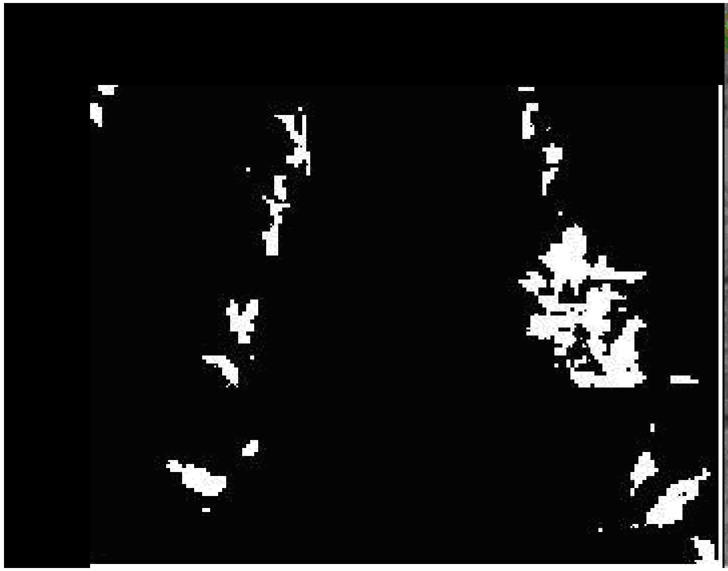
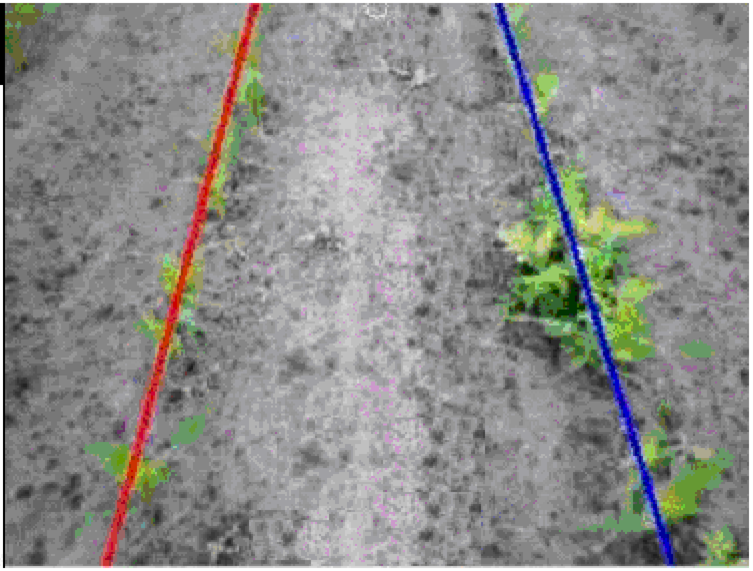
Edge Linking

Example



Edge points



Strongest lines

Lane Detection and Departure Warning

University of California Santa Barbara

Edge Linking Rationale

- Edge maps are still in an *image* format
- *Image* to *data structure* transform
- Two issues
  - *Identity*: there are so many edge points, which ones should be grouped together?
  - *Representation*: now that a group of edge pixels are identified, how best to represent them?

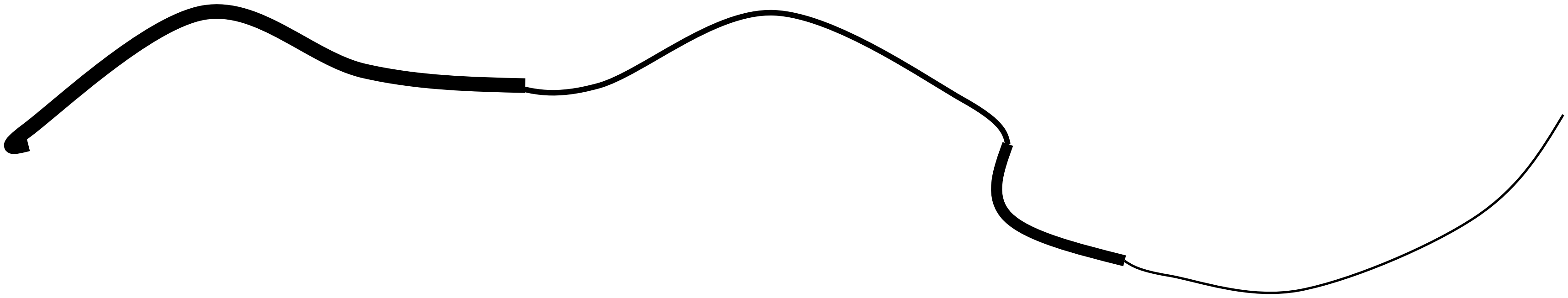
The Canny edge detector

Problem: pixels along this edge didn't survive the thresholding

thinning
(non-maximum suppression)

Hysteresis thresholding

- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use **hysteresis**
    - use a high threshold to start edge curves and a low threshold to continue them.



Hysteresis thresholding

original image

high threshold
(strong edges)

low threshold
(weak edges)

hysteresis threshold

Object boundaries vs. edges

Background

Texture

Shadows

Edge detection is just the beginning…

image | human segmentation | gradient magnitude

Berkeley segmentation database:

http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Much more on segmentation later in term…

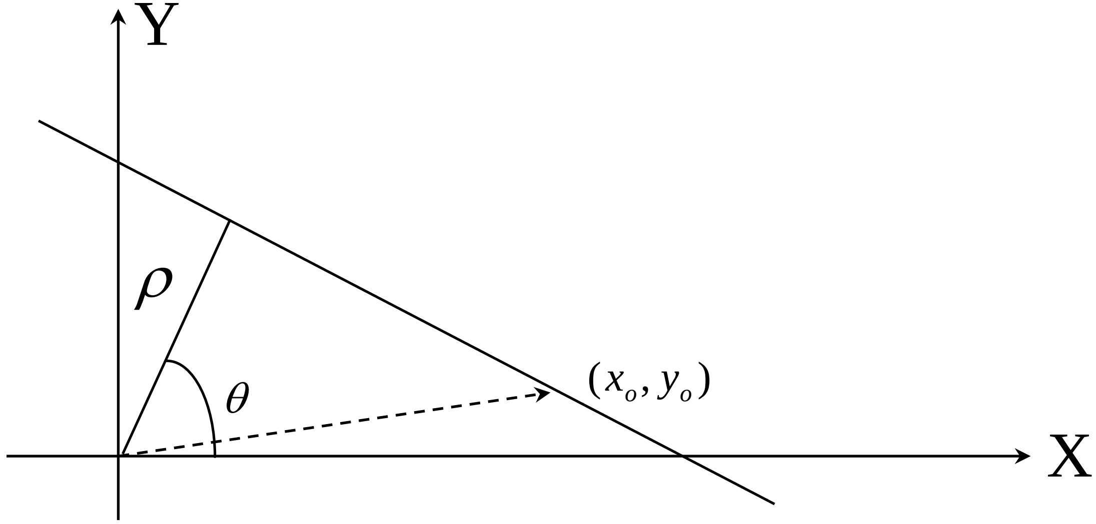
Source: L. Lazebnik

Identity

- Measurement space clustering
  - curve fitting
  - *global* technique
- Image space grouping
  - tracing or following
  - with known templates
  - *local* technique

Intuition

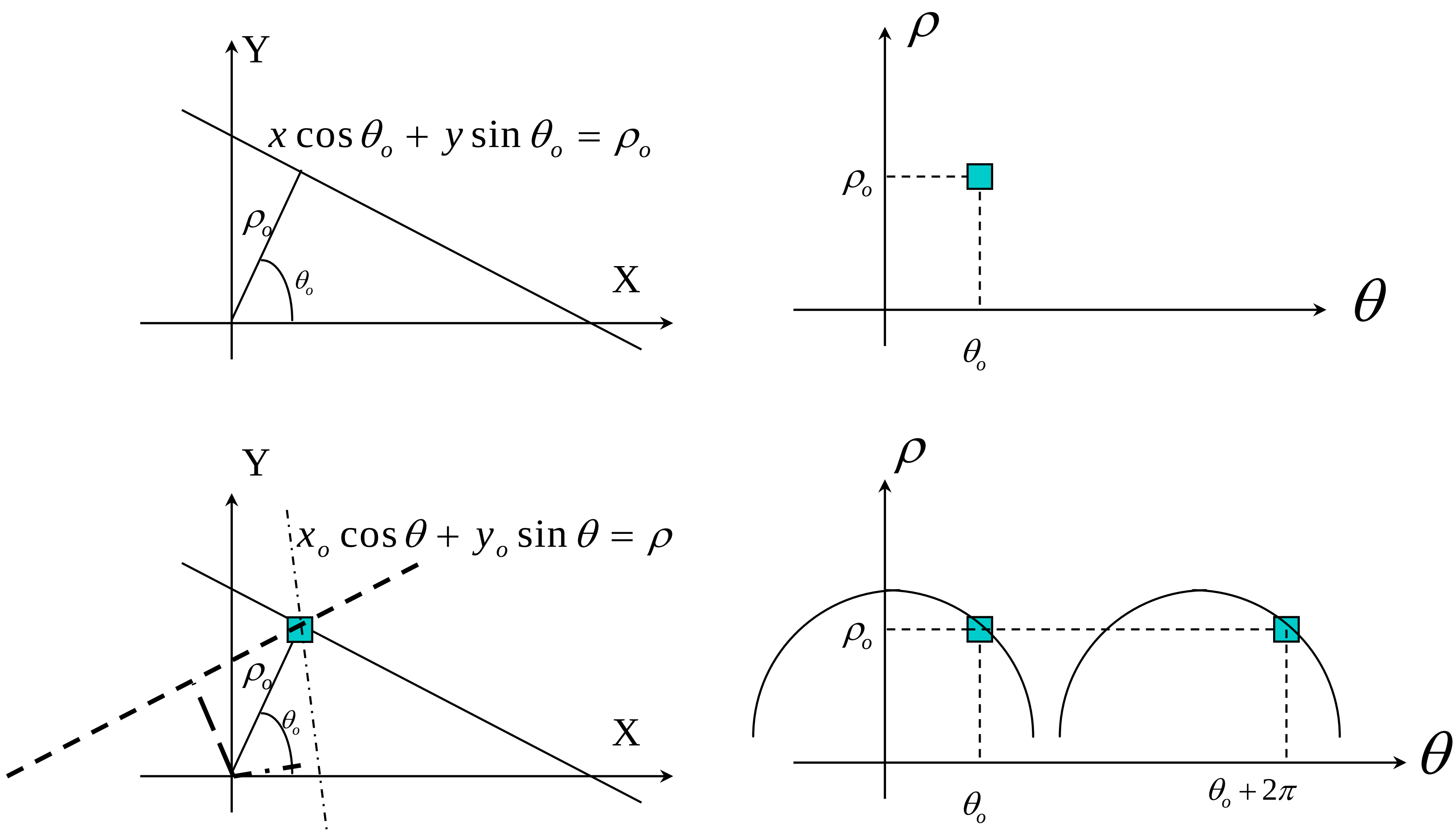
- Q: If several points fall on the same line, what "commonality" is there?



$$(x_o, y_o) \cdot (\cos\theta, \sin\theta) = \rho$$
$$x_o \cos\theta + y_o \sin\theta = \rho$$

Measurement Space Clustering

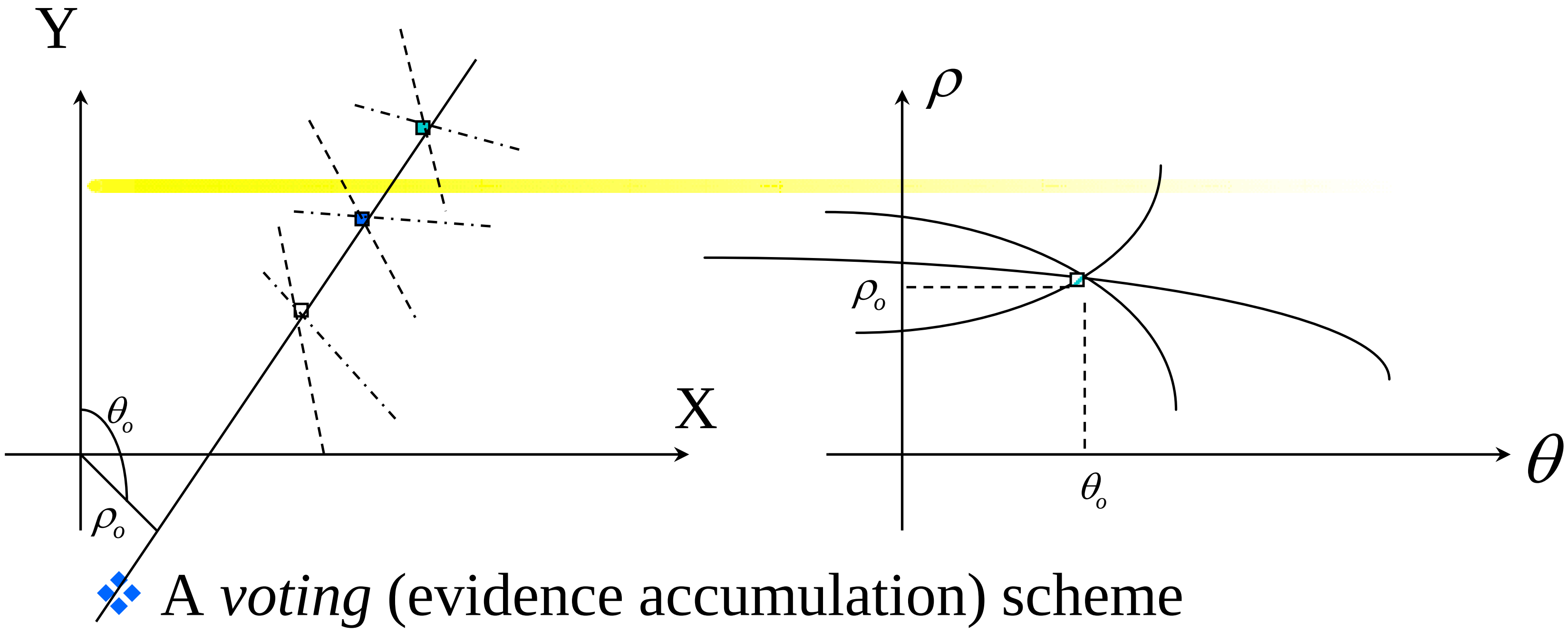
- Example: Hough transform



Duality of Representation

- Image space
  - a line
  - a point
- Measurement space
  - a point
  - a sinusoidal curve

$$
\begin{aligned}
\rho &= x_o \cos\theta + y_o \sin\theta \\
&= \sqrt{x_o^2 + y_o^2}\left(\frac{x_o}{\sqrt{x_o^2 + y_o^2}}\cos\theta + \frac{y_o}{\sqrt{x_o^2 + y_o^2}}\sin\theta\right) \\
&= \sqrt{x_o^2 + y_o^2}\cos(\theta - \alpha) \\
& where\ \alpha = \tan^{-1}\frac{y_o}{x_o}
\end{aligned}
$$



- A *voting* (evidence accumulation) scheme
- A point votes for all lines it is on
- All points (on a single line) vote for the single line they are on
- Tolerate a certain degree of occlusion
- Must know the parametric form

Hough Transform Algorithm

- Select a parametric form
- Quantize measurement space
- For each edge pixel, increment all cells satisfying the parametric form
- Locate maximum in the measurement space

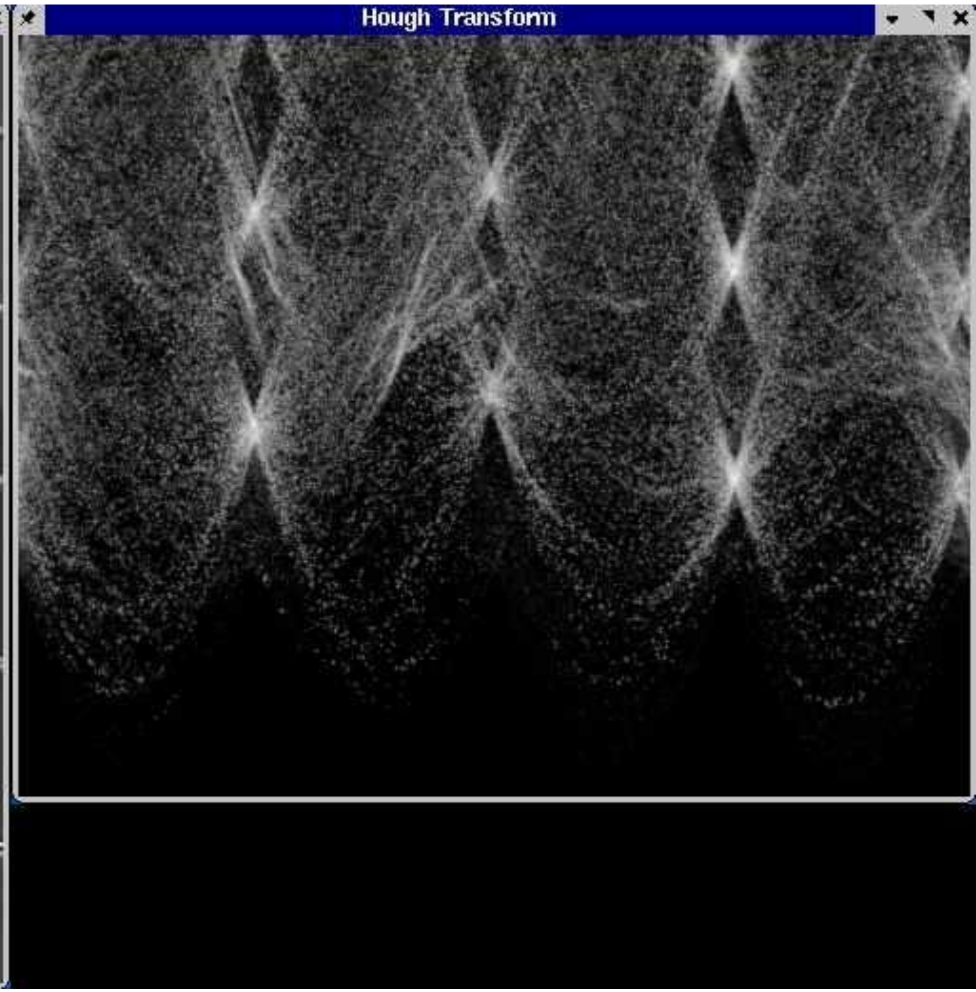
$$\rho - \theta$$

$$\theta : \min : 0^o, \max : 359^o, inc : 1^o$$
$$\rho : \min : 0, \max : N\sqrt{2}, inc : 1\,\text{pxl}$$

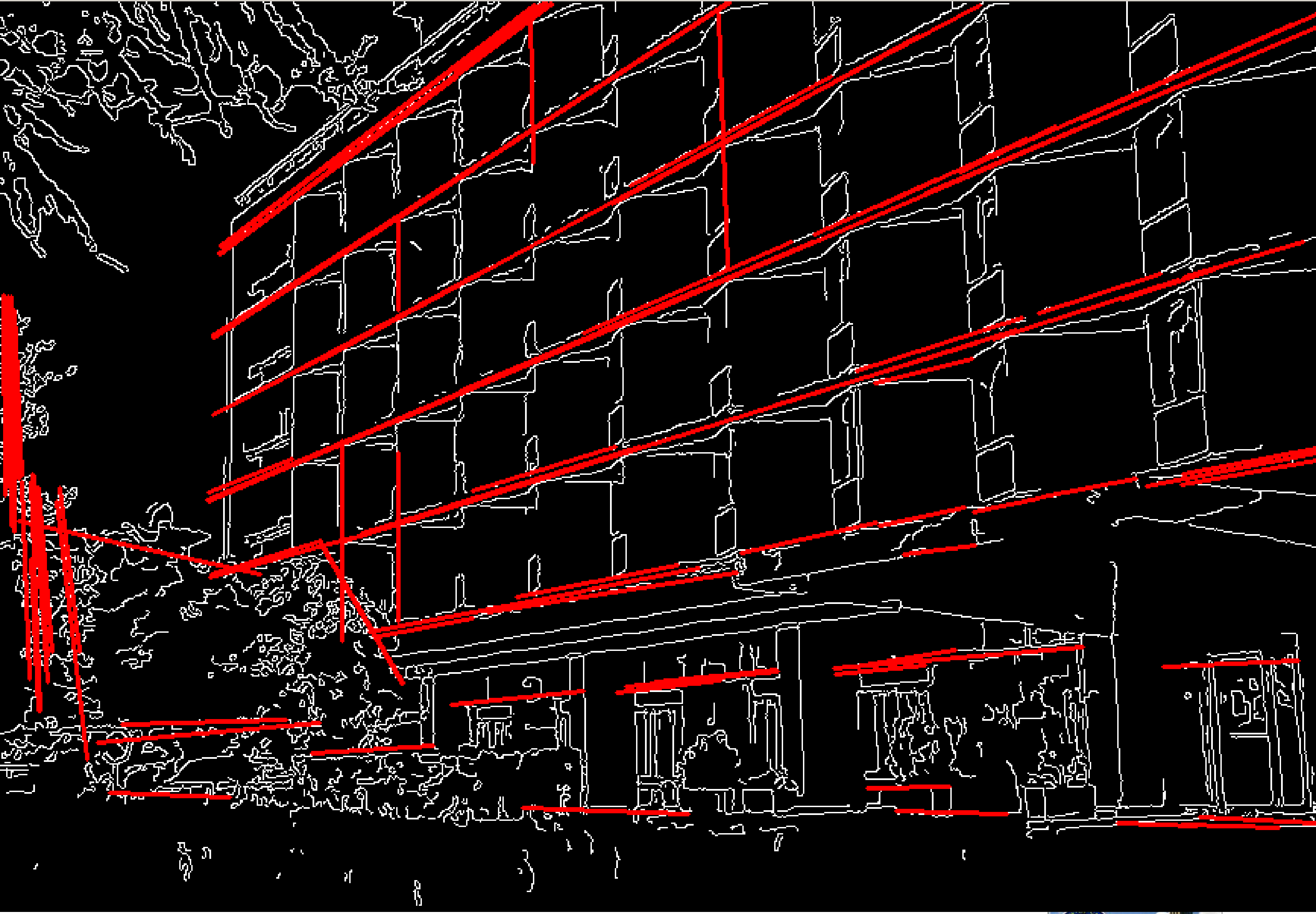
$$\text{for } \theta = 0 \text{ to } 360 \text{ inc } 1$$
$$\quad \rho = x_o \cos\theta + y_o \sin\theta$$
$$\quad (\rho, \theta) + +$$
$$\text{end}$$

Example

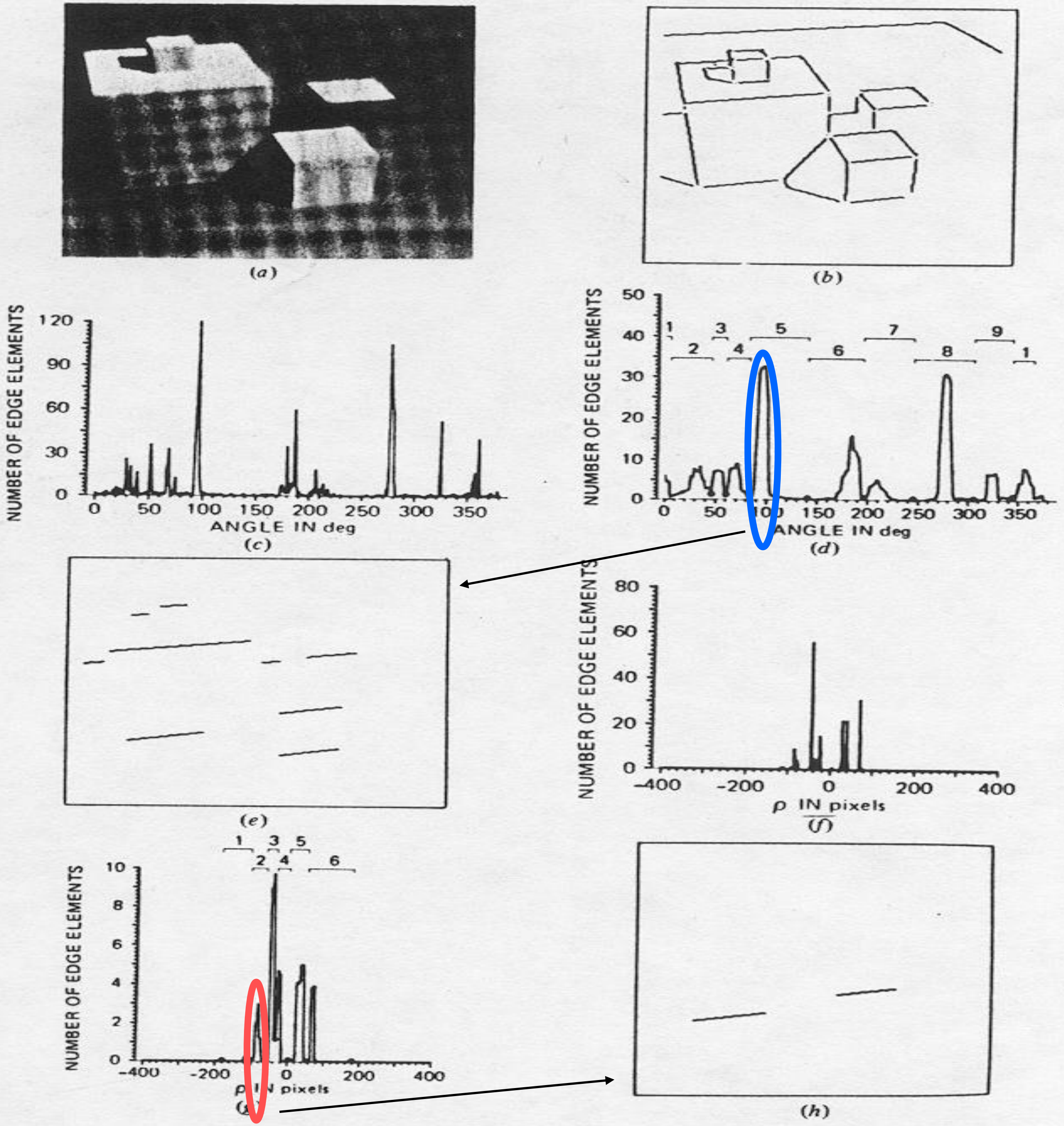
Image



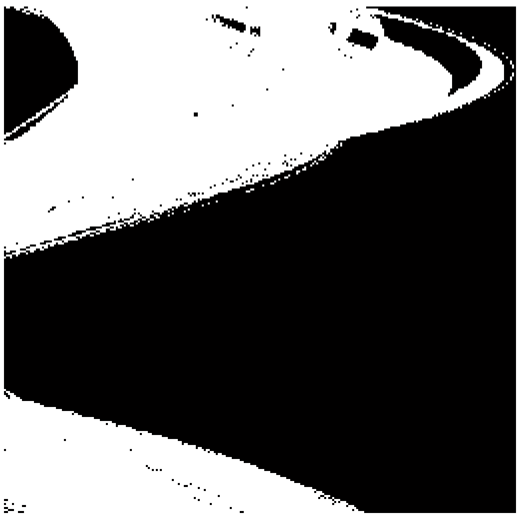
Accumulator array (θ, d)



University of California Santa Barbara



(a)
(b)
NUMBER OF EDGE ELEMENTS
120
90
60
30
0
0 50 100 150 200 250 300 350
ANGLE IN deg
(c)
NUMBER OF EDGE ELEMENTS
50
40
30
20
10
0
1 2 3 4 5 6 7 8 9 1
0 50 100 150 200 250 300 350
ANGLE IN deg
(d)
(e)
NUMBER OF EDGE ELEMENTS
80
60
40
20
0
−400 −200 0 200 400
ρ IN pixels
(f)
NUMBER OF EDGE ELEMENTS
10
8
6
4
2
0
1 2 3 4 5 6
−400 −200 0 200 400
ρ IN pixels
(g)
(h)



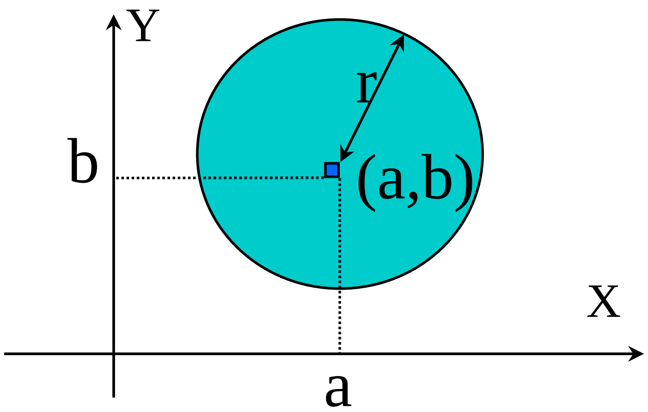
Hough Transform for Circles

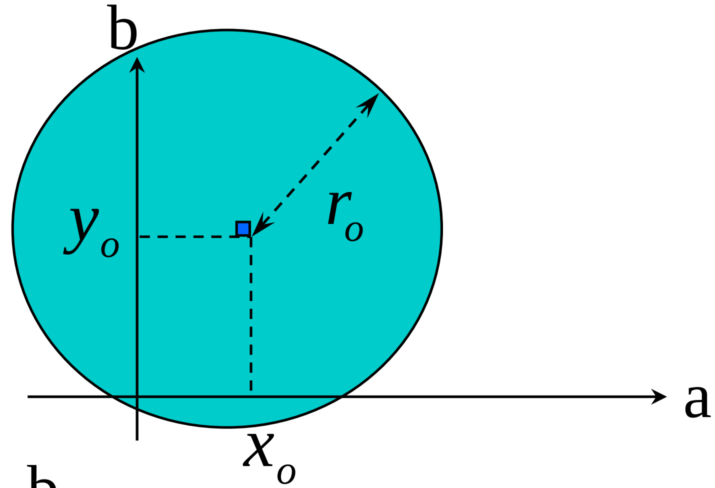
Image space

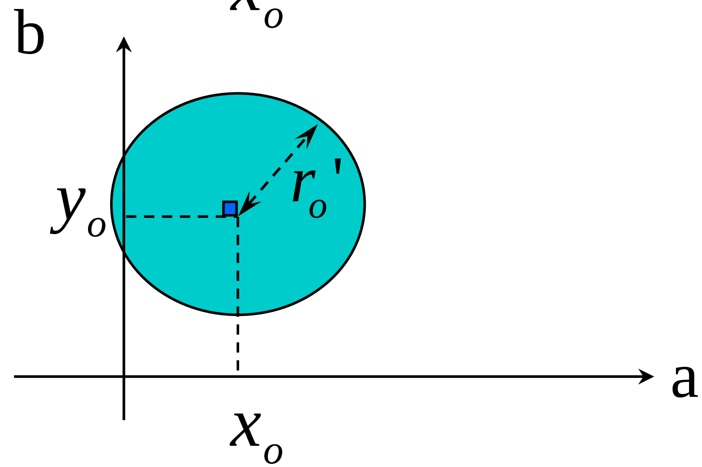
$$(x-a)^2+(y-b)^2=r^2$$

Measurement space

$$(a-x_o)^2+(b-y_o)^2=r_o^{\,2}$$

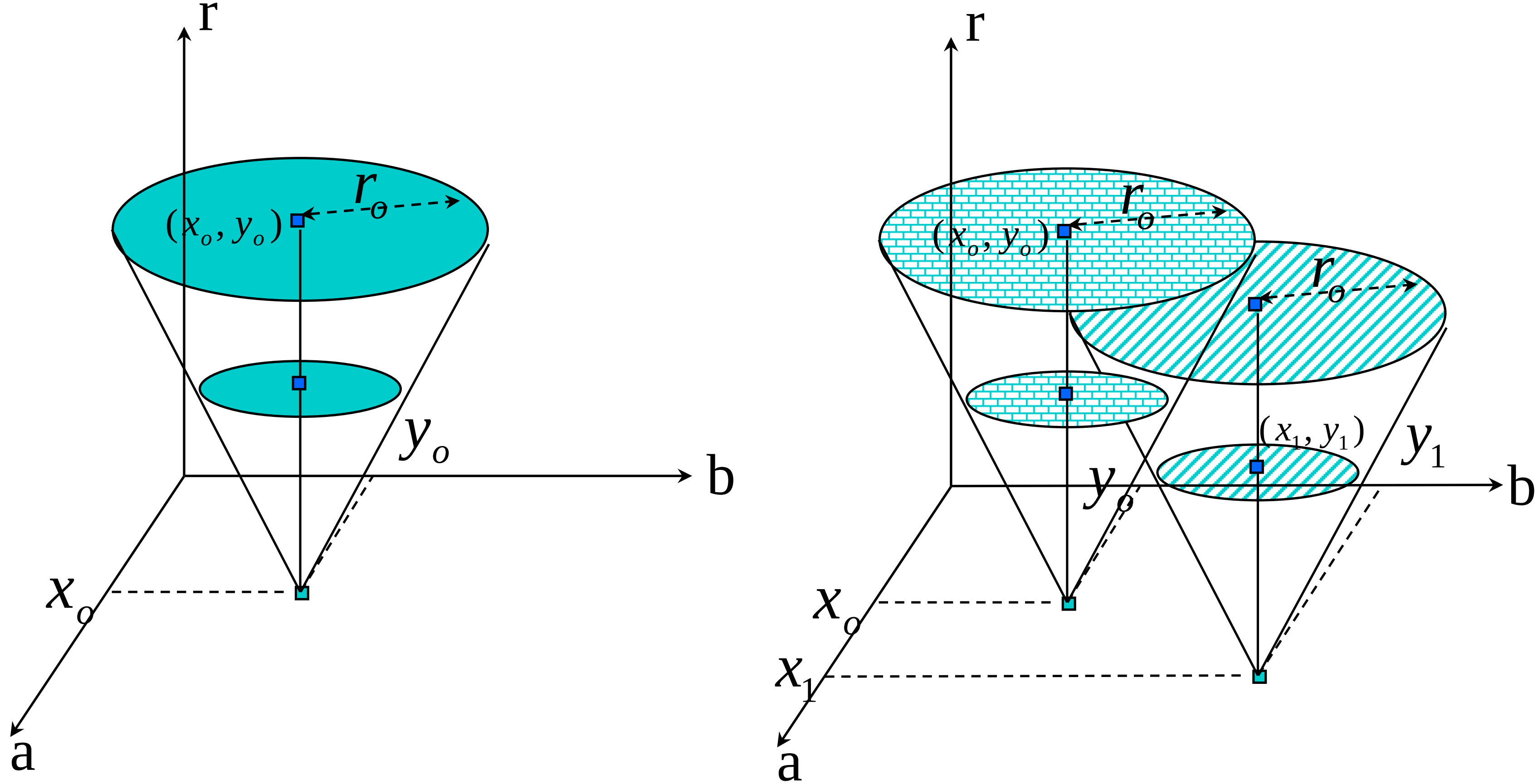






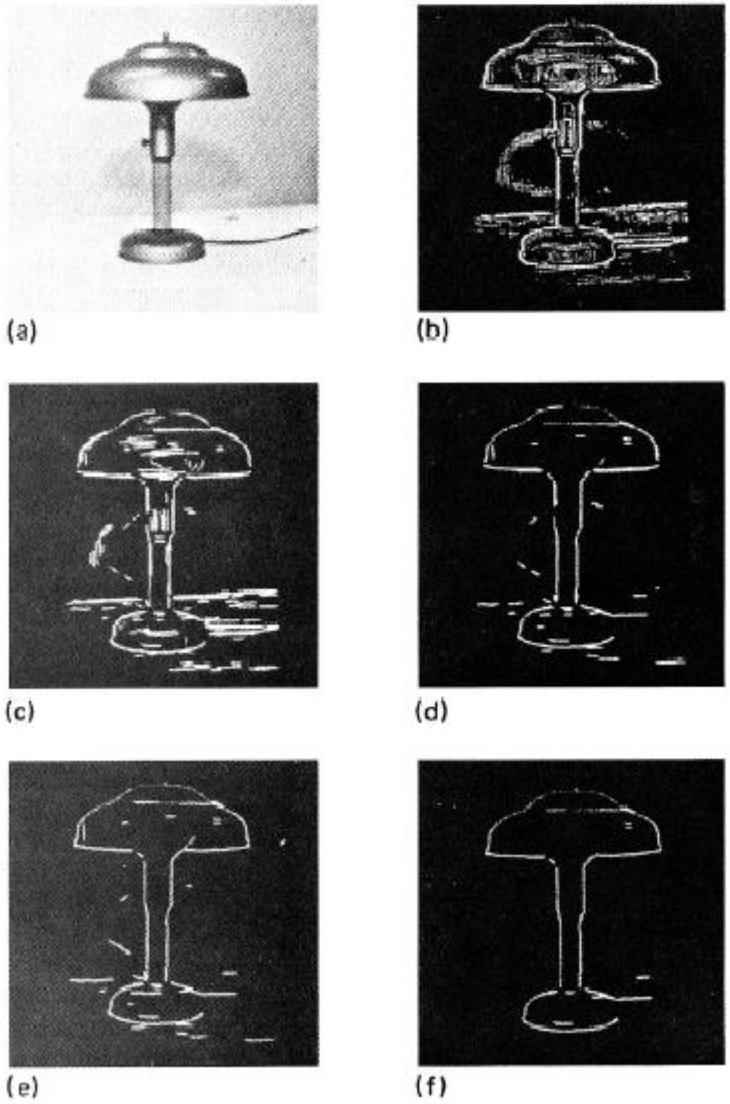
General 3D Measurement Space

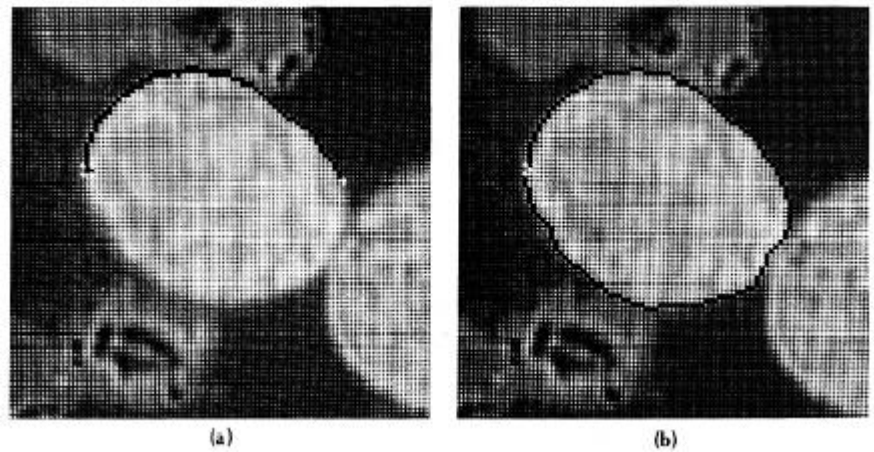
$$(a - x_o)^2 + (b - y_o)^2 = r^2$$

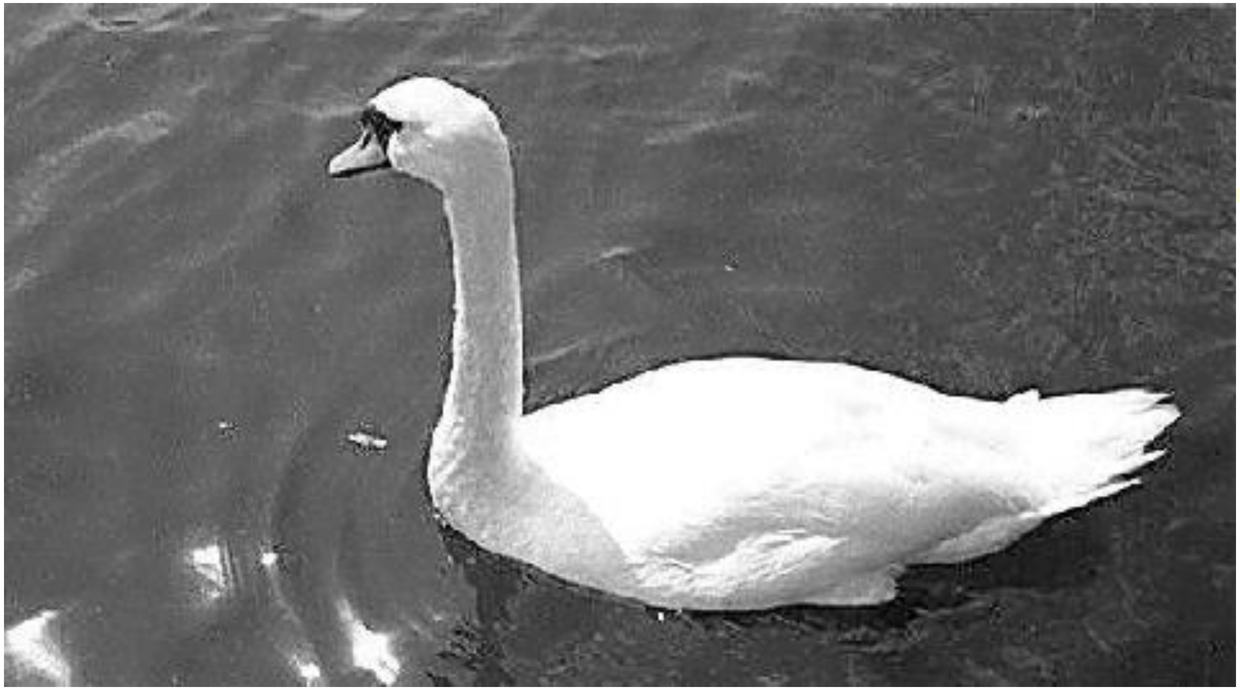


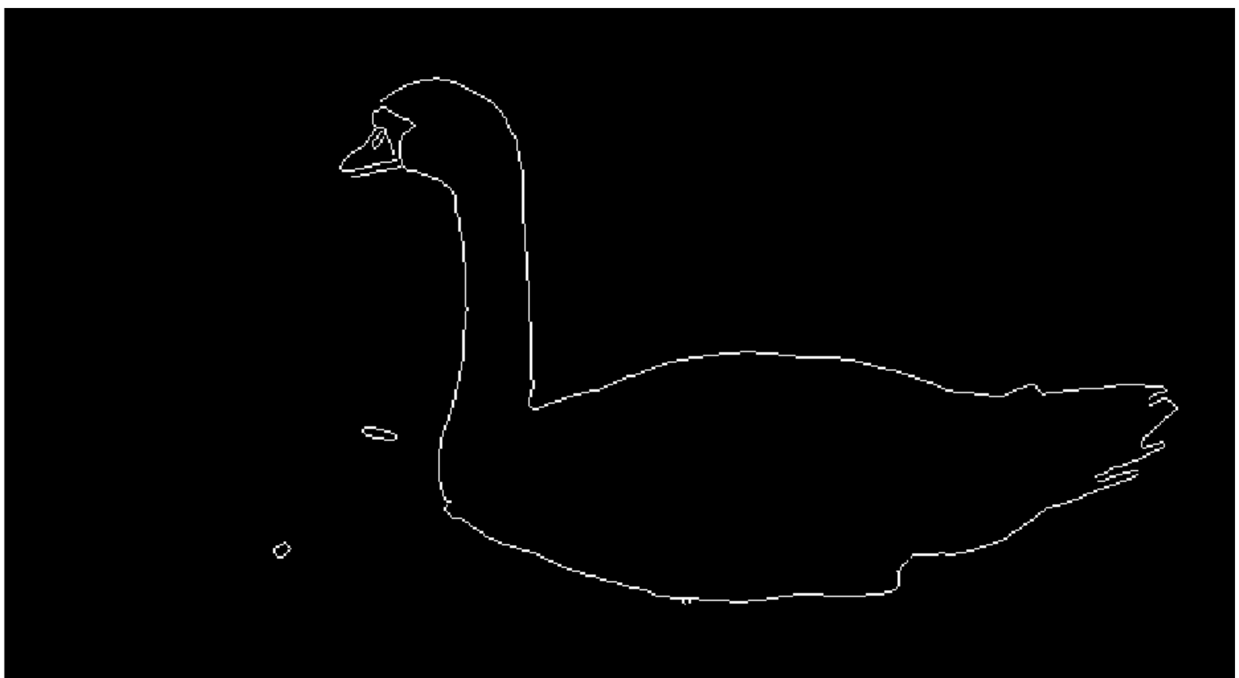
Hough Transform (cont.)

- Theoretically, Hough transform can be constructed for any parametric curve
  - a curve with n parameters
  - n-dimensional measurement space
  - (n-1)-dimensional surfaces for each image point
  - highly computationally intensive if $n>3$
  - used mainly for lines, circles, ellipses, etc.


(a)
(b)
(c)
(d)
(e)
(f)

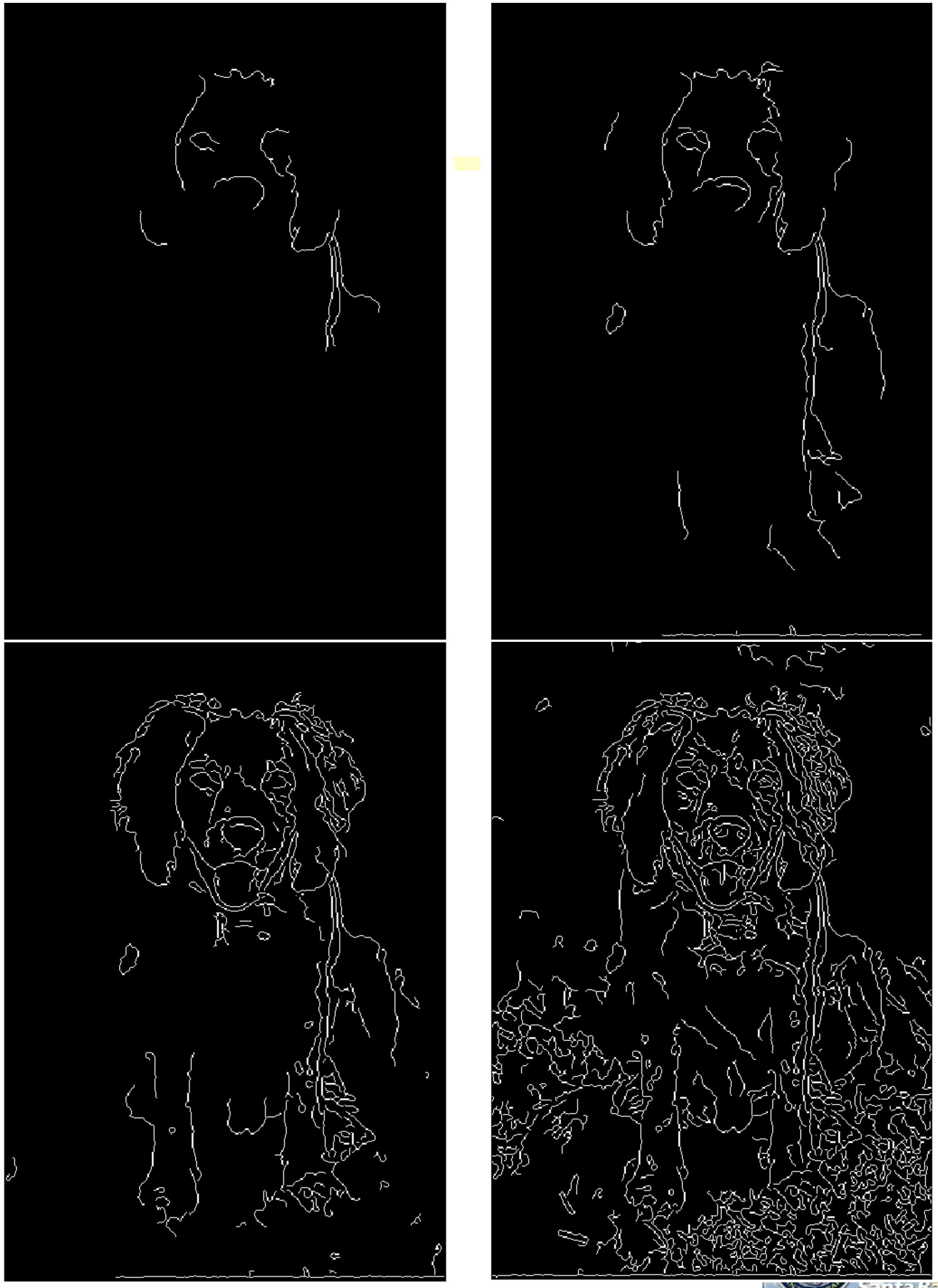

(a)
(b)





Sometimes edge detectors find the boundary pretty well

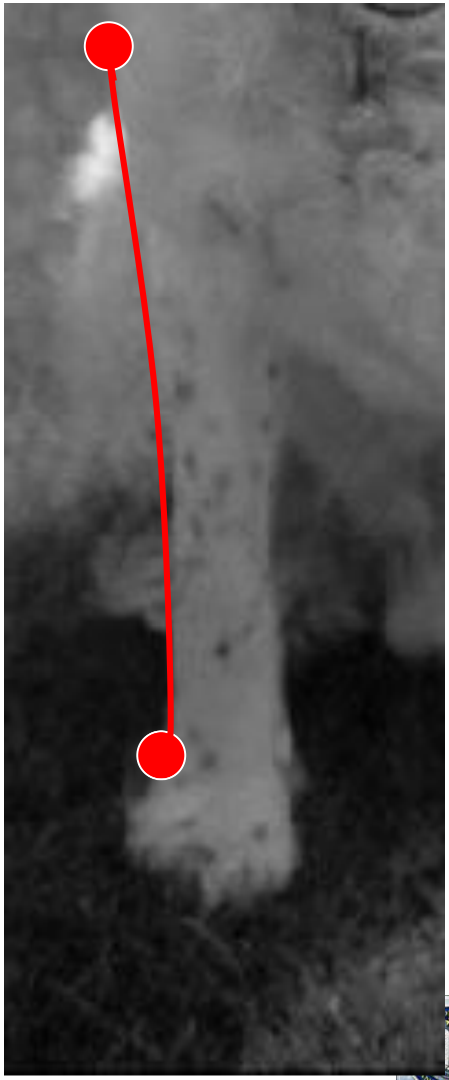
D. Jacobs



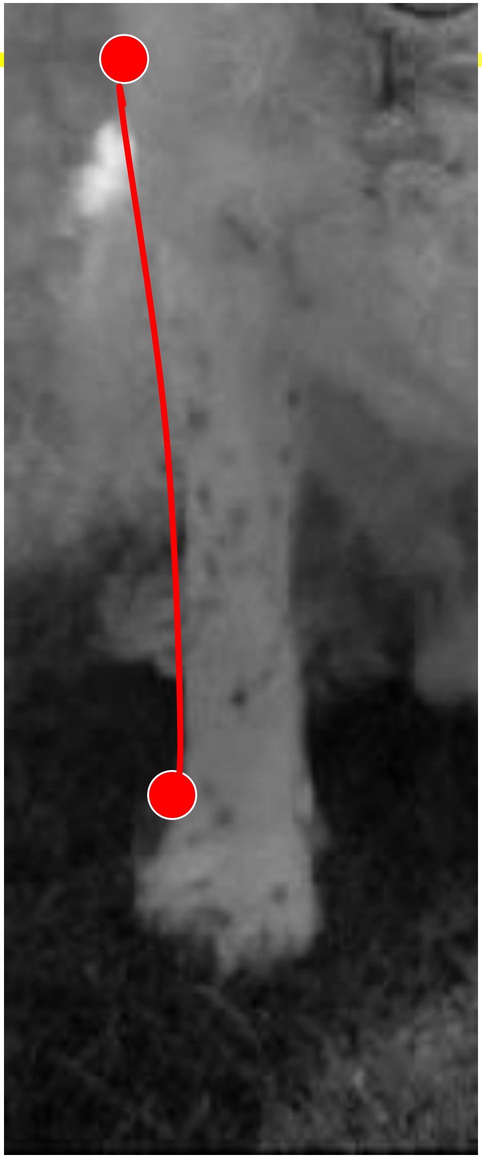
Sometimes not well at all

At times we want to find a complete bounding contour of an object:

At other times we want to find an internal or partial contour. E.g., the best path between two points:

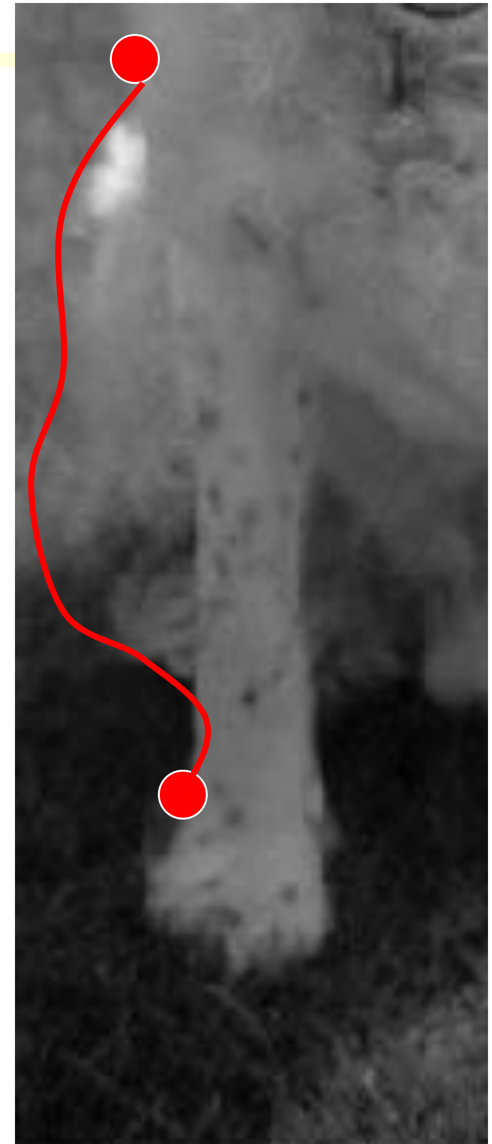


D. Jacobs



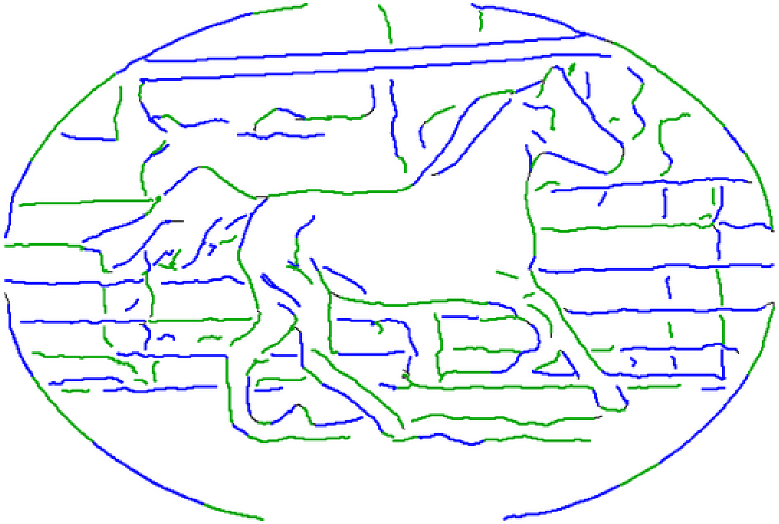
Which of these two paths is better?

How do we decide how good a path is?

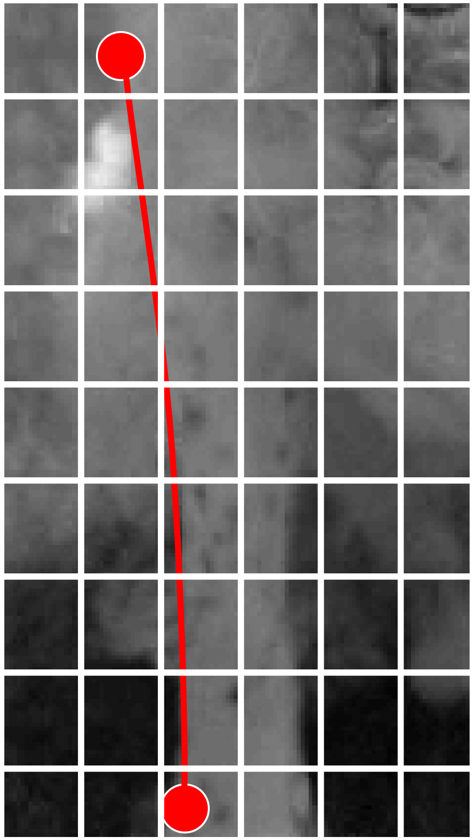


Example: edgels to line segments to contours

Original image



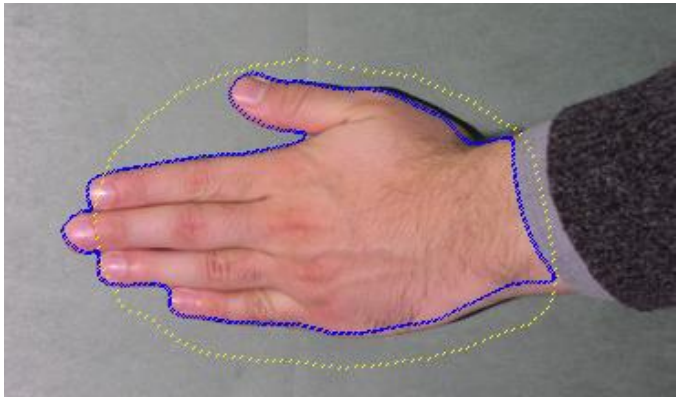
Contours derived from edgels

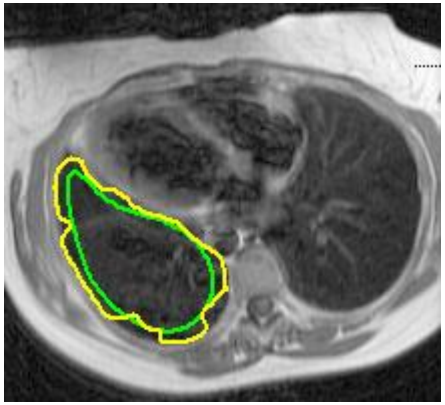


Desired properties of an image contour:

- Contour should be near/on edges
  - Strength of gradient
- Contour should be smooth (good continuation)
  - Low curvature

Active Contours (deformable contours, snakes)



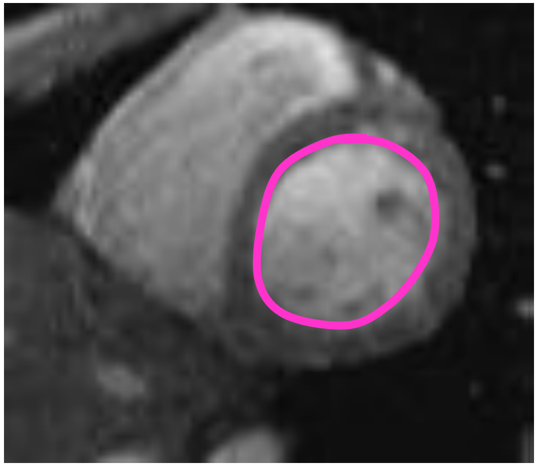


- Points, corners, lines, circles, etc., do not characterize well many objects, especially non-man-made ones
- We want other ways to describe and represent objects and image regions: Contour representations
- In particular, ***active contours*** are contour representations that conform to the (2D) shape by combining geometry and physics to make elastic, deformable shape models
  - These are often used to track contours in time, so the shape deforms to stay with the changing object

Active Contours

- Given an initial contour estimate, find the best match to the image data – evolve the contour to fit the object boundary
  - This is an optimization problem
    - Often uses dynamic programming, or something similar, in its solution
    - Iterates until final solution, or until a time limit
  - Visual evidence (support) for the contour can come from edges, corners, detected features, or even user input
- Current best contour fit can be the initial estimate for the subsequent frame (e.g., in tracking over time)
- Active contours are particularly useful when dealing with deformable (non-rigid) objects and surfaces
  - These are not easily described by edges, corners, etc.

Active Contours



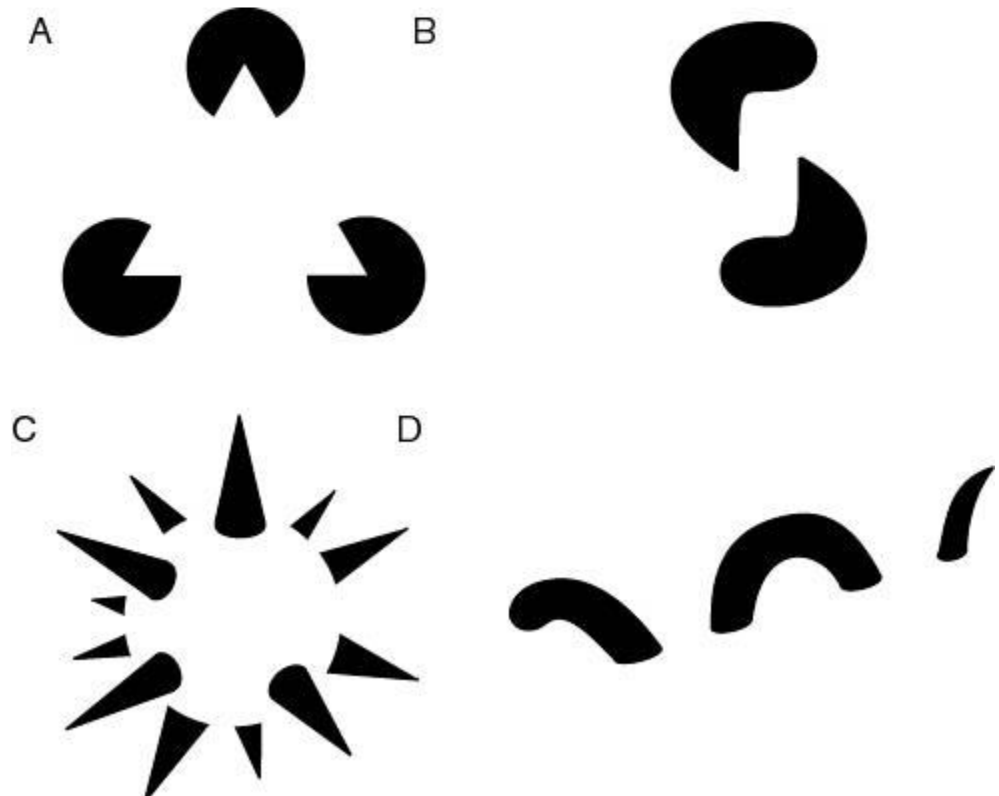
- Applications:
  - Object segmentation (for object recognition, medical imaging, etc.)
  - Tracking through time
  - Region selection (e.g., in Photoshop) – human in the loop

Contour tracking examples

- http://www.youtube.com/watch?v=laiykNbPkgg
- http://www.youtube.com/watch?v=5se69vcbqxA
- http://www.youtube.com/watch?v=ARIZzcE11Es
- http://www.youtube.com/watch?v=OFTDqGLa2p0

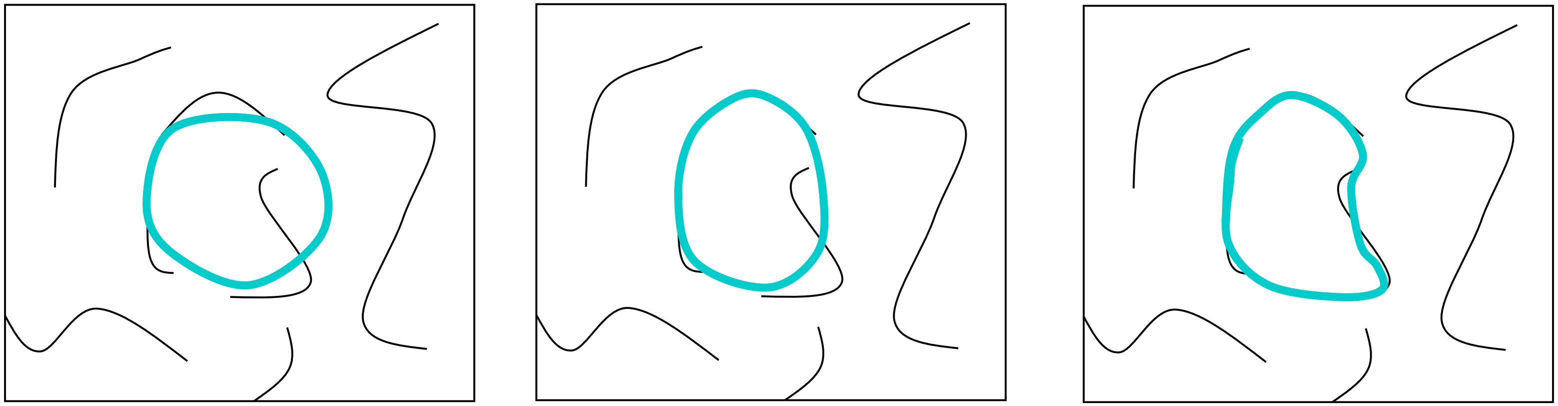
Illusory contours

Human vision seems to “fill in” where there is visual evidence of a contour



Partial contours

- Active contours can deal with occluded or missing image data



initial intermediate final

Active contours

- Think of an active contour as an elastic band, with an initial default (low energy) shape, that gets pulled or pushed to be near image positions that satisfy various criteria
  - Be near high gradients, detected points, user input, etc.
  - Don't get stretched too much
  - Keep a smooth shape
- How is the current contour adjusted to find the new contour at each iteration?
  - Define a cost function ("energy" function) that says how good a possible configuration is.
  - Seek next configuration that minimizes that cost function.

Energy minimization framework

- Framework: energy minimization
  - Bending and stretching curve = more energy
  - Good features = less energy
  - Curve evolves to minimize energy

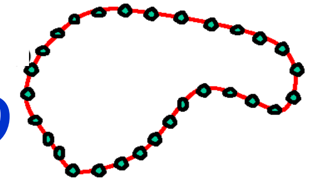
- Parametric representation of the curve

$$\boldsymbol{v}(\boldsymbol{s}) = (\boldsymbol{x}(\boldsymbol{s}), \boldsymbol{y}(\boldsymbol{s}))$$

- Minimize an energy function on $\boldsymbol{v}(\boldsymbol{s})$

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

Energy minimization framewo



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

- A good fit between the current deformable contour and the target shape in the image will yield a **low** value for this cost (energy) function
  - **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.
  - **Constraint** energy: allow for specific (often user-specified) constraints that alter the contour locally

Energy minimization

- The energy functional typically consists of three terms:

$$\varepsilon = \int \left[\varepsilon_{\text{int}}\big(\mathrm{v}(s)\big) + \varepsilon_{img}\big(\mathrm{v}(s)\big) + \varepsilon_{con}\big(\mathrm{v}(s)\big) \right] ds$$

Total energy

Internal (contour) energy

Image energy

Constraint energy

$$\varepsilon_{\text{int}}\big(\mathrm{v}(s)\big) = \left(\alpha(s) \left\| \mathrm{v}_s(s) \right\|^2 + \beta(s) \left\| \mathrm{v}_{ss}(s) \right\|^2 \right) / 2$$

Minimize length and curvature of contour

$$\varepsilon_{img} = -w \cdot \left\| \nabla I(x, y) \right\|^2$$

Maximize gradient along contour

(Minimize the negative of this)

$$\varepsilon_{con} = k \cdot \left\| \mathrm{v} - \mathrm{x} \right\|^2$$

Spring constraint (attraction)

$$\varepsilon_{con} = \frac{k}{\left\| \mathrm{v} - \mathrm{x} \right\|^2}$$

Negative spring constraint (repulsion)

Examples

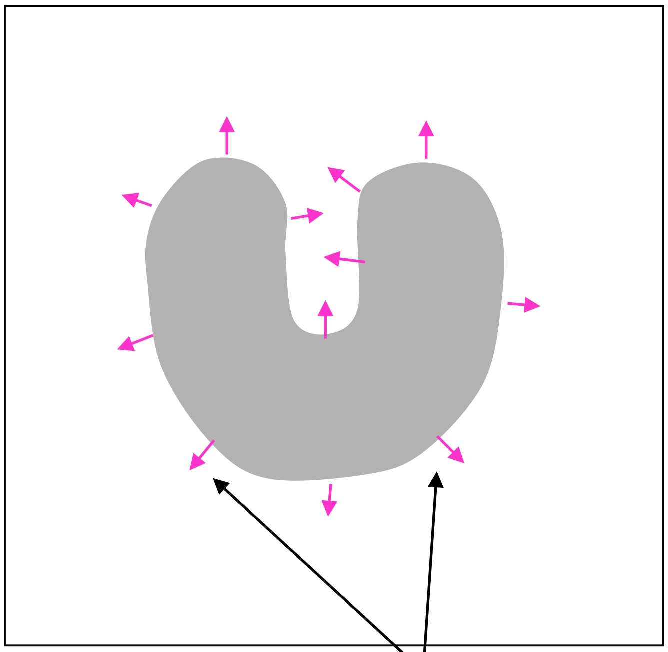
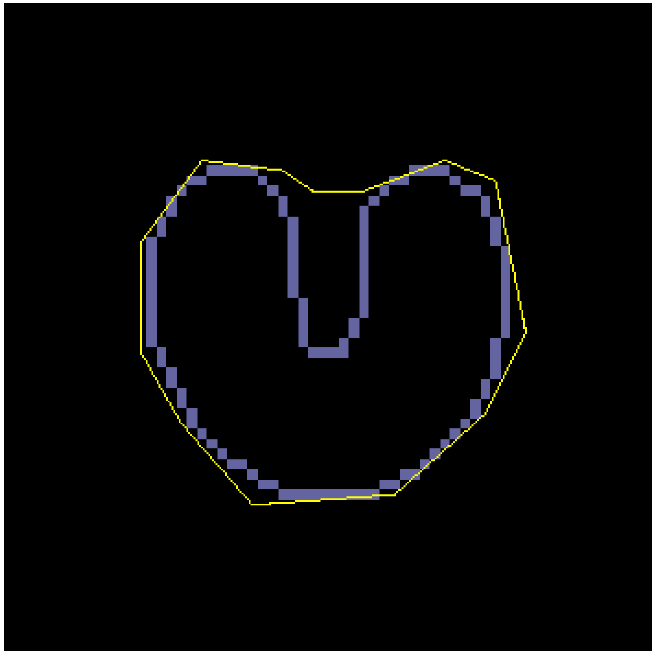


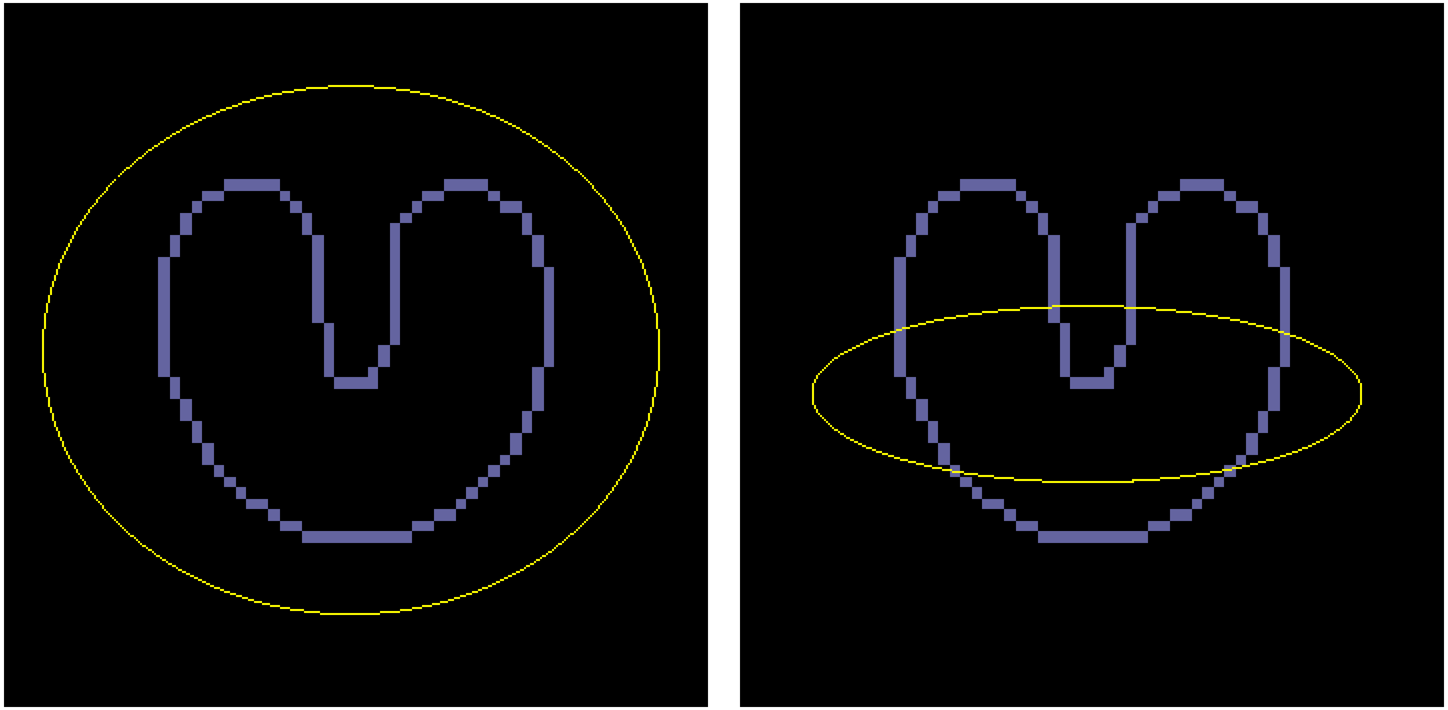
Image gradients ∇I

are large only directly on the boundary

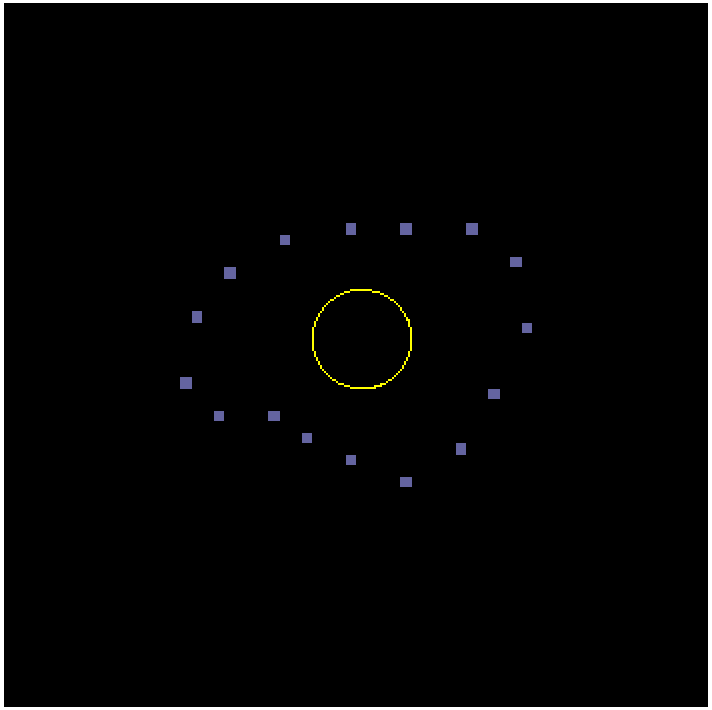


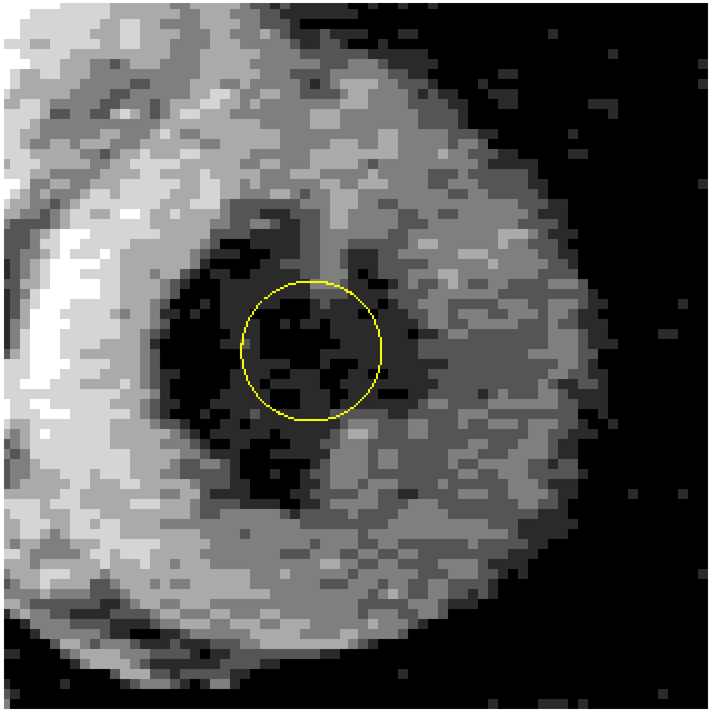
Internal model is too "tight"

Examples

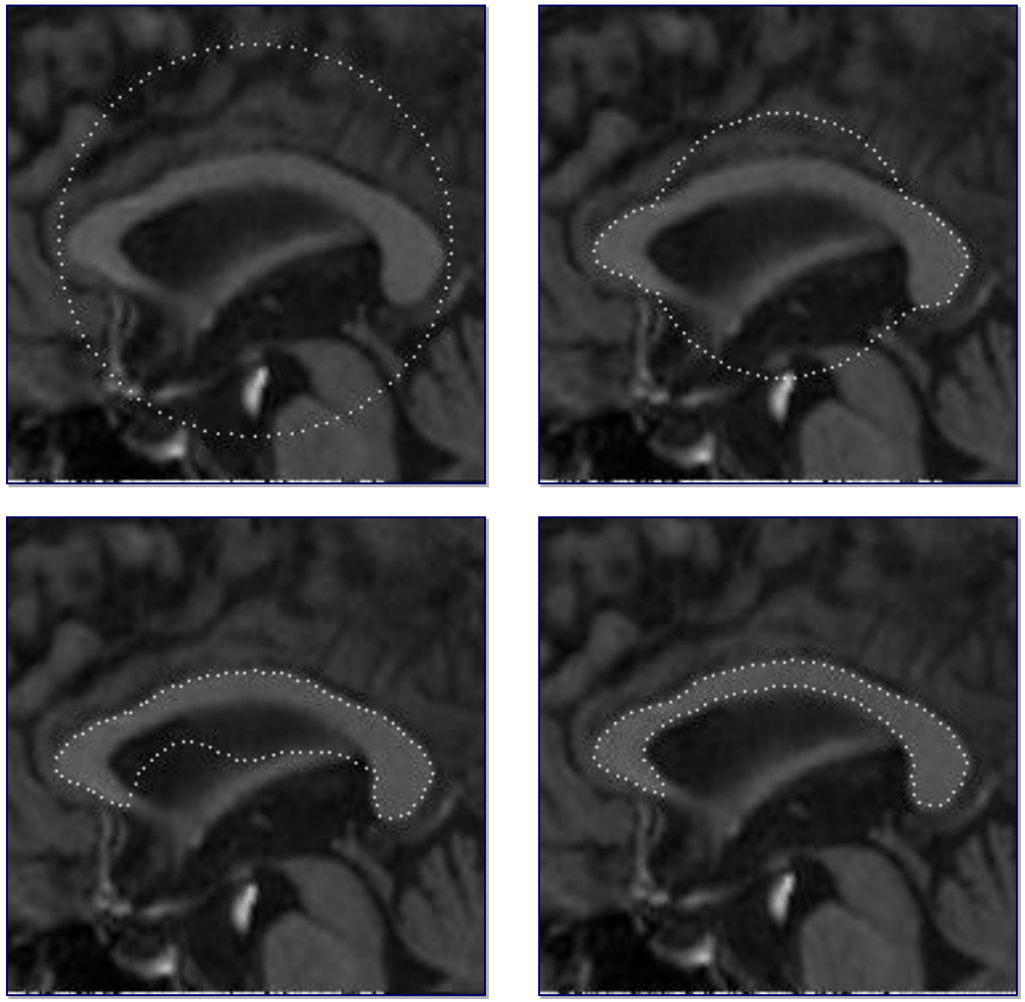


Examples

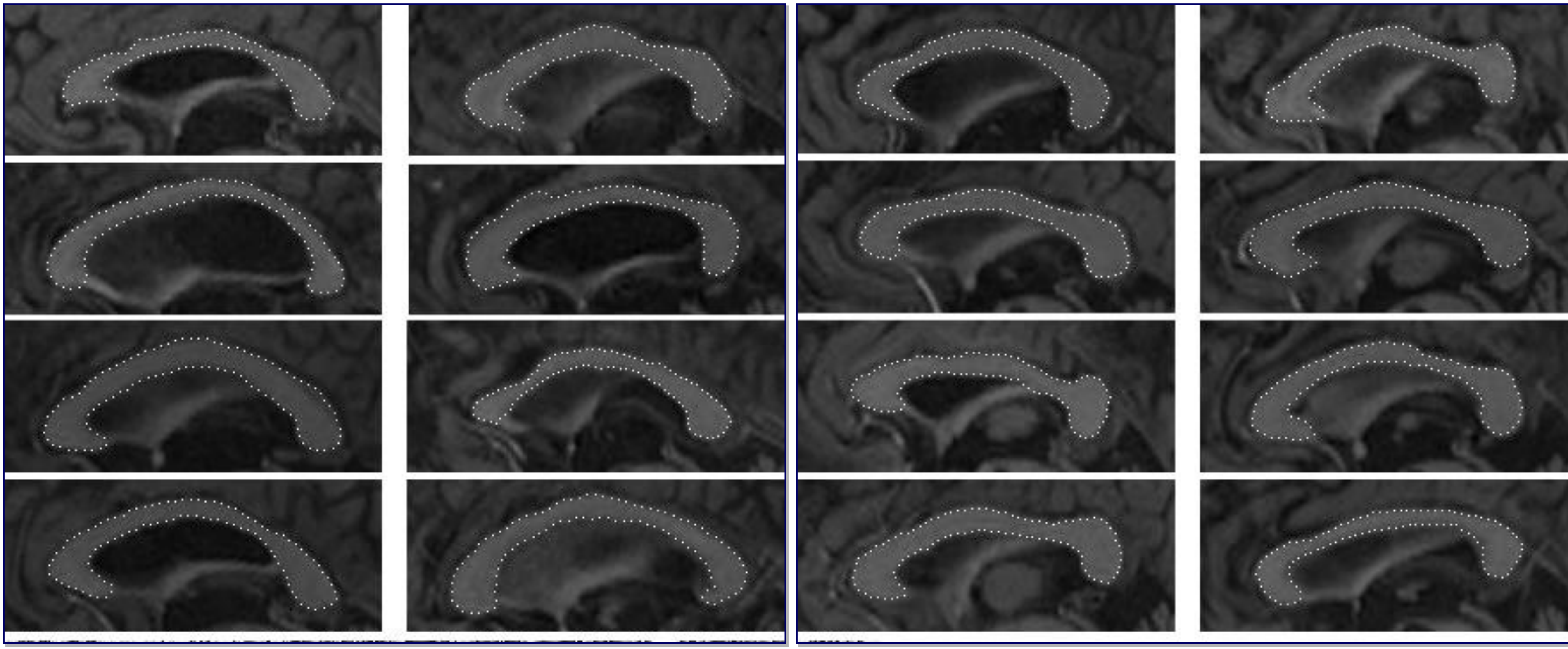




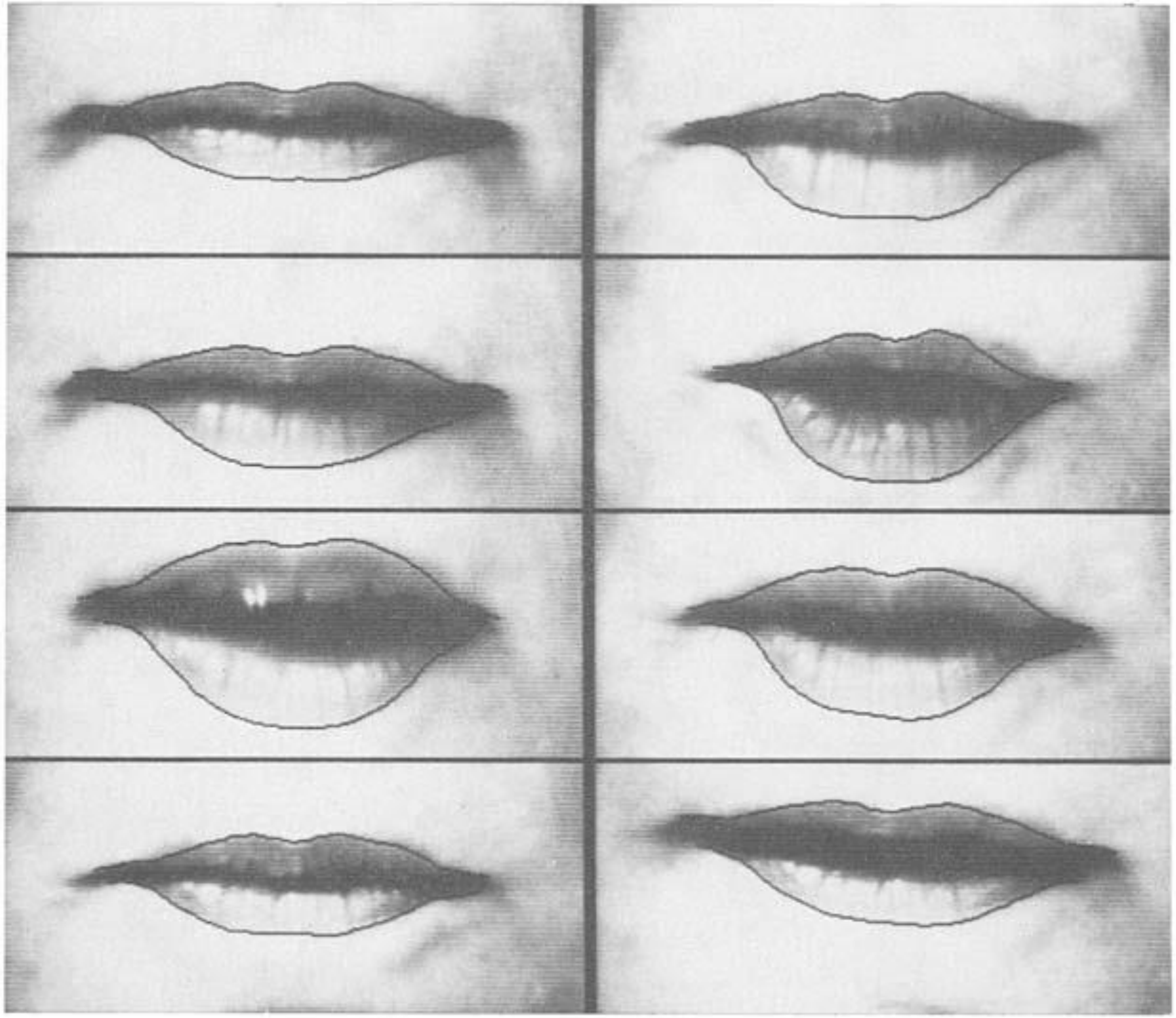
Corpus callosum example



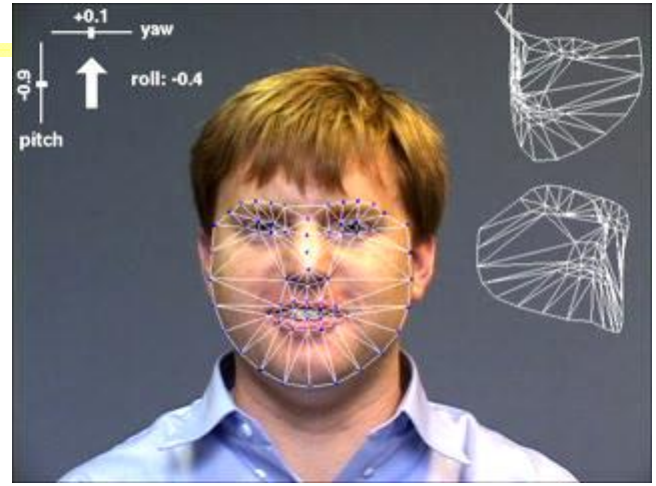
Corpus callosum example



Lips example




05:05:39:29
05:05:39:29
05:05:39:29


+0.1
yaw
roll: -0.4
-0.9
pitch

Active 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

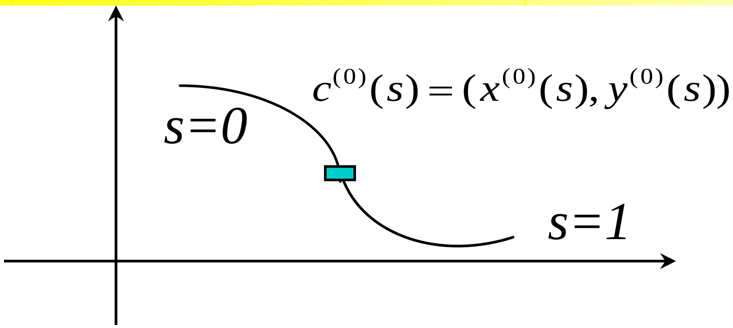
Devil in the Details

- Snake: an energy minimizing spline
- subject to
  - internal forces (*template shape*)
    - resisting stretching and compression
      - maintain natural length
    - resisting bending
      - maintain natural curvature
    - resisting twisting
      - maintain natural torsion (for 3D snake)
  - external forces (*shape detector)*
    - attract a snake to lines, edges, corners, etc.

Physics Law

- A snake's final position and shape influenced by
  - balance of all applied forces
  - total potential energy is minimum
  - a dynamic sequence is played out which is based on physics principle

A 2D snake



- Internal energy

$$E_{\text{int}} = \int \alpha(c_s(s) - c_s^{(0)}(s))^2 + \beta(c_{ss}(s) - c_{ss}^{(0)}(s))^2 \, ds$$

  – resisting stretching and compression

$$E_1 = \int (c_s(s) - c_s^{(0)}(s))^2 \, ds$$

  – resisting bending

$$E_2 = \int (c_{ss}(s) - c_{ss}^{(0)}(s))^2 \, ds$$

External energy

- point attachment

$$E = l\,|(x_o, y_o) - (x(s_o), y(s_o))|^2$$

– attach the snake to a bright line

$$E = -\int I(c(s))ds$$

– attach the snake to an edge

$$E = -\int (\nabla I(c(s)))^2 ds$$

- Treated as an minimization problem, we are looking for a function *c(s)* or *f(s,t)* that minimizes the total energy (int+ext)
- Intuitively,
  - small internal energy, less stretching, bending, twisting, closer to the natural resting state
  - small external energy, confirming to external constraints (e.g., close to attachment points, image contours, etc.)

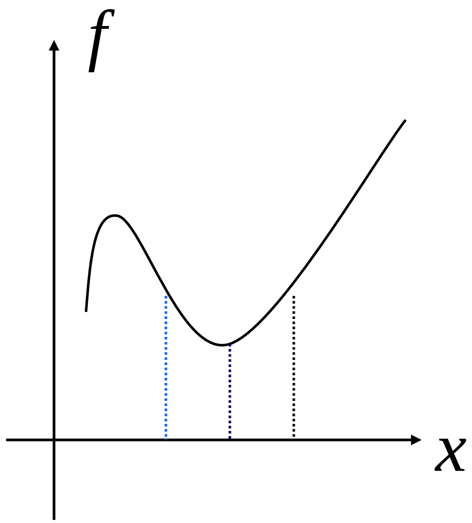
- *For those of you who are mathematics-gifted, you probably recognize this as a calculus of variation problem*
- The solution is the Euler equation (a partial differential equation)
- The energy expression is a "functional"
- Need a function to give the extremal value of the "functional"

Calculus

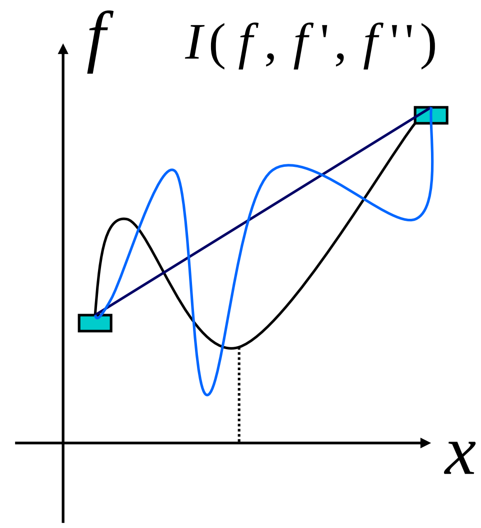
- function
- locations (extremums of function)
- derivatives
- ordinary equations

Variational Calculus

- functional
- functions (extremums of functional)
- variational derivatives
- partial differential equations



$$\frac{df}{dx} = 0$$

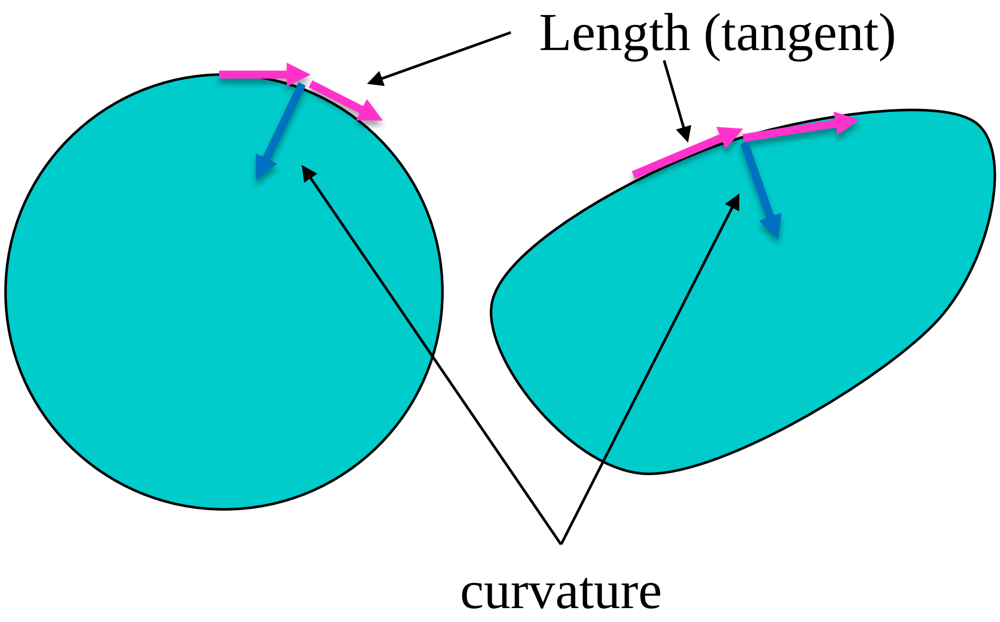


$$I_f - \frac{d}{dx} I_{f'} + \frac{d^2}{dx^2} I_{f''} = 0$$

- *For those of you who are physics-gifted, you probably recognize this as a generalized force problem*
- Again, the solution is based on the Euler equation (a partial differential equation) of variational derivatives

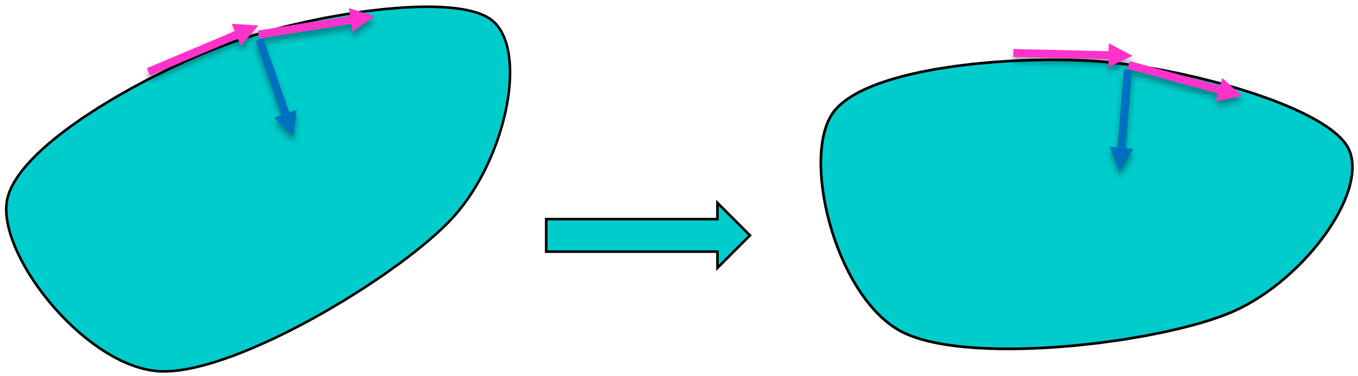
Math Detail

- Need to maintain
  - Length (no stretching)
  - Curvature (no bending)
- Both arc length and curvature are vectors!



Math Detail

- Most generally, allowing both translation and rotation (a rigid-body motion) that doesn't deform the shape
- Tangent and curvature vectors do not have to line up (under rotation), but their magnitude should be maintained
- Turn out the math becomes very messy



- Simpler formulation: translation only (or small rotation)
- Vectors should line up

Mathematical Details

Minimize

$$E_{total} = E_{\text{int}} + E_{ext}$$

$$= \int \alpha(|c_s(s)| - |c_s^{(0)}(s)|)^2 + \beta(|c_{ss}(s)| - |c_{ss}^{(0)}(s)|)^2 - \delta I(c(s)) - (\nabla I(c(s)))^2 ds$$

Simplify (translati on only)

$$E_{total} = E_{\text{int}} + E_{ext}$$

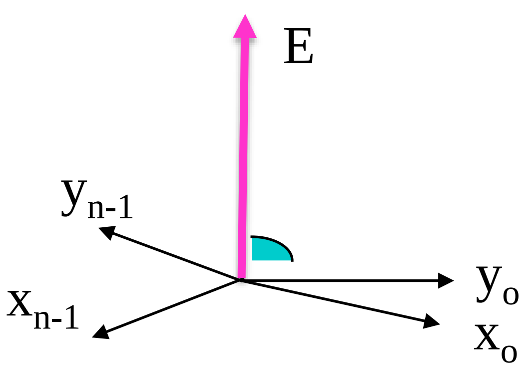
$$= \int \alpha(c_s(s) - c_s^{(0)}(s))^2 + \beta(c_{ss}(s) - c_{ss}^{(0)}(s))^2 - \delta I(c(s)) - (\nabla I(c(s)))^2 ds$$

Discretize

$$c_s(s) = c_{i+1} - c_i = (x_{i+1} - x_i, y_{i+1} - y_i)$$

$$c_{ss}(s) = c_{i+1} - 2c_i + c_{i-1} = (x_{i+1} - 2x_i + x_{i-1}, y_{i+1} - 2y_i + y_{i-1})$$

$$E(c) \Rightarrow E(x_o, y_o, \cdots, x_{n-1}, y_{n-1})$$



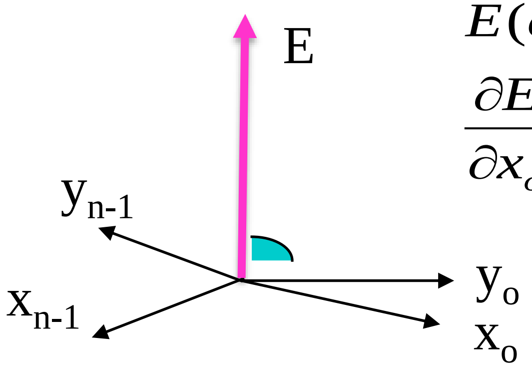
Mathematical Details

- Turn a variational calculus problem into a standard calculus problem
- 2n variables
- 2n equations (linear equations)
- Can solve a (very sparse) matrix equation of AX=B using Matlab A\B
- Sparsity comes from 1st and 2nd order derivative approximation using only neighboring points

minimize

$$E(c) \Rightarrow E(x_o, y_o, \cdots, x_{n-1}, y_{n-1})$$

$$\frac{\partial E}{\partial x_o} = \frac{\partial E}{\partial y_o} = \cdots = \frac{\partial E}{\partial x_{n-1}} = \frac{\partial E}{\partial y_{n-1}} = 0$$



Mathematical Details

Minimize

$$E_{total} = E_{\text{int}} + E_{ext}$$

$$= \int \alpha (c_s(s) - c_s{}^{(0)}(s))^2 + \beta (c_{ss}(s) - c_{ss}{}^{(0)}(s))^2 - \delta I(c(s)) - (\nabla I(c(s)))^2 \, ds$$

Discretize

$$c_s(s) = c_{i+1} - c_i = (x_{i+1} - x_i, y_{i+1} - y_i)$$

$$c_{ss}(s) = c_{i+1} - 2c_i + c_{i-1} = (x_{i+1} - 2x_i + x_{i-1}, y_{i+1} - 2y_i + y_{i-1})$$

For a particular $\mathrm{c_i}$:

$$(c_s(s) - c_s{}^{(0)}(s))^2 = ([x_{i+1} - x_i, y_{i+1} - y_i] - [x^{(0)}{}_{i+1} - x^{(0)}{}_i, y^{(0)}{}_{i+1} - y^{(0)}{}_i])^2$$

$$= [(x_{i+1} - x_i) - (x^{(0)}{}_{i+1} - x^{(0)}{}_i), (y_{i+1} - y_i) - (y^{(0)}{}_{i+1} - y^{(0)}{}_i)]^2$$

$$= ((x_{i+1} - x_i) - (x^{(0)}{}_{i+1} - x^{(0)}{}_i))^2 + ((y_{i+1} - y_i) - (y^{(0)}{}_{i+1} - y^{(0)}{}_i))^2$$

$$\frac{\partial (c_s(s) - c_s{}^{(0)}(s))^2}{\partial x_k} = 2[-((x_{k+1} - x_k) - (x^{(0)}{}_{k+1} - x^{(0)}{}_k)) + (x_k - x_{k-1}) - (x^{(0)}{}_k - x^{(0)}{}_{k-1})] + \cdots$$

1st derivatives of $\mathrm{x_{k+1}}$ and $\mathrm{x_k}$ involve $\mathrm{x_k}$

Mathematical Details

Minimize

$$E_{total} = E_{\text{int}} + E_{ext}$$

$$= \int \alpha(c_s(s) - c_s^{(0)}(s))^2 + \beta(c_{ss}(s) - c_{ss}^{(0)}(s))^2 - I(c(s)) - (\nabla I(c(s)))^2 \, ds$$

Discretize

$$c_s(s) = c_{i+1} - c_i = (x_{i+1} - x_i, y_{i+1} - y_i)$$

$$c_{ss}(s) = c_{i+1} - 2c_i + c_{i-1} = (x_{i+1} - 2x_i + x_{i-1}, y_{i+1} - 2y_i + y_{i-1})$$

For a particular $\mathrm{x_i}$:

$$(c_{ss}(s) - c_{ss}^{(0)}(s))^2 = ([x_{i+1} - 2x_i + x_{i-1}, y_{i+1} - 2y_i + y_{i-1}] - [x^{(0)}{}_{i+1} - 2x^{(0)}{}_i + x^{(0)}{}_{i-1}, y^{(0)}{}_{i+1} - 2y^{(0)}{}_i + y^{(0)}{}_{i-1}])^2$$

$$= [(x_{i+1} - 2x_i + x_{i-1}) - (x^{(0)}{}_{i+1} - 2x^{(0)}{}_i + x^{(0)}{}_{i-1}), (y_{i+1} - 2y_i + y_{i-1}) - (y^{(0)}{}_{i+1} - 2y^{(0)}{}_i + y^{(0)}{}_{i-1})]^2$$

$$= ((x_{i+1} - 2x_i + x_{i-1}) - (x^{(0)}{}_{i+1} - 2x^{(0)}{}_i + x^{(0)}{}_{i-1}))^2 + ((y_{i+1} - 2y_i + y_{i-1}) - (y^{(0)}{}_{i+1} - 2y^{(0)}{}_i + y^{(0)}{}_{i-1}))^2$$

$$\frac{\partial (c_{ss}(s) - c_{ss}^{(0)}(s))^2}{\partial x_k} = 2[-2((x_{k+1} - 2x_k + x_{k-1}) - (x^{(0)}{}_{k+1} - 2x^{(0)}{}_k + x^{(0)}{}_{k-1}))] + \ldots$$

2^{nd} derivatives of x_{k+1} and x_{k-1} also involve x_k

Mathematical Details

Minimize

$$E_{total} = E_{\text{int}} + E_{ext}$$

$$= \int \alpha(c_s(s) - c_s{}^{(0)}(s))^2 + \beta(c_{ss}(s) - c_{ss}{}^{(0)}(s))^2 - \delta I(c(s)) - (\nabla I(c(s)))^2 \, ds$$

Discretize

$$c_s(s) = c_{i+1} - c_i = (x_{i+1} - x_i, y_{i+1} - y_i)$$

$$c_{ss}(s) = c_{i+1} - 2c_i + c_{i-1} = (x_{i+1} - 2x_i + x_{i-1}, y_{i+1} - 2y_i + y_{i-1})$$

For a particular $\mathrm{c_i}$:

$$(\nabla I(c(s)))^2 = [I_x(x_i, y_i), I_y(x_i, y_i)]^2 = (I_x(x_i, y_i))^2 + (I_y(x_i, y_i))^2$$

$$\frac{\partial(\nabla I(c(s)))^2}{\partial x_k} = 2[I_x(x_k, y_k)\frac{\partial I_x}{\partial x_k} + I_y(x_k, y_k)\frac{\partial I_y}{\partial x_k}] = 2[I_x(x_k, y_k), I_y(x_k, y_k)][\frac{\partial I_x}{\partial x_k}, \frac{\partial I_y}{\partial x_k}]$$

Mathematical Details

Minimize

$$E_{total} = E_{\text{int}} + E_{ext}$$

$$= \int \alpha(c_s(s) - c_s{}^{(0)}(s))^2 + \beta(c_{ss}(s) - c_{ss}{}^{(0)}(s))^2 - \delta I(c(s)) - (\nabla I(c(s)))^2 \, ds$$

$$\frac{\partial(\nabla I(c(s)))^2}{\partial x_k} = 2[I_x(x_k, y_k)\frac{\partial I_x}{\partial x_k} + I_y(x_k, y_k)\frac{\partial I_y}{\partial x_k}] = 2[I_x(x_k, y_k), I_y(x_k, y_k)][\frac{\partial I_x}{\partial x_k}, \frac{\partial I_y}{\partial x_k}]$$

- Derivative of E (potential) is a gradient (force) field
- Minimization go in the negative gradient direction
- Pull the snake in the direction
  - Large gradient
  - Large increase in gradient
  - around a node

❖ The equation represents balance of forces!

- A force to enforce similar tangent

$$(x_{k+1} - x_k) - (x^{(0)}{}_{k+1} - x^{(0)}{}_k)$$

- A force to enforce similar curvature

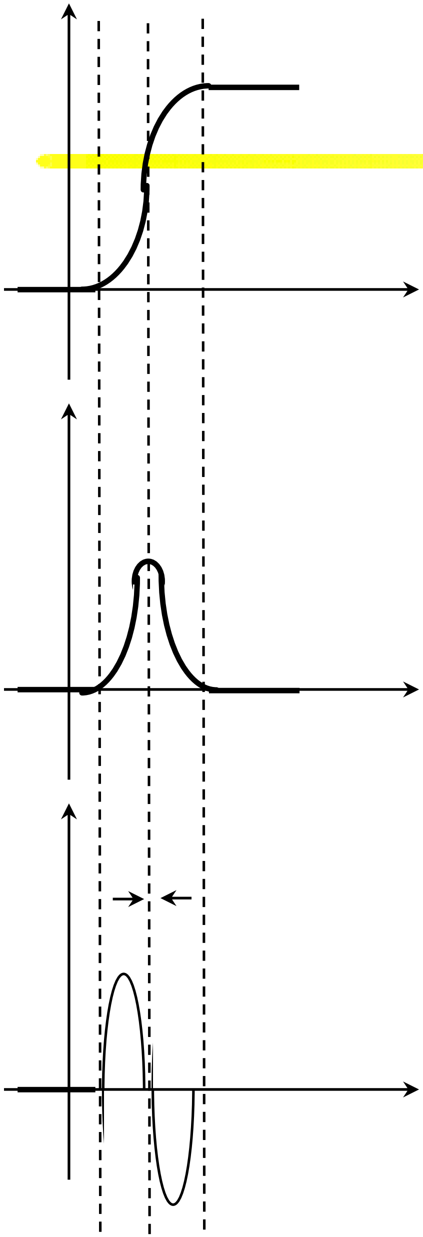
$$(x_{k+2} - 2x_{k+1} + x_k) - (x^{(0)}{}_{k+2} - 2x^{(0)}{}_{k+1} + x^{(0)}{}_k)$$

- A force to penalize non-maximum intensity

$$\frac{\partial I}{\partial x_k}$$

- A force to penalize not at zero crossing

$$\frac{\partial^2 I}{\partial x_k{}^2}$$



- Caveat:
  - Snake needs good initial position
  - Provided by initial interactive placement
  - Smooth images to enlarge “potential field”
  - Snake won’t move if
    - Gradient is zero or
    - Change of gradient is zero

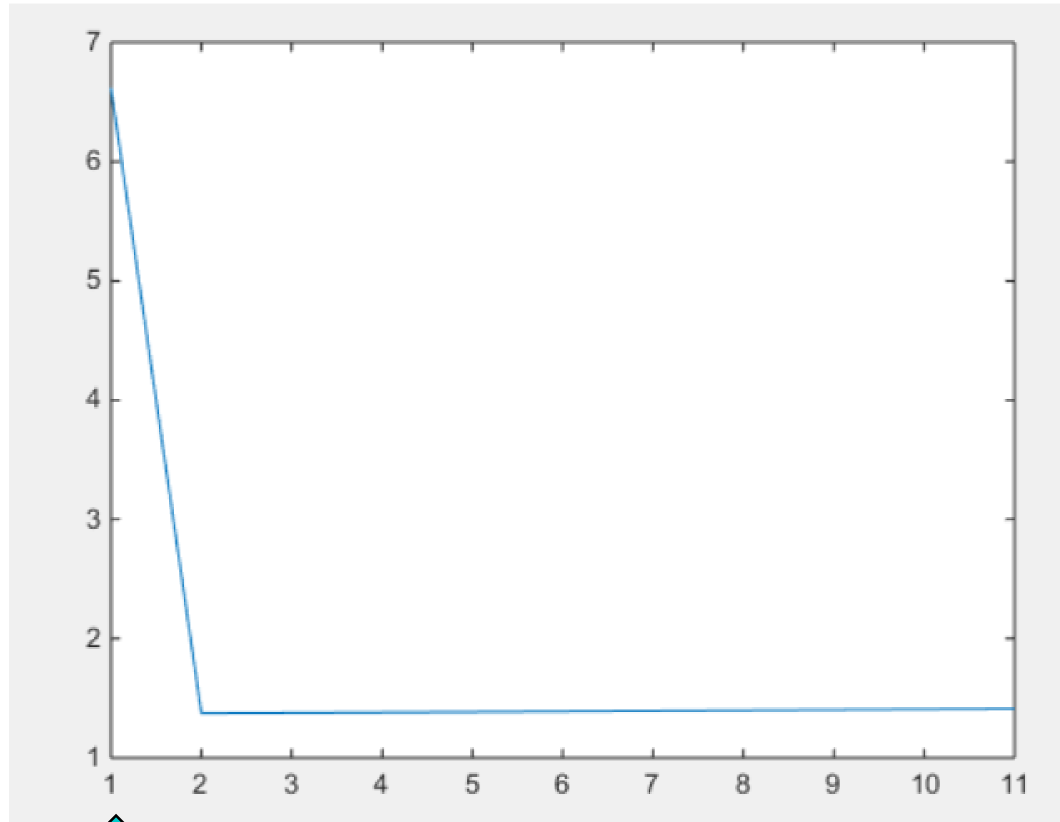
Numerical Methods

- Should result in a sparse, pentadiagonal matrix
- **AX = B**, solve with
  - Direct method **X** = inv(**A**) * **B** (preferred for small system < 20 points)
  - Iterative method (Explicit Euler, see paper)
- Caveats:
  - **A** can be numerically ill-conditioned (not diagonally dominant – the |diagonal element| is larger than the sum of |off-diagonal elements|)
- Fix: Regularization (a topic to be discussed more later)
- Minimize || AX-B||^2 + w||X||^2
- (A+ wI) X = B or X = (A+wI)*B

Numerical Methods

```
a = [
    8   -5    1    0    0    1   -5
   -5    8   -5    1    0    0    1
    1   -5    8   -5    1    0    0
    0    1   -5    8   -5    1    0
    0    0    1   -5    8   -5    1
    1    0    0    1   -5    8   -5
   -5    1    0    0    1   -5    8
];
b = rand(7,1) ;

for lambda = 0:1:10
    x = inv(a+lambda*eye(7))*b;
    err(lambda+1) = norm(a*x - b);
end
plot(err)
```



Lambda=0