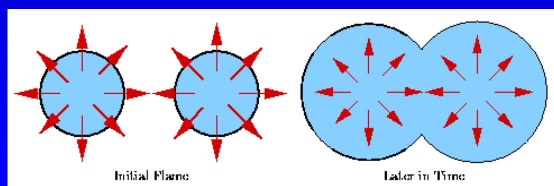
Level Set Methods

- *Contour evolution method due to J. Sethian and S. Osher, 1988*
- *www.math.berkeley.edu/~sethian/level_set.html*
- *Difficulties with snake-type methods*
  - **Hard to keep track of contour if it self-intersects during its evolution**
  - **Hard to deal with changes in topology**

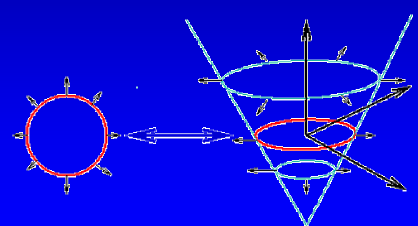


- *The level set approach:*
  - **Define problem in 1 higher dimension**
  - **Define level set function** $z = \phi(x,y,t = 0)$ **where the (x,y) plane contains the contour, and z = signed Euclidean distance transform value (negative means inside closed contour, positive means outside contour)**

How to Move the Contour?

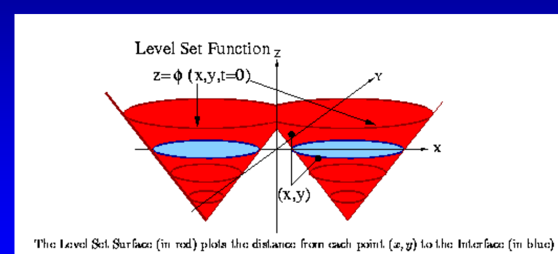
- *Move the level set function, $\phi(x,y,t)$, so that it rises, falls, expands, etc.*
- *Contour = cross section at $z = 0$, i.e.,*

  $\{(x,y) \mid \phi(x,y,t) = 0\}$



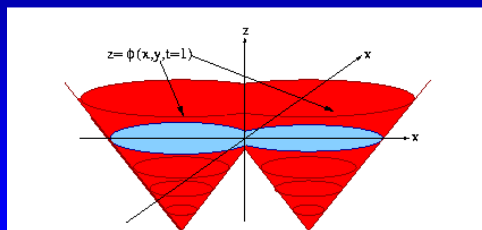
Level Set Surface

- *The zero level set (in blue) at one point in time as a slice of the level set surface (in red)*



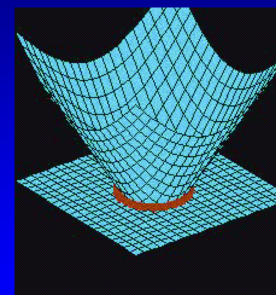
Level Set Surface

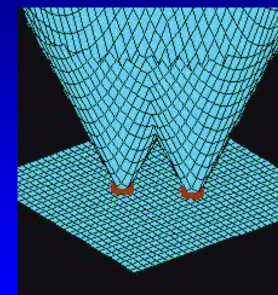
- *Later in time the level set surface (red) has moved and the new zero level set (blue) defines the new contour*



Later in Time: Red Level Set Surface has moved, yielding new Blue Interface

Level Set Surface





How to Move the Level Set Surface?

1. *Define a velocity field, F, that specifies how contour points move in time*
   - **Based on application-specific physics such as time, position, normal, curvature, image gradient magnitude**
2. *Build an initial value for the level set function, $\phi(x,y,t=0)$, based on the initial contour position*
3. *Adjust ϕ over time; contour at time t defined by $\phi(x(t), y(t), t) = 0$*

$$\frac{\partial \Phi}{\partial t} + \vec{F} \cdot \nabla \Phi = 0$$

Hamilton-Jacobi equation

$$\frac{\partial \Phi}{\partial t} + F\left(\left(\frac{\partial \Phi}{\partial x}\right)^2 + \left(\frac{\partial \Phi}{\partial y}\right)^2\right)^{1/2} = 0$$

Level Set Formulation

- *Constraint: level set value of a point on the contour with motion $x(t)$ must always be 0*

  $\phi(x(t), t) = 0$

- *By the chain rule*

  $\phi_t + \nabla \phi(x(t), t) \cdot x'(t) = 0$

- *Since F supplies the speed in the outward normal direction*

  $x'(t) \cdot n = F$, where $n = \nabla \phi / |\nabla \phi|$

- *Hence evolution equation for ϕ is*

  $\phi_t + F|\nabla \phi| = 0$

Speed Function

$$F(k) = F_0 + F_1(k) = (1 - \varepsilon k)$$

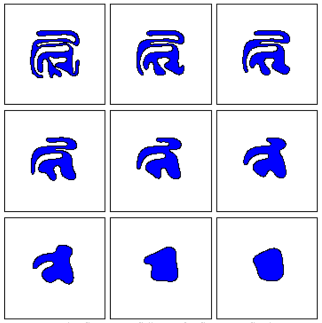
$$F(k) = k_I(x,y) * (1 - \varepsilon k)$$

$$\mathbf{k_I = \frac{1}{1 + |\nabla G_\sigma * I(x,y)|}}$$

$$\mathbf{k_I = e^{-|\nabla G_\sigma * I(x,y)|}}$$

Example: Shape Simplification

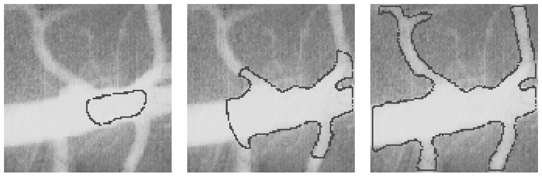
- *F = 1 – 0.1κ where κ is the curvature at each contour point*



Motion under Curvature: Collapse of a Curve to a Single Point

Example: Segmentation

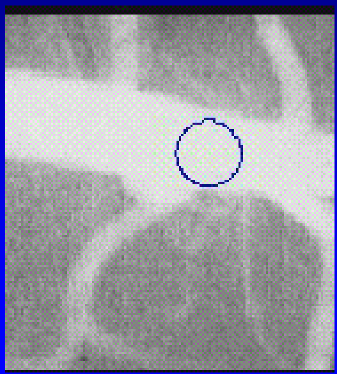
- *Digital Subtraction Angiogram*
- *F based on image gradient and contour curvature*



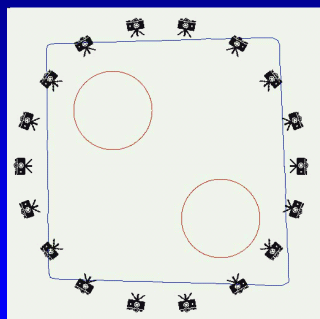
Evolving Front Driven by Function of Image Gradient.

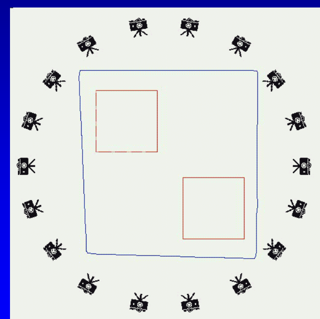
Example (cont.)

- *Initial contour specified manually*

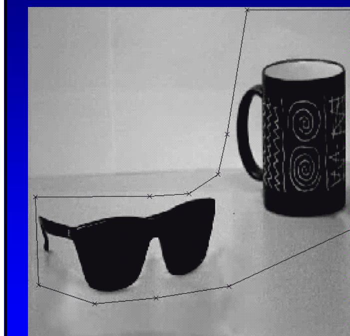


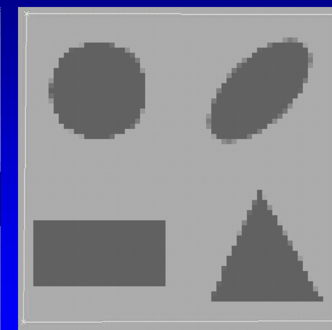
More Examples



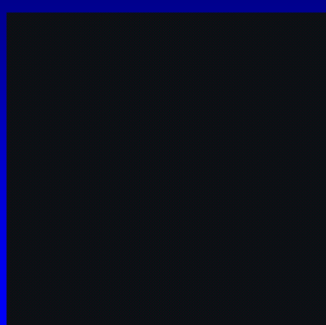


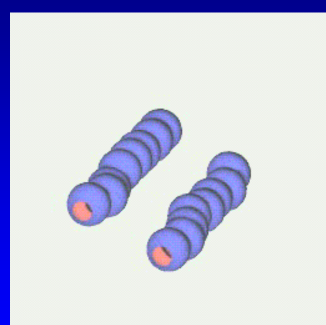
More Examples





More Examples



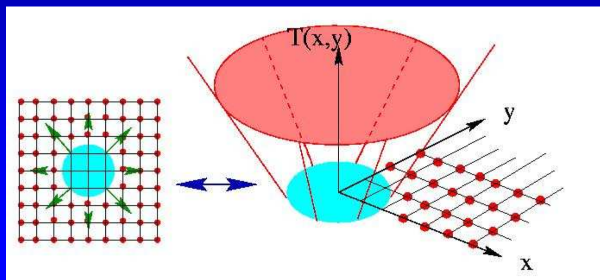


Fast Marching Method

- *J. Sethian, 1996*
- *Special case that assumes the velocity field, F, never changes sign. That is, contour is either always expanding (F>0) or always shrinking (F<0)*
- *Convert problem to a stationary formulation on a discrete grid where the contour is guaranteed to cross each grid point at most once*

Fast Marching Method

- *Compute $T(x,y)$ = time at which the contour crosses grid point (x,y)*
- *At any height, t, the surface gives the set of points reached at time t*



Fast Marching Algorithm

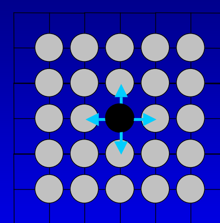
- *Compute T using the fact that*
  - *Distance = rate × time*
  - **In 1D:** $1 = F \times dT/dx$
  - **In 2D:** $1 = F \times |\nabla T|$
- *Contour at time t =*

$$\{(x,y) \mid T(x,y) = t\}$$

Fast Marching Algorithm

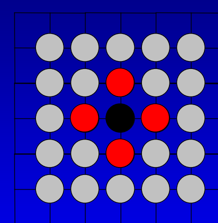
- *Construct the arrival time surface $T(x,y)$ incrementally:*
  1. Build the initial contour
  2. Incrementally add on to the existing surface the part that corresponds to the contour moving with speed F (in other words, repeatedly pick a point on the fringe with minimum T value)
  3. Iterate until F goes to 0
- *Builds level set surface by "scaffolding" the surface patches farther and farther away from the initial contour*

Fast Marching



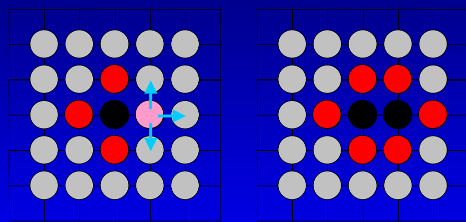
Update "downwind"

(i.e., unvisited neighbors)



Compute new possible values

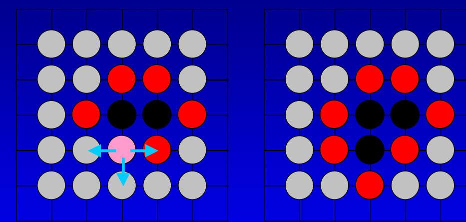
Fast Marching



Expand point on the fringe with minimum value

Update neighbors "downwind"

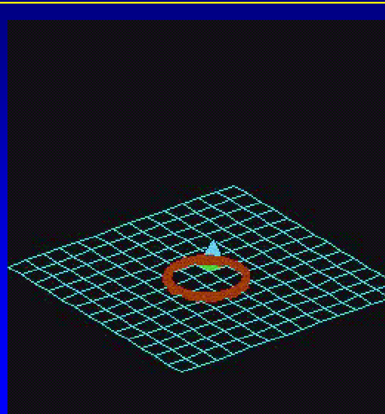
Fast Marching



Expand point on the fringe with minimum value

Update neighbors "downwind"

Fast Marching Visualization



Fast Marching + Level Set for Shape Recovery

1. ***First use the Fast Marching algorithm to obtain "rough" contour***

$$\left|\nabla \mathbf{T}\right| \mathbf{F} = 1, \quad \mathbf{F} = \mathbf{e}^{-\alpha\left|\nabla \mathbf{G}_{\sigma} * \mathbf{I}(x,y)\right|}$$

2. ***Then use the Level Set algorithm to fine tune, using a few iterations, the results from Fast Marching***

$$\frac{\partial \mathbf{\Phi}}{\partial \mathbf{t}} + \mathbf{k}_{\mathbf{I}}(1 - \varepsilon\, \mathbf{K})\left|\nabla \mathbf{\Phi}\right| - \beta\, \nabla \mathbf{P} \cdot \nabla \Phi = \mathbf{0}$$

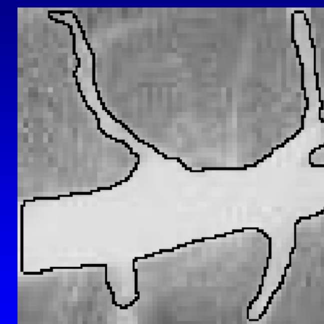
$$k_I = \frac{1}{1 + \left|\nabla G_{\sigma} * I(x,y)\right|}$$

$$P(x,y) = -\left|\nabla G_{\sigma} * I(x,y)\right|$$

Results: Segmentation using Fast Marching

No level set tuning

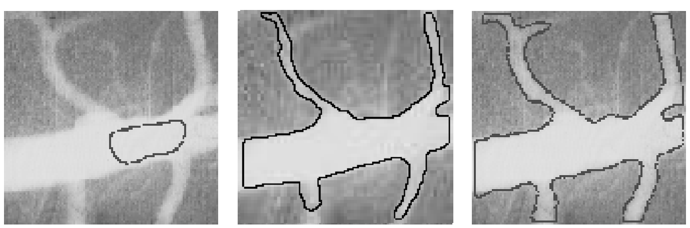
Results: Vein Segmentation



No level set tuning

With level set tuning

Results: Vein Segmentation (continued)

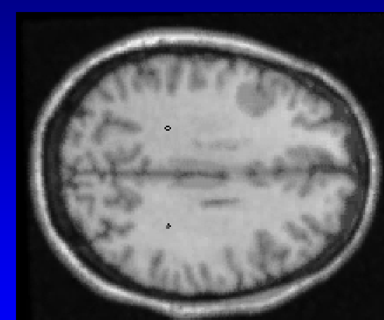


Original

Fast Marching +
Level Set Tuning

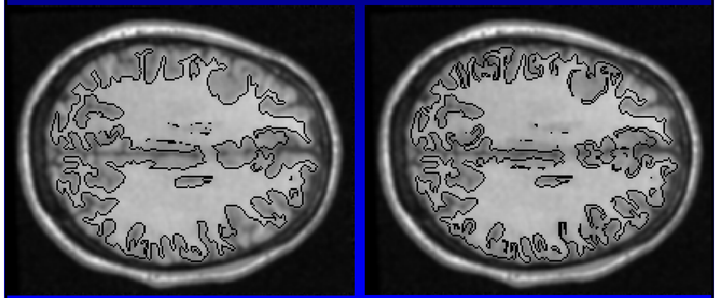
Level Set only

Results: Segmentation using Fast Marching



No level set tuning

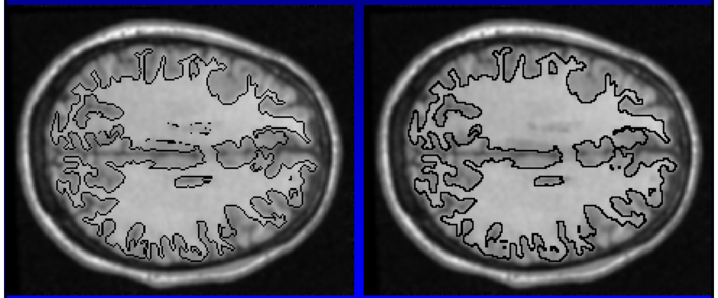
Results: Brain Image Segmentation



# of iterations = 9000 # of iterations = 12000

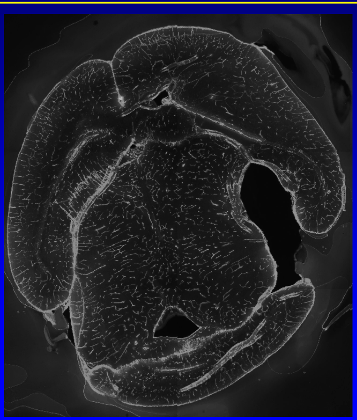
Fast marching only, no level set tuning

Results: Brain Segmentation (continued)



Without level set tuning With level set tuning

Results: Segmentation using Fast Marching



No level set tuning