Université Concordia University

FACULTY OF ENGINEERING AND COMPUTER SCIENCE

Department of Computer Science and Software Engineering

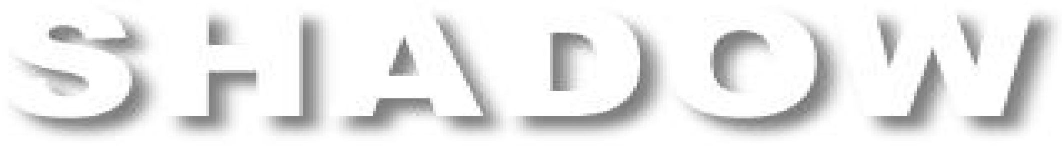
COMP498G/691G COMPUTER VISION

LECTURE 4 EDGE DETECTION

Today’s Lecture

- Edge detection
  - Slides acknowledgment: A. Farhadi, S. Seitz
- Questions

Edges and Scale

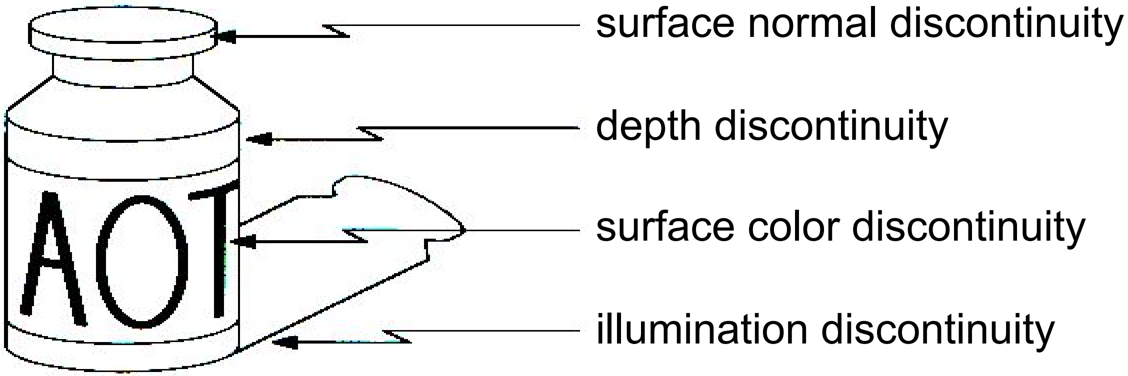


From Sandlot Science

Today's reading

- Cipolla & Gee on edge detection (on class website)
- Szeliski Ch. 3.2, 3.4, 3.5, 4.2

Origin of Edges

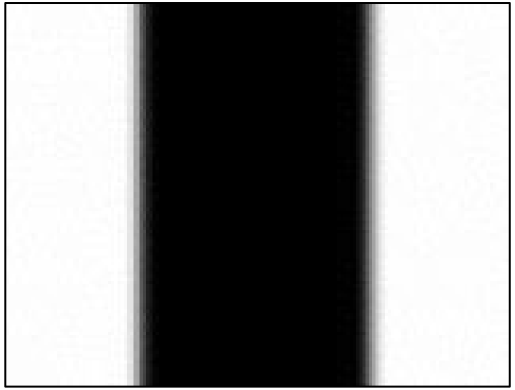


Edges are caused by a variety of factors

Characterizing edges

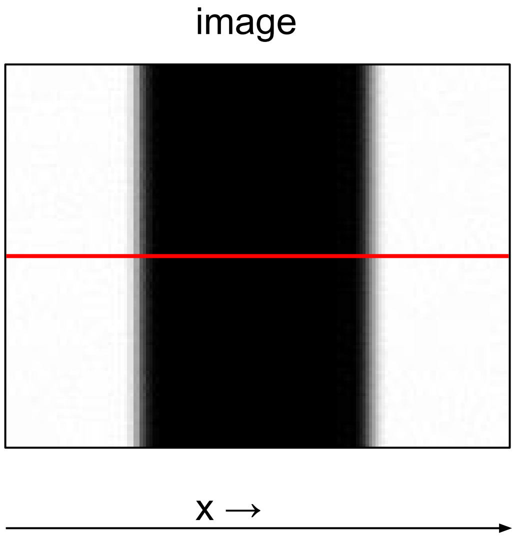
An edge is a place of rapid change in the image intensity function

image



Characterizing edges

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

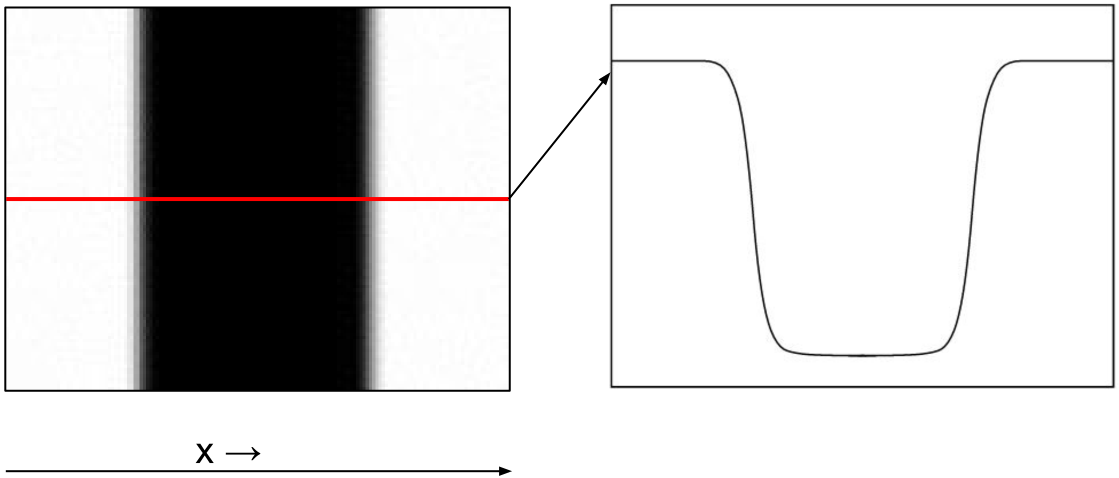


Characterizing edges

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

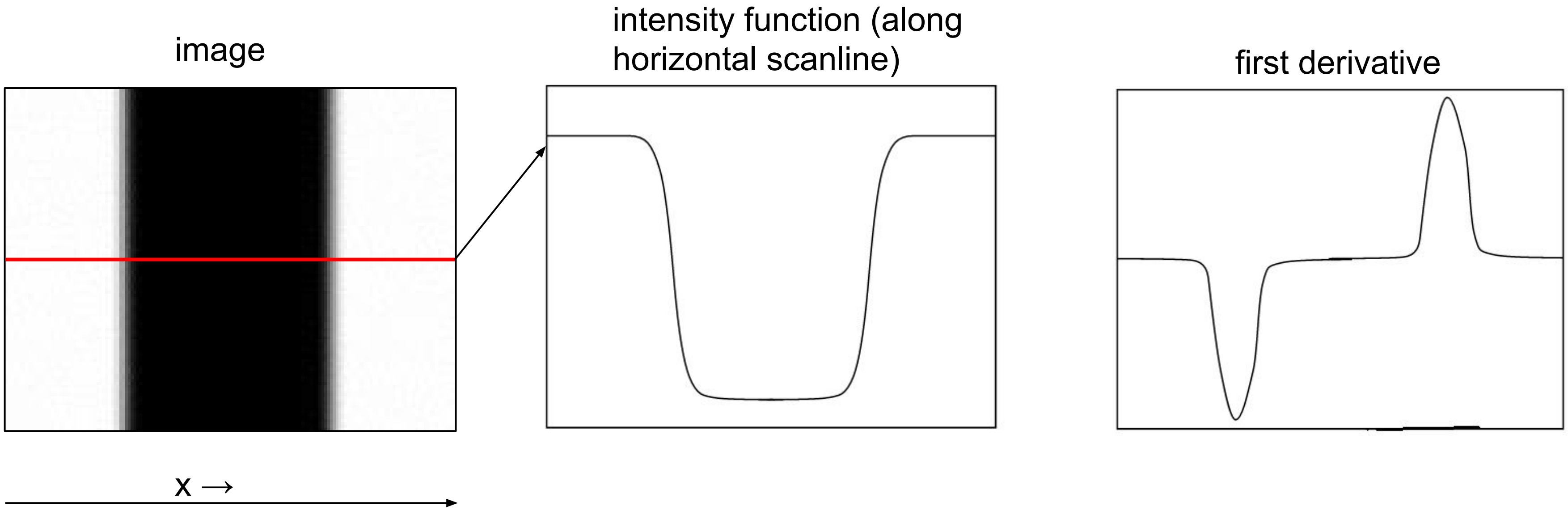
image

intensity function (along horizontal scanline)



Characterizing edges

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



Characterizing edges

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

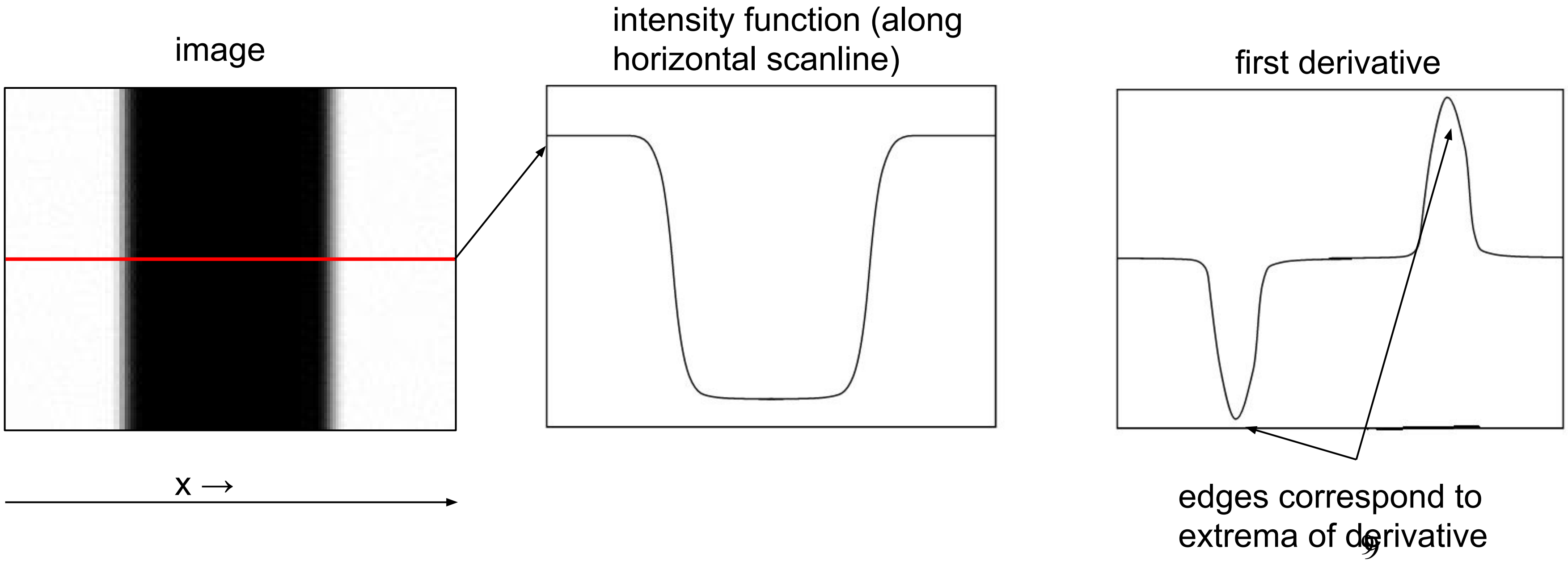


Image Gradient

$\frac{\partial f}{\partial x}$

Image Gradient

$\frac{\partial f}{\partial x}$

- The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

Image Gradient

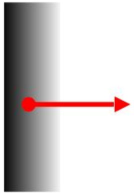
$\frac{\partial f}{\partial x}$

- 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 rapid change in intensity

Image Gradient

$\frac{\partial f}{\partial x}$

- 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 rapid change in intensity

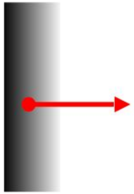


$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$

Image Gradient

$\frac{\partial f}{\partial x}$

- 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 rapid change in intensity



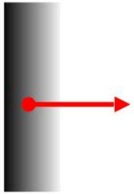
$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$

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

Image Gradient

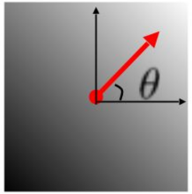
$\frac{\partial f}{\partial x}$

- 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 rapid change in intensity



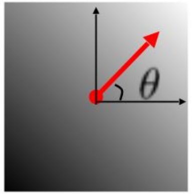
$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$

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



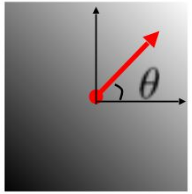
$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

Image Gradient



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

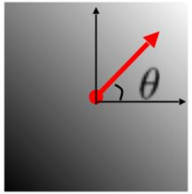
Image Gradient



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

- The gradient direction is given by: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

Image Gradient

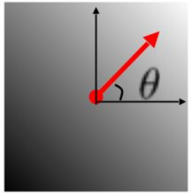


$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- 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?

Image Gradient



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

- 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

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")

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")

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

Step size is 1

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

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

s_x

The Sobel operator

Better approximations of the derivatives exist

- The Sobel operators below are very commonly used

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

The Sobel operator

Better approximations of the derivatives exist

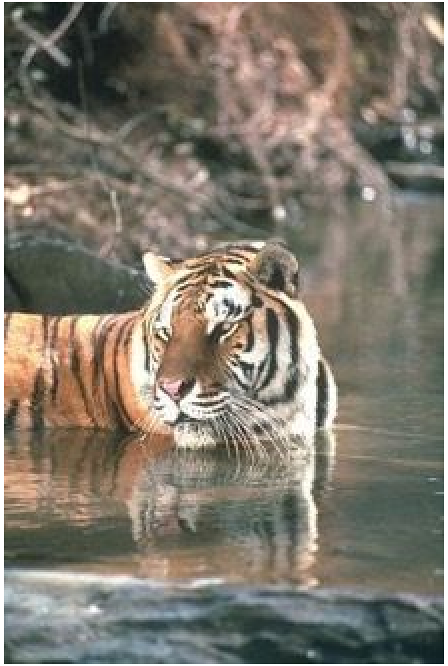
- The Sobel operators below are very commonly used

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

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

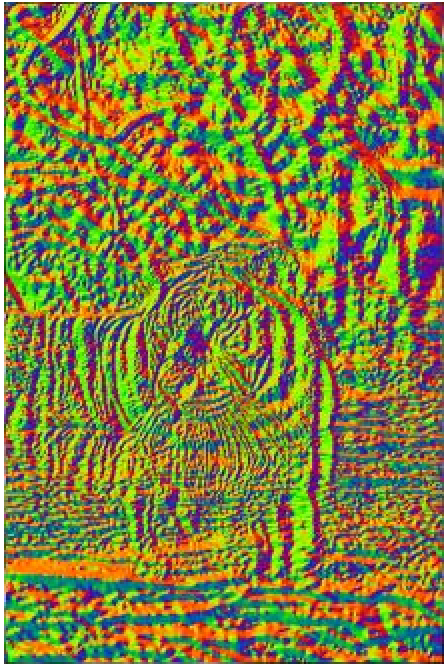
- The standard definition of the Sobel operator omits the 1/8 term
  - doesn’t make a difference for edge detection
  - the 1/8 term is needed to get the right gradient value, however

The Sobel operator



Original

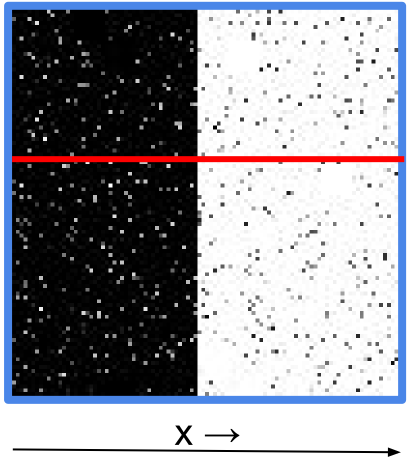
Magnitude



Orientation

Effects of noise

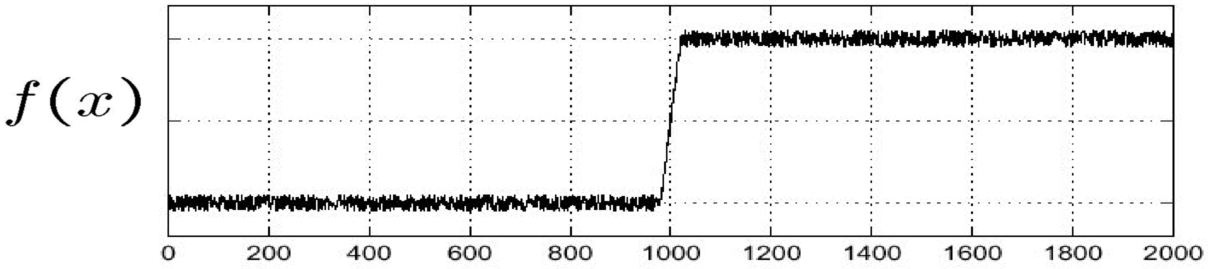
Consider a single row or column of the image

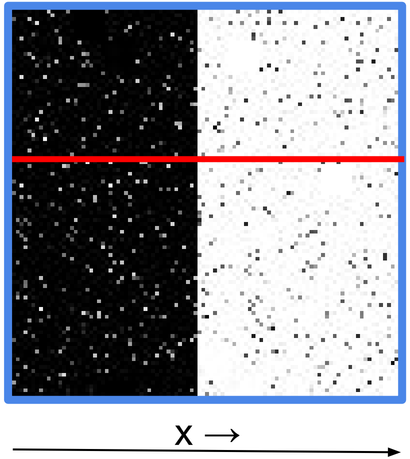


Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal



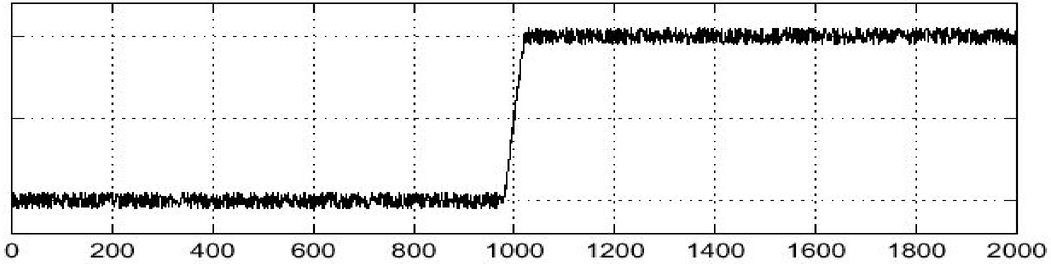


Effects of noise

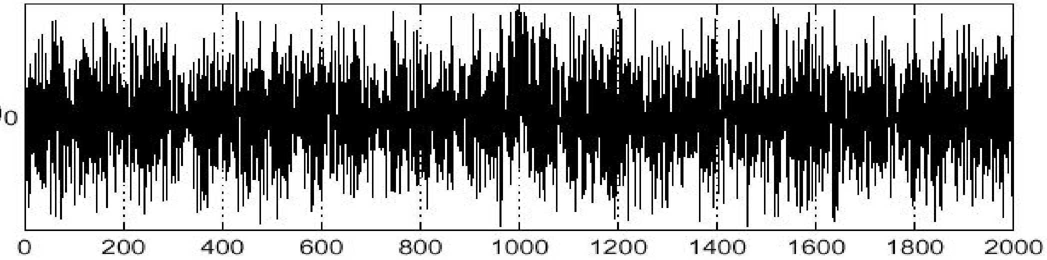
Consider a single row or column of the image

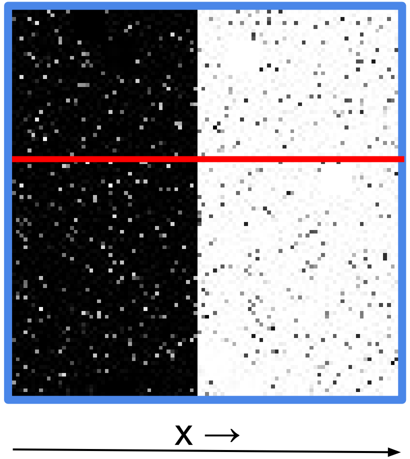
- Plotting intensity as a function of position gives a signal

$f(x)$



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



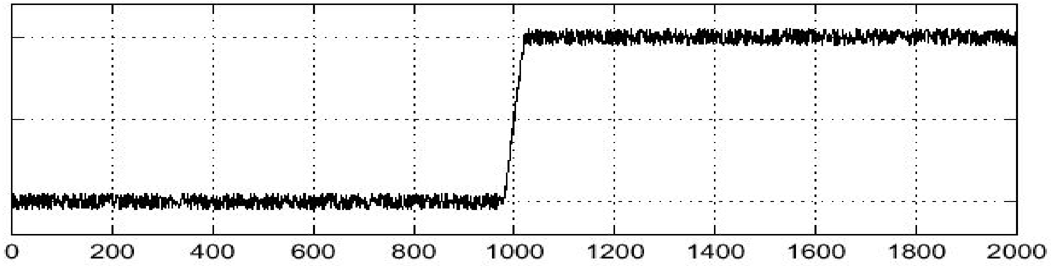


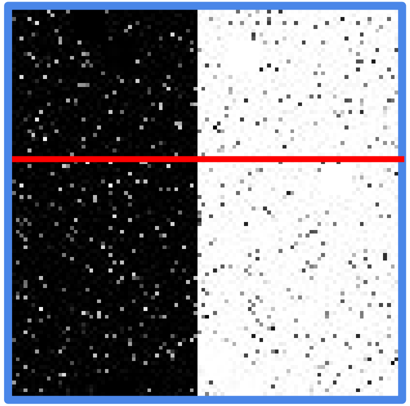
Effects of noise

Consider a single row or column of the image

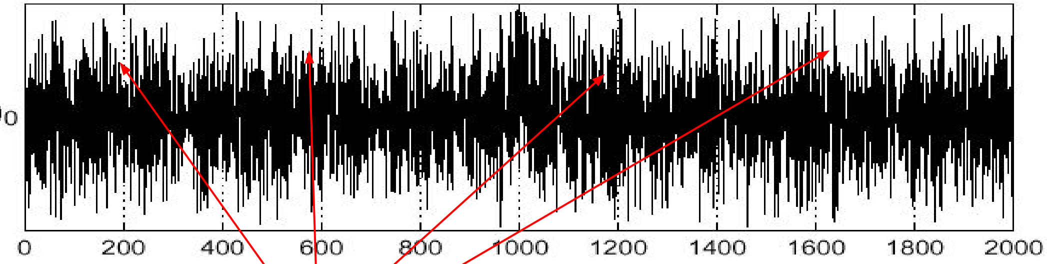
- Plotting intensity as a function of position gives a signal

$f(x)$





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



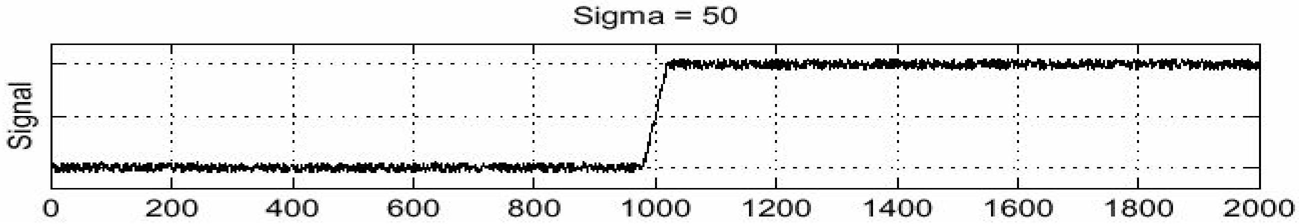
Where is the edge?

Effects of noise

- Difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbours
  - Generally, the larger the noise the stronger the response
- What can we do about it?

Solution: smooth first

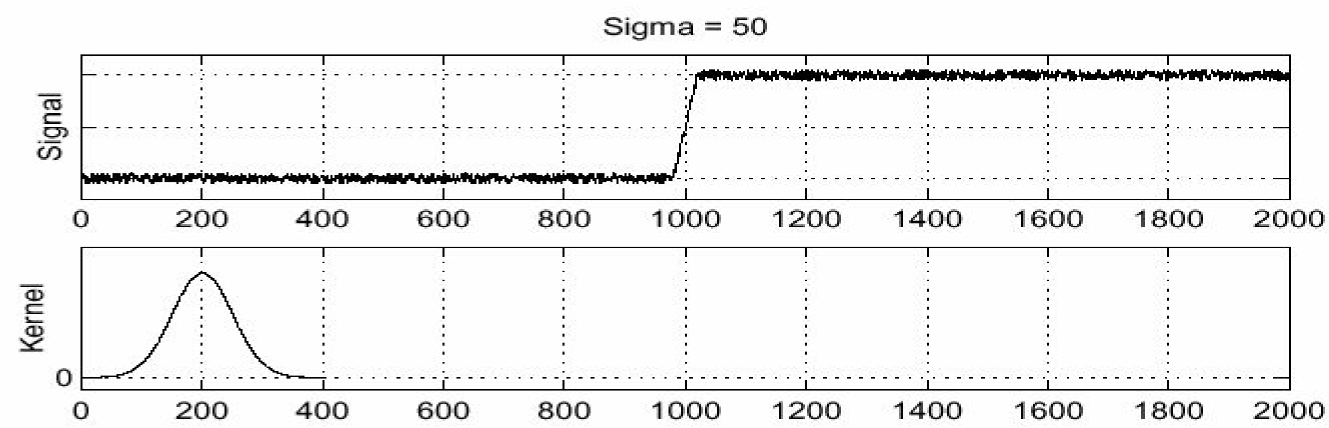
f



Solution: smooth first

f

h

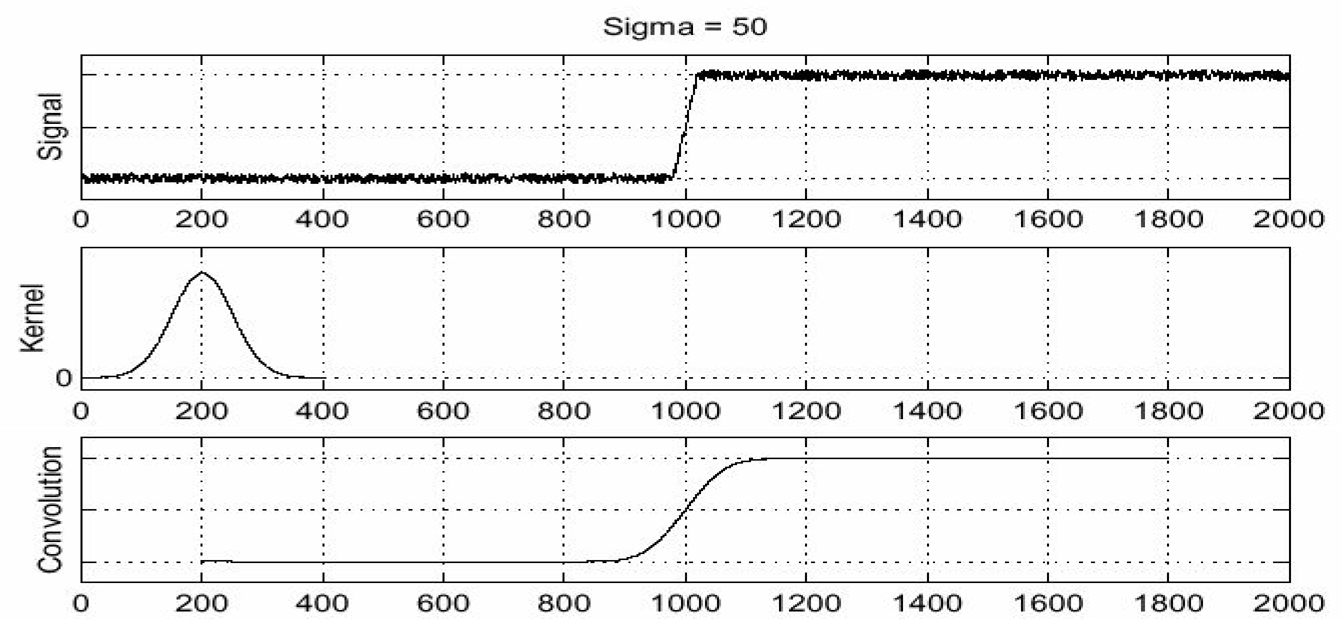


Solution: smooth first

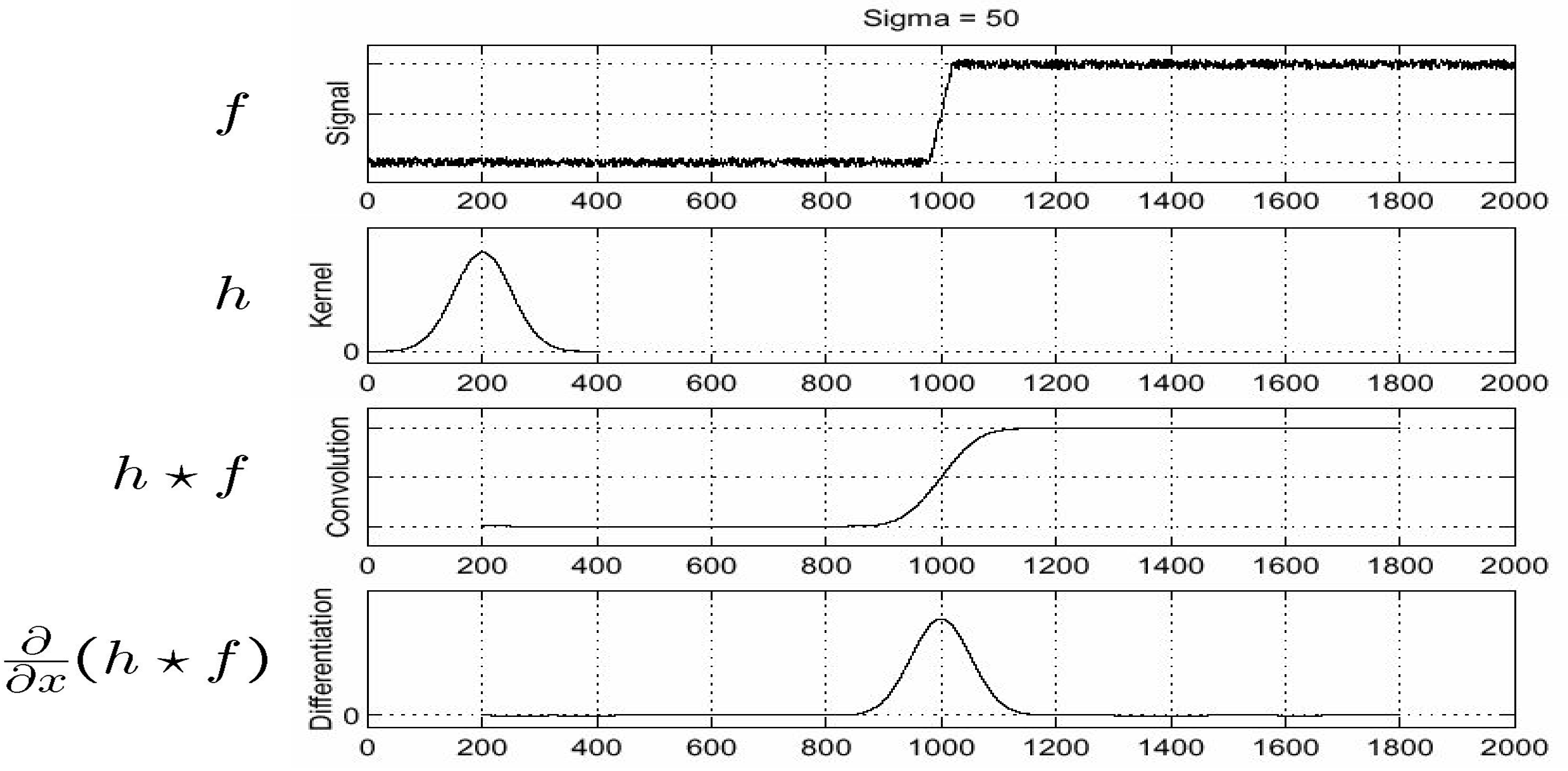
f

h

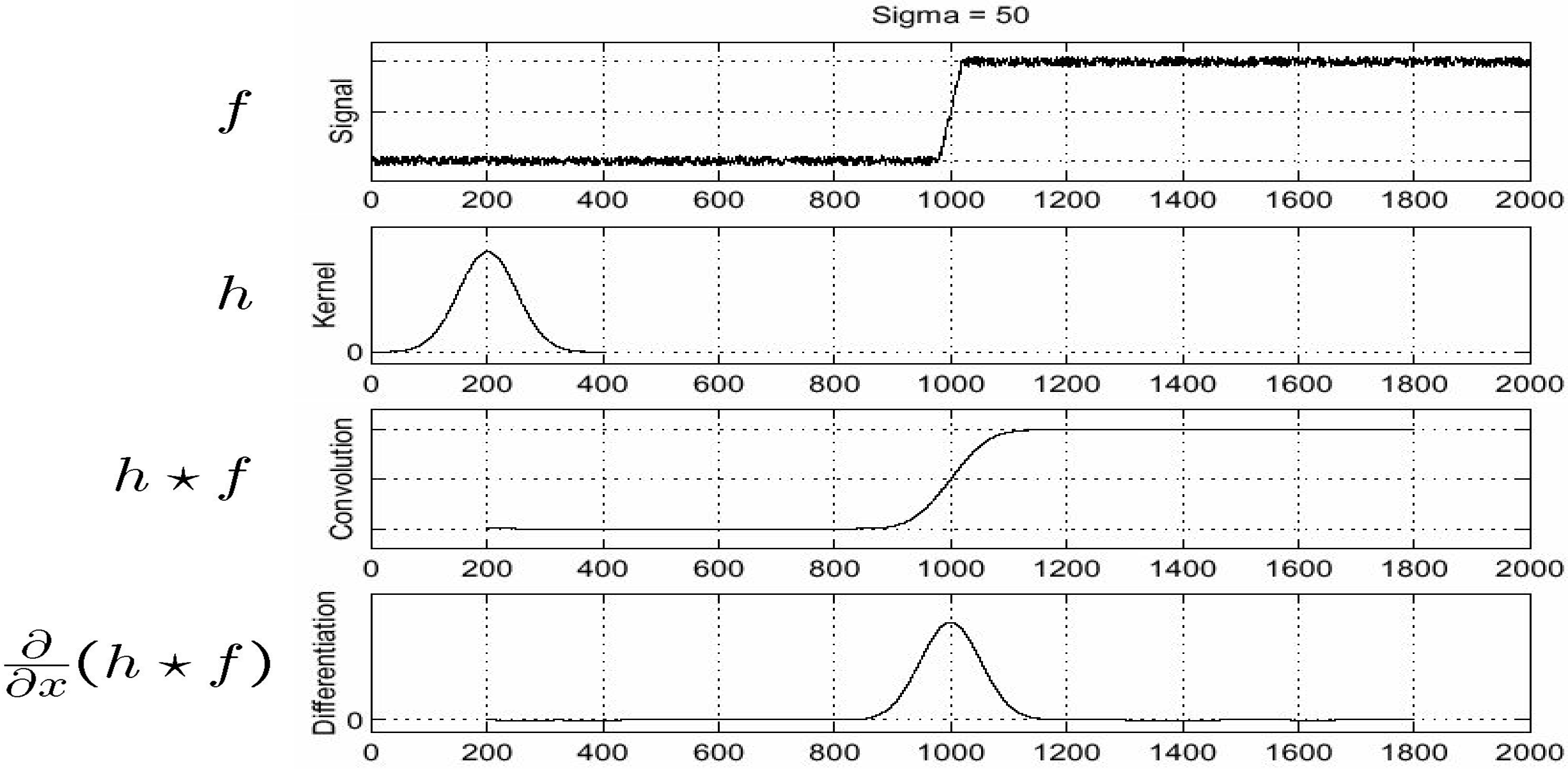
$h \star f$



Solution: smooth first

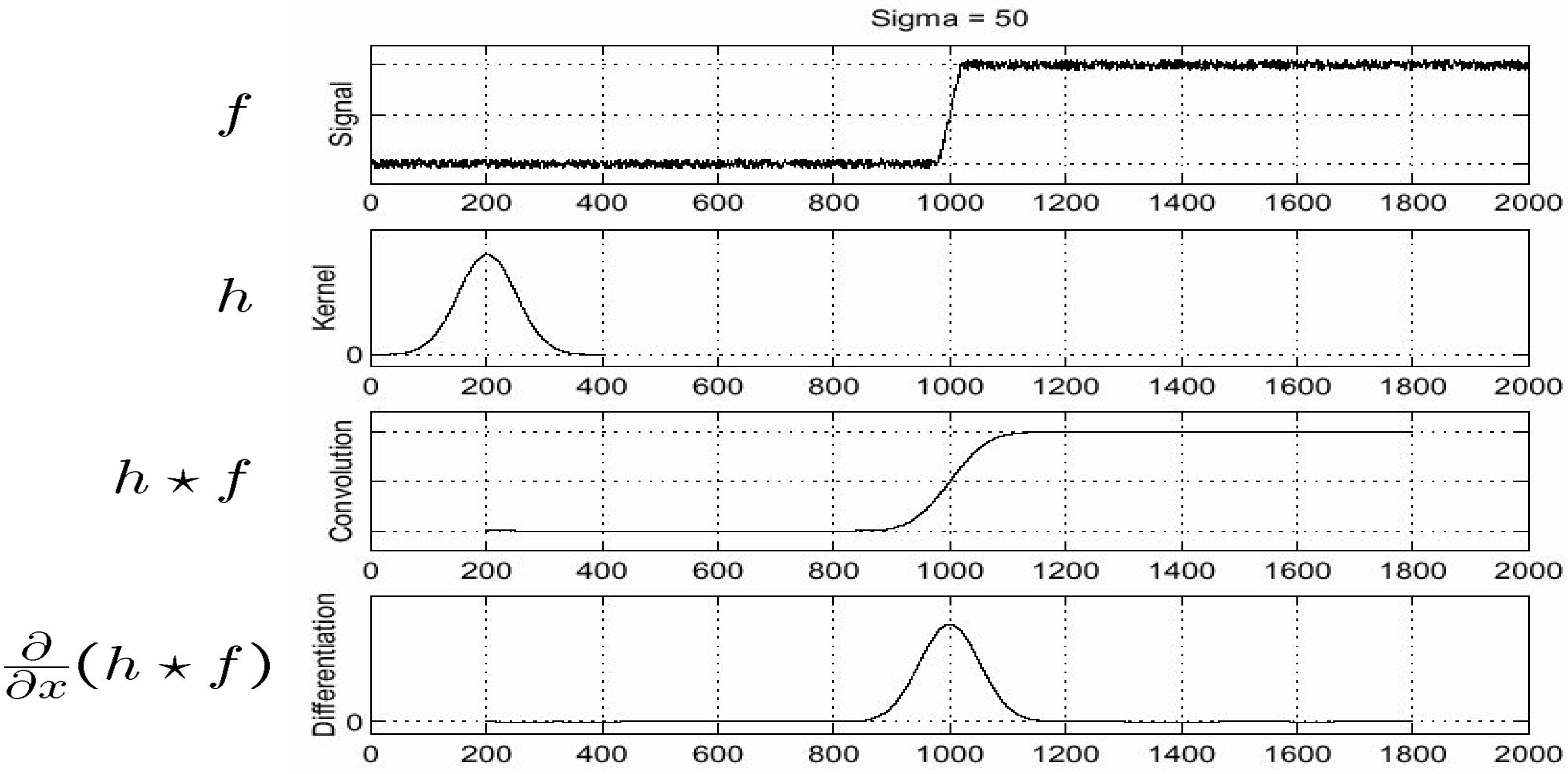


Solution: smooth first



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

- Differentiation is convolution, and convolution is associative: $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$

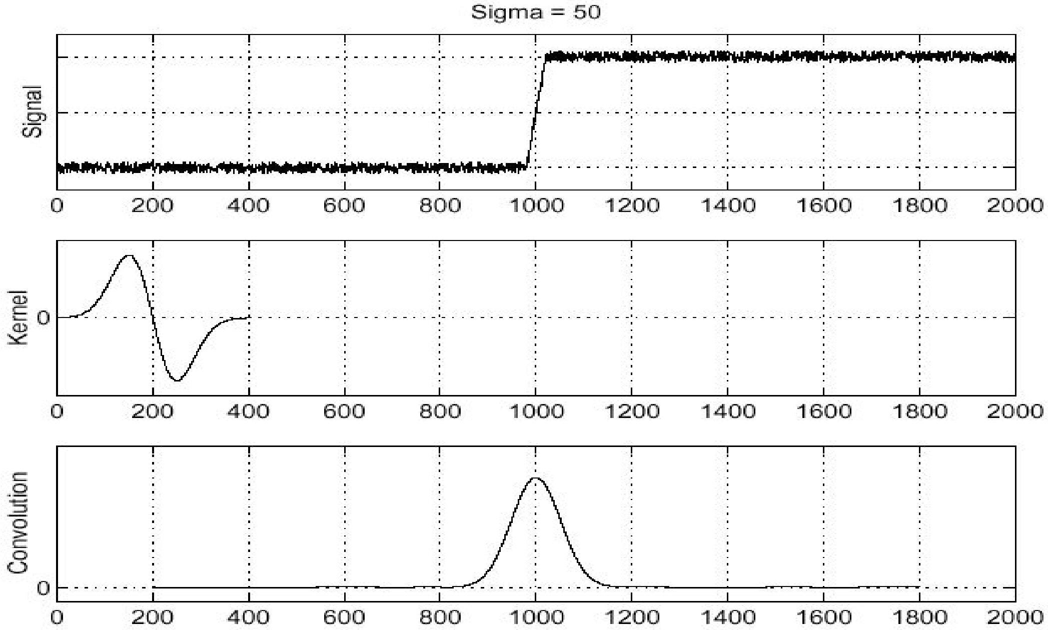
Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative: $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$
- This saves us one operation:

f

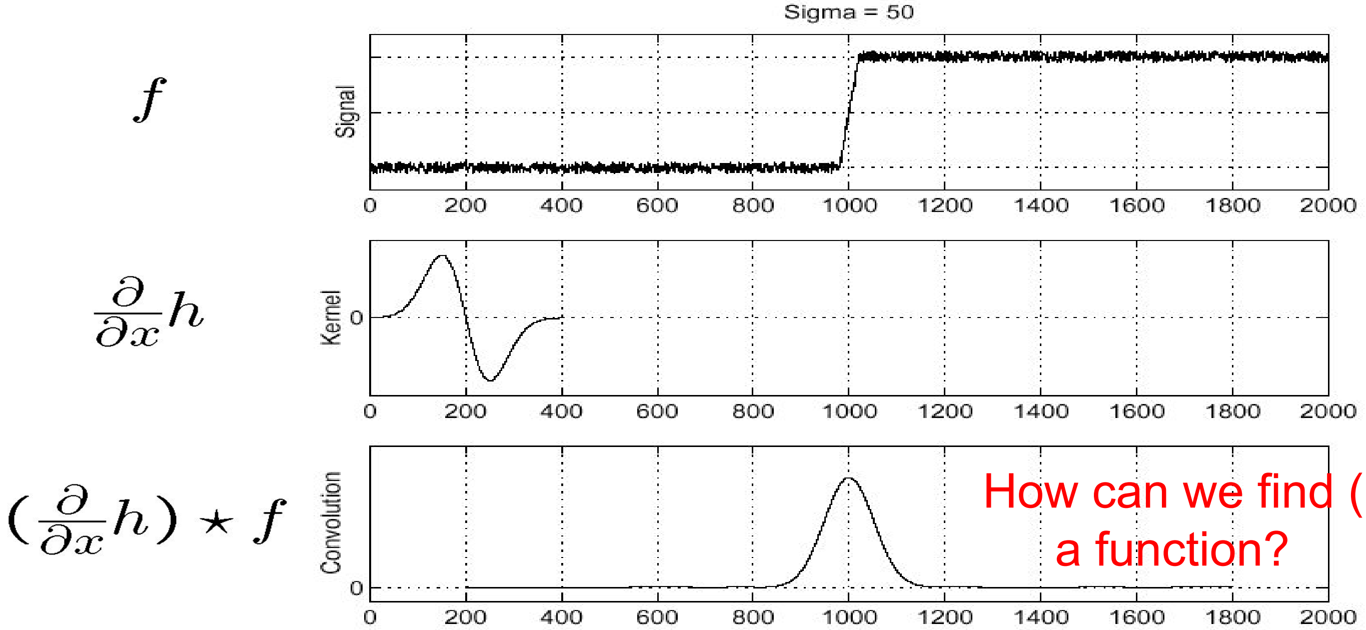
$\frac{\partial}{\partial x}h$

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



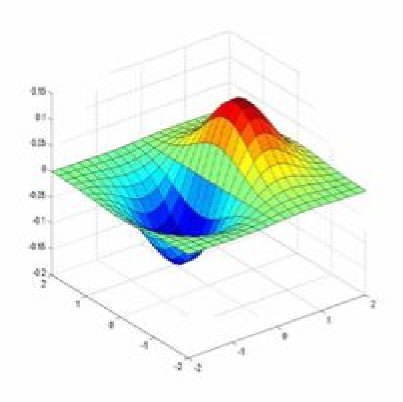
Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative: $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$
- This saves us one operation:

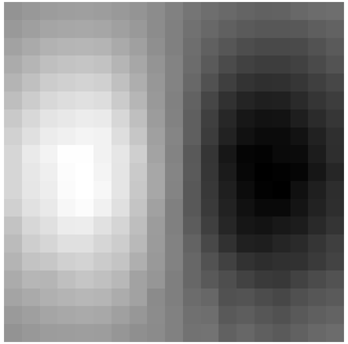


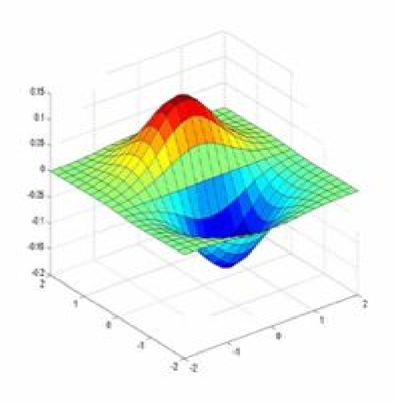
How can we find (local) maxima of a function?

Remember: Derivative of Gaussian filter

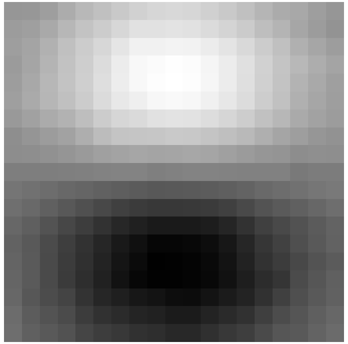


x-direction





y-direction



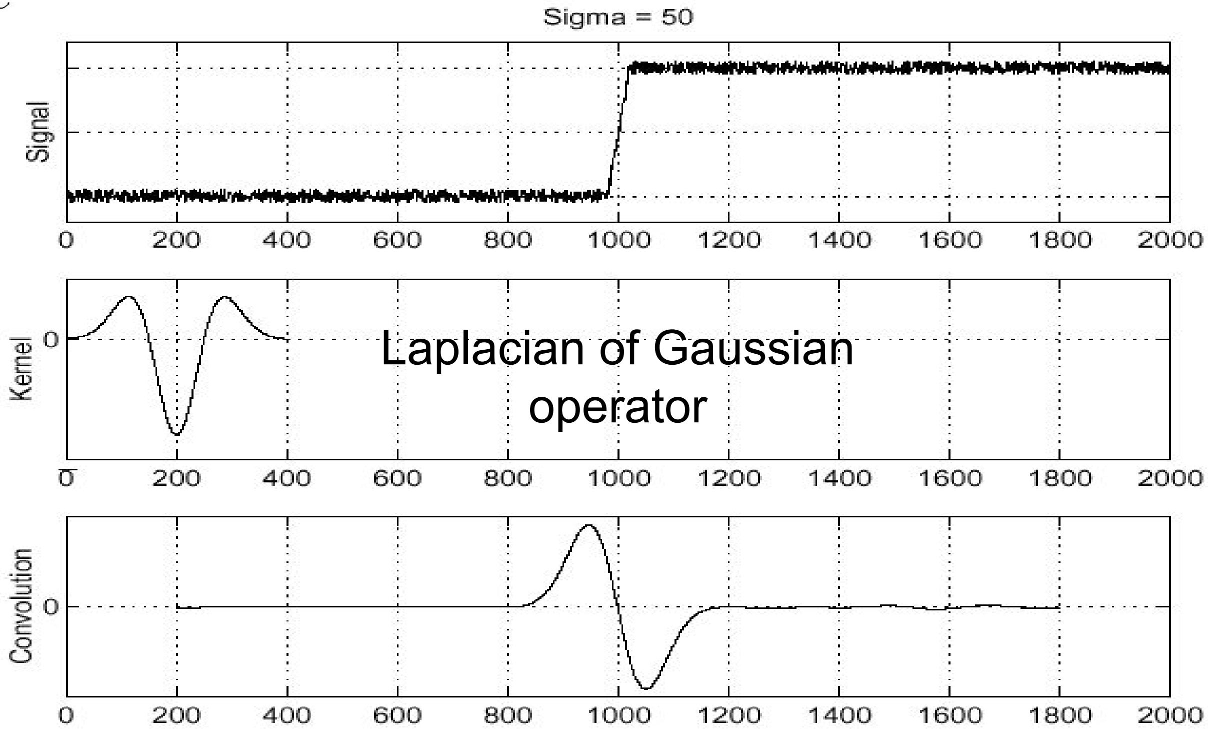
Laplacian of Gaussian

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

f

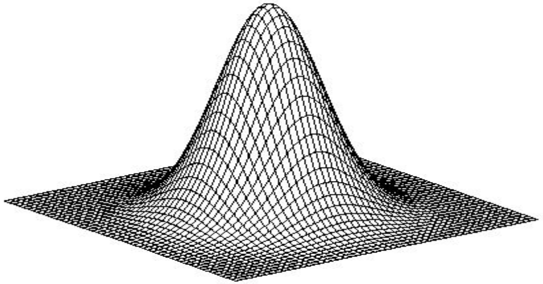
$\frac{\partial^2}{\partial x^2} h$

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



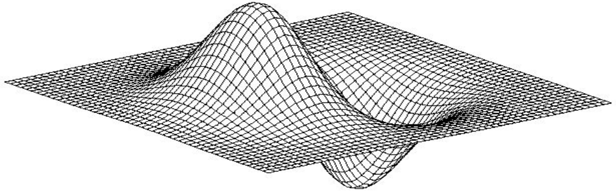
Where is the edge? Zero-crossings of bottom graph

2D edge detection filters



Gaussian

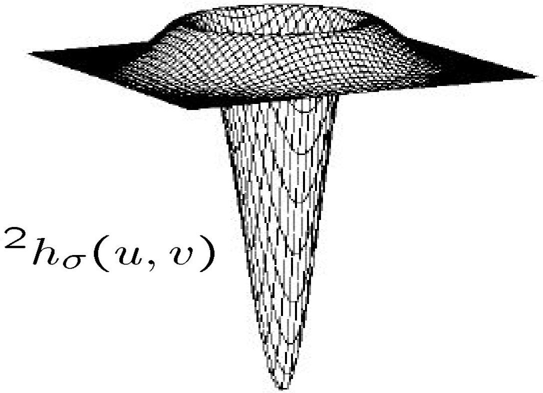
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

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

Laplacian of Gaussian



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

∇^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)

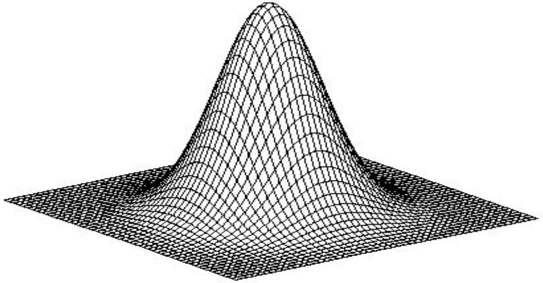
Edge detection by subtraction

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

Why does this work?

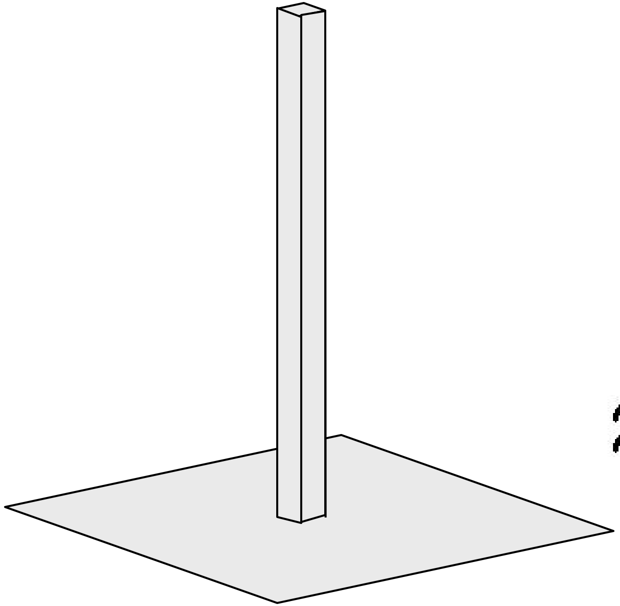
filter demo

Gaussian - image filter



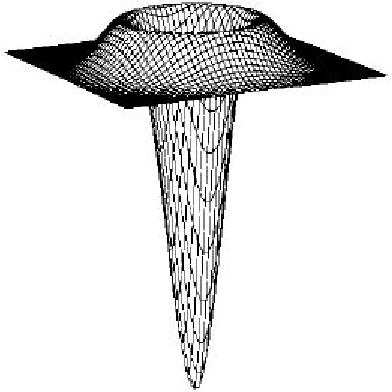
Gaussian

$$-$$



delta function

$$\approx$$



Laplacian of Gaussian

The Canny edge detector

This is probably the most widely used edge detector in computer vision

original image (Lena)

J. Canny, A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

The Canny edge detector

norm of the gradient

The Canny edge detector

thresholding

Get Orientation at Each Pixel

thresholding

Get Orientation at Each Pixel

theta = atan2(-gy, gx)

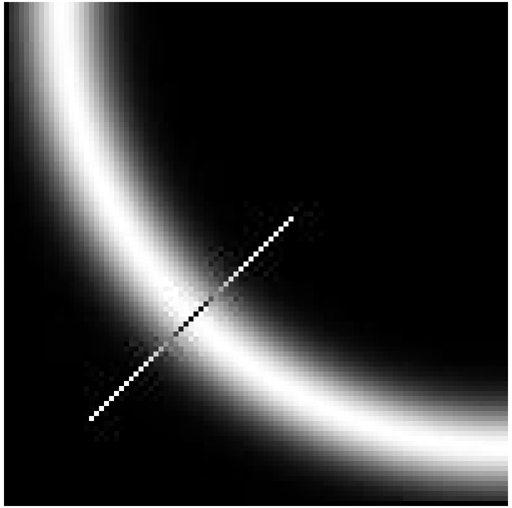
thresholding

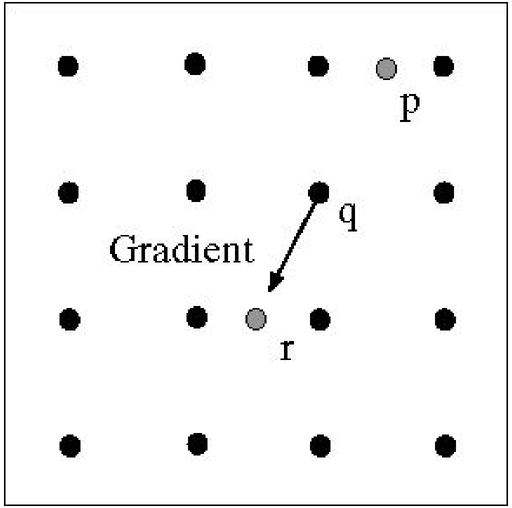
The Canny edge detector

thinning

(non-maximum suppression)

Non-maximum suppression

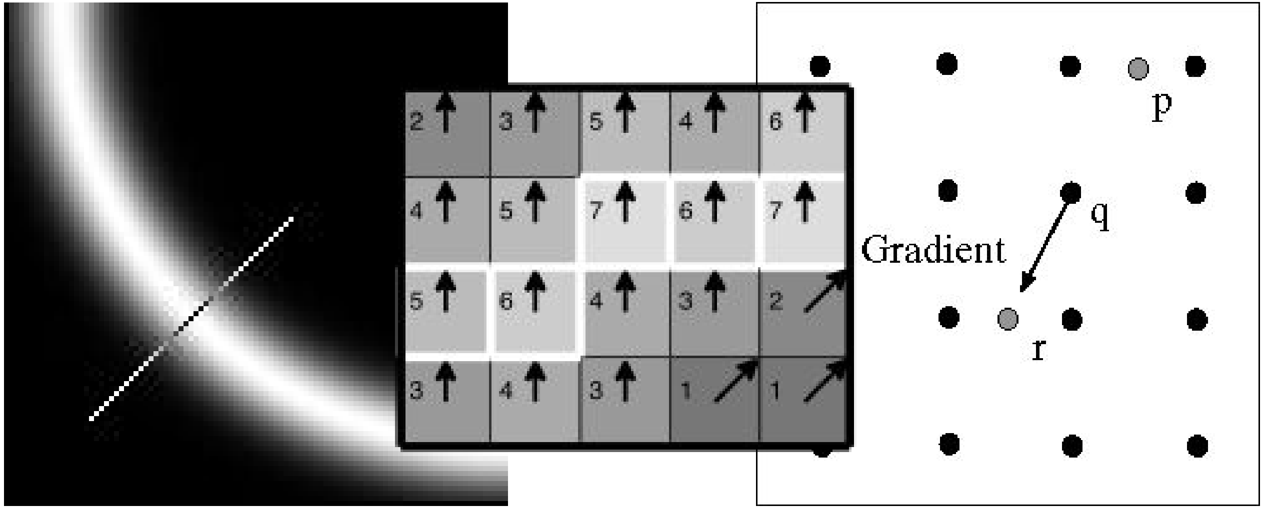




Check if pixel is local maximum along gradient direction

- requires checking interpolated pixels p and r

Non-maximum suppression



Check if pixel is local maximum along gradient direction

- requires checking interpolated pixels p and r

Canny Edges

Effect of σ (Gaussian kernel spread/size)


original


Canny with $\sigma = 1$


Canny with $\sigma = 2$

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

An edge is not a line...

An edge is not a line...

How can we detect **lines**?

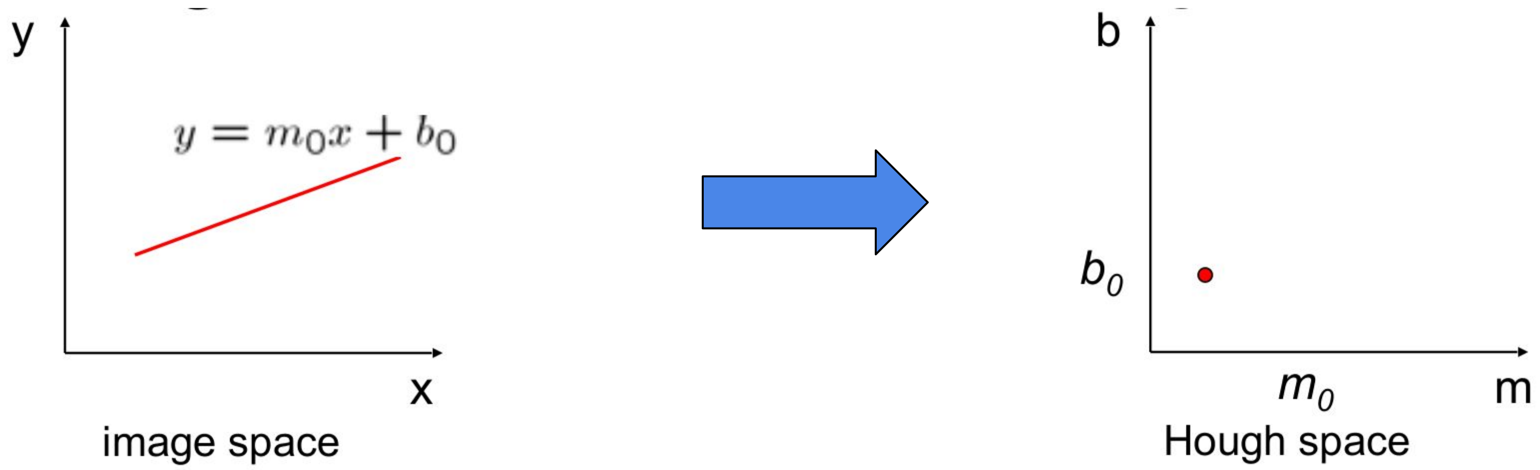
Finding lines in an image

- Option 1:
  - Search for the line at every possible position/orientation
  - What is the cost of this operation?

Finding lines in an image

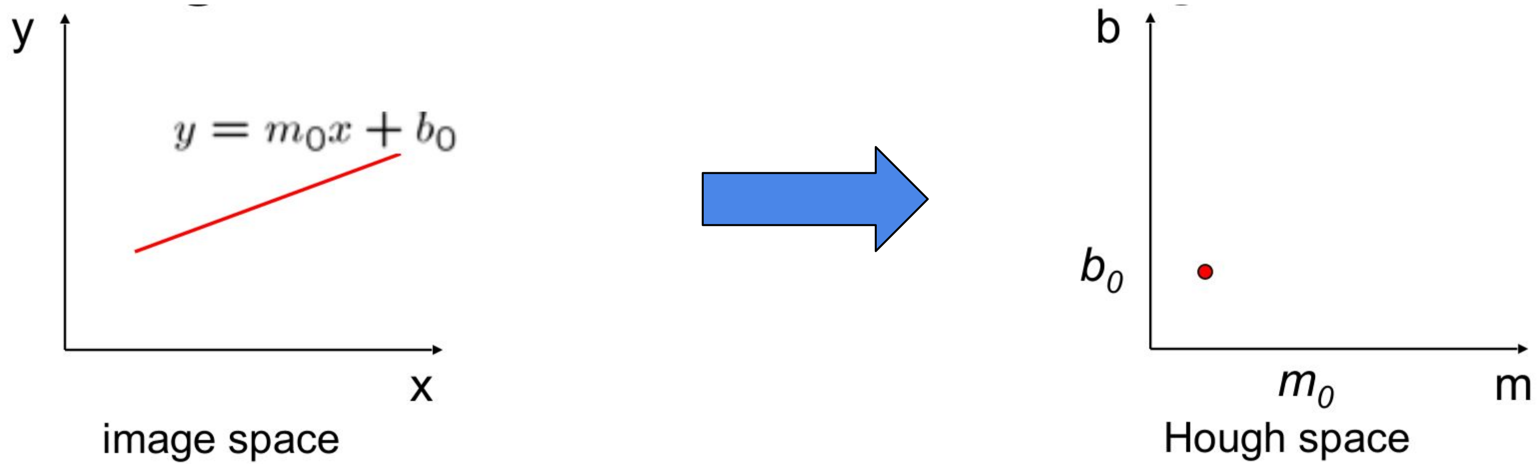
- Option 1:
    - Search for the line at every possible position/orientation
    - What is the cost of this operation?
- Option 2:
    - Use a voting scheme: hough transform

Finding lines in an image



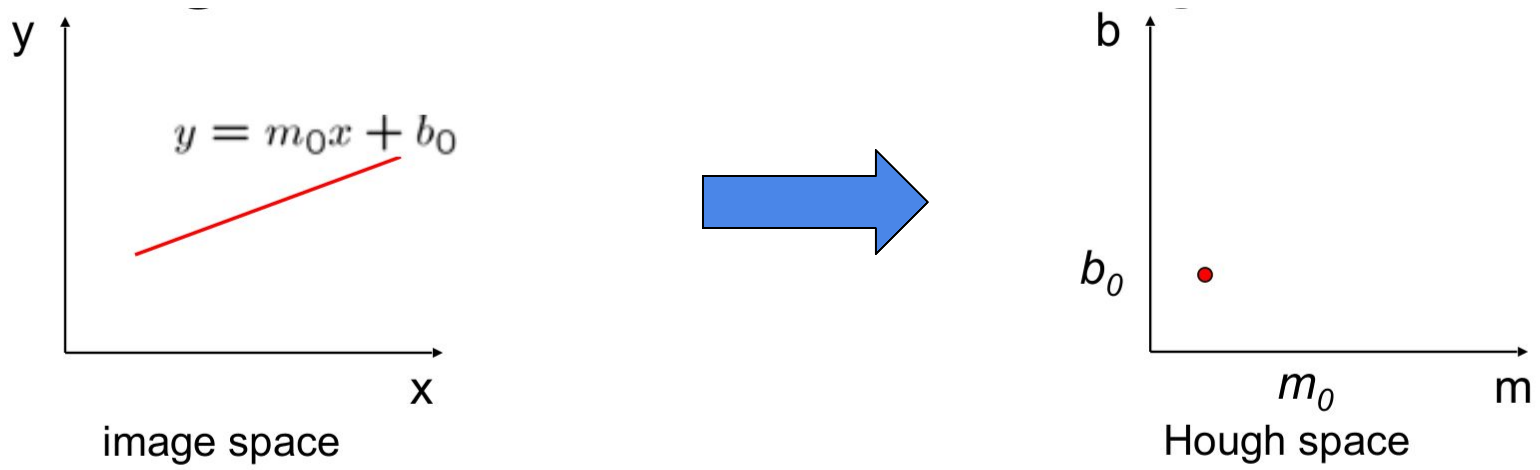
- Connection between image (x,y) and Hough (m,b) spaces
  - A line in the image corresponds to a point in Hough space

Finding lines in an image



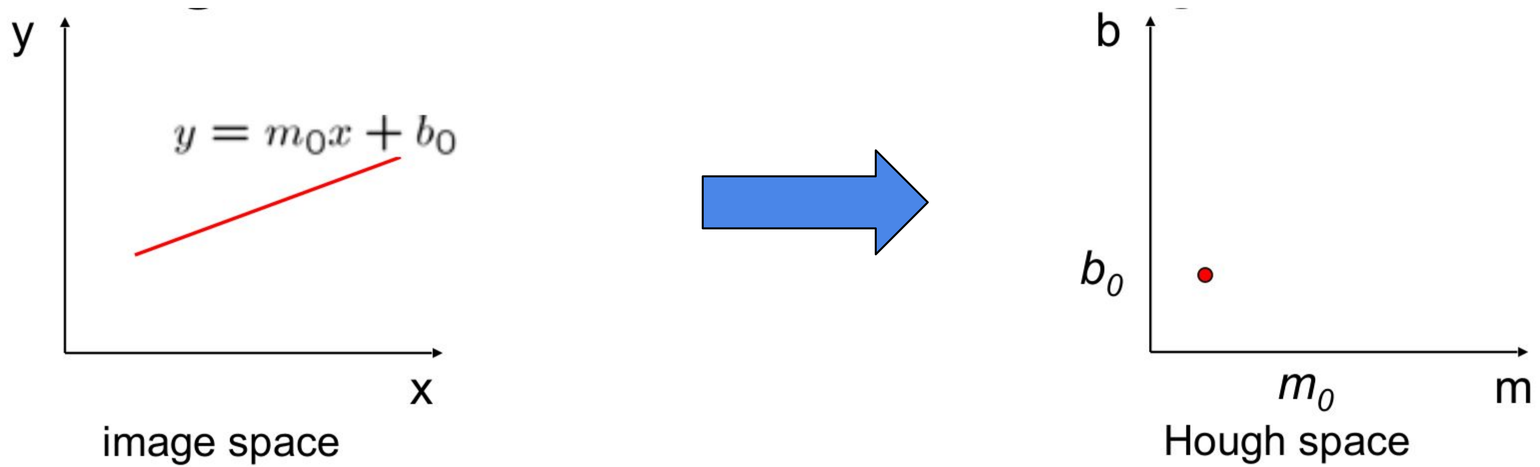
- Connection between image (x,y) and Hough (m,b) spaces
  - A line in the image corresponds to a point in Hough space
  - To go from image space to Hough space:
    - given a set of points (x,y), find all (m,b) such that y = mx+ b

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?

Finding lines in an image



- Connection between image (x,y) and Hough (m,b) spaces
  - A <u>line</u> in the image corresponds to a <u>point</u> 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

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

  - d is the perpendicular distance from the line to the origin
  - θ is the angle

Hough transform algorithm

- Basic Hough transform algorithm
  - Initialize H[d, θ] = 0
  - for each edge point I[x,y] in the image

    for θ = 0 to 180

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

    H[d,θ] += 1
  - Find the value(s) of (d,θ) where H[d,θ] is maximum
  - The detected line in the image is given by $d = x cos\theta + y sin\theta$

Hough transform algorithm

- Basic Hough transform algorithm
  - Initialize H[d, θ] = 0
  - for each edge point I[x,y] in the image

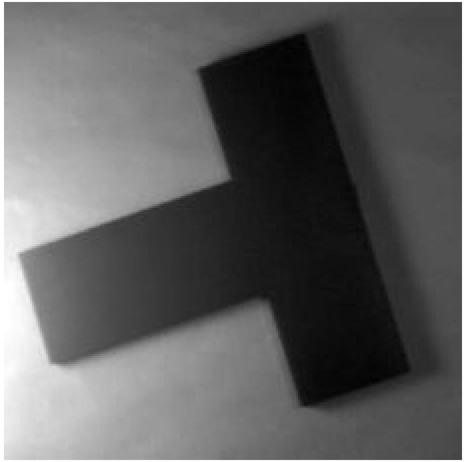
    for θ = 0 to 180

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

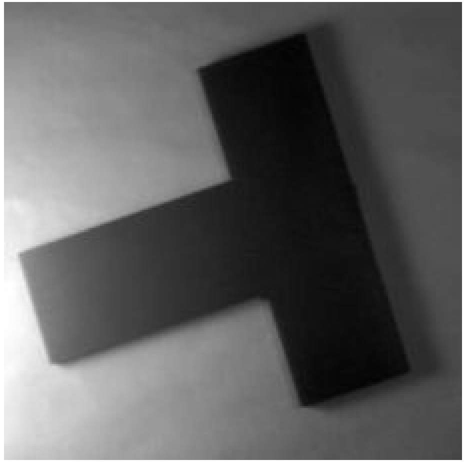
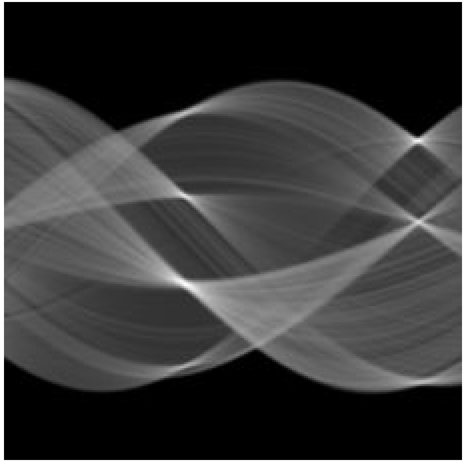
    H[d,θ] += 1
  - Find the value(s) of (d,θ) where H[d,θ] is maximum
  - The detected line in the image is given by $d = x cos\theta + y sin\theta$

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

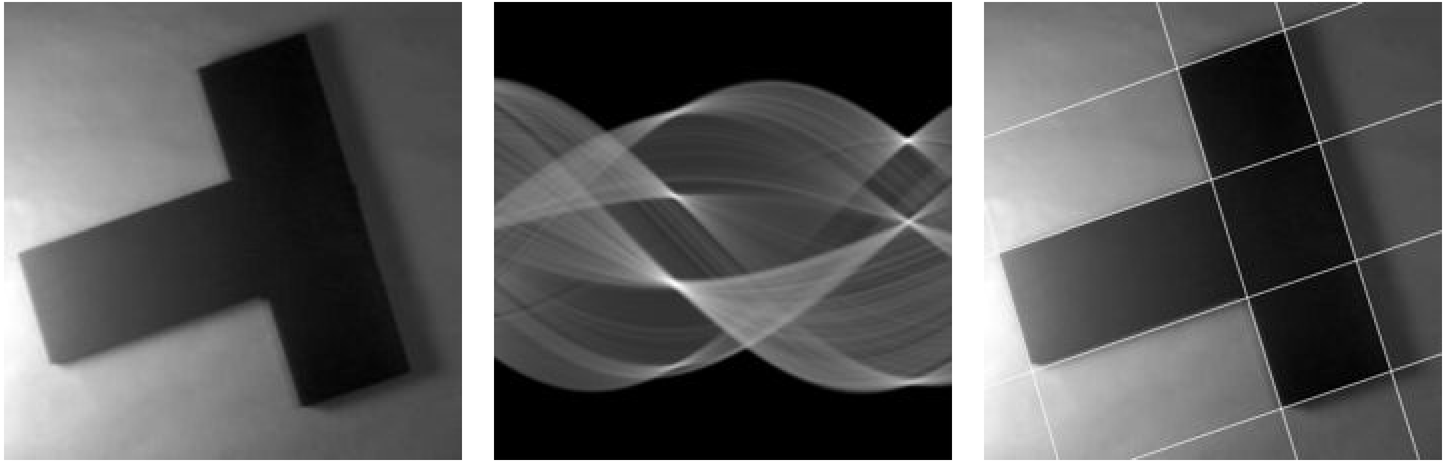
Hough transform algorithm



Hough transform algorithm

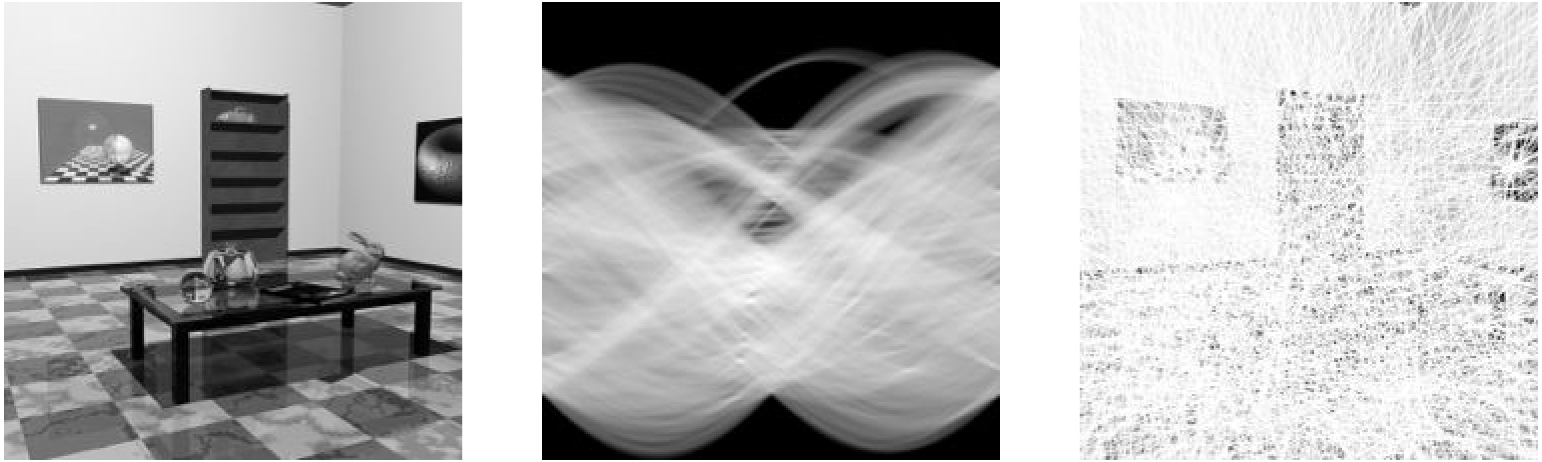
 

Hough transform algorithm



http://www.cs.utah.edu/~vpegorar/courses/cs7966/

Hough transform algorithm



http://www.cs.utah.edu/~vpegorar/courses/cs7966/

Extensions

- Extension 1: Use the image gradient
  - same
  - 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
  - same
  - 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, How?