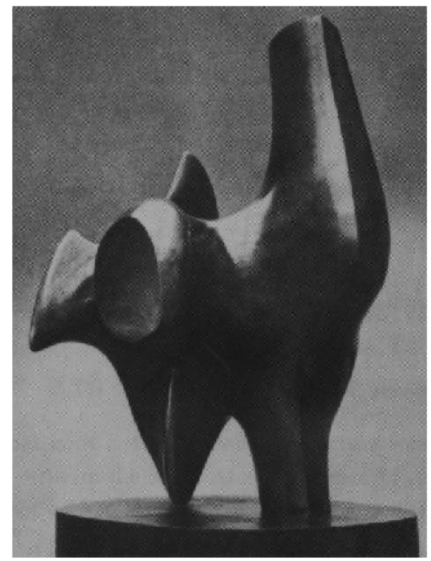
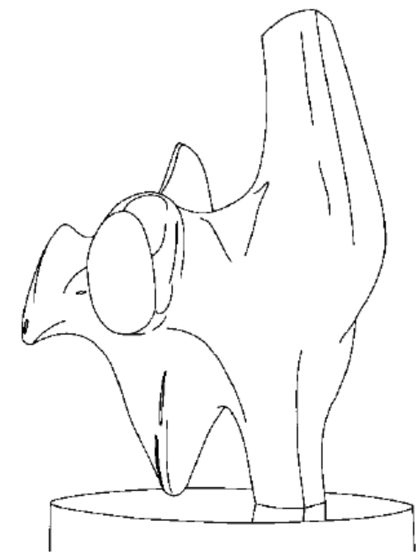
Edge Detection

Why extract edges?

- Edges and lines are used in
  - object recognition
  - image matching (e.g., stereo, mosaics)
  - document analysis
  - horizon detection
  - line following robots
  - and many more apps

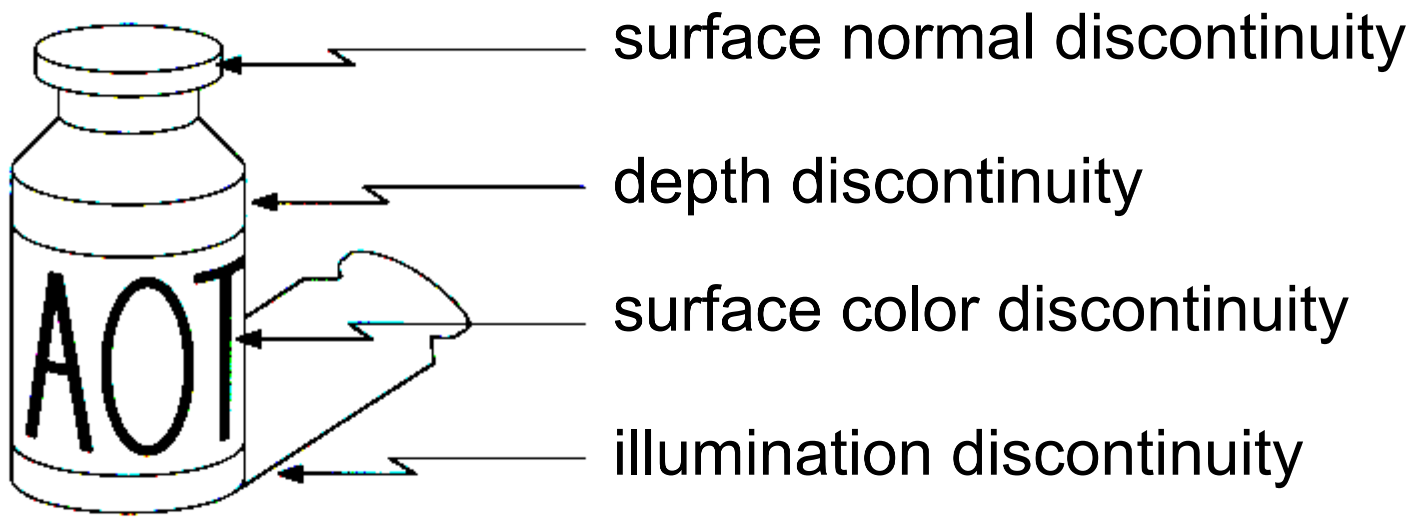




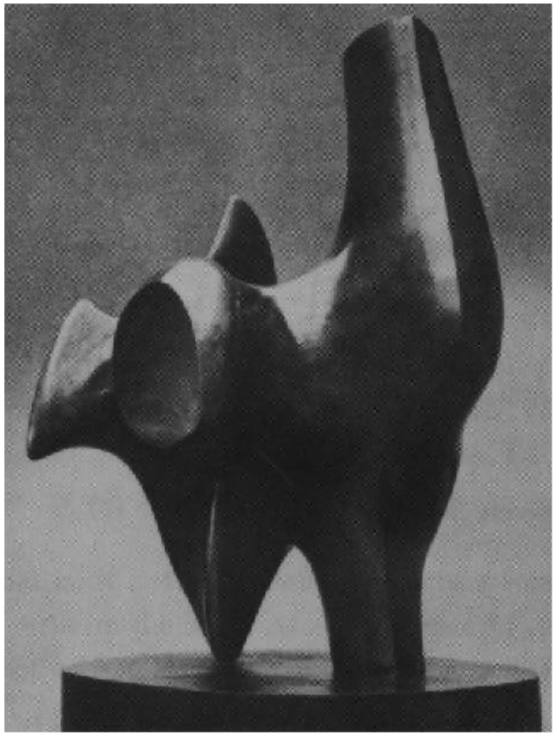
- More compact than pixels

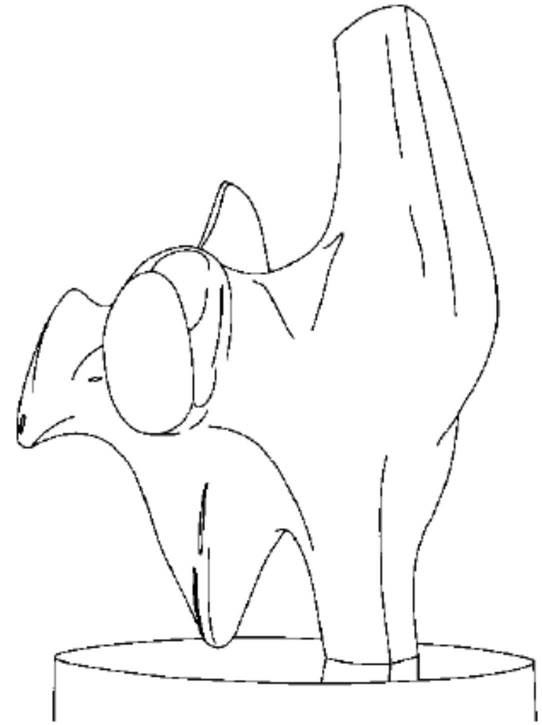
Where do edges come from?

Edges in images are caused by a variety of factors



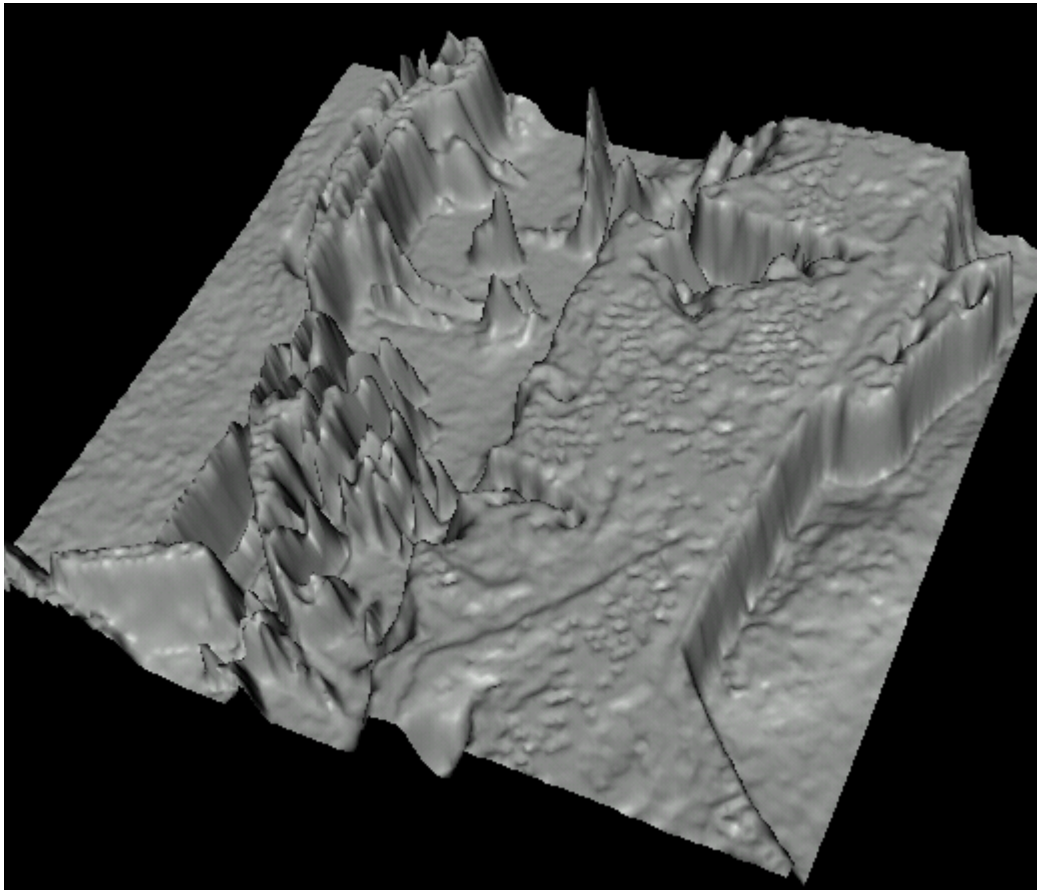
Edge detection





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

Images as functions…

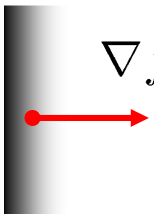


Edges look like steep cliffs

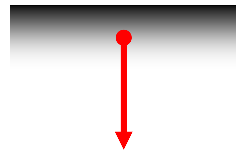
Image gradient

The gradient of an image:

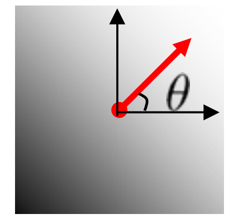
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$



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



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



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

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

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

The discrete gradient

How can we differentiate a *digital* image F[x,y]?

y →

x ↓

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

The discrete gradient

How can we differentiate a *digital* image F[x,y]?

- Answer: take discrete derivative (“finite difference”)

$$\frac{\partial f}{\partial x}[x,y] \approx F[x+1,y] - F[x,y]$$

How would you implement this as a cross-correlation?

	-1	**+1**

H

The Sobel operator

Better approximations of the derivatives exist

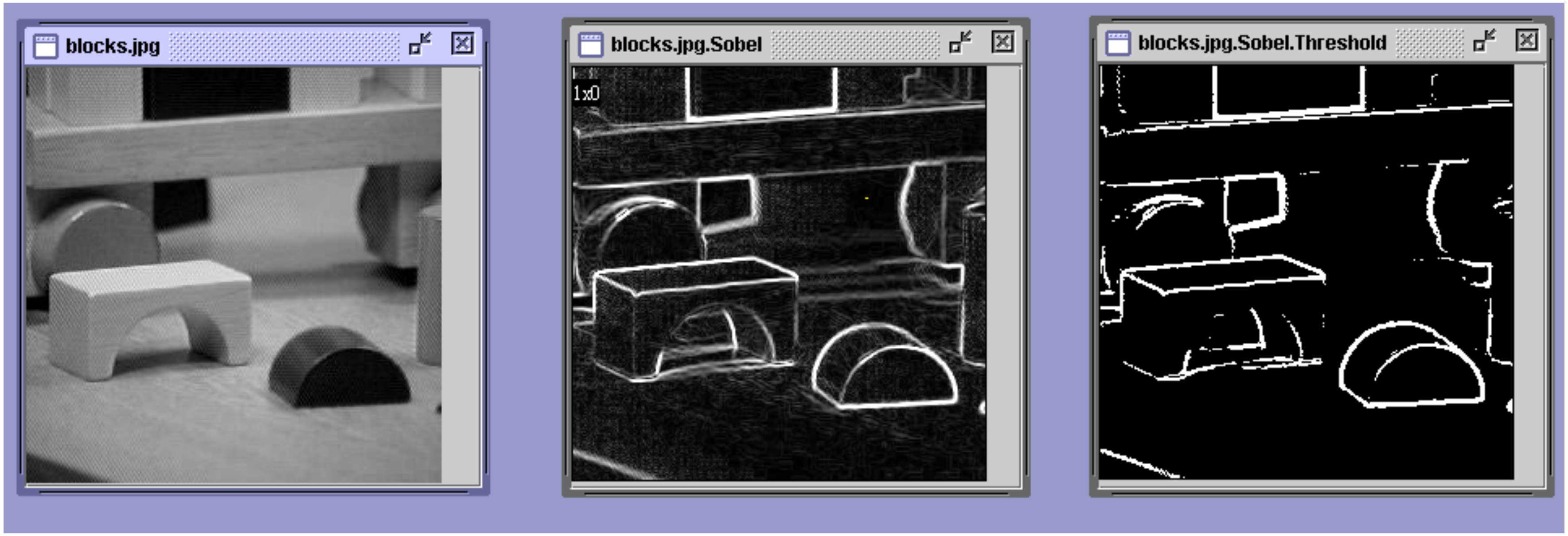
- The *Sobel* operators below are very commonly used

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{8}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

$$s_x \qquad\qquad\qquad s_y$$

- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn't make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient value, however

Edge detection using the Sobel operator



original image

Sobel gradient magnitude

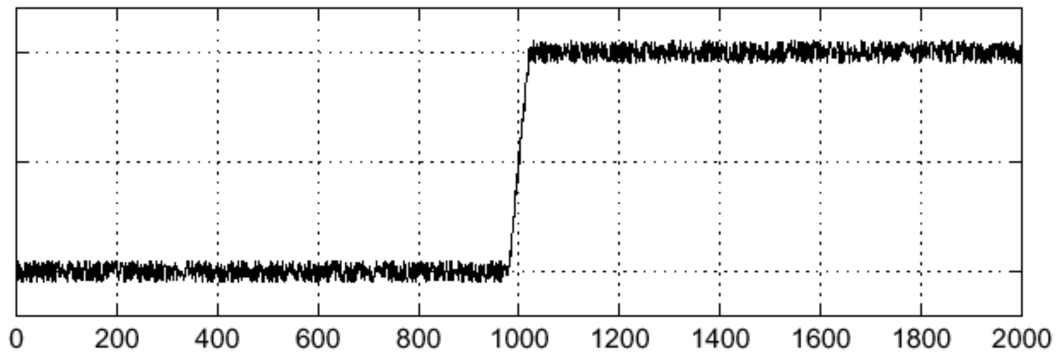
thresholded

Effects of noise

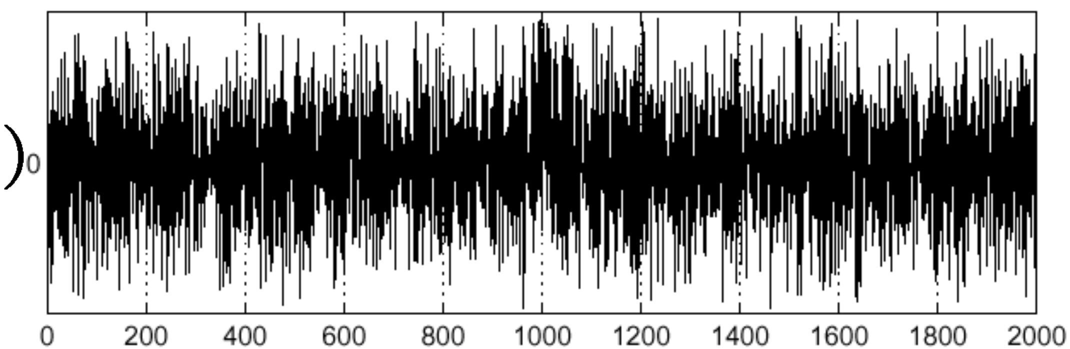
Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

$f(x)$

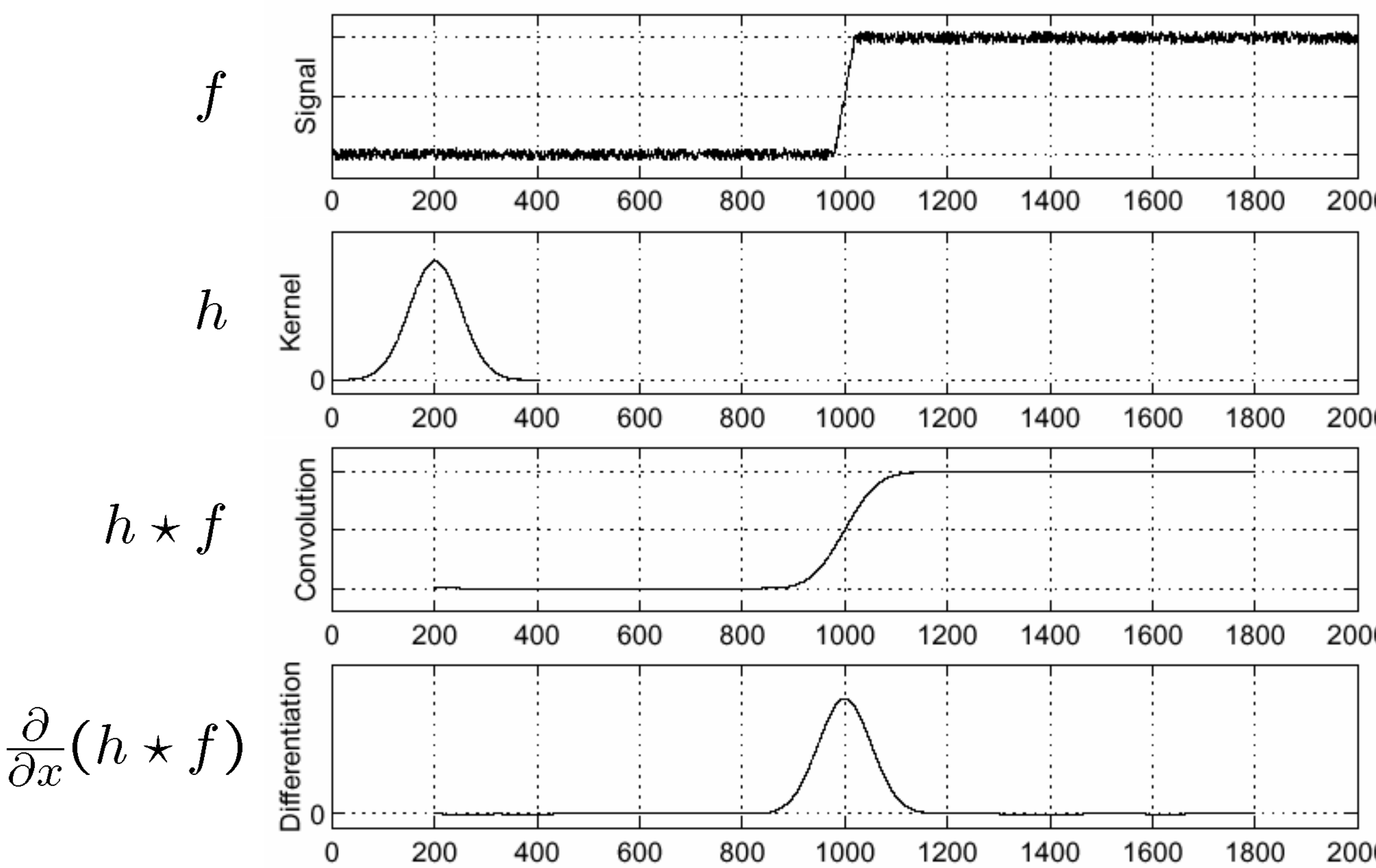


$\frac{d}{dx}f(x)$



Where is the edge?

Solution: smooth first



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

Derivative theorem of convolution

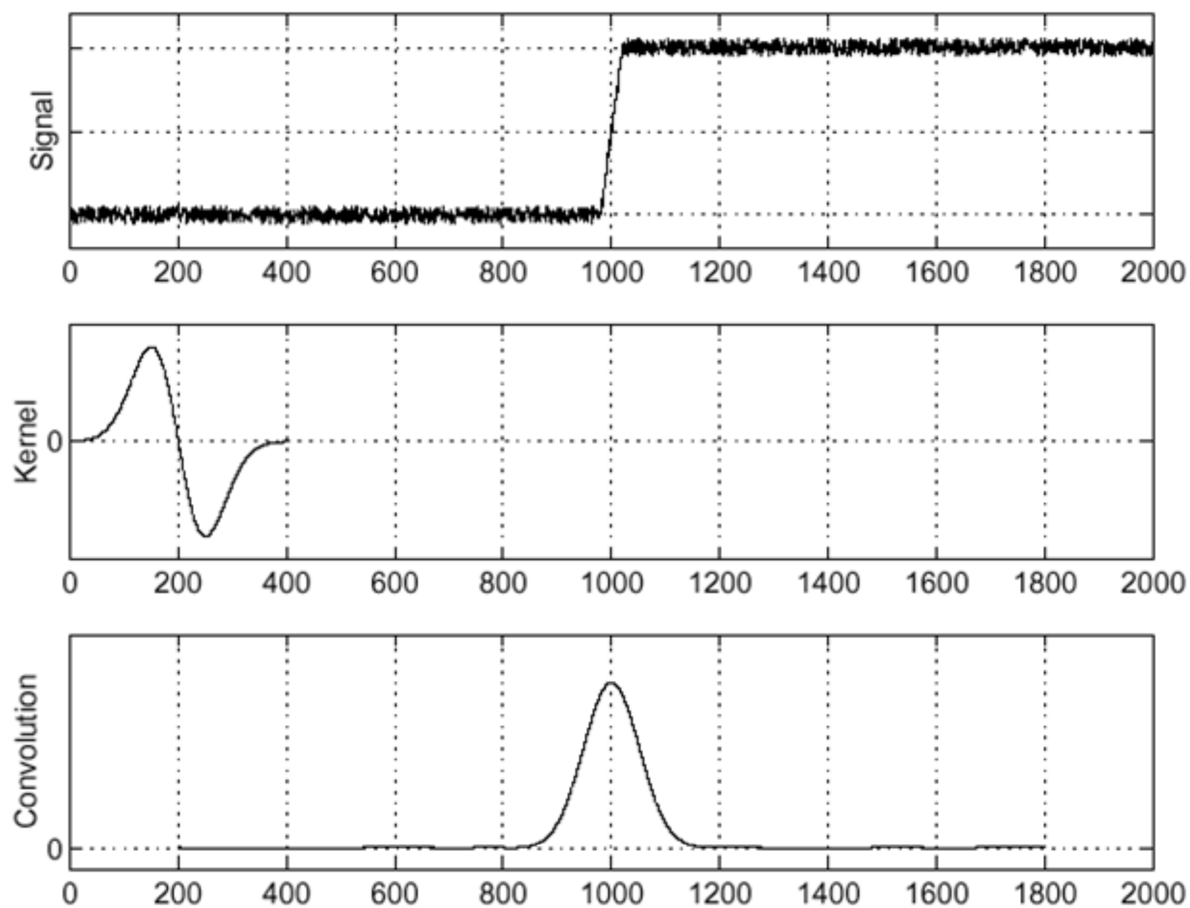
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

This saves us one operation:

f

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

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



Need to find (local) maxima of a function

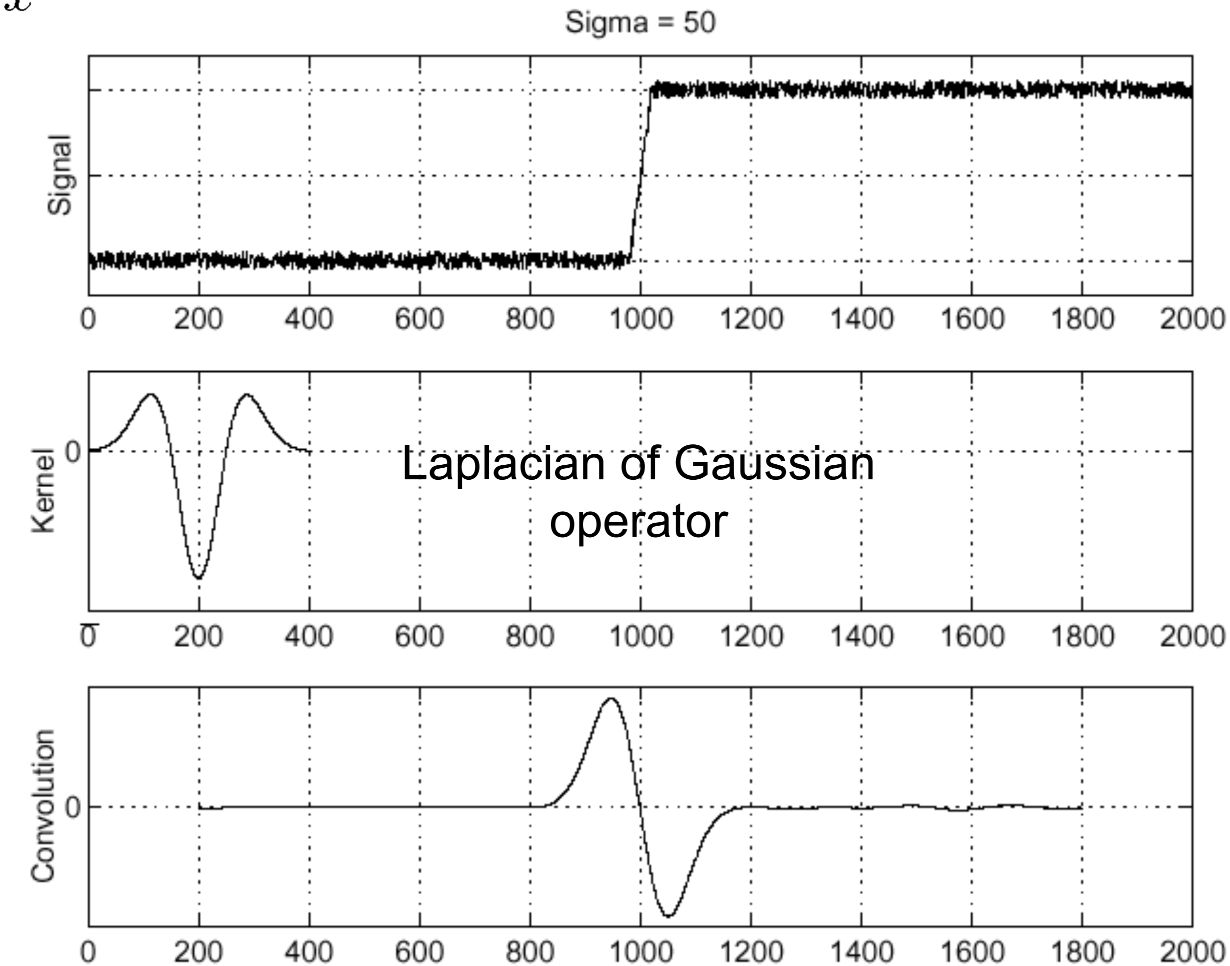
Laplacian of Gaussian

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

f

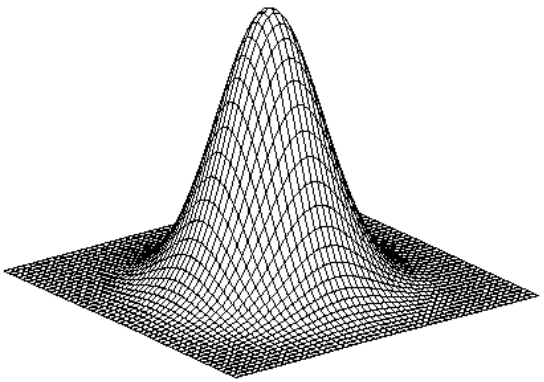
$\frac{\partial^2}{\partial x^2}h$

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



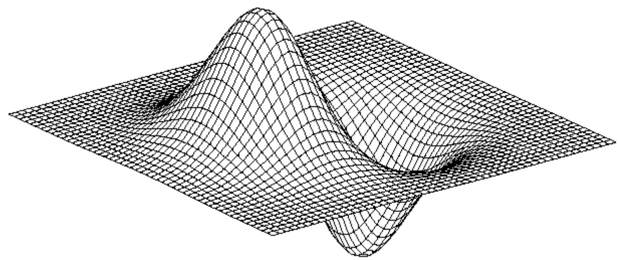
Where is the edge? Zero-crossings of bottom graph

2D edge detection filters



Gaussian

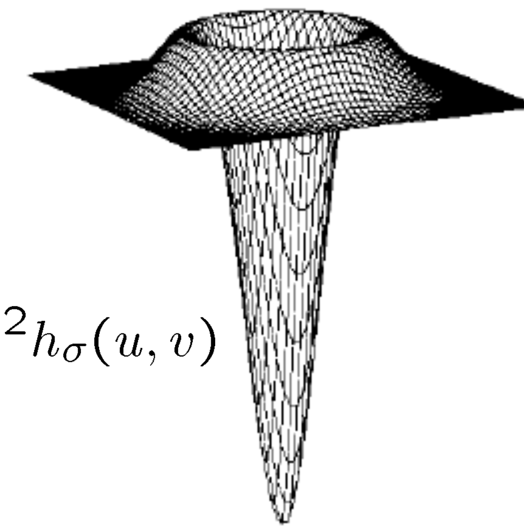
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian
(x direction)

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

Laplacian of Gaussian



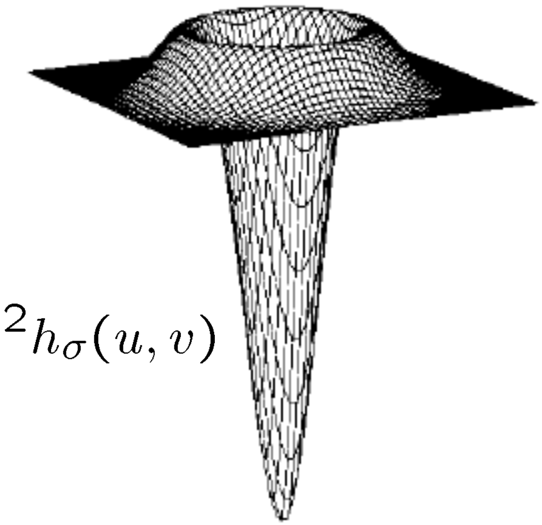
$$\nabla^2 h_\sigma(u,v)$$

∇^2 is the **Laplacian** operator:

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

LoG filter

Laplacian of Gaussian



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

0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

Discrete approximation with σ = 1.4

Edge detection using LoGs

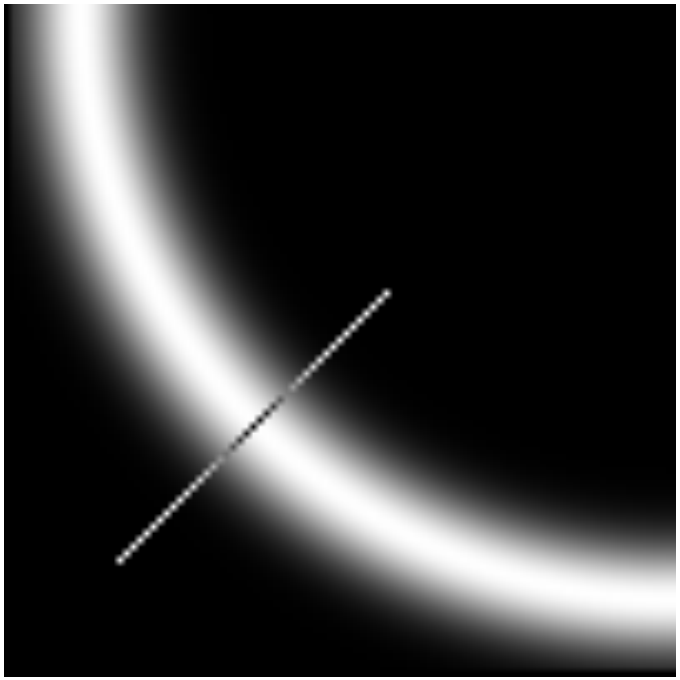
original image (Lena)

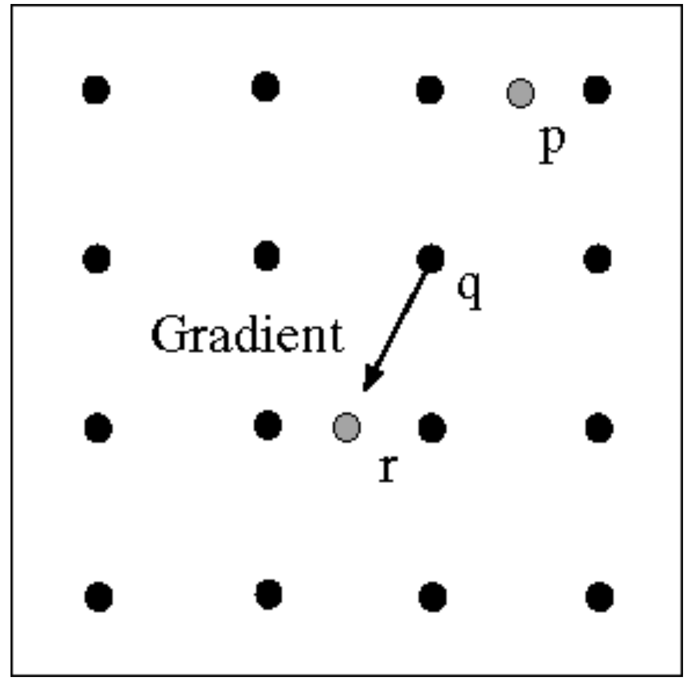
LoG followed by zero crossing detection

Canny Edge Detector

1. Smoothing: Smooth the image with a Gaussian filter with spread σ
2. Gradient: Compute gradient magnitude and direction at each pixel of the smoothed image
3. Thresholding: Threshold the gradient magnitude image such that strong edges are kept and noise is suppressed
4. Non-maximum suppression (thinning): Zero out all pixels that are not the maximum along the direction of the gradient (look at 1 pixel on each side)

Step 4: Thinning (Non-maximum suppression)





Check if pixel is local maximum along gradient direction

- requires checking interpolated pixels p and r

Canny Edge Detector

1. Smoothing: Smooth the image with a Gaussian filter with spread σ
2. Gradient: Compute gradient magnitude and direction at each pixel of the smoothed image
3. Thresholding: Threshold the gradient magnitude image such that strong edges are kept and noise is suppressed
4. Non-maximum suppression (thinning): Zero out all pixels that are not the maximum along the direction of the gradient (look at 1 pixel on each side)
5. Tracing edges: Trace high-magnitude contours and keep only pixels along these contours, so weak little segments go away

The Canny edge detector

original image (Lena)

The Canny edge detector: Step 2

norm of the gradient of smoothed image

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

The Canny edge detector: Step 3

thresholding

The Canny edge detector: Steps 4 & 5

Thinning (single pixel edges) & tracing edges

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