

## Today's Topics

Today: More on edge detection

- Edge detection using zero-crossings & image smoothing
- Scale-space techniques
- Case study: Intelligent Scissors

## The Image Laplacian

- What is the 2D mask corresponding to the image Laplacian?

$$\text{1D case} \\ \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$$

$$\text{Laplacian} \\ \nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial c^2}$$

2nd derivative along a row	+	2nd derivative along a column	=	Laplacian mask (sum of the 2 derivatives)																											
<table border="1" style="border-collapse: collapse;"><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>-2</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	1	-2	1	0	0	0		<table border="1" style="border-collapse: collapse;"><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>-2</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr></table>	0	1	0	0	-2	0	0	1	0		<table border="1" style="border-collapse: collapse;"><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>-4</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr></table>	0	1	0	1	-4	1	0	1	0
0	0	0																													
1	-2	1																													
0	0	0																													
0	1	0																													
0	-2	0																													
0	1	0																													
0	1	0																													
1	-4	1																													
0	1	0																													

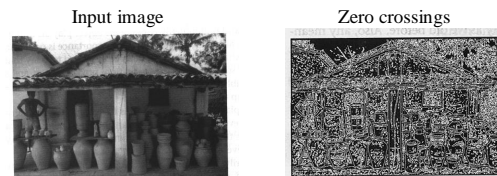
## Edge Detection in 2D Using Laplacian

1. Apply the Laplacian mask to the input 2D image
2. Find all zero-crossings in the resulting edge-enhanced image

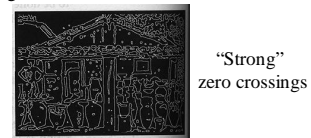
5	10	0
-2	-2	1
5	5	10

- In general, pixels in the Laplacian image may not be exactly zero
- A pixel is labelled as zero crossings if one of its 8 neighbors has opposite sign

## Zero-crossing Detection Example

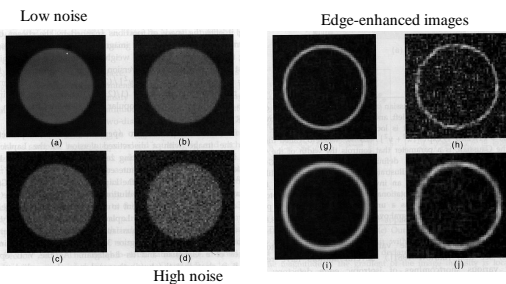


Q: How can we eliminate most of the noise-induced clutter in the image?



## Dealing with Noisy Images

Ideally, we want to remove noise before doing edge enhancement/detection



## Steps of General Approach

- Given input image I, compute a new, “smoothed” image I'



- Apply edge enhancement/edge detection to the smoothed image

Compute zero crossings of the Laplacian of I'

### Noise Removal (a.k.a Image Smoothing)

How can we smooth noisy images?

- Simplest idea: Replace pixel with the average of its neighbors
- Averaging mask:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Degree of smoothing can be controlled by choosing masks of different sizes

### Noise Removal (a.k.a Image Smoothing)

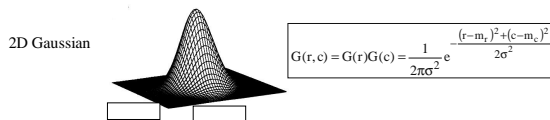
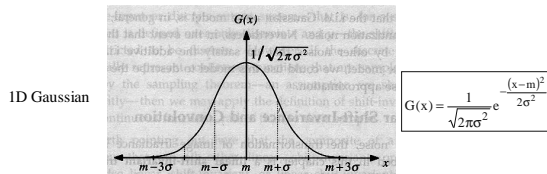
The averaging mask gives same weight to every point in a pixel's neighborhood

Smoothing operation can also be defined by assigning larger weights to pixels closer to center pixel

In 1D, one such smoothing operation is given by the Gaussian function (usually  $m$  assumed to be 0):

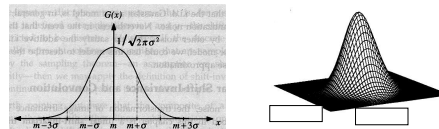
$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-m)^2}{2\sigma^2}}$$

### Gaussians in 1D and 2D



### Noise Removal Using Gaussians

Degree of smoothing controlled by parameter  $\sigma^2$  (variance)



How can we create a mask for Gaussian smoothing?

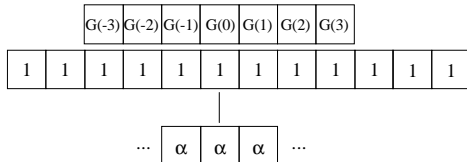
1. Sample the Gaussian at regular intervals on a square grid
2. Choose mask size so that most of the Gaussian's "mass" included in mask (usually sufficient to choose masks whose elements span the interval  $[-3\sigma, 3\sigma]$ )

G(-3)	G(-2)	G(-1)	G(0)	G(1)	G(2)	G(3)
-------	-------	-------	------	------	------	------

(mask for  $\sigma=1$ )

### Normalized vs. Unnormalized Masks

What happens if we apply to an image a smoothing mask whose elements sum to a value  $\alpha \neq 1$ ?



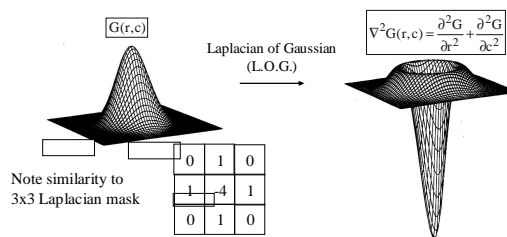
Ans: The resulting image will appear brighter ( $\alpha > 1$ ) or darker ( $\alpha < 1$ ) than the original

Normalized Gaussian mask ( $\sigma = 1$ ):

$$\begin{matrix} G(-3) & G(-2) & G(-1) & G(0) & G(1) & G(2) & G(3) \end{matrix} * \frac{1}{\alpha}$$

### Edge Detection: Laplacian of the Gaussian

Idea: Rather than smooth image first using Gaussian & then compute the Laplacian of result, derive a mask that does both operations simultaneously



Note similarity to 3x3 Laplacian mask

0	1	0
1	-4	1
0	1	0

L.O.G. mask computed by sampling L.O.G. function on pixel grid (mask size chosen to be identical to that of the corresponding Gaussian mask)

## Laplacian of Gaussian

Closed form expression  $LGG(x, y) = -\frac{1}{\pi\sigma^3} \left[ \delta - \frac{x^2 + y^2}{2\sigma^2} \right] e^{-\frac{x^2 + y^2}{2\sigma^2}}$

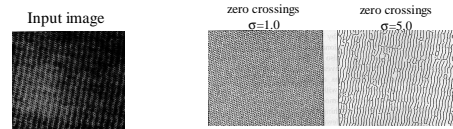
Discrete approximation to LoG function with Gaussian 1.4  
 $3*(1.4)=4.2$

0	0	0	0	0	0	0	0	0	0
0	2	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

## Marr-Hildreth Edge Detection Algorithm

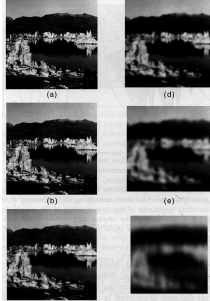
Given: An input image I & a value for variance  $\sigma^2$   
 Steps:

1. Compute L.O.G. mask corresponding to value of  $\sigma^2$
2. Apply L.O.G. mask to image I
3. Compute zero crossing pixels of result
4. Label as edges all zero crossing pixels
5. [Optional step] label as edges only those zero-crossings whose transition from + to - or from - to + is greater than a threshold



## One Remaining Question...

How do we choose the right value for  $\sigma$  ?



$\sigma$  is generally referred to as the scale parameter

## Today's Topics

More on edge detection

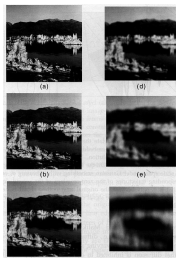
- Edge detection using zero-crossings & image smoothing
- Scale-space techniques
- Case study: Intelligent Scissors

## Choosing the "Right" Scale??

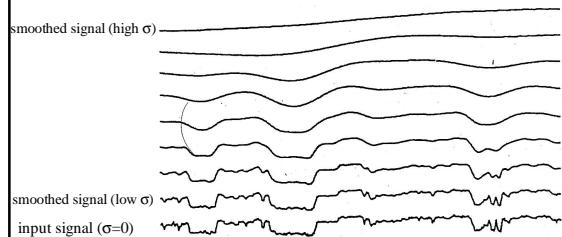
Rather than trying to decide which scale to use, we can process images at many scales & examine how the smoothed image changes with increasing values of  $\sigma$

Basic principles of scale-space approaches

- No single scale is "special"
- Treat  $\sigma$  as a continuous parameter
- As  $\sigma$  increases, we should obtain "simpler" images, i.e., no "new structures" should be created
- "Structures" that remain present for a broad range of scales signify "important" features in the image

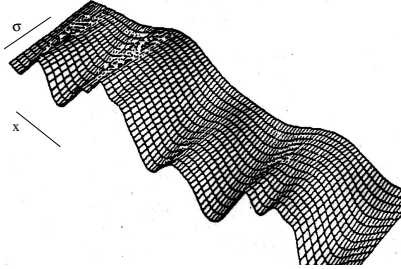


## Scale Space in 1D



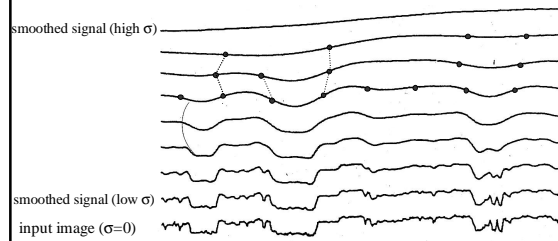
## Scale Space in 1D

Resulting Scale-Space Surface



## Zero Crossings in 1D

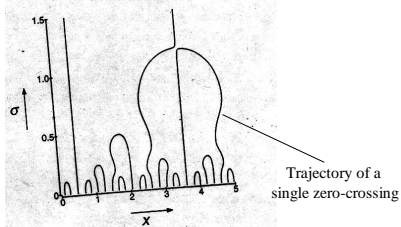
Zero crossings of 2nd image derivative = inflections of the intensity function



Idea: As the 1D image gets increasingly smoothed, the position of zero crossings moves along the smoothed curve, creating trajectory curves

## Scale Space in 1D

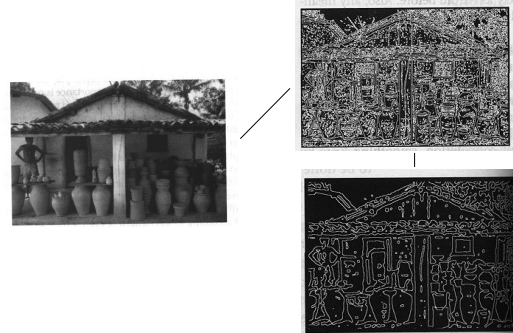
Idea: As the 1D image gets increasingly smoothed, the position of zero crossings moves along the smoothed curve, creating trajectory curves



- For the Gaussian function, trajectories form nested, upside-down U's
- The Gaussian smoothing function is the ONLY function that ensures no new zero-crossings created as  $\sigma$  increases
- Therefore, it is the only function for which zero crossing trajectories have the nesting property!!

## Scale Space in 2D

- Zero crossing contours form surfaces in  $(r,c)-\sigma$  space
- No new zero-crossings created as  $\sigma$  increases



## Today's Topics

More on edge detection

- Edge detection using zero-crossings & image smoothing
- Scale-space techniques
- Case study: Intelligent Scissors

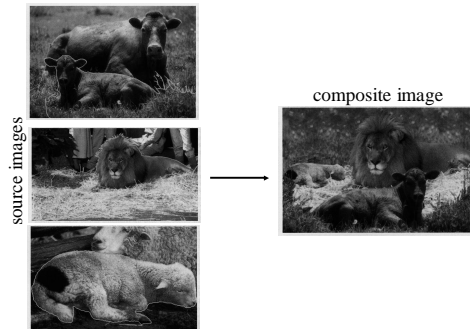
## Image Scissoring: Motivation #1

How easy would it be for an image analysis system to automatically find the person's outline (or that of a vase in the picture) from the output of an edge detector?



## Image Scissoring: Motivation #2

By scissoring portions of one or more images & pasting them together we can create new, composite images



## Image Scissoring: Requirements

- Interactive operation
- "User is always right"  
Scissoring system must allow user to select arbitrary regions
- Scissoring operations must be performed efficiently
- Scissoring interface must be simple & easy to use



## Image Scissoring

- Brute-force approach  
User chooses every single image pixel that defines the region boundary
- Here: Intelligent Scissors approach ("livewire") developed by Eric Mortensen & presented at SIGGRAPH'95

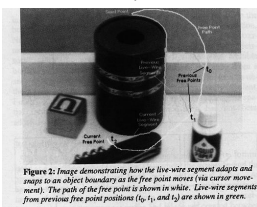


Figure 2: Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions ( $f_0$ ,  $f_1$ , and  $f_2$ ) are shown in green.

## Intelligent Scissors: Operation

- User loads image & specifies a "seed" point on boundary that must be outlined
- User then positions mouse close to object boundary
- System automatically creates a "live-wire" that connects seed & current mouse position & follows boundary as much as possible
- Live wire updated whenever mouse moves

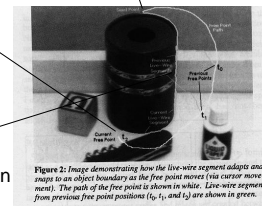


Figure 2: Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions ( $f_0$ ,  $f_1$ , and  $f_2$ ) are shown in green.

## Intelligent Scissors: Basic Idea

Approach answers one basic question:

- How should we define a path from seed to mouse that follows an object boundary as closely as possible?

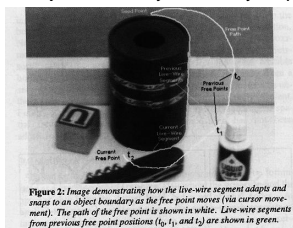


Figure 3: Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions ( $f_0$ ,  $f_1$ , and  $f_2$ ) are shown in green.

- Answer: Define a path that is as close as possible to image edges

## Intelligent Scissors: Basic Idea

Approach taken in intelligent scissors attempts to exploit user interaction while avoiding the need to detect edges corresponding to object boundaries very accurately

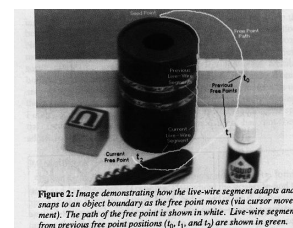
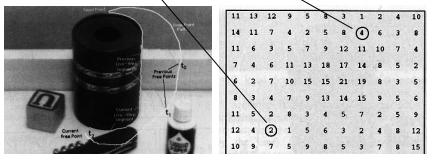


Figure 3: Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions ( $f_0$ ,  $f_1$ , and  $f_2$ ) are shown in green.

### Intelligent Scissors: Basic Idea

Solution:

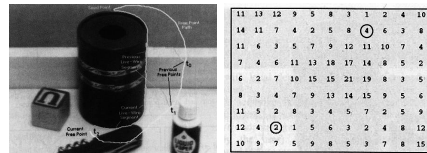
- Assign "edge-ness" weight to every pair of adjacent pixels (computed using edge-enhancement operation)
- Weights defined so that edges have very low weights
- To connect seed & mouse positions, choose the path that minimizes the total weight of pixels along the path



### Intelligent Scissors: Basic Idea

Two questions must be answered to fully-specify the algorithm:

- How do we assign weights to pixel pairs?
- How do we find the lowest-cost path between any two image pixels?



### Intelligent Scissors: Weight Assignment

Path: a sequence of adjacent pixels in the image

Link: a pair of adjacent pixels along the path



Given a link defined by pixels p & q, its weight is defined to be

$$l(p, q) = 0.43f_z(q) + 0.43f_D(p, q) + 0.14f_G(q)$$

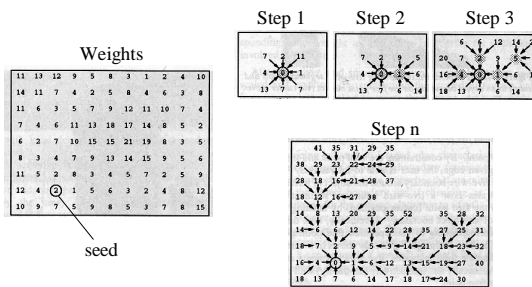
$f_z(q) = 0$  if Laplacian=0 at q  
 $f_z(q) = 1$  otherwise

term that penalizes links where the gradient orientation at p and q is very different

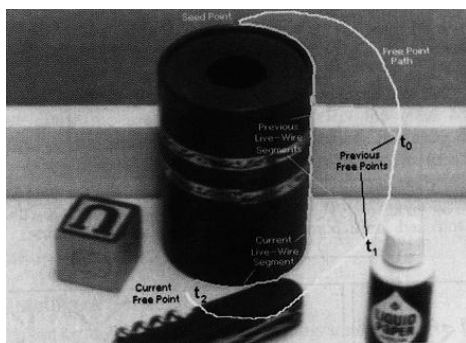
$f_G(q) = 1 - \frac{|\nabla I|}{\max(|\nabla I|)}$   
 largest possible value in image

### Intelligent Scissors: Path Optimizer

Path optimization formulated as a graph search algorithm that computes the minimum-cost, n-link path from seed to all other image pixels (use Floyd's algorithm)



### Intelligent Scissors: Results



### Intelligent Scissors: Results



### Intelligent Scissors: Results

By scissoring portions of one or more images & pasting them together we can create new, composite images

