CITS 4402 Computer Vision

Ajmal Mian

Lecture 05
Edge Detection & Hough Transform

Objectives of this lecture

- To understand what are edges
- To study the Sobel operator for gradient extraction
- To learn the Canny edge detection algorithm
- To study the general concept of the Hough transform
- To use the Hough transform for line and circle detection

Significance of Edges in Images

- There is strong evidence that some form of data compression occurs at an early stage in the human visual system
- Research suggests that one form of this compression involves finding edges and other information-high features in images
- From edges we can construct a line drawing of the scene. This results in a huge data compression, but the information content remains very high
- When we look at a sketch of a scene we do not feel that we are missing a great deal of information

What is an Edge?

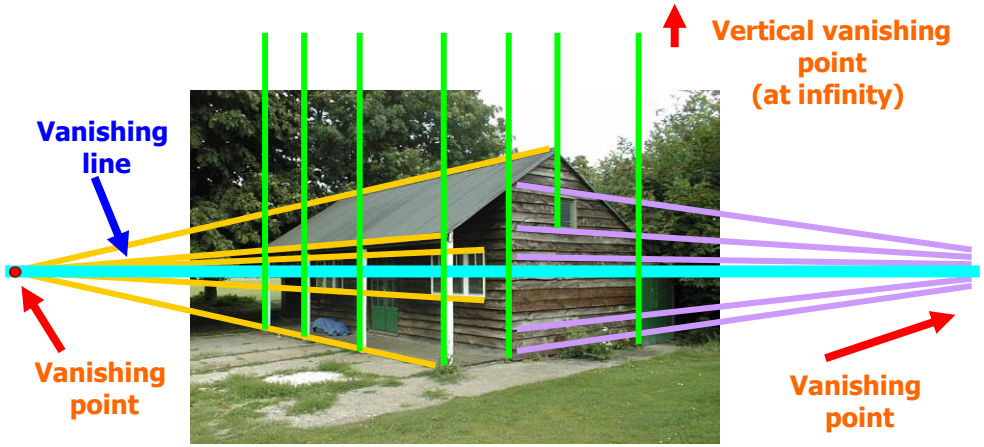
- This is not well defined but the classical approach is to define edges as being step discontinuities in the image
- Edge detection then becomes a task of finding local maxima in the first derivative or zero-crossing of the 2nd derivative
- This idea was first suggested by David Marr at MIT in the late 70's.

Edge Detection

- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)

Why do we Care about Edges?

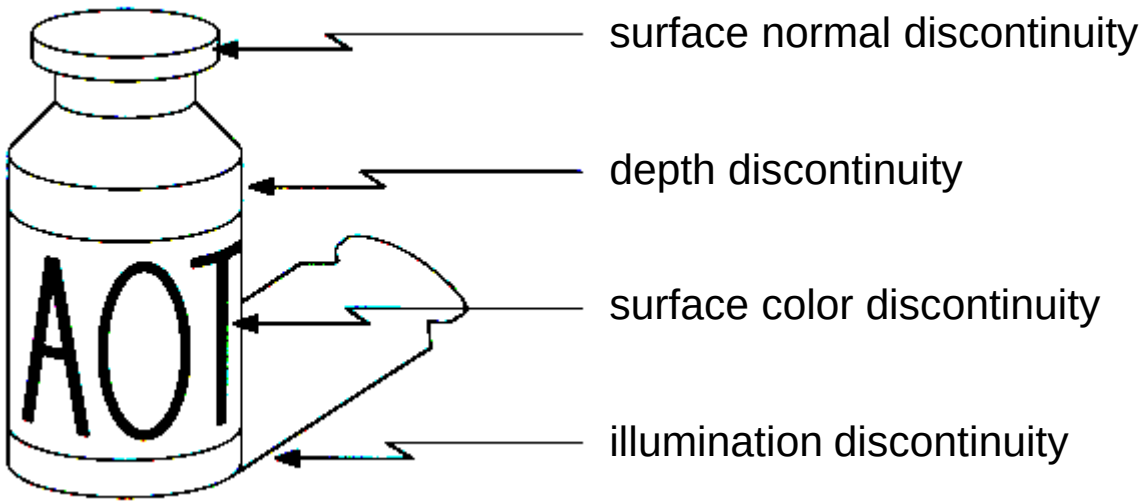
Extract information, recognize objects

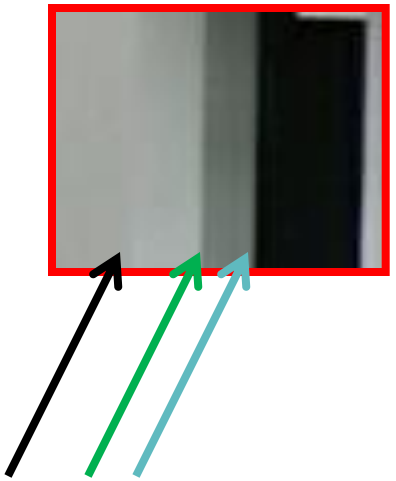


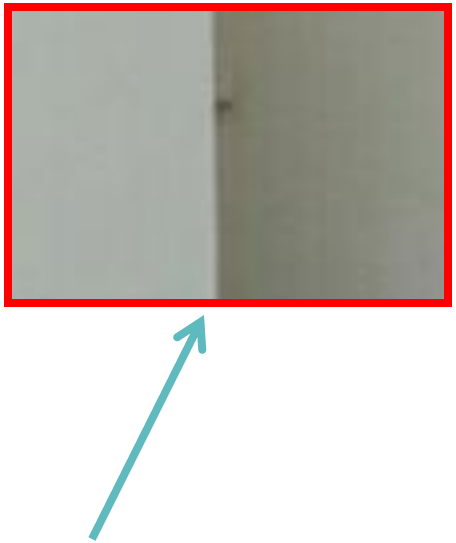
Recover geometry and viewpoint

Origin of Edges

- Edges are caused by a variety of factors

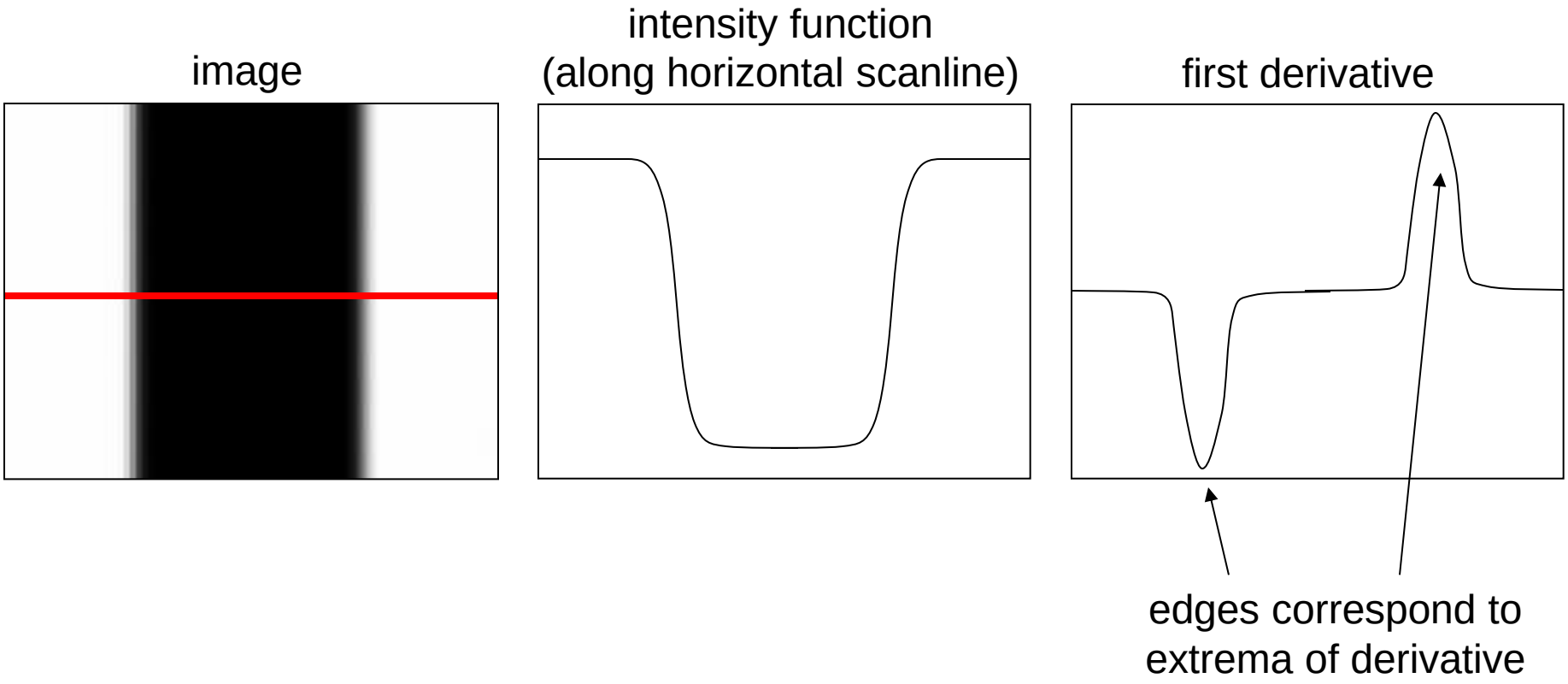






Characterizing Edges

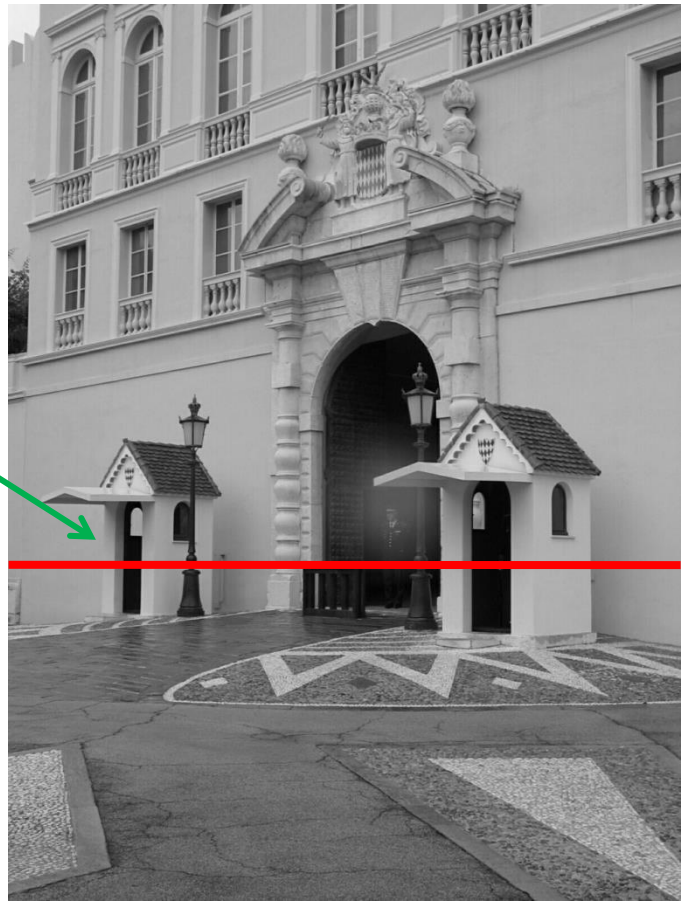
- An edge is a place of rapid change in the image intensity function

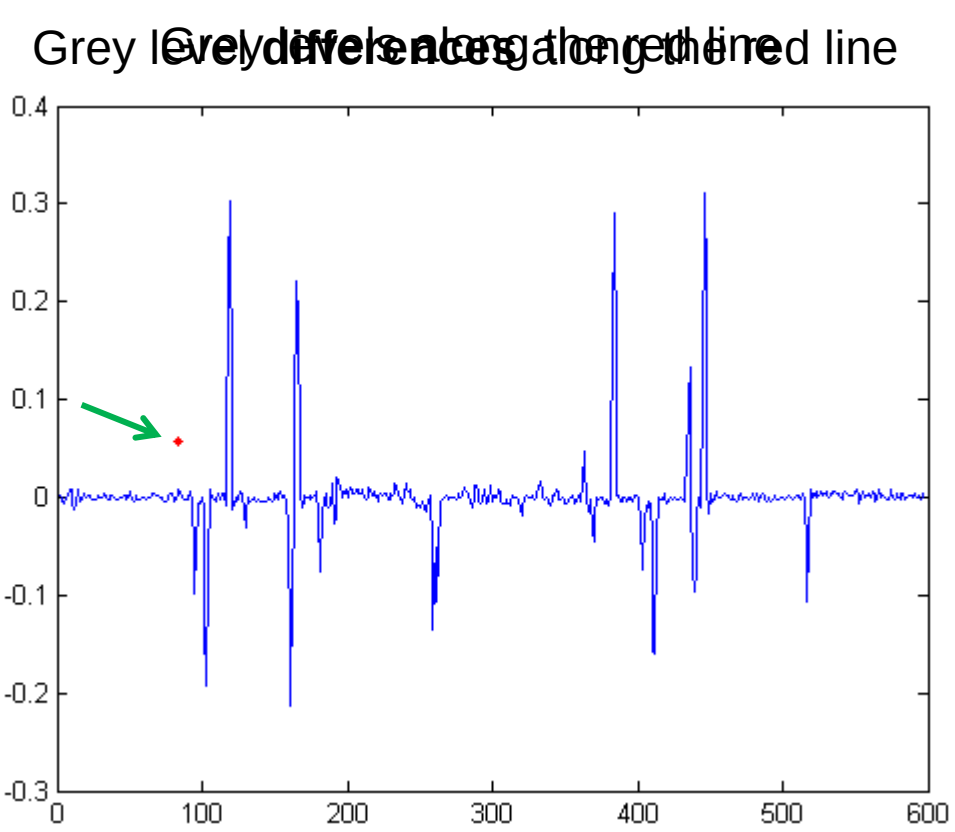


Characterizing Edges (cont.)

- In a discrete image, we calculate the gradient by taking the difference of grey values between adjacent pixels
  - **Forward difference**: $g(x) = f(x+1) - f(x)$
  - **Backward difference**: $g(x) = f(x) - f(x-1)$
  - **Central difference**: $g(x) = \big(f(x+1) - f(x-1)\big)/2$

Intensity Profile





First Derivative Edge Operator

- This is equivalent to convolving with the mask

-1	+1

- In 2D, the gradient of the image function I is given by

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

The **magnitude** is given by $\sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$

The **orientation** is given by $\text{atan2}\left(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x}\right)$

- The simplest gradient operator is the **Roberts Cross Operator**. It uses the two masks:

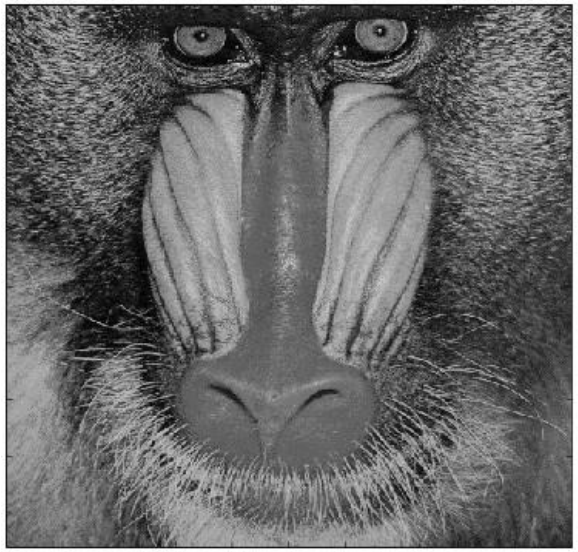
0	1
-1	0

and

1	0
0	-1

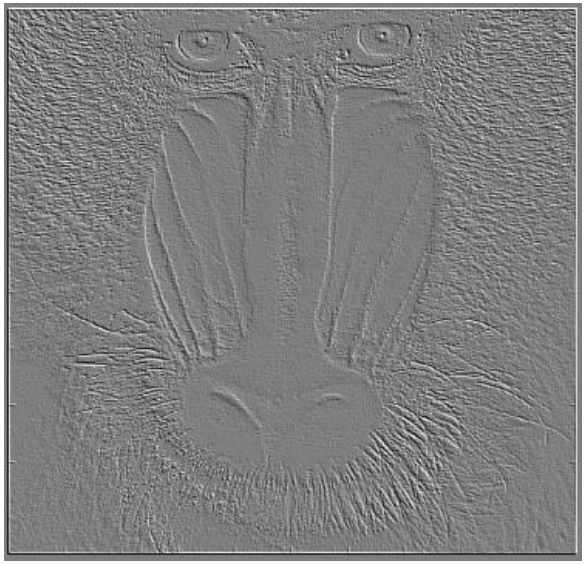
↘ Using the **Roberts Cross** operator…

Input
I



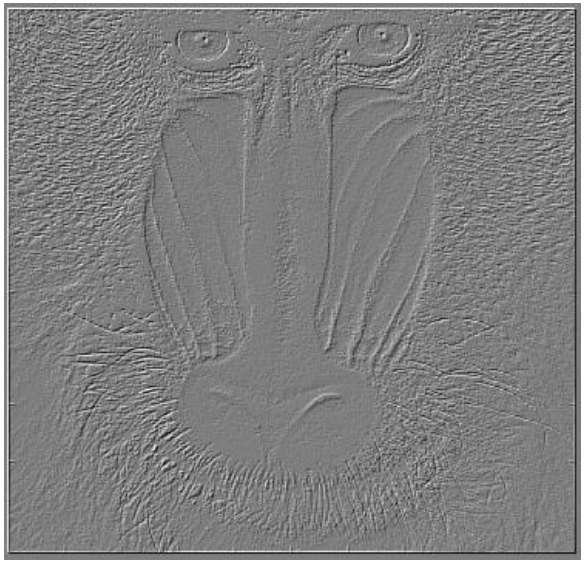
$\frac{\partial I}{\partial x_1}$

(x_1 = 135° line)



$\frac{\partial I}{\partial x_2}$

(x_2 = 45° line)



$$\sqrt{\left(\frac{\partial I}{\partial x_1}\right)^2 + \left(\frac{\partial I}{\partial x_2}\right)^2}$$

First Derivative Edge Operator (cont.)

- Problem determining where to place the result of the convolution:
- use an odd number of pixels

-1	0	1

- A 3 × 3 approximation for $\frac{\partial I}{\partial x}$ is given by the mask

-1	0	1
-1	0	1
-1	0	1

This is the **Prewitt** operator

- The well-known **Sobel** operator is a variation on this theme, giving more emphasis to the centre cell

for $\frac{\partial I}{\partial x}$

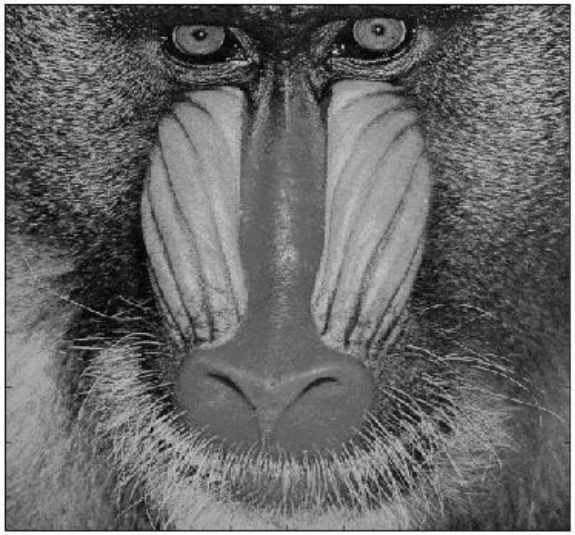
-1	0	1
-2	0	2
-1	0	1

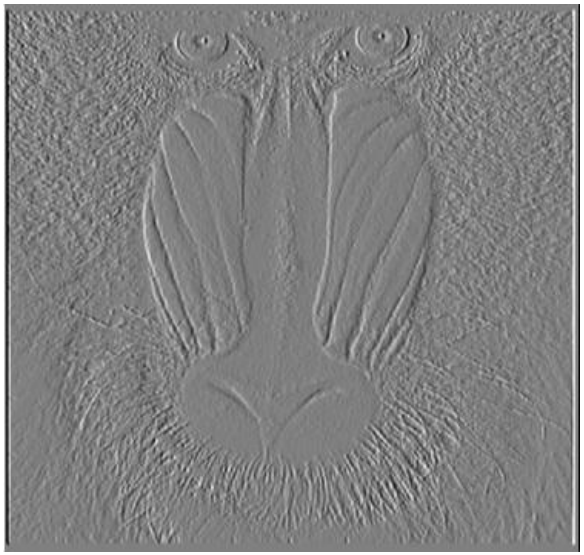
for $\frac{\partial I}{\partial y}$

1	2	1
0	0	0
-1	-2	-1

where *I* is the greyscale image

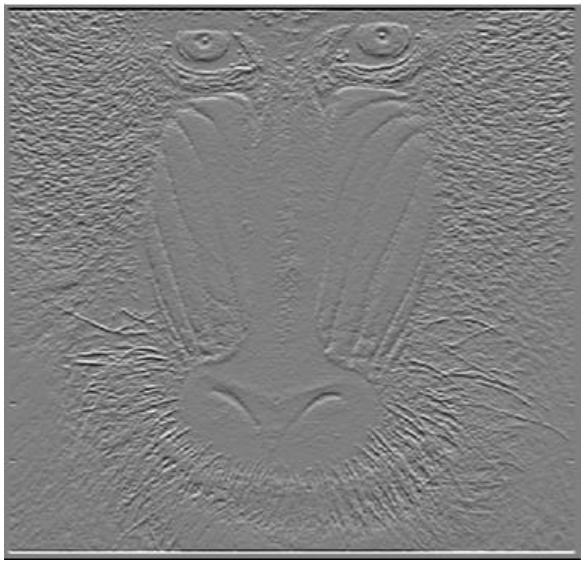
Input
I

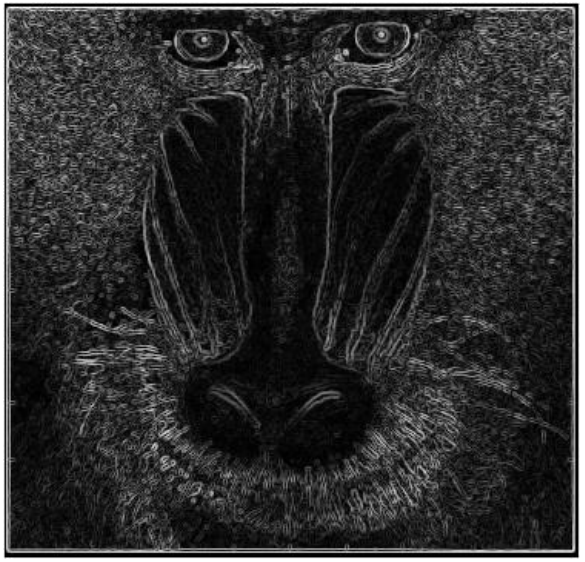




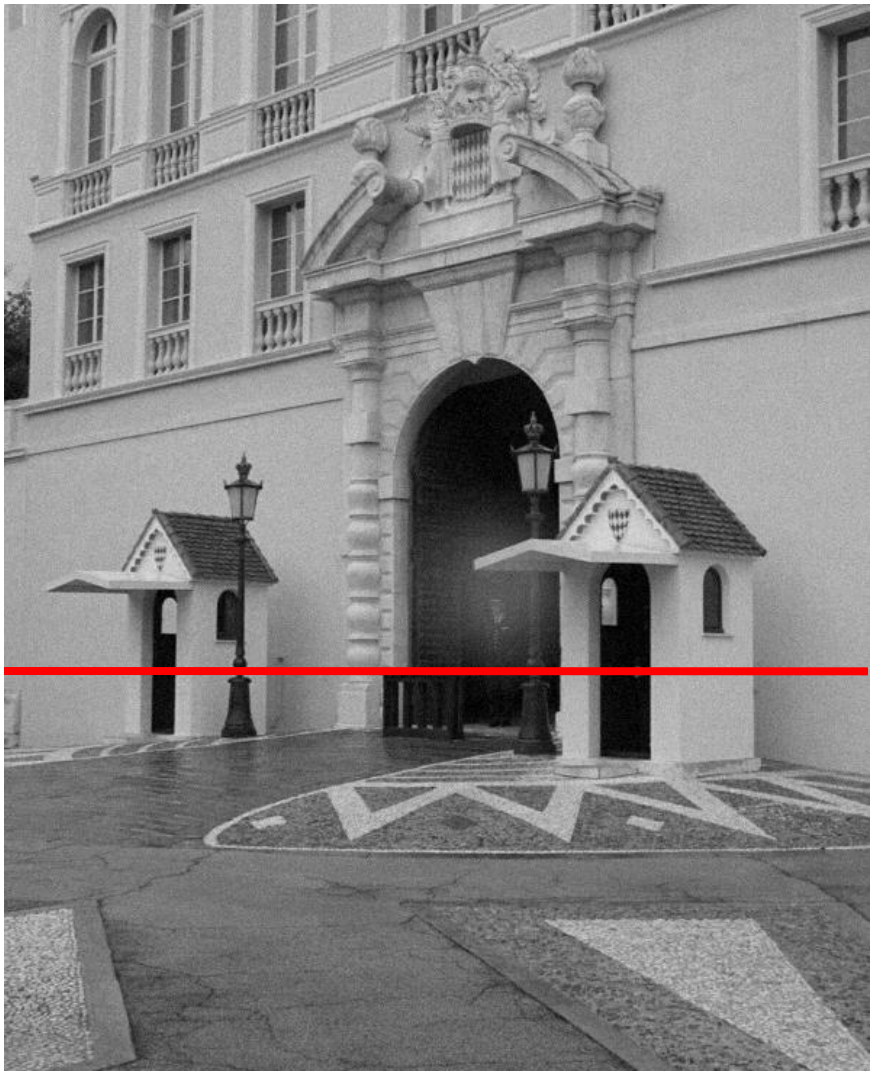
$\frac{\partial I}{\partial x}$

$\frac{\partial I}{\partial y}$

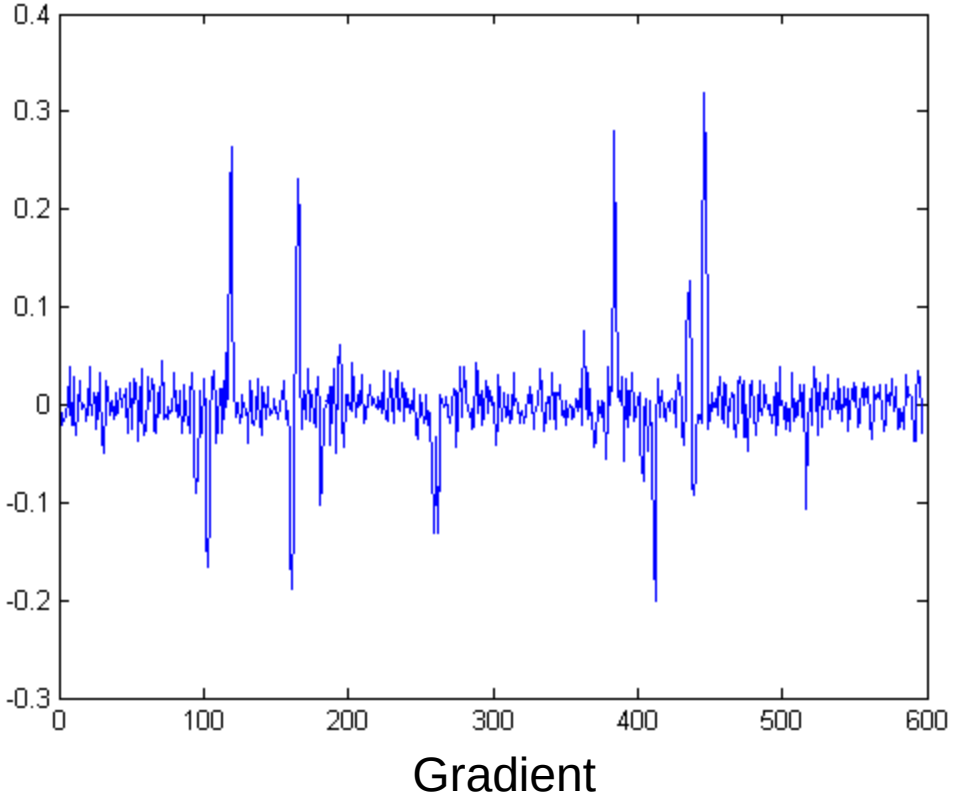




$$\sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

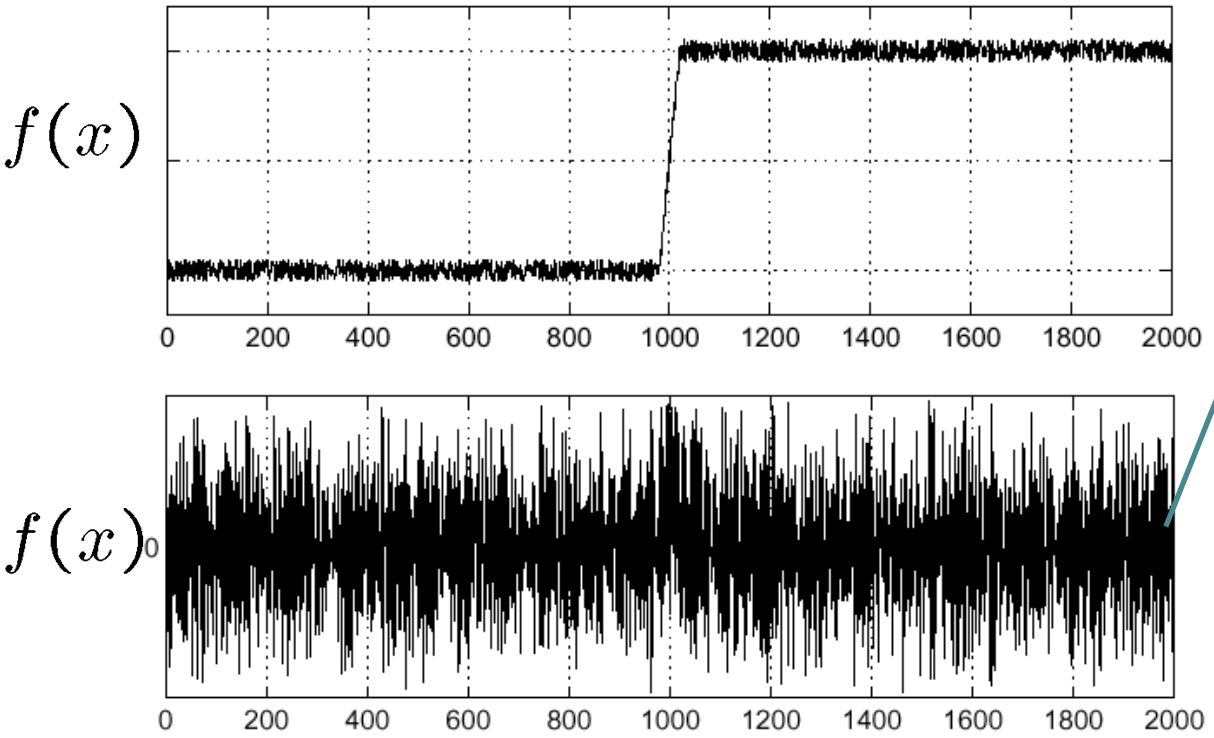


The main problem we have to deal with is noise!



Effects of Noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

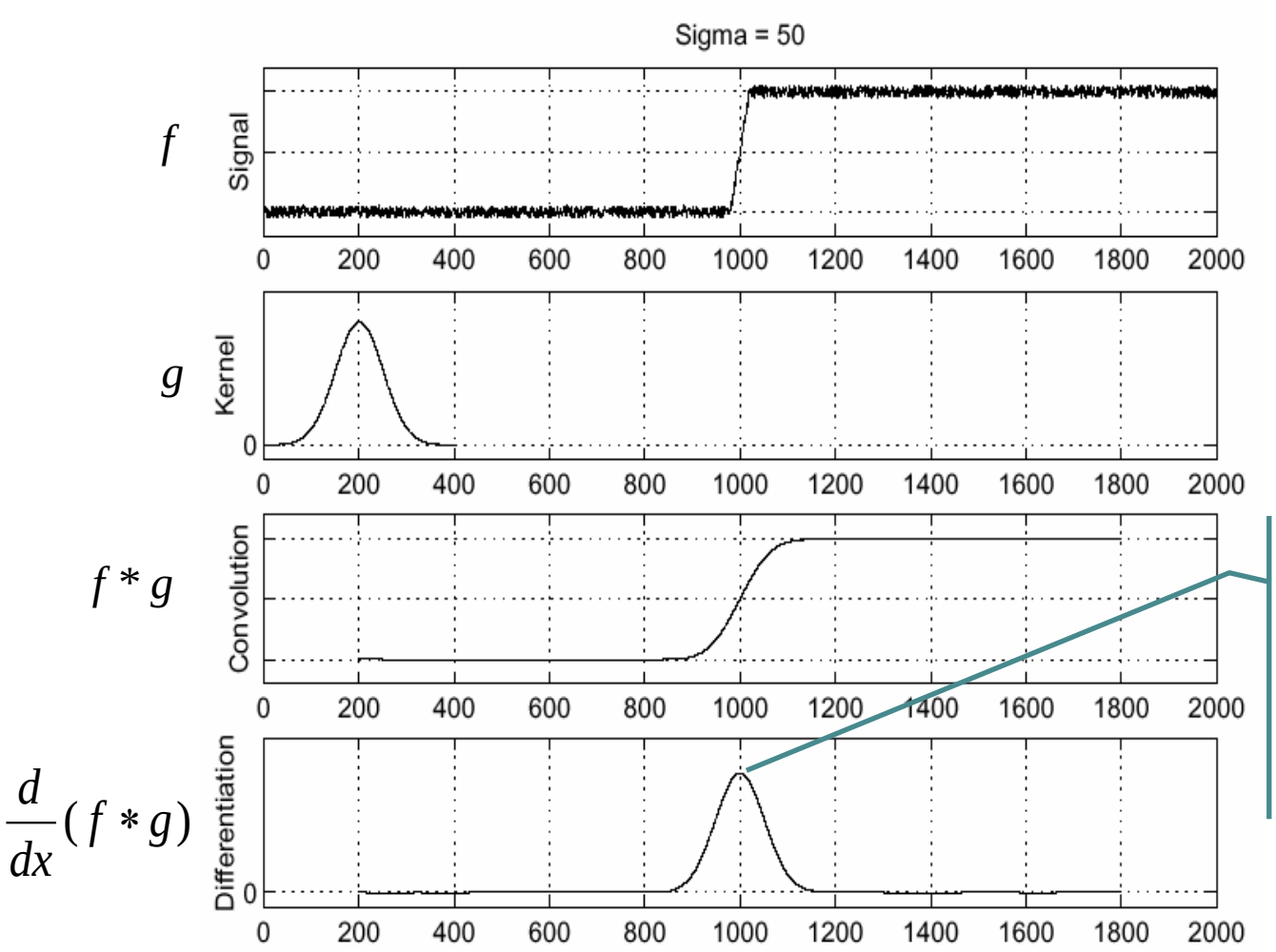


Where is the edge?

Effects of Noise

- Difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response

- What can we do about it?



Edges correspond to the peaks in the derivative of the Gaussian

Solution: Smoothen First

1. Convolve the image with a Gaussian mask to smooth it
2. Calculate derivatives of the smoothed image

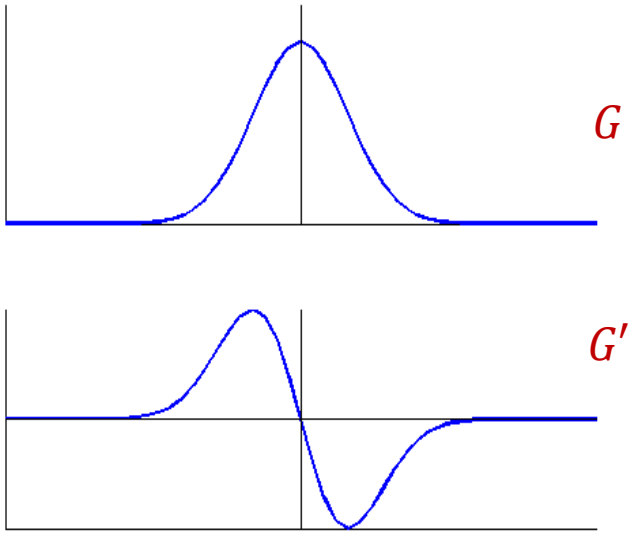
Both of these operations can be combined into one computationally efficient operation.

If we calculate the derivative of the Gaussian mask and convolve the image with this mask, we compute the derivative of the smoothed image in one hit.

That is, $(I \otimes G)' = I \otimes G'$

derivative

image Gaussian mask



The Prewitt and Sobel operators can be considered to be crude discrete approximations of G' over 3 pixels. Use of larger masks allow broader Gaussians to be represented.

Derivative Theorem of Convolution

- Differentiation is convolution, and convolution is associative:

LHS: 2 filters: g and d/dx

$$\frac{d}{dx}(f \otimes g) = f \otimes \frac{d}{dx}(g)$$

RHS: 1 filter: $d/dx(g)$

- Differentiation is convolution, and convolution is associative:

$$\frac{d}{dx}(f \otimes g) = f \otimes \frac{d}{dx}(g)$$

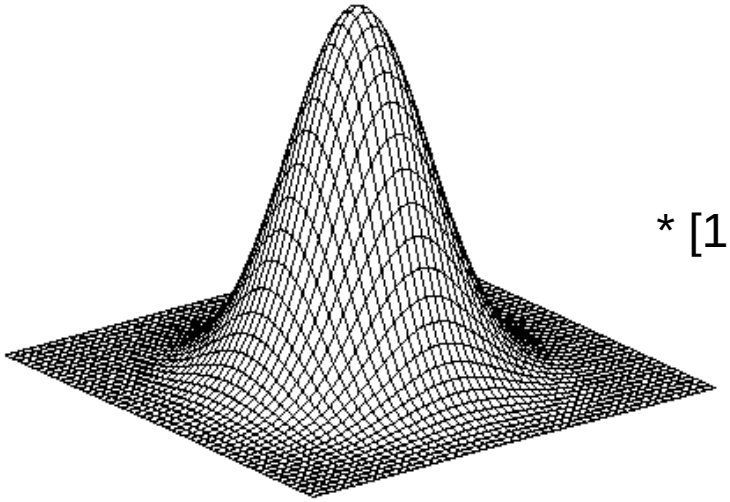
f

$$\frac{d}{dx}(g)$$

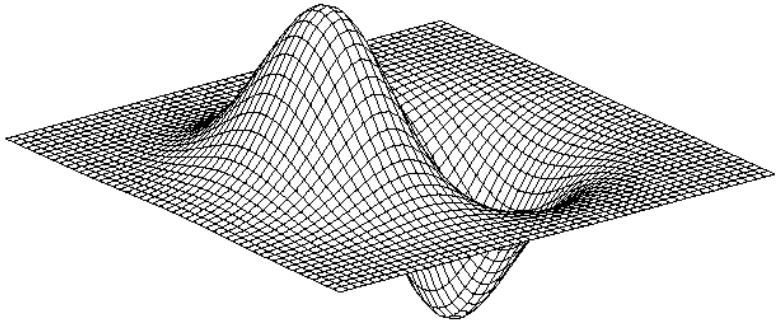
$$f \otimes \frac{d}{dx}(g)$$

Source: S. Seitz

Derivative of a Gaussian Filter

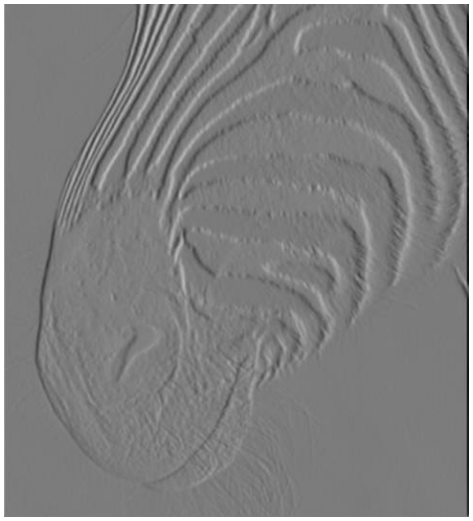


* [1 -1] =

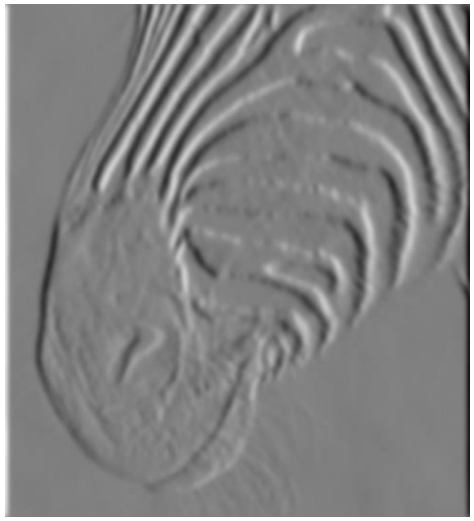


Tradeoff between Smoothing and Localization

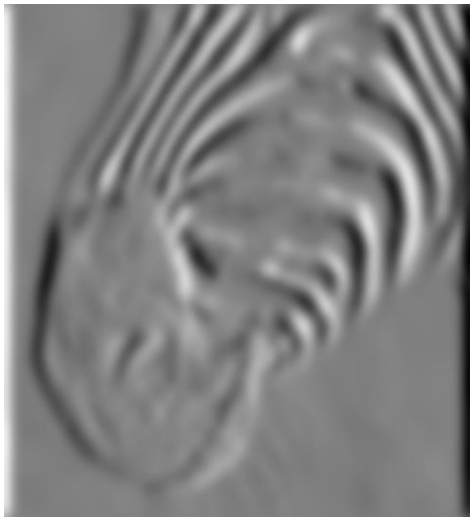
- Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.



1 pixel



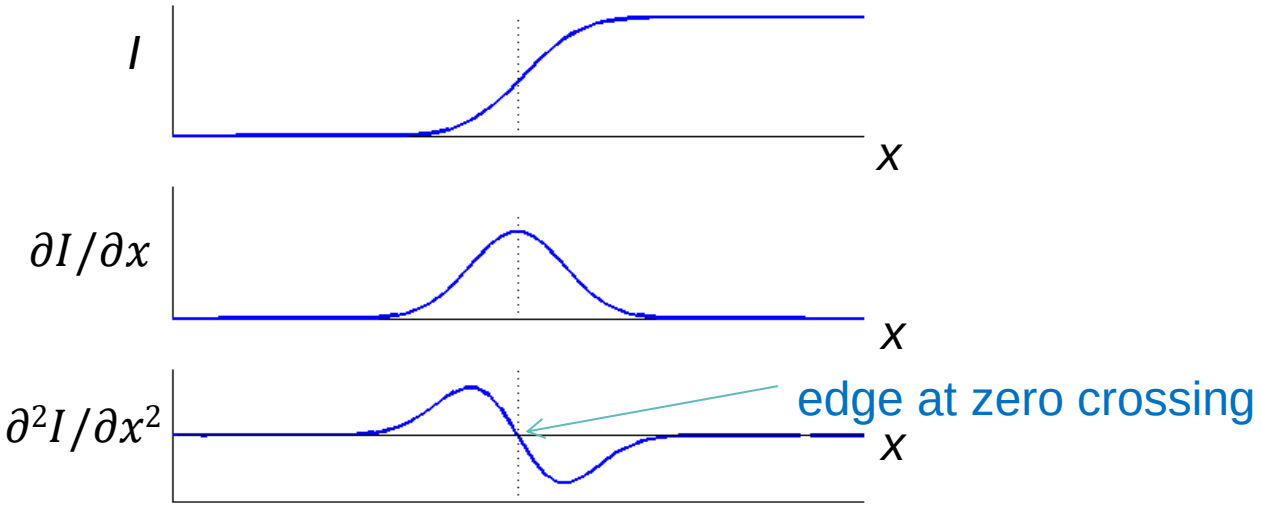
3 pixels



7 pixels

Second Derivative Edge Operators

- A maximum of the 1st derivative will occur at a zero crossing of the 2nd derivative

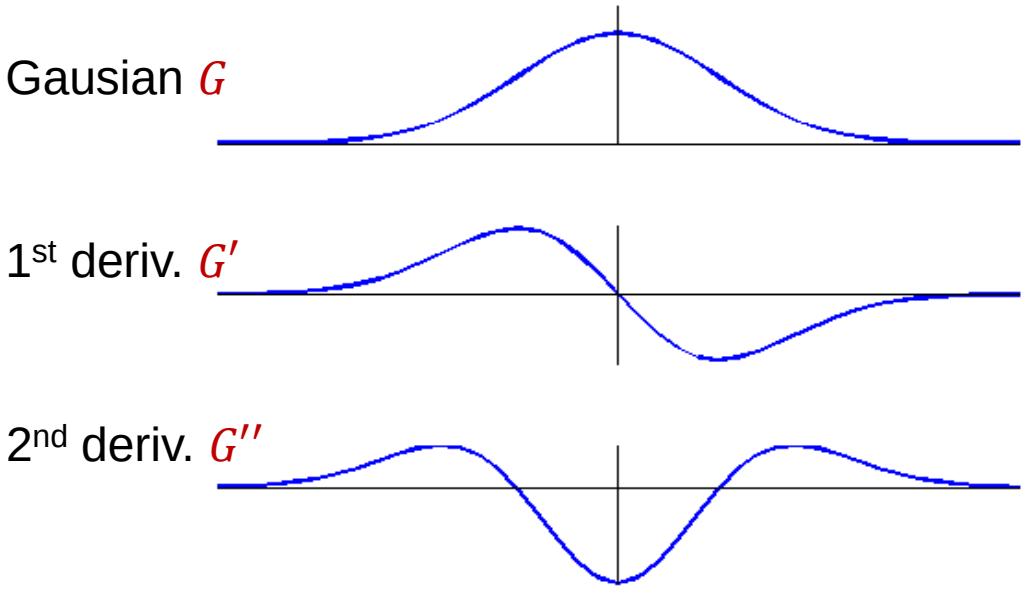


- To get horizontal and vertical edges we look at 2nd derivative in both x and y directions. This is the Laplacian of I :

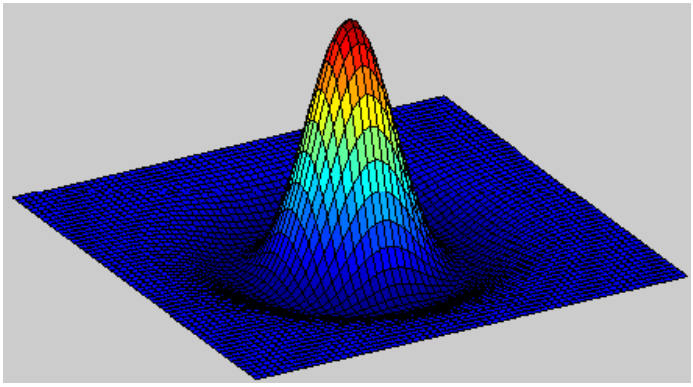
$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Second Derivative Edge Operators (cont.)

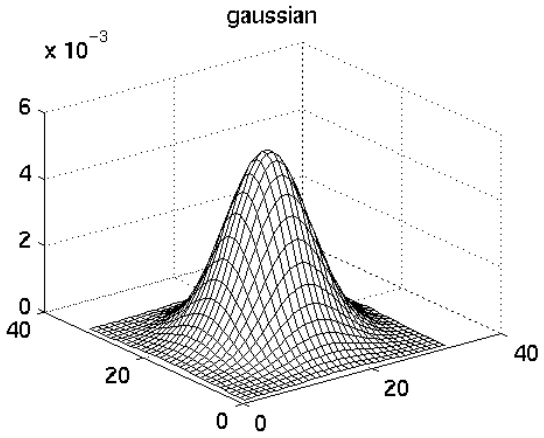
- The Laplacian is linear and rotationally symmetric
- Again we need to consider about smoothing the image with a Gaussian filter. It is well known that $\nabla^2(G \otimes I) = \nabla^2 G \otimes I$
- $\nabla^2 G$ is known as the **Laplacian of the Gaussian** operator

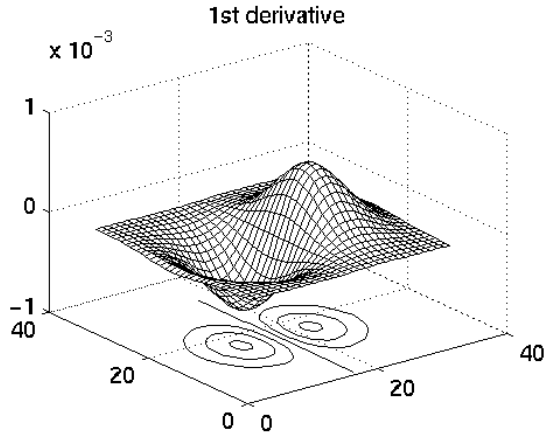


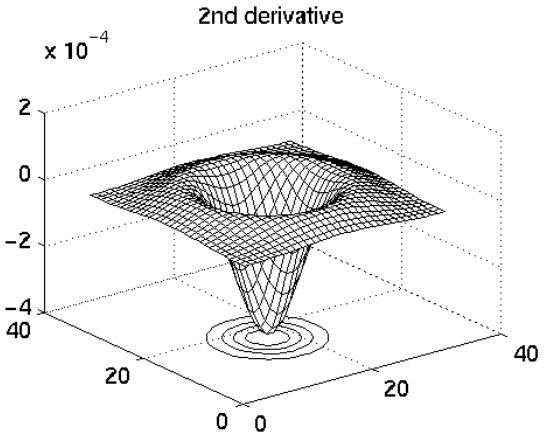
Surface profile of G"
(the surface has been flipped upward for visualization)



Second Derivative Edge Operators (cont.)

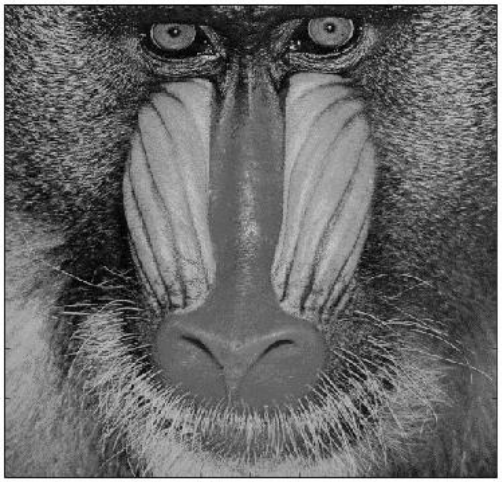


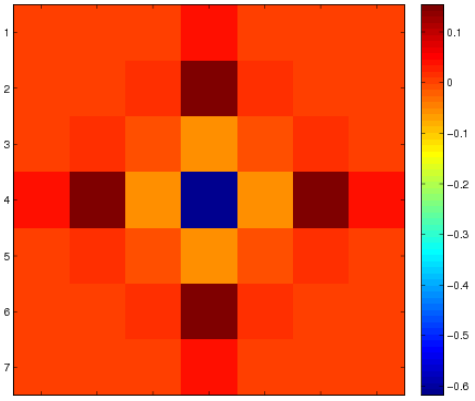




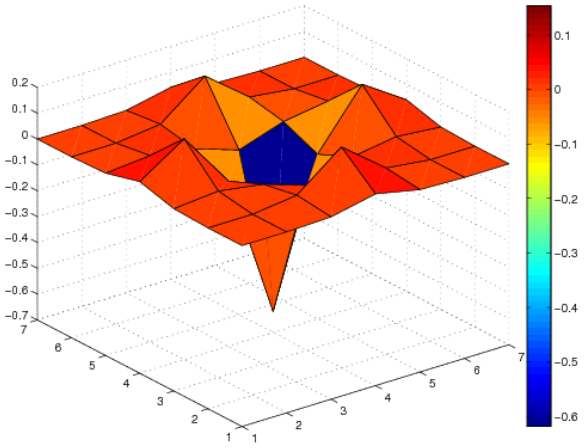
Surface profiles of G, G', and G''

LoG Operator – An Example





A 7 x 7 LoG operator

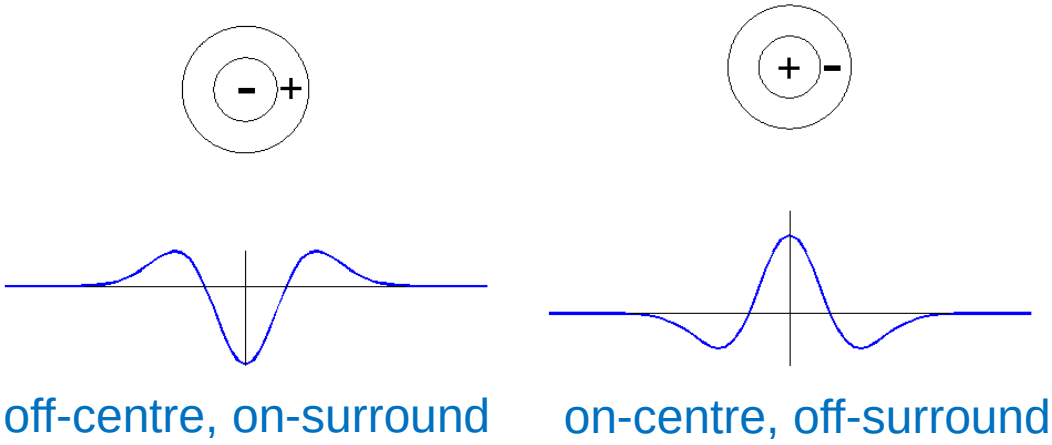


Convolution output

(binary edge map. Dark pixels denote edge pixels)

Second Derivative Edge Operators (cont.)

- This operator became widely used, because…
  1. Researchers found receptive fields in animals that behaved like this operator



off-centre, on-surround on-centre, off-surround

  2. The operator was symmetric – you find edges in all orientations
  3. Zero crossings of the 2nd derivative always form closed contours

Second Derivative Edge Operators (cont.)

However there are some problems

1. Being a 2nd derivative operator, the influence of noise is considerable
2. There are many other types of receptive fields in the eye, so this is not the whole story
3. Always generating closed contours is not realistic
4. It will mark edges at some locations which are not edges

Designing an Edge Detector

- **Criteria for a good edge detector:**
  - **Good detection:** the optimal detector should find all real edges, ignoring noise or other artifacts
  - **Good localization**
    - the edges detected must be as close as possible to the true edges
    - the detector must return one point only for each true edge point

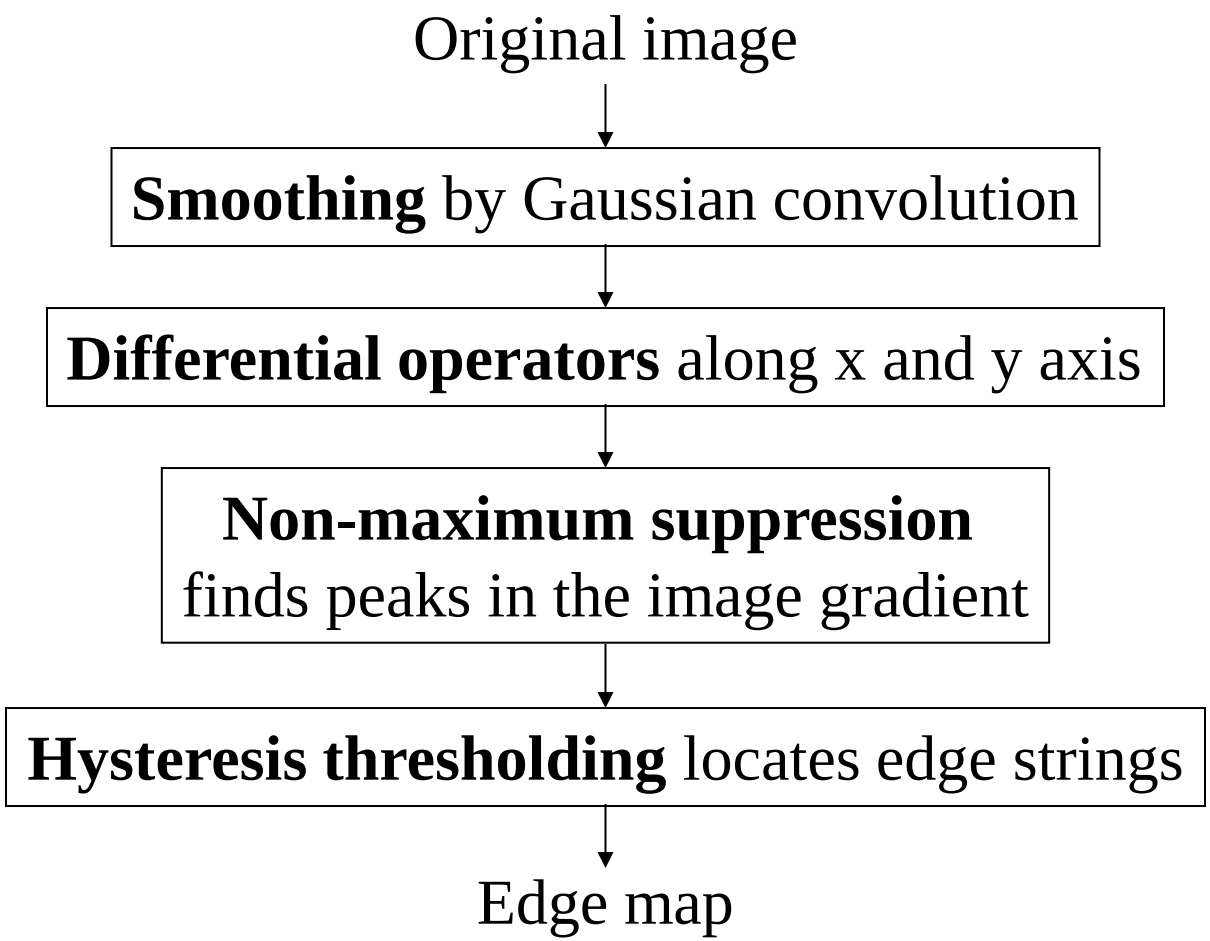
↘ **Cues of edge detection**

- Differences in color, intensity, or texture across the boundary
- Continuity and closure
- High-level knowledge

The Canny Edge Detector

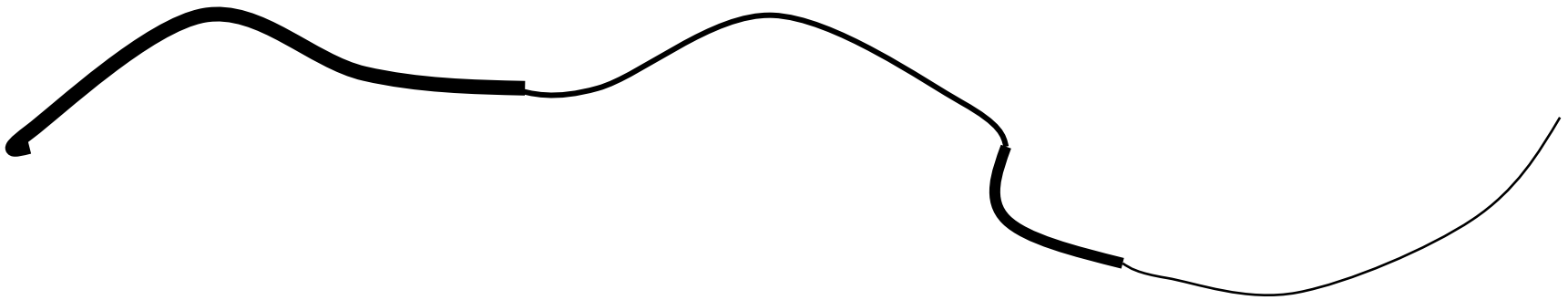
- This is the current 'standard' and is used widely
- Devised by John Canny as part of his Masters thesis at MIT in 1983
- He treated edge detection as an **optimization problem** – maximize the detection of step edges in the presence of noise. The optimization criteria are:
  1. Maximize signal to noise ratio – good detection
  2. Good localization of edges – be accurate
  3. Minimize number of responses to a single edge in the presence of noise – avoid false positives
- Canny came up with a function that optimized his 3 criteria that was the sum of 4 exponential terms. However, this function looked very much like the 1st derivative of a Gaussian ! So this is what ended up being used.

The Canny Edge Detector



Hysteresis Thresholding

- Threshold to eliminate ‘insignificant edges’
- This involves using **two threshold values** with one being 2-3 times as large as the other.
- Any pixel with an edge gradient greater than the higher threshold is considered a valid edge point
- Any pixels connected to a valid edge point with a gradient value greater than the lower threshold is also considered a valid edge point.

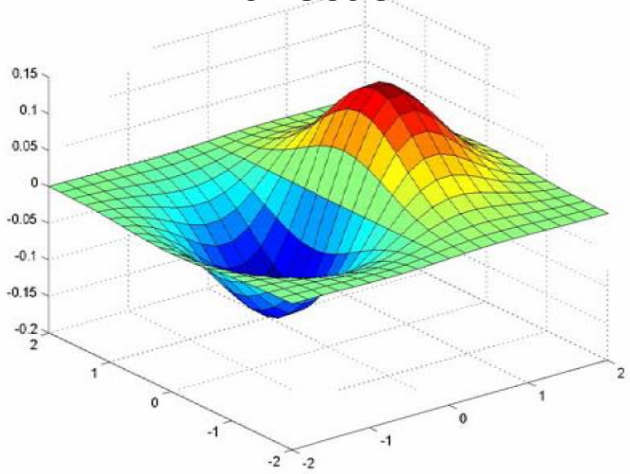


Example

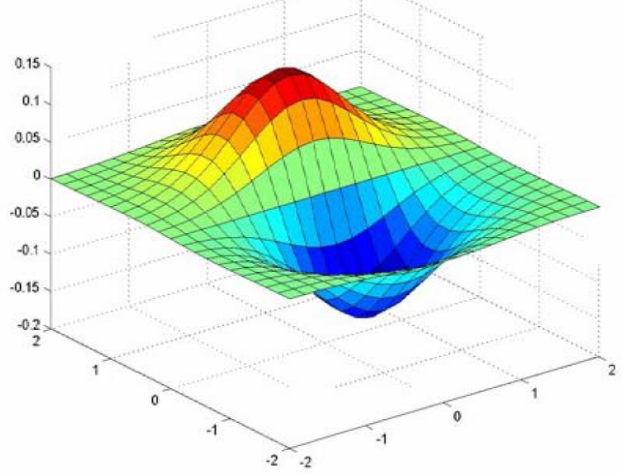
original image (Lena)

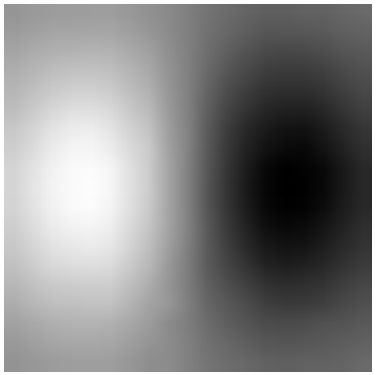
Derivative of Gaussian Filter

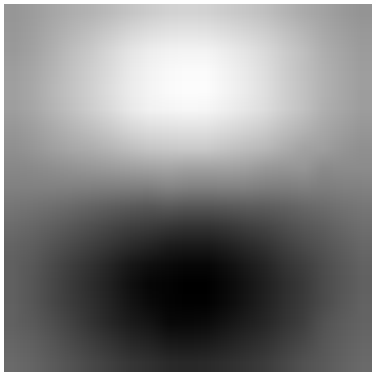
x-direction



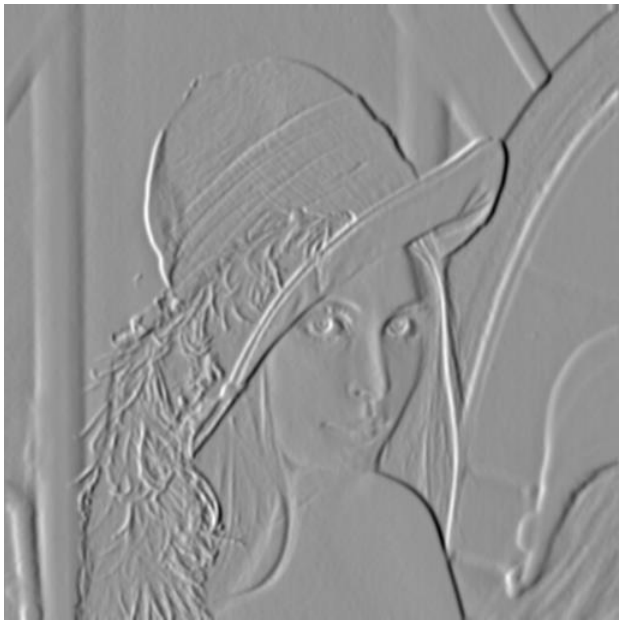
y-direction







Compute Gradients (DoG)



X-Derivative of Gaussian

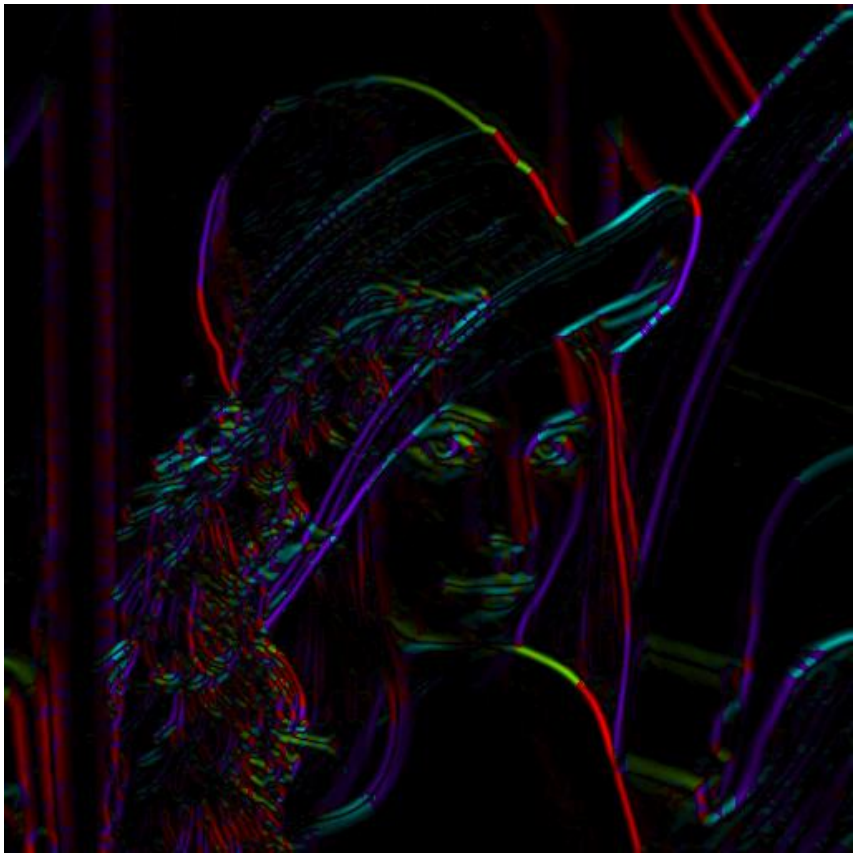
Y-Derivative of Gaussian

Gradient Magnitude

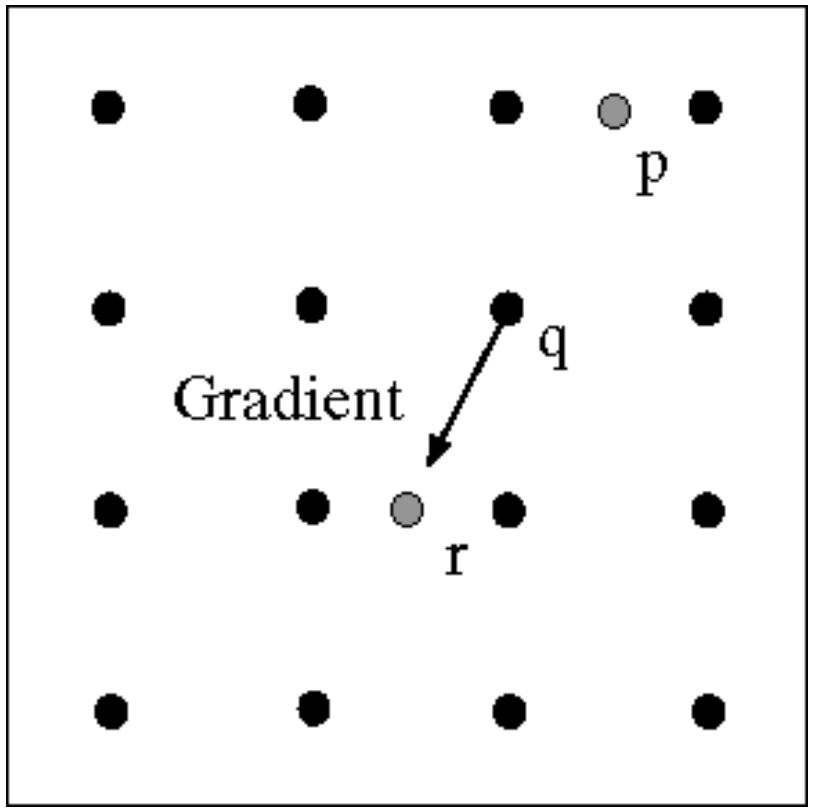
Get Orientation at Each Pixel

- Threshold at minimum level
- Get orientation

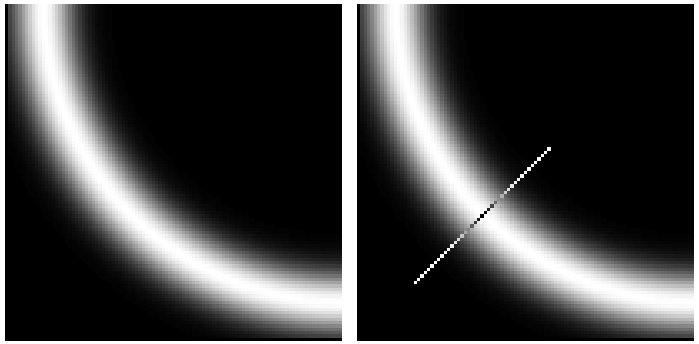
theta = atan2(gy, gx)



Non-maximum Suppression for Each Orientation



At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.



Before Non-max Suppression

After non-max suppression

Hysteresis thresholding

- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels

Final Canny Edges

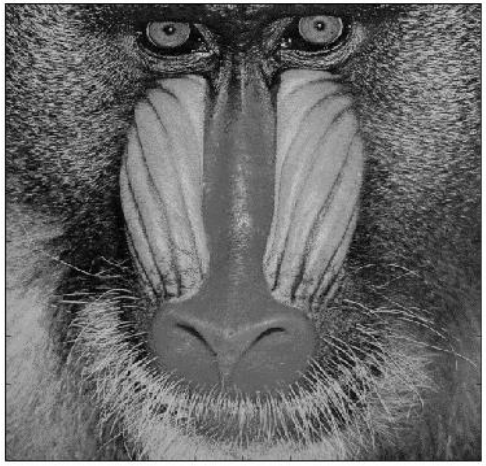
Effect of σ (Gaussian kernel spread/size)

- The choice of σ depends on desired behavior
  - large σ detects large scale edges
  - small σ detects fine features

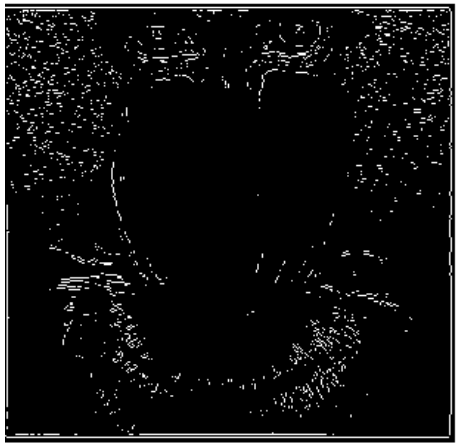
original

Canny with $\sigma = 1$

Canny with $\sigma = 2$



Prewitt



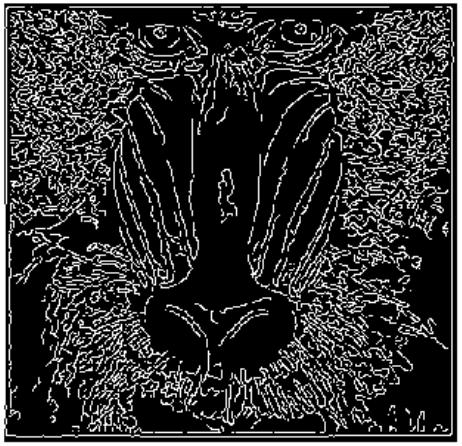
Sobel

```
outim1=edge(im,'prewitt');
outim2=edge(im,'sobel');
outim3=edge(im,'log');
outim4=edge(im,'canny');
```

Outputs from Prewitt and Sobel can be improved by adjusting the threshold values

LoG

Canny



Prewitt

Sobel

```
outim1=edge(im,'prewitt');
outim2=edge(im,'sobel');
outim3=edge(im,'log');
outim4=edge(im,'canny');
```

Outputs from Prewitt and Sobel can be improved by adjusting the threshold values

LoG

Canny

Prewitt

Sobel

```
outim1=edge(im,'prewitt');
outim2=edge(im,'sobel');
outim3=edge(im,'log');
outim4=edge(im,'canny');
```

Outputs from Prewitt and Sobel can be improved by adjusting the threshold values

LoG

Canny

Problems with Gradient Based Edge Detectors

1. **You have to choose threshold values.**

   If you double the size of an image leaving its grey values unchanged, all the gradient values will be halved…
2. **You have to choose the amount of Gaussian smoothing** (also known as the scale of analysis). This also affects gradients and hence the choice of threshold values. Also affects the perceived feature positions.
3. **Corners are often missed** because the 1D gradient at corners is usually small.
4. **1st derivative operators will only find step-like features.** There are many feature types, e.g. lines.

↘ There are still new edge detection methods proposed today, but gradient based methods remain by far the most widely used ones.

Analyzing Edge Images

- We have only found edge points
- How can we find and describe more complex features (e.g. straight lines)?

Hough Transform

Each straight line in this image can be described by an equation

Each white point if considered in isolation could lie on an infinite number of straight lines

The idea of the Hough transform is that each point 'votes' for every line that could pass through it.

The equation of the line that gets most votes 'wins'.

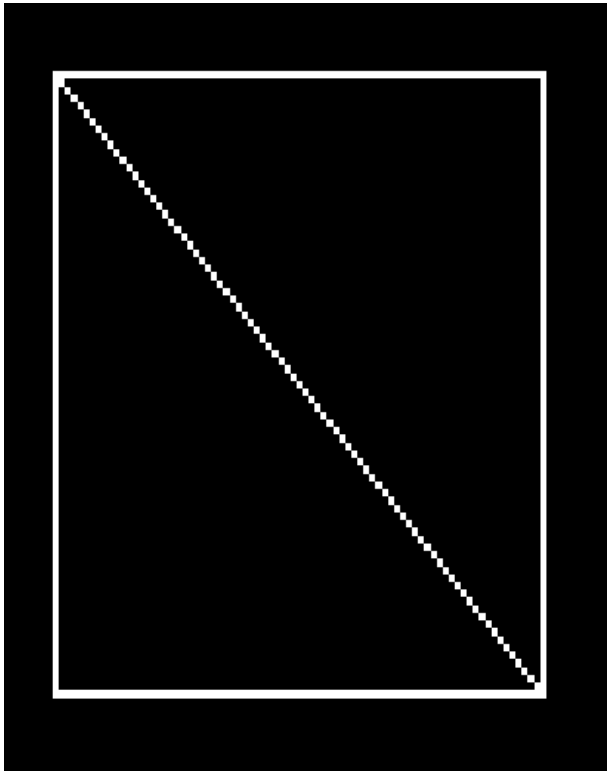


Image and Parameter Spaces

Equation of Line: $y = mx + c$

Find: (m, c)

Consider point: (x_i, y_i)

$y_i = mx_i + c \quad or \quad c = -x_i m + y_i$

Parameter space also called Hough Space

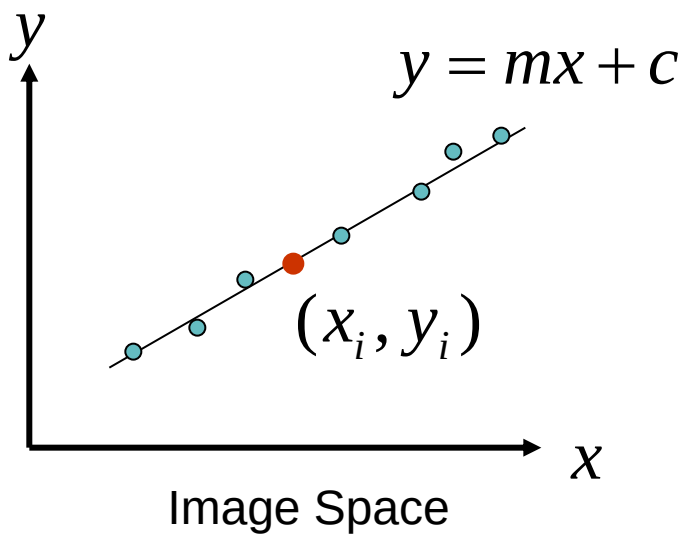
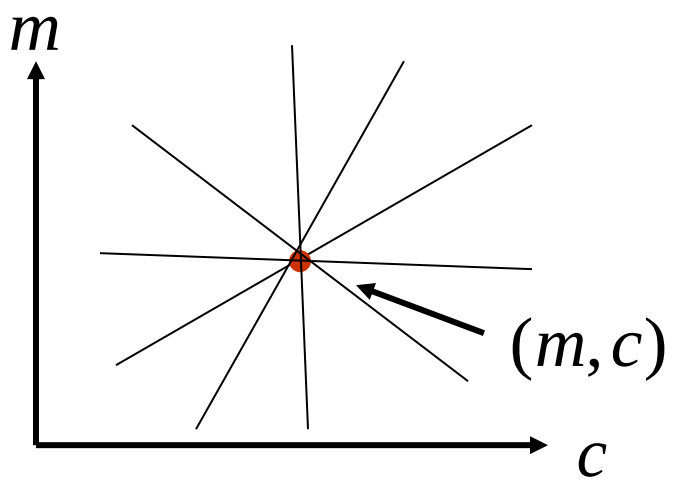


Image Space



Parameter Space

Hough Transform – Algorithm

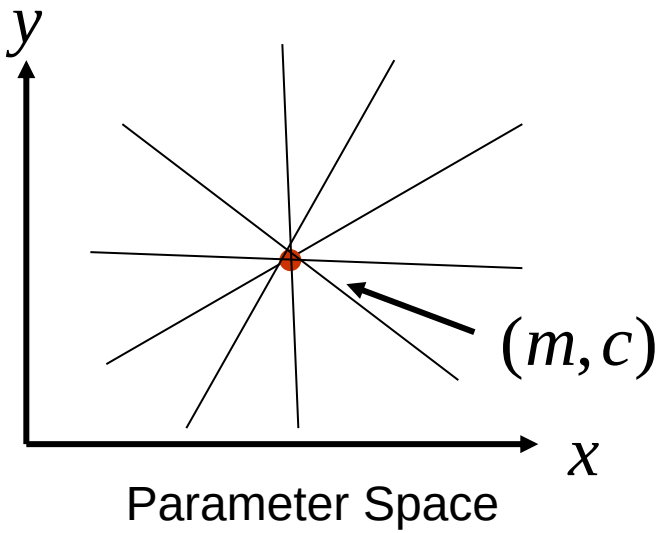
1. Quantize Parameter Space (m, c)
2. Create Accumulator Array $A(m, c)$
3. Set $A(m, c) = 0 \quad \forall m, c$
4. For each image edge (x_i, y_i) increment:

$$A(m, c) = A(m, c) + 1$$

if (m, c) lies on the line:

$$c = -x_i m + y_i$$

6. Find local maxima in $A(m, c)$



Parameter Space

$A(m, c)$

	1						1		
		1				1			
			1		1				
				2					
			1		1				
		1				1			
	1						1		

Better Parameterization

NOTE: $-\infty < m < \infty$

Large Accumulator

More memory and computations

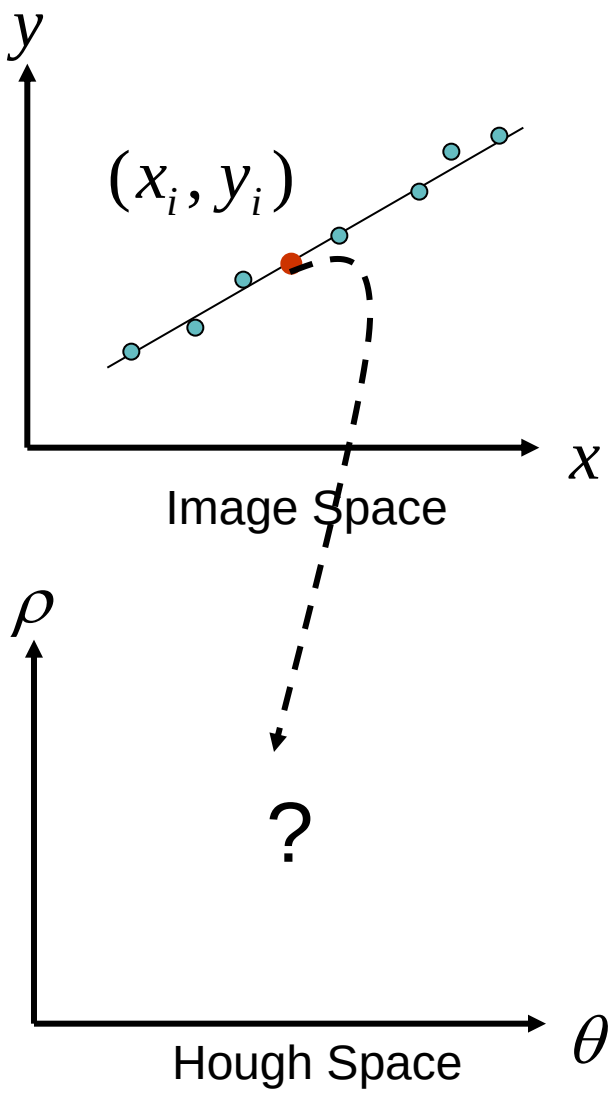
Improvement: (Finite Accumulator Array Size)

Line equation: $x\cos\theta - y\sin\theta + \rho = 0$

Here $0 \le \theta < 2\pi$

$0 \le \rho \le \rho_{\max}$

Given points (x_i, y_i) find (ρ, θ)



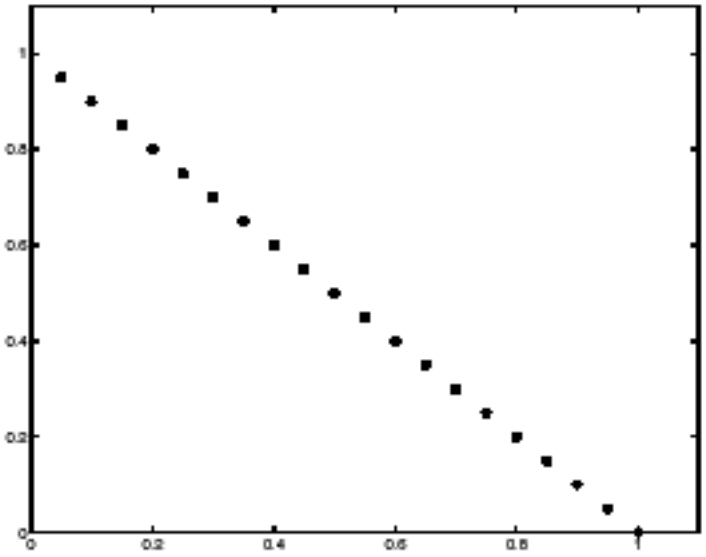
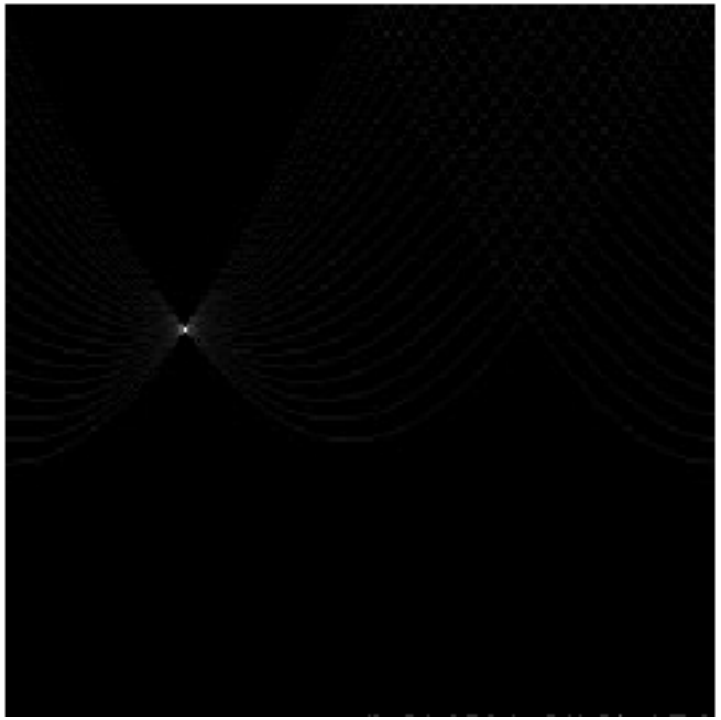
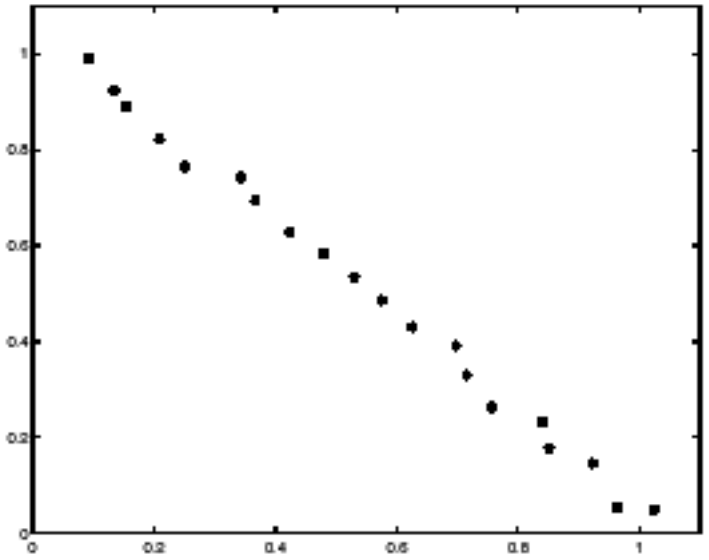


Image space



Votes

(Horizontal axis is θ, vertical is rho)



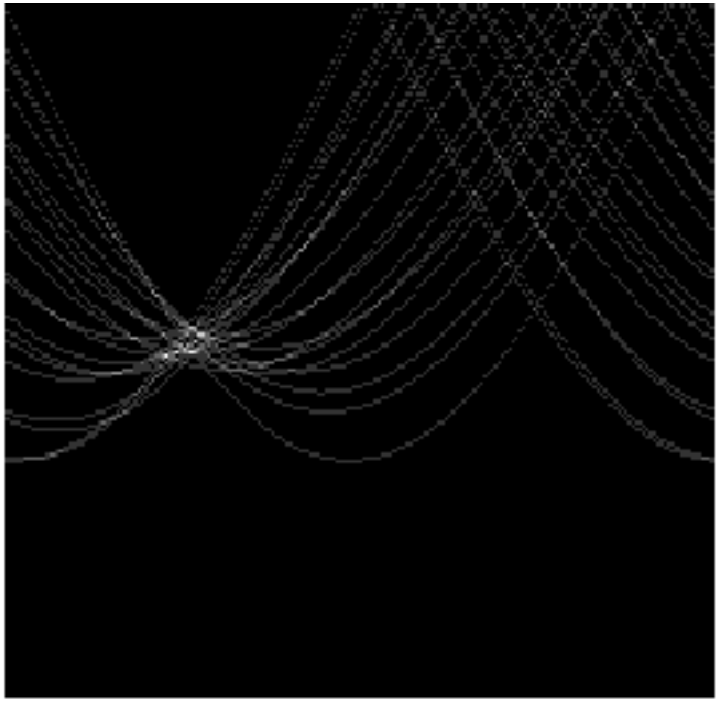


Image space

Votes

(Horizontal axis is θ, vertical is rho)

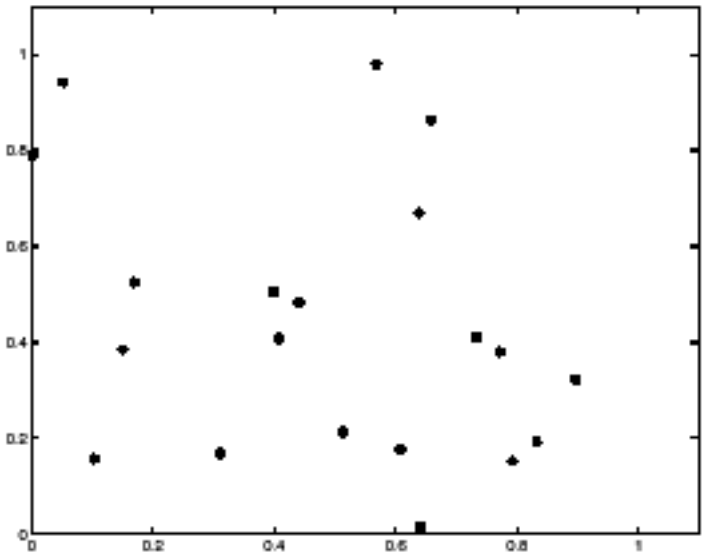
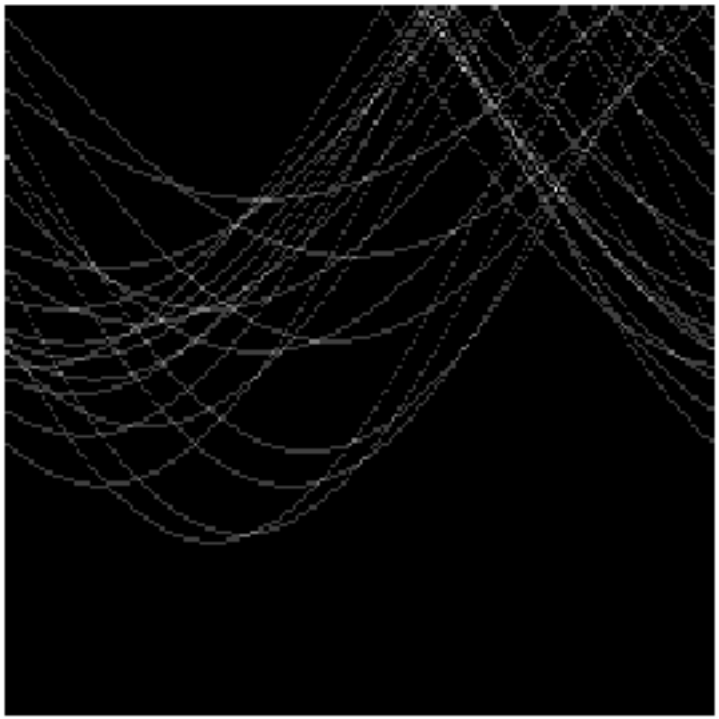


Image space



Votes

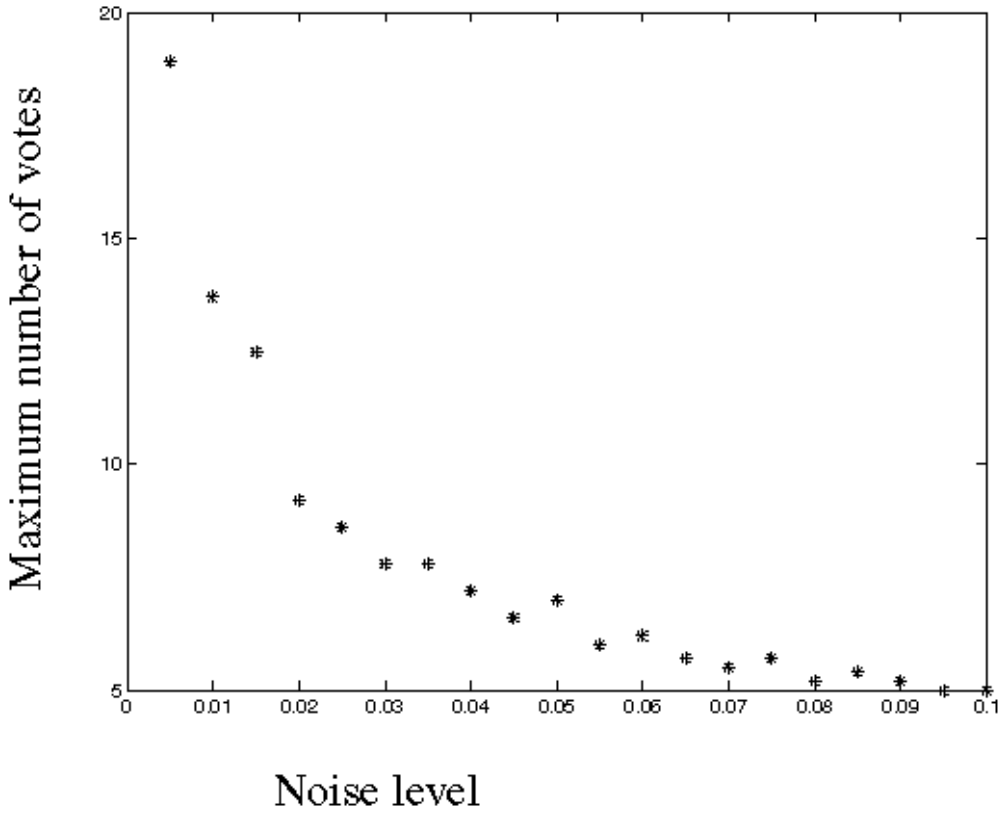
(Horizontal axis is θ, vertical is rho)

Mechanics of the Hough transform

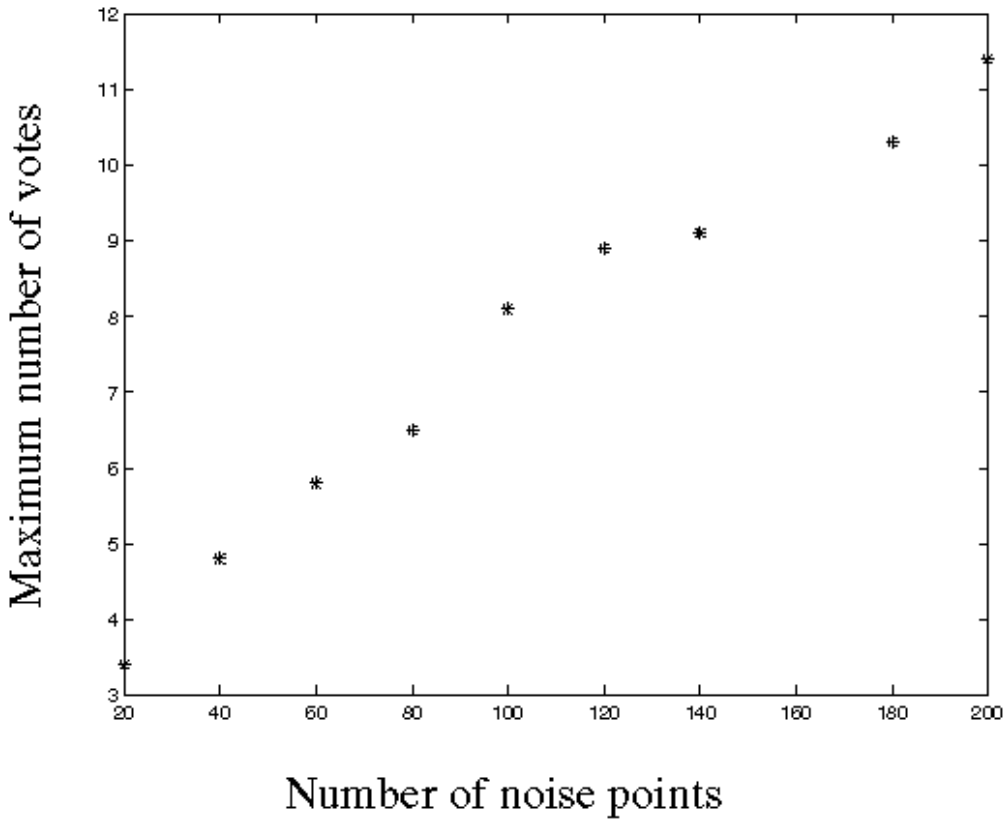
- Difficulties
  - how big should the cells be? (too big, and we merge quite different lines; too small, and noise causes lines to be missed)
  - small cell sizes => more computation!
  - large cell sizes => less accuracy!

- How many lines?
  - Count the peaks in the Hough array
  - Treat adjacent peaks as a single peak, i.e. must perform non-maximum suppression

Fewer votes land in a single bin when noise increases.

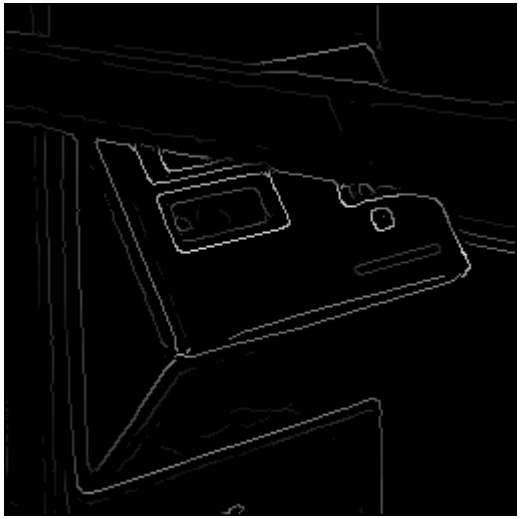


Adding more clutter increases number of bins with false peaks.

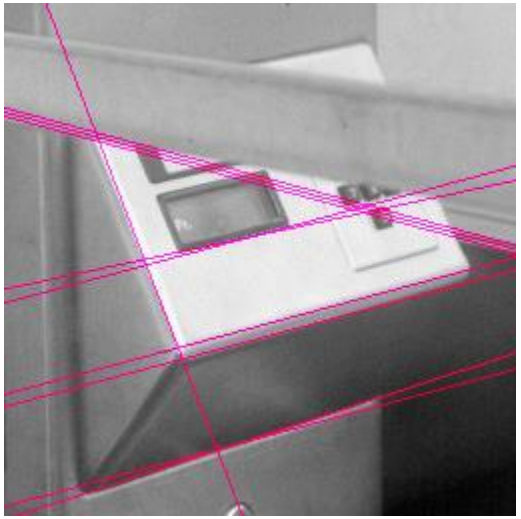


Hough Line Detection

Original



Edge Detection



Found Lines

http://homepages.inf.ed.ac.uk/rbf/HIPR2/linedetdemo.htm

Finding Circles by Hough Transform

- Equation of Circle:

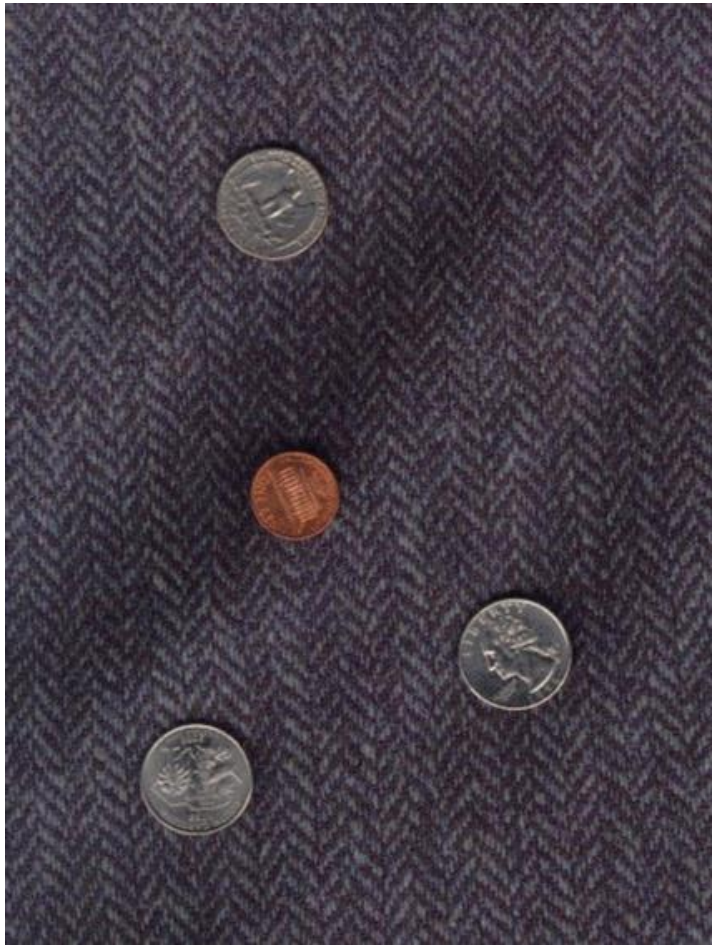
$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- If radius is known: (2D Hough Space)
  - Accumulator Array $A(a, b)$

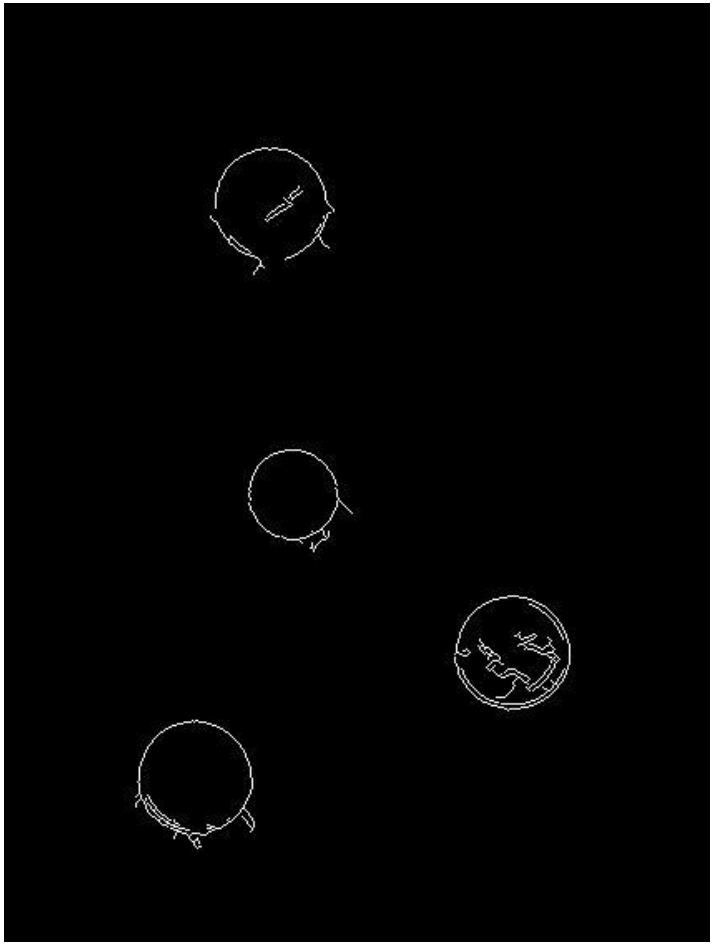
- If radius is not known: 3D Hough Space!
  - Use Accumulator array $A(a, b, r)$

http://www.markschulze.net/java/hough/

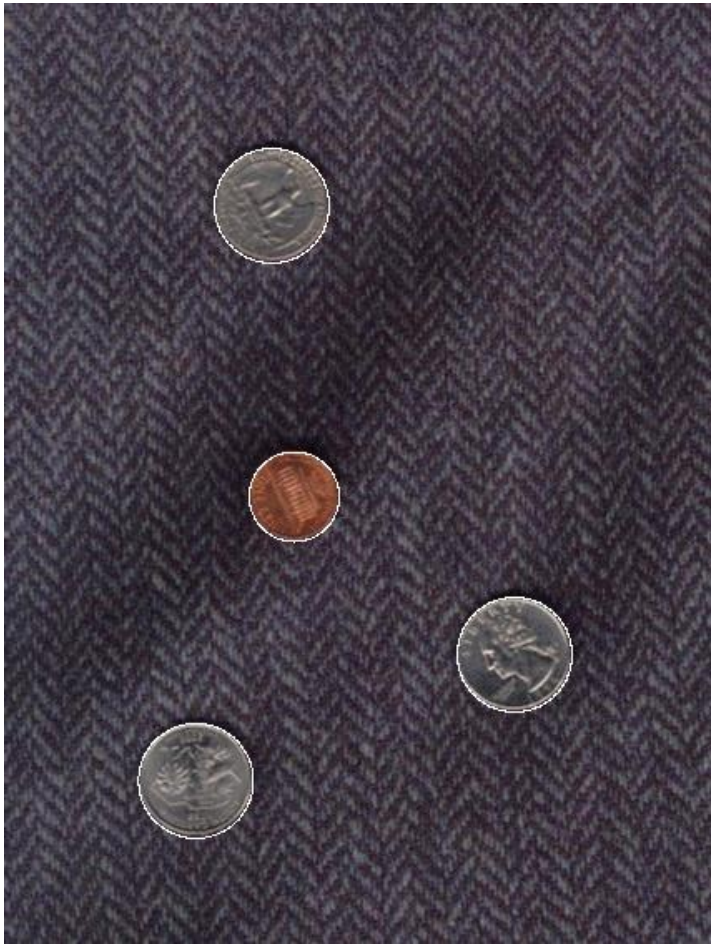
Original



Edges (note noise)

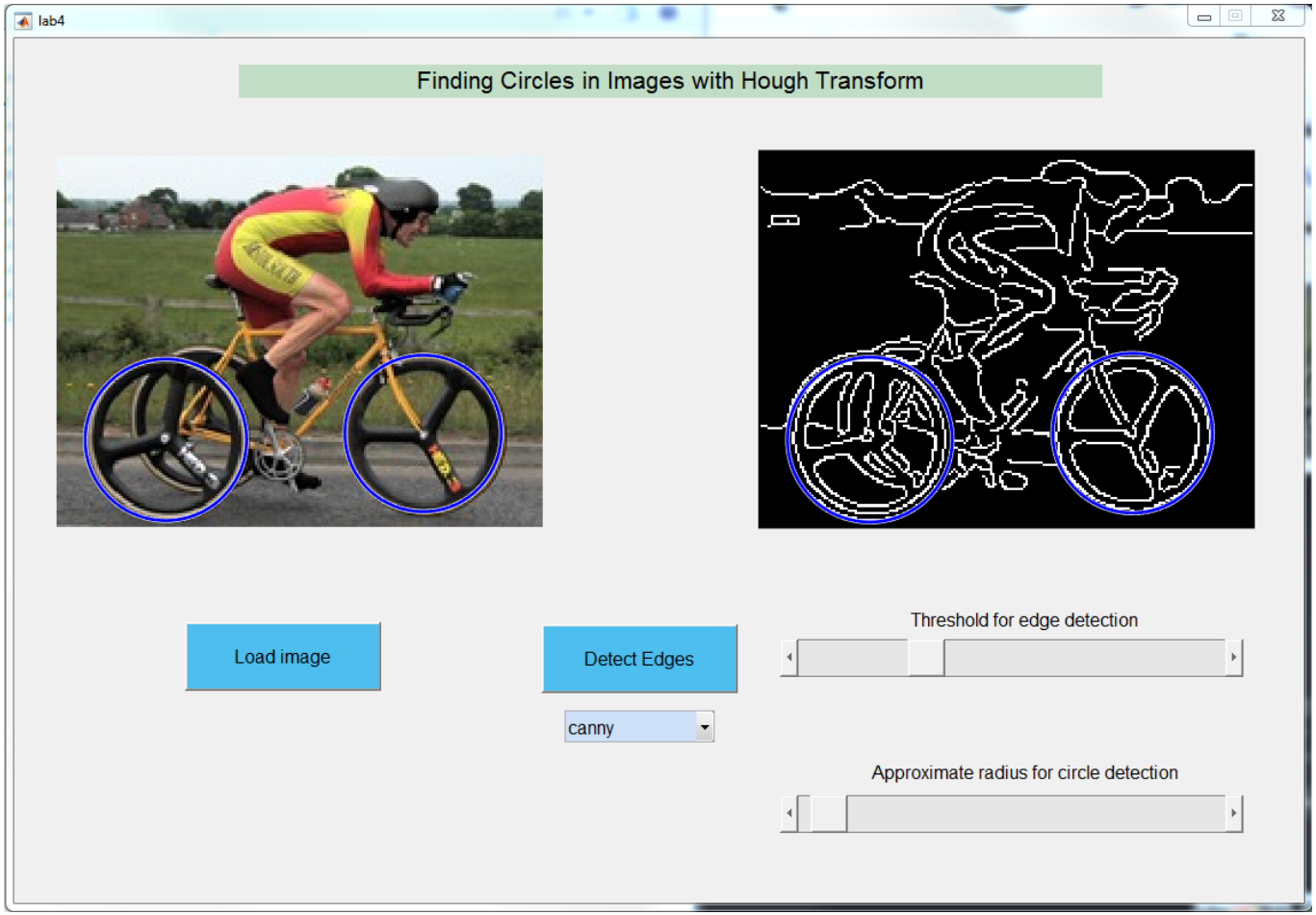


Finding Coins



Note that because the quarters and penny are different sizes, a different Hough transform (with separate accumulators) was used for each circle size.

Week 5 Lab


lab4
Finding Circles in Images with Hough Transform
Load image
Detect Edges
canny
Threshold for edge detection
Approximate radius for circle detection

Week 5 Lab

- Detect circles in images e.g. the pupil or bicycle tyres.
- You will need to do edge detection first. Use a popup menu to choose an edge detection algorithm (min two options) and a slidebar to adjust the threshold of the edge detector. See lab sheet for details.
- Another slide bar should vary the approximate radius for the Hough circle detection. Show the detected circles on both images.
- You are allowed to use the Matlab inbuilt circle detection function.
- Marks distribution:
  - GUI that can at least load/display an image – 1 mark
  - Detect edges with button – 1 mark
  - Detect edges with slide bar – 1 mark
  - Detect circles with slide bar – 2 marks

Summary

- Image edges
- Sobel operator
- Canny edge detection
- Hough transform
- Line and circle detection

Acknowledgements: The slides are based on previous lectures by A/Prof Du Huynh and Prof Peter Koveski. Other material has been taken from Wikipedia, computer vision textbook by Forsyth & Ponce, and OpenCV documentation.