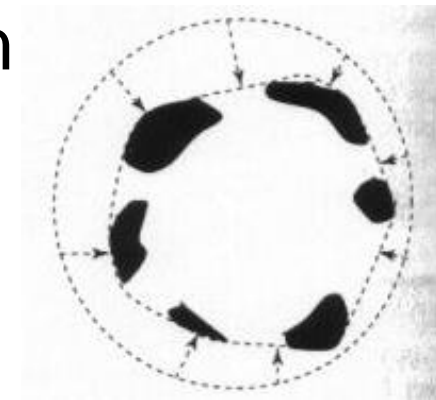


# TP11 - Fitting: Deformable contours

Computer Vision, FCUP, 2018/19

Miguel Coimbra

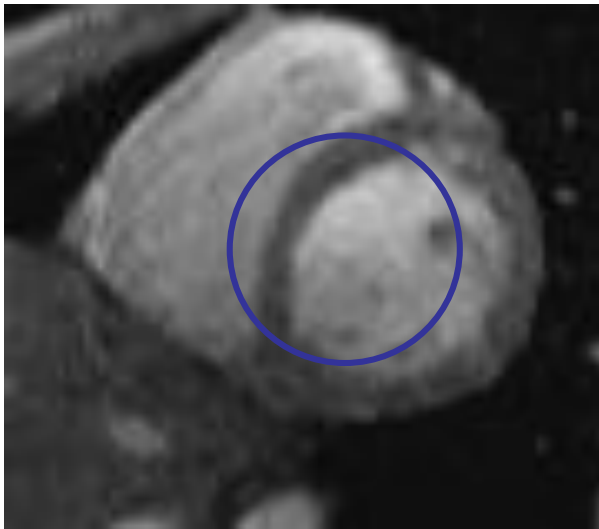
Slides by Prof. Kristen Grauman



# Deformable contours

a.k.a. active contours, snakes

**Given:** initial contour (model) near desired object

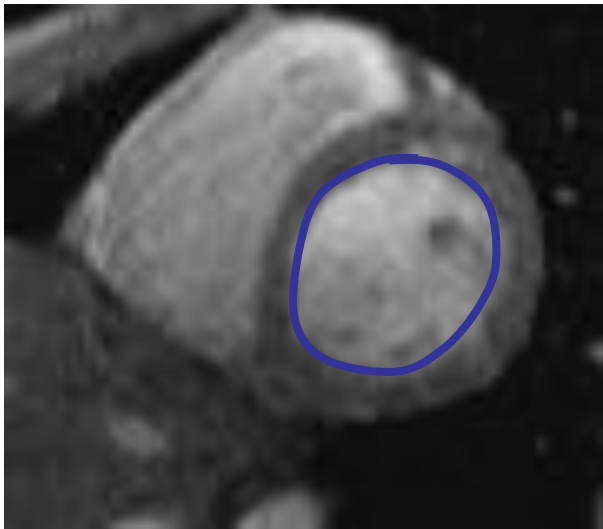


# Deformable contours

a.k.a. active contours, snakes

**Given:** initial contour (model) near desired object

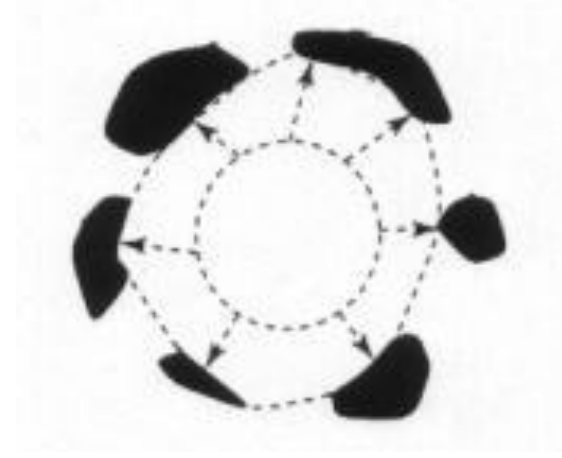
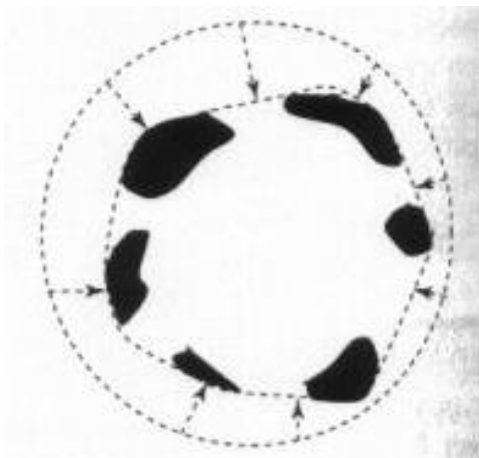
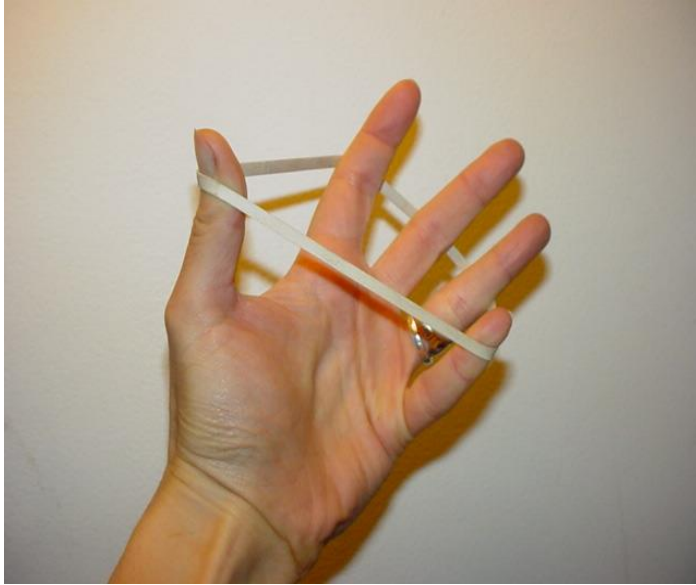
**Goal:** evolve the contour to fit exact object boundary



**Main idea:** elastic band is iteratively adjusted so as to

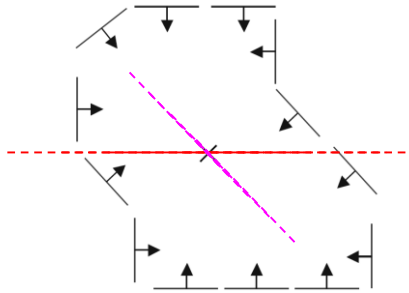
- be near image positions with high gradients, **and**
- satisfy shape “preferences” or contour priors

# Deformable contours: intuition



# Deformable contours vs. Hough

Like generalized Hough transform, useful for shape fitting; but



## Hough

Rigid model shape

Single voting pass can detect multiple instances

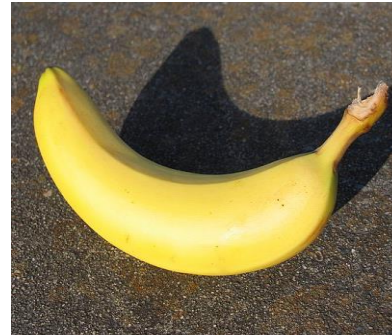
## Deformable contours

Prior on shape types, but shape iteratively adjusted (*deforms*)

Requires initialization nearby

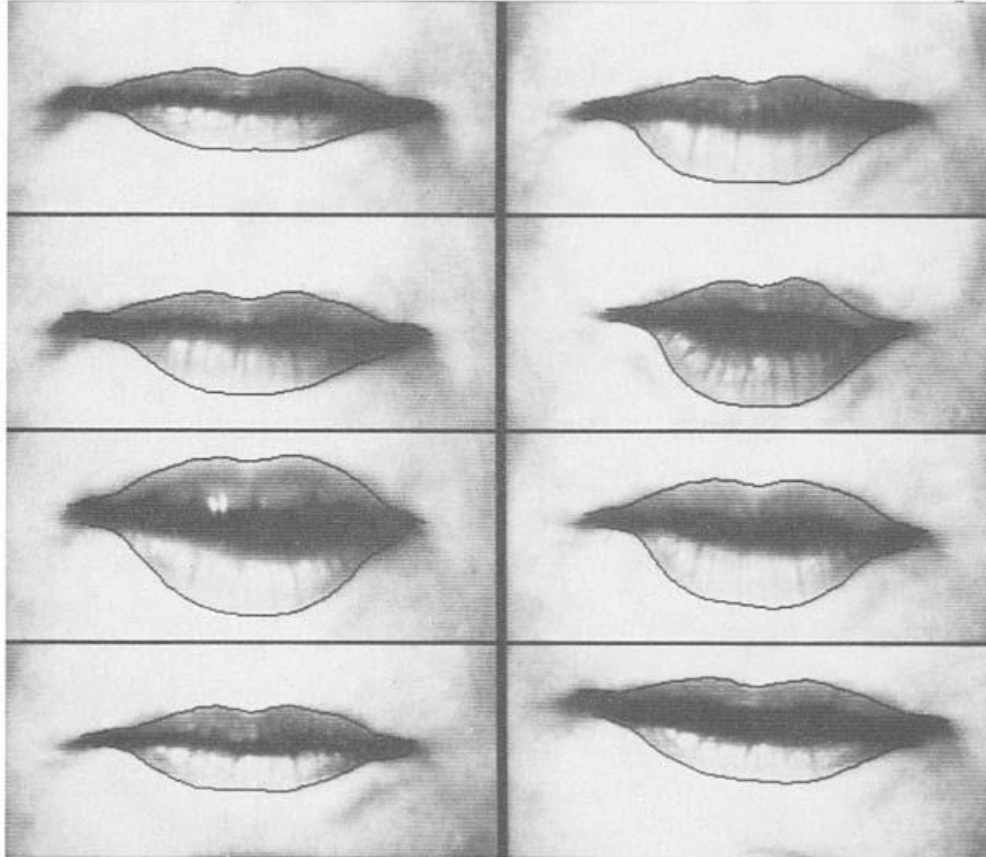
One optimization “pass” to fit a single contour

# Why do we want to fit deformable shapes?



- Some objects have similar basic form but some variety in the contour shape.

# Why do we want to fit deformable shapes?



- Non-rigid, deformable objects can change their shape over time, e.g. lips, hands...

# Why do we want to fit deformable shapes?

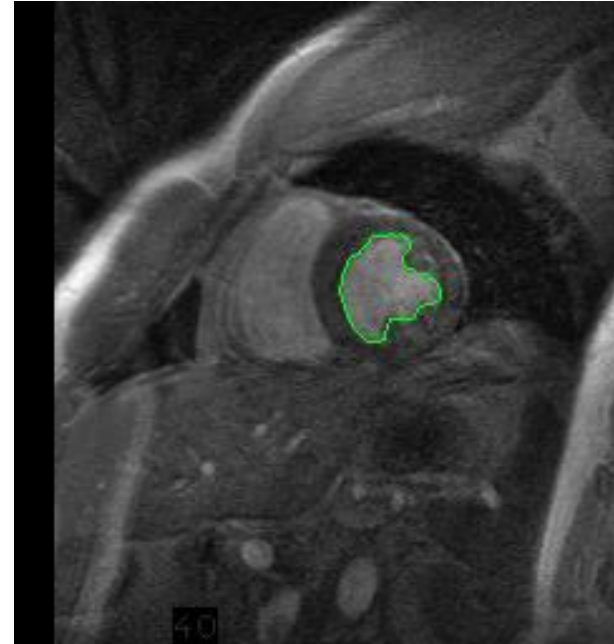
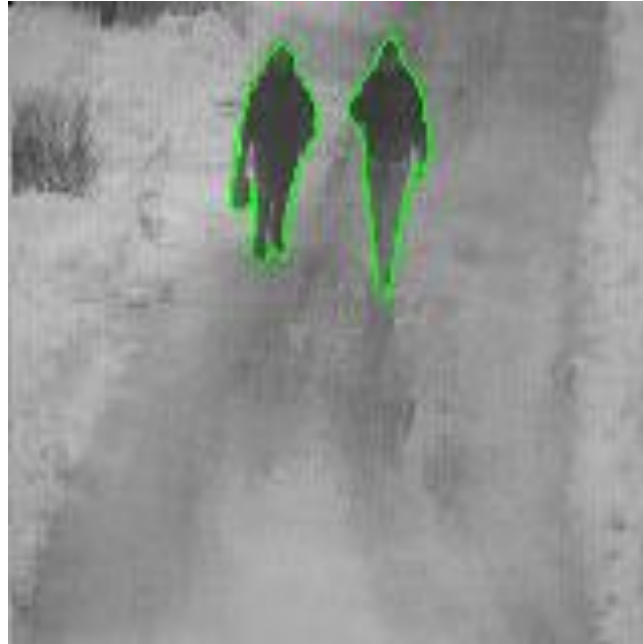
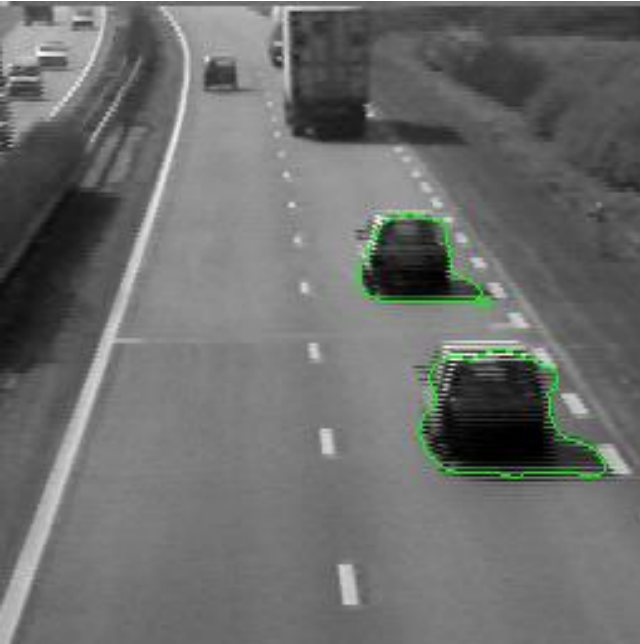


- Non-rigid, deformable objects can change their shape over time, e.g. lips, hands...





# Why do we want to fit deformable shapes?



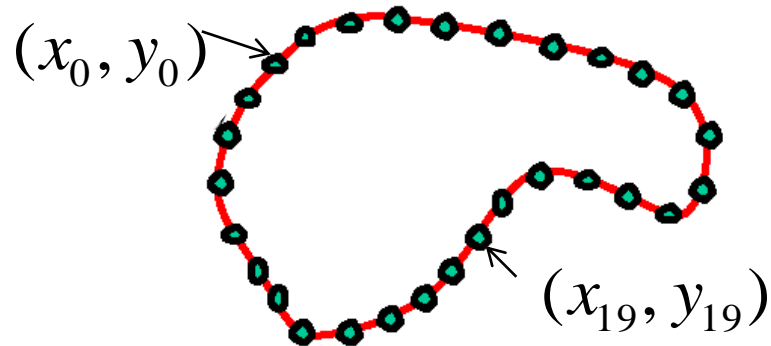
- Non-rigid, deformable objects can change their shape over time.

# Aspects we need to consider

- Representation of the contours
- Defining the energy functions
  - External
  - Internal
- Minimizing the energy function
- Extensions:
  - Tracking
  - Interactive segmentation

# Representation

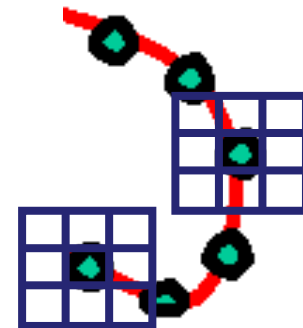
- We'll consider a discrete representation of the contour, consisting of a list of 2d point positions (“vertices”).



$$v_i = (x_i, y_i),$$

$$\text{for } i = 0, 1, \dots, n - 1$$

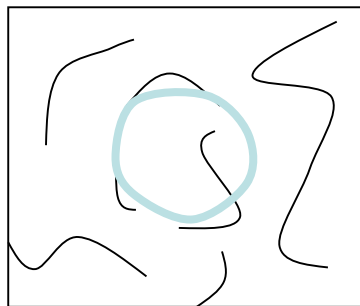
- At each iteration, we'll have the option to move each vertex to another nearby location (“state”).



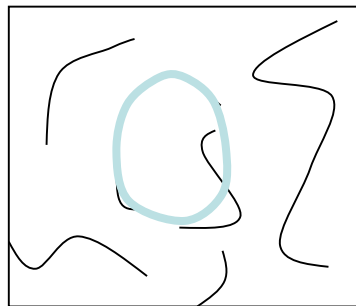
# Fitting deformable contours

How should we adjust the current contour to form the new contour at each iteration?

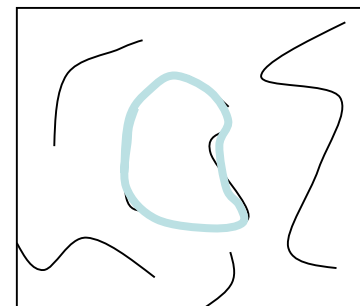
- Define a cost function (“energy” function) that says how good a candidate configuration is.
- Seek next configuration that minimizes that cost function.



initial



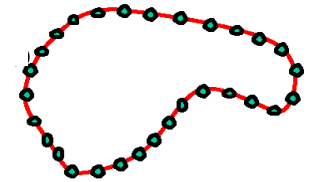
intermediate



final

# Energy function

The total energy (cost) of the current snake is defined as:



$$E_{total} = E_{internal} + E_{external}$$

**Internal** energy: encourage *prior* shape preferences: e.g., smoothness, elasticity, particular known shape.

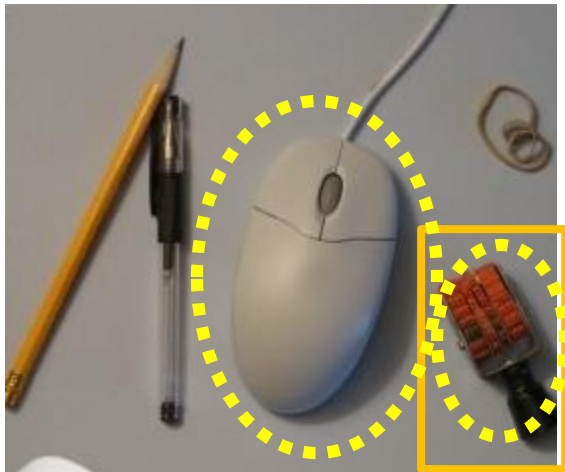
**External** energy (“image” energy): encourage contour to fit on places where image structures exist, e.g., edges.

A good fit between the current deformable contour and the target shape in the image will yield a **low** value for this cost function.

# External energy: intuition

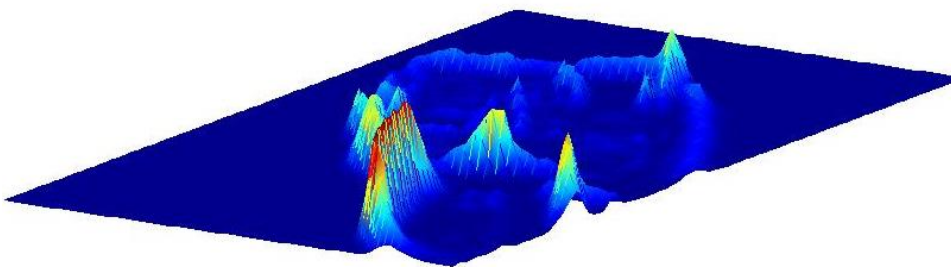
- Measure how well the curve matches the image data
- “Attract” the curve toward different image features
  - Edges, lines, texture gradient, etc.

# External image energy



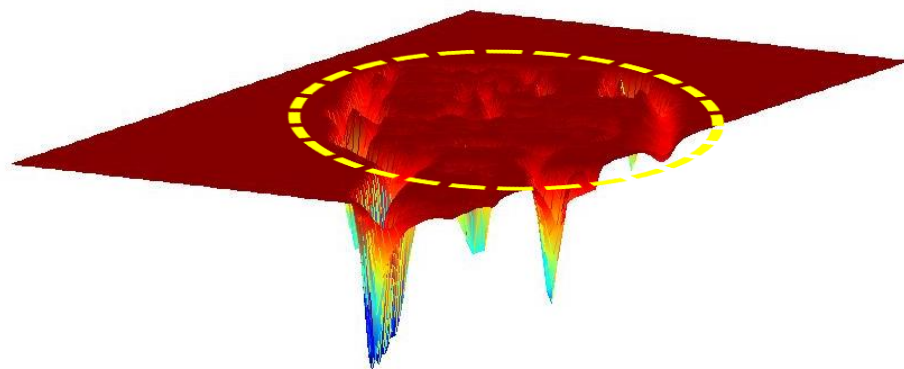
How do edges affect “snap” of rubber band?

Think of external energy from image as gravitational pull towards areas of high contrast



Magnitude of gradient

$$G_x(I)^2 + G_y(I)^2$$

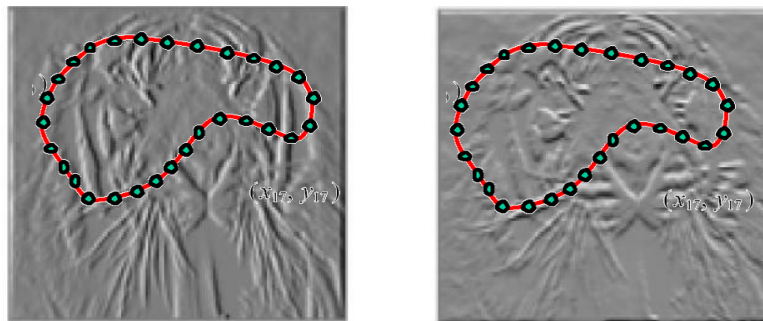


- (Magnitude of gradient)

$$-\left(G_x(I)^2 + G_y(I)^2\right)$$

# External image energy

- Gradient images  $G_x(x, y)$  and  $G_y(x, y)$



- External energy at a point on the curve is:

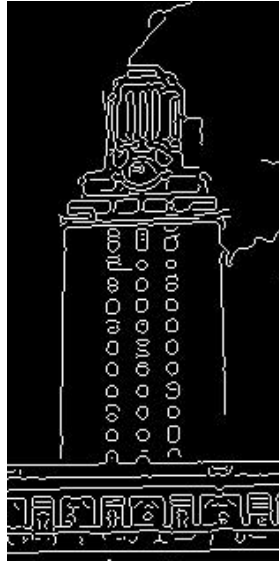
$$E_{external}(v) = -(|G_x(v)|^2 + |G_y(v)|^2)$$

- External energy for the whole curve:

$$E_{external} = - \sum_{i=0}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2$$



# Internal energy: intuition



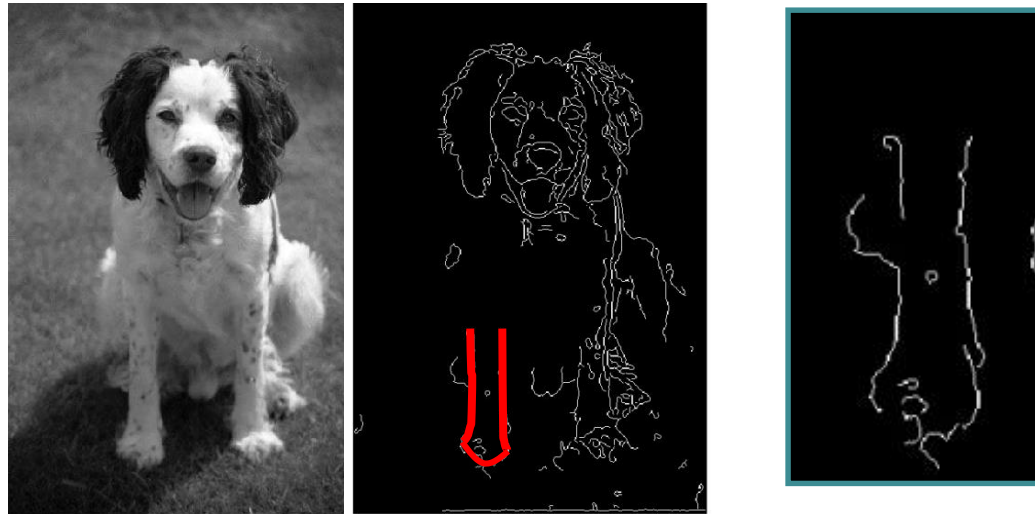
What are the underlying boundaries in this fragmented edge image?



And in this one?

# Internal energy: intuition

*A priori*, we want to favor **smooth** shapes, contours with **low curvature**, contours similar to a **known shape**, etc. to balance what is actually observed (i.e., in the gradient image).



# Internal energy

For a *continuous* curve, a common internal energy term is the “bending energy”.

At some point  $v(s)$  on the curve, this is:

$$E_{internal}(v(s)) = \alpha \left| \frac{dv}{ds} \right|^2 + \beta \left| \frac{d^2v}{d^2s} \right|^2$$

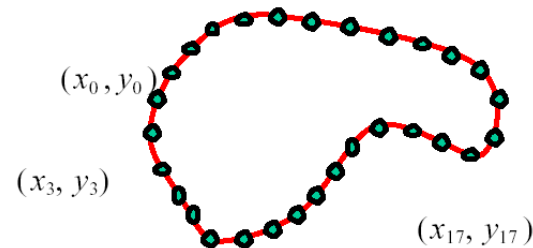
Tension,  
Elasticity

Stiffness,  
Curvature



# Internal energy

- For our discrete representation,



$$\mathbf{v}_i = (x_i, y_i) \quad i = 0 \dots n-1$$

$$\frac{d\mathbf{v}}{ds} \approx \mathbf{v}_{i+1} - \mathbf{v}_i \quad \frac{d^2\mathbf{v}}{ds^2} \approx (\mathbf{v}_{i+1} - \mathbf{v}_i) - (\mathbf{v}_i - \mathbf{v}_{i-1}) = \mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}$$


- Note these are derivatives relative to **position**---not spatial image gradients.

$$E_{internal} = \sum_{i=0}^{n-1} \alpha \|\mathbf{v}_{i+1} - \mathbf{v}_i\|^2 + \beta \|\mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}\|^2$$

Why do these reflect **tension** and **curvature**?

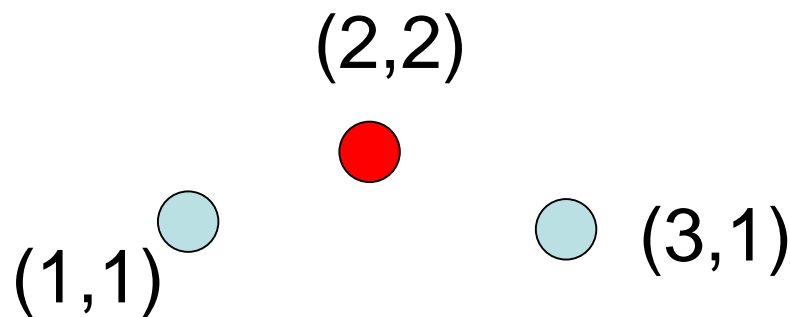
# Example: compare curvature

$$\begin{aligned} E_{\text{curvature}}(v_i) &= \|v_{i+1} - 2v_i + v_{i-1}\|^2 \\ &= (x_{i+1} - 2x_i + x_{i-1})^2 + (y_{i+1} - 2y_i + y_{i-1})^2 \end{aligned}$$

 (2,5)



$$\begin{aligned} (3 - 2(2) + 1)^2 + (1 - 2(5) + 1)^2 \\ = (-8)^2 = 64 \end{aligned}$$

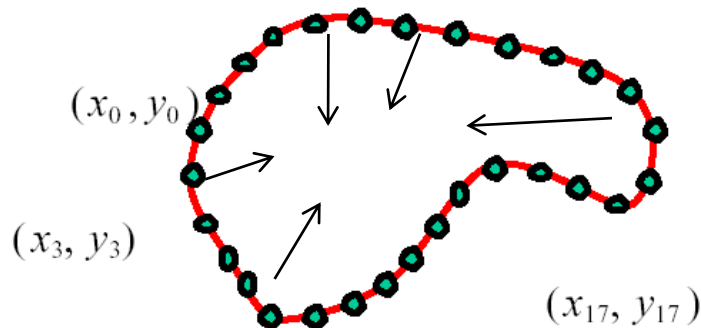


$$\begin{aligned} (3 - 2(2) + 1)^2 + (1 - 2(2) + 1)^2 \\ = (-2)^2 = 4 \end{aligned}$$

# Penalizing elasticity

- Current elastic energy definition uses a discrete estimate of the derivative:

$$\begin{aligned} E_{elastic} &= \sum_{i=0}^{n-1} \alpha \left\| \mathbf{v}_{i+1} - \mathbf{v}_i \right\|^2 \\ &= \alpha \cdot \sum_{i=0}^{n-1} (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 \end{aligned}$$



*What is the possible problem with this definition?*

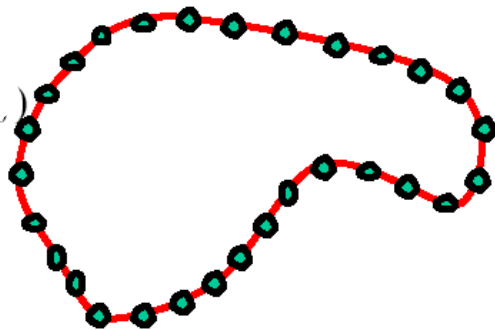
# Penalizing elasticity

- Current elastic energy definition uses a discrete estimate of the derivative:

$$E_{elastic} = \sum_{i=0}^{n-1} \alpha \|v_{i+1} - v_i\|^2$$

Instead:

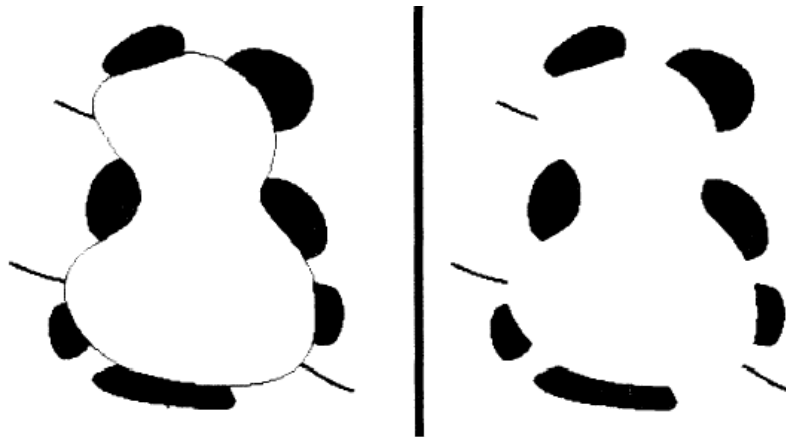
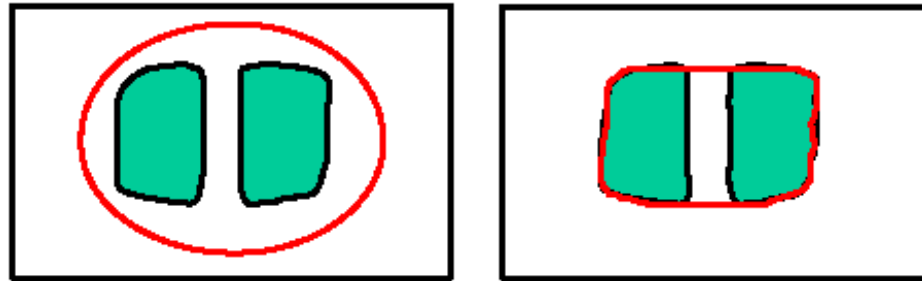
$$= \alpha \cdot \sum_{i=0}^{n-1} \left( (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 - \bar{d} \right)^2$$



where  $d$  is the average distance between pairs of points – updated at each iteration.

# Dealing with missing data

- The preferences for low-curvature, smoothness help deal with missing data:



Illusory contours found!

[Figure from Kass et al. 1987]

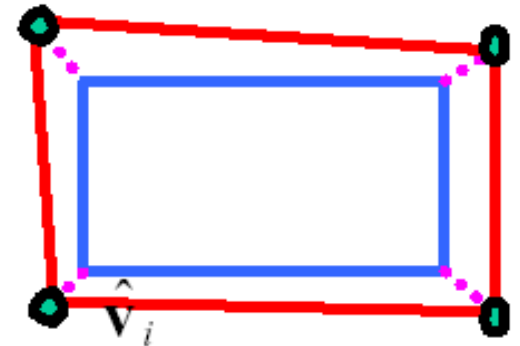


# Extending the internal energy: capture shape prior

- If object is some smooth variation on a known shape, we can use a term that will penalize deviation from that shape:

$$E_{internal}^+ = \alpha \cdot \sum_{i=0}^{n-1} (v_i - \hat{v}_i)^2$$

where  $\{\hat{v}_i\}$  are the points of the known shape.



# Total energy: function of the weights

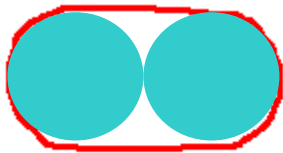
$$E_{total} = E_{internal} + \gamma E_{external}$$

$$E_{external} = - \sum_{i=0}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2$$

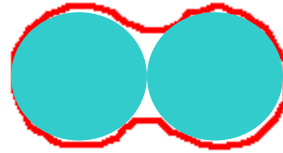
$$E_{internal} = \sum_{i=0}^{n-1} \alpha \left( \bar{d} - \|v_{i+1} - v_i\| \right)^2 + \beta \|v_{i+1} - 2v_i + v_{i-1}\|^2$$

# Total energy: function of the weights

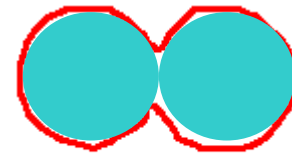
- e.g.,  $\alpha$  weight controls the penalty for internal elasticity



large  $\alpha$



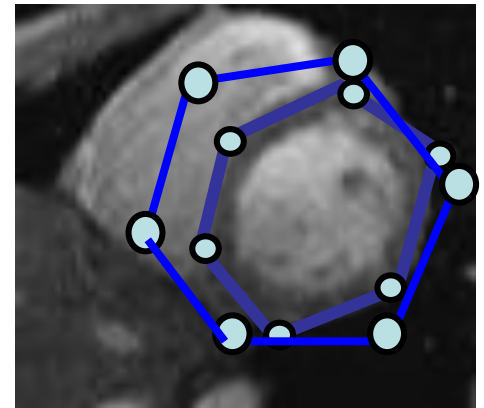
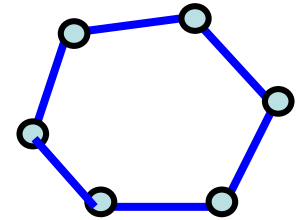
medium  $\alpha$



small  $\alpha$

# Recap: deformable contour

- A simple elastic snake is defined by:
  - A set of  $n$  points,
  - An internal energy term (tension, bending, plus optional shape prior)
  - An external energy term (gradient-based)
  
- To use to segment an object:
  - Initialize in the vicinity of the object
  - Modify the points to minimize the total energy

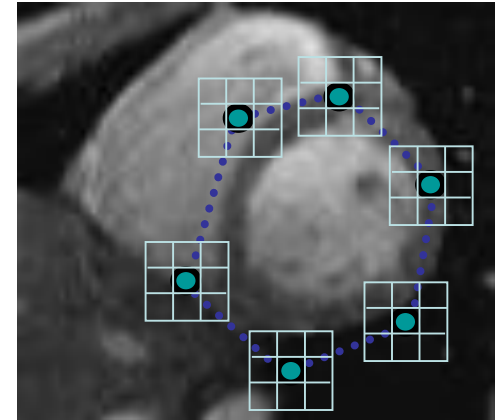


# Energy minimization

- Several algorithms have been proposed to fit deformable contours.
- We'll look at two:
  - Greedy search
  - Dynamic programming (for 2d snakes)

# Energy minimization: greedy

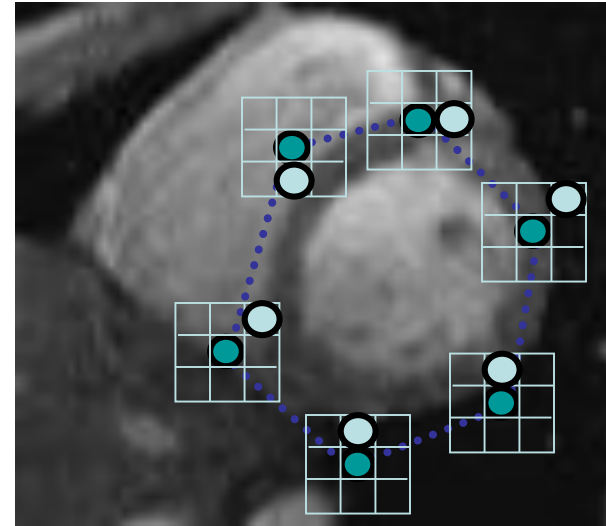
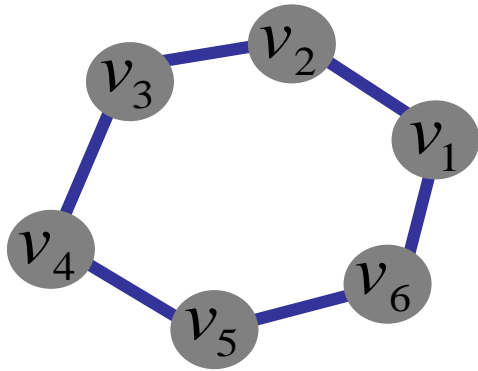
- For each point, search window around it and move to where energy function is minimal
  - Typical window size, e.g., 5 x 5 pixels
- Stop when predefined number of points have not changed in last iteration, or after max number of iterations
- Note:
  - Convergence not guaranteed
  - Need decent initialization



# Energy minimization

- Several algorithms have been proposed to fit deformable contours.
- We'll look at two:
  - Greedy search
  - Dynamic programming (for 2d snakes)

# Energy minimization: dynamic programming



With this form of the energy function, we can minimize using dynamic programming, with the *Viterbi* algorithm.

Iterate until optimal position for each point is the center of the box, i.e., the snake is optimal in the local search space constrained by boxes.



# Energy minimization: dynamic programming

- Possible because snake energy can be rewritten as a sum of pair-wise interaction potentials:

$$E_{total}(v_1, \dots, v_n) = \sum_{i=1}^{n-1} E_i(v_i, v_{i+1})$$

- Or sum of triple-interaction potentials.

$$E_{total}(v_1, \dots, v_n) = \sum_{i=1}^{n-1} E_i(v_{i-1}, v_i, v_{i+1})$$

# Snake energy: pair-wise interactions

$$E_{total}(x_1, \dots, x_n, y_1, \dots, y_n) = - \sum_{i=1}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2 \\ + \alpha \cdot \sum_{i=1}^{n-1} (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2$$

Re-writing the above with  $v_i = (x_i, y_i)$  :

$$E_{total}(v_1, \dots, v_n) = - \sum_{i=1}^{n-1} \|G(v_i)\|^2 + \alpha \cdot \sum_{i=1}^{n-1} \|v_{i+1} - v_i\|^2$$

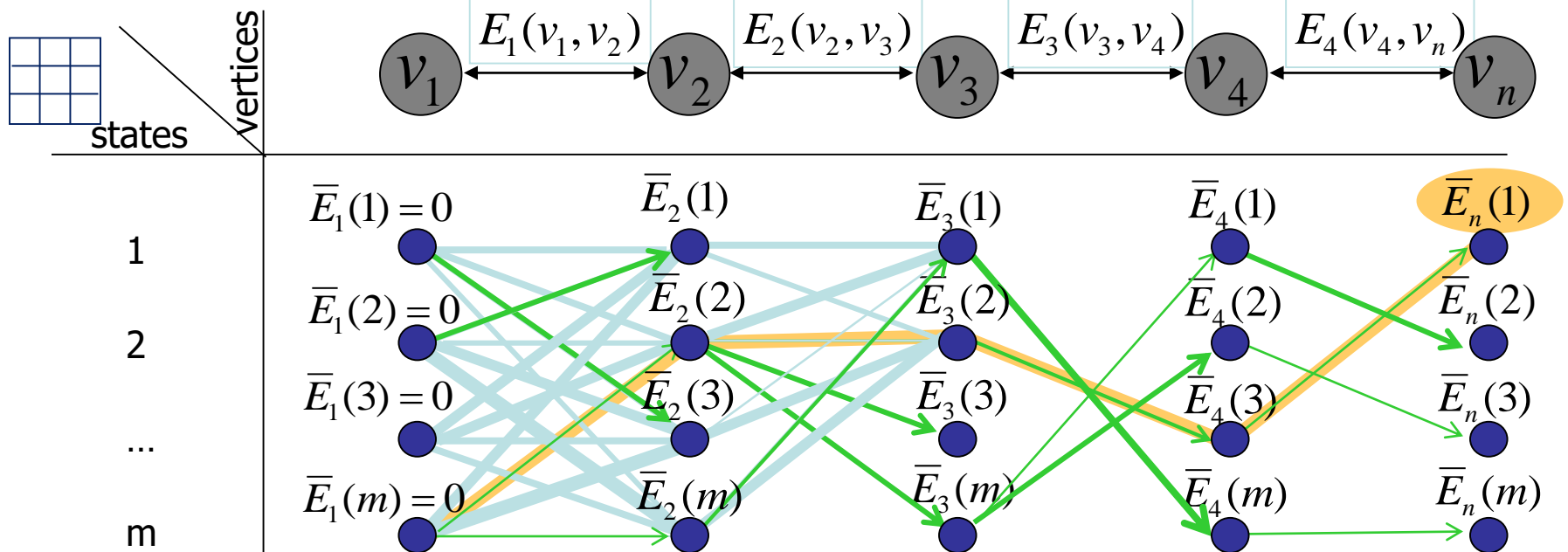
$$E_{total}(v_1, \dots, v_n) = E_1(v_1, v_2) + E_2(v_2, v_3) + \dots + E_{n-1}(v_{n-1}, v_n)$$

where  $E_i(v_i, v_{i+1}) = -\|G(v_i)\|^2 + \alpha \|v_{i+1} - v_i\|^2$

# Viterbi algorithm

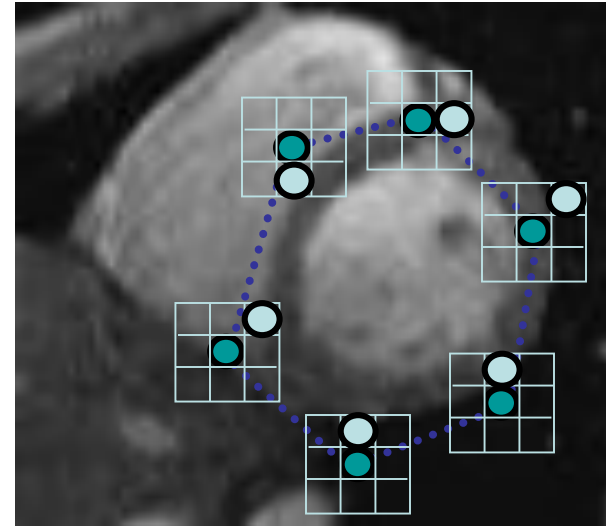
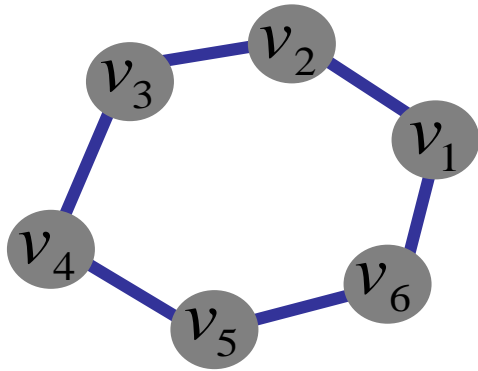
Main idea: determine optimal position (state) of predecessor, for each possible position of self. Then backtrack from best state for last vertex.

$$E_{total} = E_1(v_1, v_2) + E_2(v_2, v_3) + \dots + E_{n-1}(v_{n-1}, v_n)$$



Complexity:  $O(nm^2)$  vs. brute force search \_\_\_\_\_?

# Energy minimization: dynamic programming



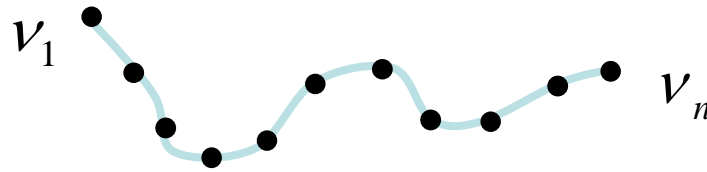
With this form of the energy function, we can minimize using dynamic programming, with the *Viterbi* algorithm.

Iterate until optimal position for each point is the center of the box, i.e., the snake is optimal in the local search space constrained by boxes.

# Energy minimization: dynamic programming

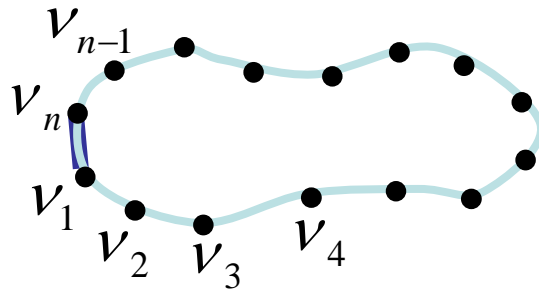
DP can be applied to optimize an open ended snake

$$E_1(v_1, v_2) + E_2(v_2, v_3) + \dots + E_{n-1}(v_{n-1}, v_n)$$



For a closed snake, a “loop” is introduced into the total energy.

$$E_1(v_1, v_2) + E_2(v_2, v_3) + \dots + E_{n-1}(v_{n-1}, v_n) + E_n(v_n, v_1)$$



Work around:

- 1) Fix  $v_1$  and solve for rest .
- 2) Fix an intermediate node at its position found in (1), solve for rest.

# Aspects we need to consider

- Representation of the contours
- Defining the energy functions
  - External
  - Internal
- Minimizing the energy function
- Extensions:
  - Tracking
  - Interactive segmentation

# Tracking via deformable contours

1. Use final contour/model extracted at frame  $t$  as an initial solution for frame  $t+1$
2. Evolve initial contour to fit exact object boundary at frame  $t+1$
3. Repeat, initializing with most recent frame.



Tracking Heart Ventricles  
(multiple frames)

# Tracking via deformable contours



[Visual Dynamics Group](#), Dept. Engineering Science, University of Oxford.

- Applications:
- Traffic monitoring
  - Human-computer interaction
  - Animation
  - Surveillance
  - Computer assisted diagnosis in medical imaging

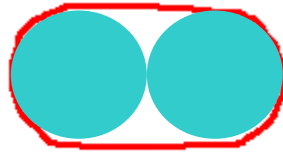


# 3D active contours



# Limitations

- May over-smooth the boundary



- Cannot follow topological changes of objects



# Limitations

- External energy: snake does not really “see” object boundaries in the image unless it gets very close to it.

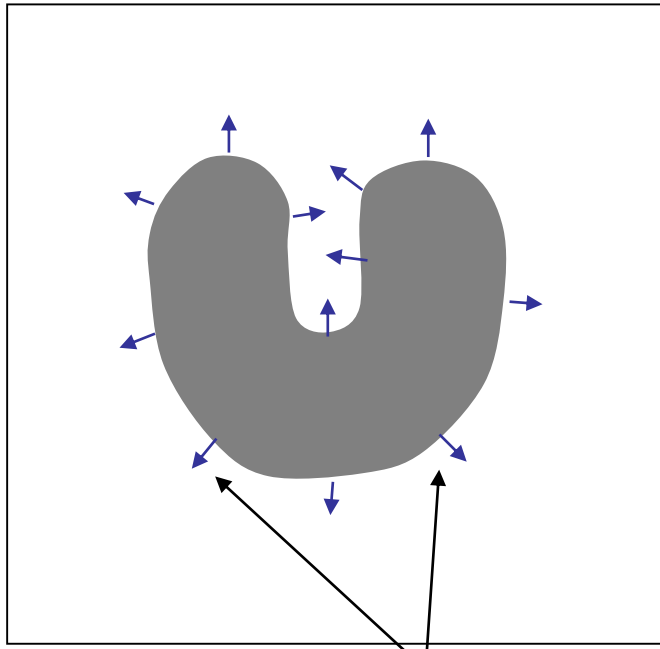
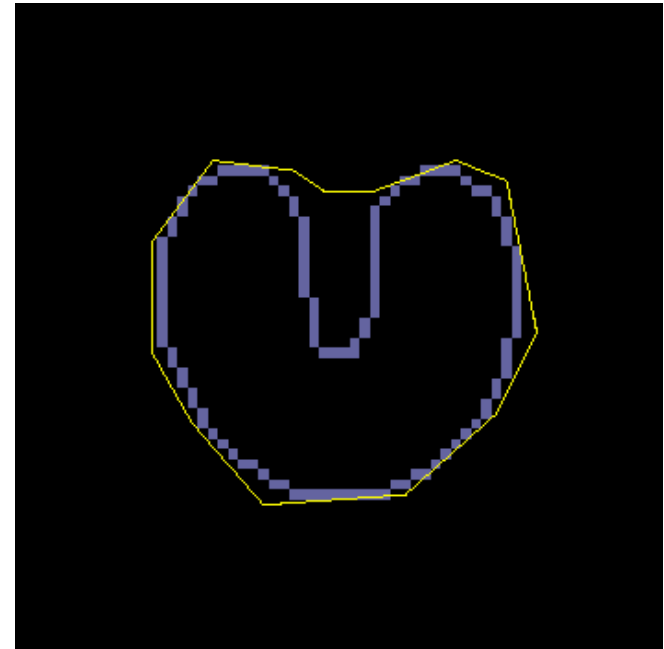


image gradients  $\nabla I$   
are large only directly on the boundary



# Distance transform

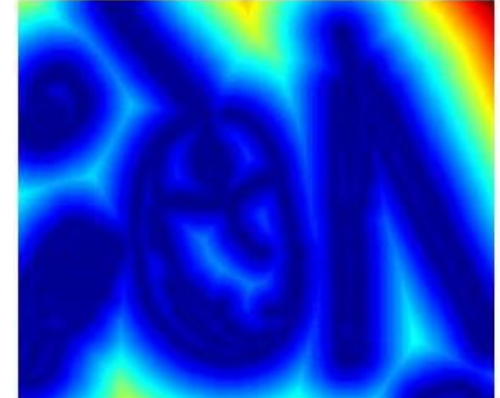
- External image can instead be taken from the **distance transform** of the edge image.



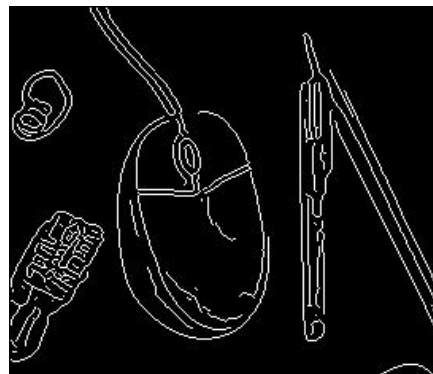
original



-gradient



distance transform



edges

Value at  $(x,y)$  tells how far that position is from the nearest edge point (or other binary image structure)

```
>> help bwdist
```

# Deformable contours: pros and cons

## Pros:

- Useful to track and fit non-rigid shapes
- Contour remains connected
- Possible to fill in “subjective” contours
- Flexibility in how energy function is defined, weighted.

## Cons:

- Must have decent initialization near true boundary, may get stuck in local minimum
- Parameters of energy function must be set well based on prior information

# Summary

- Deformable shapes and active contours are useful for
  - Segmentation: fit or “snap” to boundary in image
  - Tracking: previous frame’s estimate serves to initialize the next
- Fitting active contours:
  - Define terms to encourage certain shapes, smoothness, low curvature, push/pulls, ...
  - Use weights to control relative influence of each component cost
  - Can optimize 2d snakes with Viterbi algorithm.
- Image structure (esp. gradients) can act as attraction force for *interactive* segmentation methods.