Edge Detection

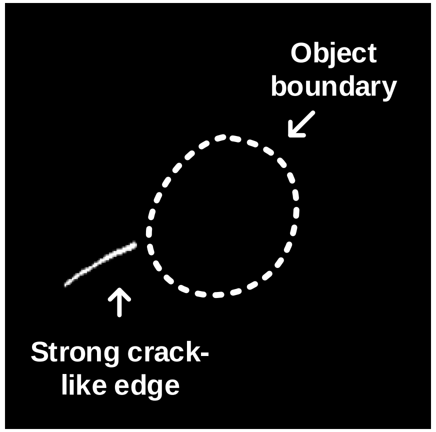
Chul Min Yeum
Assistant Professor
Civil and Environmental Engineering
University of Waterloo, Canada

CIVE 497 – CIVE 700: Smart Structure Technology

Last updated: 2021-01-06

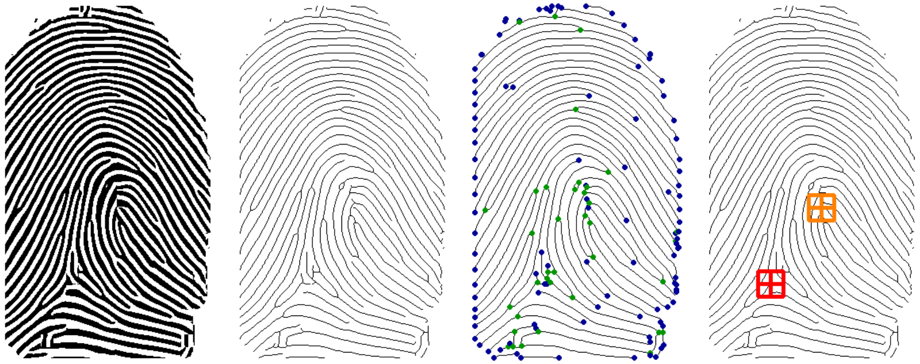
UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING

Edge Detection Applications



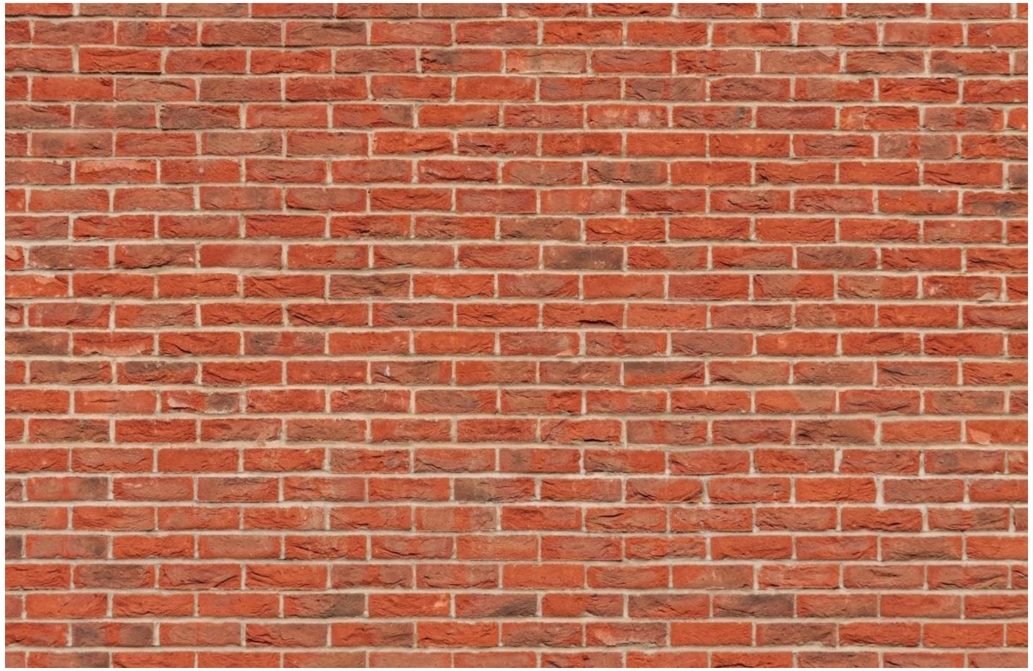
Crack detection

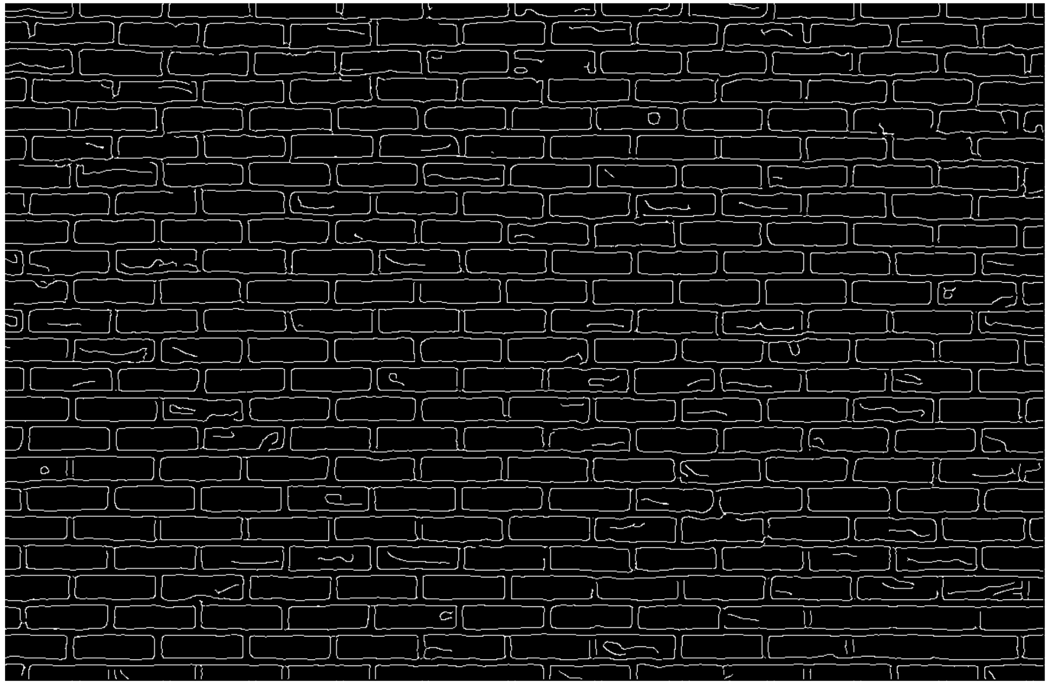
Defect detection (machine vision)



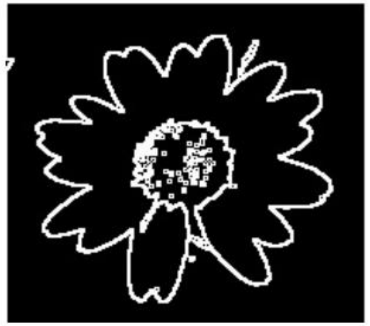
Fingerprint detection

Why Do We Detect Edges?



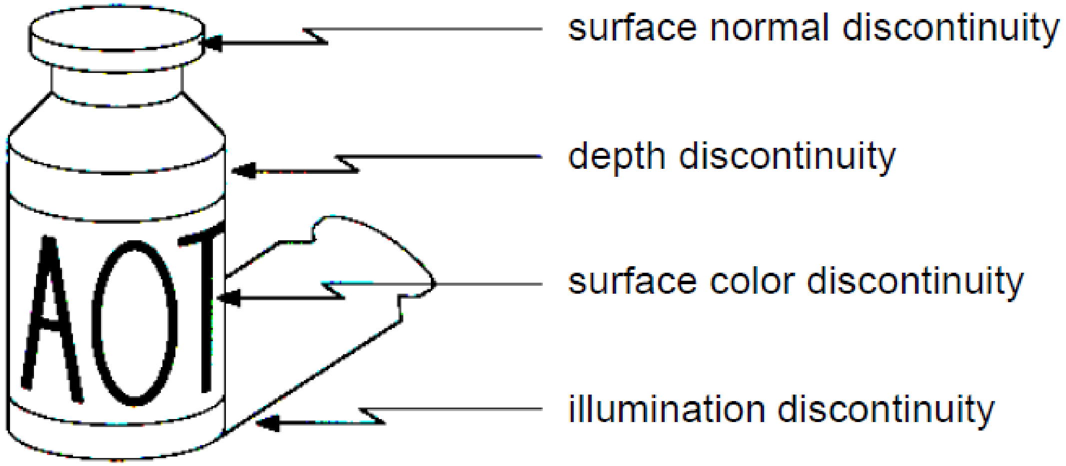


- Convert a 2D image into a set of curves
  - Semantic and shape information
  - More compact than pixels

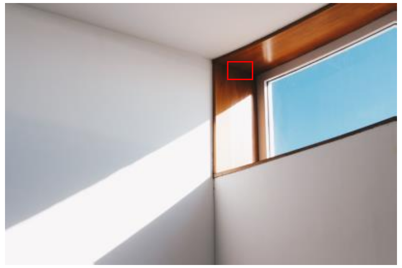


Tutorial

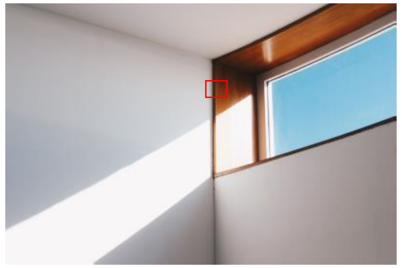
Origin of Edges



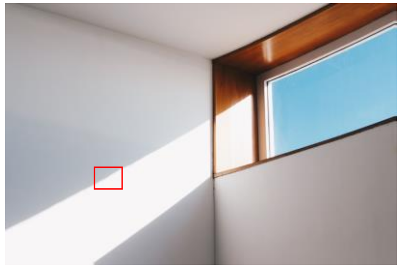
Surface normal discontinuity



Material discontinuity



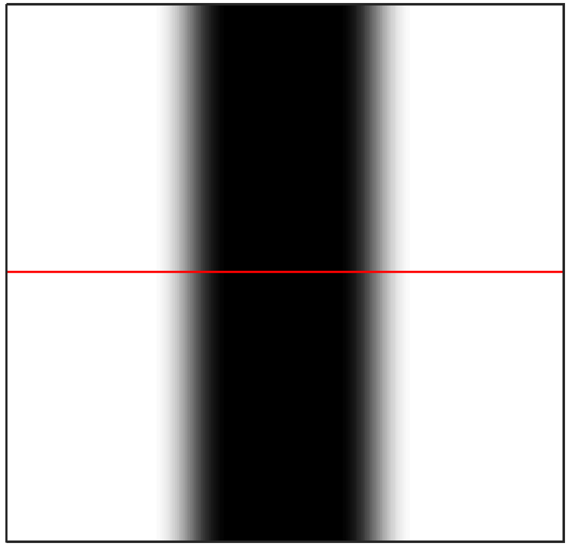
Lighting discontinuity



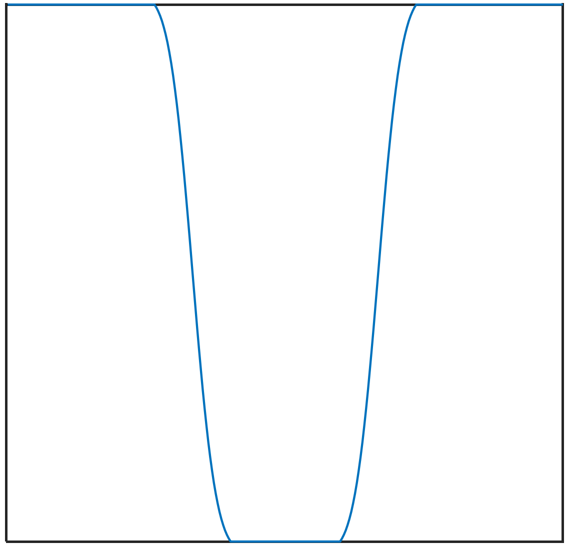
Tutorial

Characterizing Edge by Differentiation

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

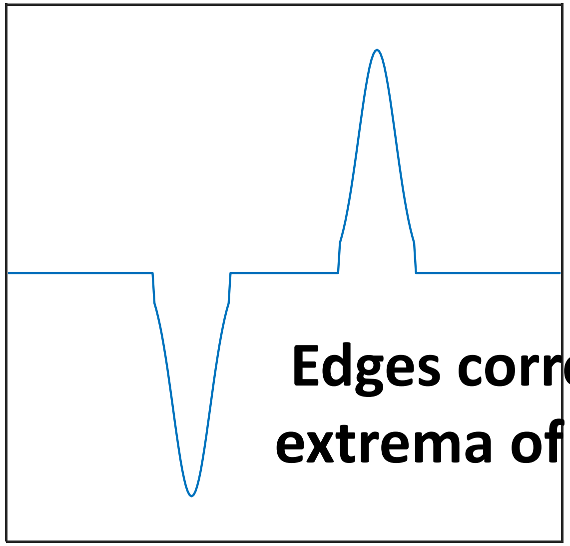


Image



Intensity function along a red line

$$\frac{d}{dx}$$



Edges correspond to extrema of derivative

First derivative

Image Derivatives

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

Option 1: Reconstruct a continuous image, f, then compute the derivative

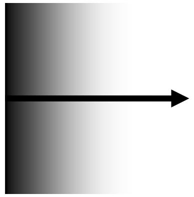
Option 2: Take discrete derivative (finite difference)

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

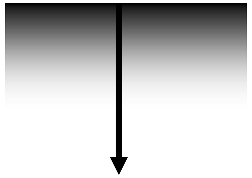
$\Delta x = 1$ pixel

Image Gradient

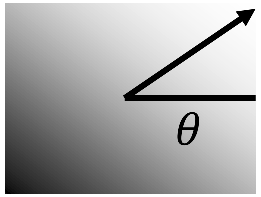
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



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



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



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

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

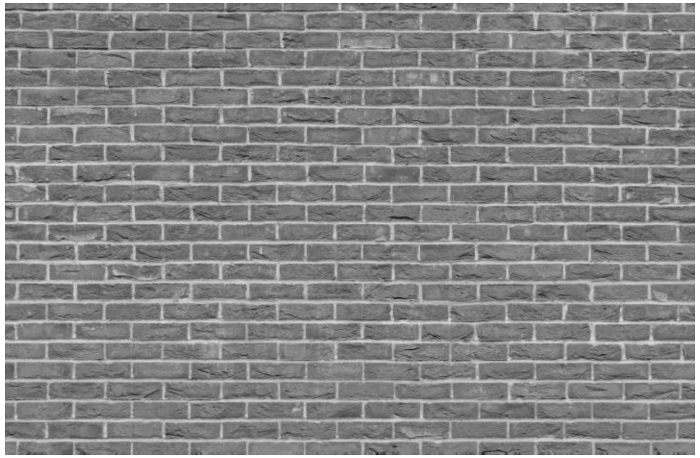
The gradient direction is given by

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

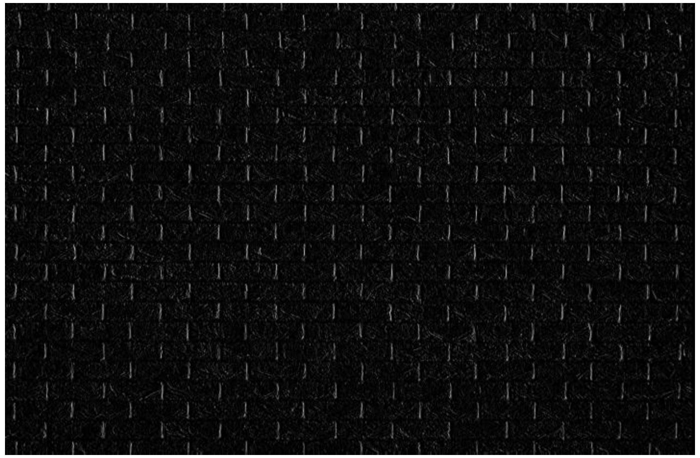
The edge strength is given by the gradient magnitude

$$||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

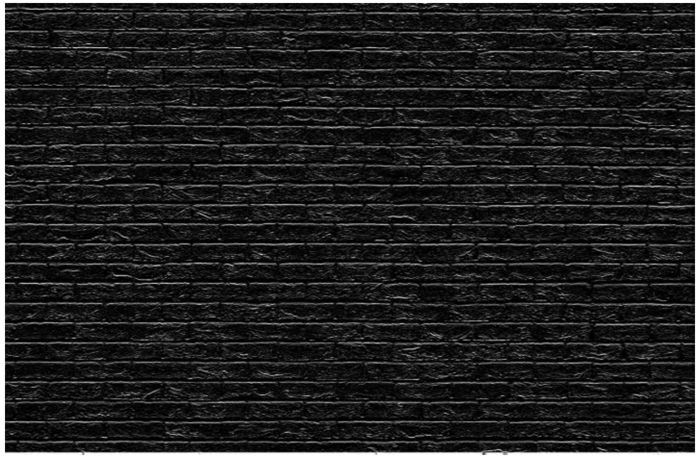
Example: Image Gradient



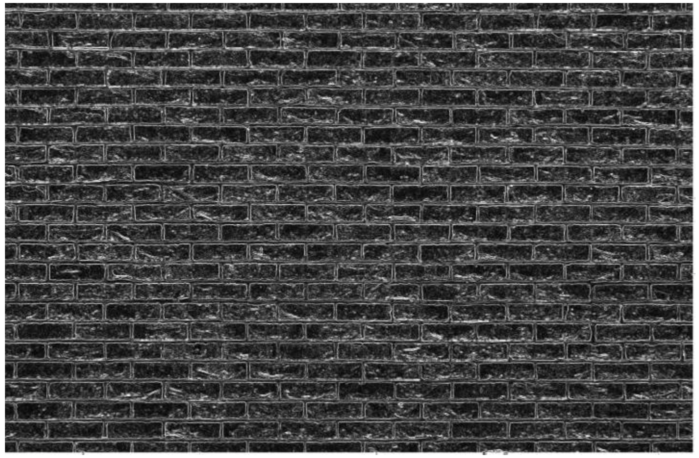
f



$\partial f / \partial x$



$\partial f / \partial y$

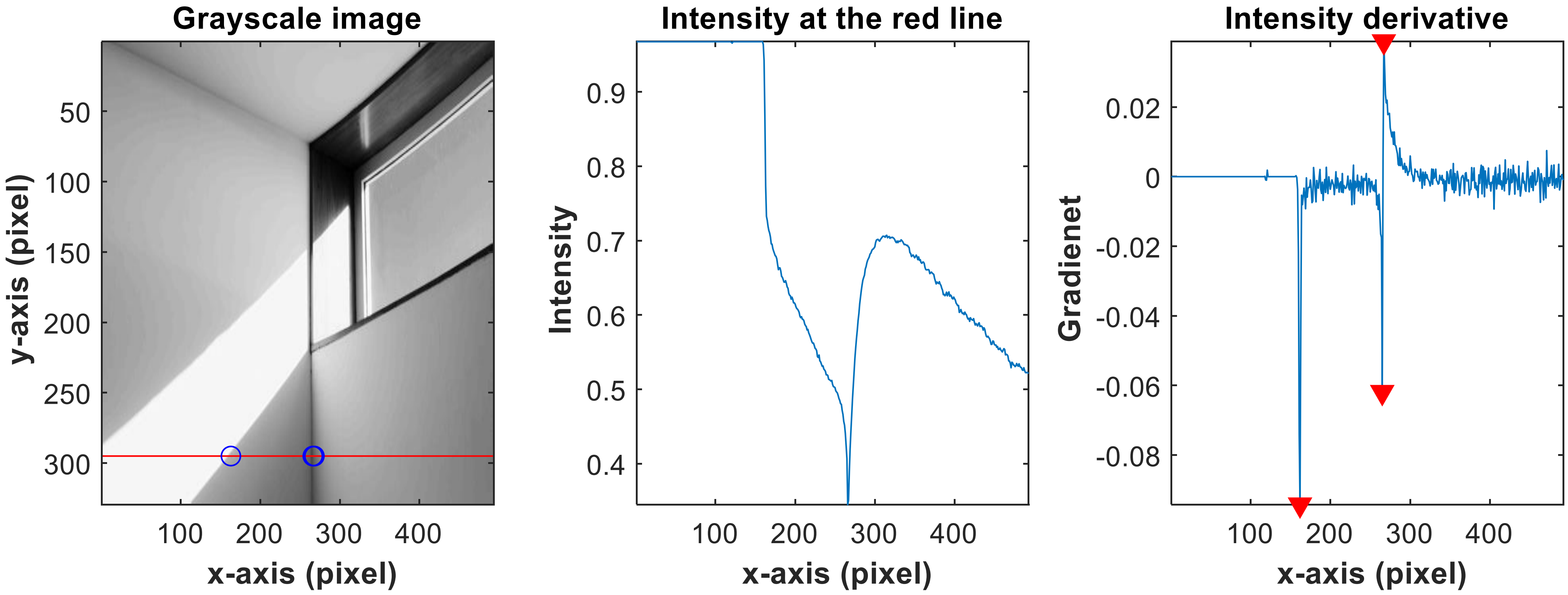


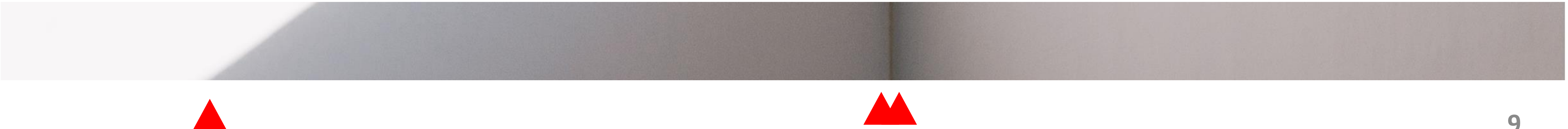
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial Y}\right)^2}$$

grayscale

Tutorial

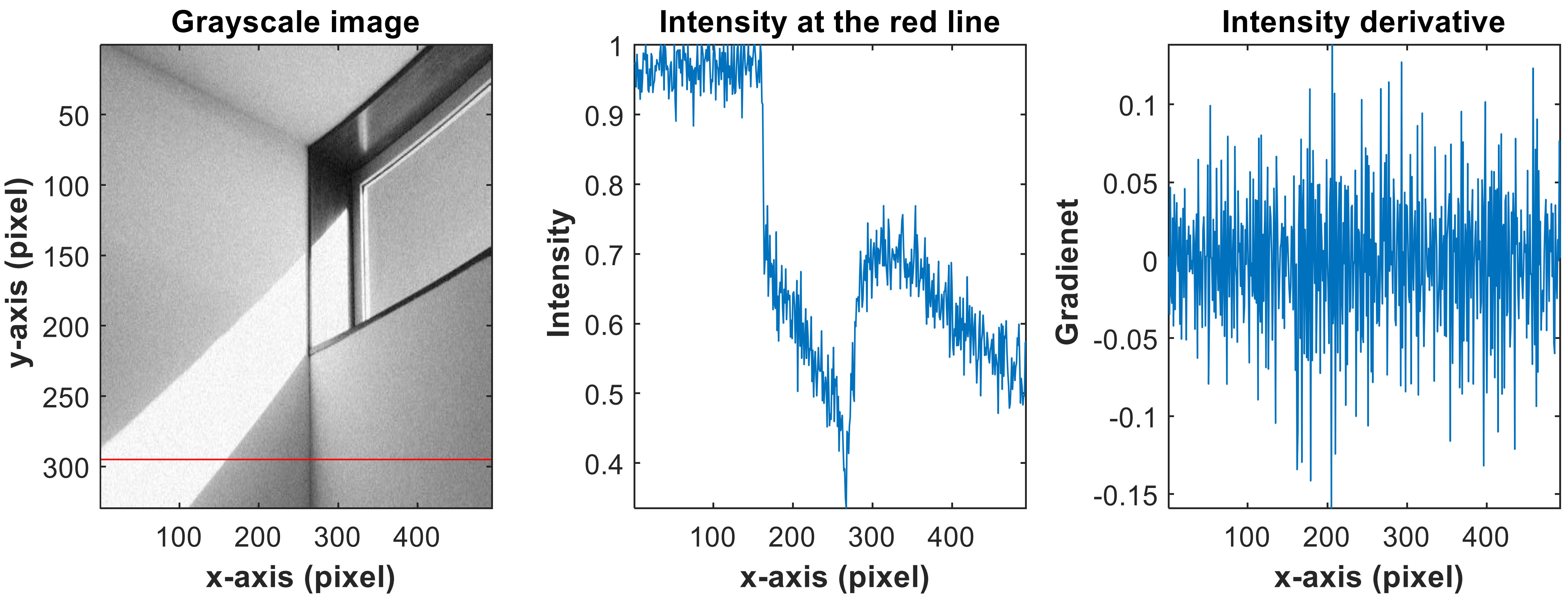
Intensity Changes on an Image





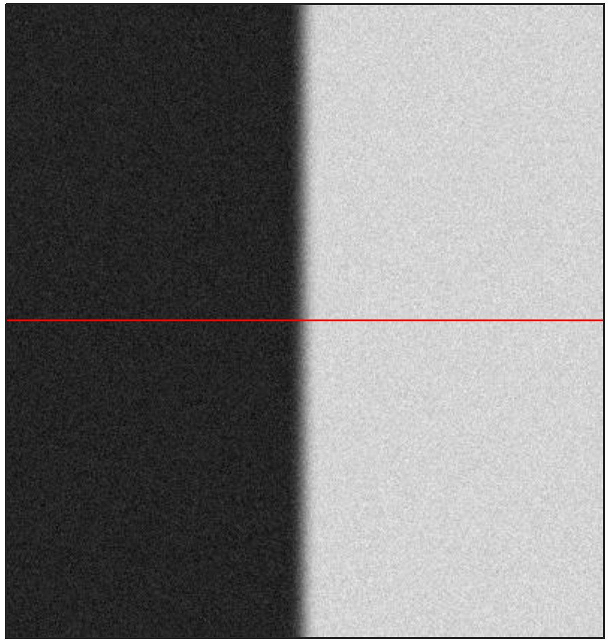
Tutorial

Intensity Changes on an Image (with Noise)

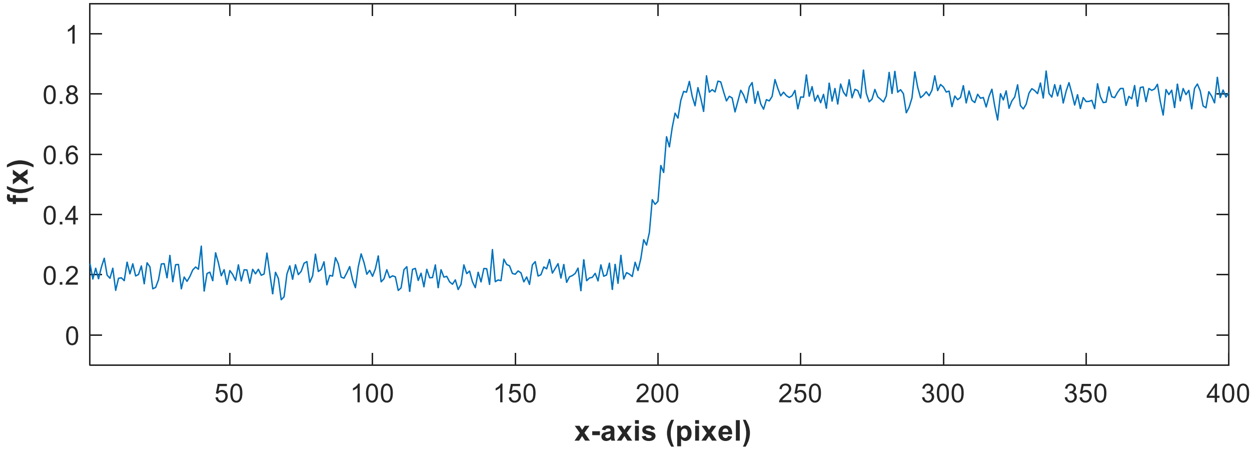


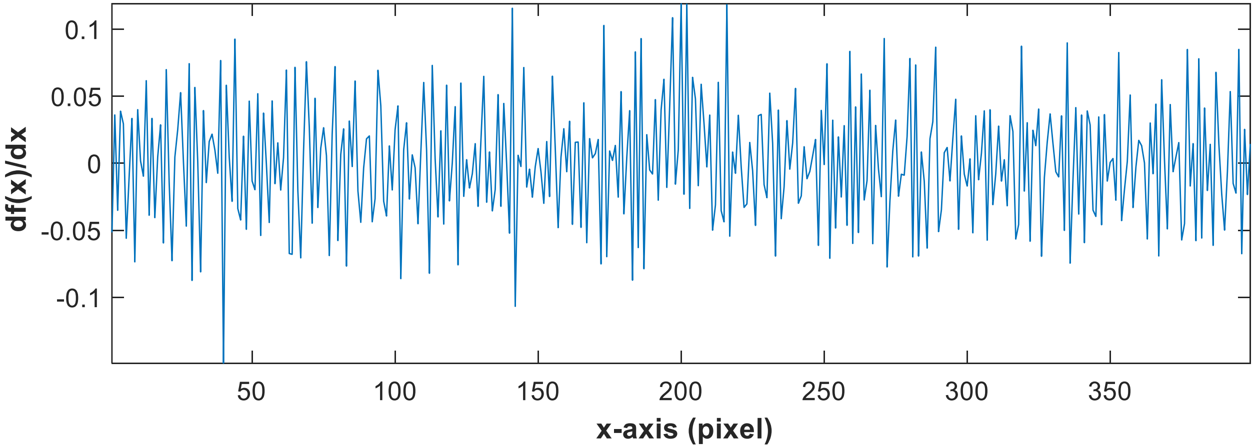
Tutorial

Effects of Noise



Noisy input image





Q: Where is the edge?

Tutorial

(Review) Properties of Fourier Transforms

Supplement

1. Time scaling

$$F\big(x(at)\big) = \frac{1}{|a|} X\left(\frac{f}{a}\right)$$

Inverse spreading relationship

2. Time reversal

$$F\big(x(-t)\big) = X(-f)$$

3. Differentiation

$$\boxed{F\big(\dot{x}(t)\big) = i2\pi f X(f)}$$

4. Time shifting

$$F\big(x(t - t_0)\big) = e^{-i2\pi f t_0} X(f)$$

Only phase shift !

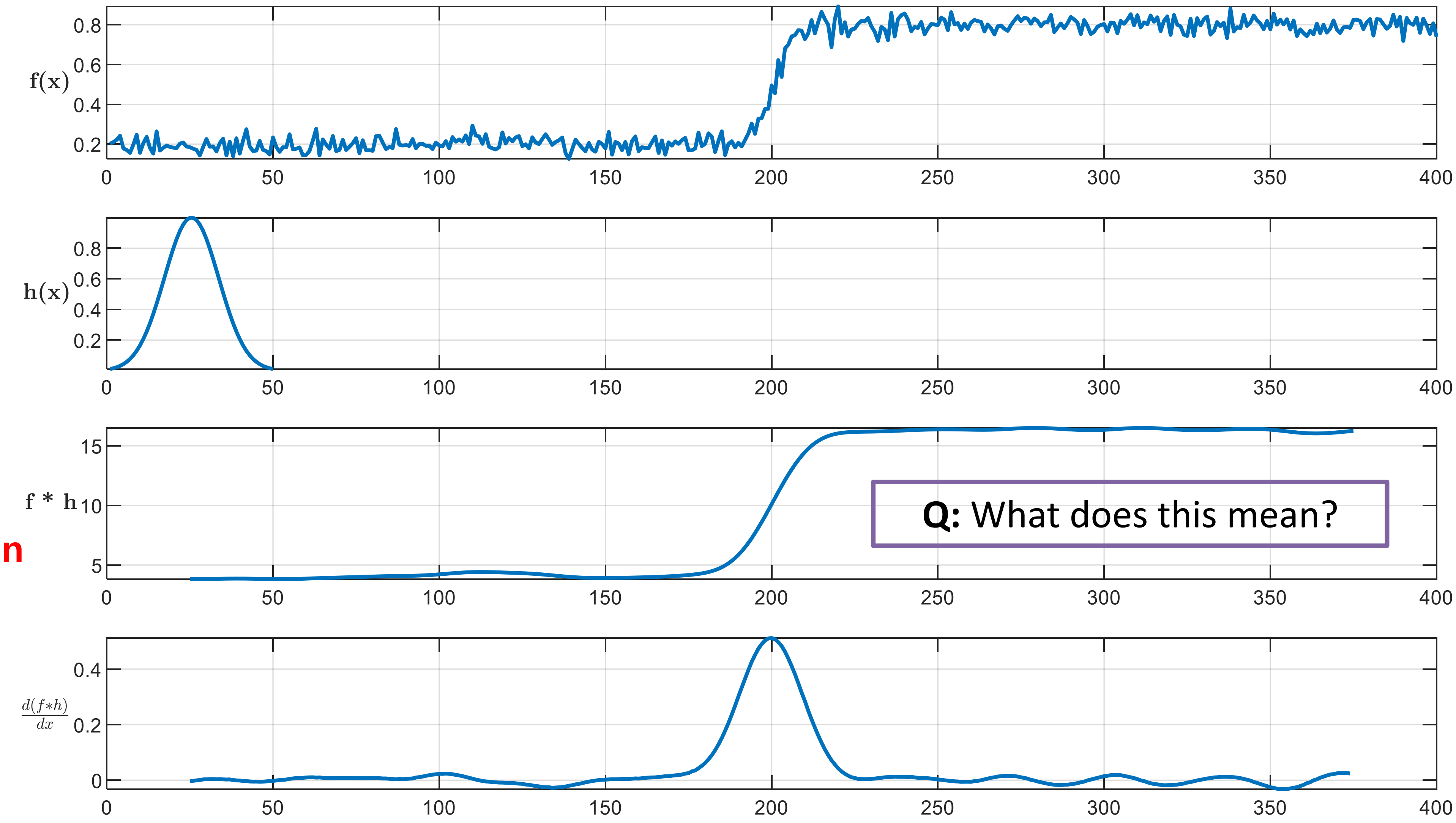
Sine wave

5. Modulation

$$F\big(x(t)e^{i2\pi f_0 t}\big) = X(f - f_0)$$

$$F(x(t)\cos(2\pi f_0 t)) = \frac{1}{2}[X(f - f_0) + X(f + f_0)]$$

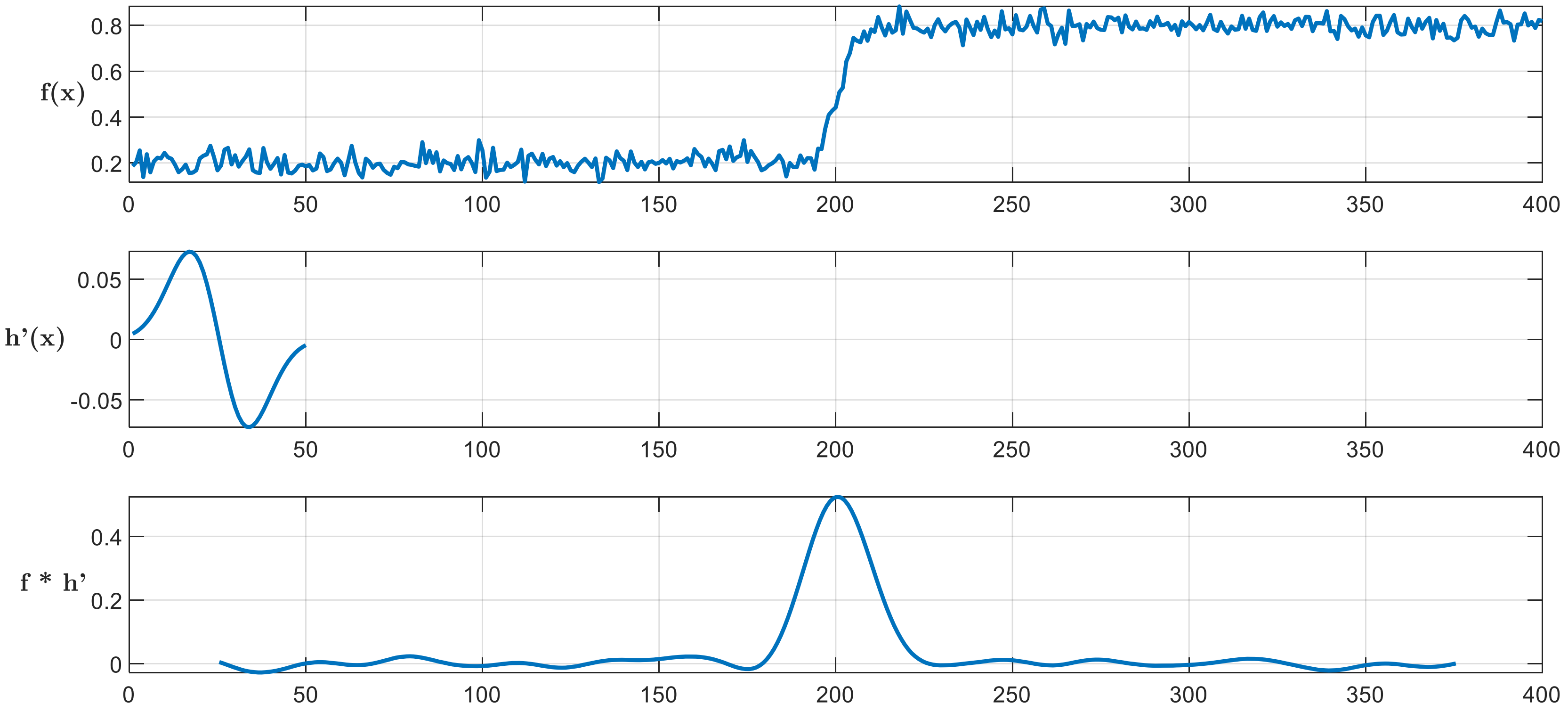
Smoothing First and Applying a Derivative Operator



convolution

Smoothing First and Applying a Derivative Operator (Continue)

Relationship with differentiation: $(F * H)' = F' * H = F * H'$

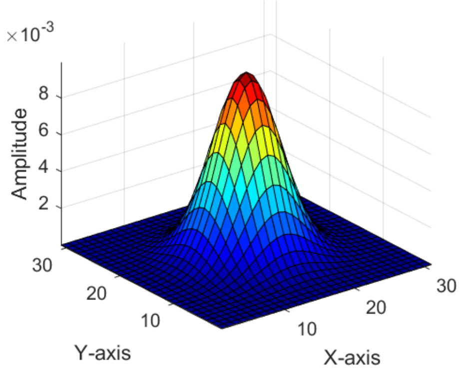


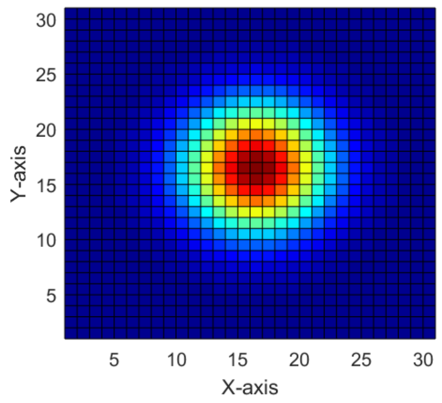
$$(f * h)' = f * h'$$

Q: What is the advantage?

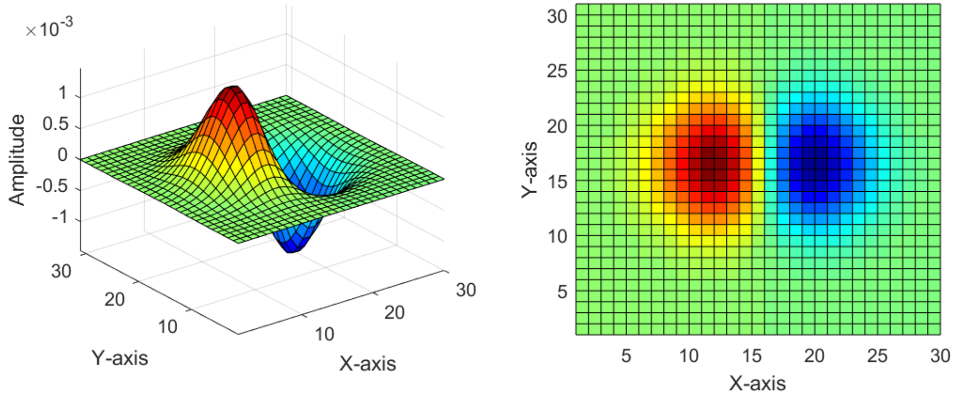
Tutorial

2D Edge Detection Filter



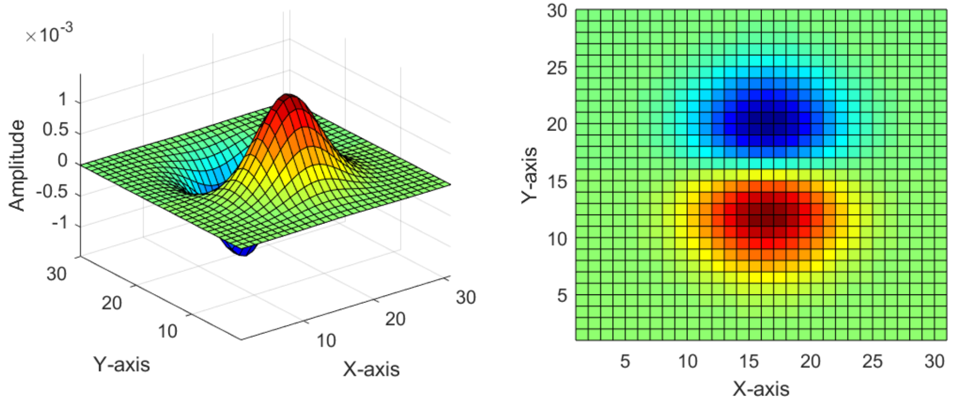


$$\frac{\partial h_\sigma(x,y)}{\partial x}$$



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

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



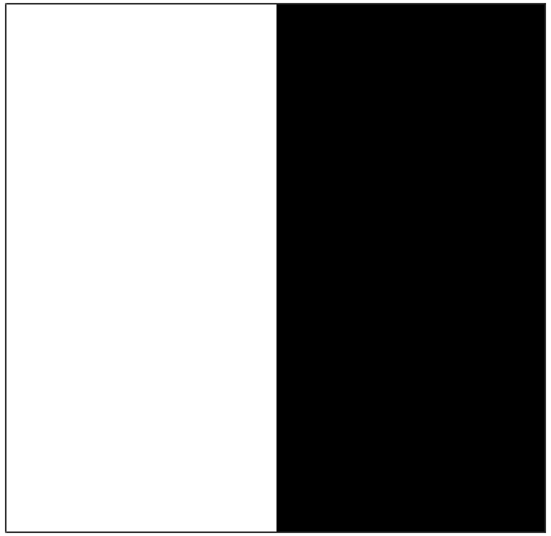
Gaussian kernel

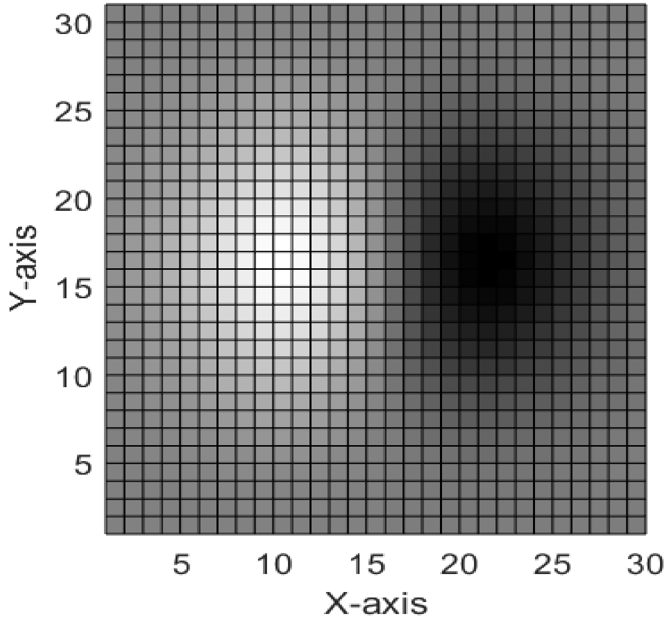
0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

Derivative of Gaussian kernel

Tutorial

2D Edge Detection Filter (Continue)





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

?

Prewitt operator

Sobel Operator

Common approximation of derivative of Gaussian

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

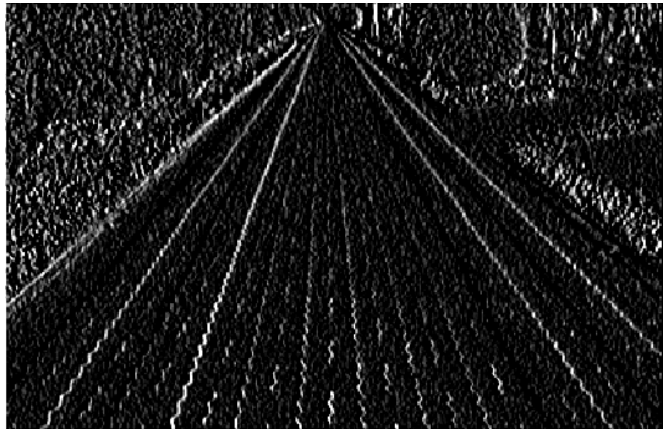
s_x

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

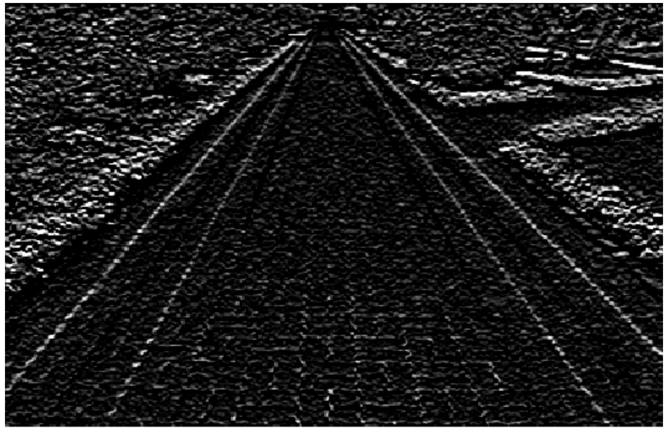
s_y

Sobel Operator (Example)

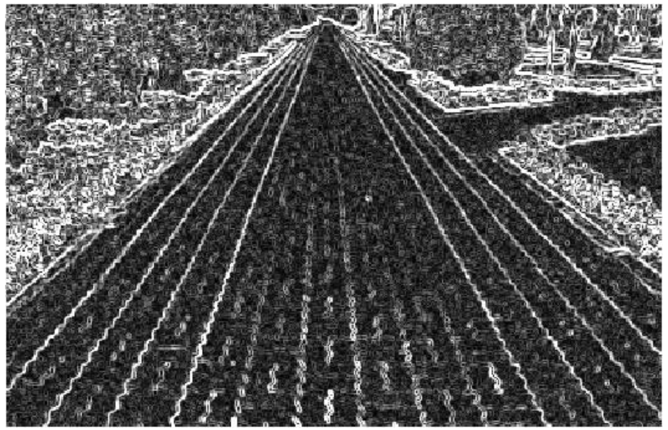
f



s_x



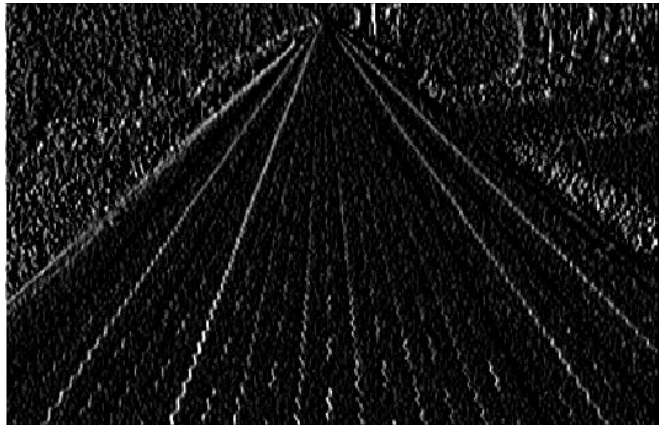
s_y



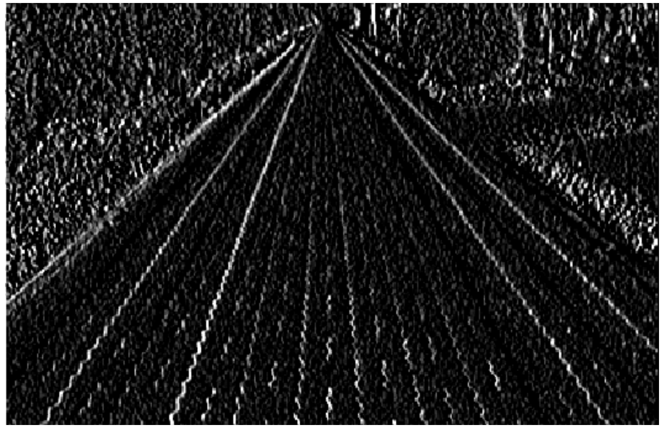
$\|\nabla S\| = \sqrt{(s_x)^2 + \left(s_y\right)^2}$

Tutorial

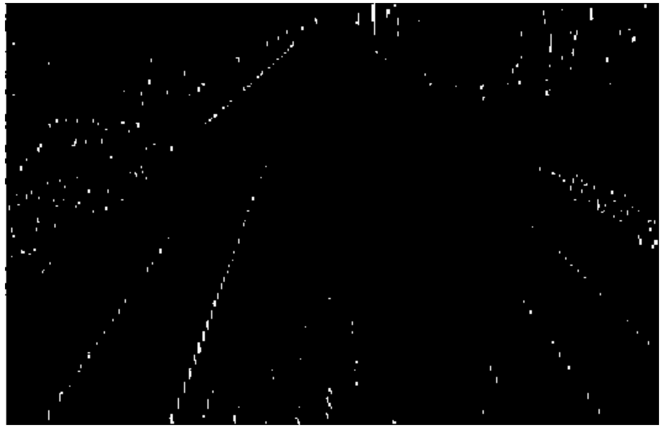
Comparison of Derivative Operators (Prewitt and Sobel)



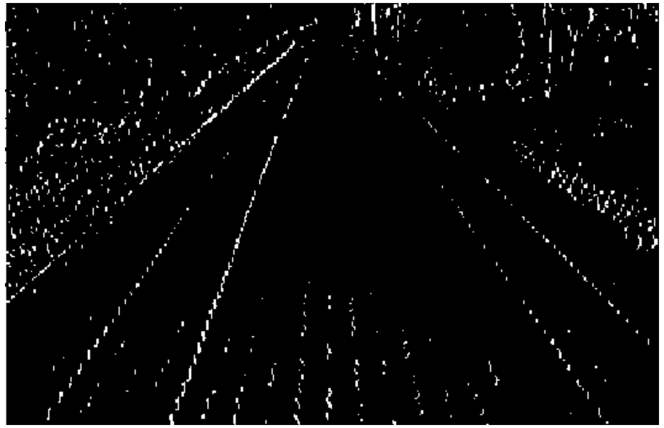
f_x



s_x



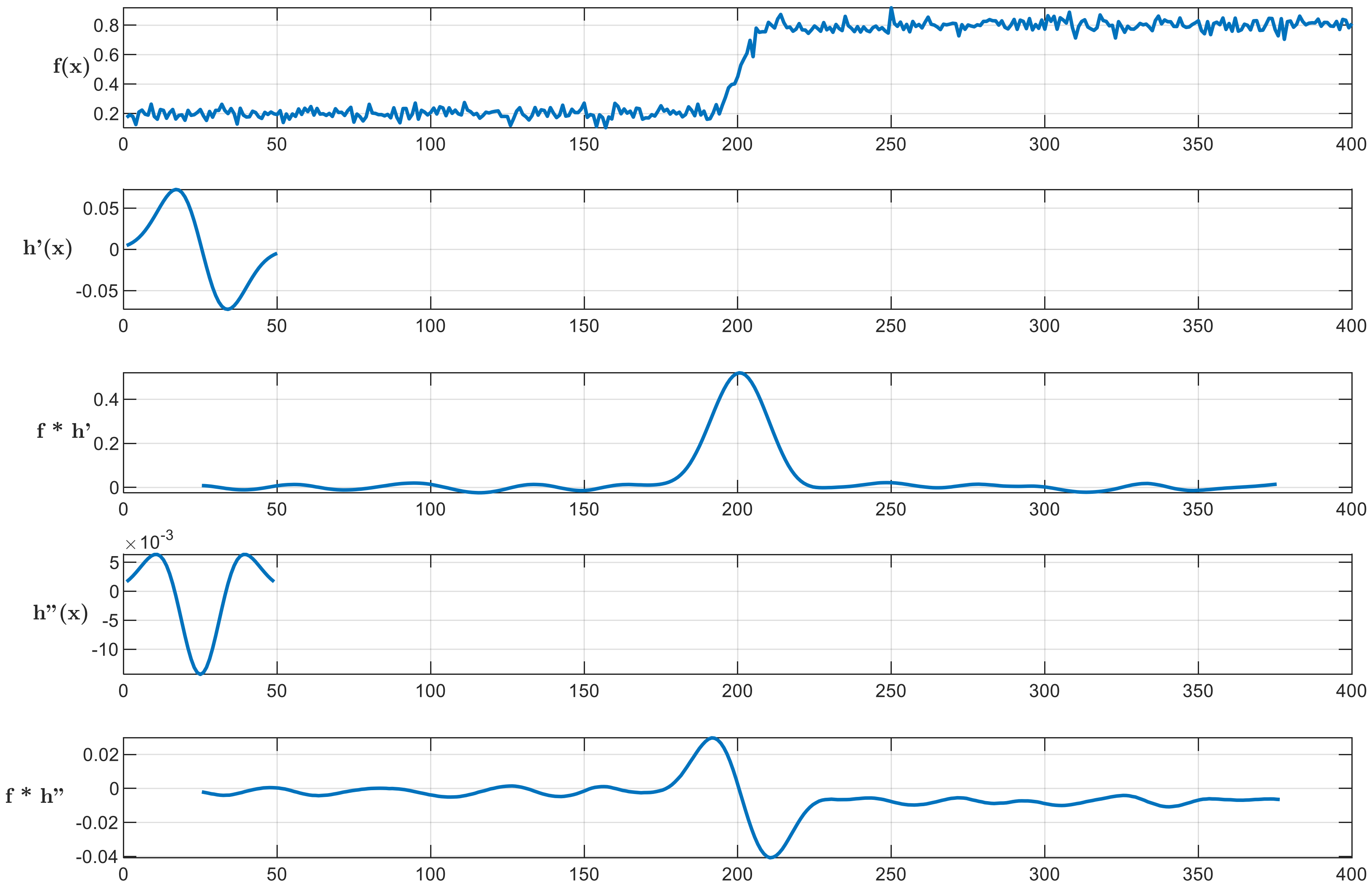
$f_x > 0.75$



$s_x > 0.75$

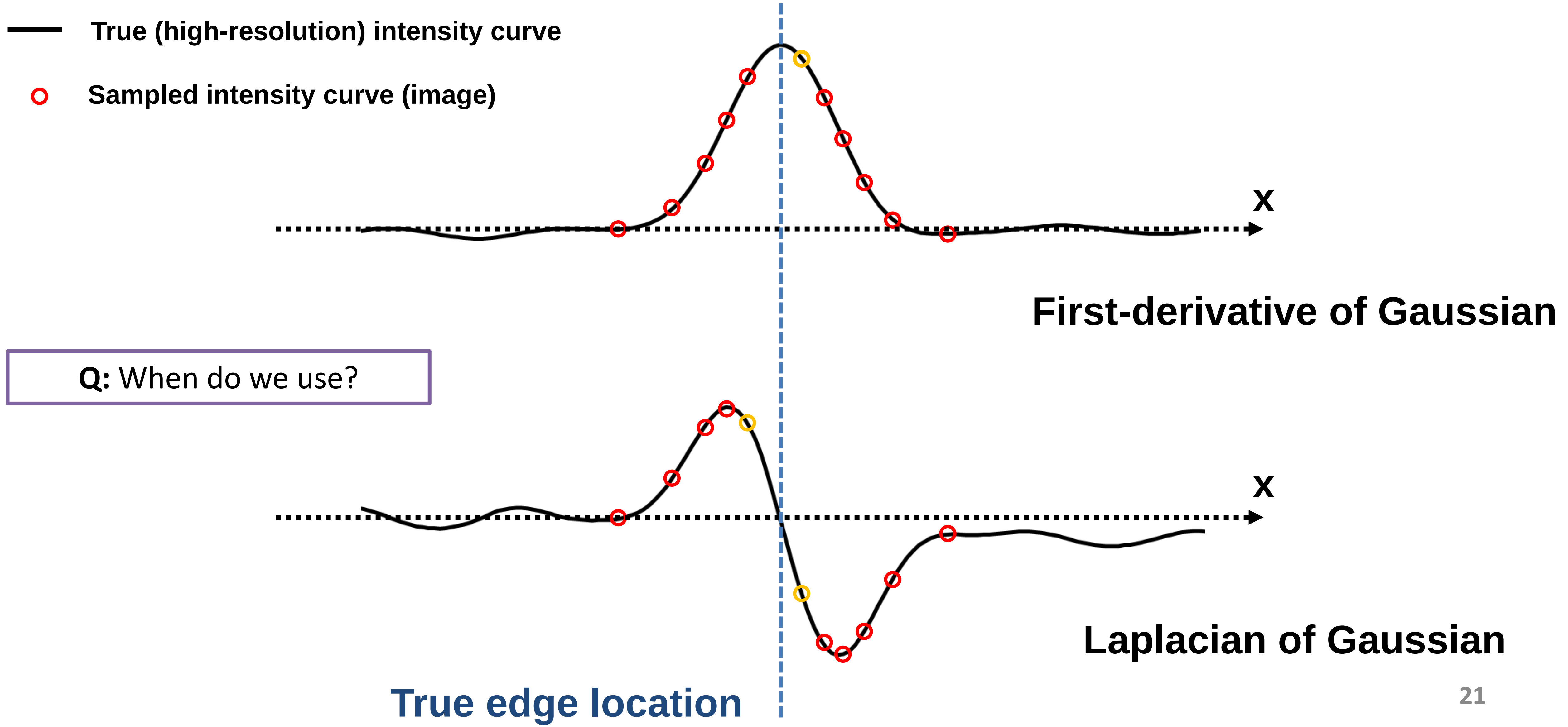
Tutorial

Laplacian of Gaussian (1D Example)



Tutorial

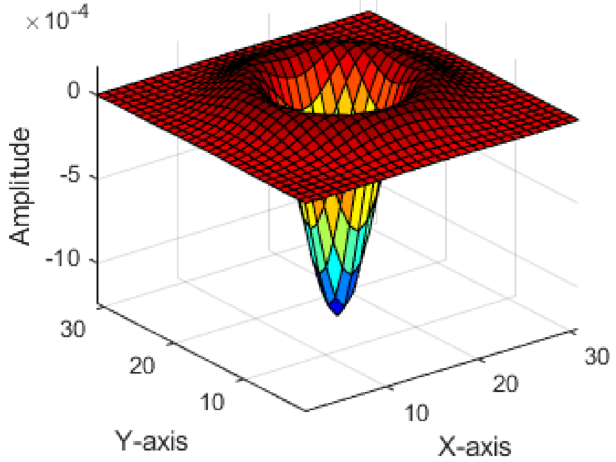
(Optional) First-derivative Gaussian vs Laplacian of Gaussian (1D Example) Supplement

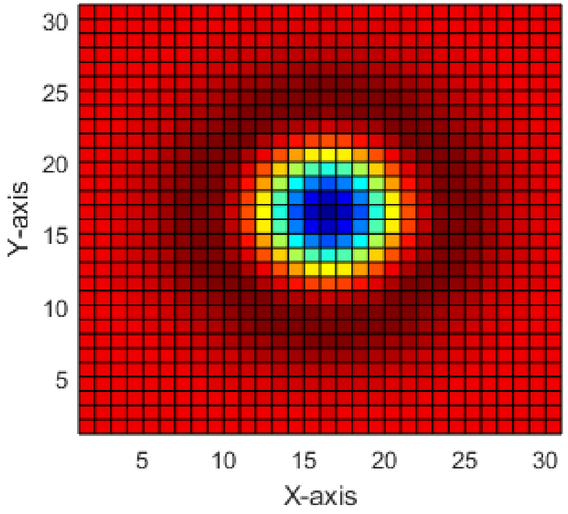


Q: When do we use?

Laplacian of Gaussian

$$\nabla^2 h_\sigma(x,y)$$





0	-1	0
-1	4	-1
0	-1	0

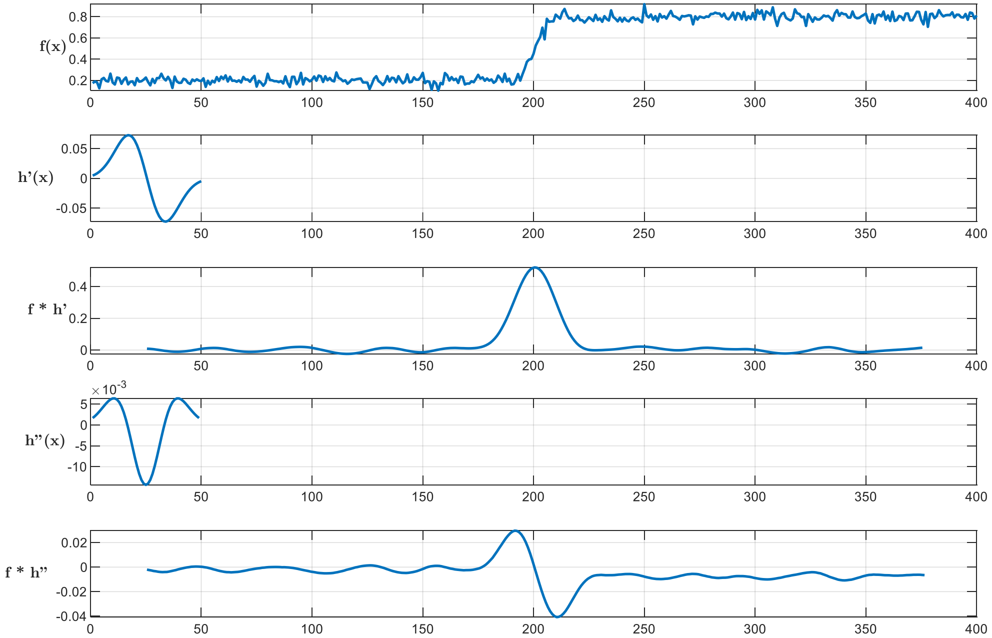
$$\nabla^2 h_\sigma(x,y) = \frac{\partial^2 h_\sigma}{\partial x^2} + \frac{\partial^2 h_\sigma}{\partial y^2}$$

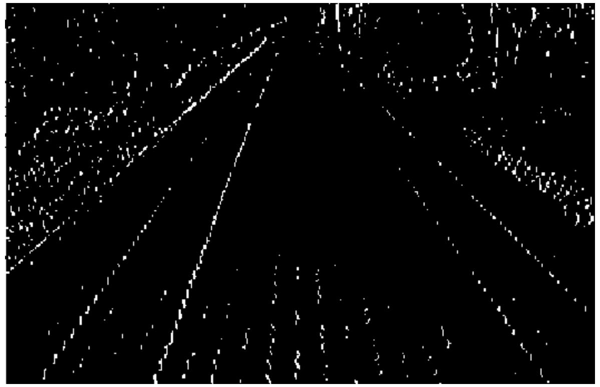
$$h_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Q: What's the effect of LoG as a linear filter?

Implementation Issues for Gradient-based Edge Detection

- The gradient magnitude is large along a thick “trail” or “ridge,” so how do we identify the actual edge points?
- How do we link the edge points to form curves?





Canny Edge Detector

- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Steps

Step 1. Filter image with x, y derivatives of Gaussian

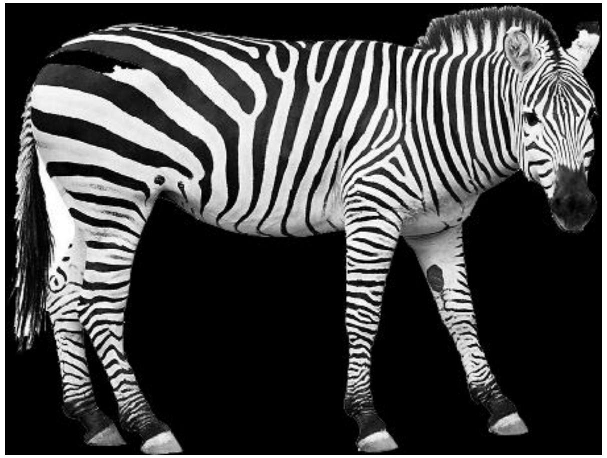
Step 2. Find magnitude and orientation of gradient

Step 3. Non-maximum suppression: Thin multi-pixel wide “ridges” down to single pixel width

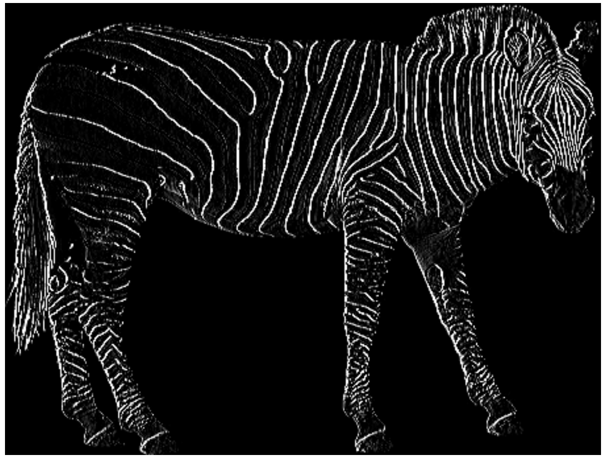
Step 4. Thresholding and linking (hysteresis): Define two thresholds: **low and high** and use the high threshold to start edge curves and the low threshold to continue them

- MATLAB: edge(image, ‘canny’)

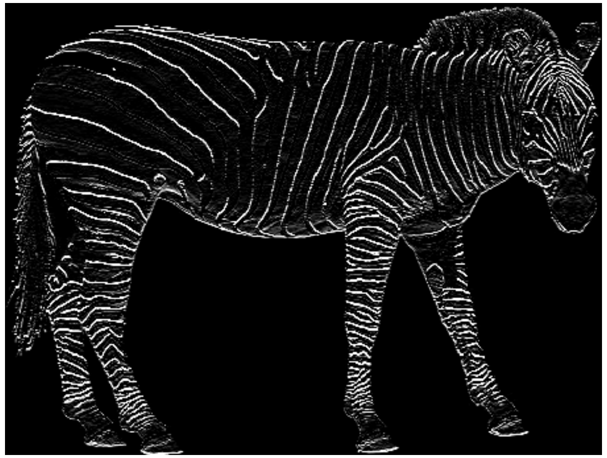
Original Image and Its Gradient



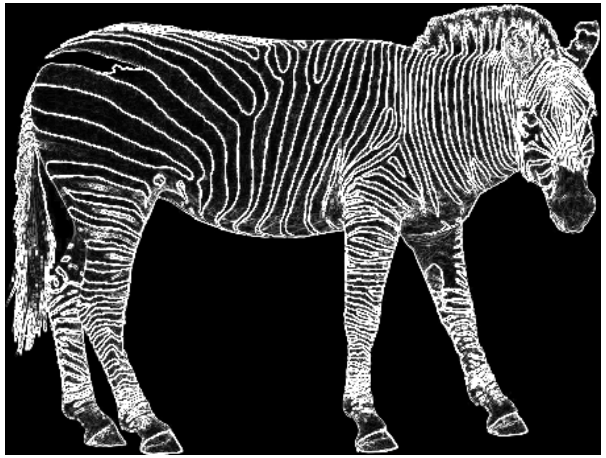
Original Image



X-Derivative of Gaussian



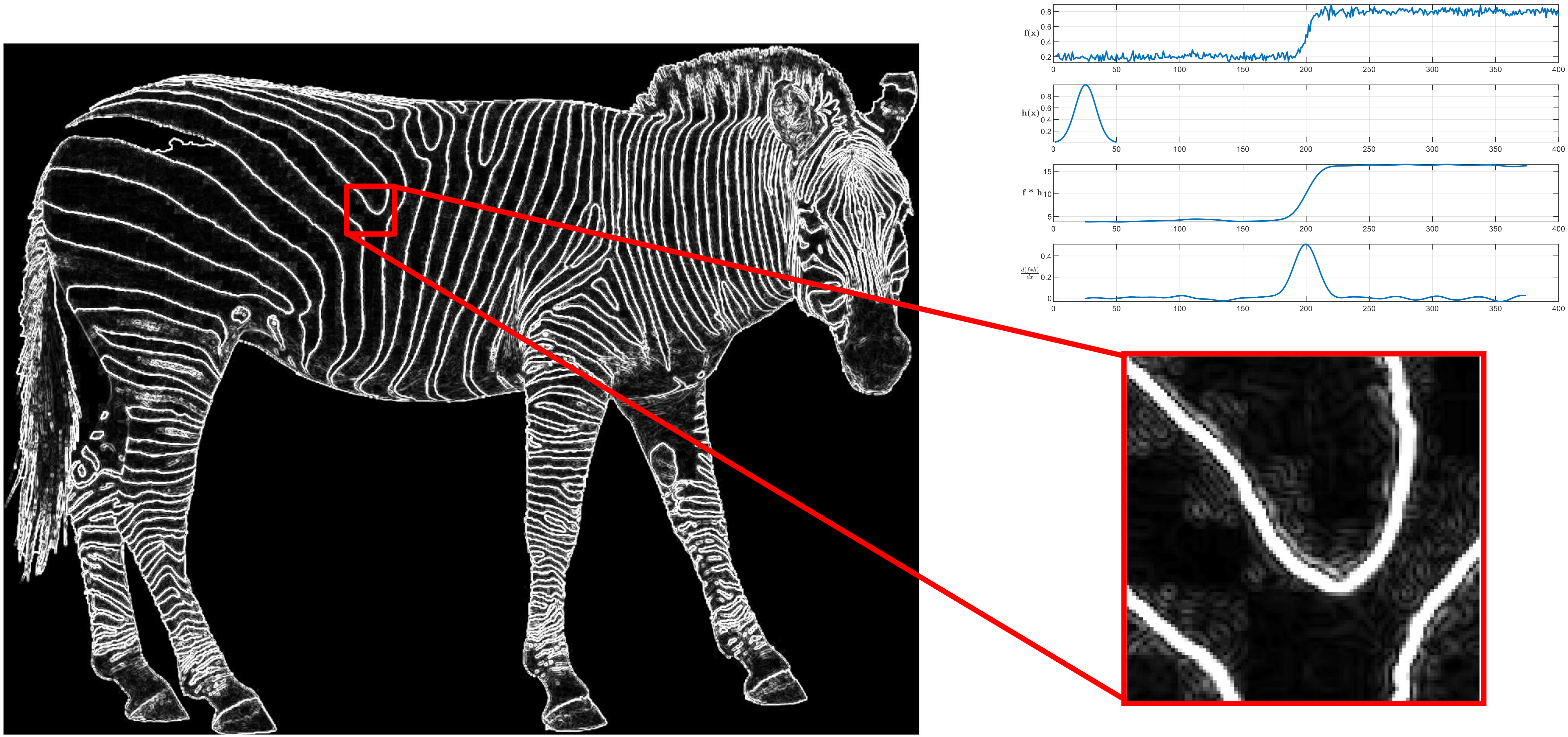
Y-Derivative of Gaussian



Gradient Magnitude

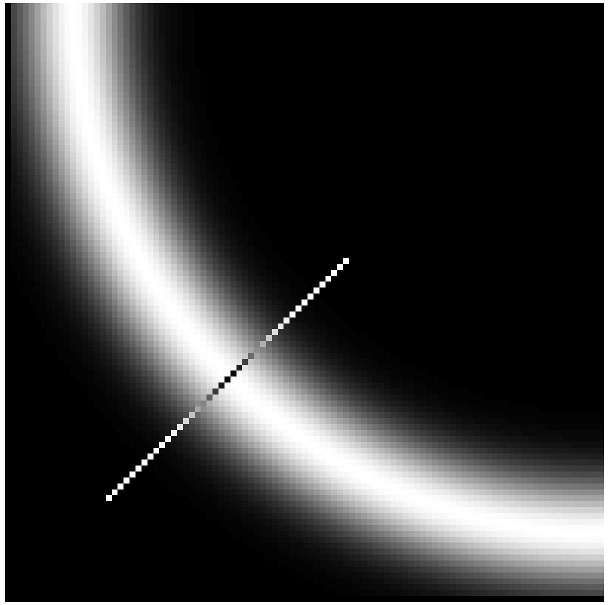
Tutorial

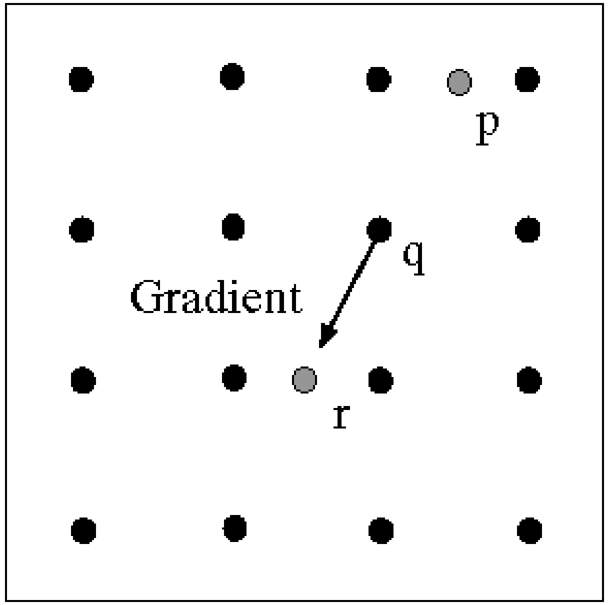
Finding Edges

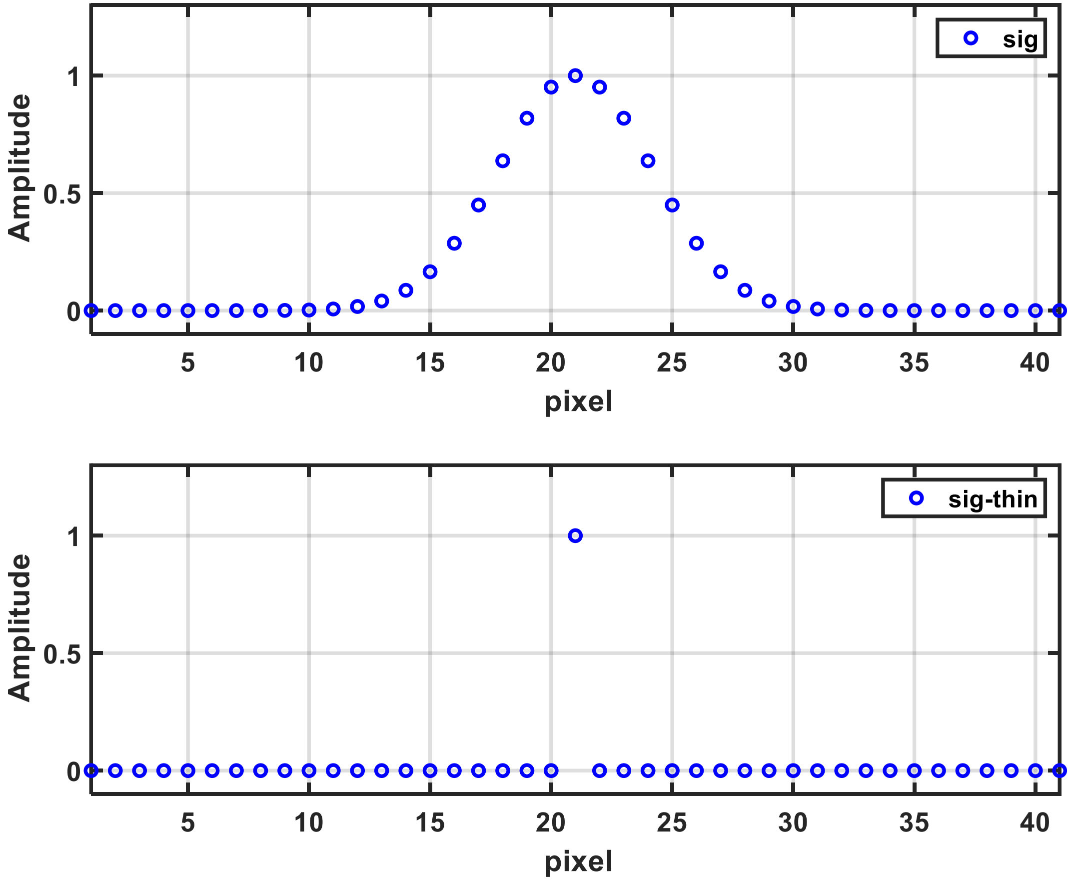


Tutorial

Non-maximum Suppression

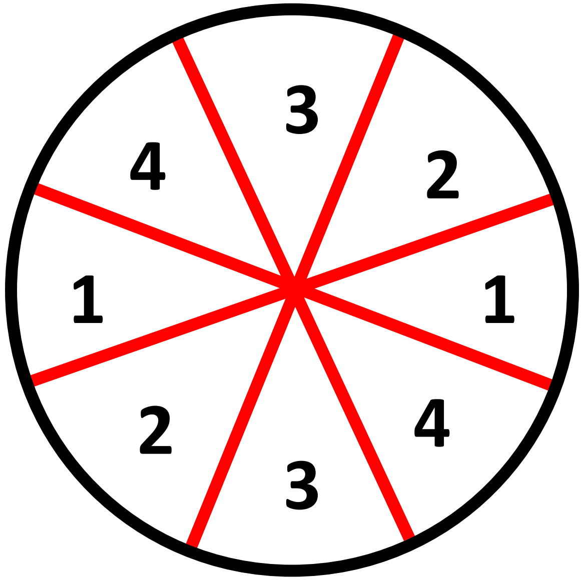


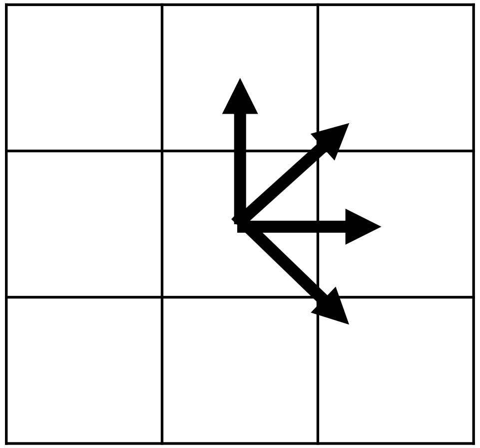


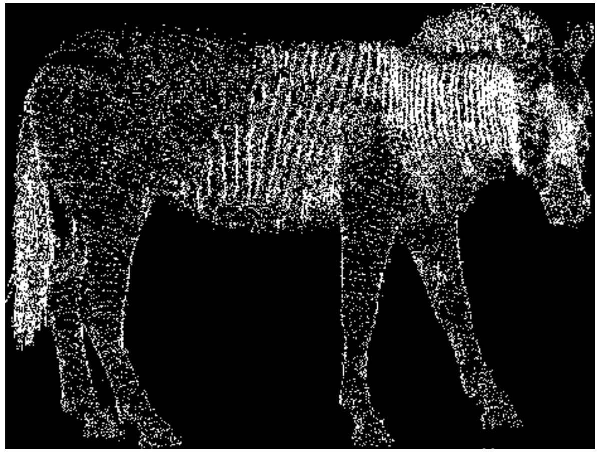


- Check if pixel is local maximum along gradient direction
  - Requires interpolating pixels p and r

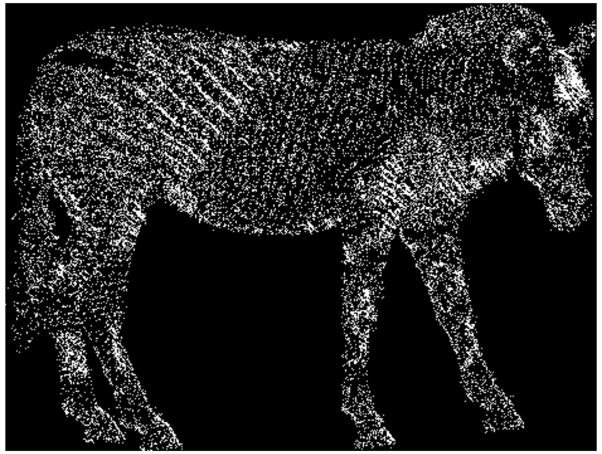
Non-maximum Suppression (Continue)



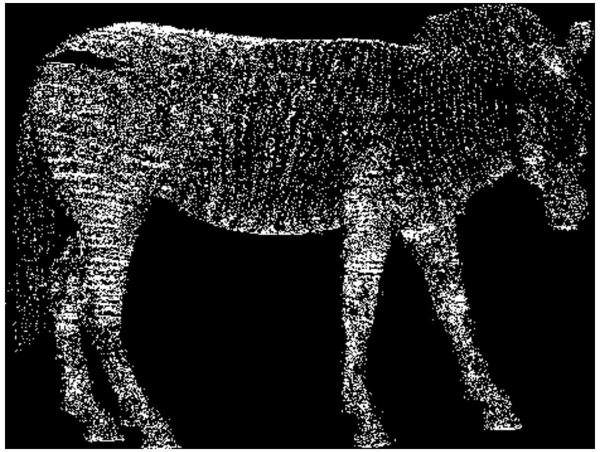




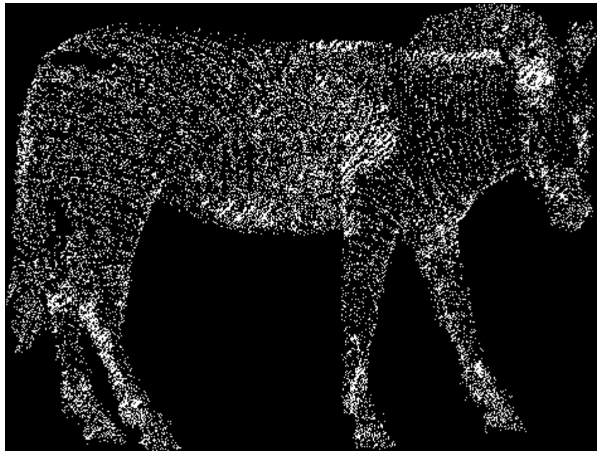
X direction (1)



45^{o} direction (2)



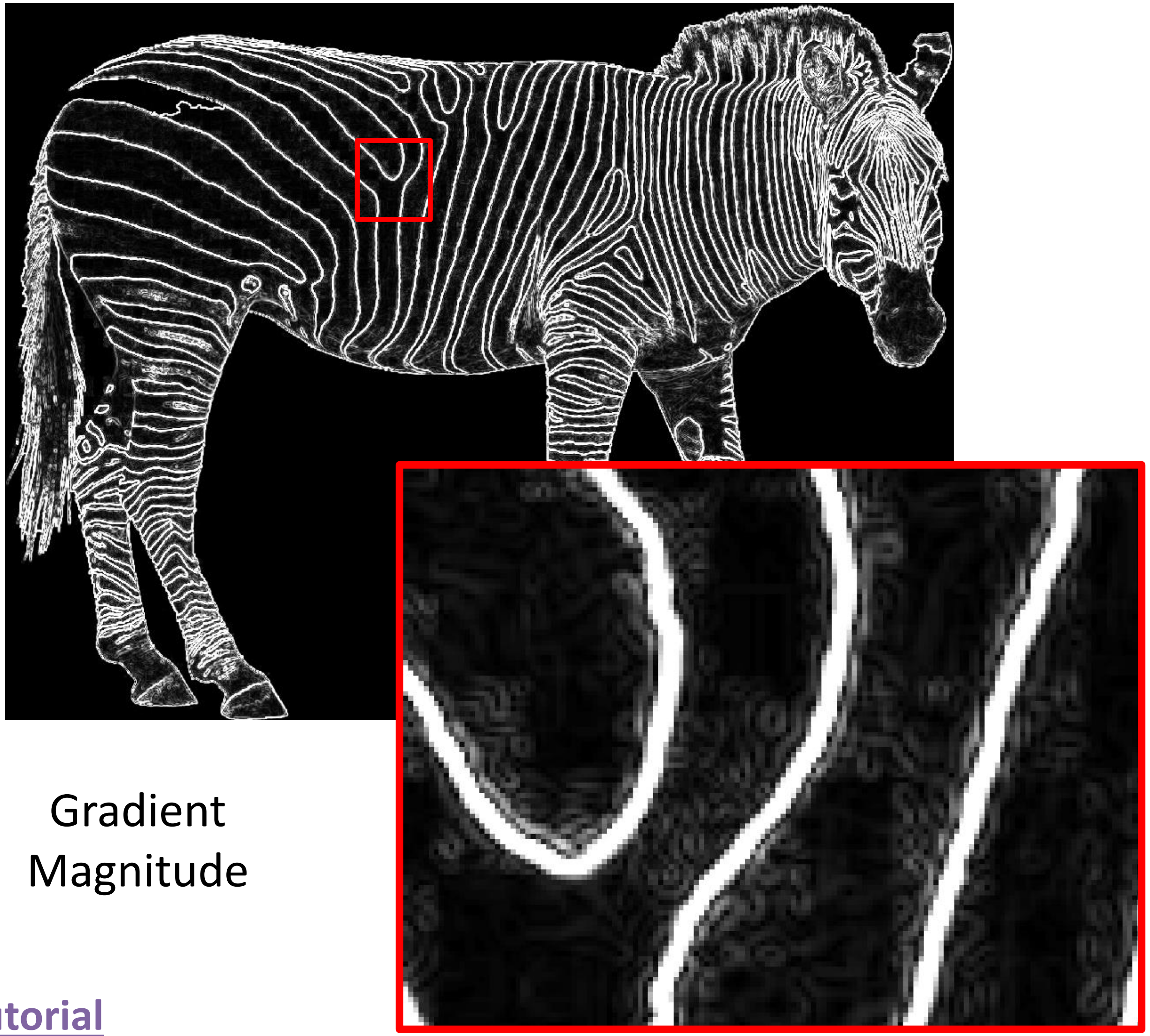
Y direction (3)



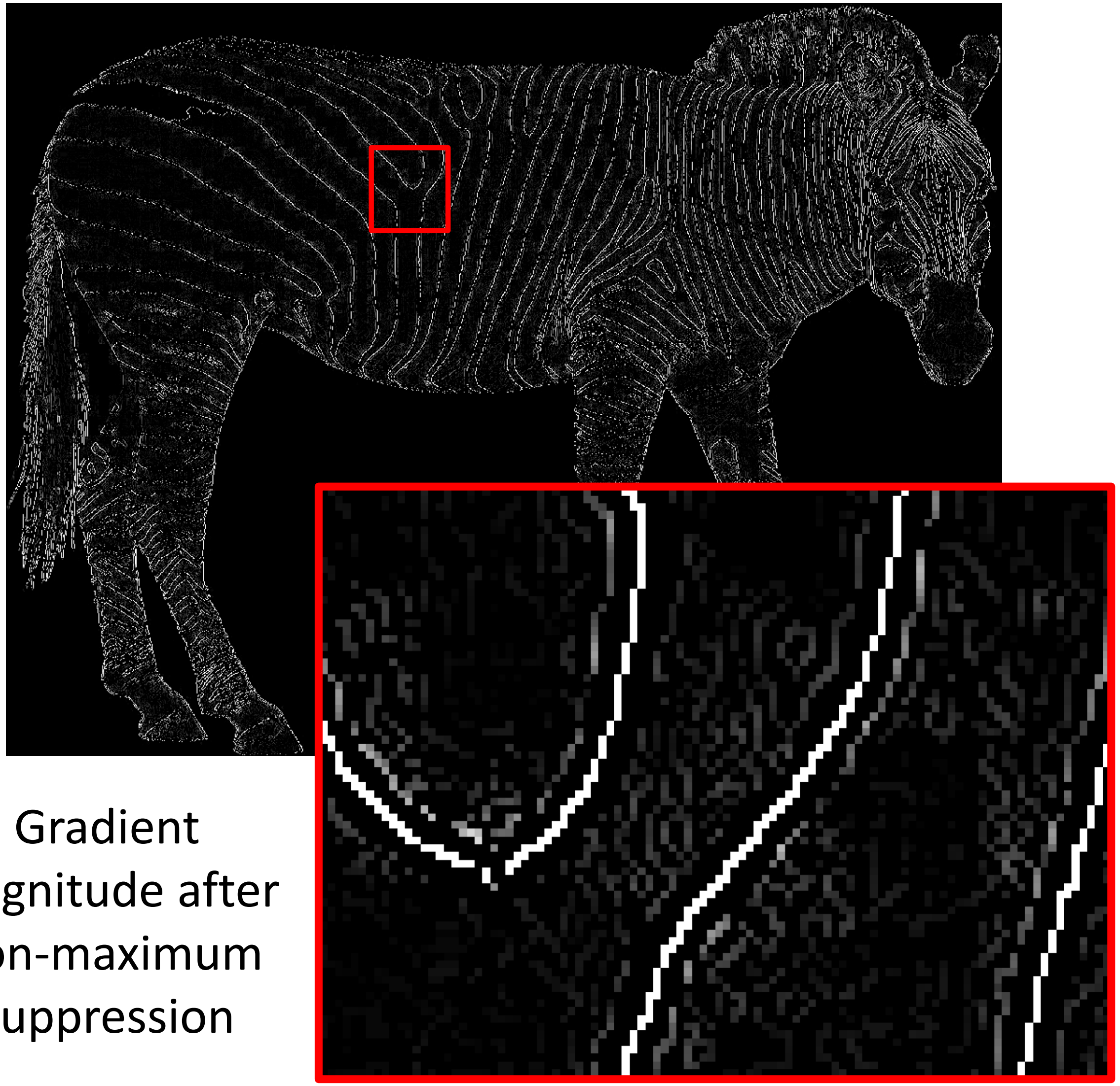
-45^{o} direction (4)

Tutorial

After Non-maximum Suppression



Gradient Magnitude



Gradient magnitude after non-maximum suppression

Tutorial

Hysteresis Thresholding

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

Finalize the detection of edges by ***suppressing all the other edges*** *that are weak and not connected to strong edges.*

Tutorial

(Revisit) Effect of Gaussian Window Sizes

Original

$f1$

σ =1

$f2$

σ =5

$f3$

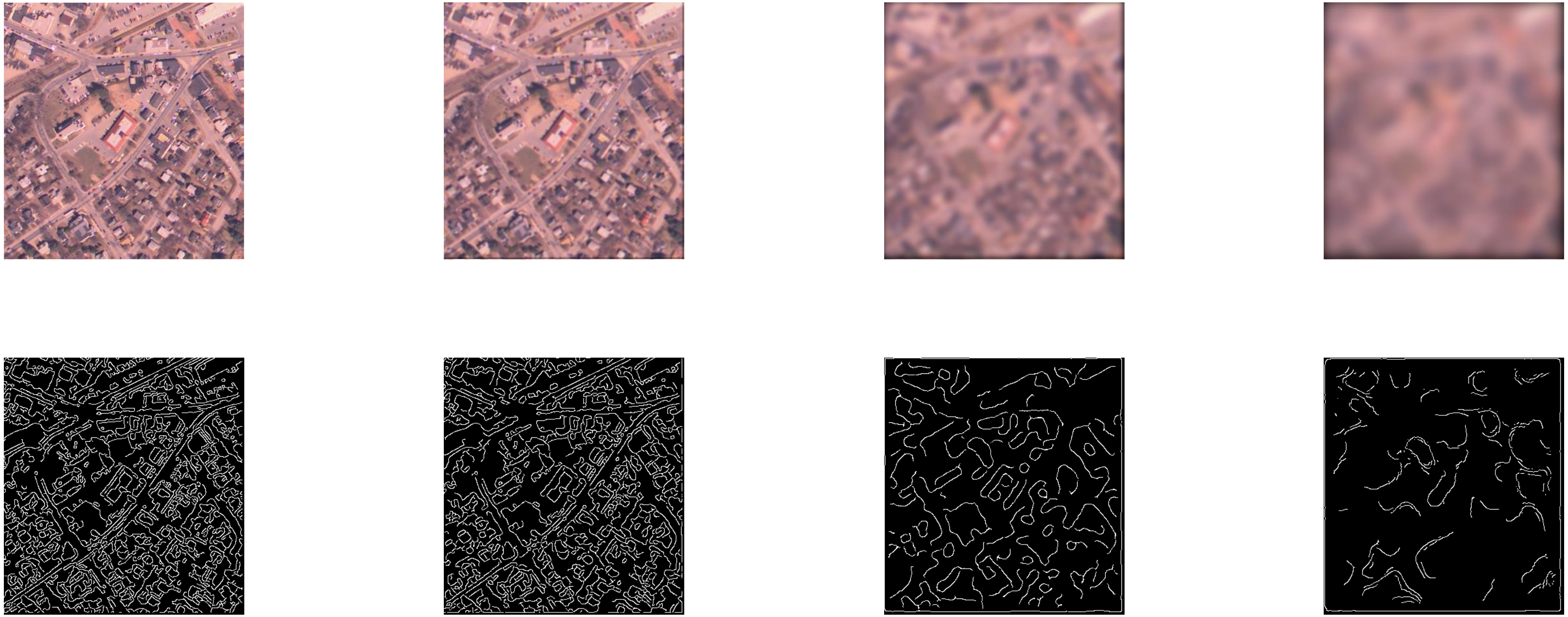
σ =10

$f4$

σ =30

```
f1 =  fspecial('gaussian', 101,1);
f2 =  fspecial('gaussian', 101,5);
f3 =  fspecial('gaussian', 101,10);
f4 =  fspecial('gaussian', 101,30);
```

Effect of σ (Gaussian Kernel Size)

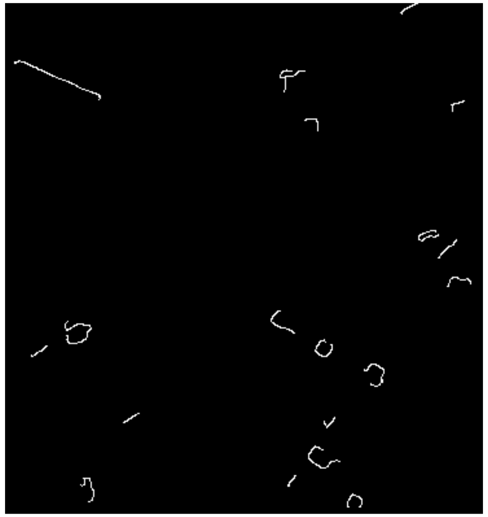


The choice of σ depends on desired behavior

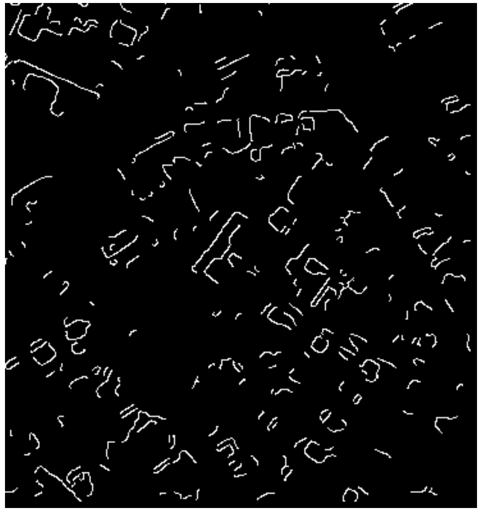
- large σ detects large scale edges
- small σ detects fine features

Tutorial

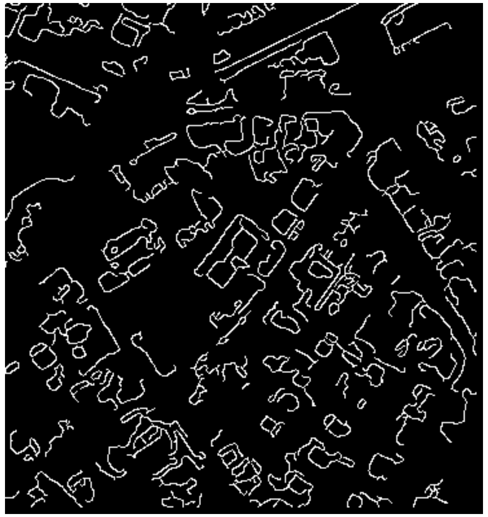
Effect of High and Low Thresholds



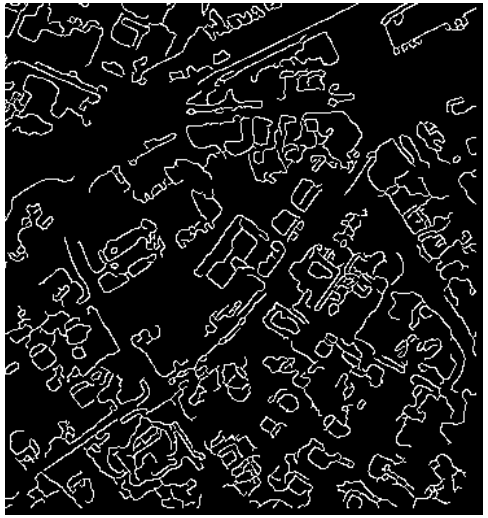
[0.4 0.8]



[0.4 0.5]



[0.2 0.5]



[0.1 0.5]

Lower a low threshold => increase weak edges

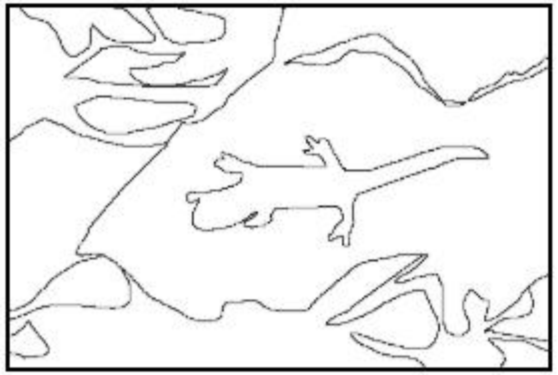
Higher a high threshold => decrease strong edges

Tutorial

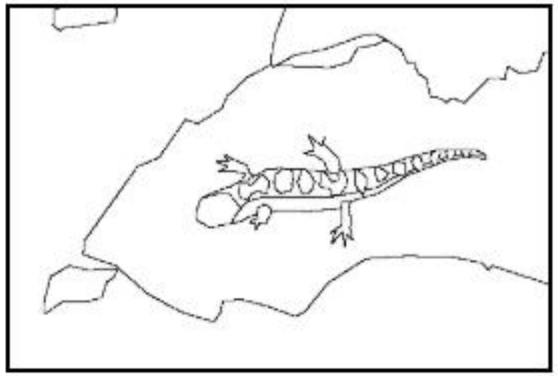
Contour Detection and Image Segmentation Resources (Computer Vision Group in Berkeley)

Supplement

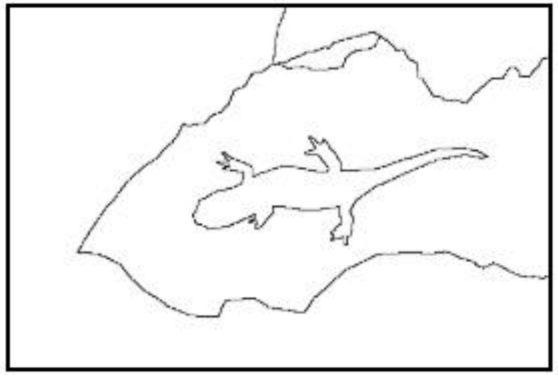
Original Image



Subject 1



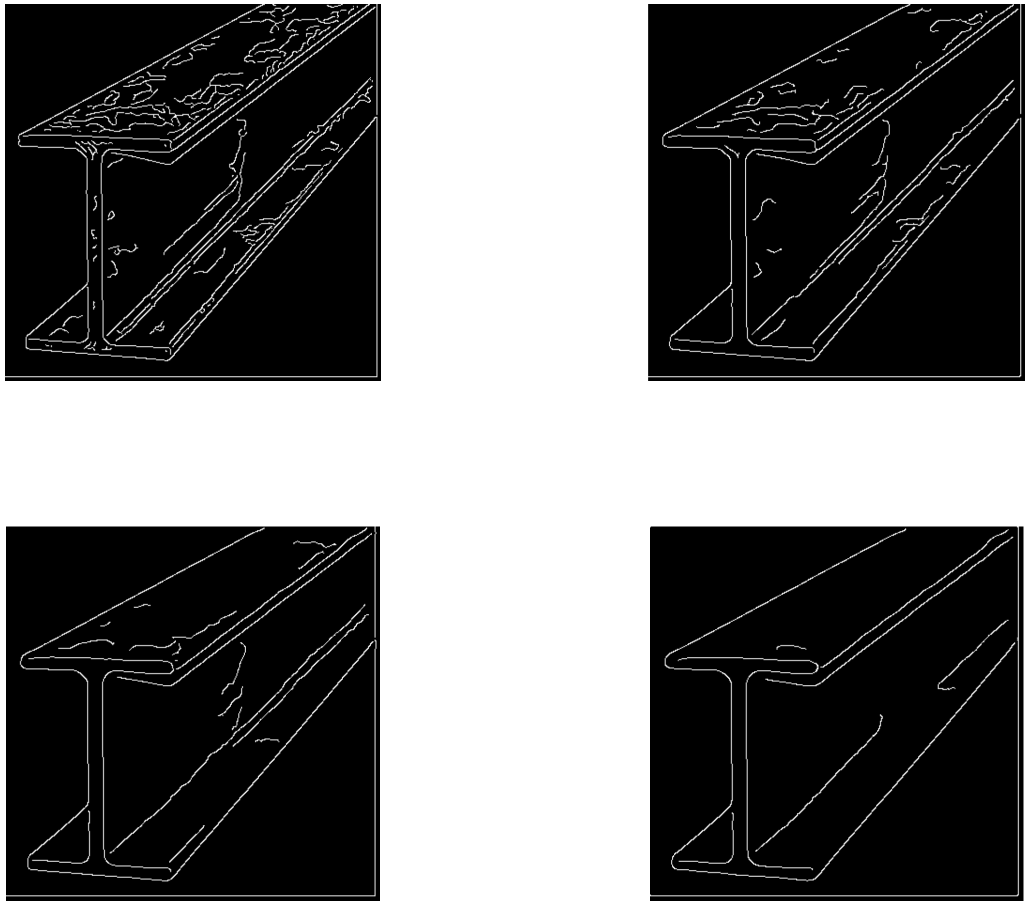
Subject 2



Subject 3

https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html

Hough Transform



The basic idea is to examine the parameter space for **lines**, rather than the image spaces.

Tutorial

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

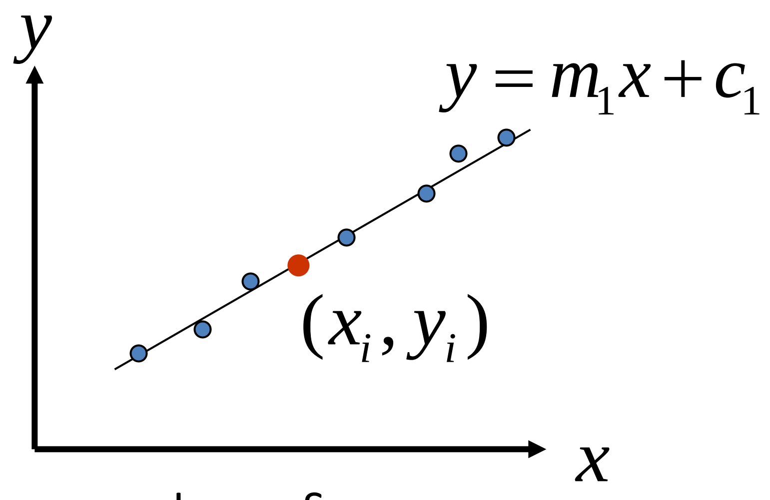
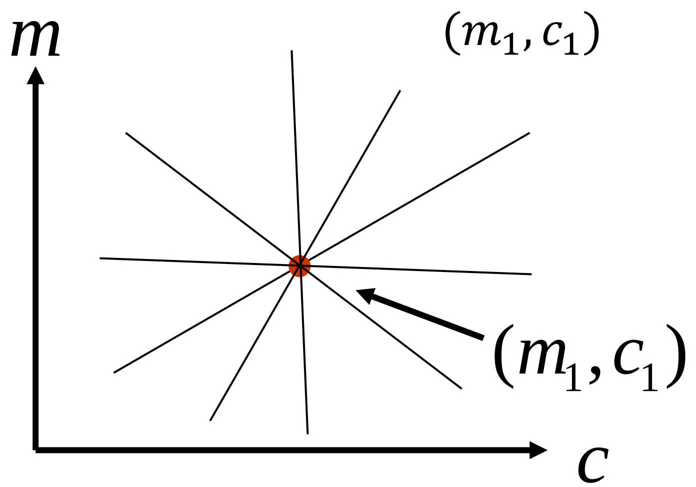


Image Space



Parameter Space

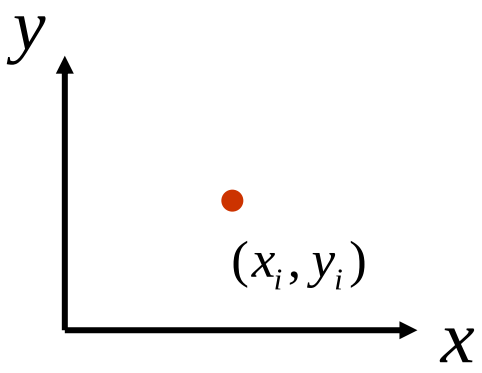
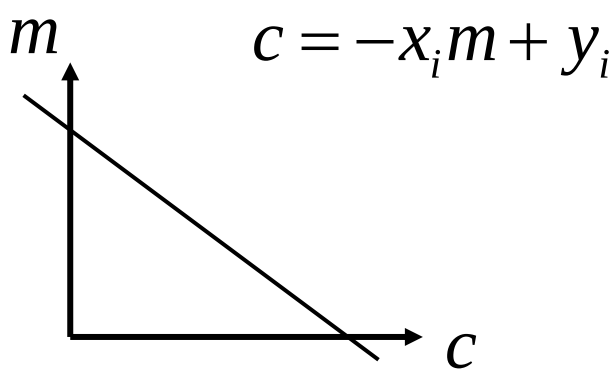
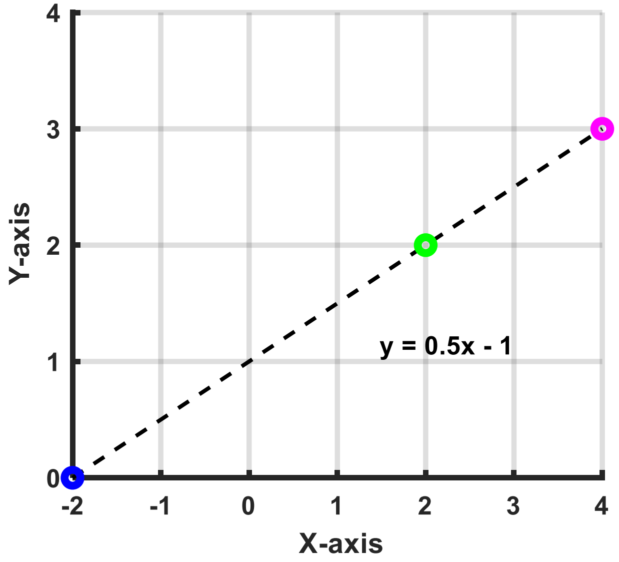


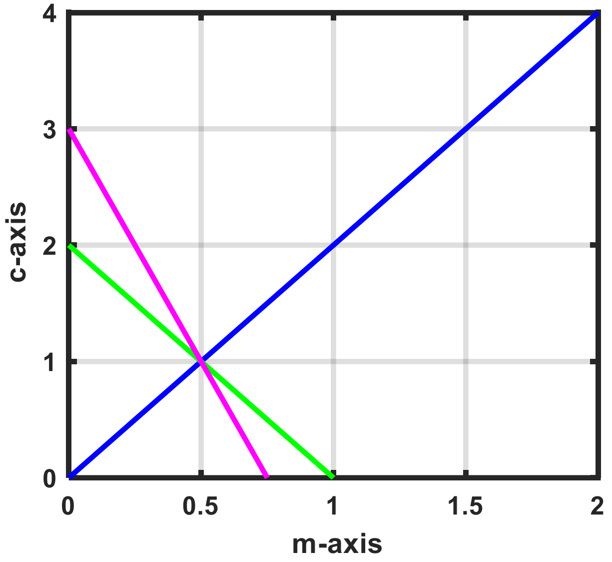
Image Space



Parameter Space

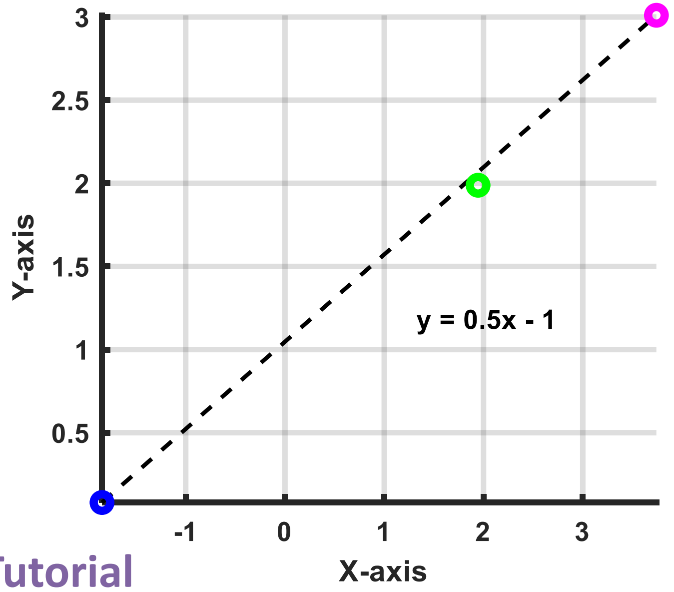
Example: Mapping Points in the Parameter Space

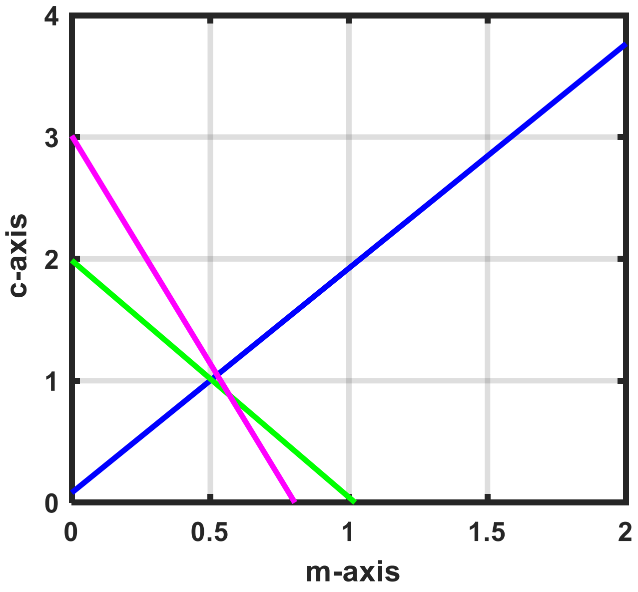




X	Y
-2	0
2	2
4	3

$$c = -2m + 0$$
$$c = 2m + 2$$
$$c = -4m + 3$$





Adding noise

X	Y
-1.84	0.08
1.95	1.99
3.74	3.01

Three lines are not intersect the same point

Q. What's the problem?

Tutorial

Line Detection by Hough Transform

Algorithm:

- Quantize Parameter Space (m, c)
- Create Accumulator Array $A(m, c)$
- Set $A(m,c) = 0 \quad \forall m, c$

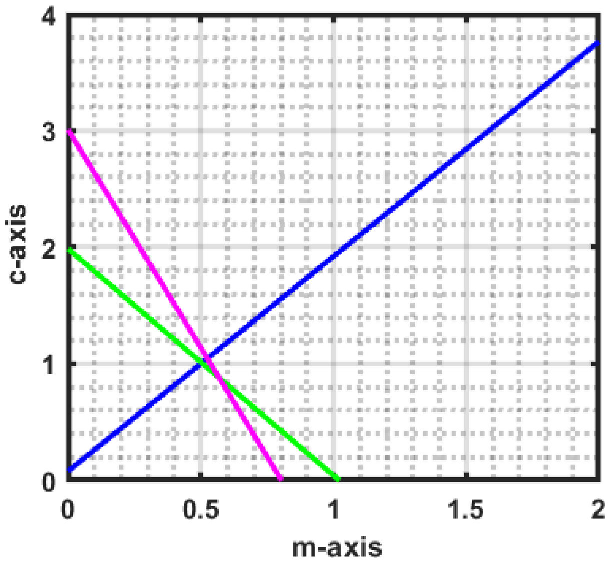
- For each image edge (x_i, y_i) increment:

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

if (m, c) lies on the line: $c = -x_i m + y_i$

- Find local maxima in $A(m, c)$

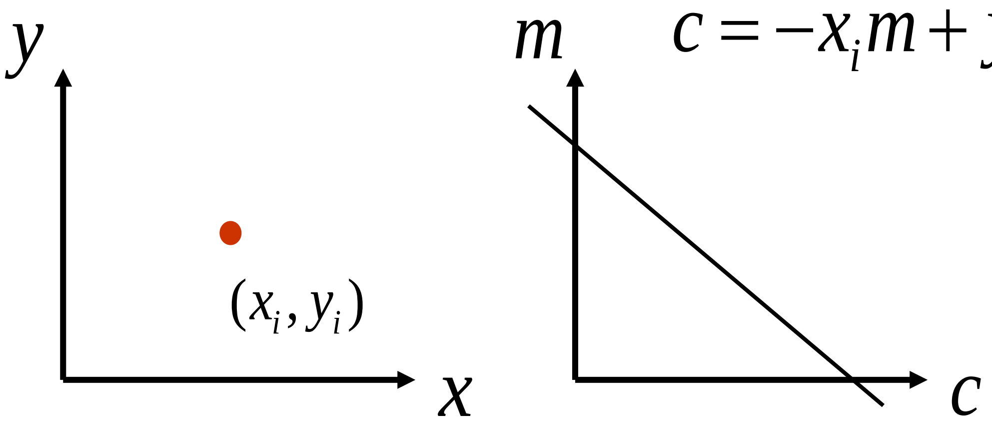
Q: What's the problem?



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

$A(m, c)$

Another Representation of a Line Equation



$$c = -x_i m + y_i$$

Image Space **Parameter (Hough) Space**

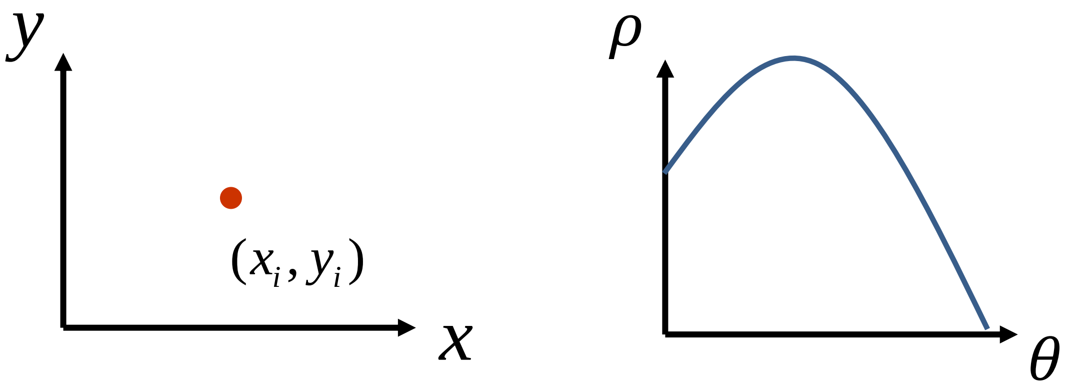
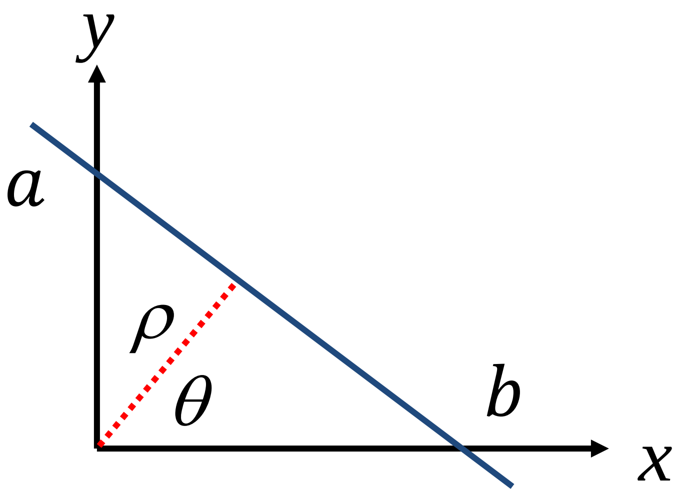


Image Space **Parameter (Hough) Space**

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



$$a = \frac{\rho}{\cos\theta}, \quad b = \frac{\rho}{\sin\theta}$$

$$m = -\frac{b}{a} = -\frac{\cos\theta}{\sin\theta}, \quad c = \frac{\rho}{\sin\theta}$$

$$\therefore\ y = -\frac{\cos\theta}{\sin\theta}x + \frac{\rho}{\sin\theta}$$

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

Better Parameterization

NOTE: $-\infty \leq m \leq \infty$

- Large accumulator
- More memory and computations

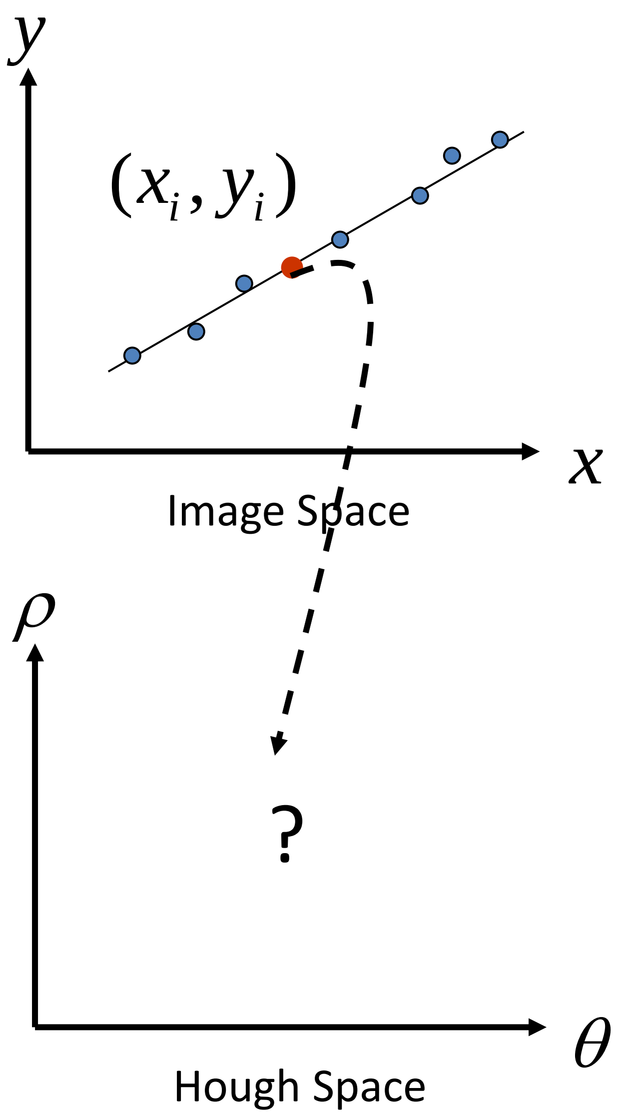
Improvement: (**Finite accumulator array size**)

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

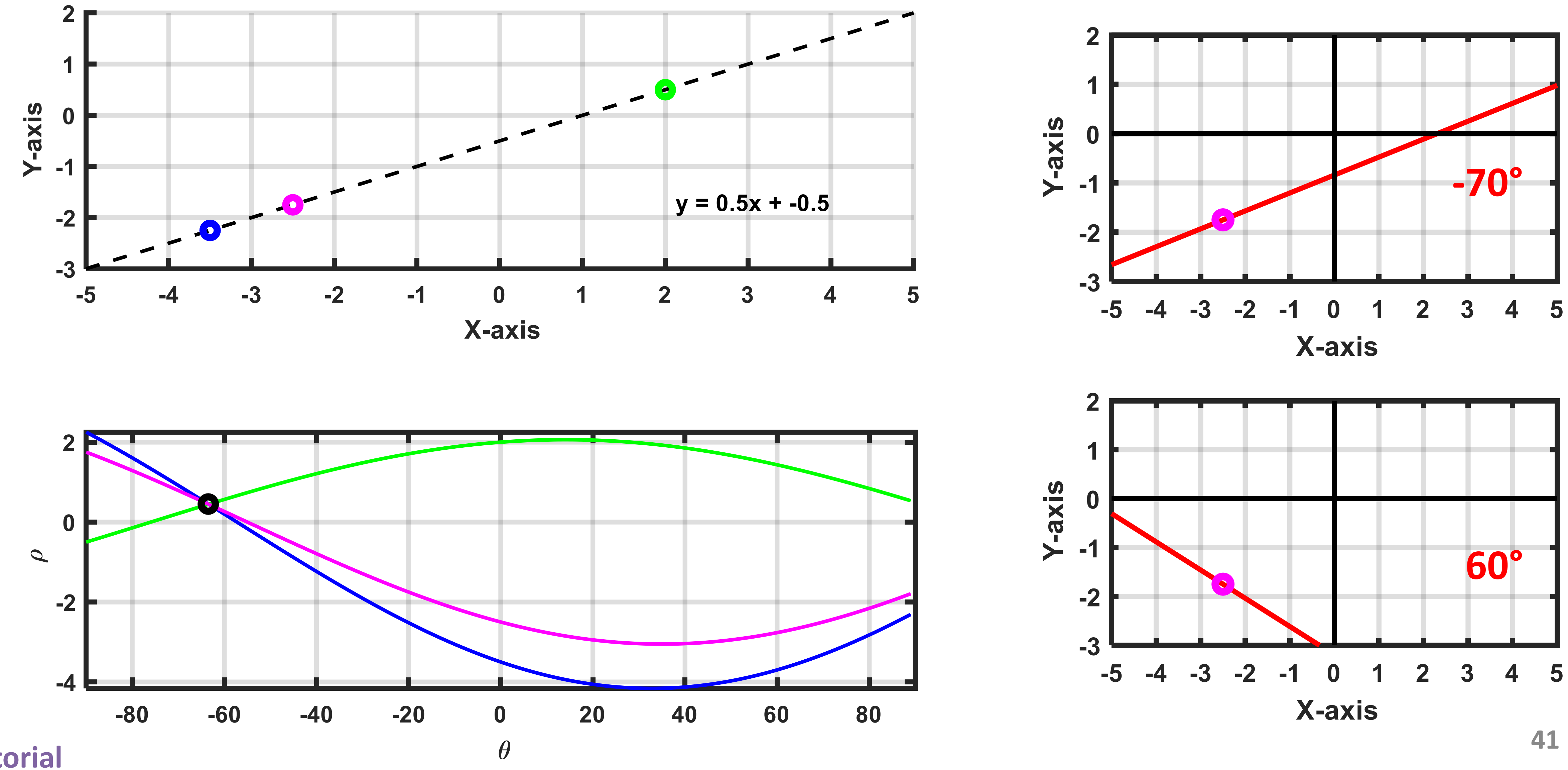
Here $-\pi/2 \leq \theta \leq \pi/2$

$-\rho_{\max} \leq \rho \leq \rho_{\max}$

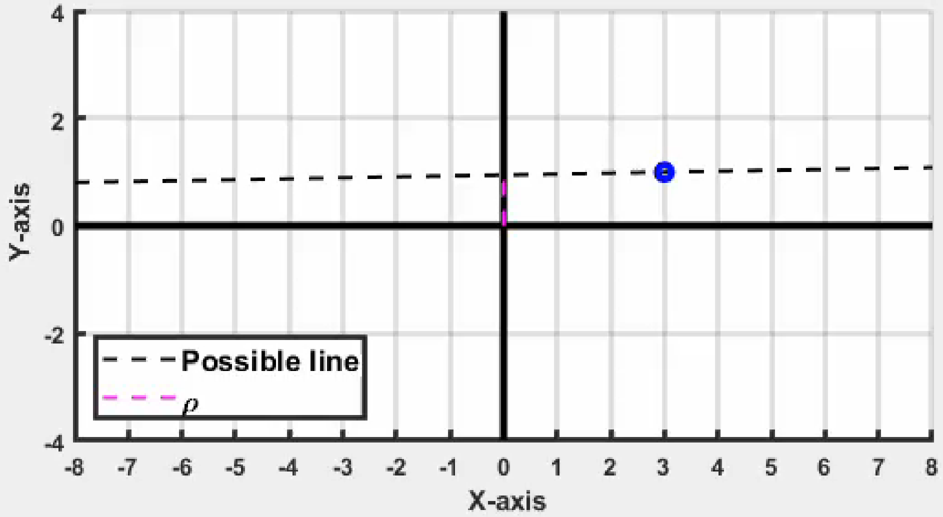
Given points (x_i, y_i) find (x_i, y_i)

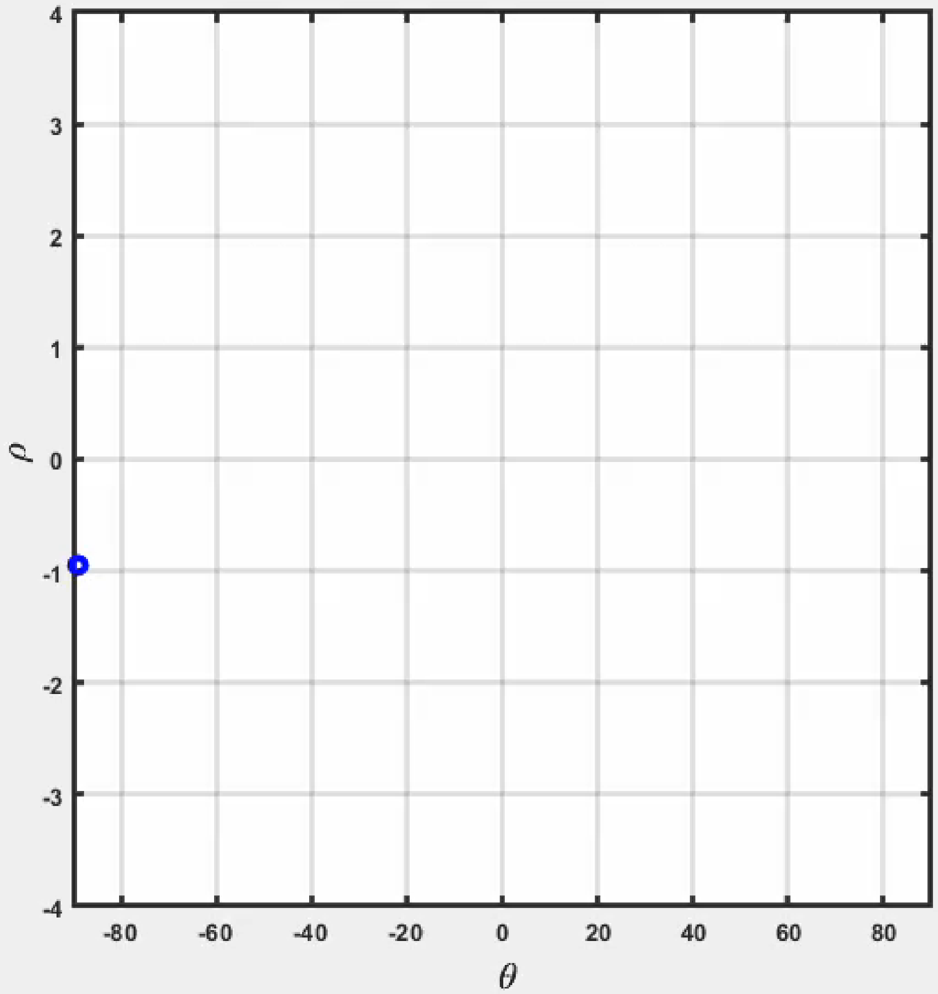


Example: Line Passing Through Three Dots



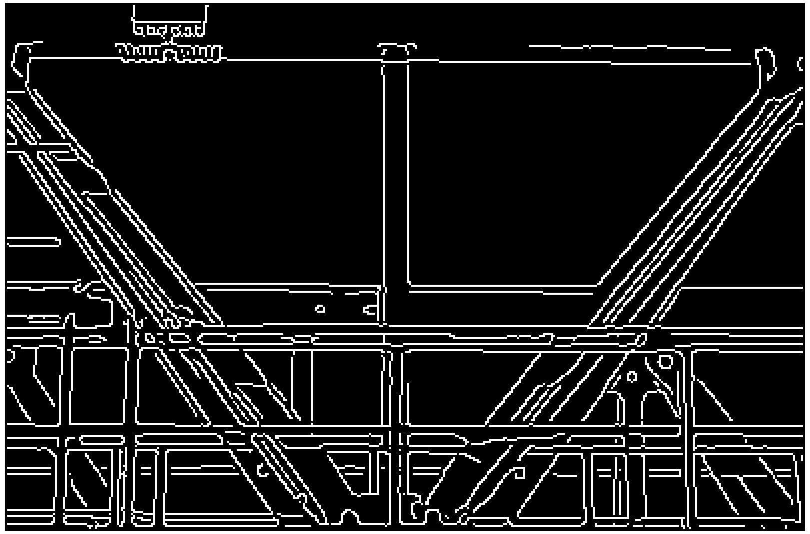
Hough Transform Demo





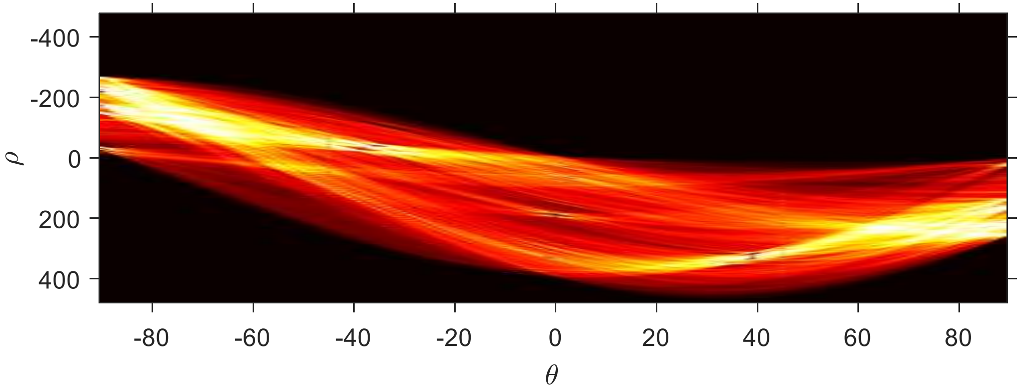
Tutorial

Example: Line Detection



Edge image (canny edge detector)

Original image



Hough space

Line from the first peak

Principles of the Hough Transform

- How many lines?
  - Count the peaks in the Hough array
  - Treat adjacent peaks as a single peak
- Which points belong to each line?
  - Search for points close to the line
  - Solve again for line and iterate
- Difficulties
  - how big should the cells be? (too big, and we merge quite different lines; too small, and noise causes lines to be missed)

Slide Credits and References

- Lecture notes: Gordon Wetzstein
- Lecture notes: Noah Snavely
- Lecture notes: L. Fei-Fei
- Lecture notes: D. Frosyth
- Lecture notes: James Hayes
- Lecture notes: Yacov Hel-Or