

Edge Detection

SHADOW

From [Sandlot Science](#)

Today's reading

- [Cipolla & Gee on edge detection](#) (available online)

Announcements

Mailing list: cse455a_au12@uw.edu

- you should have received a message

Office hours [online](#)

Project 1 out last Thursday (due 10/10)

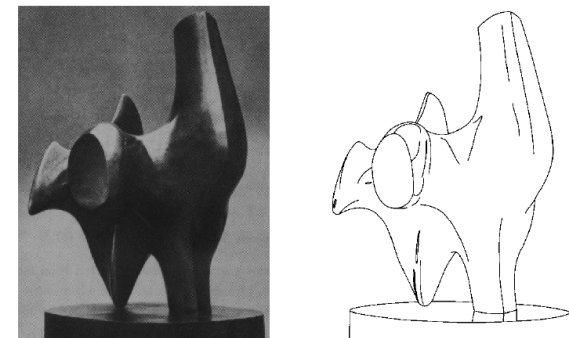
- posted on [course web page](#)

I'm out next week

- office hours via email only next week
- guest lecturer for next 3 classes: Ira Kemelmacher

Filtering demo

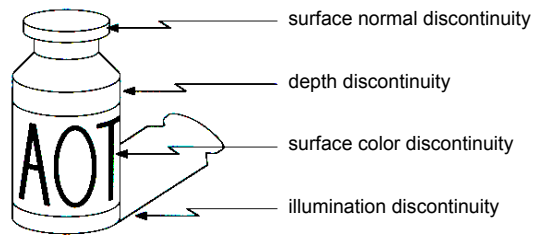
Edge detection



Convert a 2D image into a set of curves

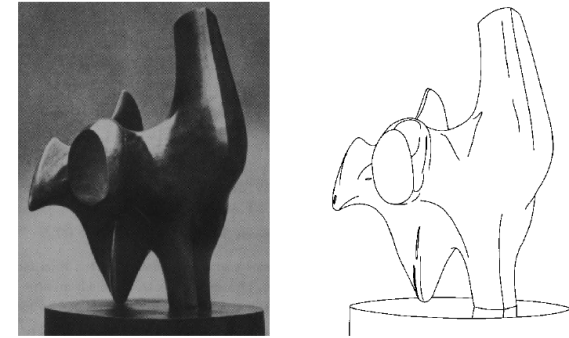
- Extracts salient features of the scene
- More compact than pixels

Origin of Edges



Edges are caused by a variety of factors

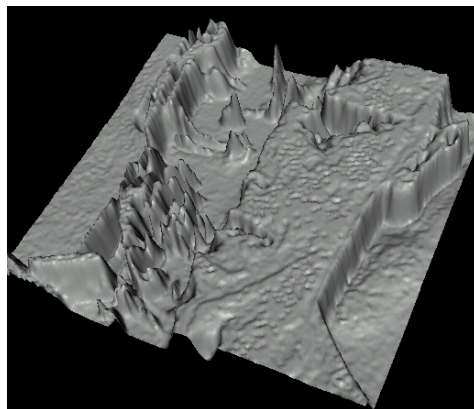
Edge detection



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

snoop demo

Images as functions...

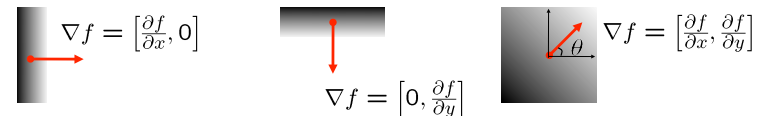


Edges look like steep cliffs

Image gradient

The gradient of an image:

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



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

The gradient direction is given by:

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

- 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 $F[x,y]$?

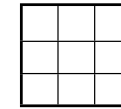
The discrete gradient

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

- Option 1: reconstruct a continuous image, then take gradient
- Option 2: 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?



H

filter demo

The Sobel operator

Better approximations of the derivatives exist

- The *Sobel* operators below are very commonly used

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

S_x

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

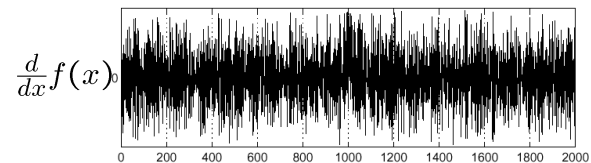
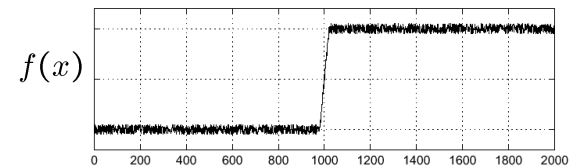
S_y

- 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

Effects of noise

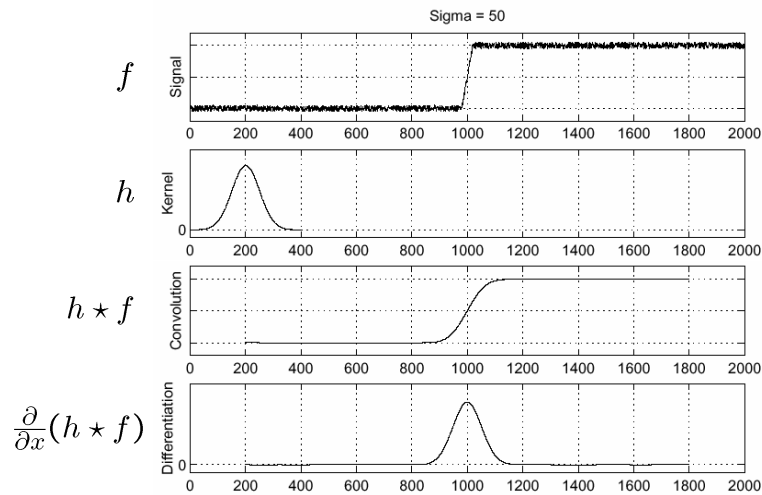
Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal



Where is the edge?

Solution: smooth first

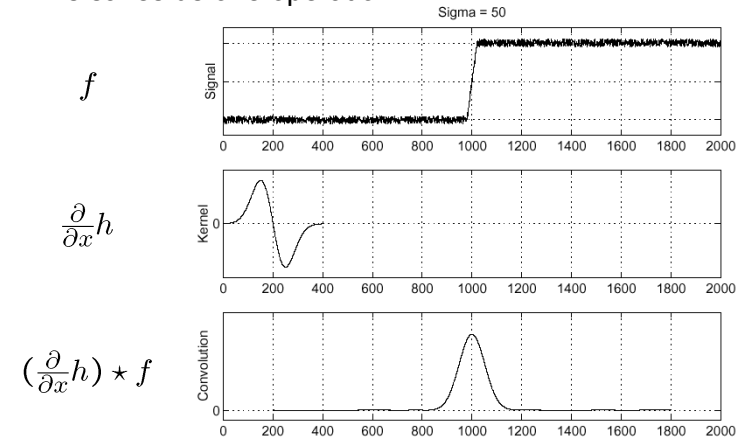


Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

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

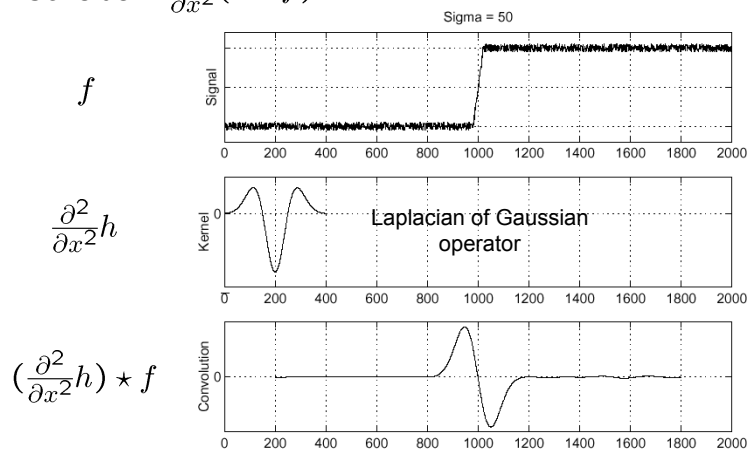
This saves us one operation:



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

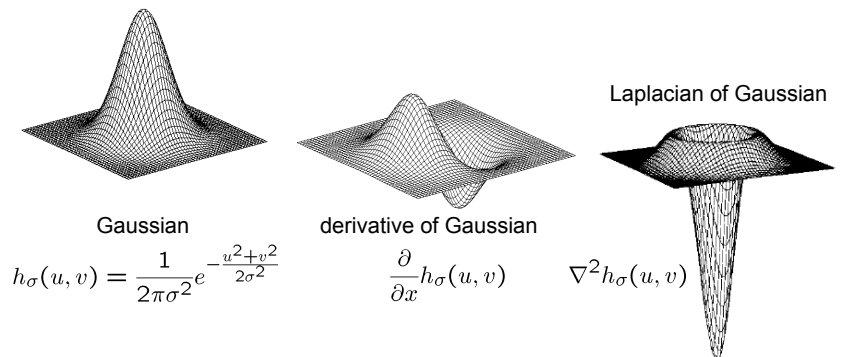
Laplacian of Gaussian

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



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

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

Laplacian of Gaussian

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

∇^2 is the **Laplacian operator**:

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

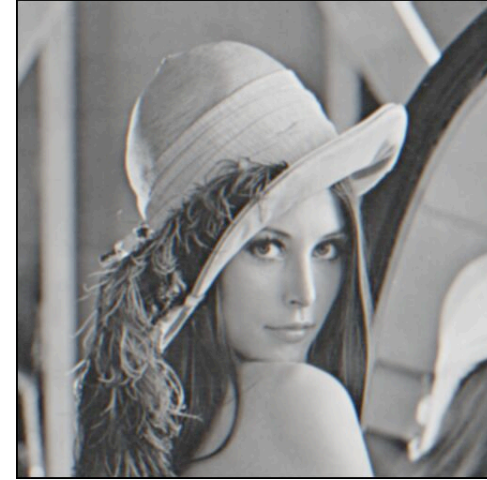
filter demo

Edge detection by subtraction



original

Edge detection by subtraction



smoothed (5x5 Gaussian)

Edge detection by subtraction

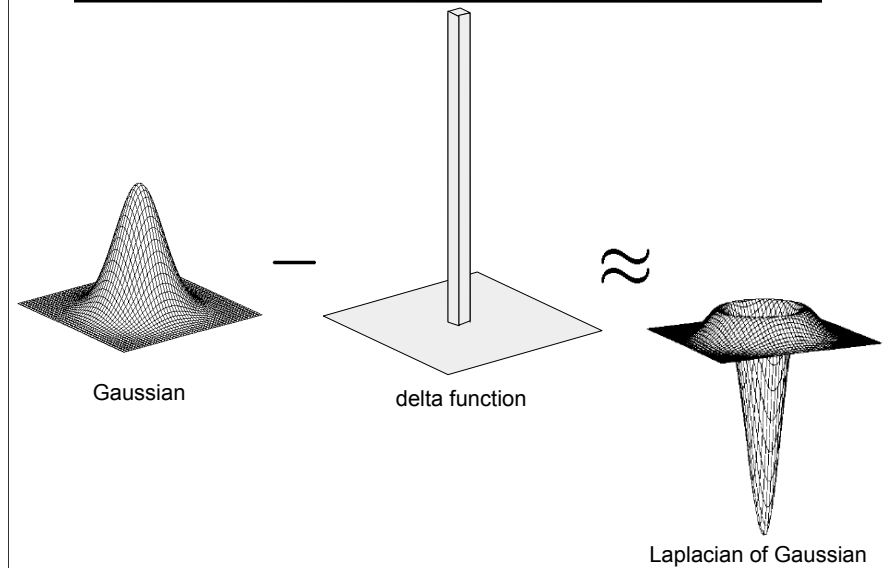


smoothed - original
(scaled by 4, offset +128)

filter demo

Why does
this work?

Gaussian - image filter



The Canny edge detector



original image (Lena)

The Canny edge detector



norm of the gradient

The Canny edge detector



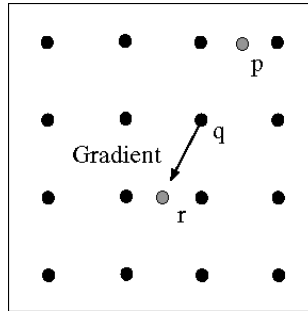
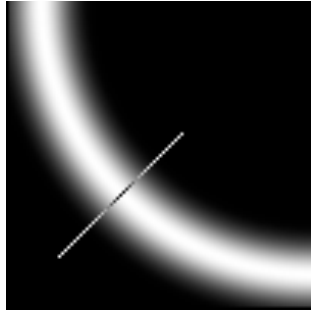
thresholding

The Canny edge detector



thinning
(non-maximum suppression)

Non-maximum suppression



Check if pixel is local maximum along gradient direction

- requires checking interpolated pixels p and r

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