Lecture 11
Image segmentation-II: Watershed and active contours

Dr. Xiran Cai

Email: caixr@shanghaitech.edu.cn

Office: 3-438 SIST

Tel: 20684431

ShanghaiTech University

Previously

- **Pixel-based segmentation:**

  each pixel is segmented based on gray-level values, no contextual information, only histogram.

  -Example: Hough transform

- **Edge-based segmentation:**

  Detects and links edge pixels to form contours.

- **Region-based segmentation:**

  considers gray-levels from neighboring pixels by

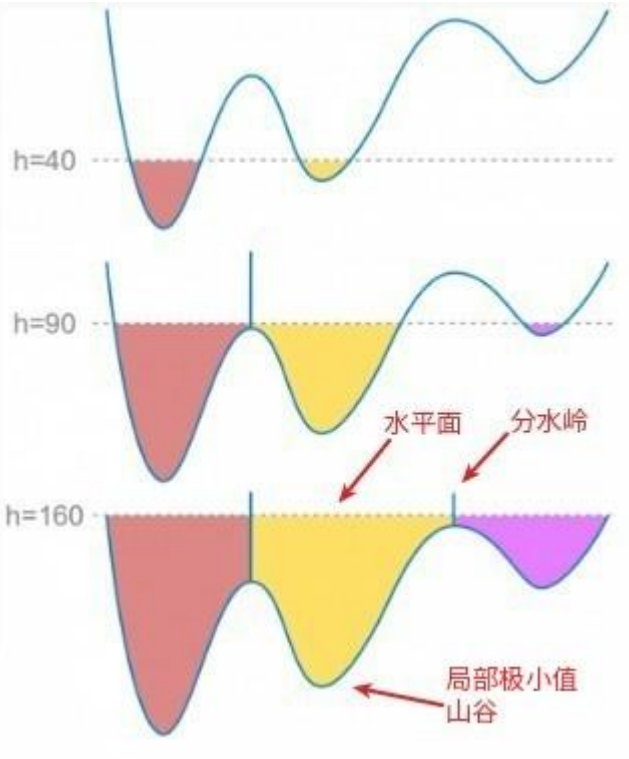
  – including similar neighboring pixels (region growing),

  – split-and-merge,

  – super-pixel segmentation (k-means).

Watershed segmentation (分水岭)

- Look at the image as a 3D topographic surface, (x,y,intensity), with both valleys and mountains.
- Assume that there is a hole at each minimum, and that the surface is immersed into a lake.
- The water will enter through the holes at the minima and flood the surface.
- To avoid two different basins to merge, a dam is built.
- Final step: the only thing visible would be the dams.
- The connected dam boundaries correspond to the watershed lines.



Watershed segmentation

- Can be used on images derived from:
  - Intensity image
  - Edge enhanced image
  - Distance transformed image
  - Thresholded image. From each foreground pixel, compute the distance to a background pixel.
  - Gradient of the image
- Most common: **gradient image**

Watershed algorithm

- Let $g(x, y)$ be the input image (often a gradient image).
- Let $M_1, \cdots, M_R$ be the coordinates of the regional minima.
- Let $C(M_i)$ be a set consisting of the coordinates of all points belonging to the catchment basin associated with the regional minimum M_i.
- Let $T[n]$ be the set of coordinates (s, t) where $g(s, t) < n$

$$T[n] = \{(s, t) | g(s, t) < n\}$$

This is the set of coordinates lying below the plane $g(x, y) = n$

This is the candidate pixels for inclusion into the catchment basin, but we must take care that the pixels do not belong to a different catchment basin.

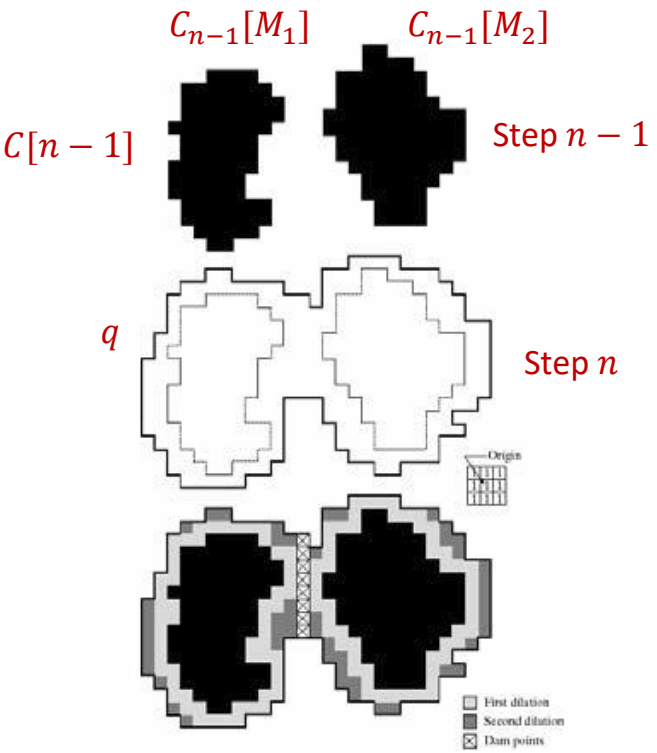
Watershed algorithm

- The topography will be flooded with integer flood increments from $n = \min - 1$ to $n = \max + 1$.
- Let $C_n(M_i)$ be the set of coordinates of points in the catchment basin associated with M_i, flooded at stage n.
- This must be a connected component and can be expressed as $C_n(M_i) = C(M_i) \cap T[n]$ (only the portion of $T[n]$ associated with basin M_i)
- Let $T[n]$ be the set of flooded water, let $C[n]$ be union of all flooded catchments at stage n:

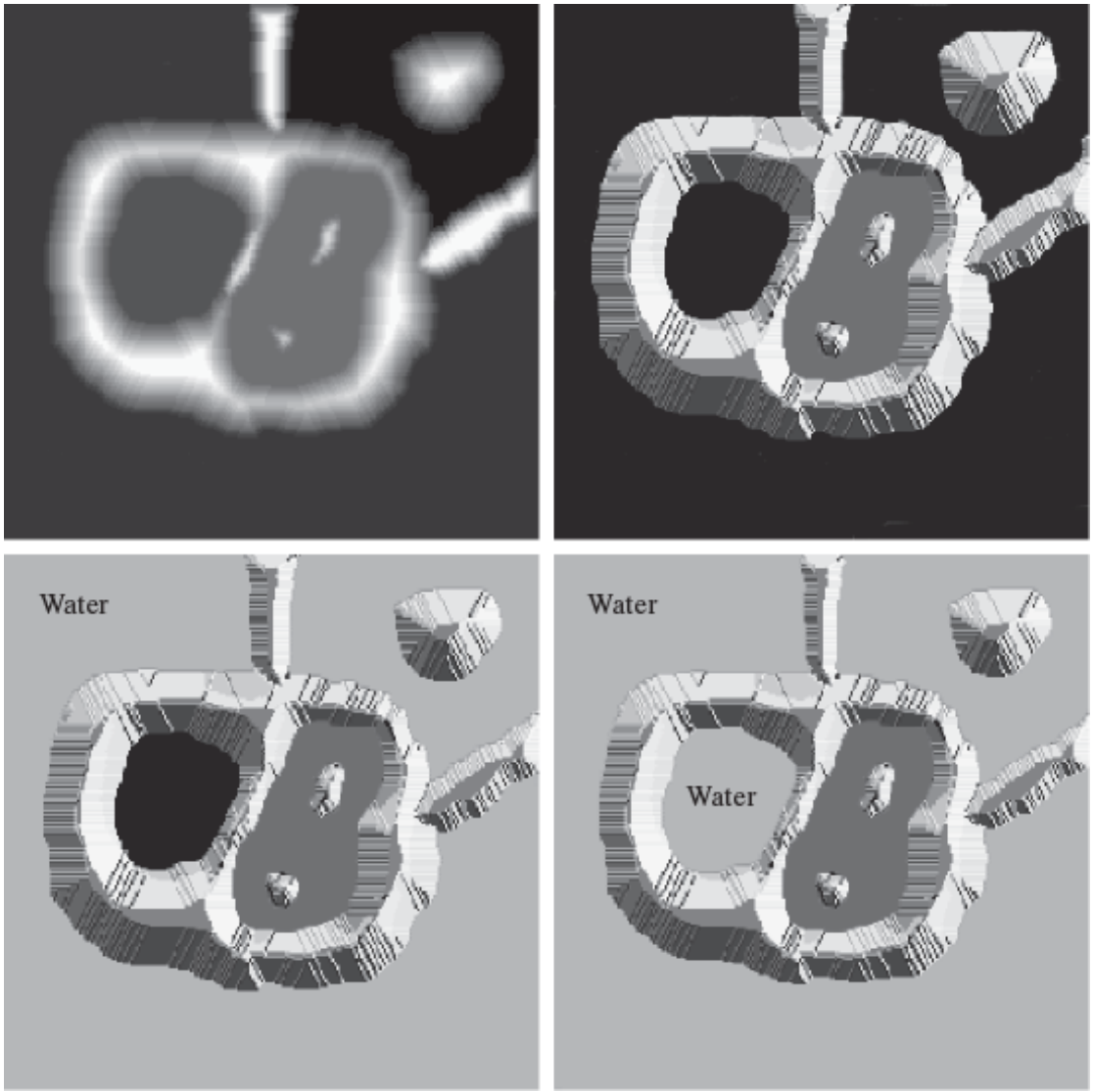
$$C[n] = \bigcup_{i=1}^{R} C_n(M_i) \text{ and } C[max + 1] = \bigcup_{i=1}^{R} C(M_i)$$

Dam construction

- Stage $n-1$: two basins forming separate connected components.
- To consider pixels for inclusion in basin k in the next step (after flooding), they must be part of $T[n]$, and also be part of the connected component q of $T[n]$ that $C_{n-1}[k]$ is included in.
- Use morphological dilation iteratively.
- Dilation of $C[n-1]$ is constrained to q.
- The dilation can not be performed on pixels that would cause two basins to be merged (form a single connected component)



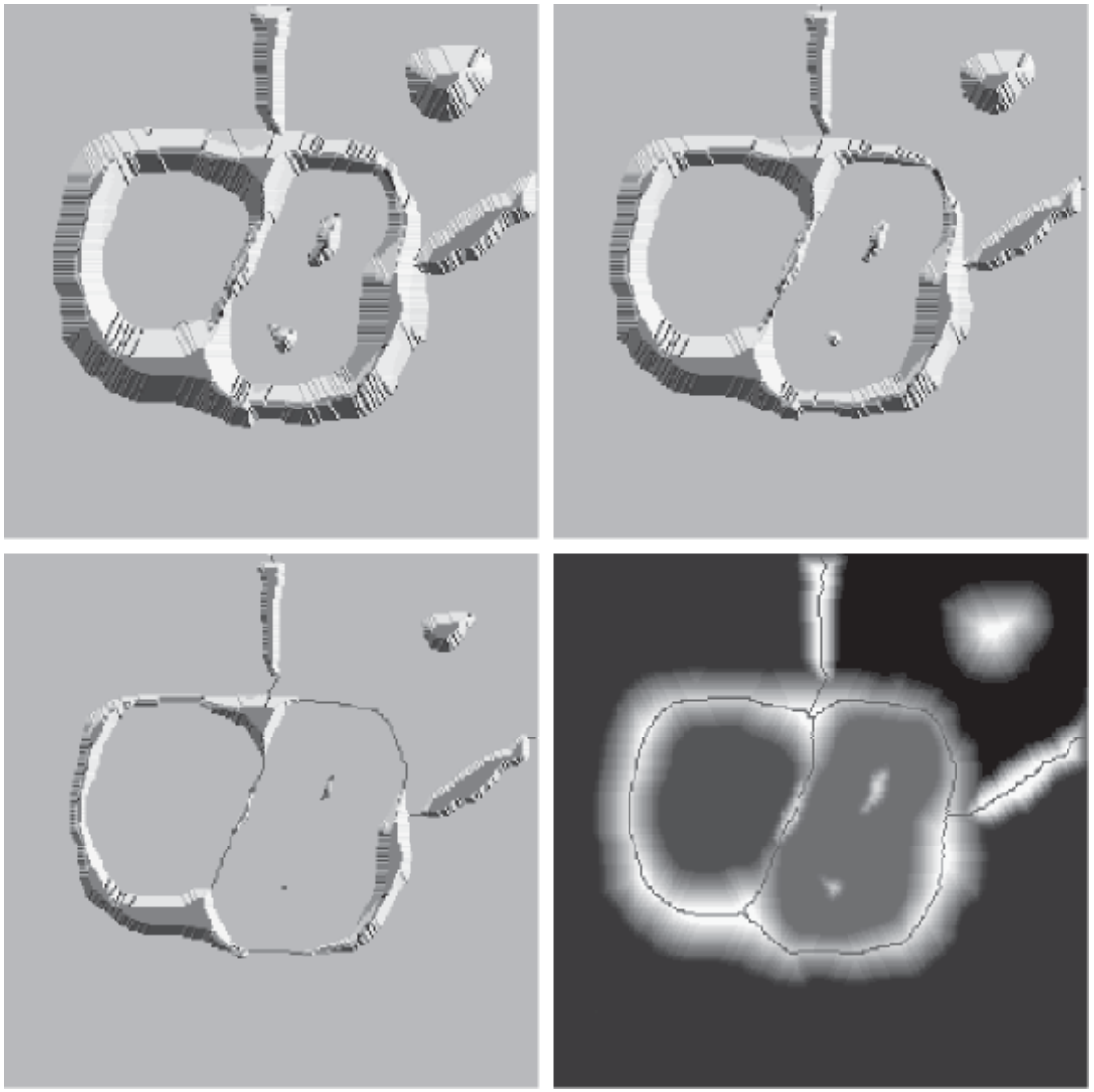
Watershed procedure (1)



a c
b d

FIGURE 10.60
(a) Original image.
(b) Topographic view. Only the background is *black*. The basin on the left is slightly lighter than black.
(c) and (d) Two stages of flooding. All constant dark values of gray are intensities in the original image. Only constant *light gray* represents "water."
(Courtesy of Dr. S. Beucher, CMM/ Ecole des Mines de Paris.)
(*Continued on next page.*)

Watershed procedure (2)

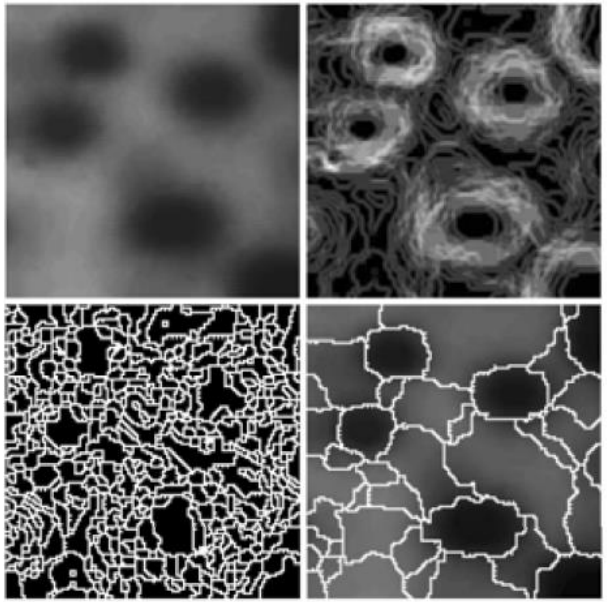


e f
g h

FIGURE 10.60
(Continued)
(e) Result of further flooding. (f) Beginning of merging of water from two catchment basins (a short dam was built between them). (g) Longer dams. (h) Final watershed (segmentation) lines superimposed on the original image. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

Over-segmentation or fragmentation

Image I



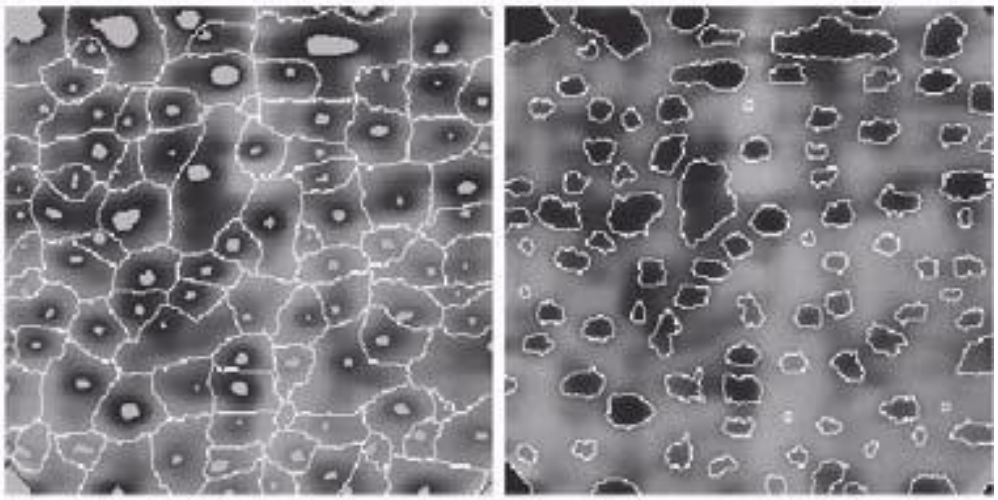
Gradient magnitude image (G)

Watershed of G

Watershed of smoothed G

- Using the gradient image directly can cause over-segmentation because of noise and small irrelevant intensity changes.
- Improved by smoothing the gradient image or using markers

Solution: Watershed with markers



- A marker is an extended connected component in the image
- Can be found by intensity, size, shape, texture etc
- Internal markers are associated with the object (a region surrounded by bright point (of higher altitude))
- External markers are associated with the background (watershed lines)
- Segment each sub-region by some segmentation algorithm

Watershed summary

- **Advantages**
  - Gives connected components
  - A priori information can be implemented in the method using markers
- **Disadvantages :**
  - Often needs preprocessing to work well
  - Fragmentation or “over-segmentation” can be a problem

Active Contours (SNAKEs)

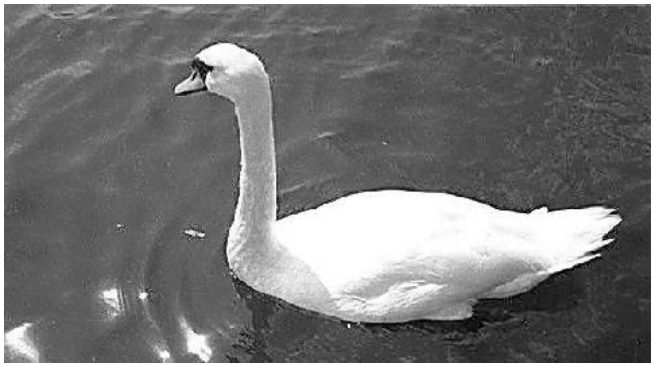
- Back to boundary detection
  - This time using perceptual grouping.
  - This is non-parametric
- We're not looking for a contour of a specific shape.
  - Just a good contour.

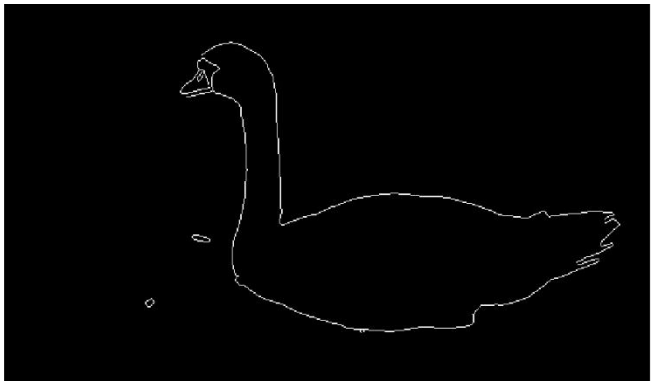
About SNAKEs

- Kass, Witkin and Terzopoulos, IJCV.
- “Dynamic Programming for Detecting, Tracking, and Matching Deformable Contours”, by Geiger, Gupta, Costa, and Vlontzos, IEEE Trans. PAMI 17(3)294-302, 1995
- E. N. Mortensen and W. A. Barrett, Intelligent Scissors for Image Composition, in ACM Computer Graphics (SIGGRAPH `95), pp. 191-198, 1995

Boundary following

Good case





Bad case

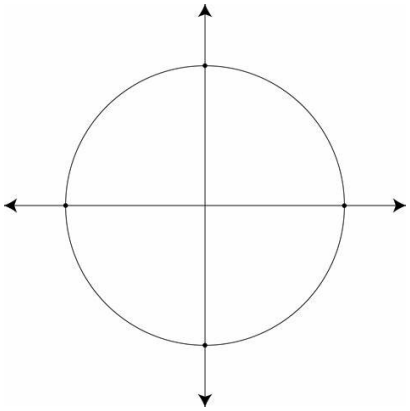
Improve Boundary Detection

- Idea: segment using curves, not pixels.
- We want a segmentation curve that
  1. **Conforms to image edges.**
  2. **Generates a smooth and varying curve.**

Parametric Curves

- Consider $\begin{bmatrix} x(s) \\ y(s) \end{bmatrix}$ $s \in [0,1]$ continuous.

- e.g.,

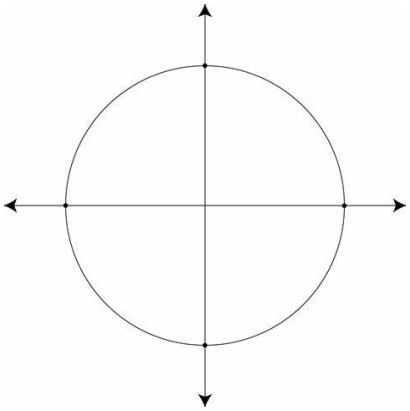


??

Parametric Curves

- Consider $\begin{bmatrix} x(s) \\ y(s) \end{bmatrix}$ $s \in [0,1]$ continuous.

- e.g.,

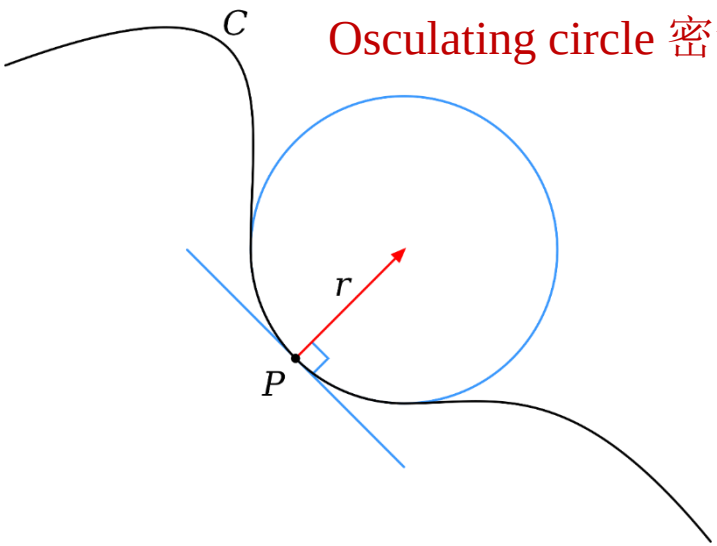


$$\begin{bmatrix} x(s) \\ y(s) \end{bmatrix} = \begin{bmatrix} r\cos(2\pi s) \\ r\sin(2\pi s) \end{bmatrix}$$

$$\begin{bmatrix} x(s) \\ y(s) \end{bmatrix} = \begin{bmatrix} r(s)\cos(2\pi s) \\ r(s)\sin(2\pi s) \end{bmatrix}$$

- We define a curve using $C(s) = [x(s), y(s)]$.

Curvature 曲率

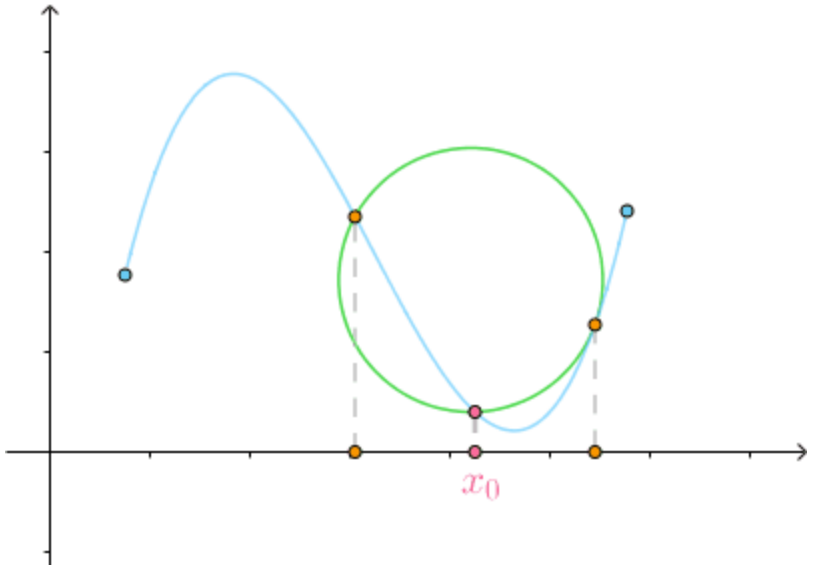


Osculating circle 密切圆

$$\text{Curvature } K(s) = \frac{1}{R(s)}$$

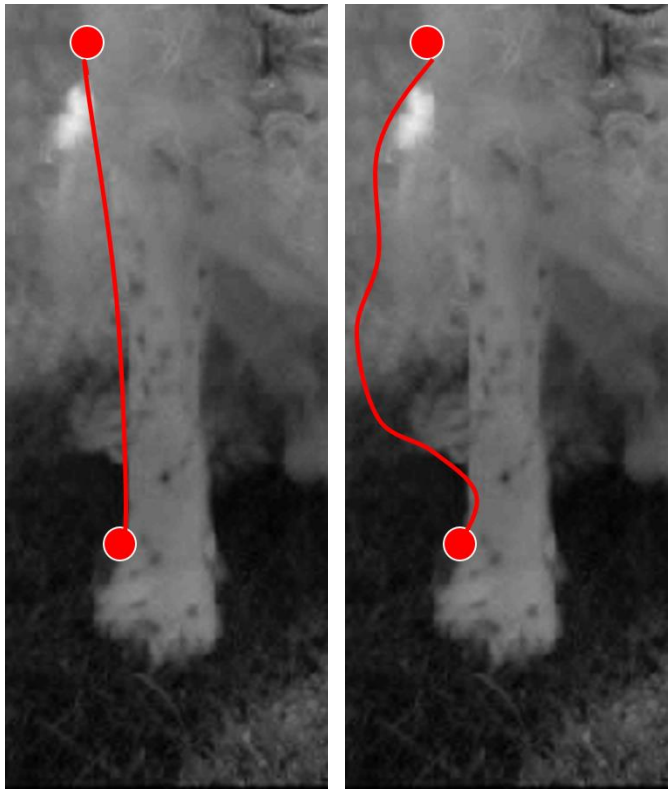
- Let *C* be a plane curve (the precise technical assumptions are given below). The curvature of *C* at a point is a measure of how sensitive its tangent line is to moving the point to other nearby points.
- It is natural to define the curvature of a straight line to be constantly zero. The curvature of a circle of radius *r* should be large if *r* is small and small if *r* is large. Thus the curvature of a circle is defined to be the reciprocal of the radius.

Curvature 曲率



Curvature of plane curves

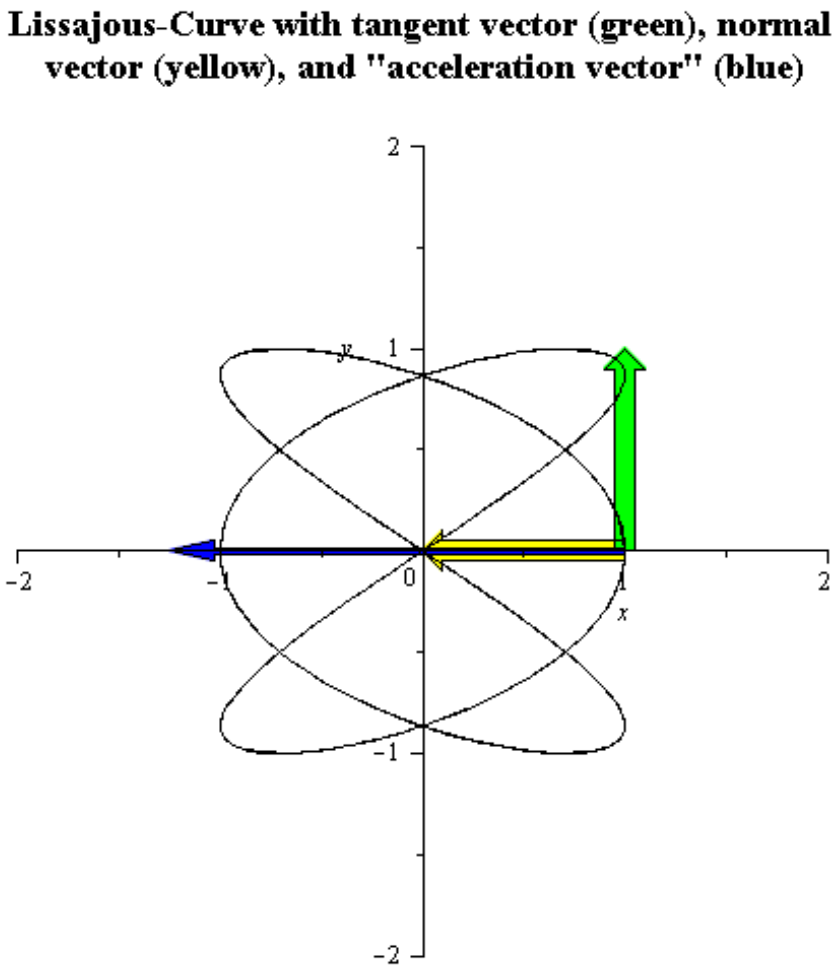
- How do we decide how good a path is? Which of two paths is better?

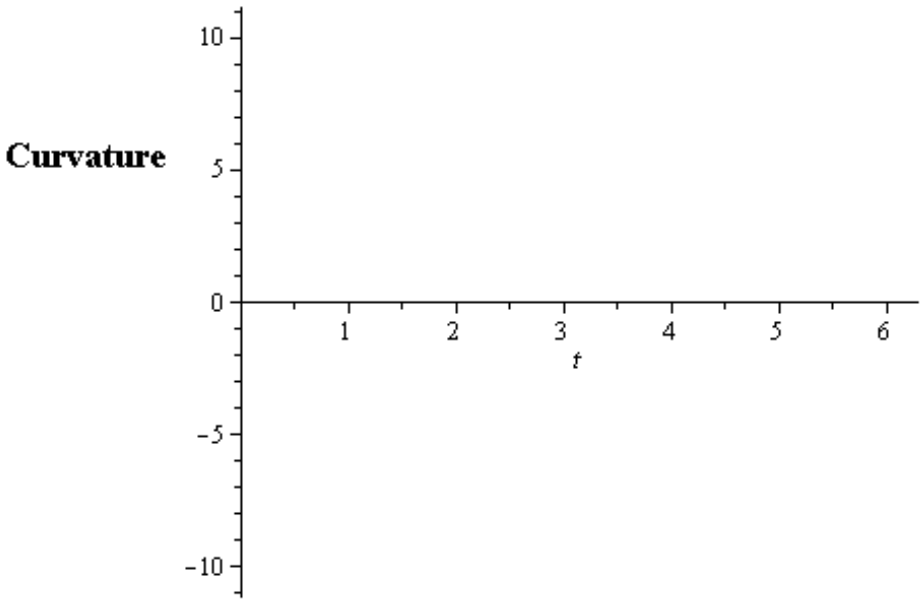


- $T(s) = C'(s)$ is considered as the velocity vector or the unit tangent vector of the curve $C(s)$.
- $\kappa(s) = \frac{1}{R(s)} = C"(s)$ is the curvature of curve $C(s)$.

$$k(t) = \frac{x_1'(t) \cdot x_2''(t) - x_1''(t) \cdot x_2'(t)}{\left(x_1'(t)^2 + x_2'(t)^2\right)^{\frac{3}{2}}}, \qquad N(t) = \frac{1}{\|\gamma'(t)\|} \cdot \begin{bmatrix} -x_2'(t) \\ x_1'(t) \end{bmatrix}$$

Curvature of plane curves





Find energy for the curve

- Idea: we want to define an energy function $E(c)$ that matches our intuition about what makes a good segmentation.
- Curve will iteratively evolve to reduce/ minimize $E(c)$.
- $E(c) = E_{internal}(c) + E_{external}(c)$.
  - ✓ $E_{internal}(c)$ depends only on the shape of the curve.
  - ✓ $E_{external}(c)$ depends on image intensities.

Energy for shape of the curve

$$E_{internal}(c) = \int_0^1 \alpha \|c'(s)\|^2 + \beta \|c''(s)\|^2 \, ds.$$

- *Low* $c'(s)$ keeps curve not too "stretchy"
- *Low* $c''(s)$ keeps curve not too "bendy"

Energy for image intensities

$$E_{external}(c) = \int_0^1 -\|\nabla I(c(s))\|^2 ds$$

$$= \int_0^1 -\{\left[\frac{\partial I}{\partial x}(x(s), y(s))\right]^2 + \left[\frac{\partial I}{\partial y}(x(s), y(s))\right]^2\} ds$$

- No edge , then $\nabla I=0$, $E_{external}(c) = 0$
- Big edge, then ∇I=big positive value , $E_{external}(c)$ =big negative value

How to minimize *E(c)*?

- **Requires: variational calculus. (变分微积分)**

  1. In practice for digital images, we solve the problem by creating a curve $C(s,t)$. Where t represents the iteration.
  2. Curve approximated by k discrete points (x_i, y_i).
  3. Then we step $C(s,t-1)$ to $C(s,t)$ by taking a step along gradient of $E(C)$: $\frac{\partial E}{\partial C}$.
  4. A snake minimize $E(C)$ must satisfy the Euler equation: $-\nabla E_{external}(c) = 0$.

- **Result: Curve inches along until points around perimeter stop changing.**

Try this

- Launch “snake.m”.
- Load image “circle”. Click the button “Set new points”, initialize starting points.
- Click the button “start” to start.

Problem with basic snake (E_{ext})

- Contour never "sees" strong edges that are far away.
- Small gradient: Snake gets hung up.
- When there is no gradient for external Energy, then only internal Energy working.
- Can not work for outer boundary.

Gradient vector flow (GVF)

- **Idea: instead of using exactly the image gradient, create a new vector field over image plane:**
- $\vec{V}(x,y) = \begin{bmatrix} V_x(x,y) \\ V_y(x,y) \end{bmatrix}$ **vector field of the curve.**
- $\vec{e}(x,y) = \begin{bmatrix} e_x(x,y) \\ e_y(x,y) \end{bmatrix}$ **vector field of the edge map in an image.**
- $\vec{V}(x,y)$ **is defined to minimize:**
- $\iint \mu[\left(\frac{\partial V_x}{\partial x}\right)^2 + \left(\frac{\partial V_x}{\partial y}\right)^2 + \left(\frac{\partial V_y}{\partial x}\right)^2 + \left(\frac{\partial V_y}{\partial y}\right)^2] + \|\nabla e\|^2 \left\|\vec{V} - \vec{e}\right\|^2 \, dxdy$
- **Intuition:** ∇e **is big: gradient is large,** $\vec{V}$ **follows edge gradient faithfully;** ∇e **is small: gradient is small, follows along to be as smooth as possible, trades off smooth vs how faithful.**

C. Xu and J.L. Prince, ``Gradient Vector Flow: A New External Force for Snakes," Proc. IEEE Conf. on Comp. Vis. Patt. Recog. (CVPR), Los Alamitos: Comp. Soc. Press, pp. 66-71, June 1997

Extensions

- Active shape models
- Active appearance models
- Level sets
- FAST: FMRIB's Automated Segmentation Tool

Discussion

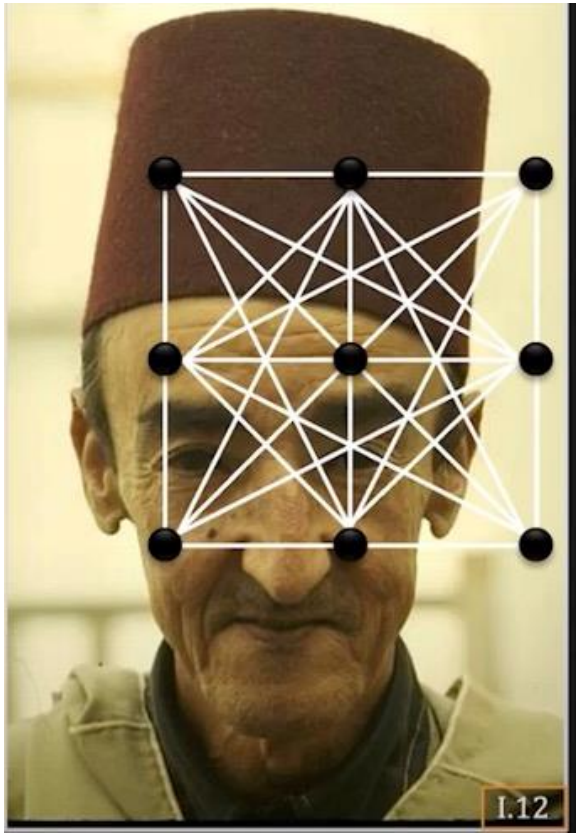
- Try your own image with convolutional snake and GVF snake using the provided implementation or looking for better demon online.
- Find out what snake can do and what snake can not.

Graph-cut segmentation

- Images as Graphs:
  - A vertex for each pixel

Graph-cut segmentation

- Images as Graphs:
  - A vertex for each pixel
  - An edge between each pair of pixels
  - Graph Notation: G = {V, E},where V and E are the sets of vertices and edges, respectively.



$$G = \{V, E\}$$

V: Graph nodes ⟷ Image={pixels}

E: edges connection nodes — Pixel similarity

Segmentation = Graph partition

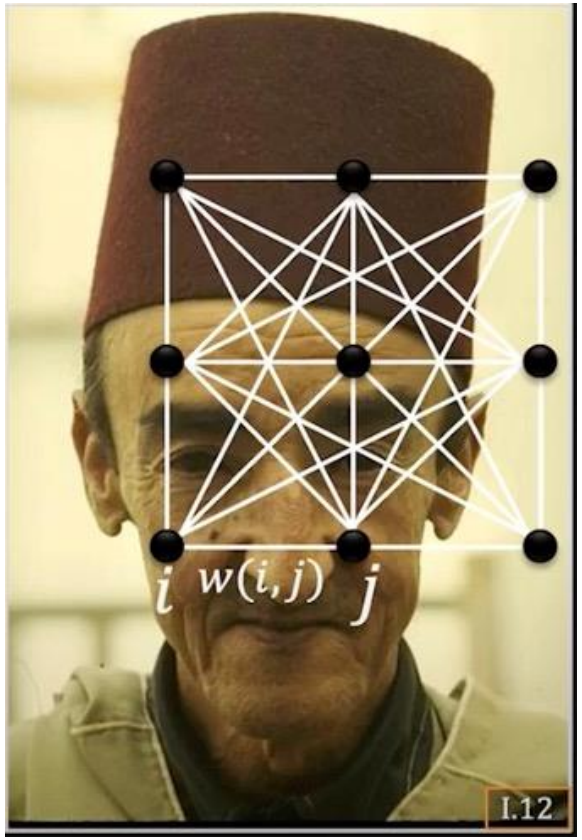
Graph-cut segmentation

- Images as Graphs:
  - A vertex for each pixel
  - An edge between each pair of pixels
  - Graph Notation: G = {V, E},where V and E are the sets of vertices and edges, respectively.
  - Pixel Dissimilarity

$$S(\mathbf{f}_i, \mathbf{f}_j) = \sqrt{\left(\sum_k (f_{ik} - f_{jk})^2\right)}$$

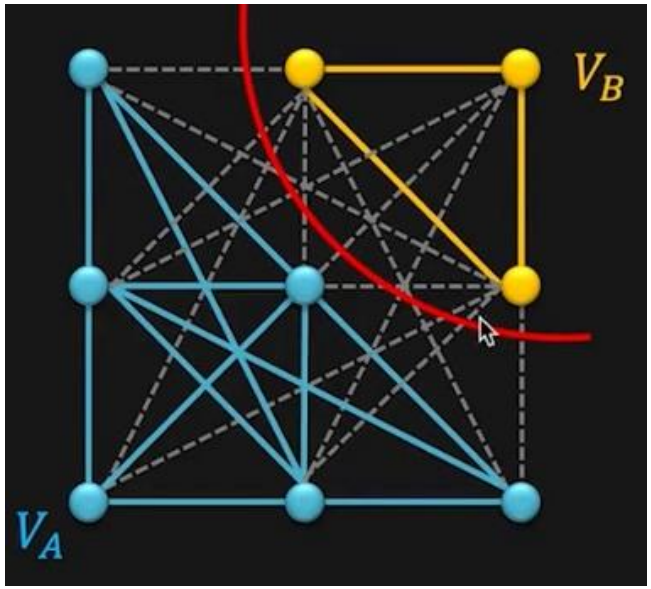
  - Pixel Affinity:

$$w(i,j) = A(\mathbf{f}_i, \mathbf{f}_j) = e^{\left\{\frac{-1}{2\sigma^2} S(\mathbf{f}_i, \mathbf{f}_j)\right\}}$$



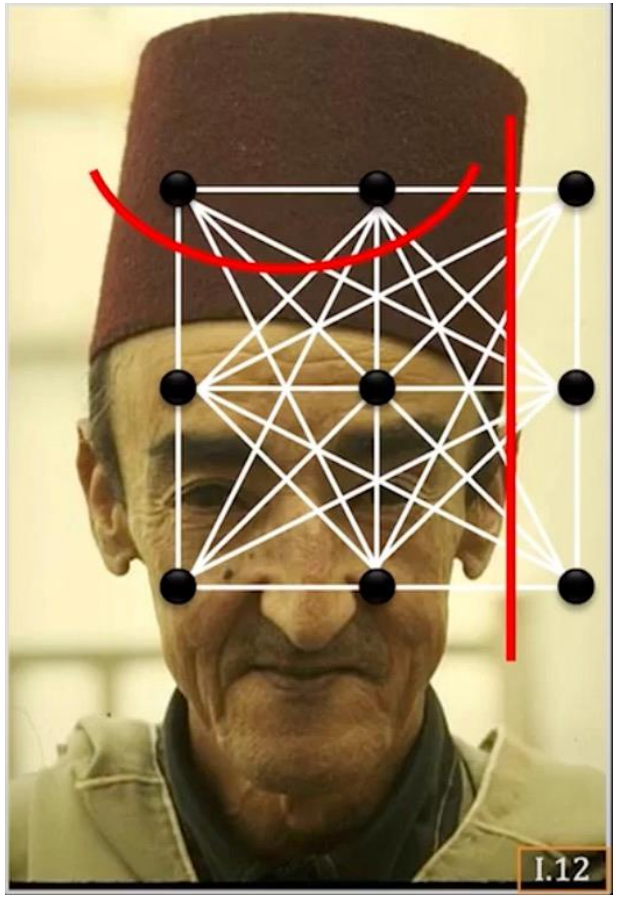
Graph-cut segmentation

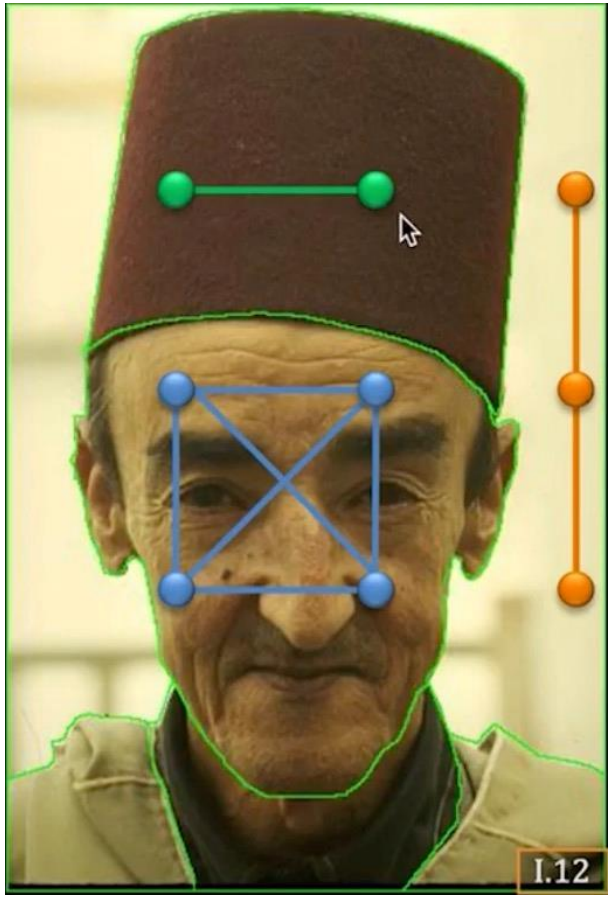
- Cut C = {V_A, V_B} is a partition of vertices V of a graph G = {V, E} into two disjoint subsets V_A, and V_B.
- Cut-set: set of edges whose vertices are in different subsets of partition
- Cost of Cut: Sum of weights of cut-set edges.



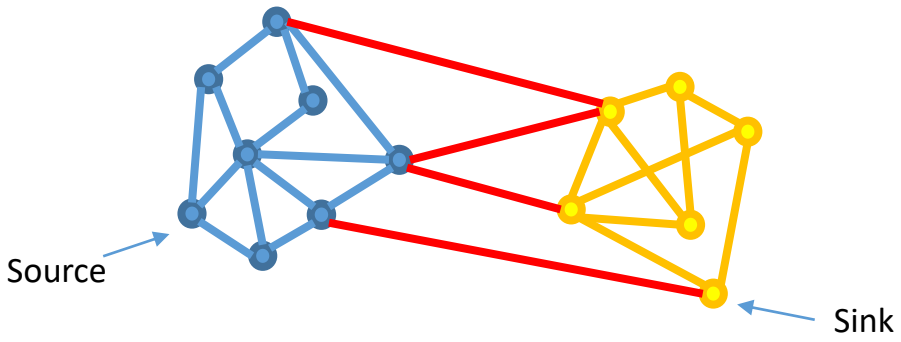
$$cut(V_A, V_B) = \sum_{u \in V_A, v \in V_B} w(u, v)$$

Graph-cut segmentation





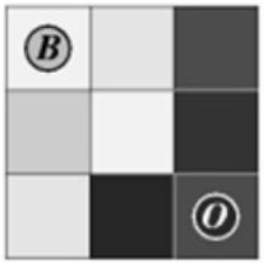
Graph Cut and Flow



- 1) Given a source (s) and a sink node (t)
- 2) Define Capacity on each edge, $C_{ij} = W_{ij}$
- 3) Find the maximum flow from s->t, satisfying the capacity constraints:

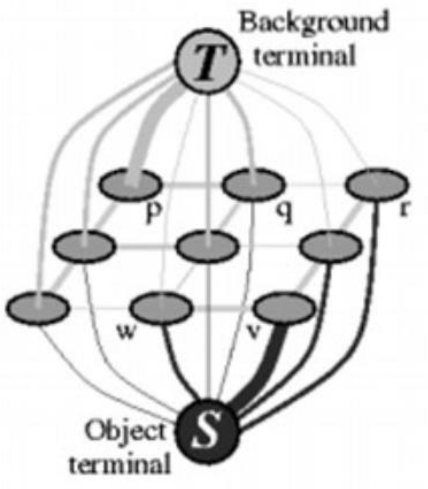
$$\boldsymbol{Min.Cut = Max.}\ \textbf{Flow}$$

Min Cut

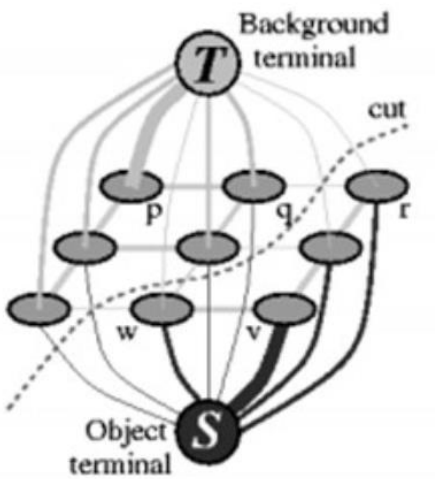


(a) Image with seeds.

(d) Segmentation results.



(b) Graph.



(c) Cut.

Min Cut

- The minimum cut: i.e., C that minimizes

$$C = \sum_{(i,j)\in C} w_{ij}$$

- N-links: between adjacent pixels, we could use

$$w_{ij} = e^{-\frac{\|I_i - I_j\|^2}{\delta^2}}$$

- T-links: let user "scribble" on image to denote some initial foreground and background pixels, which forms probability distributions F_B, F_F.

$$w_{iF} = -F_F(I_i); w_{iB} = F_B(I_i)$$

$$argmin\ aR(L) + E(L)$$