## Announcements

- Mailing list: csep576@cs.washington.edu
  - you should have received messages
- Project 1 out today (due in two weeks)
- Carpools

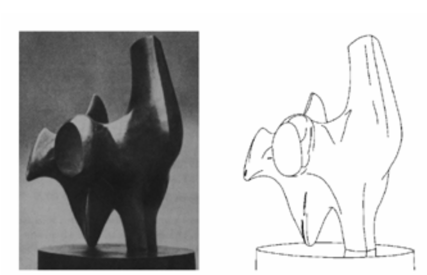
## Edge Detection


From Sandlot Science

Today's reading

- Forsyth, chapters 8, 15.1

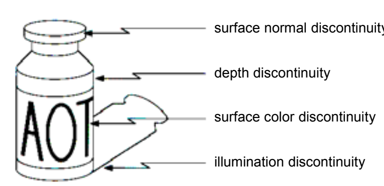
## Edge detection



Convert a 2D image into a set of curves

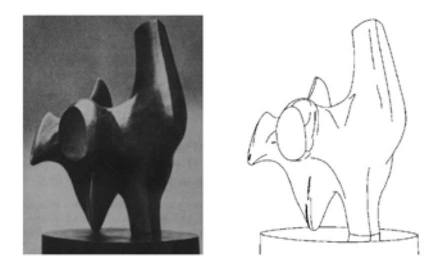
- Extracts salient features of the scene
- More compact than pixels

## Origin of Edges




Edges are caused by a variety of factors

## Edge detection



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

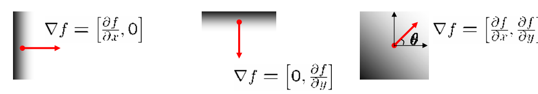
snoop demo

## 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]?

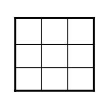
## The discrete gradient

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

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

$$s_x \qquad\qquad s_y$$

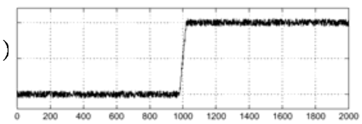
- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn't make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient value, however

# Effects of noise

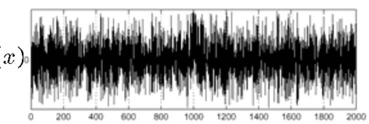
Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

$f(x)$



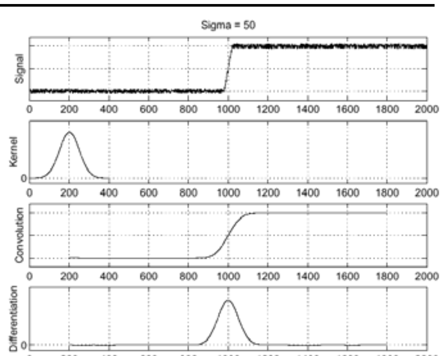
$\frac{d}{dx}f(x)$



Where is the edge?

# Solution: smooth first

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



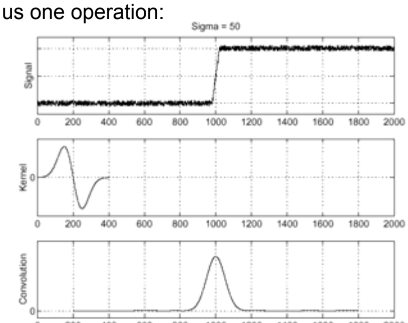

Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

# Derivative theorem of convolution

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

This saves us one operation:

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



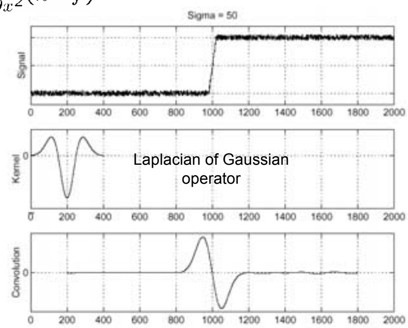
## Laplacian of Gaussian

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

$f$

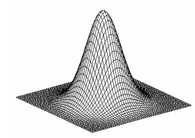
$\frac{\partial^2}{\partial x^2}h$

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



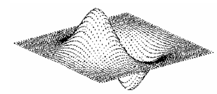

Where is the edge? Zero-crossings of bottom graph

## 2D edge detection filters



Gaussian

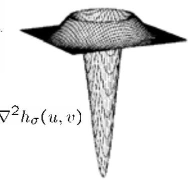
$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2}e^{-\frac{u^2+v^2}{2\sigma^2}}$



derivative of Gaussian

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

Laplacian of Gaussian



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

$\nabla^2$ 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)

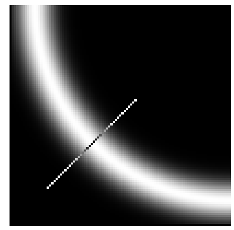
## The Canny edge detector

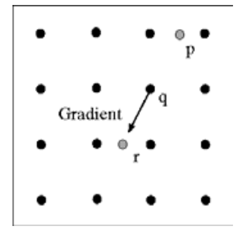
norm of the gradient

## The Canny edge detector

thresholding

## Non-maximum suppression






Check if pixel is local maximum along gradient direction

- requires 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 $\sigma$ depends on desired behavior

- large $\sigma$ detects large scale edges
- small $\sigma$ detects fine features

## Edge detection by subtraction

original

## Edge detection by subtraction

smoothed (5x5 Gaussian)

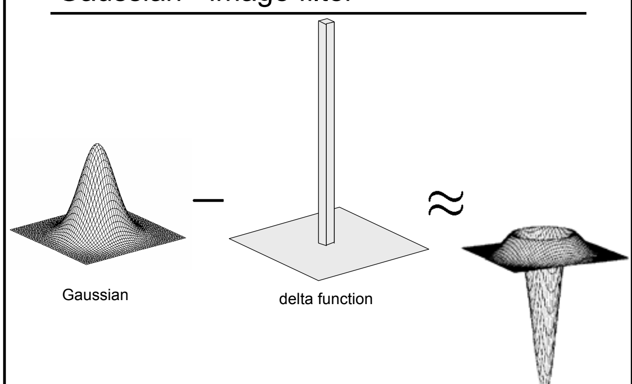
## Edge detection by subtraction

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

Why does this work?

filter demo

## Gaussian - image filter




Laplacian of Gaussian

## An edge is not a line...

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

## Finding lines in an image

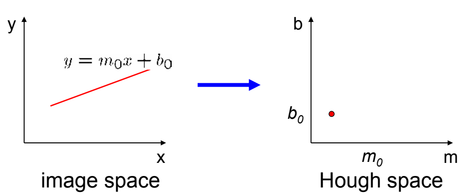
Option 1:

- Search for the line at every possible position/orientation
- What is the cost of this operation?

Option 2:

- Use a voting scheme: Hough transform

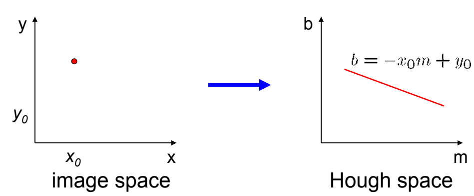
## Finding lines in an image




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

## 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 $b = -x_0m + y_0$
  - this is a line in Hough space

## Hough transform algorithm

Typically use a different parameterization

$$d = x cos\theta + y sin\theta$$

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


## Hough transform algorithm

Typically use a different parameterization

$$d = x cos\theta + y sin\theta$$

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


Basic Hough transform algorithm

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

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, $\theta$) based on image gradient at (x,y)
   H[d, $\theta$] += 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, $\theta$) based on image gradient at (x,y)
   H[d, $\theta$] += 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, $\theta$) to give more/less resolution

Extension 4

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

Hough circle demo