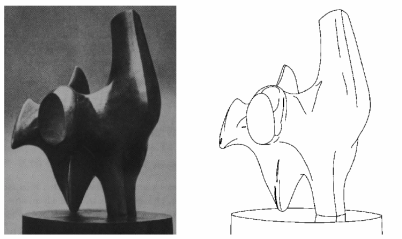
Edge Detection

From Sandlot Science

Today's readings

- Cipolla and Gee
  - supplemental: Forsyth, chapter 9
- Watt, 10.3-10.4

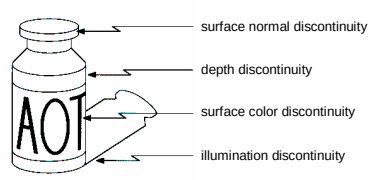
Edge detection



Convert a 2D image into a set of curves

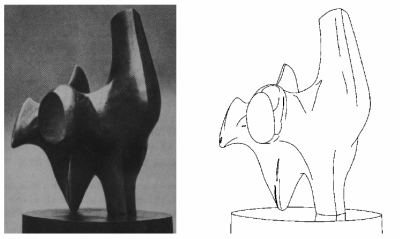
- Extracts salient features of the scene
- More compact than pixels

Origin of Edges



Edges are caused by a variety of factors

Edge detection



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

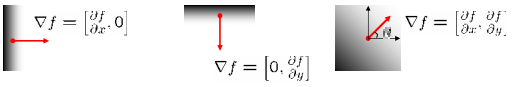
snoop demo

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 change 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]?

- 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

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

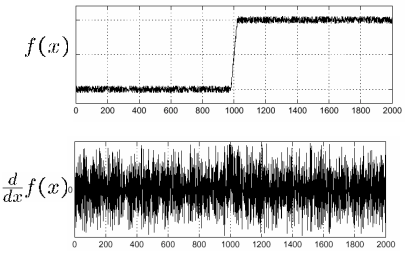
$$\frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad H_y$$

- 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

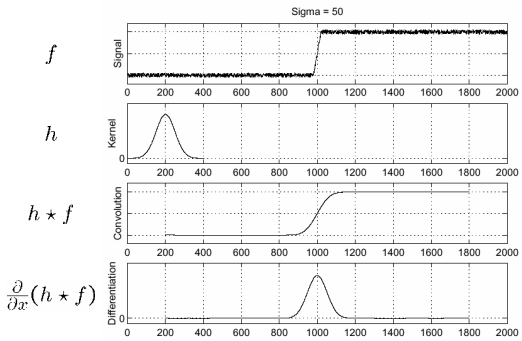
Consider a single row or column of the image

- Plotting intensity as a function of position gives a *signal*



Where is the edge?

Solution: smooth first

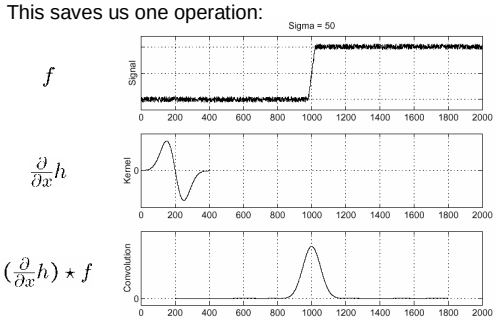


Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

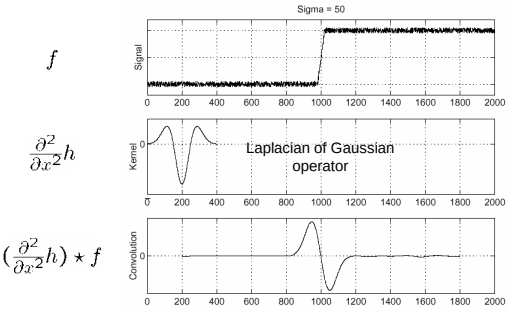
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

This saves us one operation:

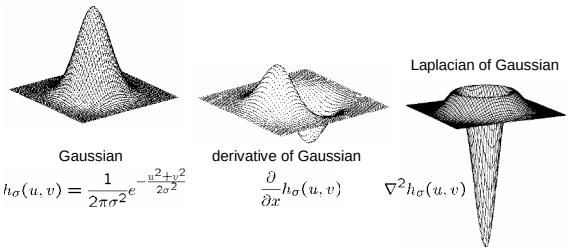


Laplacian of Gaussian

Look for zero-crossings of $\frac{\partial^2}{\partial x^2}(h \star f)$



2D edge detection filters



∇^2 is the **Laplacian** operator:

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

filter demo

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, edge following)

Effect of Gaussian kernel width

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

Edge detection by subtraction

original

Edge detection by subtraction

smoothed (5x5 Gaussian)

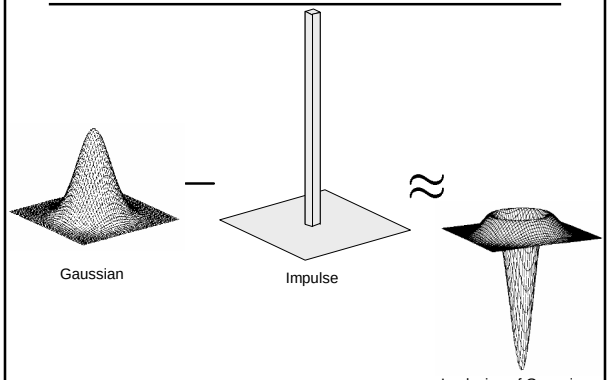
Edge detection by subtraction

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

Why does this work?

filter demo

Gaussian - subtraction filter



An edge is not a line...

How can we detect ***lines*** ?

Finding lines in an image

Option 1:

- Search for the object at every possible position in the image
- What is the cost of this operation?

Option 2:

- Use a voting scheme: Hough transform

Finding lines in an image

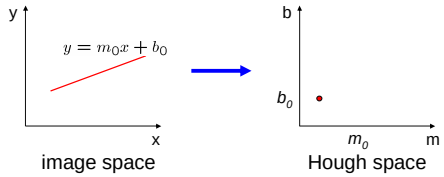
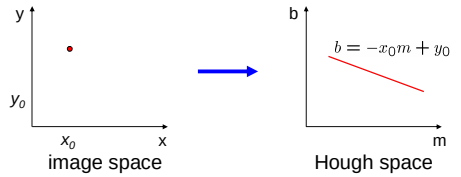


image space Hough space

Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b

Finding lines in an image



Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b
- What does a point (x_0, y_0) in the image space map to?
  - A: the solutions of b = $-x_0$m + y_0
  - this is a line in Hough space

Hough transform algorithm

Typically use a different parameterization

$$d = xcos\theta + ysin\theta$$

- d is the perpendicular distance from the line to the origin
- θ is the angle this perpendicular makes with the x axis
- Why?

Hough transform algorithm

Typically use a different parameterization

$$d = xcos\theta + ysin\theta$$

- d is the perpendicular distance from the line to the origin
- θ is the angle this perpendicular makes with the x axis
- Why?

Basic Hough transform algorithm

1. Initialize H[d, θ]=0
2. for each edge point I[x,y] in the image
   for θ = 0 to 180
   $d = xcos\theta + ysin\theta$
   H[d, θ] += 1
3. Find the value(s) of (d, θ) where H[d, θ] is maximum
4. The detected line in the image is given by $d = xcos\theta + ysin\theta$

What's the running time (measured in # votes)?

Hough line demo

Extensions

Extension 1: Use the image gradient

1. same
2. for each edge point I[x,y] in the image
   compute unique (d, θ) based on image gradient at (x,y)
   H[d, θ] += 1
3. same
4. same

What's the running time measured in votes?

Extensions

Extension 1: Use the image gradient

1. same
2. for each edge point I[x,y] in the image
   compute unique (d, θ) based on image gradient at (x,y)
   H[d, θ] += 1
3. same
4. same

What's the running time measured in votes?

Extension 2

- give more votes for stronger edges

Extension 3

- change the sampling of (d, θ) to give more/less resolution

Extension 4

- The same procedure can be used with circles, squares, or any other shape

Hough circle demo

Summary

Things to take away from this lecture

- What is an edge and where does it come from
- Edge detection by differentiation
- Image gradients
  - continuous and discrete
  - filters (e.g., Sobel operator)
- Effects of noise on gradients
- Derivative theorem of convolution
- Derivative of Gaussian (DoG) operator
- Laplacian operator
  - Laplacian of Gaussian (LoG)
- Canny edge detector (basic idea)
  - Effects of varying sigma parameter
- Approximating an LoG by subtraction
- Hough Transform