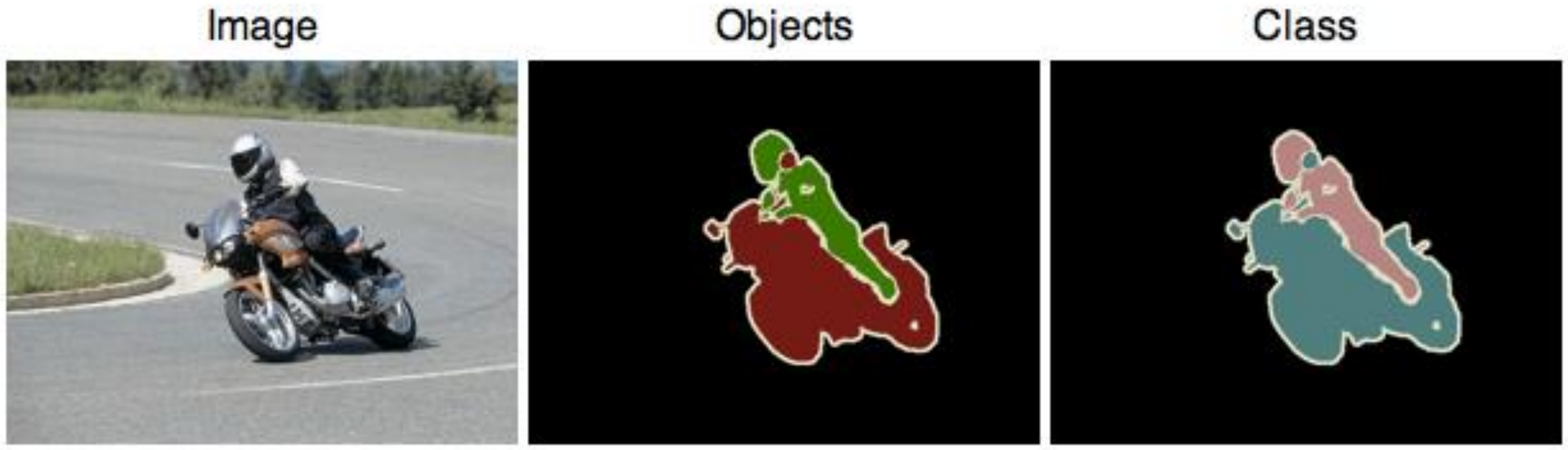
Shape recognition

What we see

What we really see

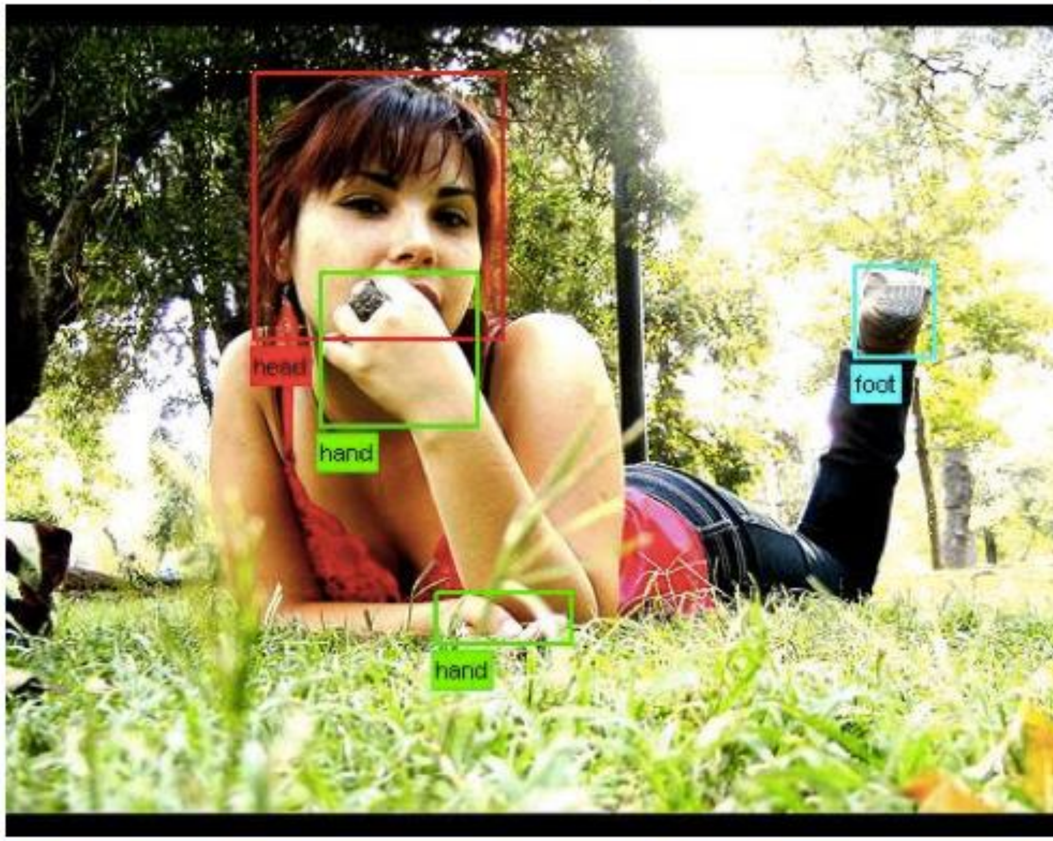
Object Detection

Object Segmentation



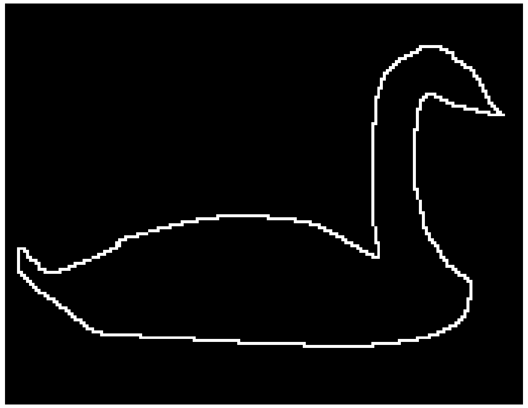
Image

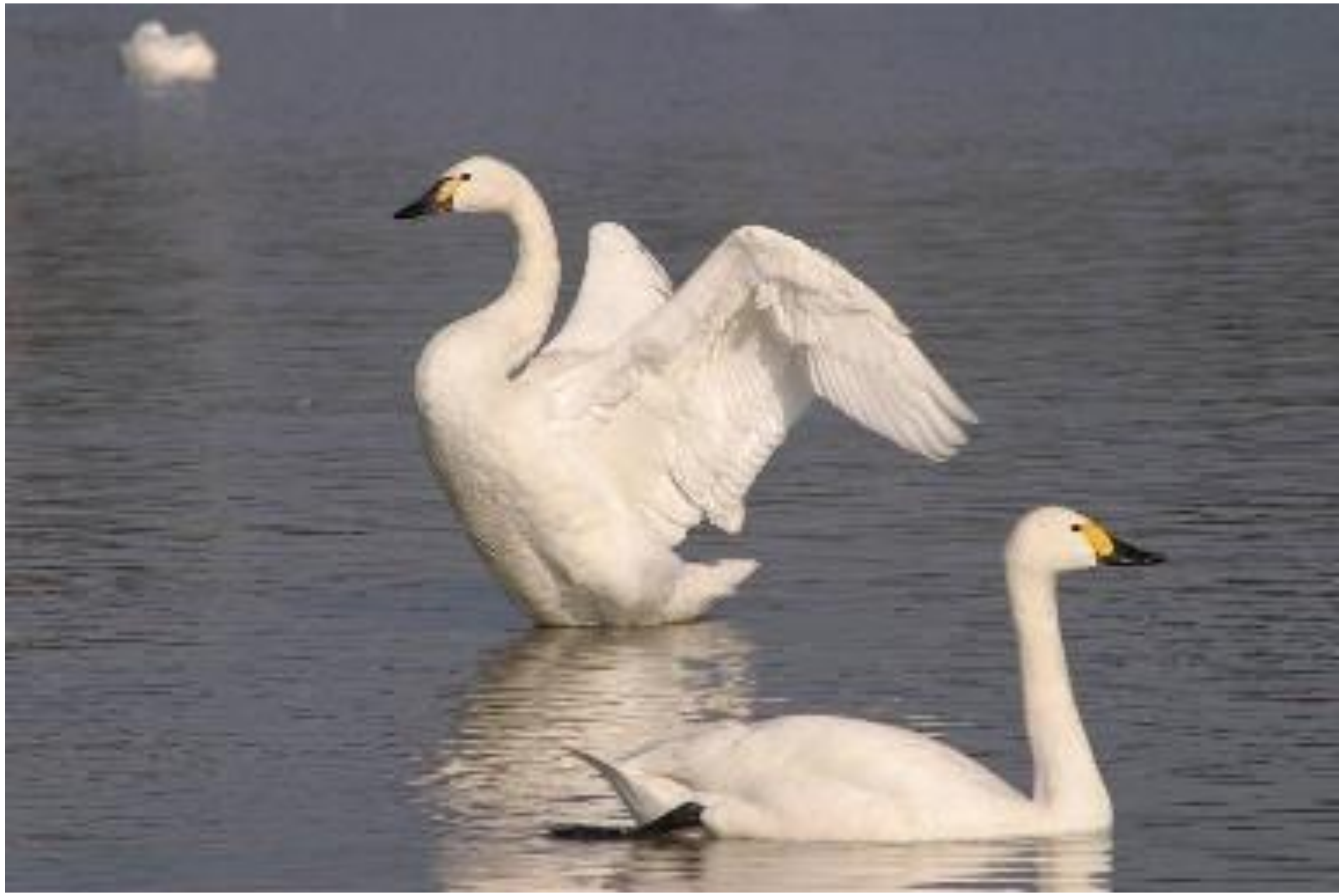
Person Layout



Basic Shape Comparison

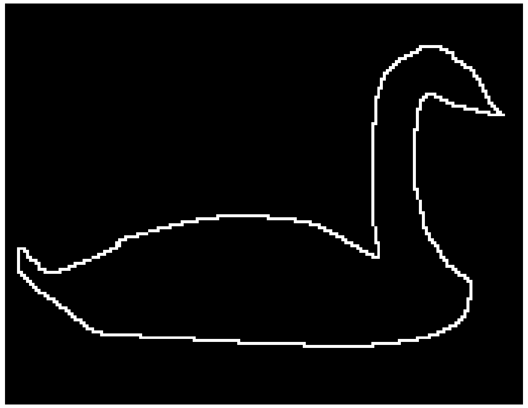
Contour based shape matching

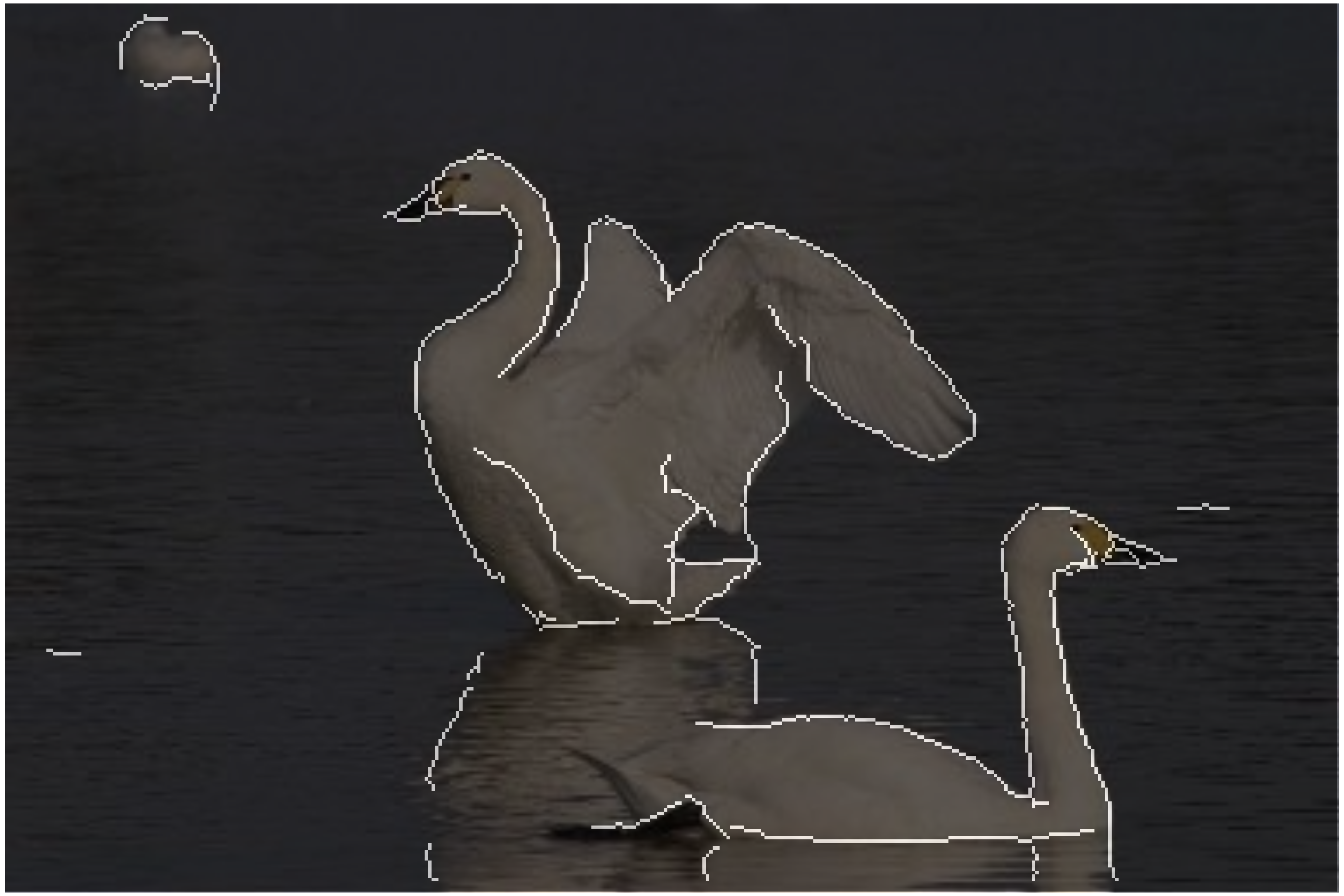

template shape


query image

How to find the template shape in the query image?

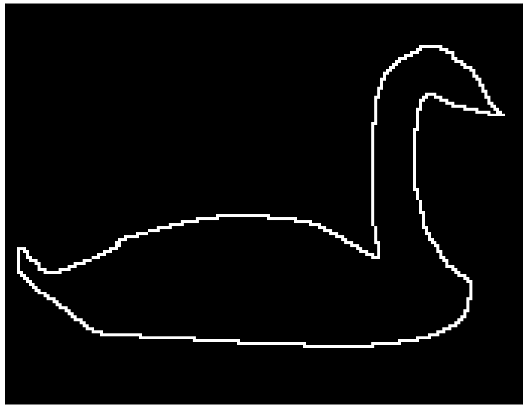
Contour based shape matching

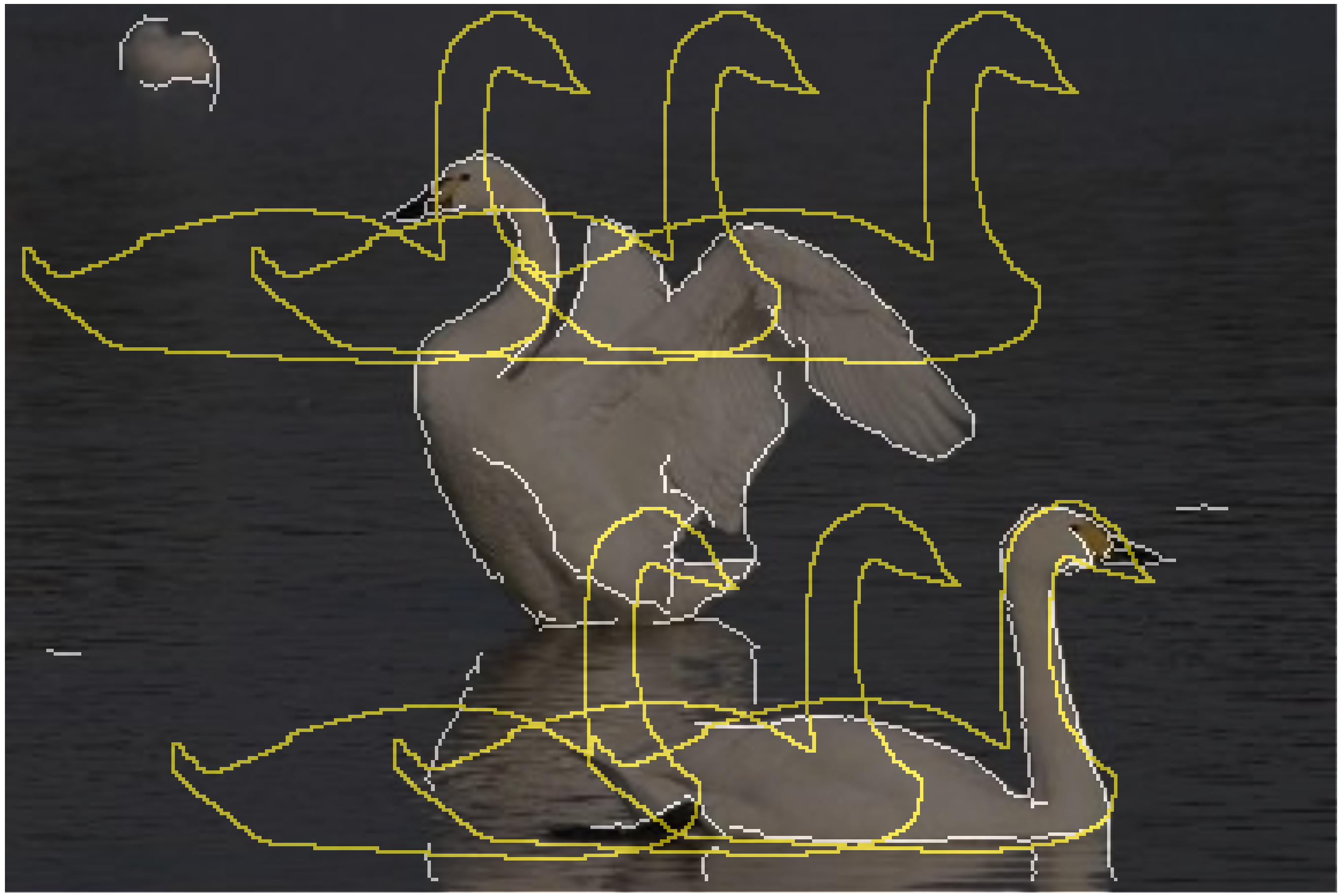

template shape


query image

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

Contour based shape matching

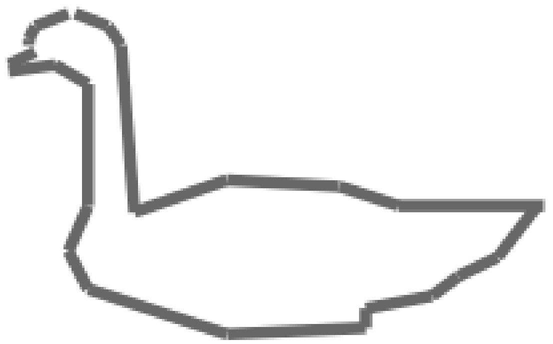


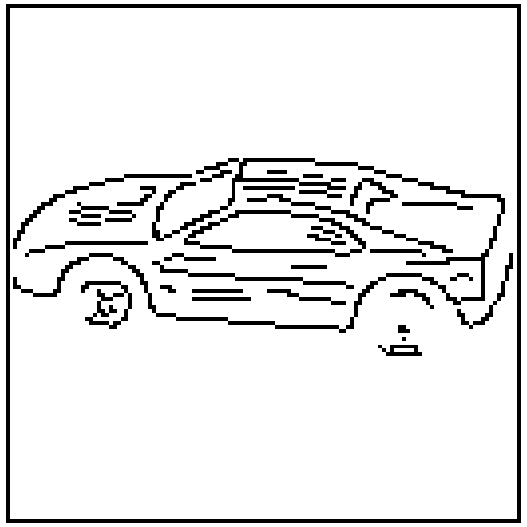


template shape

query image

Slide template over query image edge map





Let p, q be two edge sets to be compared

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

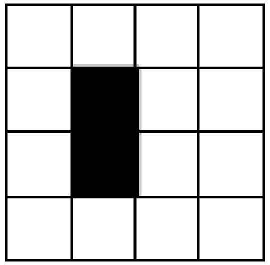
Distance transform: $D_q(x)$

Distance Transform Definition

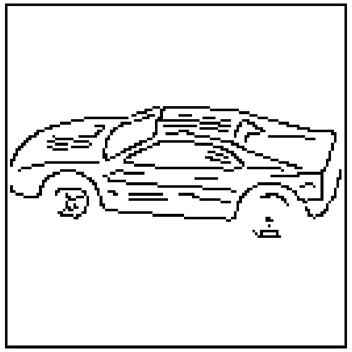
Set of points, P, some distance $\| \bullet \|$

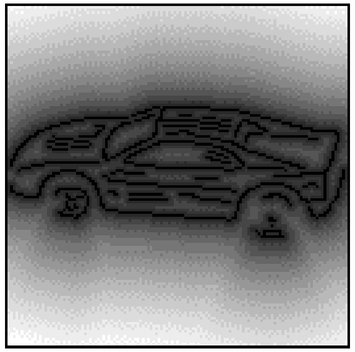
$$D_P(x) = \min_{y \in P} \| x - y \|$$

- 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

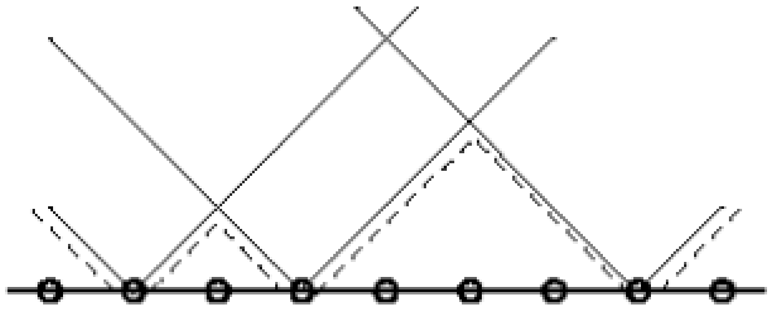




- Two pass O(n) algorithm for 1D L_1 norm (for simplicity just distance)
  1. Initialize: For all j
     $D[j] \leftarrow 1_{\mathbf{P}}[j]$
  2. Forward: For j from 1 up to n-1
     $D[j] \leftarrow \min(D[j], D[j-1]+1)$
  3. Backward: For j from n-2 down to 0
     $D[j] \leftarrow \min(D[j], D[j+1]+1)$

1	0

0	1

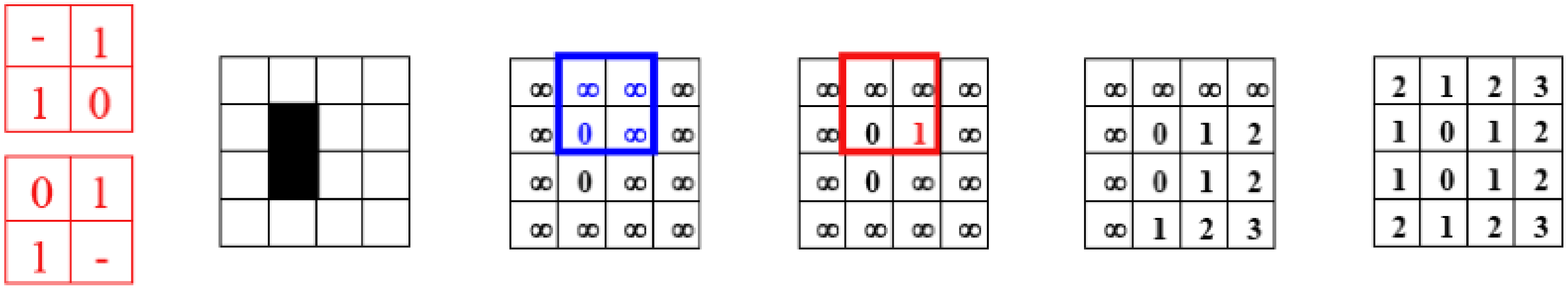


∞	0	∞	0	∞	∞	∞	0	∞

∞	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



-	1
1	0

0	1
1	-

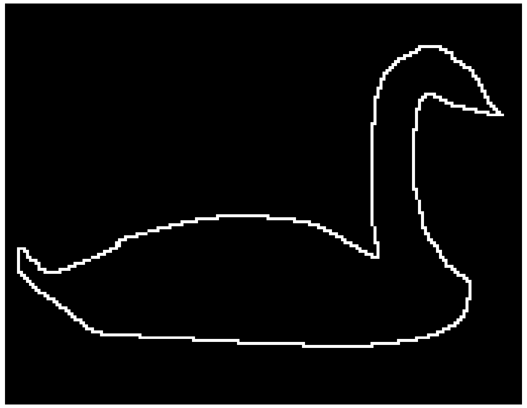
∞	∞	∞	∞
∞	0	∞	∞
∞	0	∞	∞
∞	∞	∞	∞

∞	∞	∞	∞
∞	0	1	∞
∞	0	∞	∞
∞	∞	∞	∞

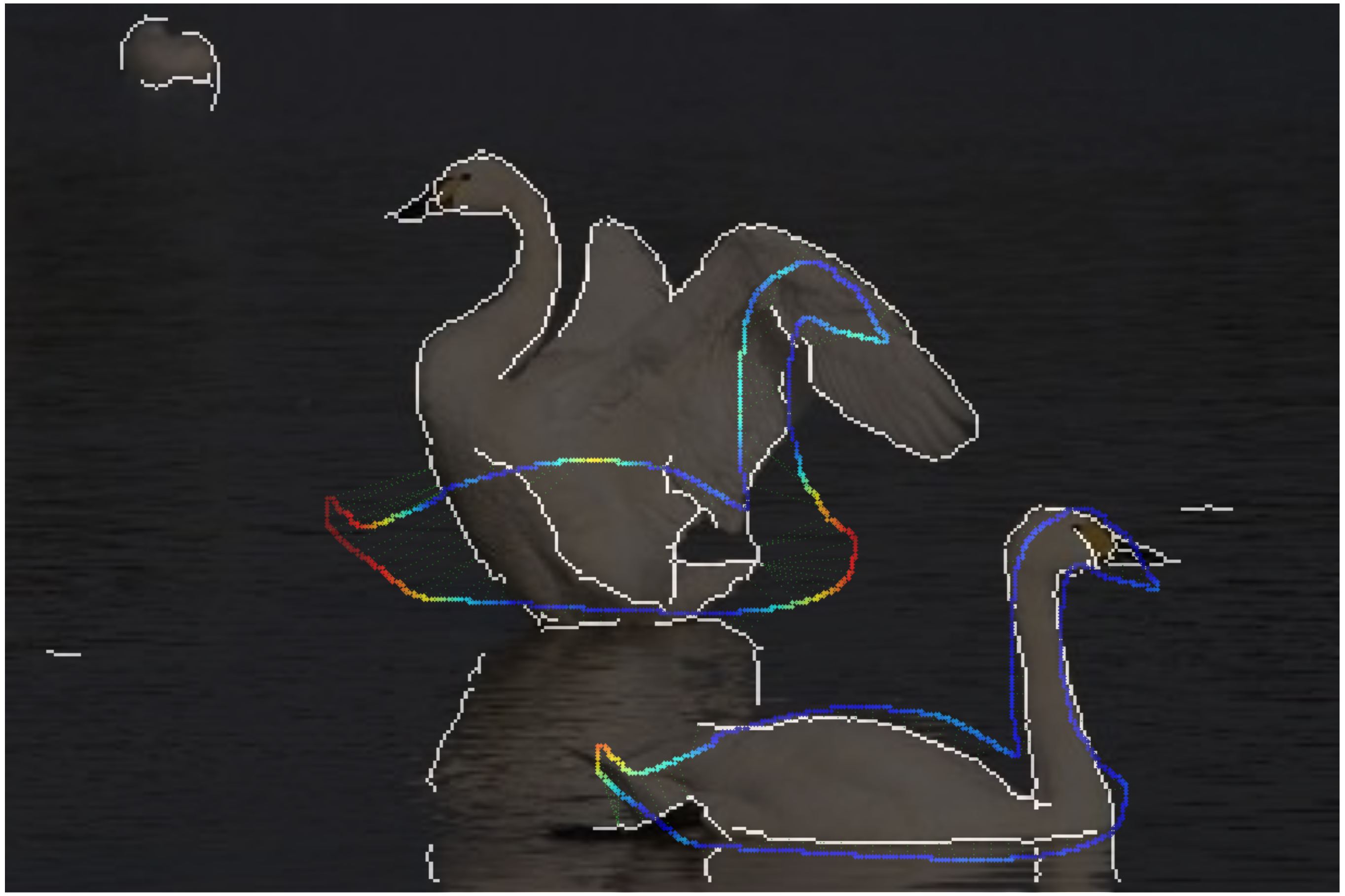
∞	∞	∞	∞
∞	0	1	2
∞	0	1	2
∞	1	2	3

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

Contour based shape matching



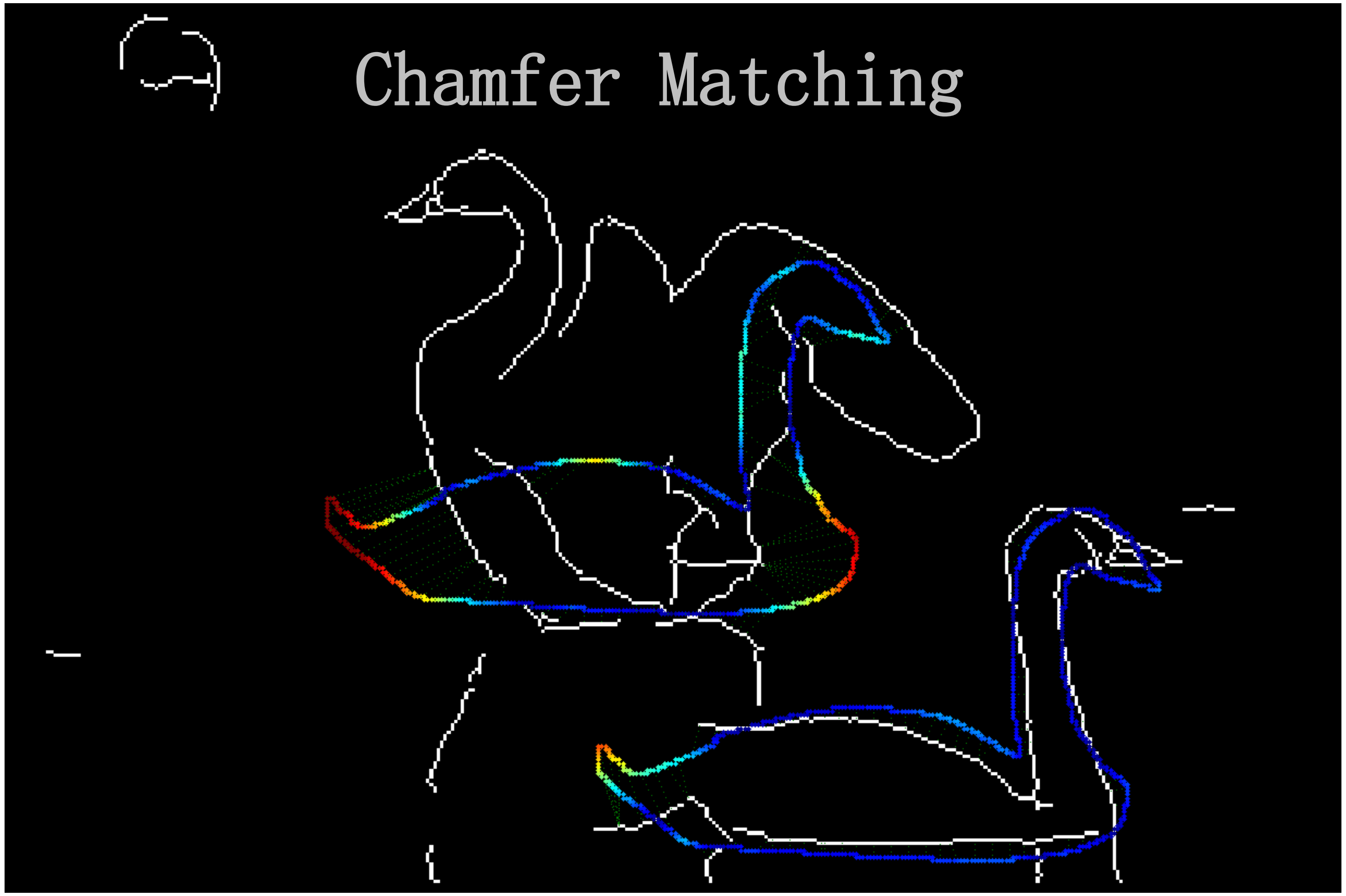
template shape



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)

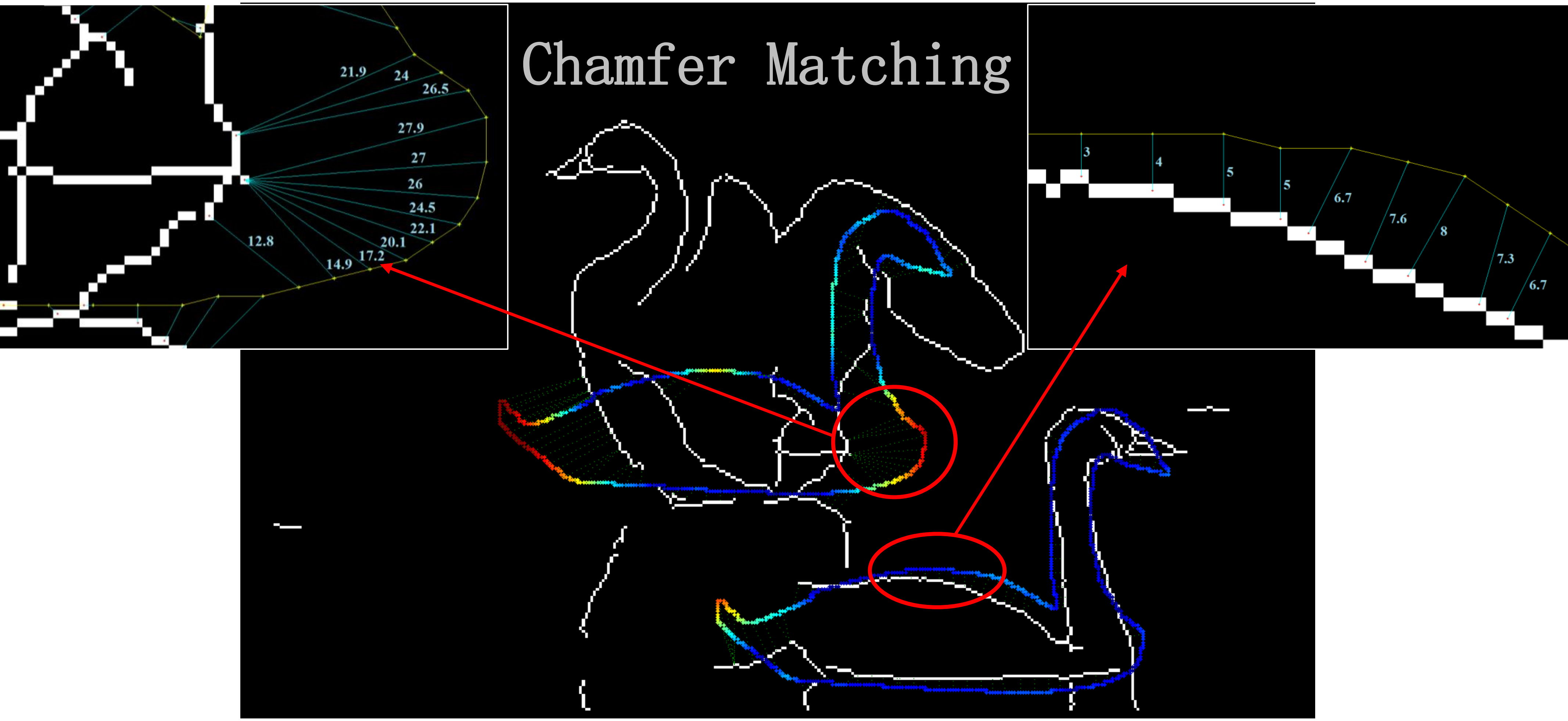
Chamfer Matching



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

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

Chamfer Matching



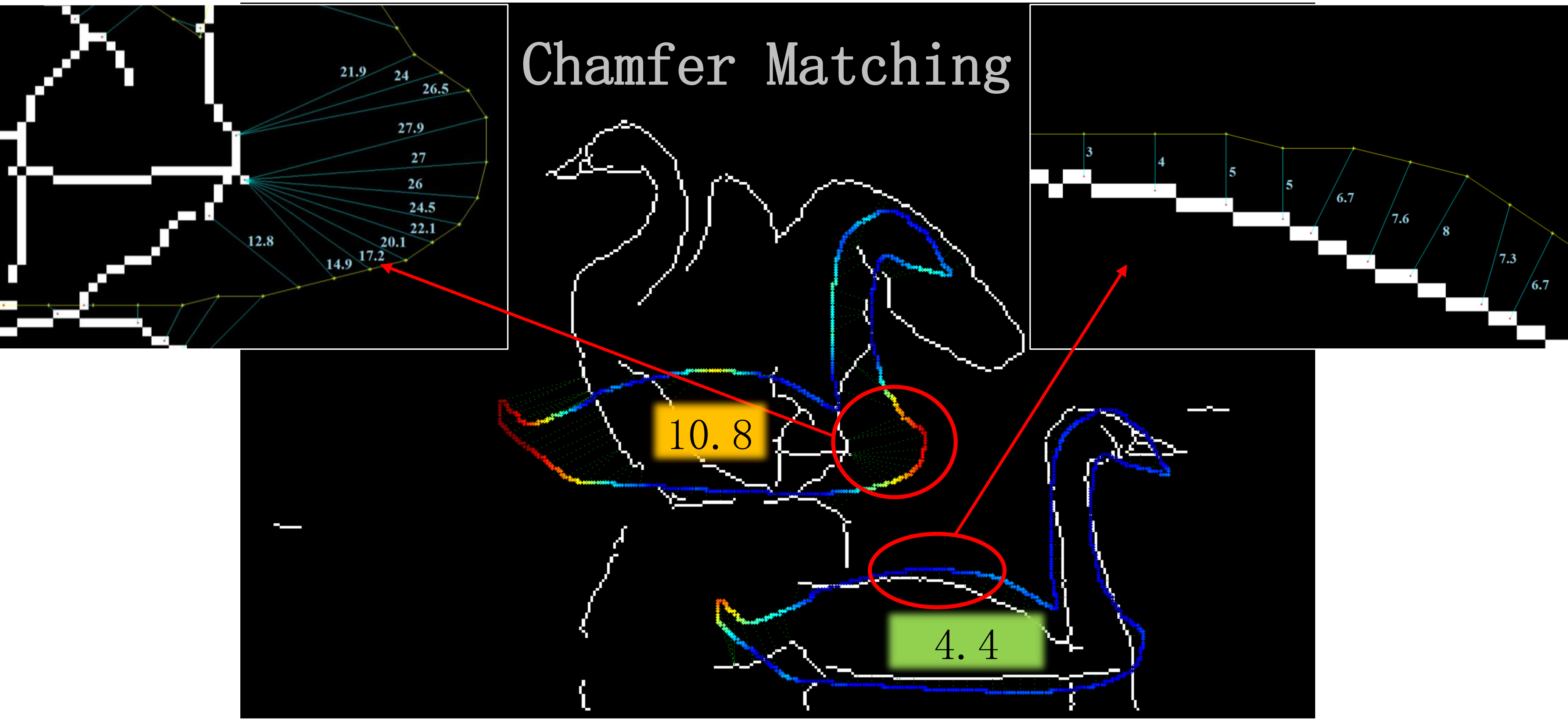
1. Slide template *T* by *(u, v)* over query image edge map *E*
2. **Matching cost of each pixel *p* in the shifted template *T(u, v)* is its shortest distance to any edge pixel *q* in the edge map *E***

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

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

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

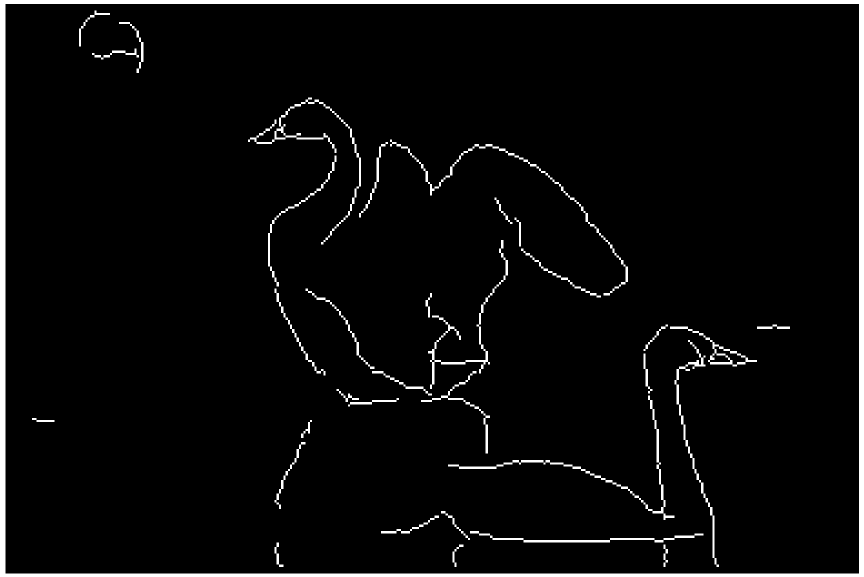
Chamfer Matching



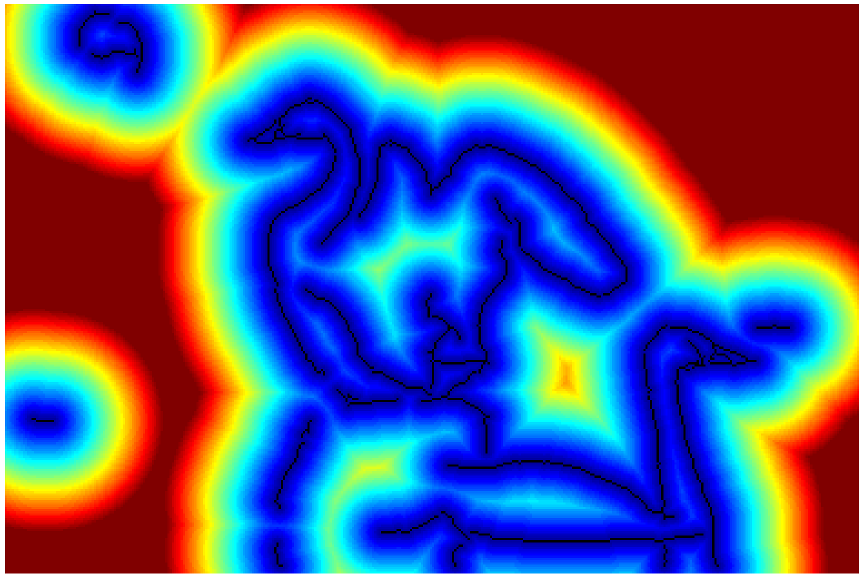
1. Slide template *T* by *(u, v)* over query image edge map *E*
2. Matching cost of each pixel *p* in the shifted template *T(u, v)* is its shortest distance to any edge pixel *q* in the edge map *E*
3. **Total cost of the shifted template is the average cost of each shifted template pixel**

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

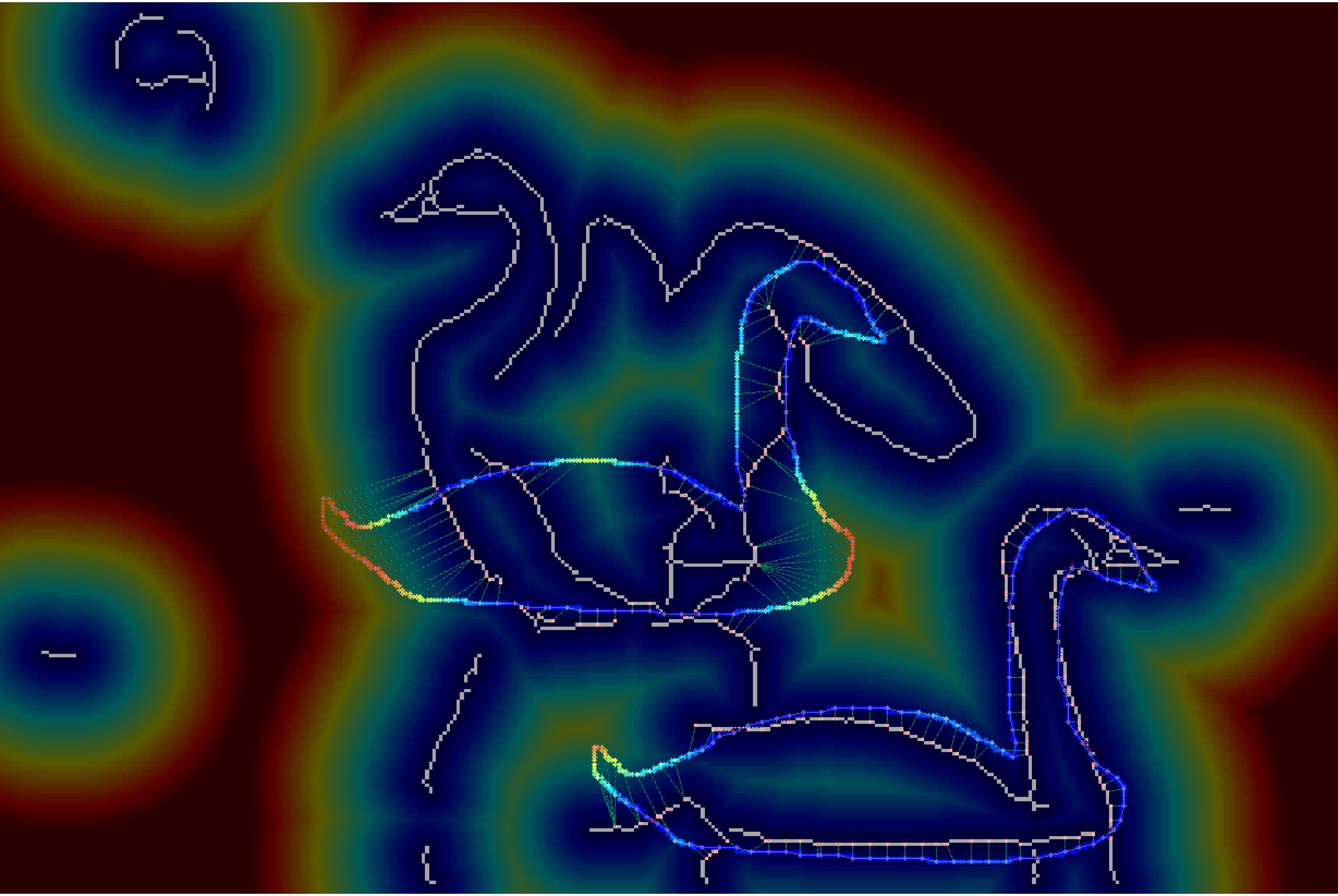
generalized distance transform

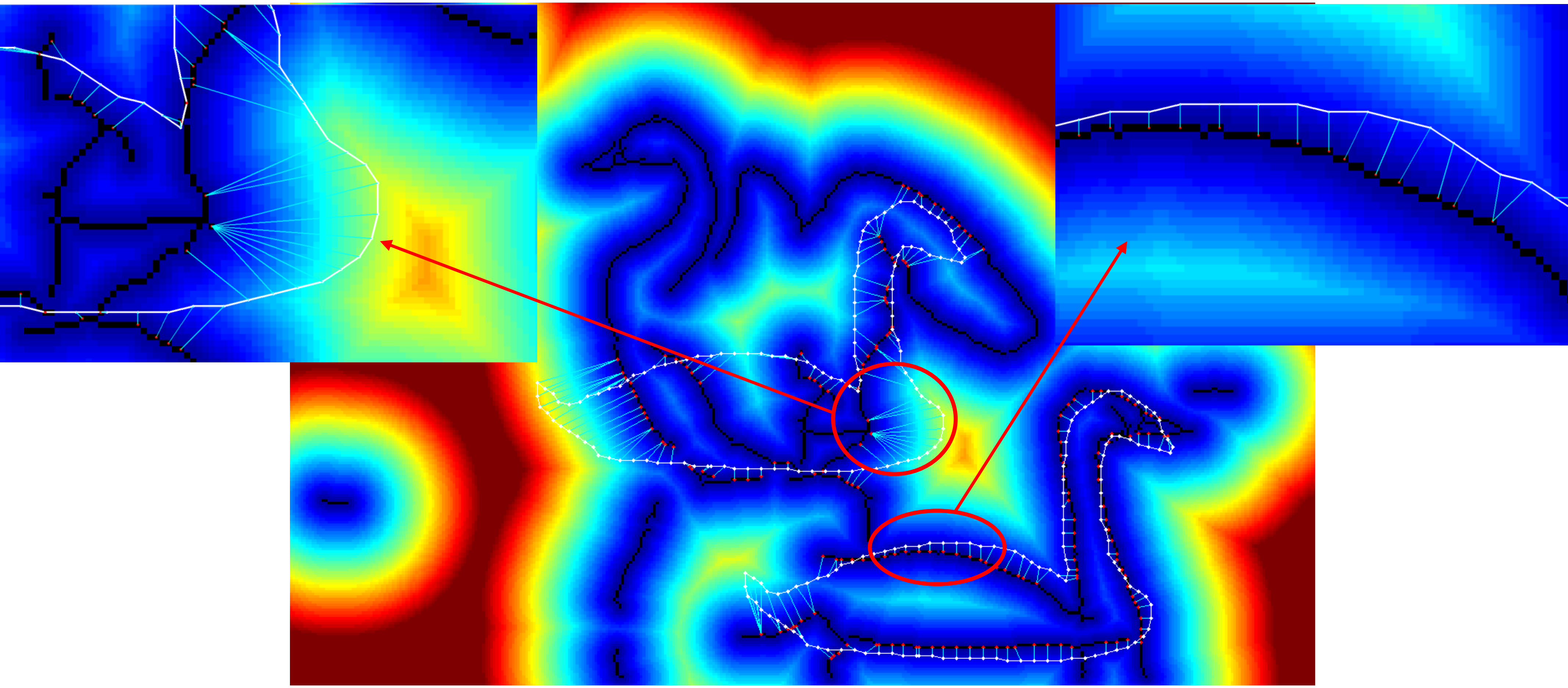


E



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





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

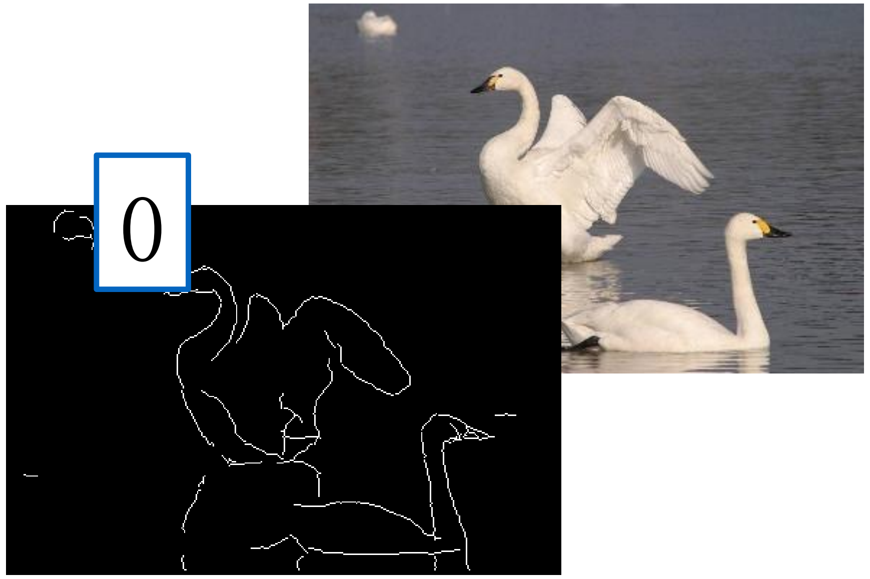
Evaluation time for each shift is just $\mathcal{O}(\|T\|\,\mathcal{O}(\min_{q\in E}\|p-q\|_2)) = \mathcal{O}(\|T\|)$

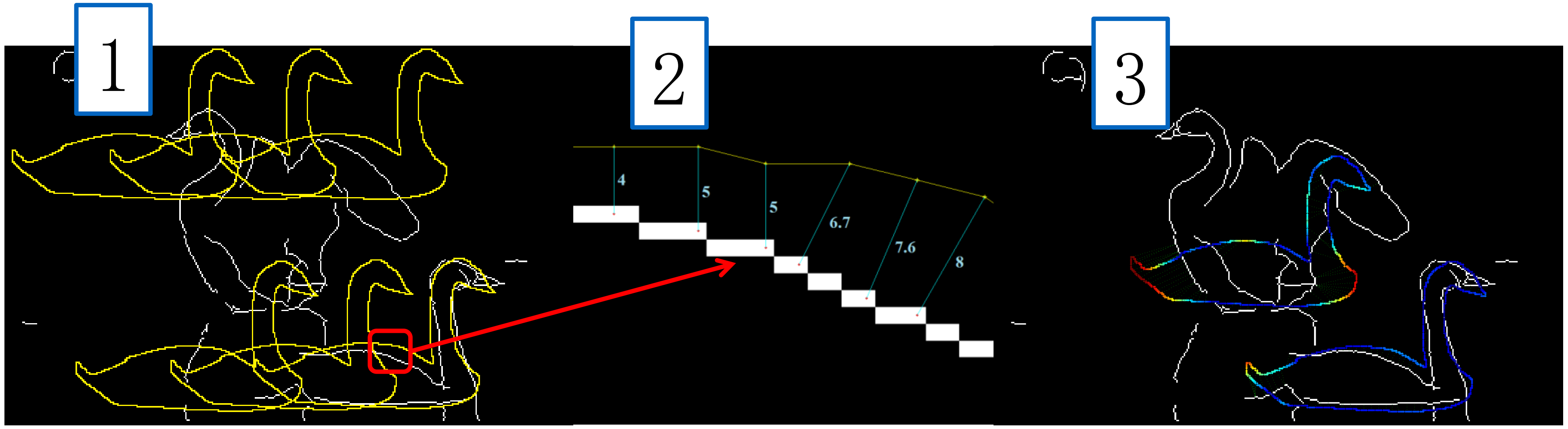
Total running time for *m* shifts is: (typically, *m* = *n*)

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

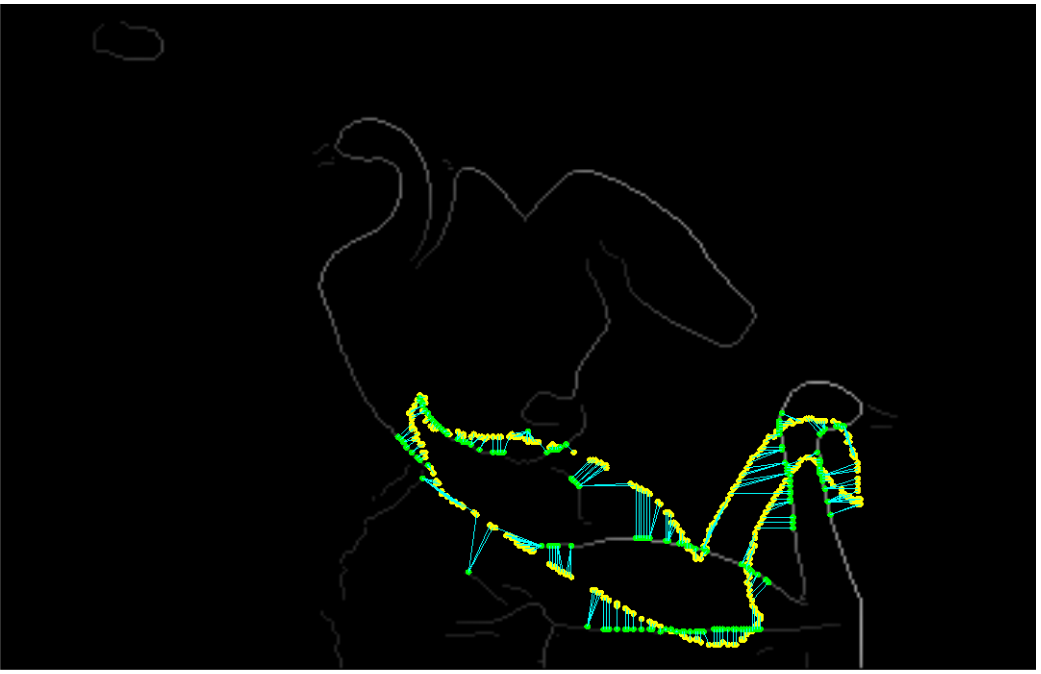
Chamfer Matching Review

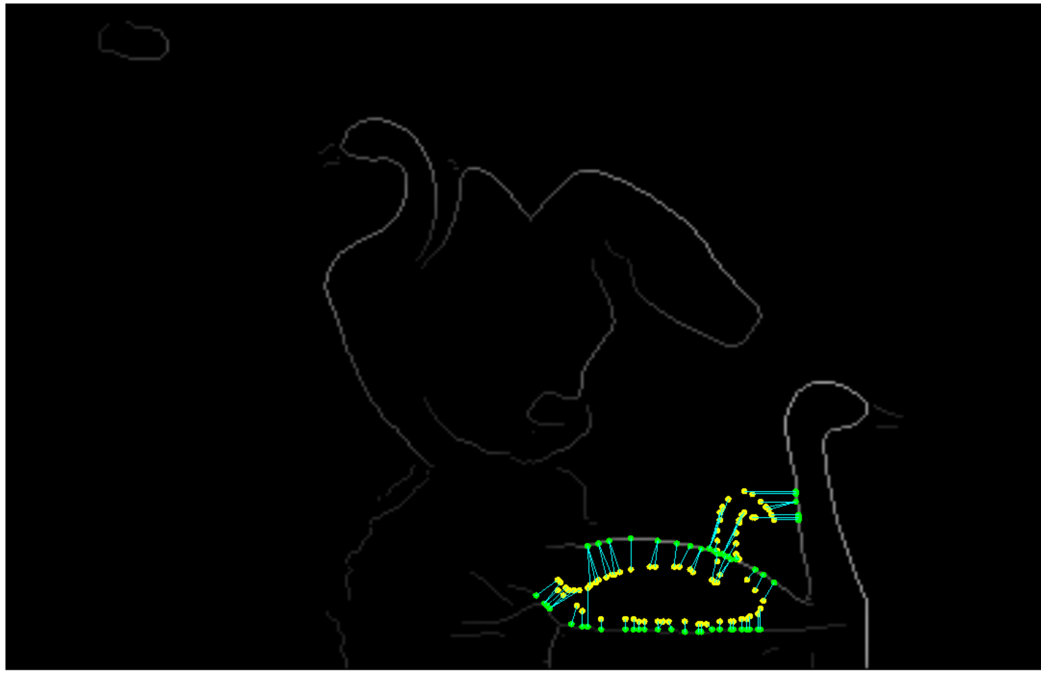
0. Detect edges in query image
1. Slide template over query image edge map
2. Find closest edge pixel in image for each shifted template pixel
3. At each location, compute average distance from each pixel in template to closest edge in image
4. Lowest cost match wins

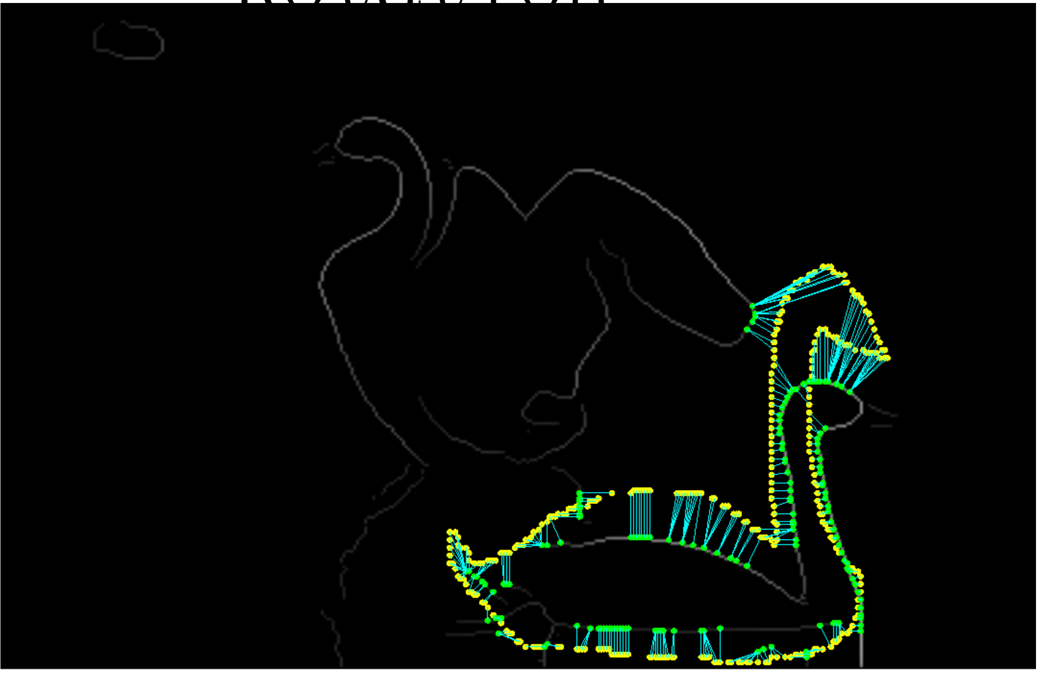


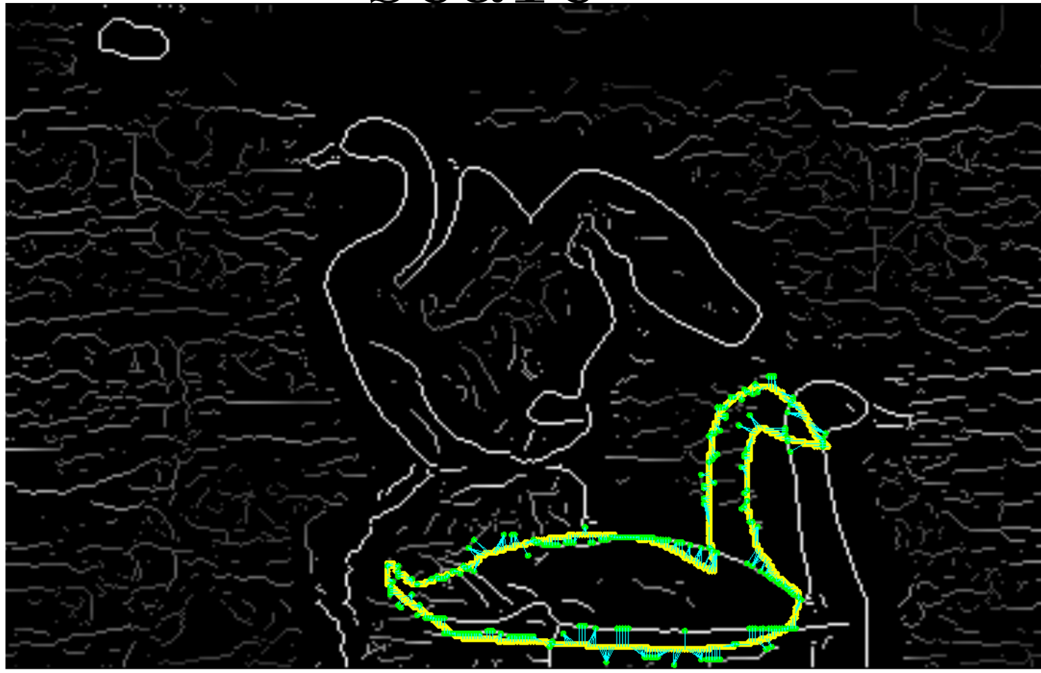


Weaknesses of Chamfer Matching?


Rotation

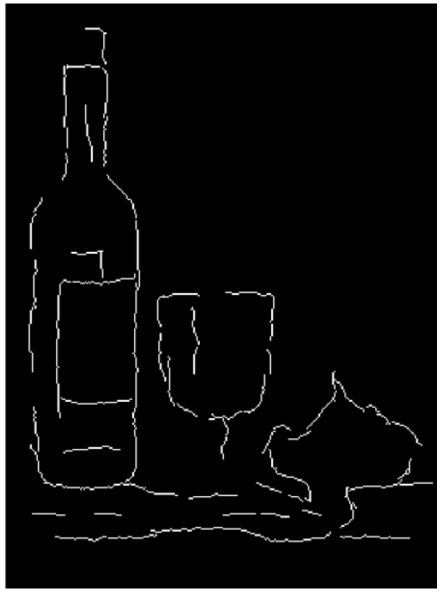

Scale

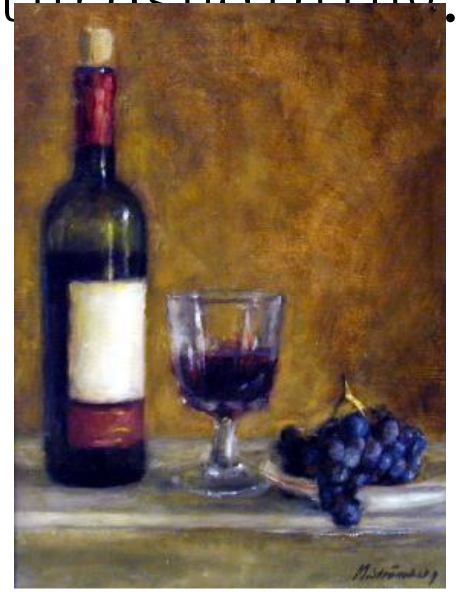

Aspect ratio

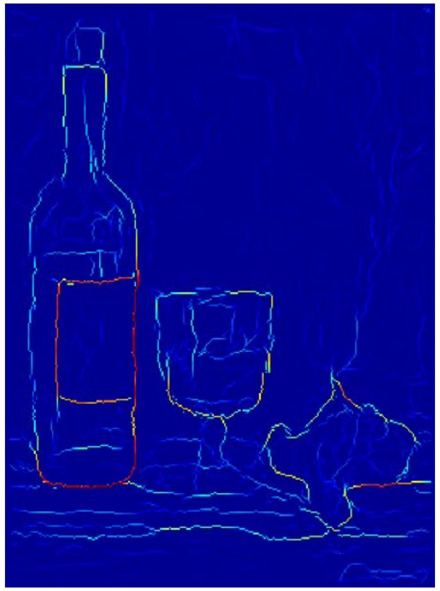

Bad edge map threshold / Clutter

Some Alternatives

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







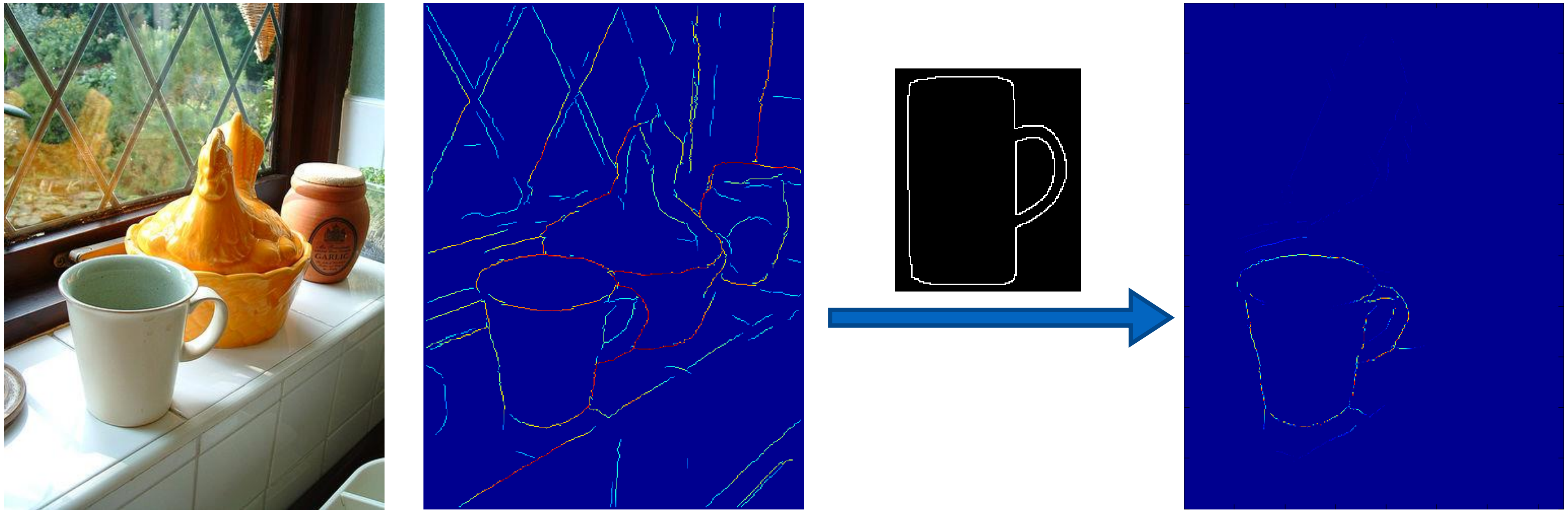
$$c(p) = \min_{q:f(q)>0}\left(\left(\frac{1}{f(q)^2}-1\right)+\|p-q\|+\lambda|\varphi(p)-\varphi(q)|\right)$$

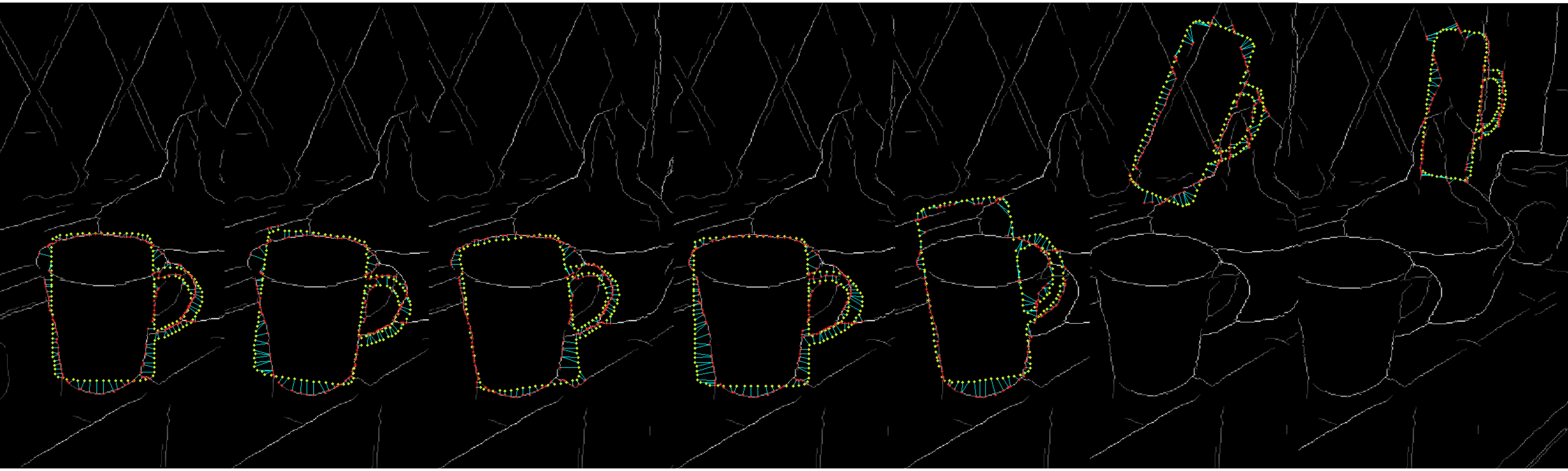
Where *f(q)* is the "edgeness" of pixel q, and *f(q)* is in [0,1].

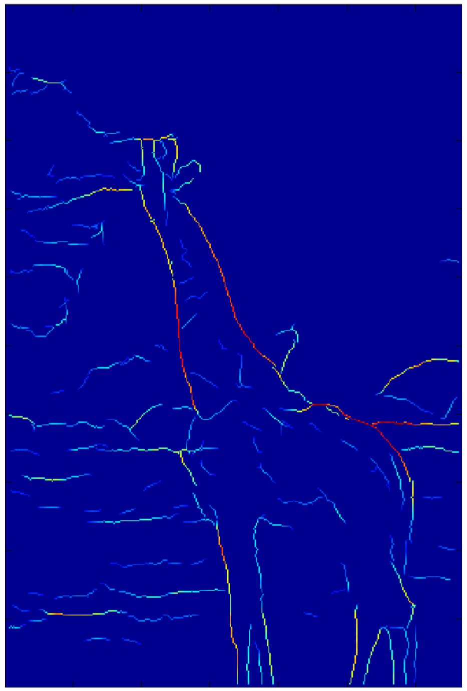
Distance transform still applies.

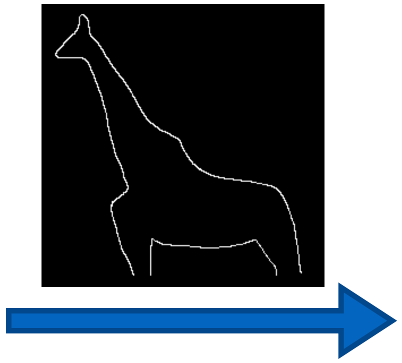
Voting from low cost matches:

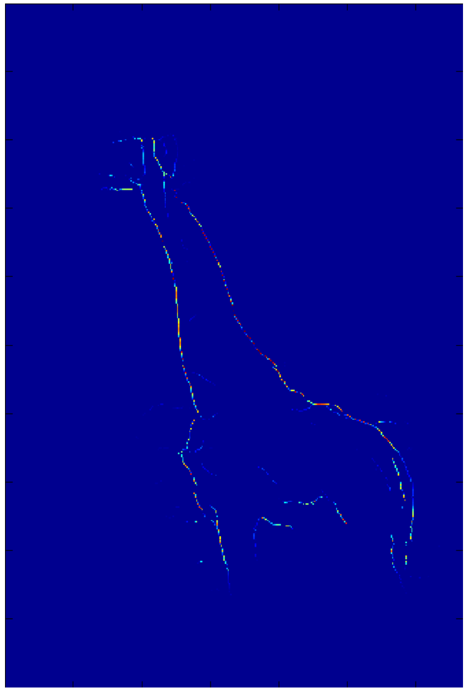
Each hypothesis votes for edge pixels in the query image that participates in the match.

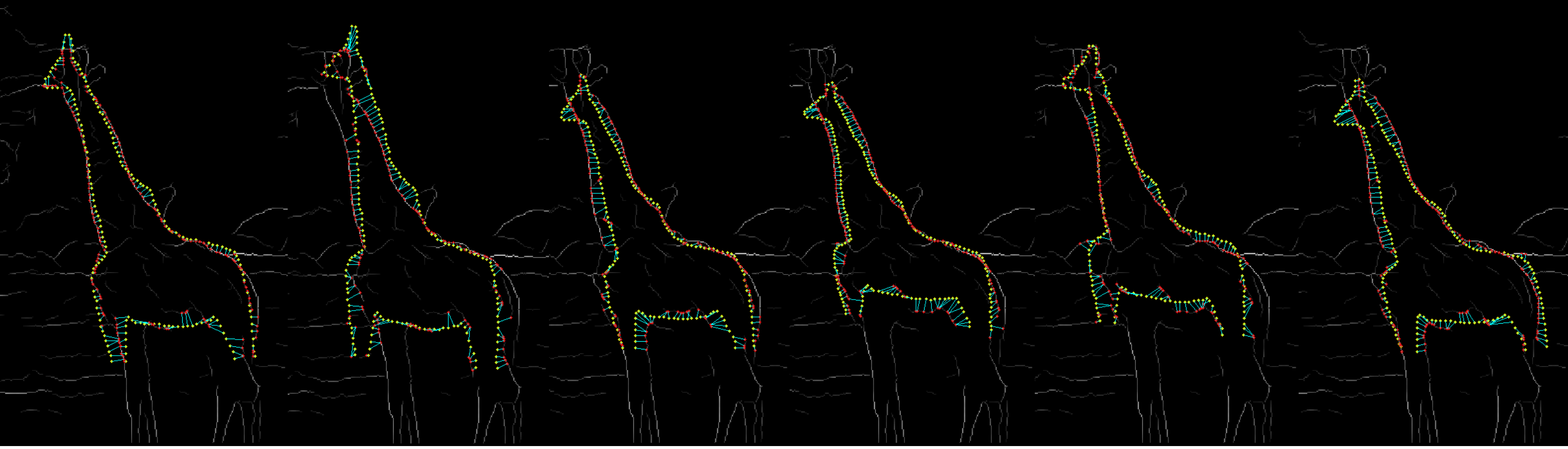


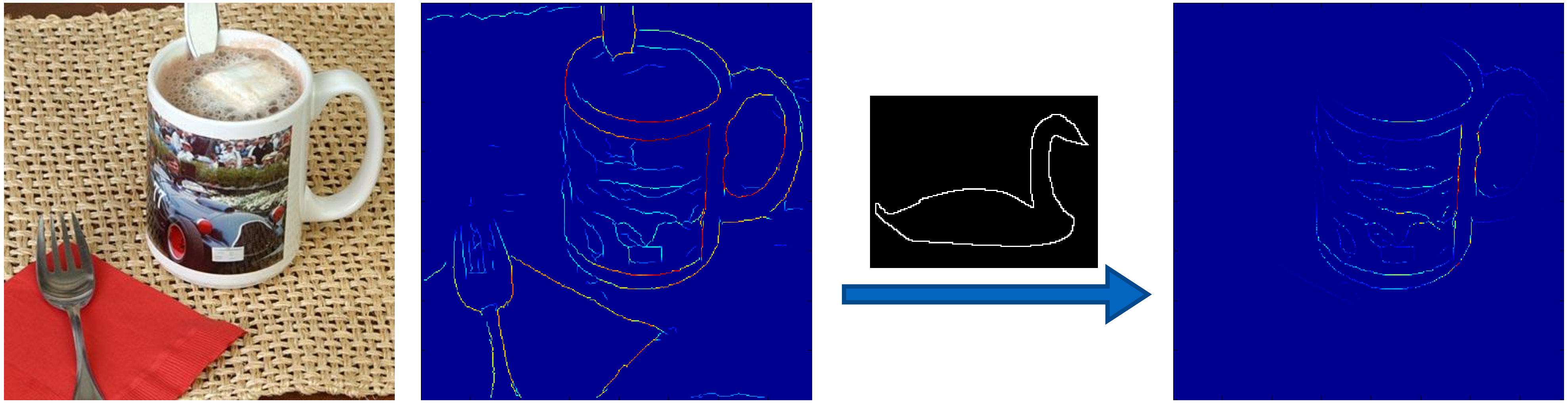
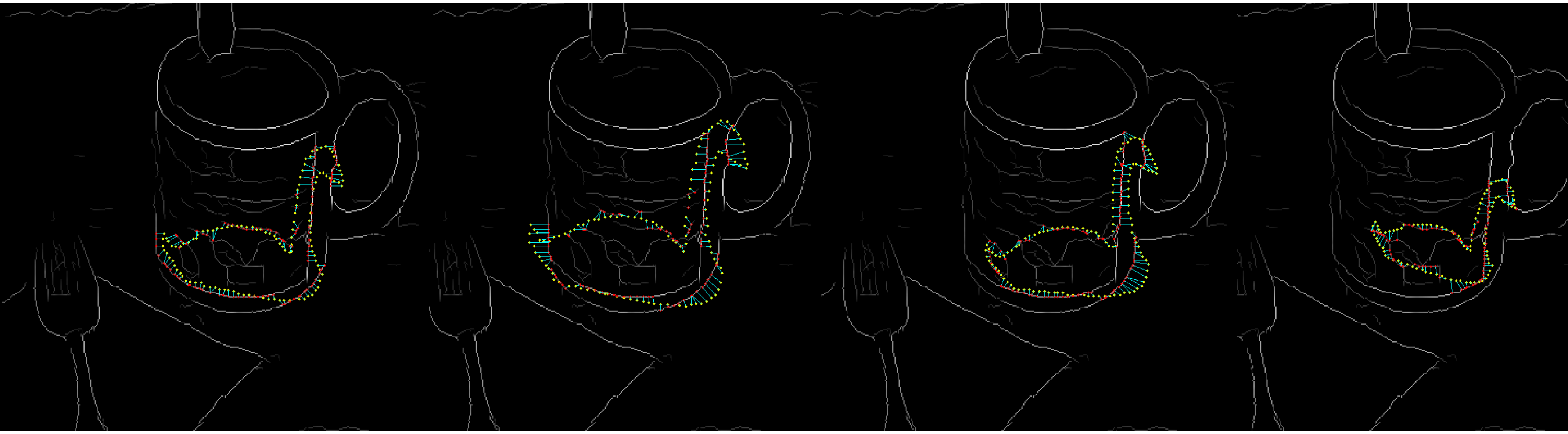


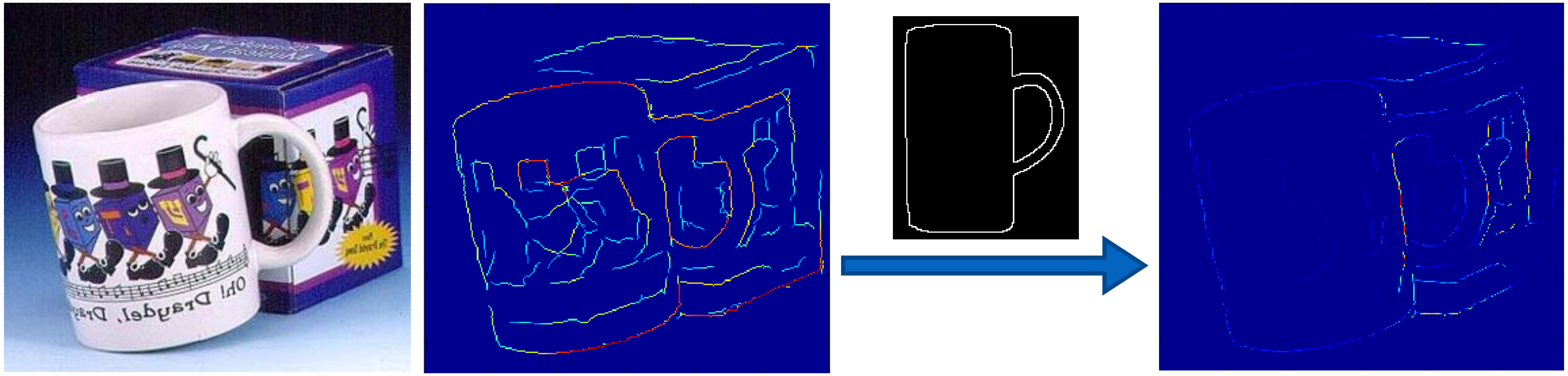










What results in high chance of accidental alignment?

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