CS 4495 Computer Vision

Linear Filtering 2: Templates, Edges

Aaron Bobick

School of Interactive Computing

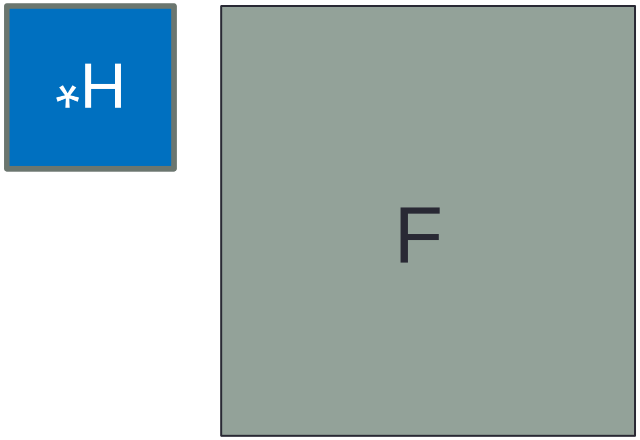
Last time: Convolution

- Convolution:
  - Flip the filter in both dimensions (right to left, bottom to top)
  - Then apply cross-correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

$$G = H * F$$

Notation for convolution operator



K. Grauman

Convolution vs. correlation

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i-u, j-v]$$

$$G = H \star F$$

(Cross-)correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i+u, j+v]$$

$$G = H \otimes F$$

- When H is symmetric, no difference. We tend to use the terms interchangeably.
- Convolution with an impulse (centered at 0,0) is the identity

K. Grauman

Filters for features

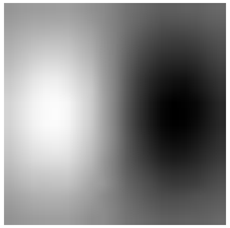
- Previously, thinking of filtering as a way to remove or reduce **noise**
- Now, consider how filters will allow us to abstract higher-level "**features**".
    - Map raw pixels to an intermediate representation that will be used for subsequent processing
    - Goal: reduce amount of data, discard redundancy, preserve what's useful

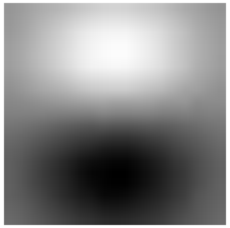
K. Grauman

Template matching

- Filters as **templates**:

  Note that filters look like the effects they are intended to find --- “matched filters”

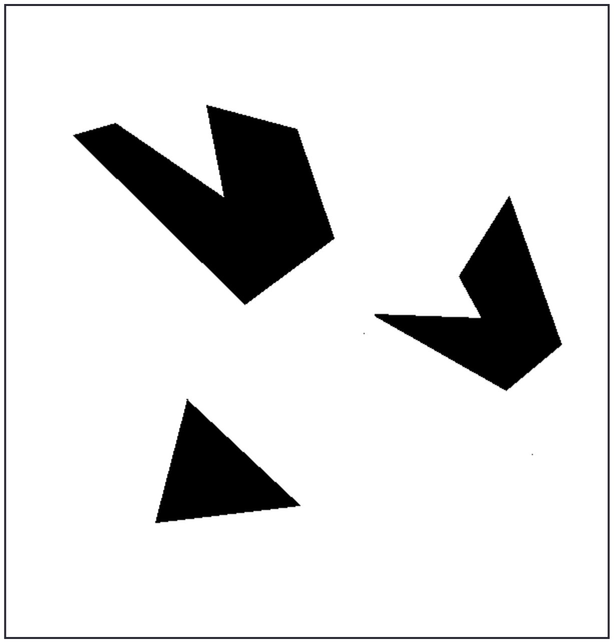




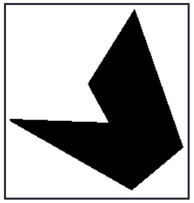
- Use (*normalized*) cross-correlation score to find a given pattern (template) in the image.
  - Normalization needed to control for relative brightness. More in problem sets.

K. Grauman

Template matching



Scene

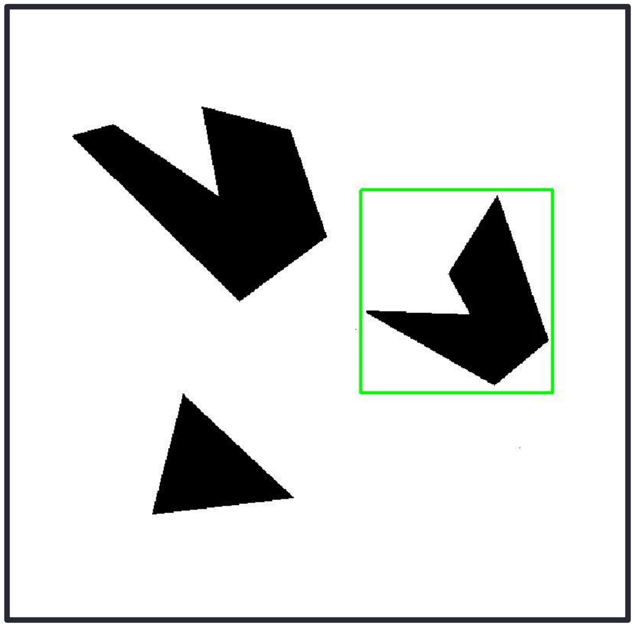


Template (mask)

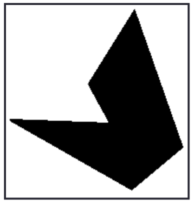
A toy example

K. Grauman

Template matching



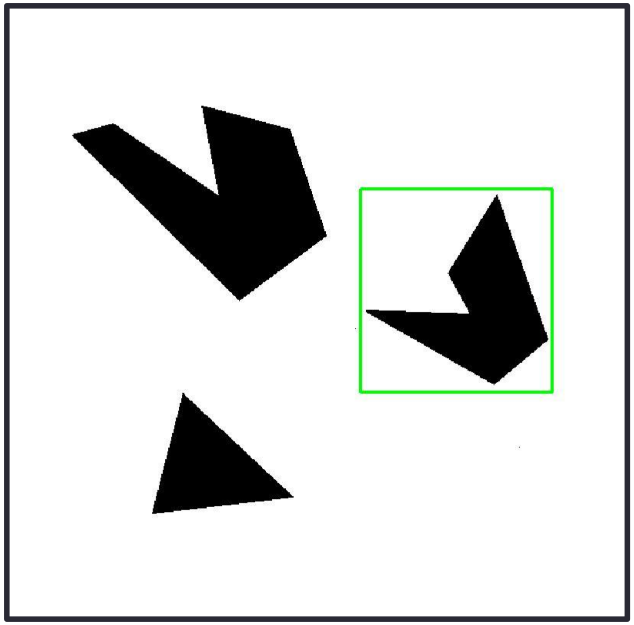
Detected template



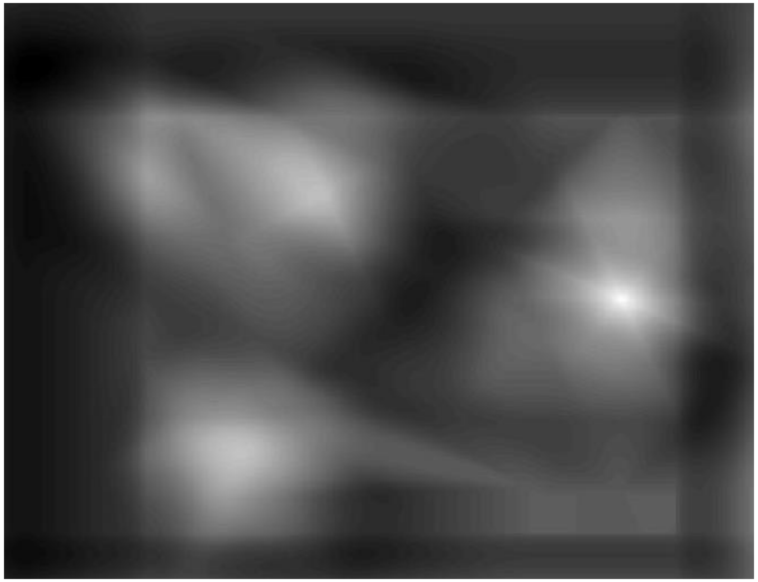
Template

K. Grauman

Template matching



Detected template

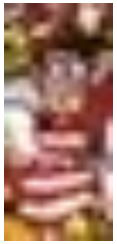


Correlation map

K. Grauman

Where's Waldo?

Scene

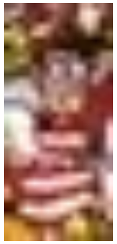


Template

K. Grauman

Where's Waldo?

Detected template



Template

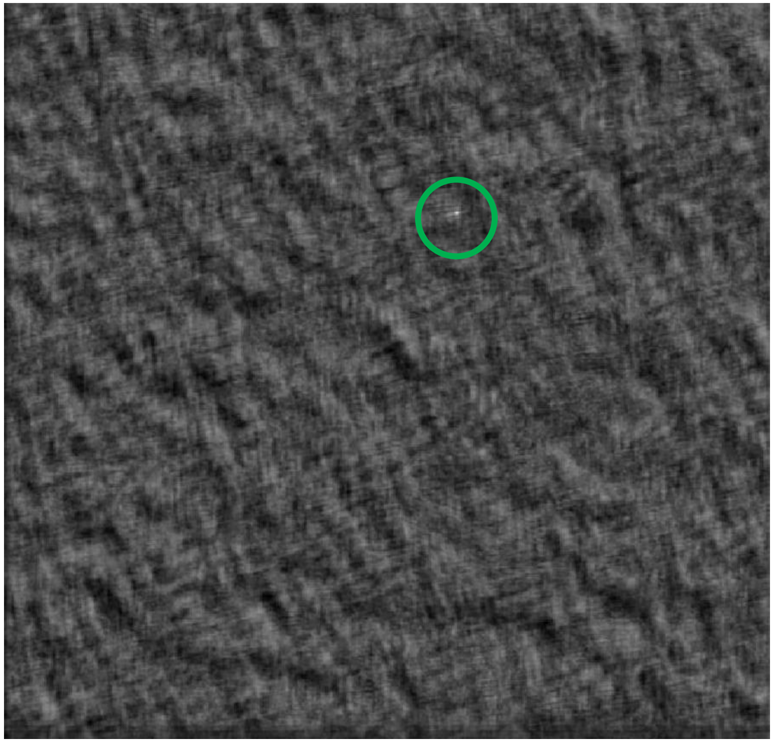
K. Grauman

Template demo…

- In directory C:\Bobick\matlab\CS4495\Filter
- echodemo waldotemplate

Where's Waldo?

Detected template



Correlation map

K. Grauman

Template matching

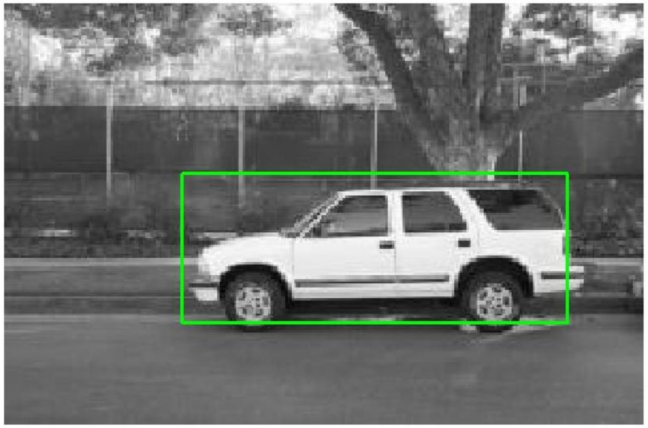
Scene

Template

What if the template is not identical to some subimage in the scene?

K. Grauman

Template matching



Detected template

Template

Match can be meaningful, if scale, orientation, and general appearance is right.

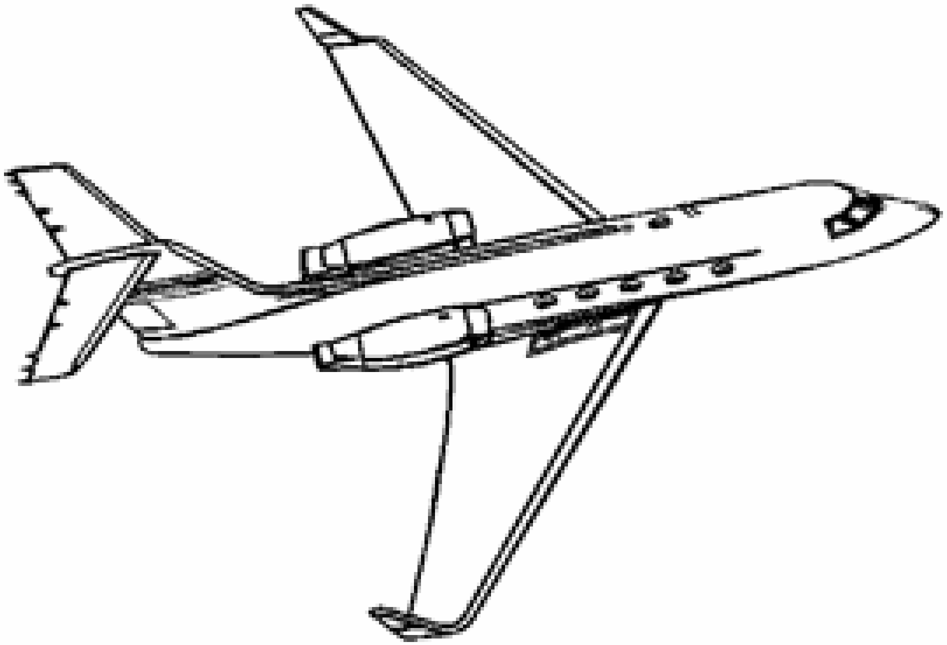
K. Grauman

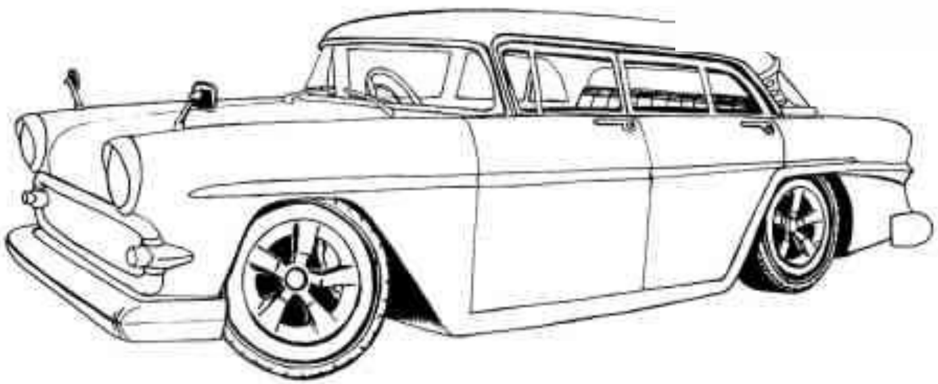
Generic features…

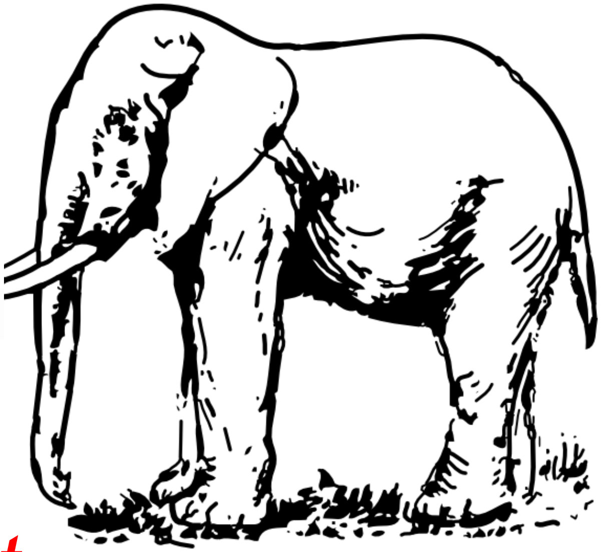
- When looking for a specific object or pattern, the **features** can be defined for that pattern – we will do this later in the course for specific object recognition.

- But for **generic** images, what would be good features? What are the parts or properties of the image that encode its “meaning” for human (or other biological) observers?

- Some examples of greatly reduced images…

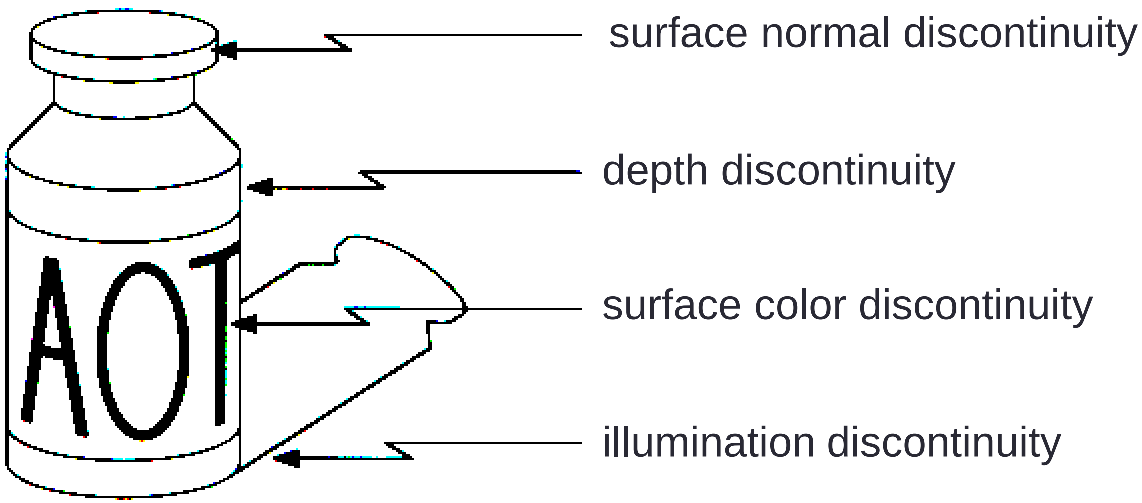






Edges seem to be important…

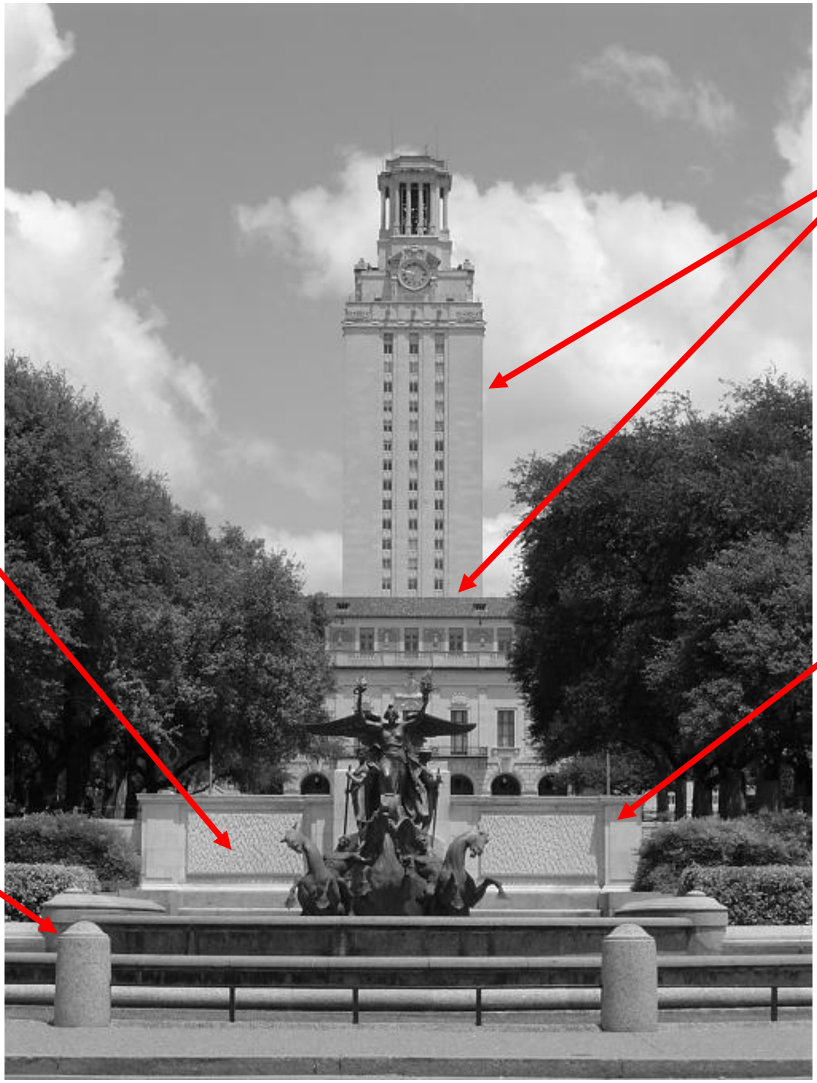
Origin of Edges



- Edges are caused by a variety of factors
- Information theory view: edges encode change, change is what is hard to predict, therefore edges efficiently encode an image

In a real image

Reflectance change: appearance information, texture

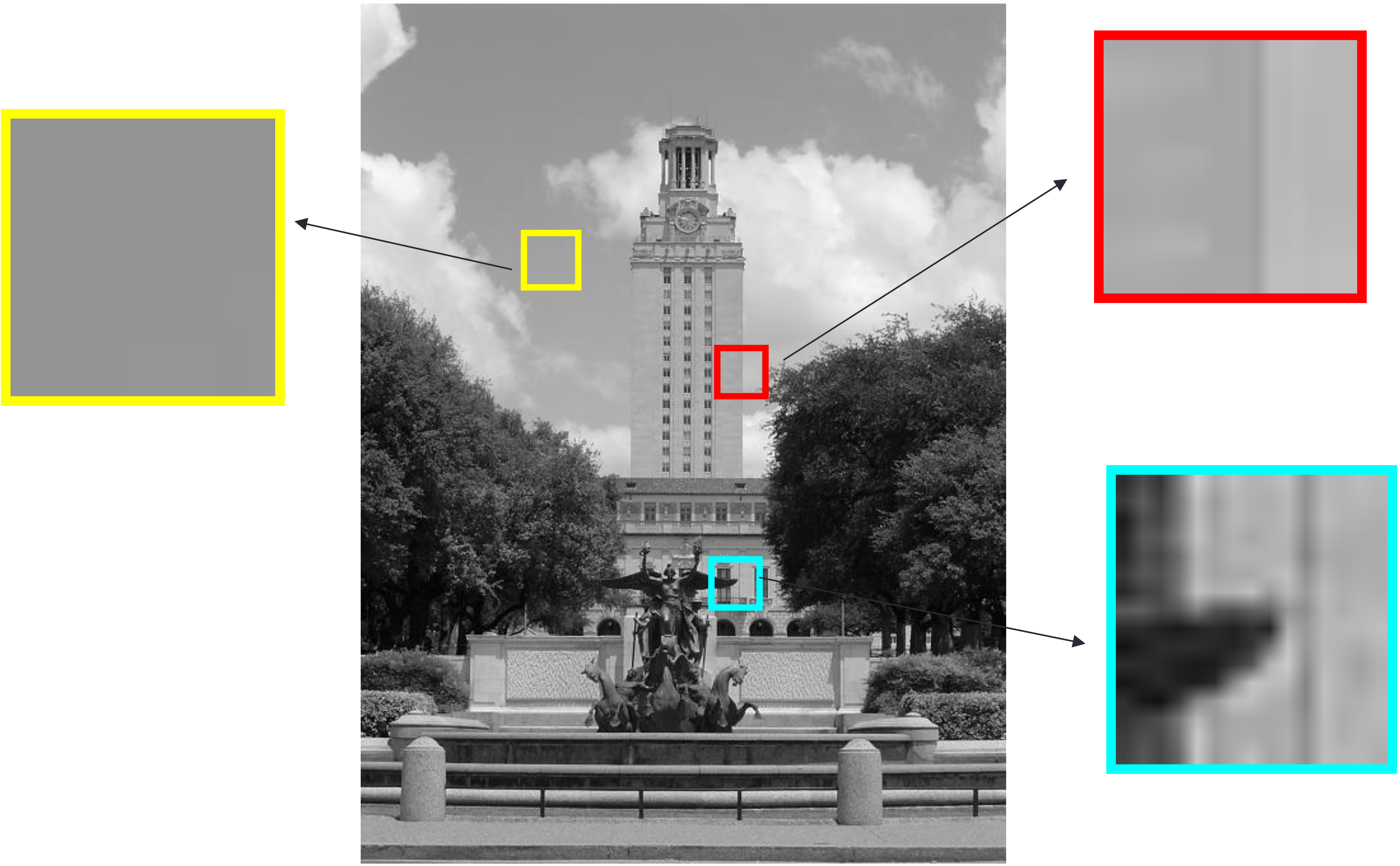


Depth discontinuity: object boundary

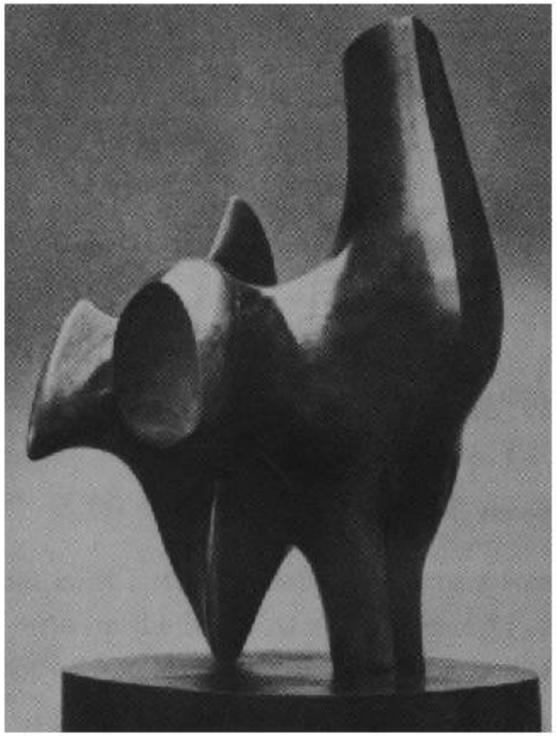
Cast shadows

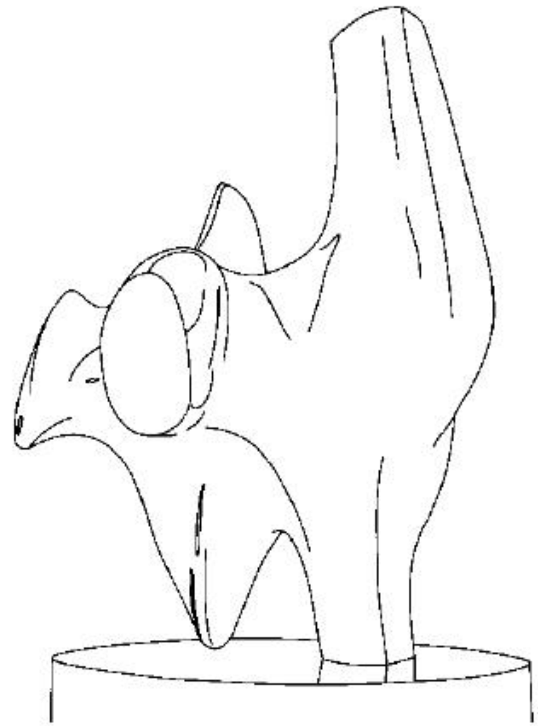
Change in surface orientation: shape

Contrast and invariance



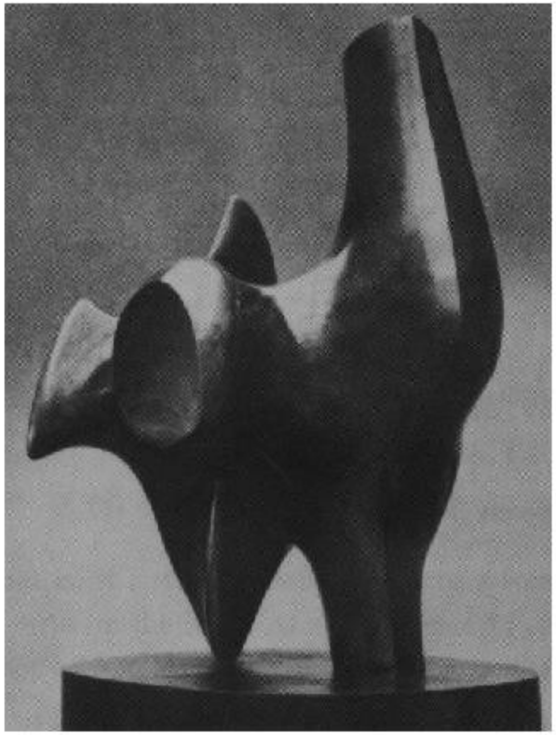
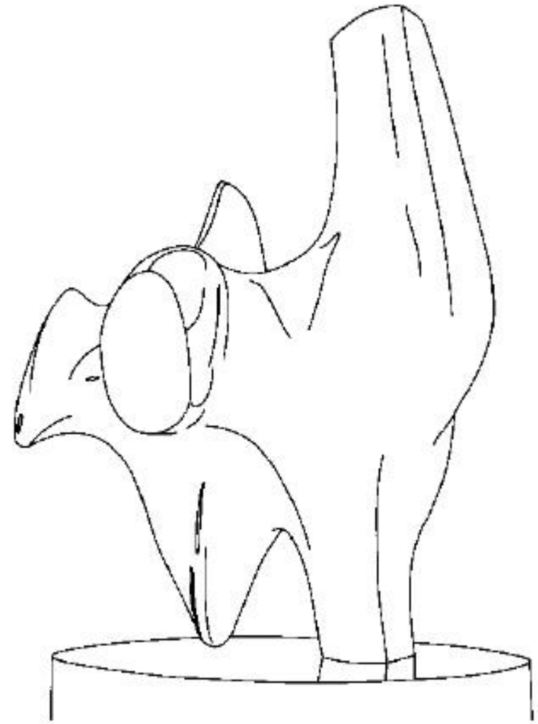
Edge detection





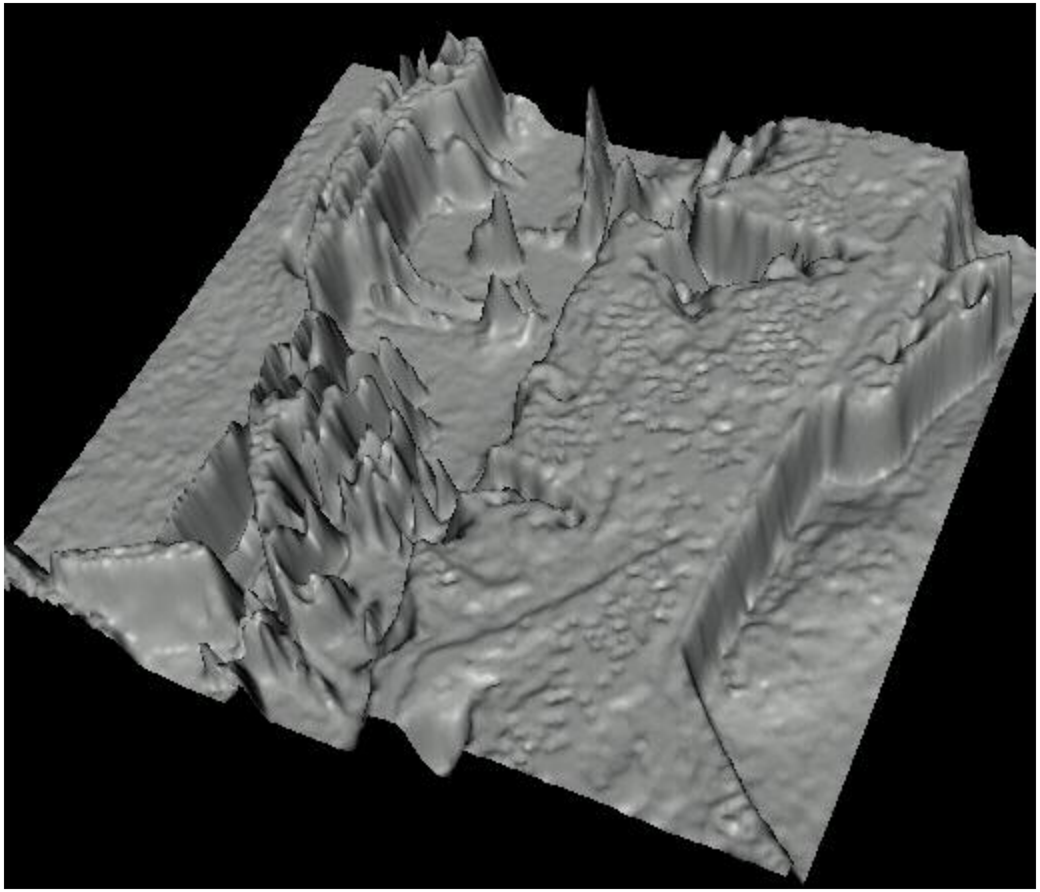
- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels

Edge detection

- How can you tell that a pixel is on an edge?

Images as functions…



- Edges look like steep cliffs

Edge Detection

Basic idea: look for a neighborhood with strong signs of change.

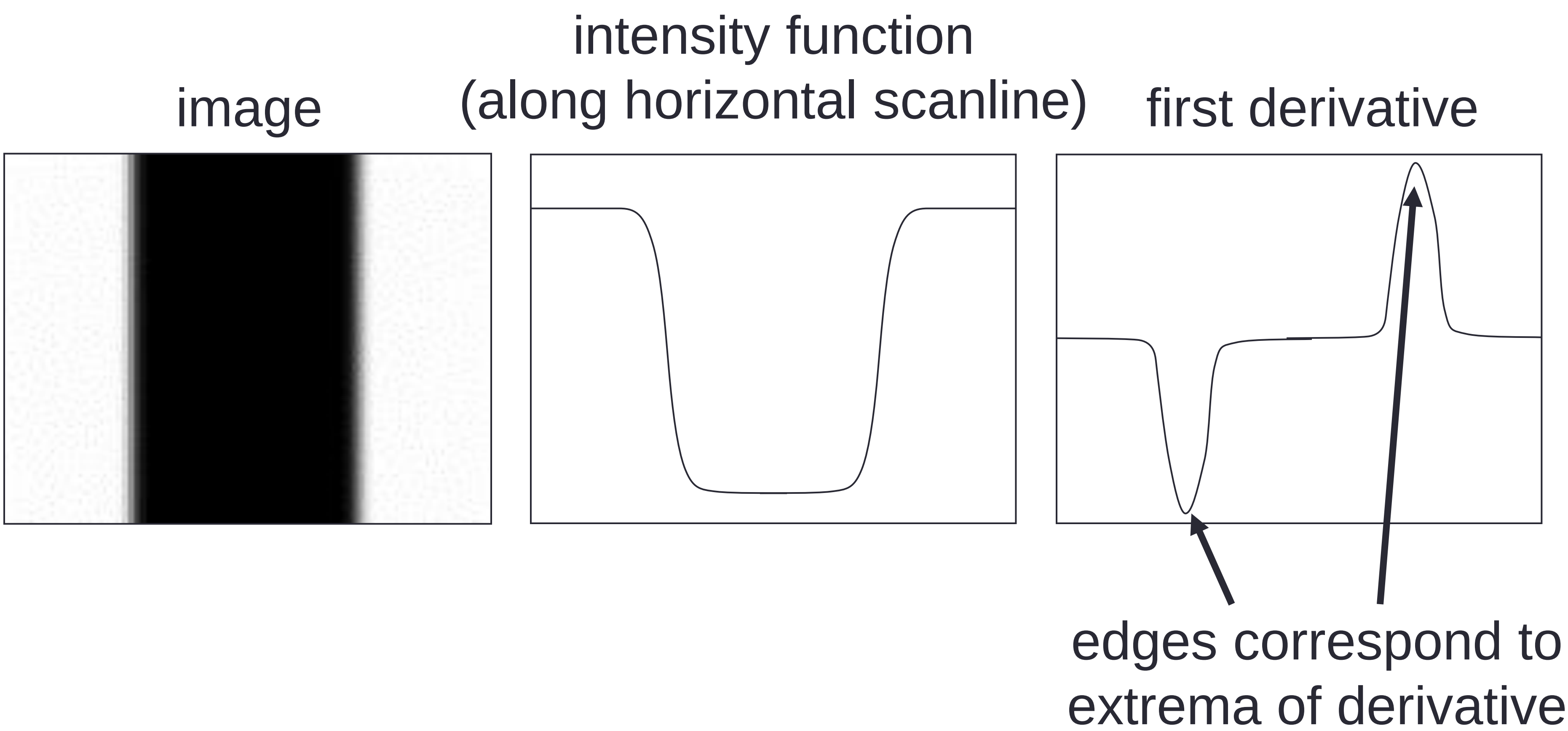
Problems:

- neighborhood size
- how to detect change

81	82	26	24
82	33	25	25
81	82	26	24

Derivatives and edges

An edge is a place of rapid change in the image intensity function.



Source: L. Lazebnik

Differential Operators

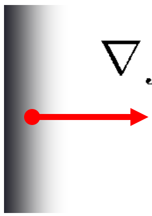
- Differential operators – here we mean some operation that when applied to the image returns some derivatives.
- We will model these “operators” as masks/kernels which when applied to the image yields a new function that is the image ***gradient function***.
- We will then threshold the this ***gradient function*** to select the edge pixels.
- Which brings us to the question:

What’s a gradient?

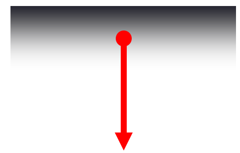
Image gradient

The gradient of an image:

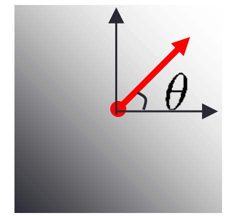
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient points in the direction of most rapid increase in intensity

The gradient direction is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- how does this relate to the direction of the edge?

The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Discrete gradient

For 2D function, f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

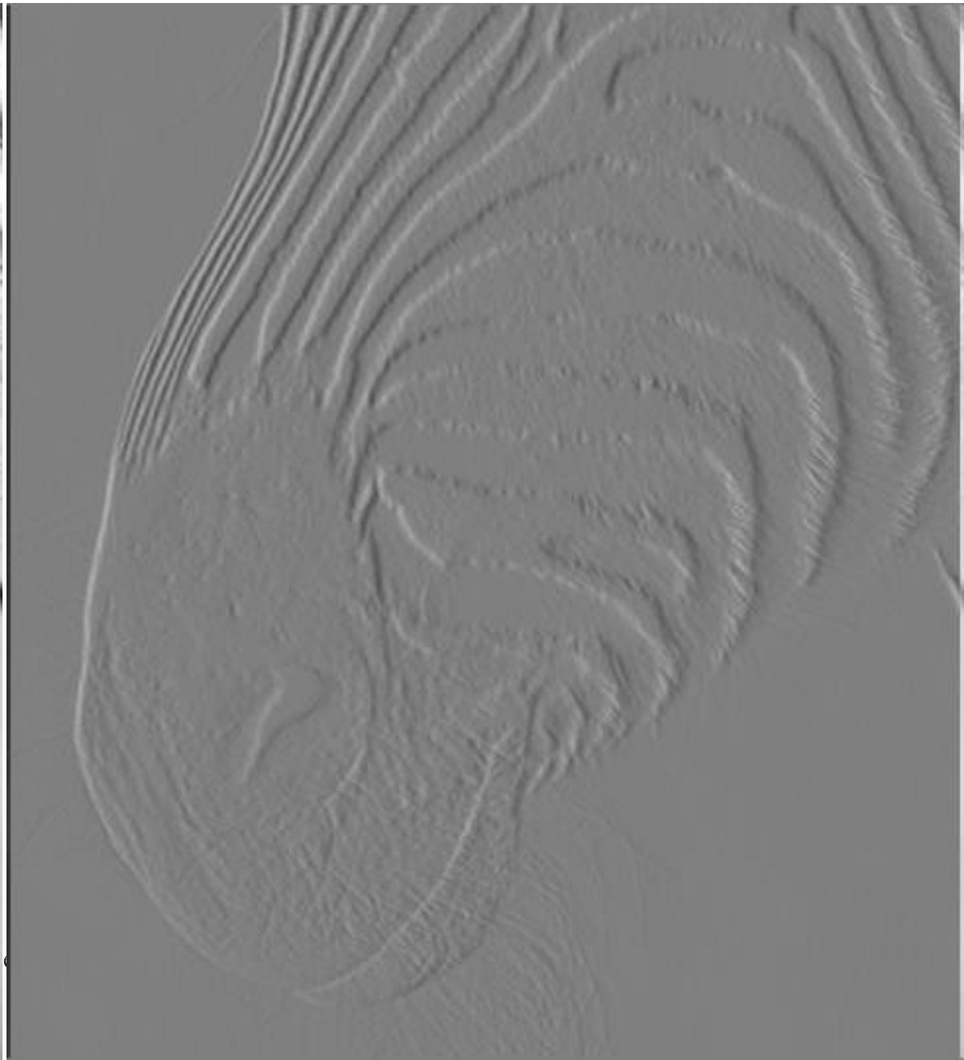
For discrete data, we can approximate using *finite* differences:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

$$\approx f(x+1, y) - f(x,y)$$

"right derivative"
But is it???

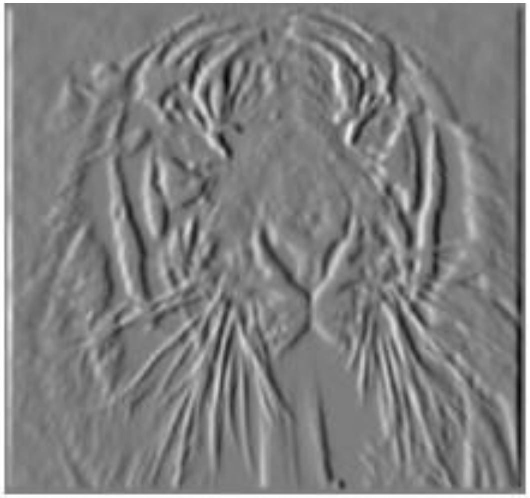
Finite differences

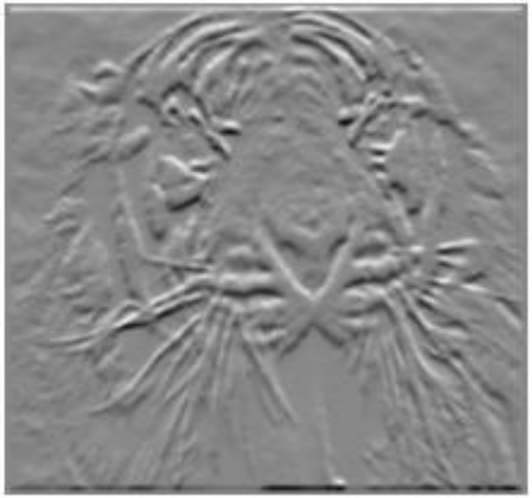


Partial derivatives of an image

$$\frac{\partial f(x,y)}{\partial x}$$

$$\frac{\partial f(x,y)}{\partial y}$$





-1	1

-1
1

? or

1
-1

Which shows changes with respect to x?

(showing correlation filters)

Differentiation and convolution

- For 2D function, f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

- For discrete data, we can approximate using finite differences:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

- To implement above as convolution, what would be the associated filter?

The discrete gradient

- We want an "operator" (mask/kernel) that we can apply to the image that implements:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

How would you implement this as a cross-correlation?

(not flipped)

0	0
-1	+1
0	0

H

Not symmetric around image point; which is "middle" pixel?

0	0	0
-1/2	0	+1/2
0	0	0

H

Average of "left" and "right" derivative. See?

Example: Sobel operator

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$s_x \qquad\qquad s_y$$

On a pixel of the image I

- Let g_x be the response to mask S_x (sometimes * 1/8)
- Let g_y be the response to mask S_y

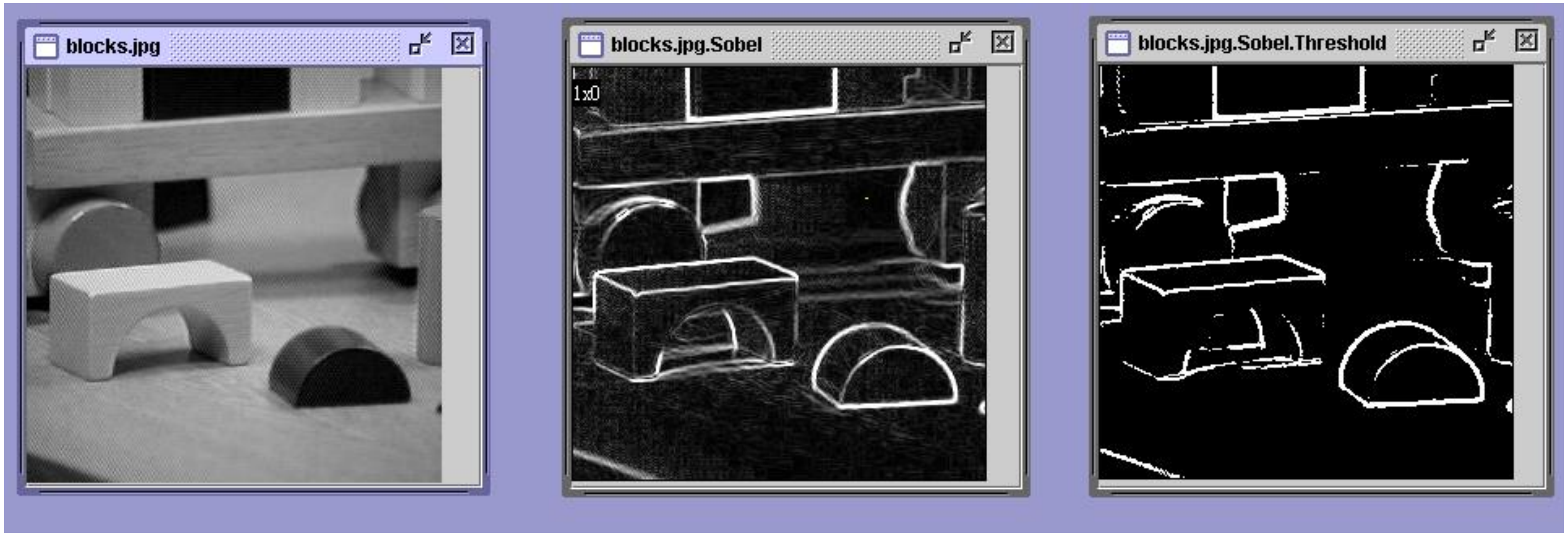
What is the gradient?

(Sobel) Gradient is $\nabla I = [g_x \;\; g_y]^T$

$g = (g_x^2 + g_y^2)^{1/2}$ is the gradient magnitude.

$\theta = \text{atan2}(g_y, g_x)$ is the gradient direction.

Sobel Operator on Blocks Image



original image

gradient magnitude

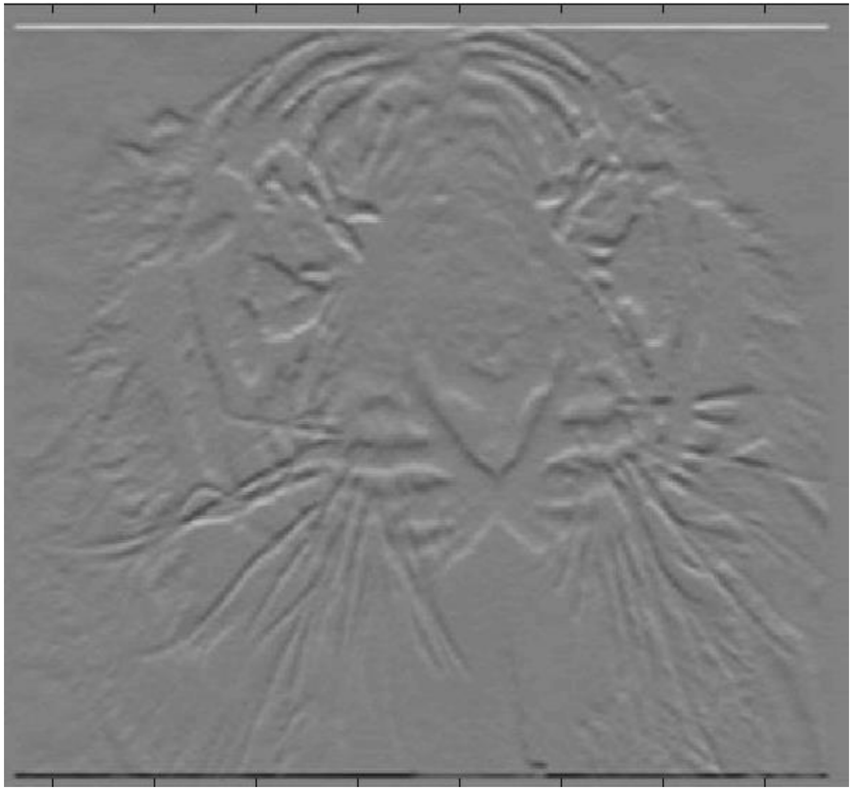
thresholded gradient magnitude

Some Well-Known Masks for Computing Gradients

	Sx	Sy
• Sobel:	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$
• Prewitt:	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$
• Roberts	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Matlab does edges

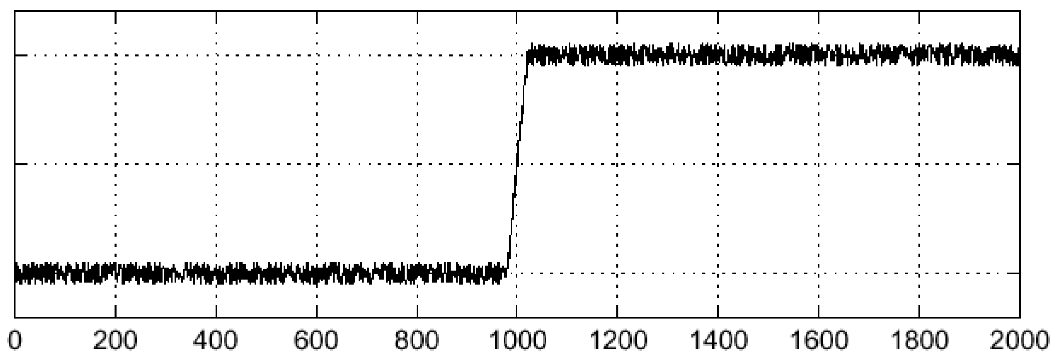
```
>> My = fspecial('sobel');
>> outim = imfilter(double(im), My);
>> imagesc(outim);
>> colormap gray;
```



But…

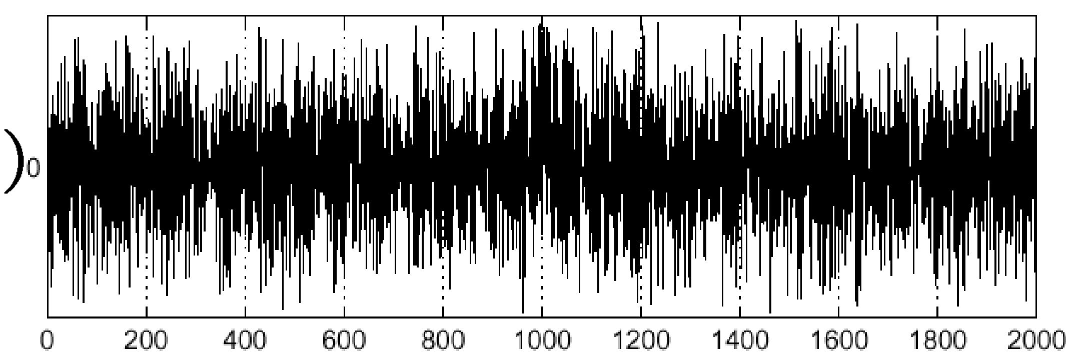
- Consider a single row or column of the image
  - Plotting intensity as a function of *x*

$$f(x)$$



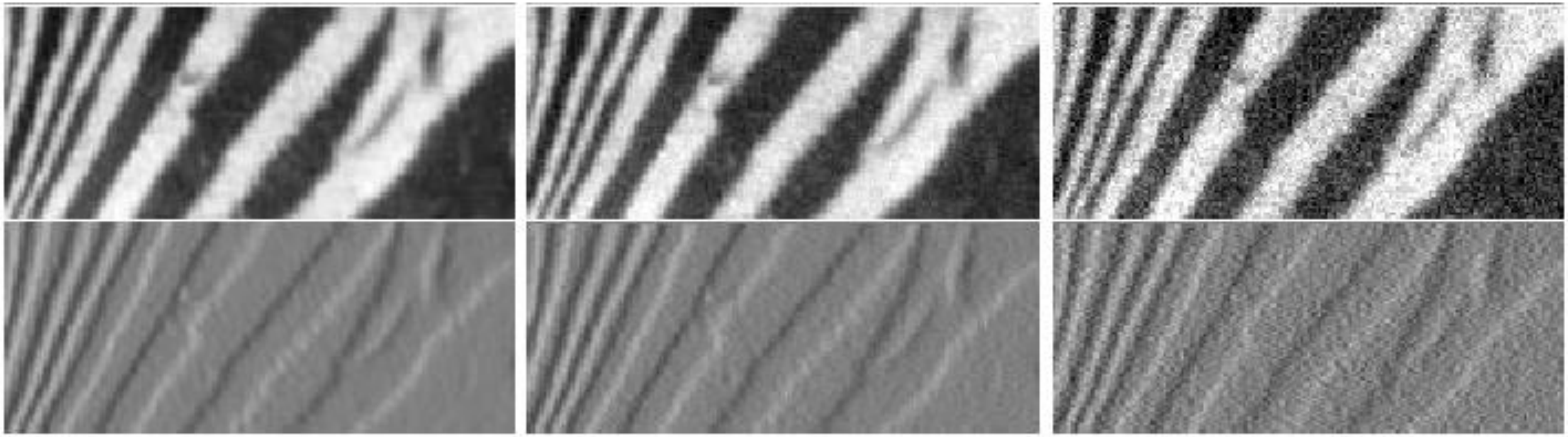
  - Apply derivative operator….

$$\frac{d}{dx}f(x)$$



Uh, where's the edge?

Finite differences responding to noise



Increasing noise ->
(this is zero mean additive gaussian noise)

D. Forsyth

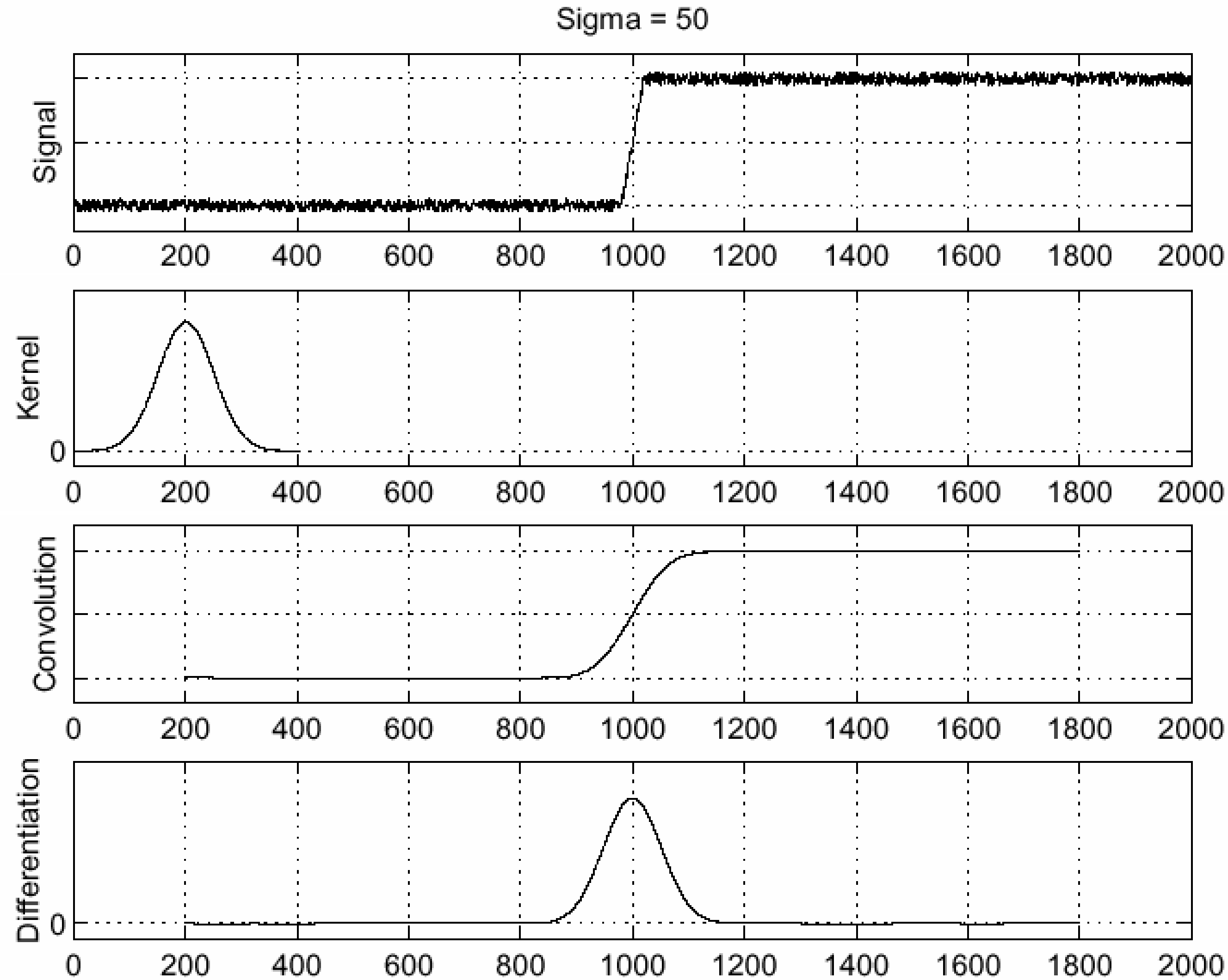
Solution: smooth first

f

h

$h * f$

$\frac{\partial}{\partial x}(h * f)$



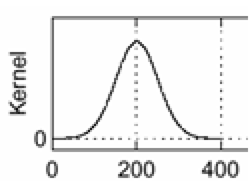
Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h * f)$

Derivative theorem of convolution

- This saves us one operation: $\frac{\partial}{\partial x}(h * f) = (\frac{\partial}{\partial x}h) * f$

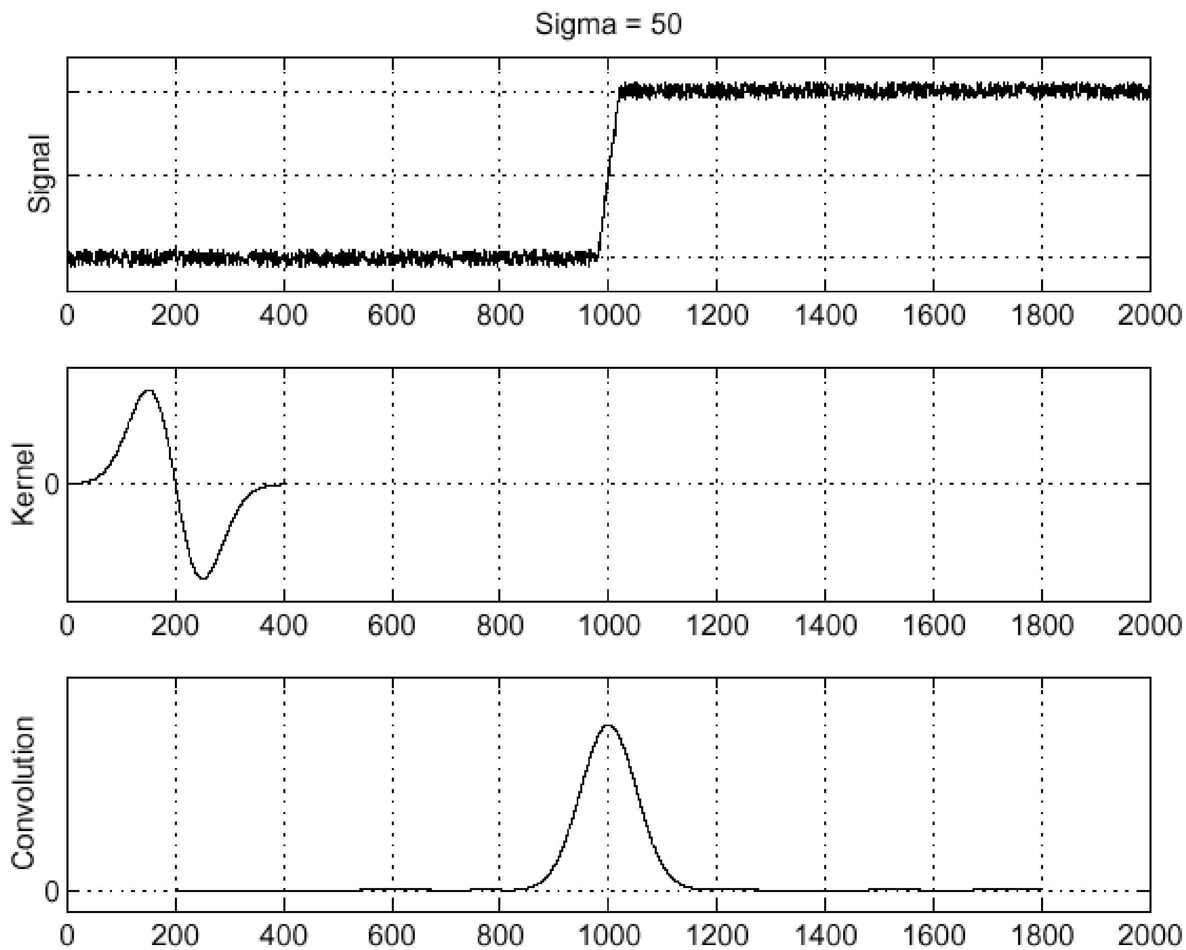
f

h



$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) * f$



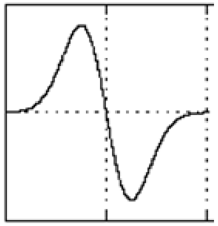
How can we find (local) maxima of a function?

2nd derivative of Gaussian

- Consider $\frac{\partial^2}{\partial x^2}(h * f)$

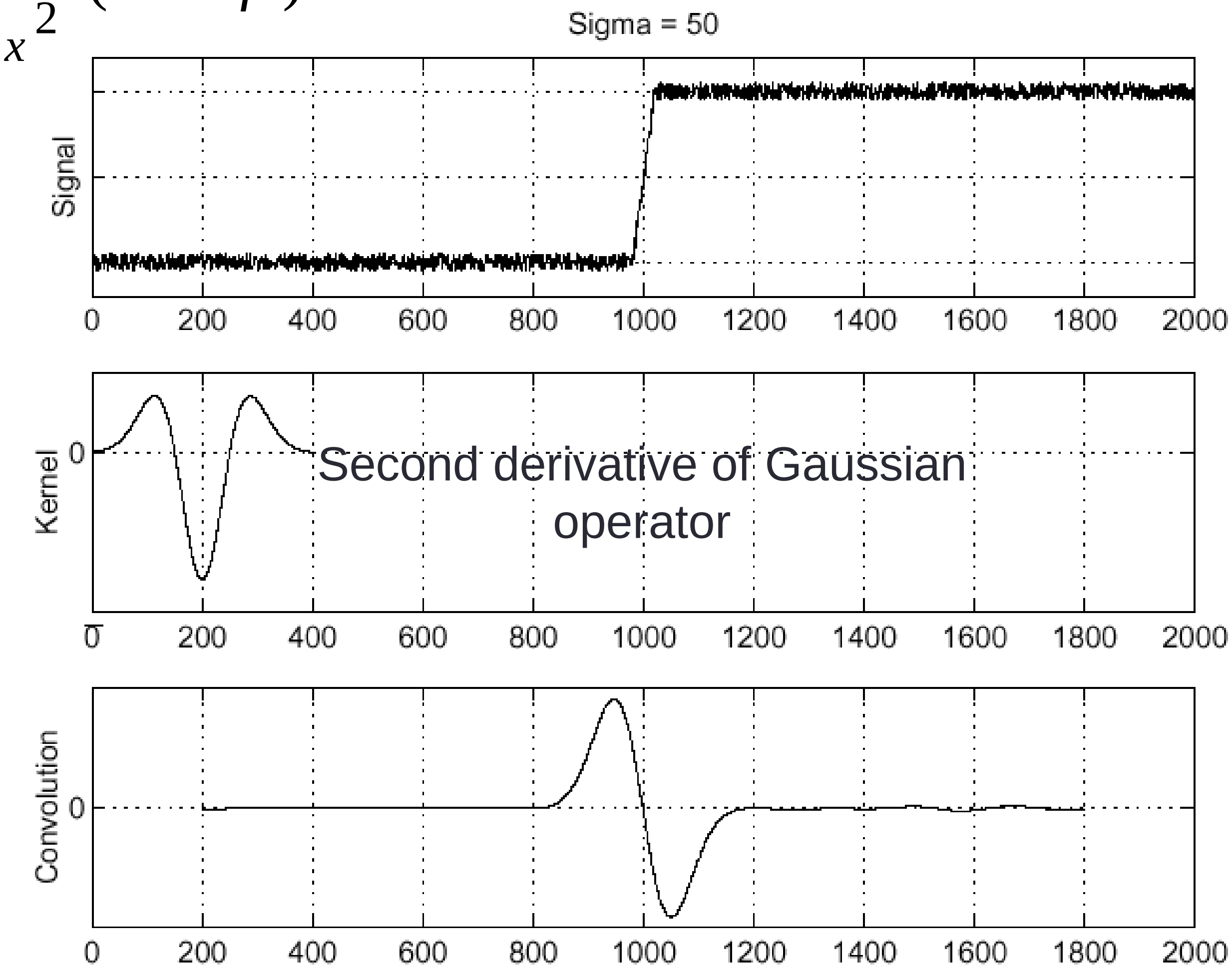
f

$\frac{\partial}{\partial x} h$



$\frac{\partial^2}{\partial x^2} h$

$\frac{\partial^2}{\partial x^2}(h * f)$



Where is the edge? Zero-crossings of bottom graph

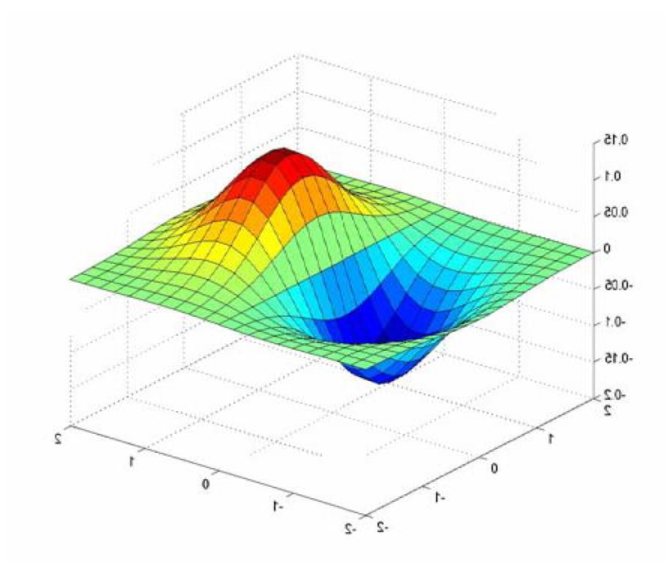
What about 2D?

Derivative of Gaussian filter

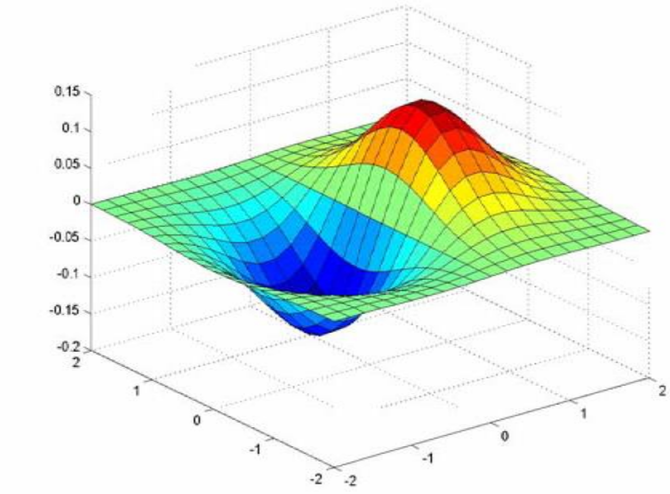
$$(I \otimes g) \otimes h = I \otimes (g \otimes h)$$

$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \end{bmatrix}$$

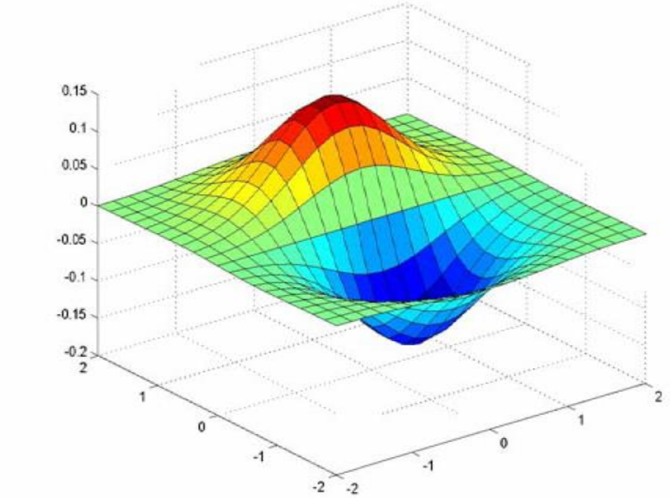
Why is this preferable?



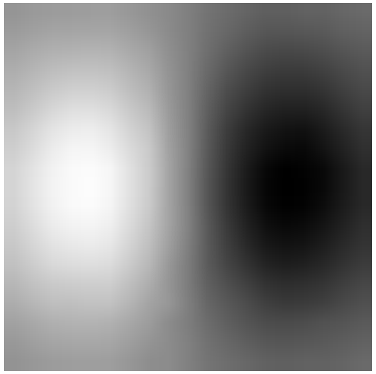
Derivative of Gaussian filters



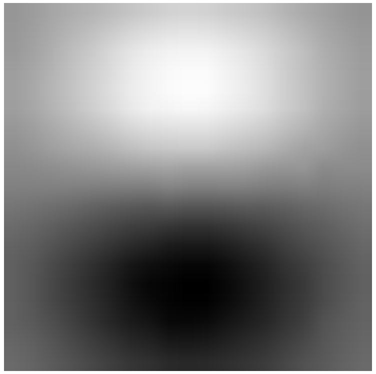
x-direction



y-direction



Is this for correlation or convolution?

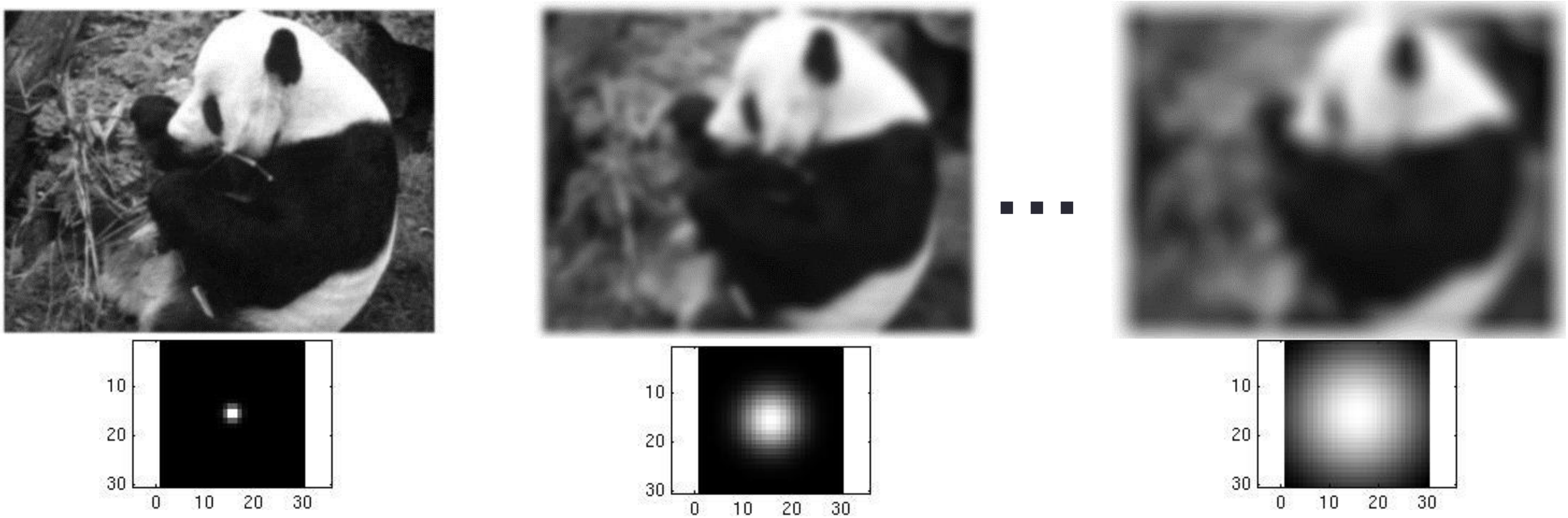


And for y it's always a problem!

Source: L. Lazebnik

Smoothing with a Gaussian

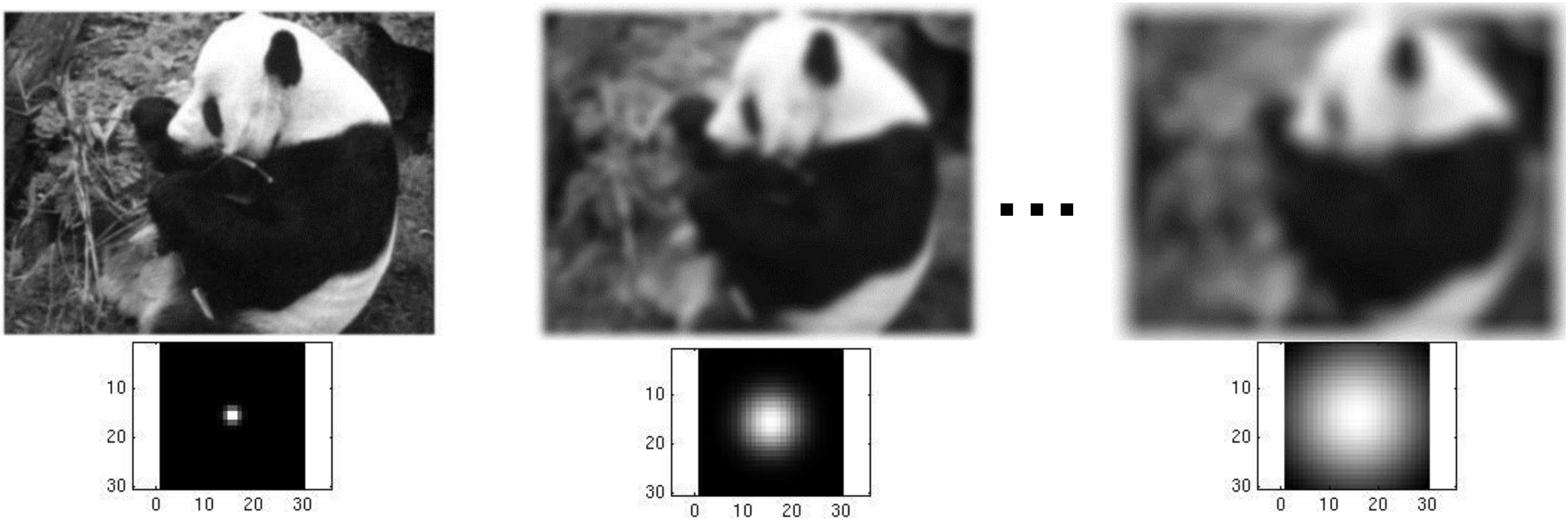
Parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.



```
for sigma=1:3:10
  h = fspecial('gaussian`, fsize, sigma);
  out = imfilter(im, h);
  imshow(out);
  pause;
end
```

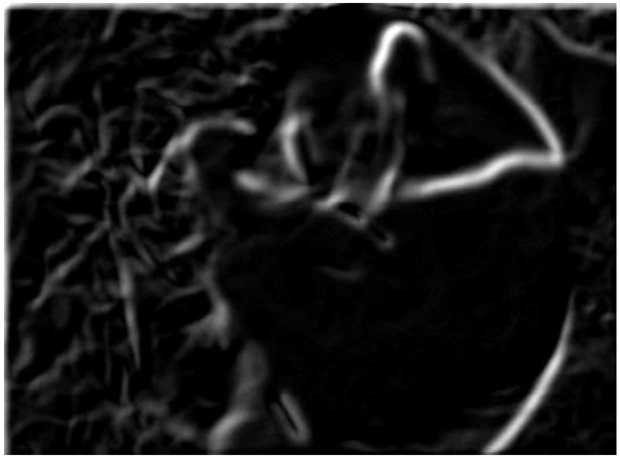
Smoothing with a Gaussian

Recall: parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.



Effect of σ on derivatives

σ = 1 pixel



σ = 3 pixels

The apparent structures differ depending on Gaussian's scale parameter.

Larger values: larger scale edges detected
Smaller values: finer features detected

Gradients -> edges

- Primary edge detection steps:
- 1. Smoothing: suppress noise
- 2. Edge “enhancement”: filter for contrast
- 3. Edge localization
  - Determine which local maxima from filter output are actually edges vs. noise
    - Threshold, Thin



Canny edge detector

- Filter image with derivative of Gaussian
- Find magnitude and orientation of gradient
- **Non-maximum suppression**:
  - Thin multi-pixel wide “ridges” down to single pixel width
- Linking and thresholding (**hysteresis**):
  - Define two thresholds: low and high
  - Use the high threshold to start edge curves and the low threshold to continue them

- MATLAB: **`edge(image, ‘canny’);`**
- **`>>help edge`**

Source: D. Lowe, L. Fei-Fei

The Canny edge detector

original image (Lena)

The Canny edge detector

magnitude of the gradient

The Canny edge detector

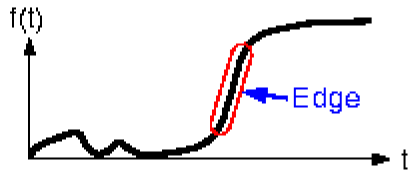
thresholding

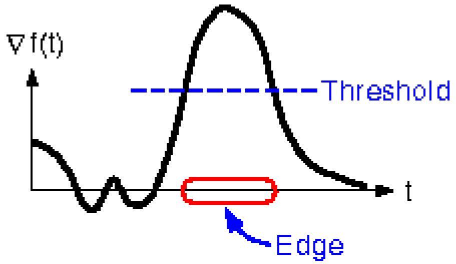
The Canny edge detector

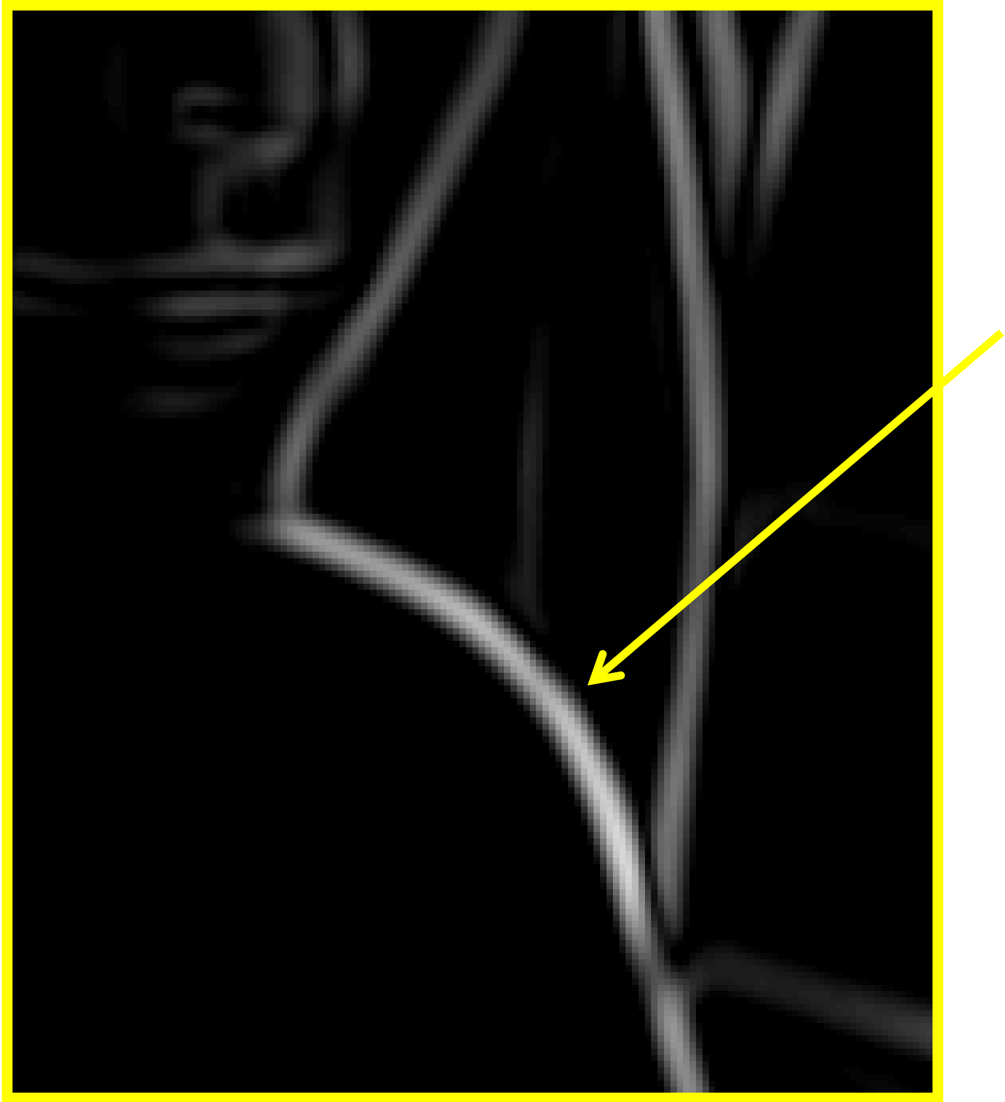
Problem: pixels along this edge didn't survive the thresholding

thinning
(non-maximum suppression)

The Canny edge detector

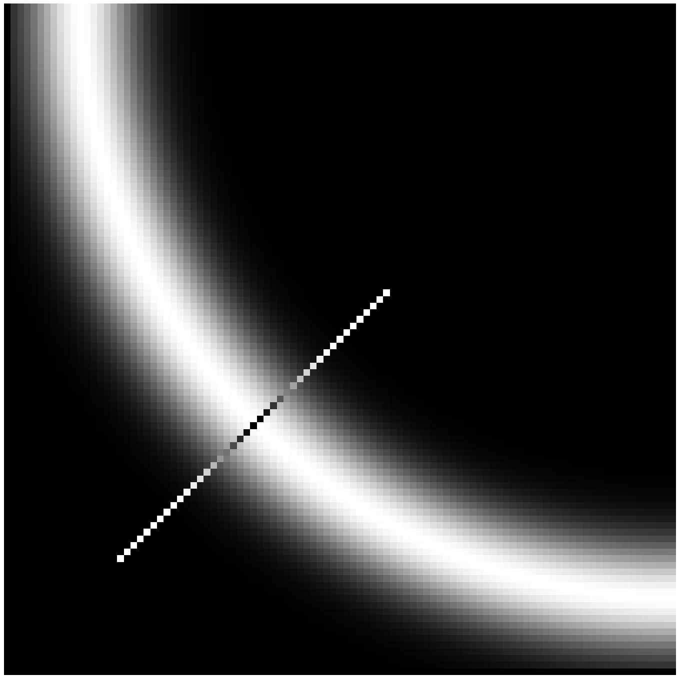


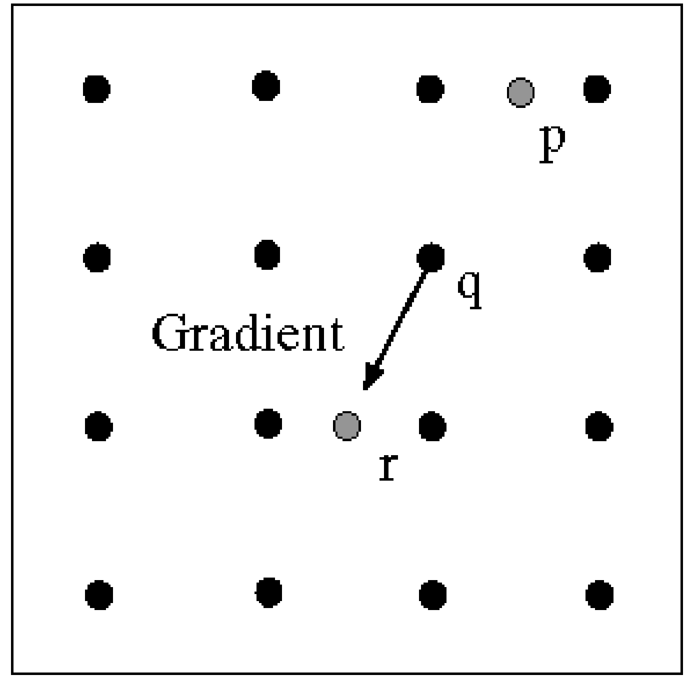




How to turn these thick regions of the gradient into curves?

Non-maximum suppression





- Check if pixel is local maximum along gradient direction
  - can require checking interpolated pixels p and r

The Canny edge detector

thinning

(non-maximum suppression)

Effect of σ (Gaussian kernel spread/size)

original　　Canny with $\sigma = 1$　　Canny with $\sigma = 2$

The choice of σ depends on desired behavior

- large σ detects large scale edges
- small σ detects fine features

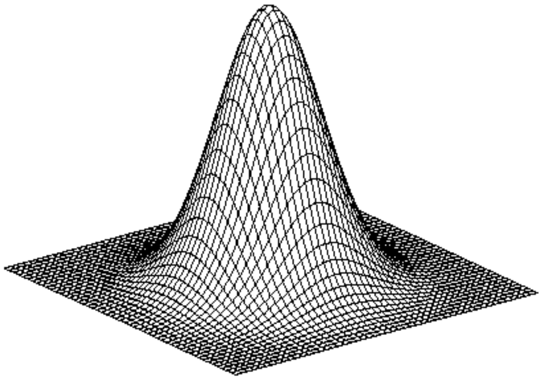
So, what scale to choose?

It depends what we're looking for.

Too fine of a scale…can't see the forest for the trees.

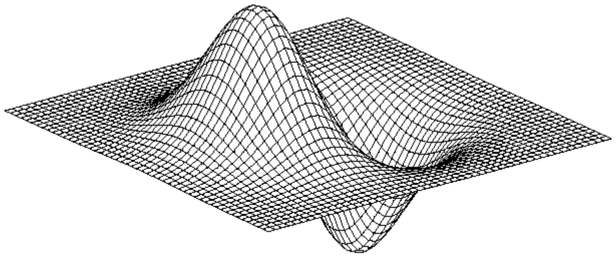
Too coarse of a scale…can't tell the maple grain from the cherry.

Single 2D edge detection filter



Gaussian

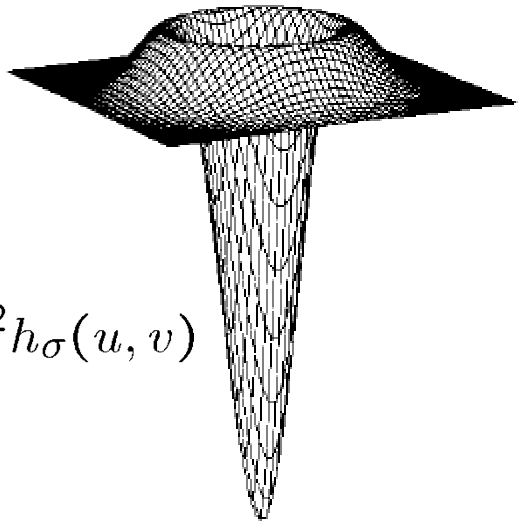
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian



$$\nabla^2 h_\sigma(u,v)$$

∇^2 is the **Laplacian** operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Edge demo in CS4495/Edges

Finish on Thurs…