CS664 Lecture #24: Curve evolution, differential geometry, level sets

Some material taken from:

- **Yuri Boykov & Olga Veksler, Western Ontario**
- **Pushmeet Kohli, Oxford Brookes**
- **Donald Tanguay, HP Labs**
- **Nikos Paragios, Ecole Centrale de Paris**
  http://cermics.enpc.fr/~paragios/tutorial.ppt

Announcements

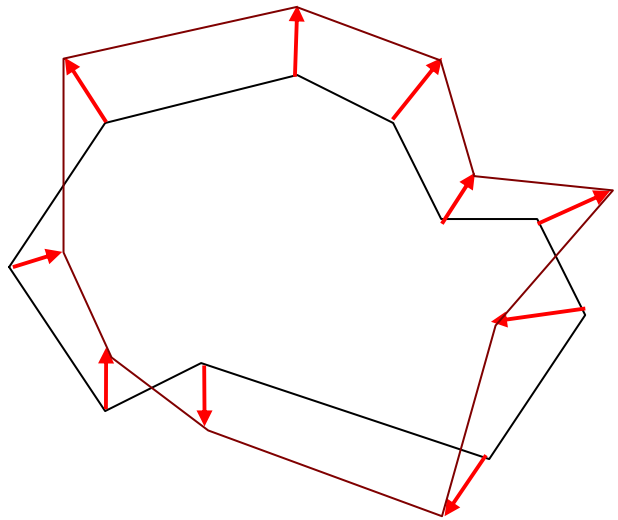
- 1-paragraph final project description due by email on 11/23
- Final quiz will be on 11/29
- PS3 will be out soon, due Friday 12/2
- Final project will be due Thursday 12/15

Curve evolution overview

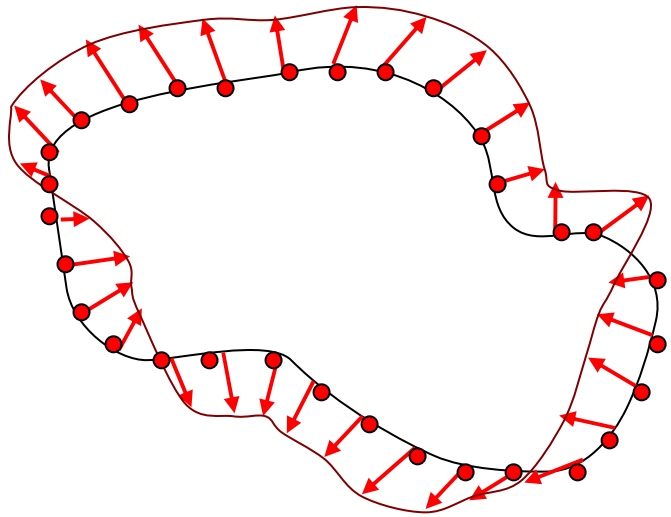
- We will consider “virtual time” within a single image
  - Given an initial curve, move it to line up with image features (i.e., edges)
  - Keep the curve reasonably smooth
- Notation: we will write a curve as $C(p)=(x(p),y(p))$, $p\in[0,1]$
  - Note that we take total derivatives w.r.t. p

Implementing snake evolution

- Discrete:
  - Stored as vertices
  - Each vertex is moved iteratively



- Continuous:
  - Stored as particles
  - Each particle is moved
  - New particles are computed (interpolation)



Curve evolution of snakes

- We can simplify our energy function to

$$E(C) = \int_0^1 \alpha |C_p(p)|^2 + g(C(p))dp.$$

- Calculus of variations says that at a local minimum of the energy we have

$$\alpha C_{pp}(p) + \nabla g(C(p)) = 0.$$

  - Justification: small changes don't change this
  - We can minimize this via gradient descent
  - Many ugly issues with this…

Arc length parameterization

- If we replaced p by ϕ, where $\phi(r)=p$, $r\in[c,d]$, first term in energy would become

$$\int_c^d |(C\circ\phi)'(r)|^2(\phi'(r))^{-1}dr$$

  - Second term is even worse!
- Natural parameterization is in terms of arc length (distance along curve)
  - s(p) is the distance from the origin to p:

$$s(p)=\int_0^p \sqrt{x_q^2(q)+y_q^2(q)}dq$$

Reparameterization

- We can use s to reparameterize the curve
  - Instead of p∈[0,1] use s∈[0,L]

$$L = \int_0^1 \left| C_p \right| dp$$

- Many nice properties, e.g.
  - Unit velocity $|C_s| = 1$
  - Curve and its derivatives are invariant to rotation and translation
  - Intrinsic properties of curves don't depend upon parameterizations, only on curves
    - With this parameterization, curve and all derivatives are intrinsic

Forces acting on the curve

- Instead of an energy function E that we minimize, we think about a force acting on the curve over time
  - Force can do gradient descent over E

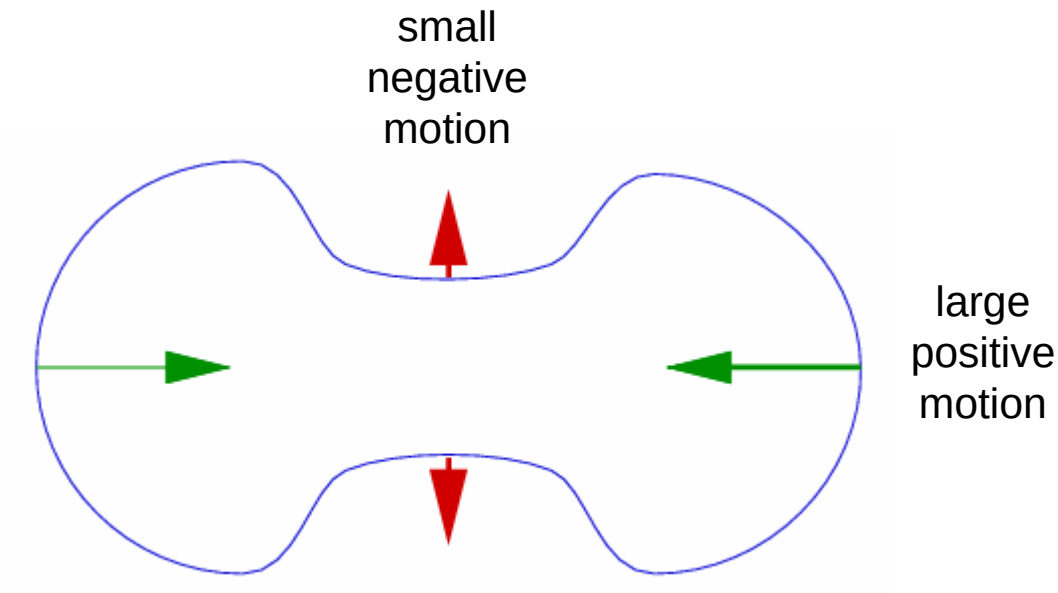
At minimum: $\alpha C_{ss}(s) + \nabla g(C(s)) = 0$

Curve evolution: $C_t = \alpha C_{ss}(s) - \nabla g(C(s))$

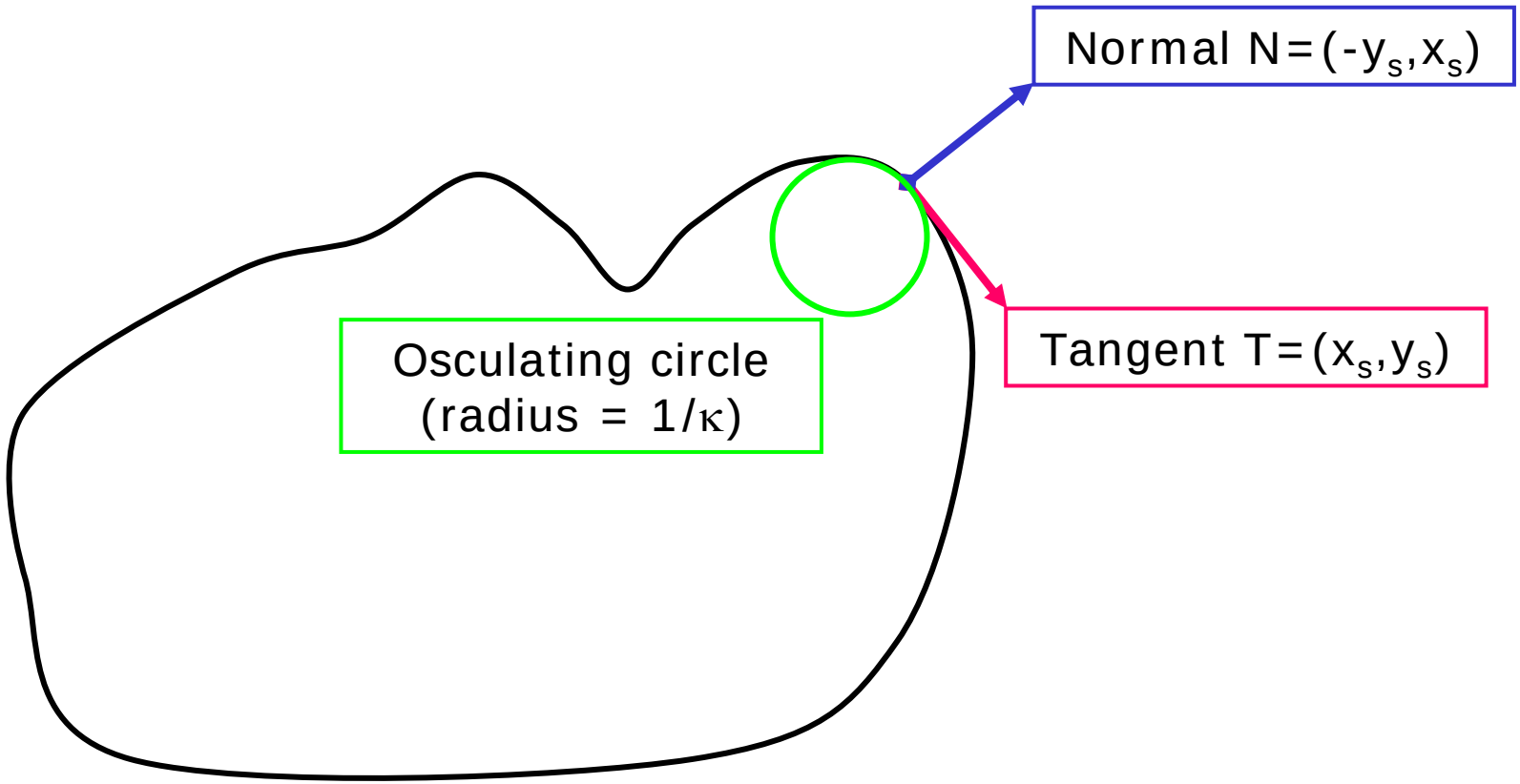
Generalized curve evolution

- Natural forces involve internal and external terms
  - As usual, external are easy
    - Roll towards edges in the image
  - Internal ones have many interesting variants
- Curvature flow
  - Smooths out curves (geometric heat equation)
  - Each particle moves perpendicularly to the curve with speed proportional to curvature

Curvature flow



Local coordinate system



Geometric flows

- Natural coordinates are defined by N,T
  - $C_t = V_N N + V_T T$
  - We can ignore the second term since it only affects the particles, not the curve
- For a geometric flow, $\beta = V_N$ is independent of the parameterization
  - Geometric heat equation has $\beta = \kappa$
  - This takes any simple (non-self intersecting) curve first into a convex shape, then into a circular point that vanishes
  - Time to vanish is area enclosed

Geodesic active contours

- An elegant solution which is very close to the original snake energy is simply

$$E(C) = \int g(C(s))ds$$

  - Where did the internal energy go??
  - Replaced by minimizing curve length
    - Geodesic, where g controls the local metric
- Curve evolution to minimize E is

$$C_t = (\kappa g - \langle \nabla g, N \rangle)N$$

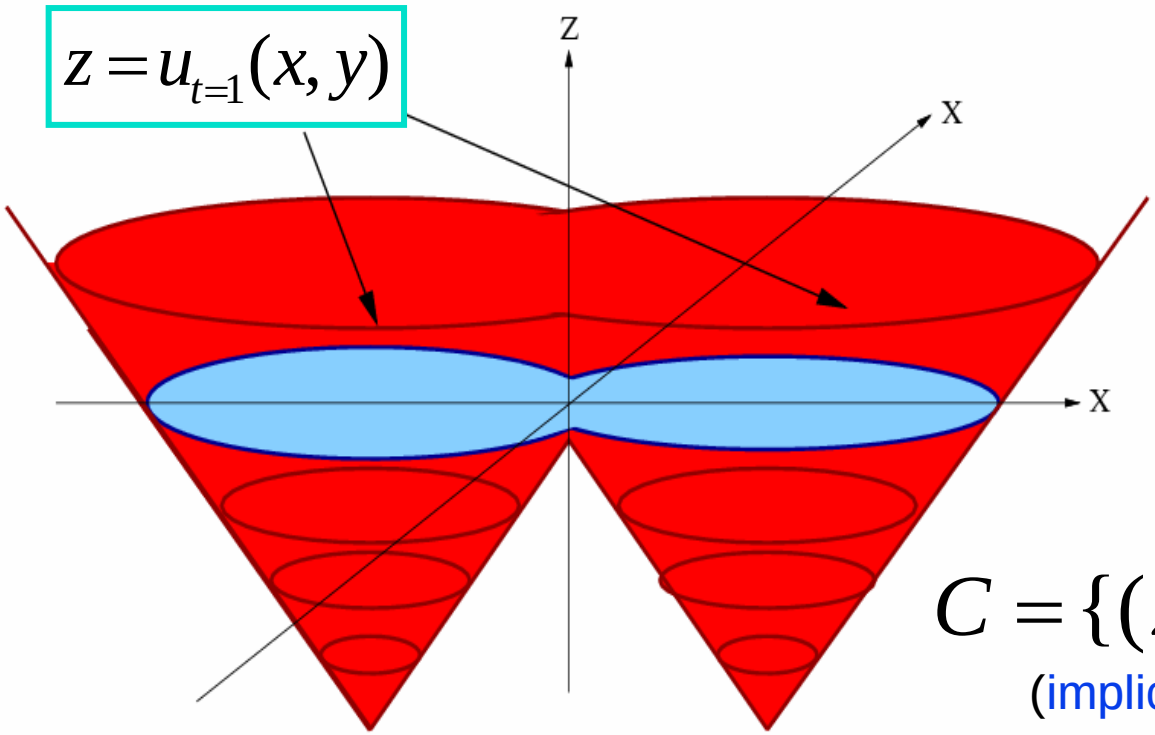
Level sets

- Curve evolution is great. But:
  - Topology changes are a major problem
  - Also, numerical issues and stopping conditions
    - Particles tend to colide, and solutions oscillate
- Level sets address these beautifully
- Basic idea: implicit function for curve
  - Think of a topological map of a park
  - There are curves drawn at a fixed height
    - I.e., the points where height = 1 mile
  - We will use this surface to evolve the curve

Level set surfaces

- To evolve a curve **C**, first compute a surface **u** such that **C** is level set of **u**
  - Obviously, not unique
  - Standard choice: signed distance function
    - **u**(x,y) = distance to nearest point in **C**
    - Negative if inside of **C**
- Instead of evolving the curve, we evolve the surface!
  - Sometimes called “contour propagation”, “interface tracking”, etc.

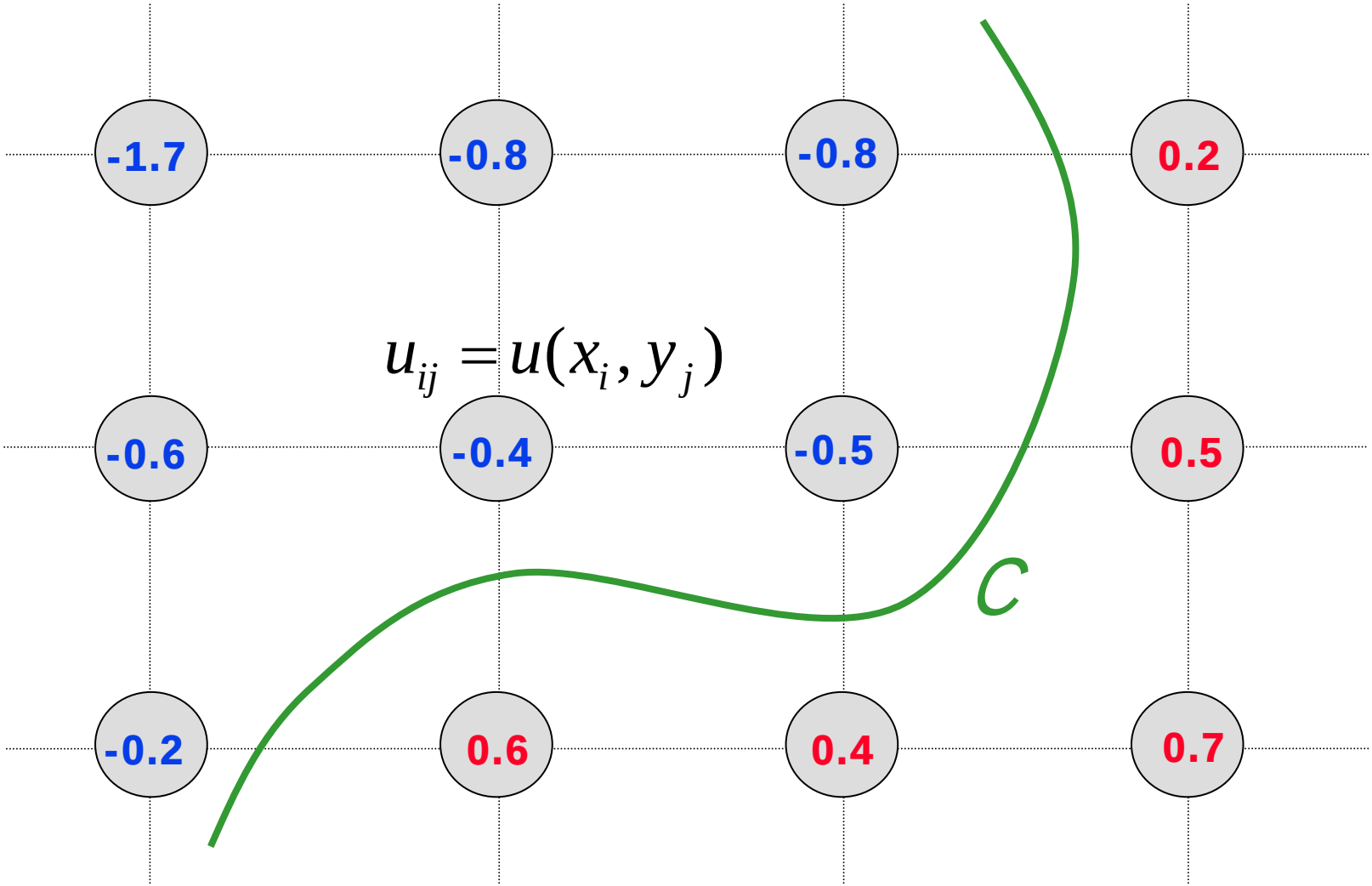
Example



$$C = \{(x, y) : u(x, y) = 0\}$$

(implicit contour representation)

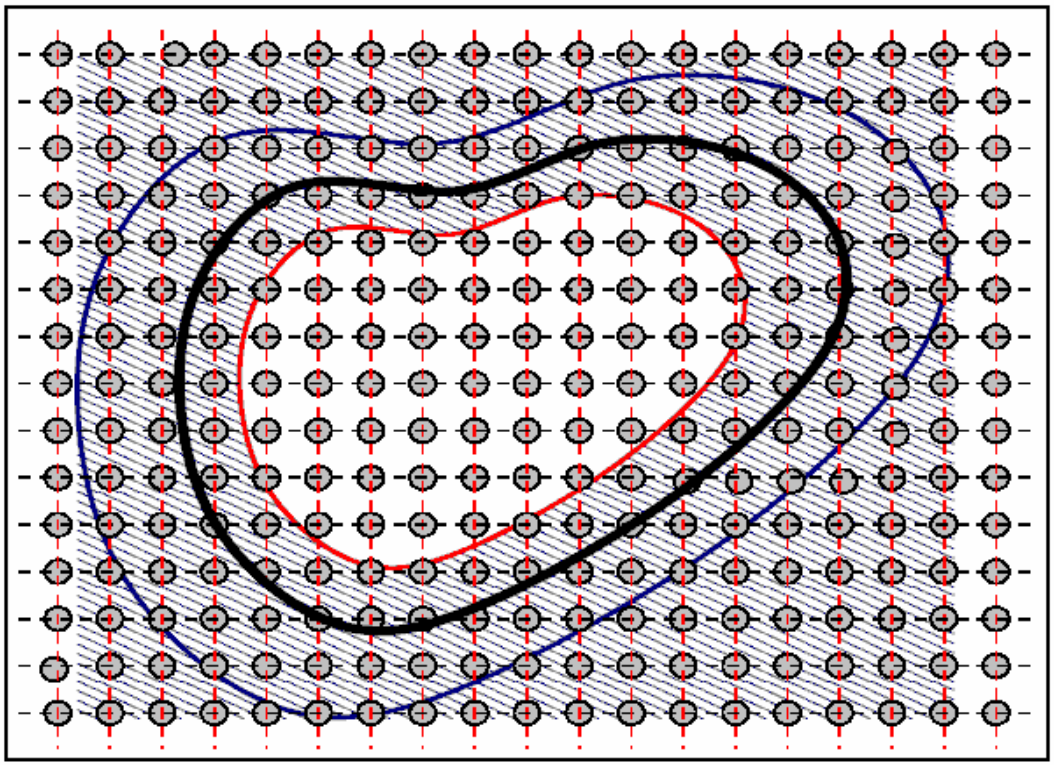
Example



Level set evolution

- Instead of evolving **C** we evolve **u**
- Curve evolution: $\frac{\partial \mathbf{C}}{\partial t} = \beta N$
- Level set evolution: $\frac{\partial \mathbf{u}}{\partial t} = \beta \|\nabla \mathbf{u}\|$
- Amazing fact: if you evolve **u** by $\mathbf{u}_t$, *every* level set of **u** will evolve according to $\mathbf{C}_t$

Speedup: narrow band



Outward Band
$\Phi(s) = +d$

Front Position
$\Phi(s) = 0$

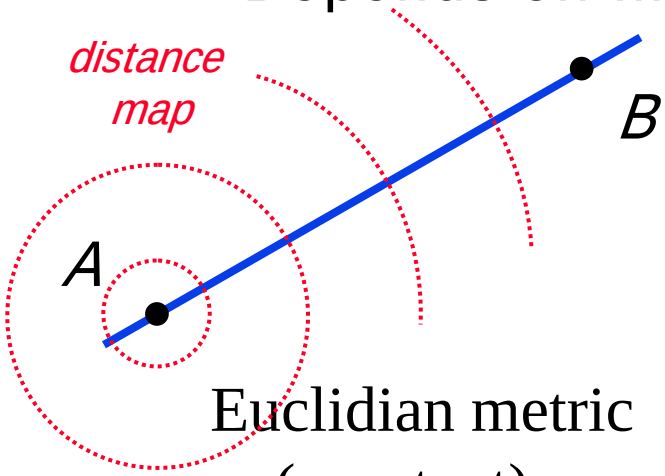
Inward Band
$\Phi(s) = -d$

State of the art

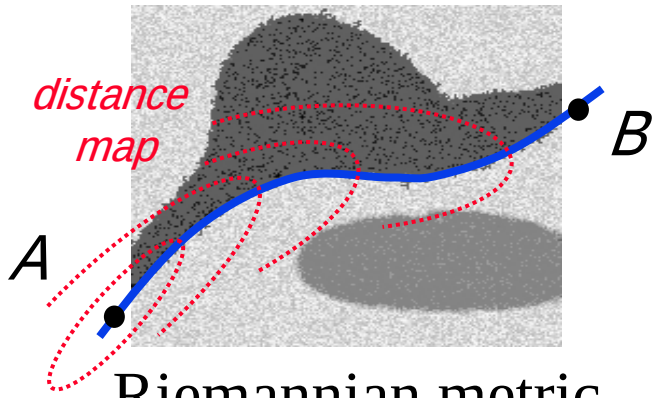
- Almost everyone who does curve evolution these days uses level sets
  - Handles changes in topology
  - Numerical stability
  - Natural generalization to higher dimensions
    - Evolving surfaces (“minimal surfaces”)
- Typically add some sort of constant force (“balloon pressure”) plus curvature
- It would be great to have a better way to incorporate shape

Better optimization?

- Geodesic active contours are very elegant
  - Use a local (Riemannian) metric
    - Depends on image features



Euclidian metric
(constant)

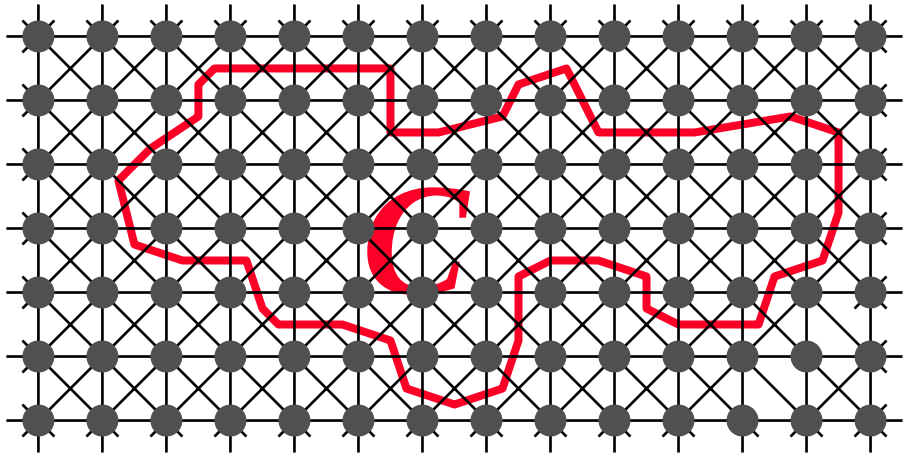


Riemannian metric
(space varying, tensor $D(p)$)

- But level sets only yield a local optimum!

Can we do better?

- Shortest paths are 1D only, but it seems like something graph-like could be used…



$$\|C\| = \sum_{e \in C} |e|$$

- Cost of a cut is “length” (“area”) of the corresponding curve (surface)

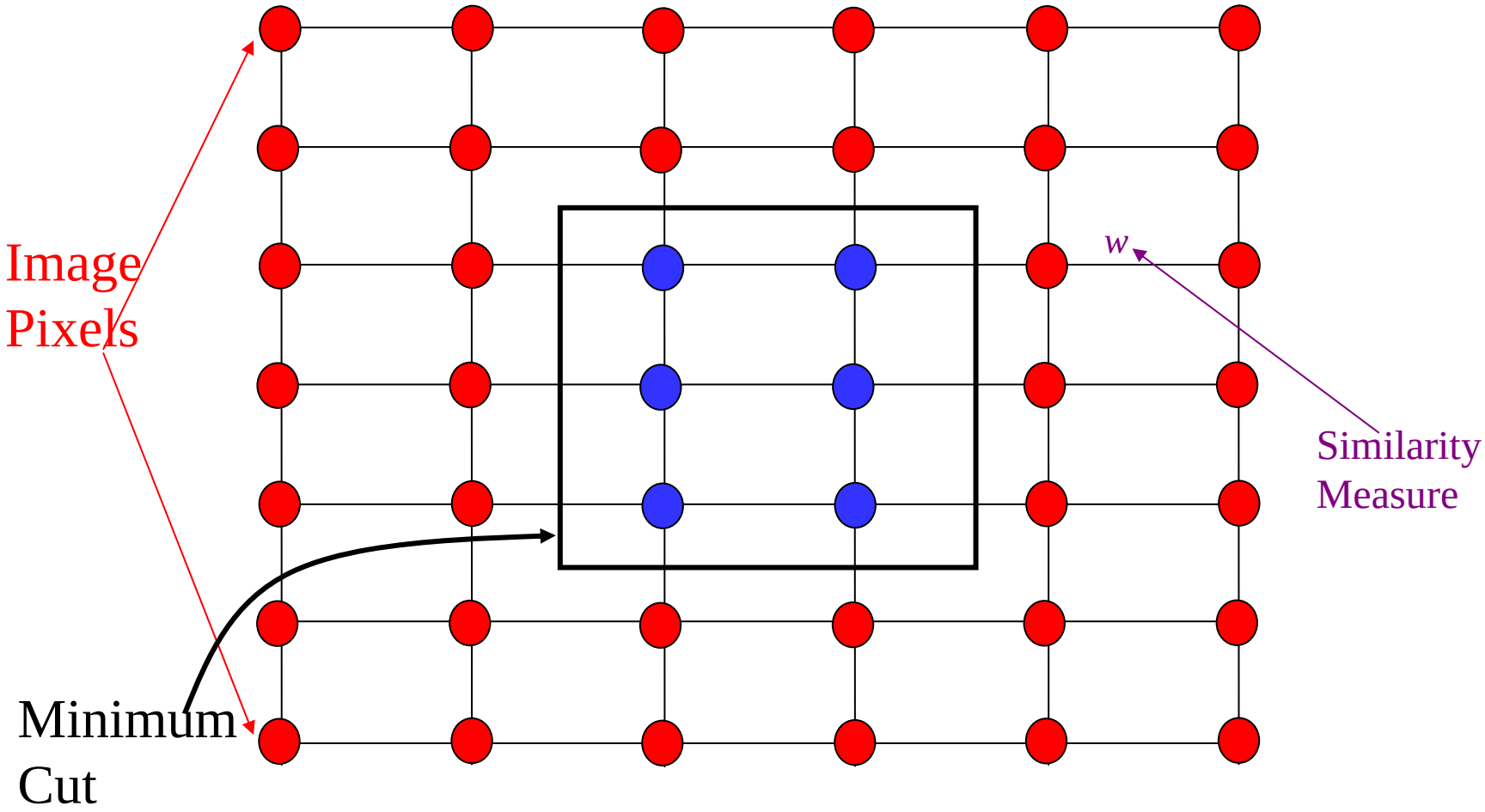
Geo-cuts

- It is possible to choose the edge weights so that the cost of a cut corresponds to the length (area)
  - Under arbitrary Riemannian metric, or beyond
- You can find a minimal surface which is actually minimal!
  - Under given boundary conditions
  - As in binary segmentation from graph cuts

Segmentation via min cut?

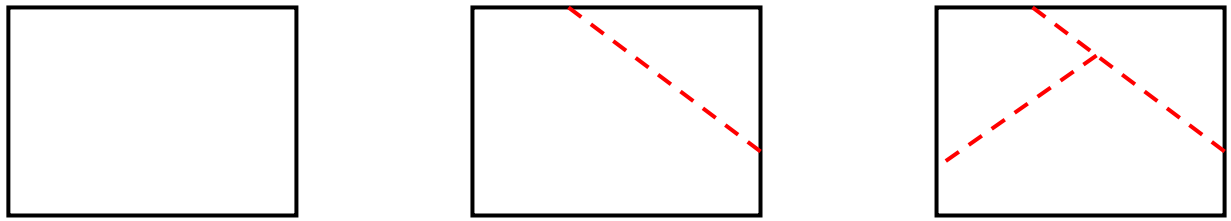
- Why can't we use min cut to directly segment an image?
  - Weights between edges are "affinities"
    - I.e., decreasing function of $|I_p - I_q|$
  - Where do the source and sink go?
- Two answers
  - Use variant on min cut where there is no source and sink
    - E.g., Karger's algorithm
  - Nested cuts

Segmentation via min cut

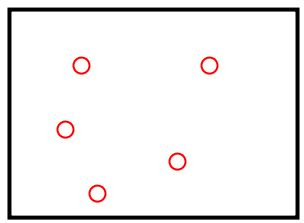


Wu and Leahy

- Recursively split the image in 2
  - Each stage is fast

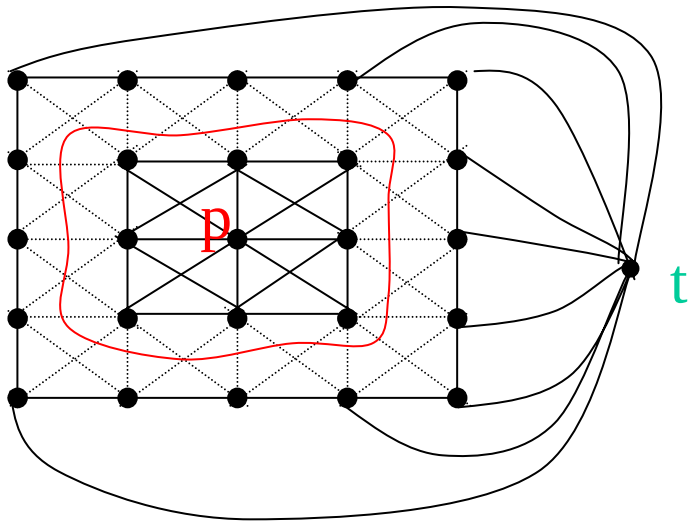


- Problems:
  - Image may have no natural binary split!
  - Bias towards small segmentations

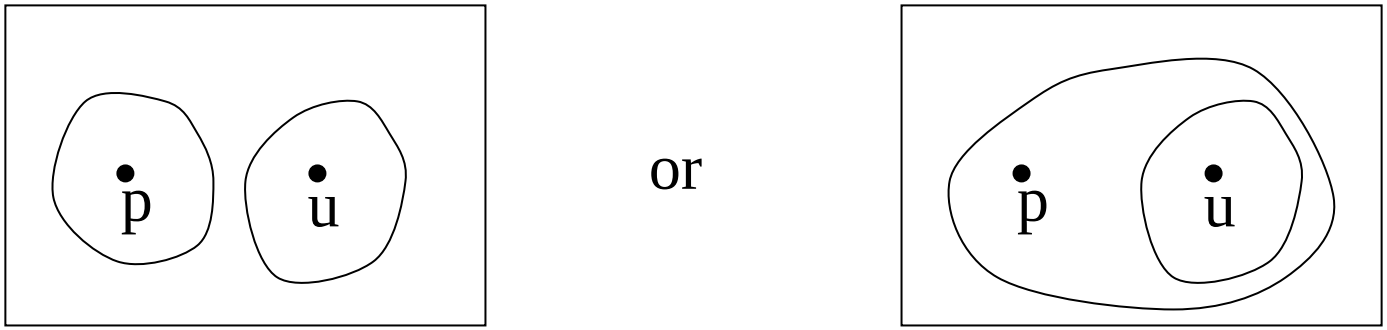


Nested cuts (Veksler)

- Each pixel should be in a “good” region
  - Compute an p-t min cut for each pixel p
    - t will lie outside the image boundary



Properties



Recurse for each region: