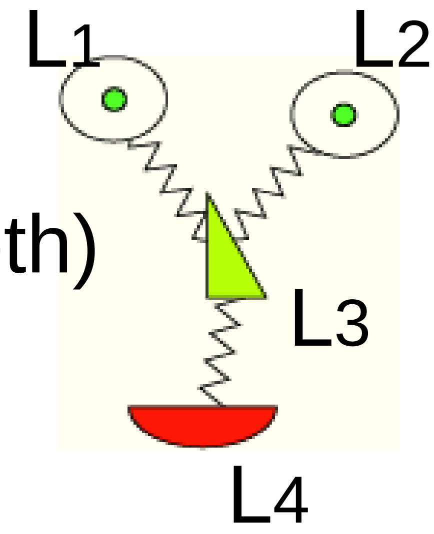
Step 2-3: Gather information.

processing from the bottom of the tree (nodes with max. depth) backward to the root of the tree

Step 4-5: Decide at root, and propagate

Make decision at the tree root, and recursively propagate the information down

Step 2: Gather Information for leaves nodes



or the leaf nodes, j, (nodes with max. depth)

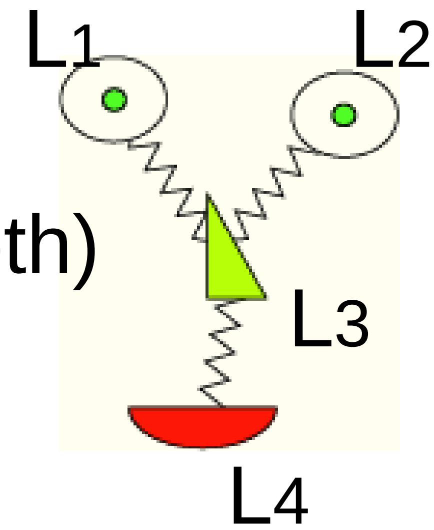
Compute the following table, indexed by its possible parent node assignment:

$$B_j(l_i) = \min_{l_j} \left(m_j(l_j) + d_{ij}(l_i, l_j) \right),$$

parent node label

Given a parent node label, find the best label for itself

Step 2: Gather Information for leaves nodes

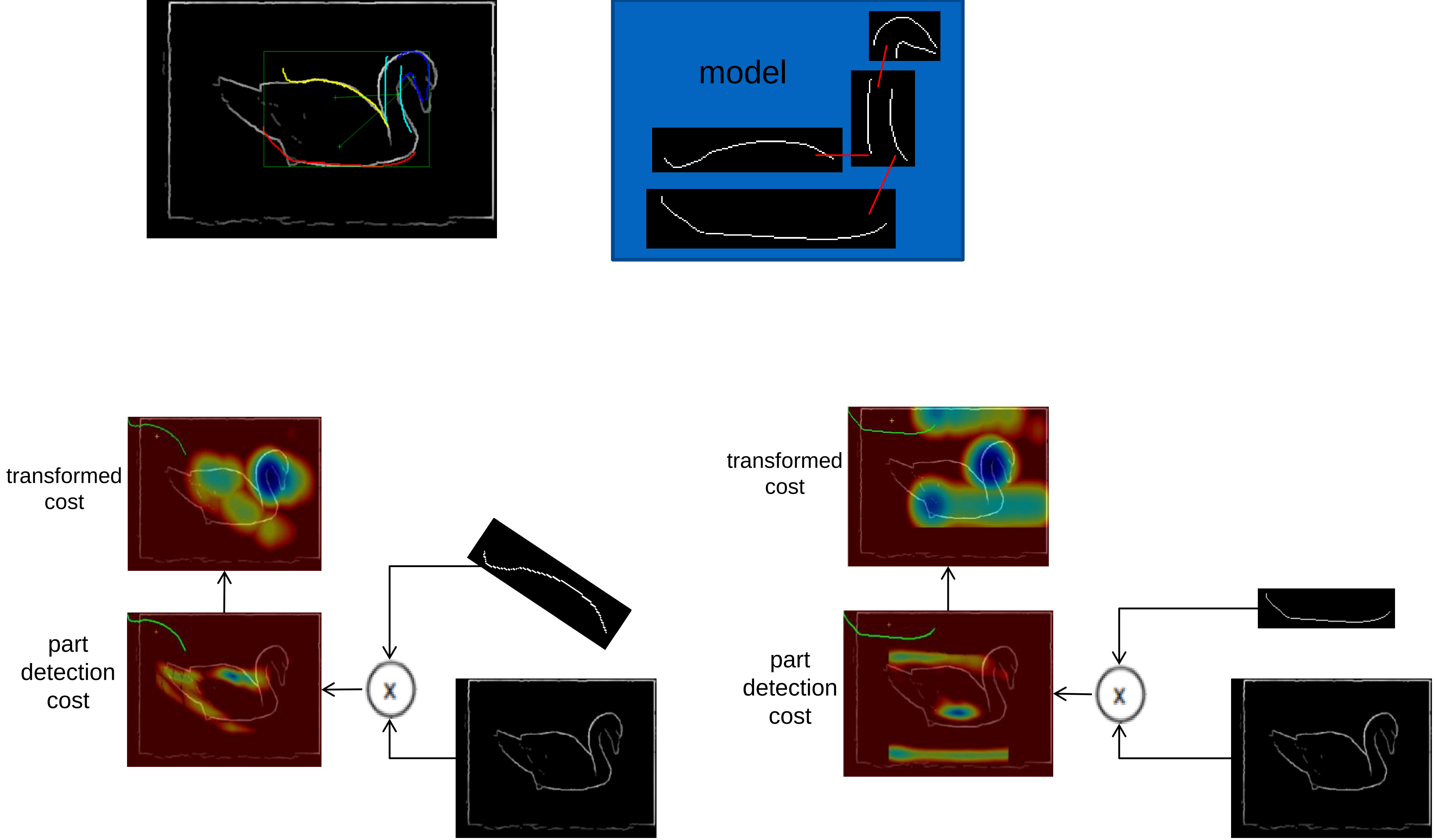


or the leaf nodes, j, (nodes with max. depth)

Compute the following table, indexed by its possible parent node assignment:

$$B_j(l_i) = \min_{l_j} \left(m_j(l_j) + d_{ij}(l_i, l_j) \right),$$

Important: we need to store both the optimal value l_j, as well the cost at the optimal label l_j


model
transformed cost
part detection cost
transformed cost
part detection cost

Step 3: Gather Information at inside node

for inside nodes, j, (not root, not leaves)

Compute the following table, indexed by its possible parent node assignment:

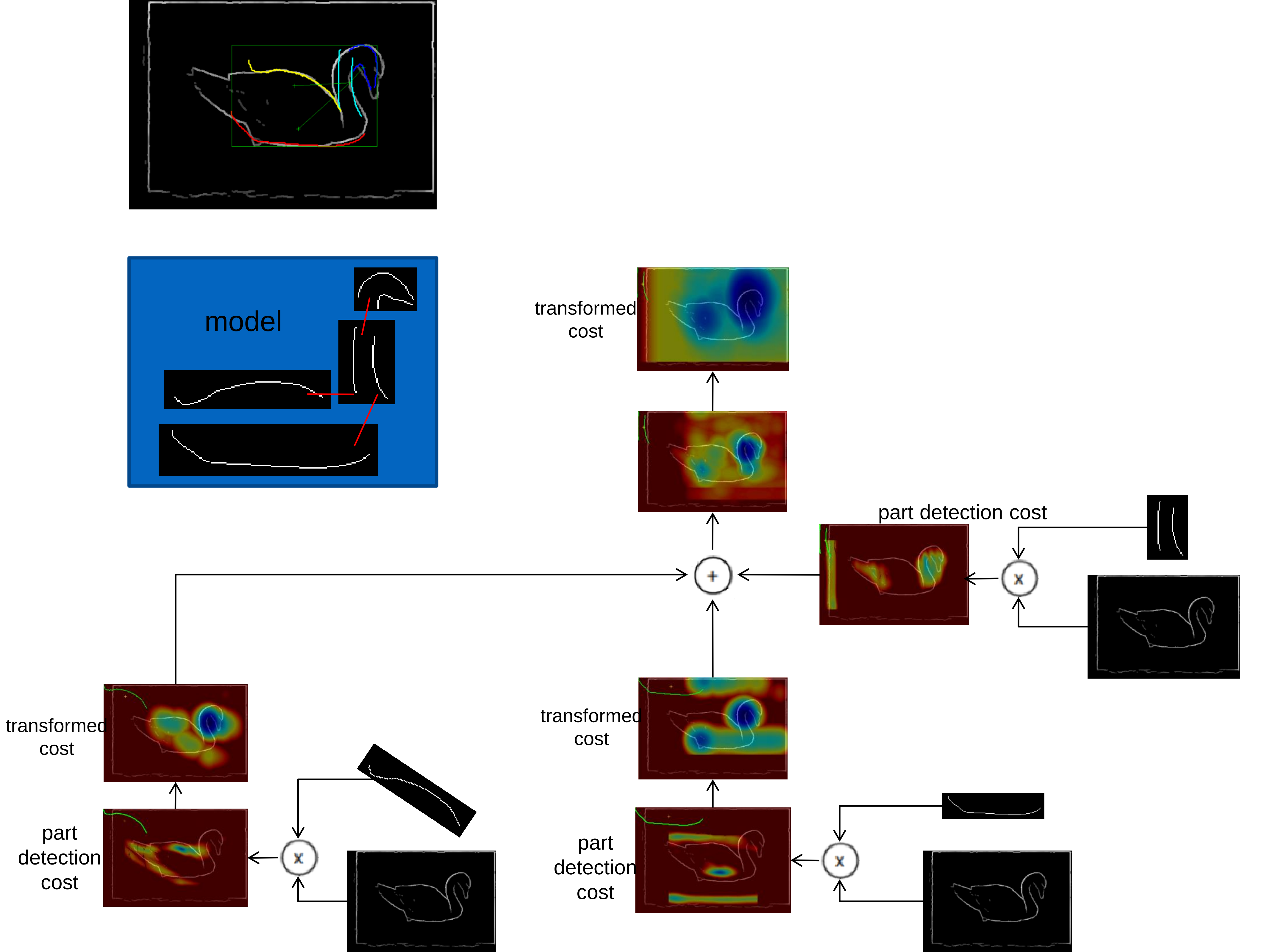
$$B_j(l_i) = \min_{l_j}\left(m_j(l_j) + d_{ij}(l_i, l_j) + \sum_{v_c \in C_j} B_c(l_j)\right).$$

parent node label

do the best
for itself

considering parent's
(i) preference

considering votes from
all its children (c)


model
transformed cost
part detection cost
transformed cost
part detection cost
transformed cost
part detection cost
+
x
x
x

Step 3: Make decision at the root node

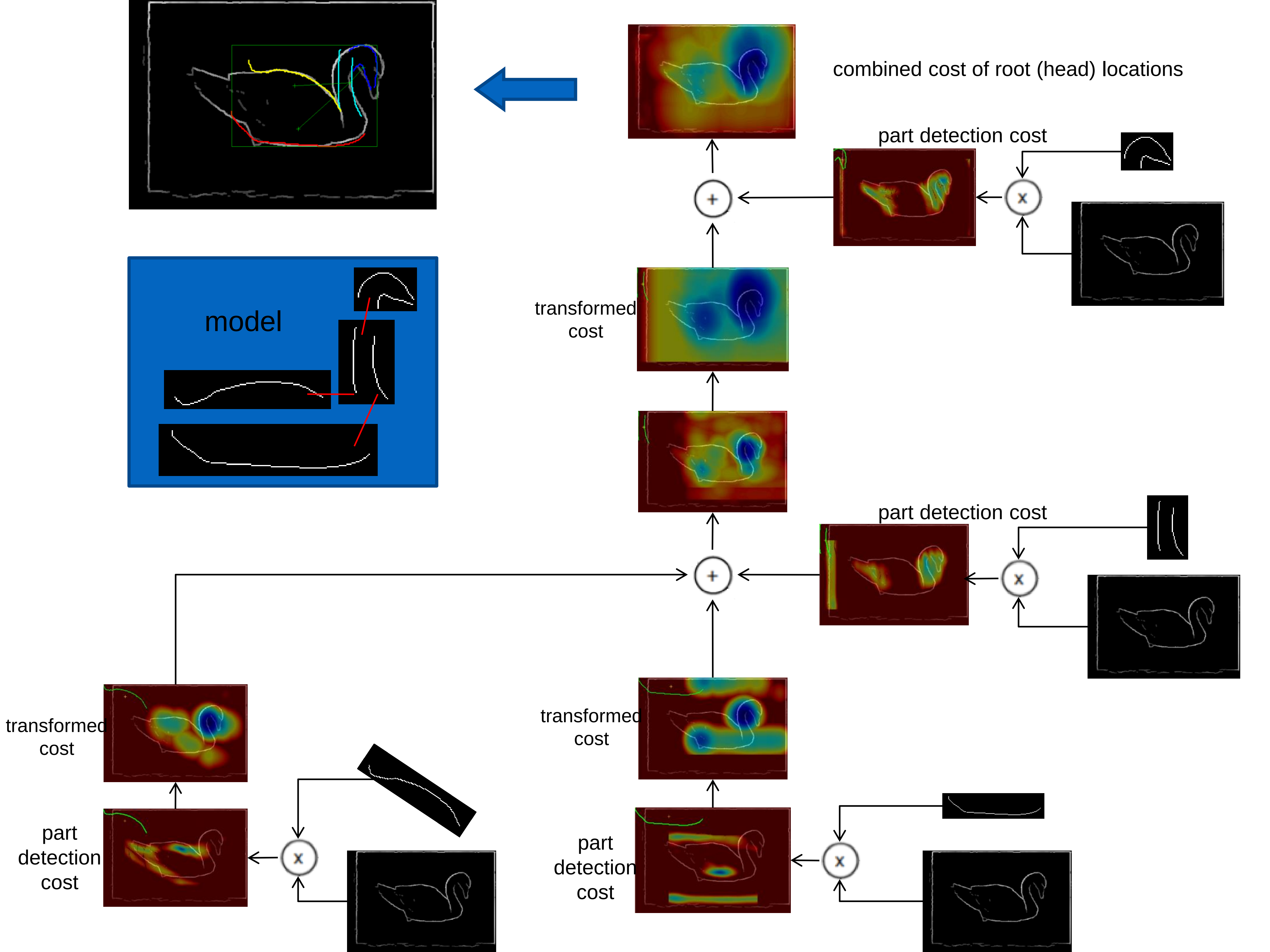
$$l_r^* = \arg\min_{l_r} \left(m_r(l_r) + \sum_{v_c \in C_r} B_c(l_j) \right)$$

do the best
for itself

considering votes from
all its children (c)

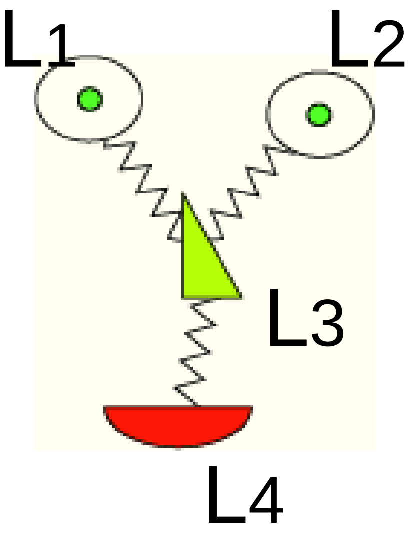
The decision at the root is purely local, no need to check with anyone else.

Good to the root, but one wrong choice, it effects the whole tree.



combined cost of root (head) locations
part detection cost
model
transformed cost
part detection cost
transformed cost
transformed cost
part detection cost
part detection cost

Step 4: recursively propagate information down



Given parent node is decided

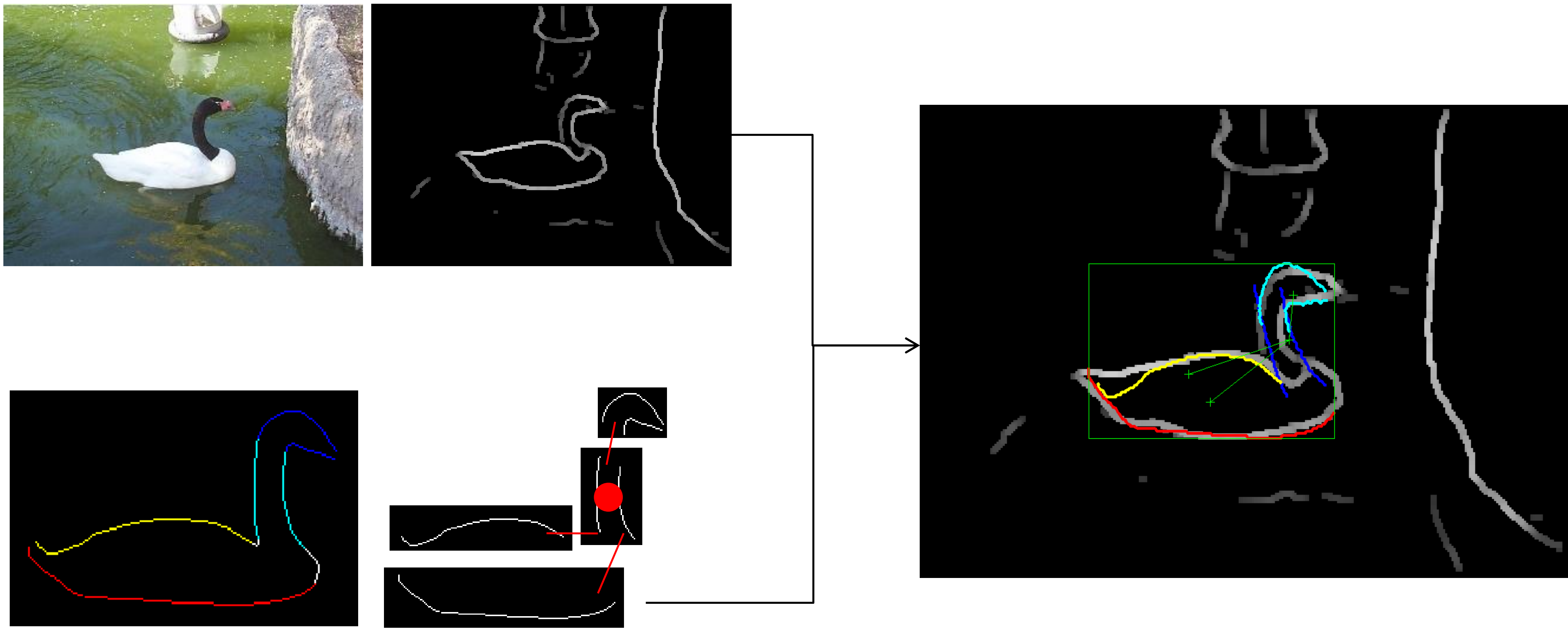
$$B_j(l_i)$$

current node label decision can be directly read off from the table

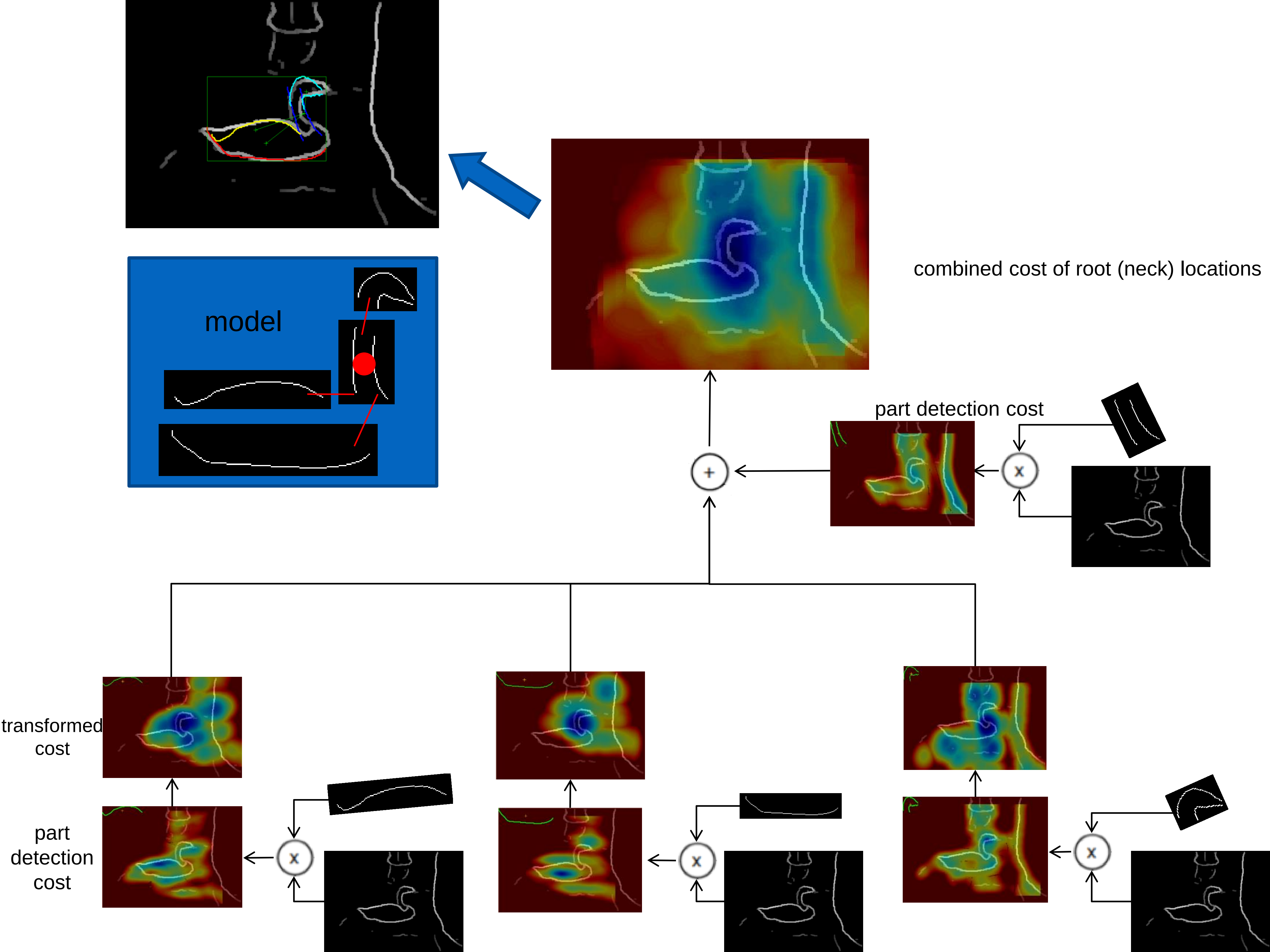
$$B_j(l_i)$$

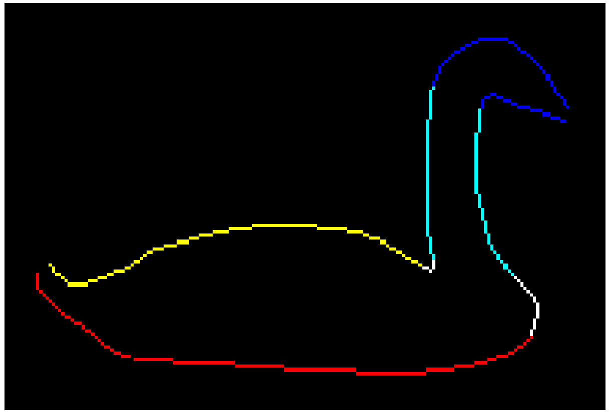
decide by read off from table

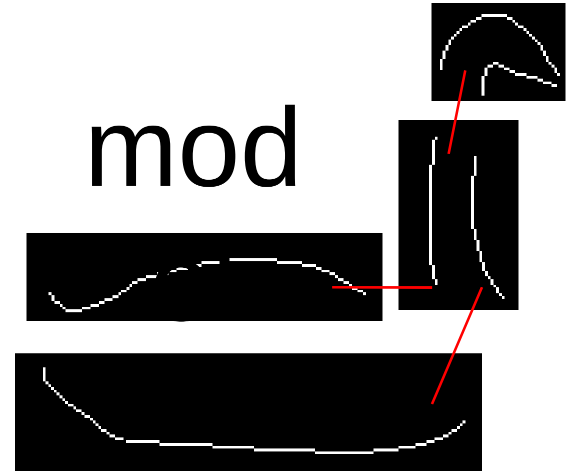
Deformable part model detection with 4 parts

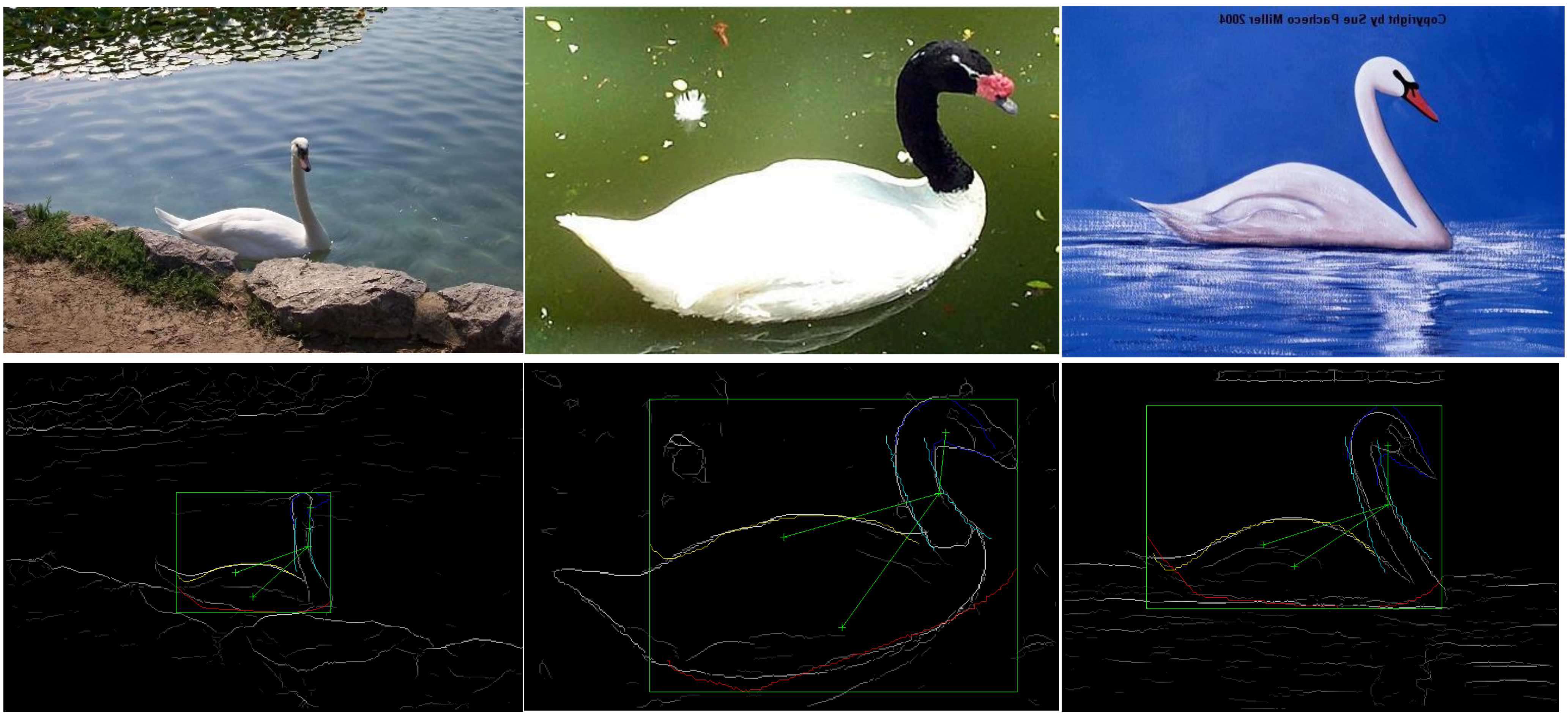


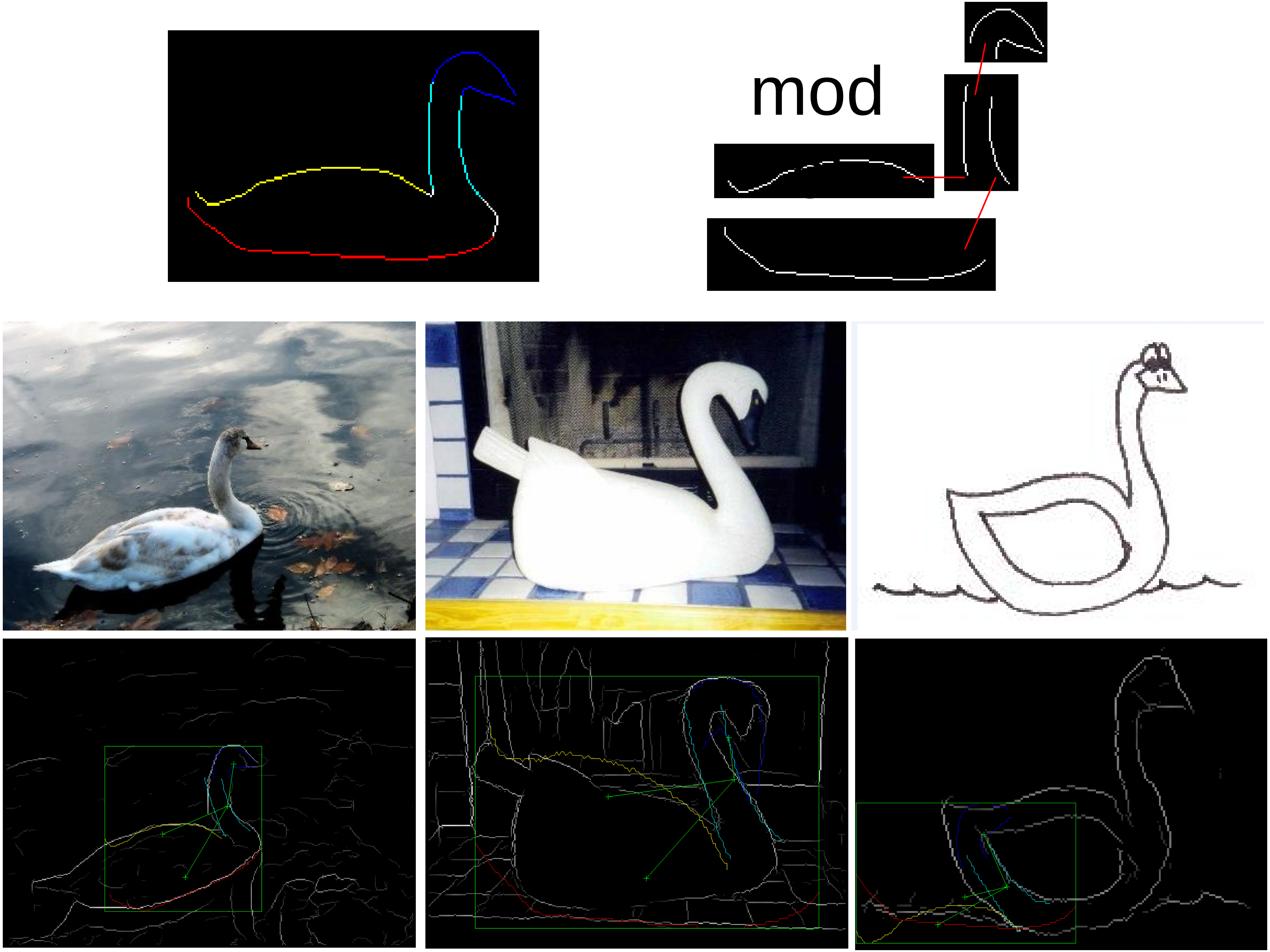
model

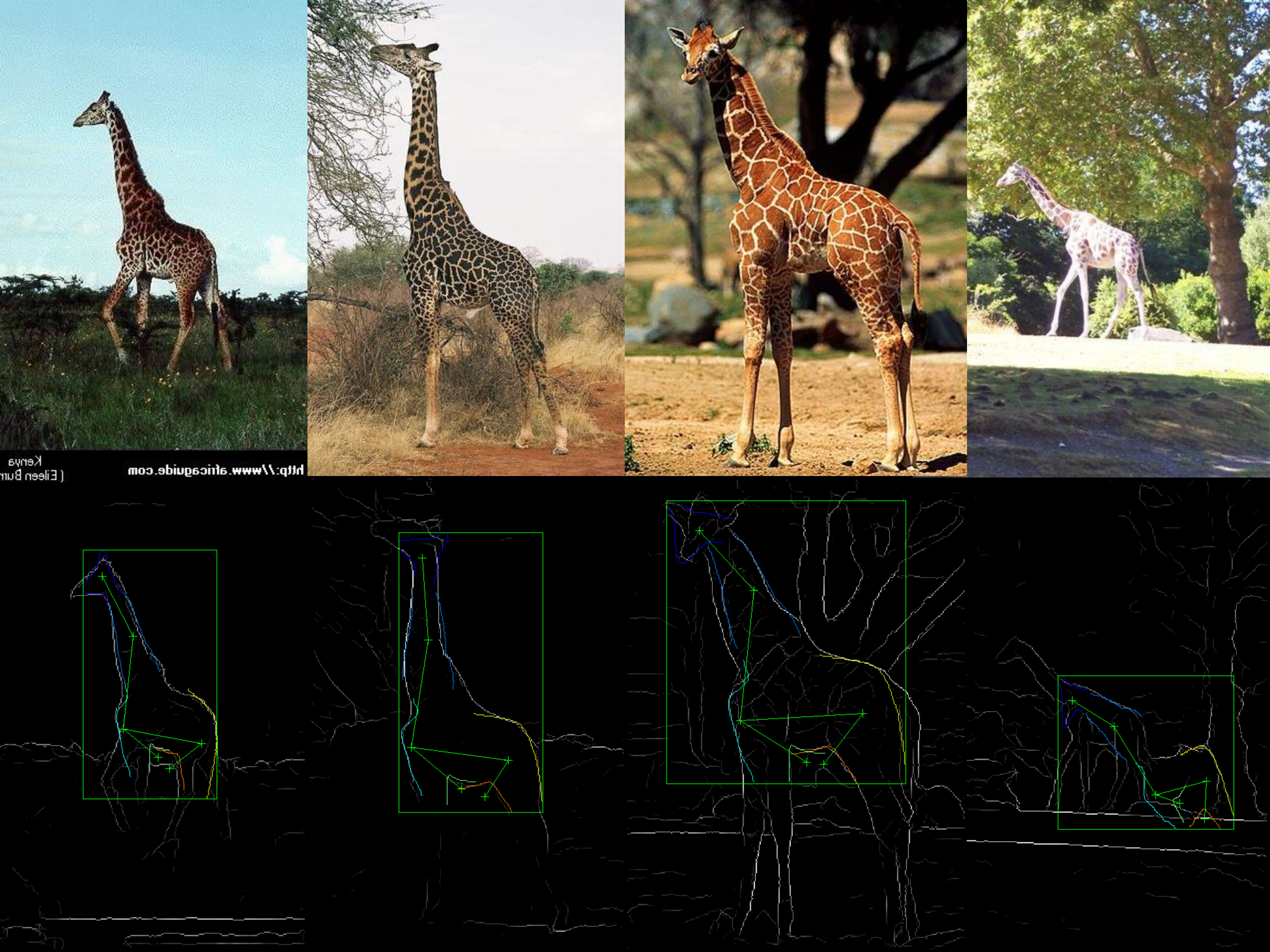

model
combined cost of root (neck) locations
part detection cost
transformed cost
part detection cost




mod


Copyright by Sue Pacheco Miller 2004

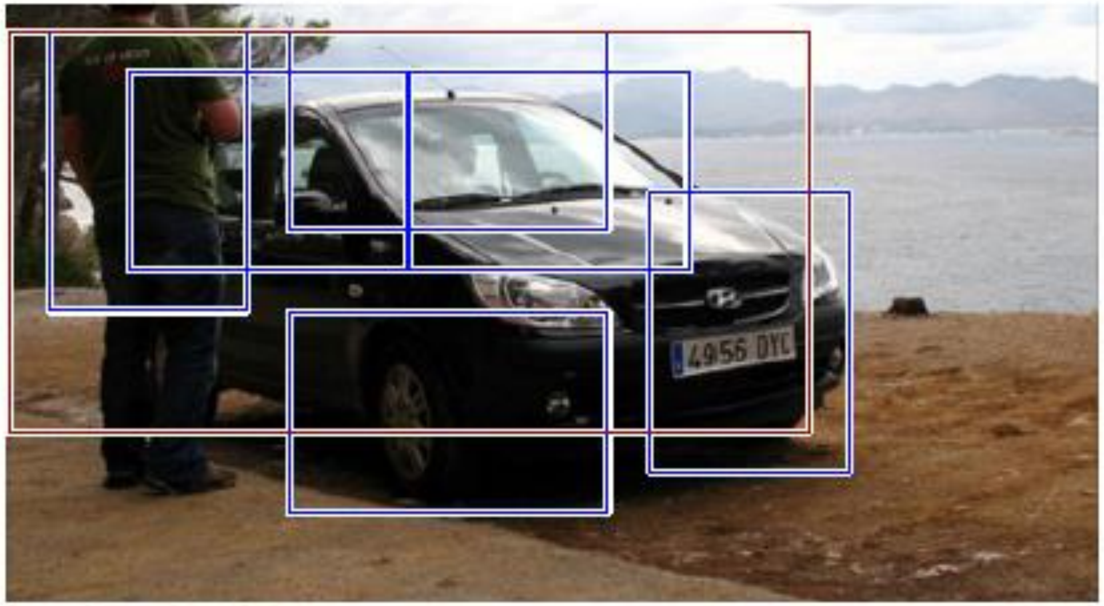

mod

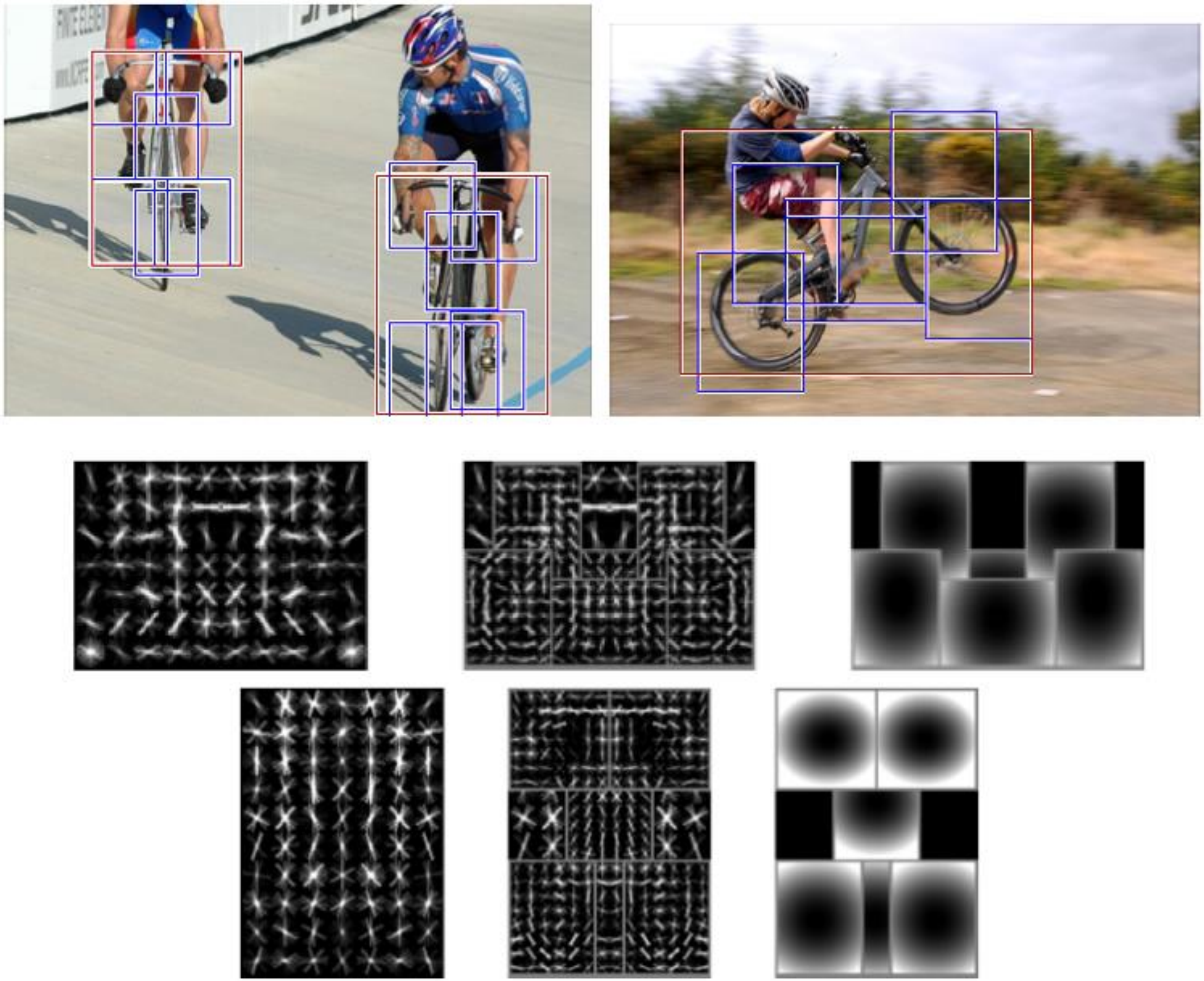

Kenya
http://www.africaguide.com

Learning Pictorial Structure

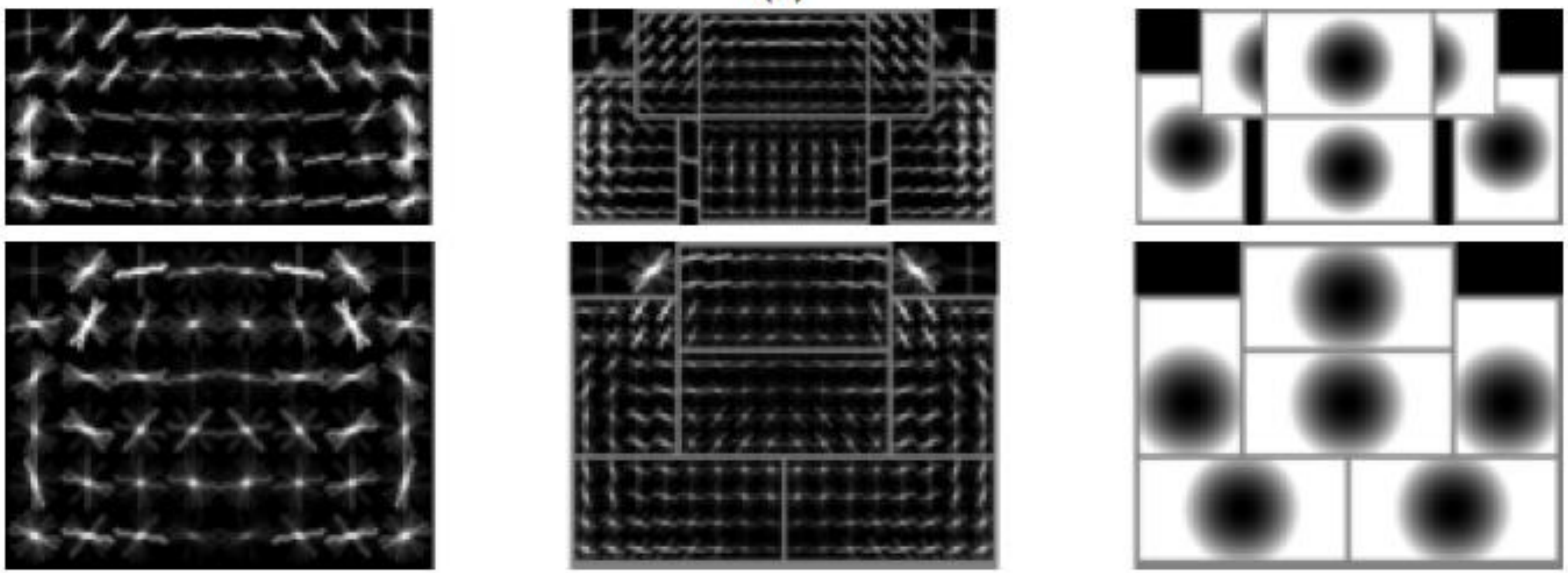
A Modern Version

1) fine level with deformable parts
2) coarse level with a fixed template model

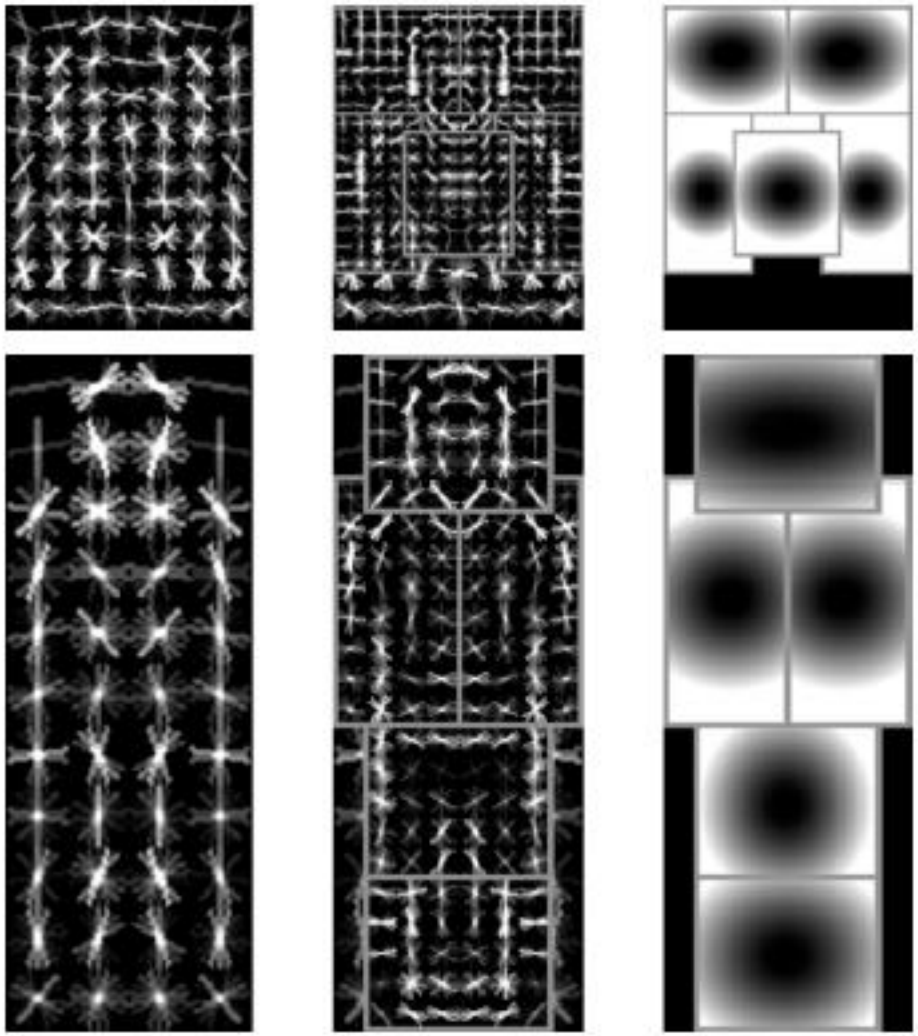




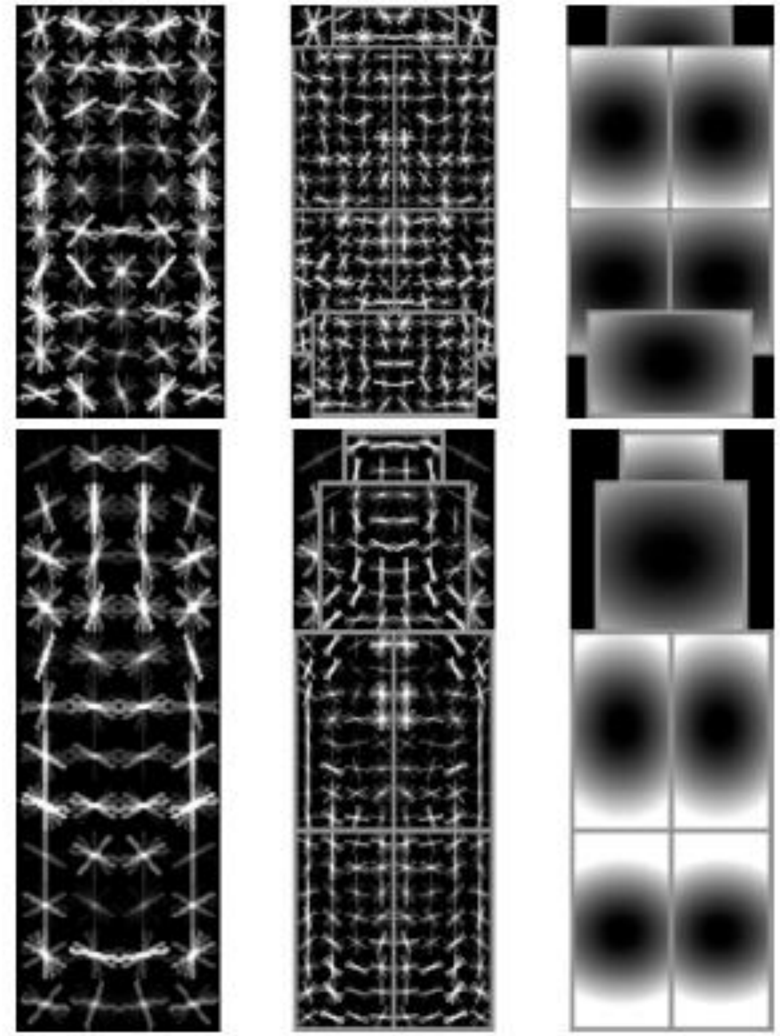
(c)



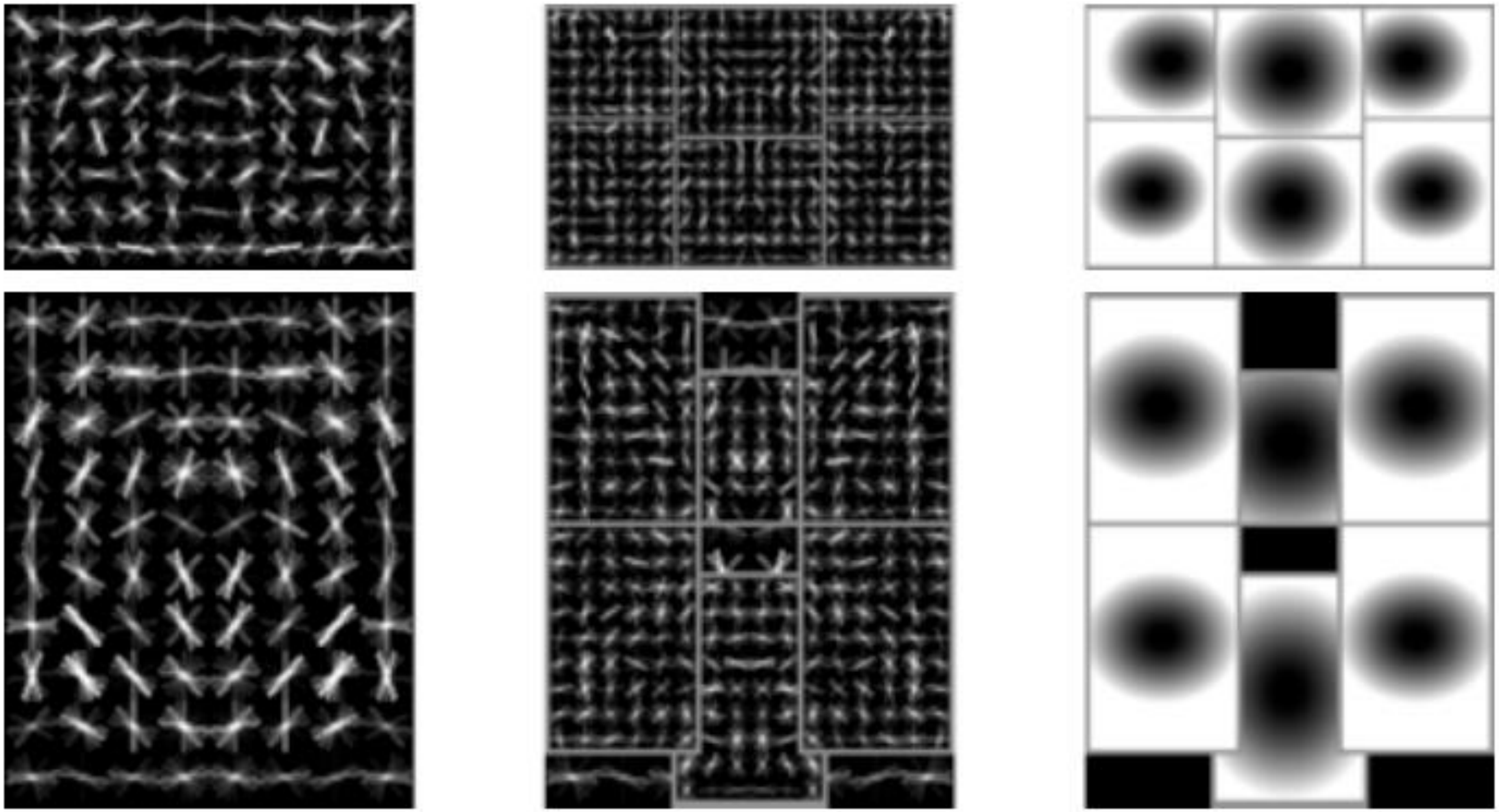
person

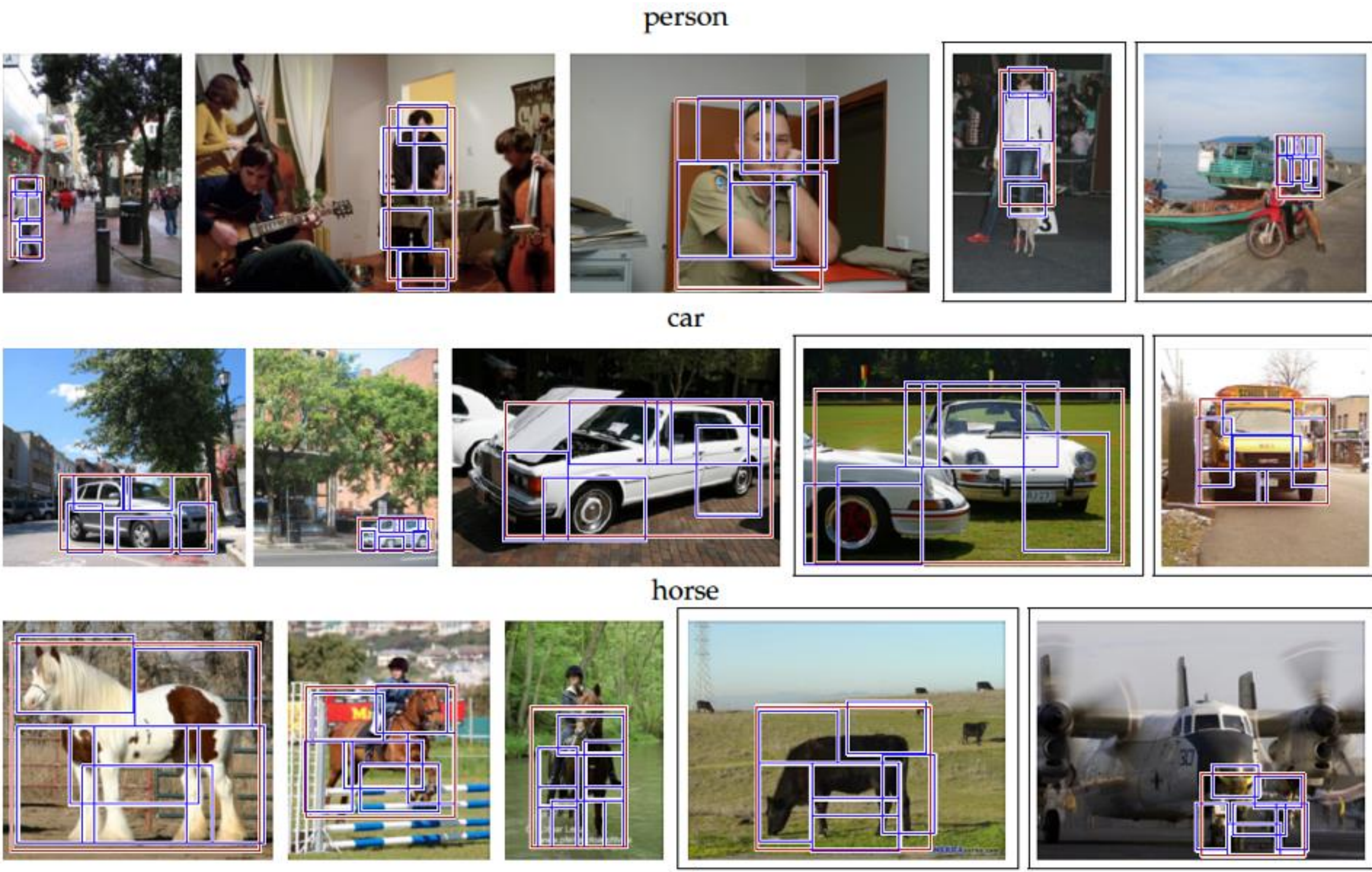


bottle

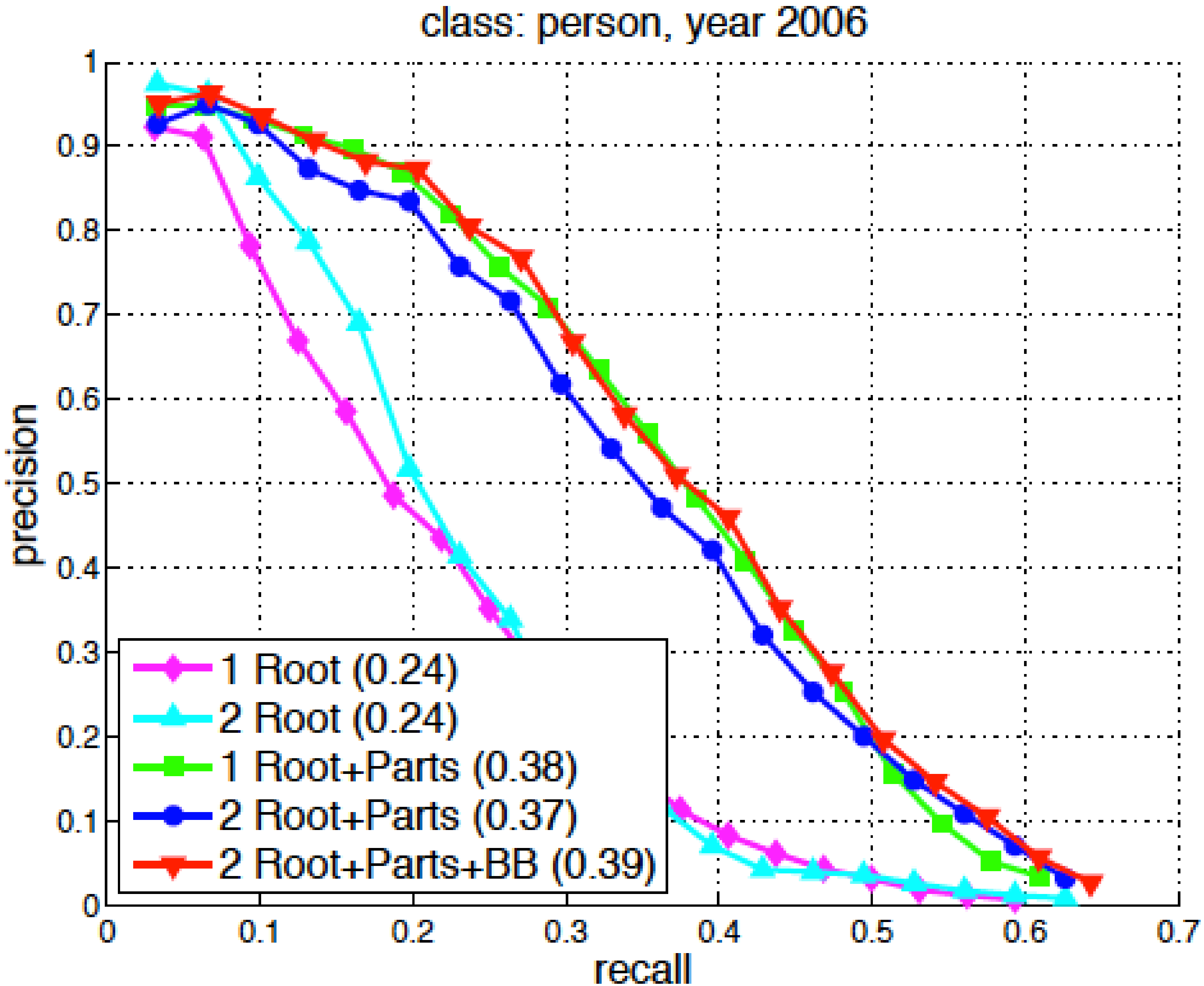


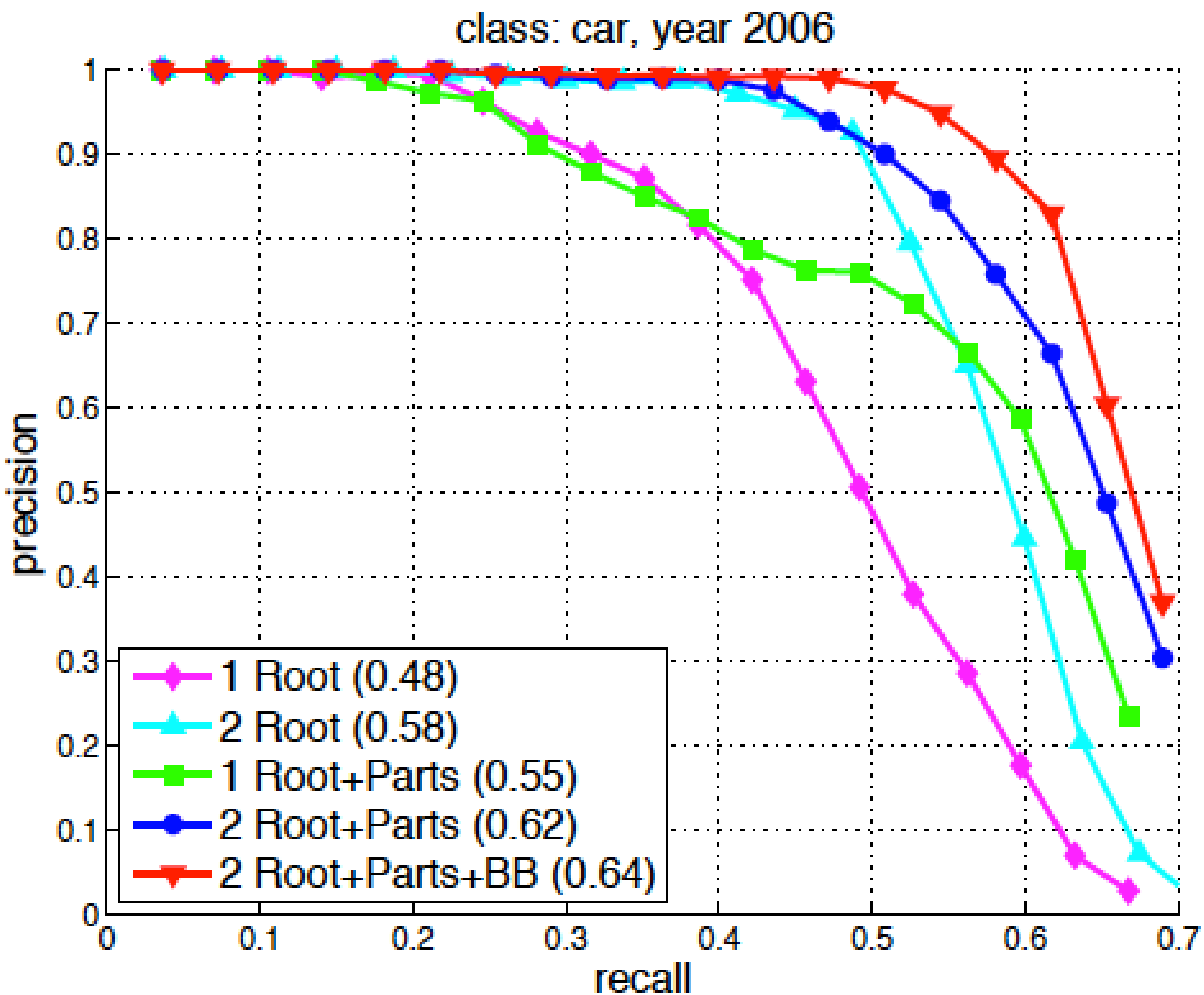
cat




person
car
horse


sofa
bottle
cat


class: person, year 2006
precision
recall
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
1 Root (0.24)
2 Root (0.24)
1 Root+Parts (0.38)
2 Root+Parts (0.37)
2 Root+Parts+BB (0.39)


class: car, year 2006
precision
recall
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
1 Root (0.48)
2 Root (0.58)
1 Root+Parts (0.55)
2 Root+Parts (0.62)
2 Root+Parts+BB (0.64)