UCF CENTER FOR RESEARCH IN COMPUTER VISION

CAP5415 Computer Vision

Yogesh S Rawat

yogesh@ucf.edu

HEC-241

Questions?

Edge Detection

Lecture 4

Outline

- What is edge detection?
- Why we need edge detection?
- Challenges
  - Noise
- How to detect edges?
  - Prewitt
  - Sobel
  - Marr-Hildreth
  - Canny

Edge Detection

Lecture 4

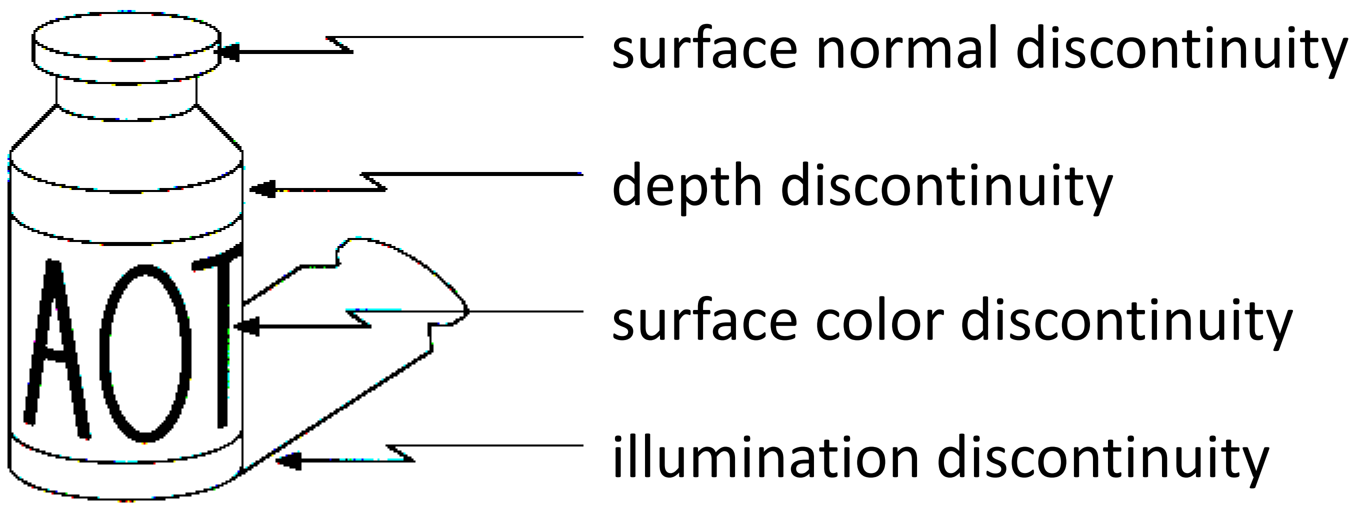
Basics of edge detection

Edge Detection

- Identify sudden changes in an image
  - Semantic and shape information
  - Marks the border of an object
  - More compact than pixels

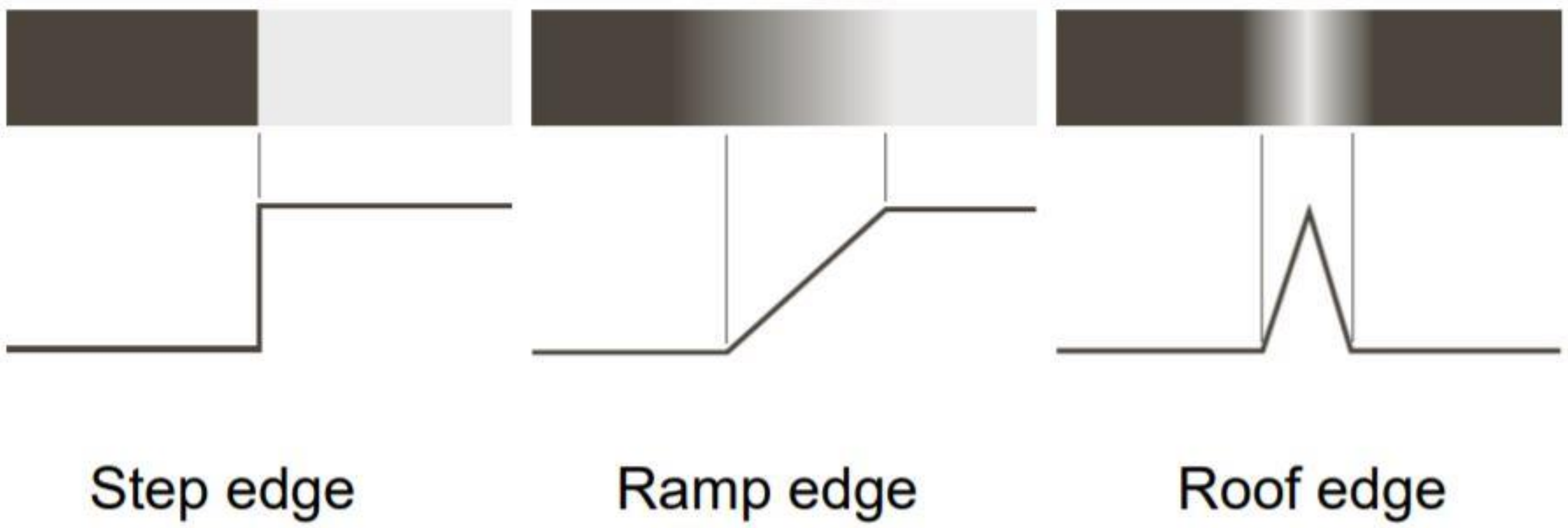
Origins of Edges

- Edges are caused by a variety of factors



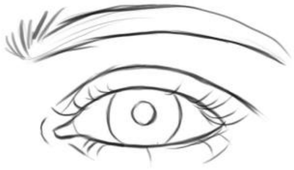
Types of edges

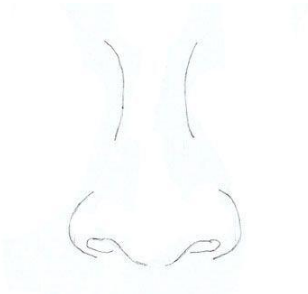
- Edge models

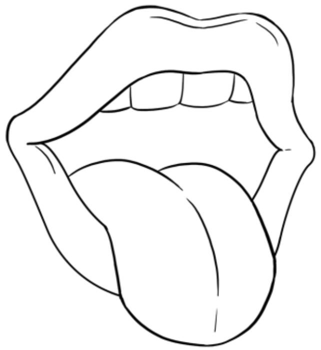


Why edge detection?

- Extract useful information from images
  - Recognizing objects
- Recover geometry



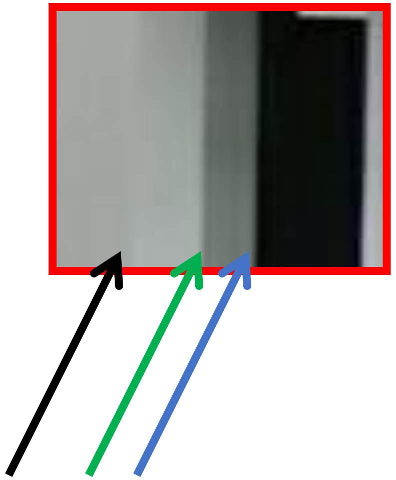




Closeup of edges

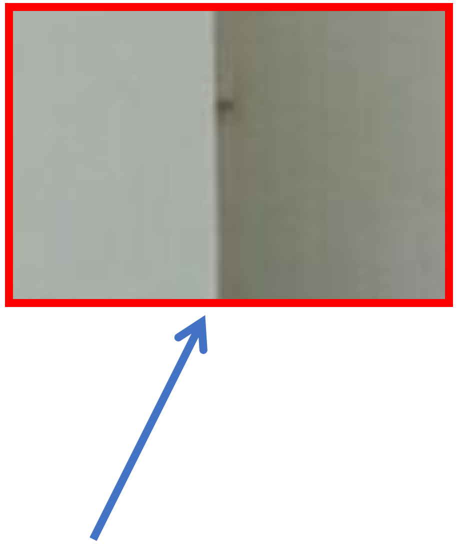
Source: D. Hoiem

Closeup of edges



Source: D. Hoiem

Closeup of edges



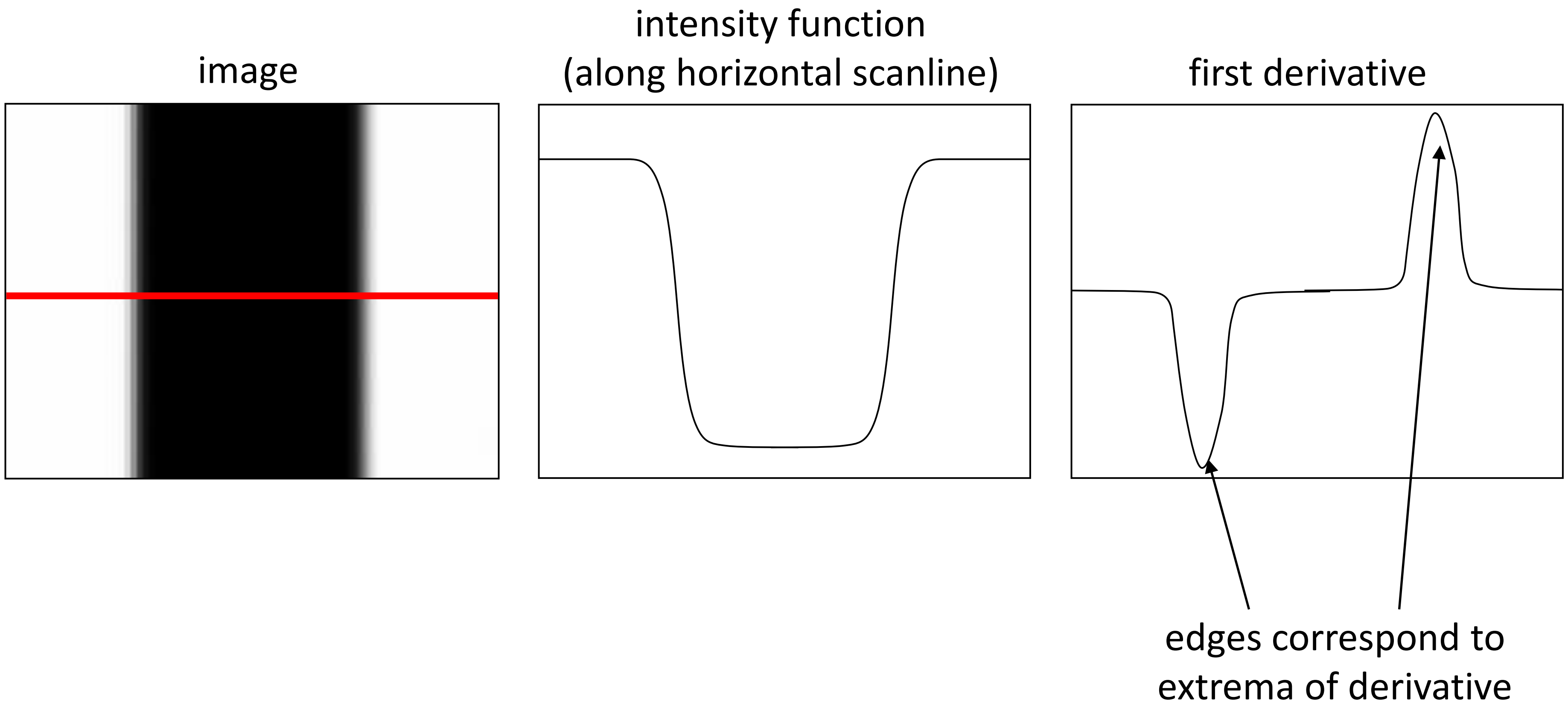
Source: D. Hoiem

Closeup of edges

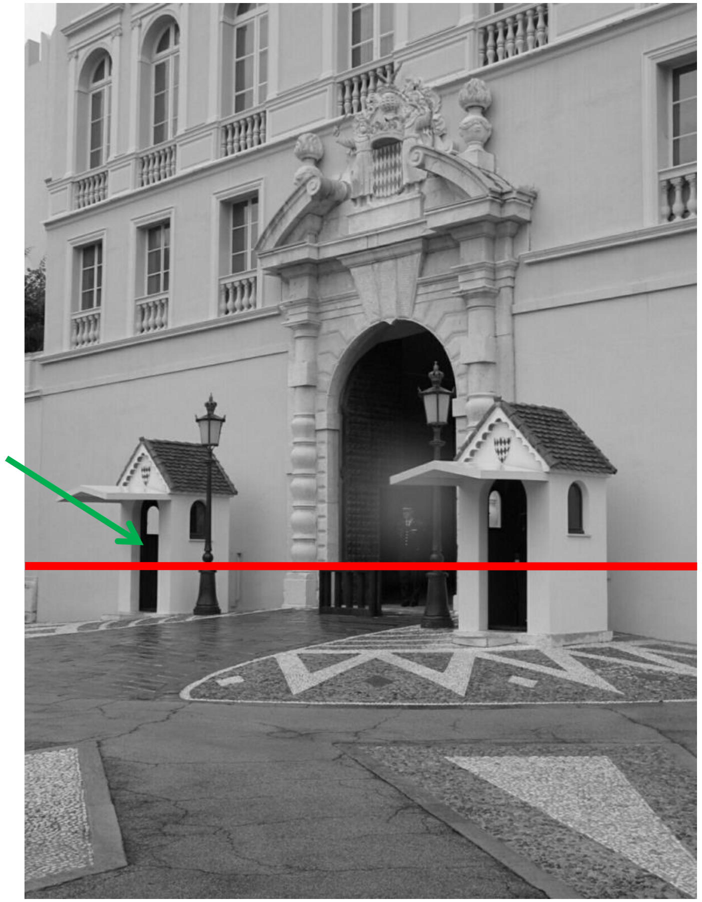
Source: D. Hoiem

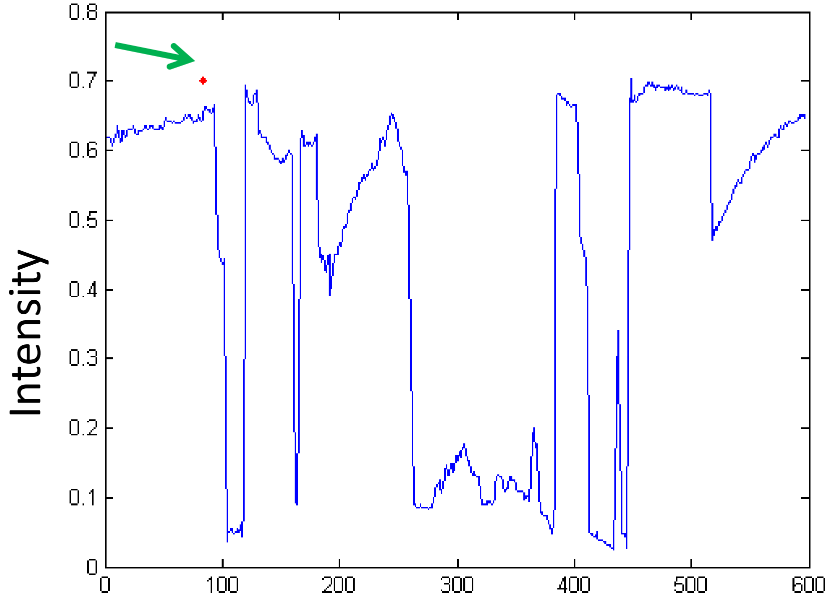
Characterizing edges

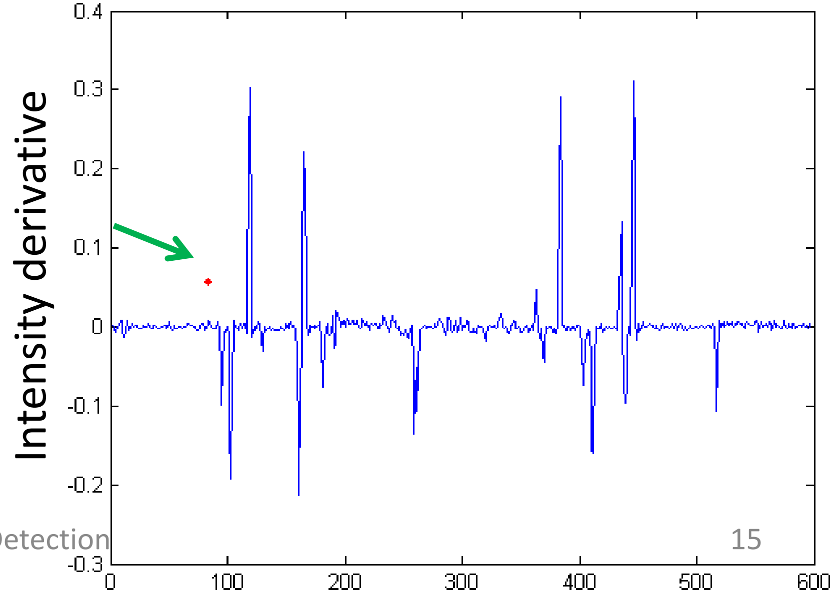
- An edge is a place of rapid change in the image intensity function



Intensity profile

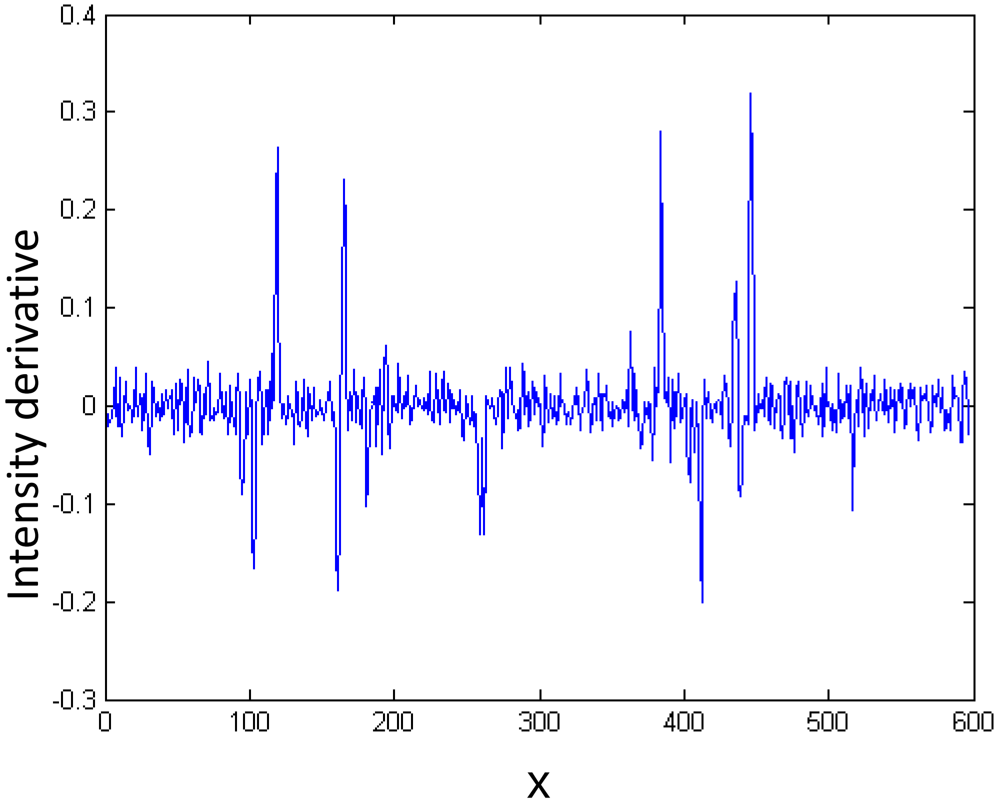






Source: D. Hoiem

With a little Gaussian noise

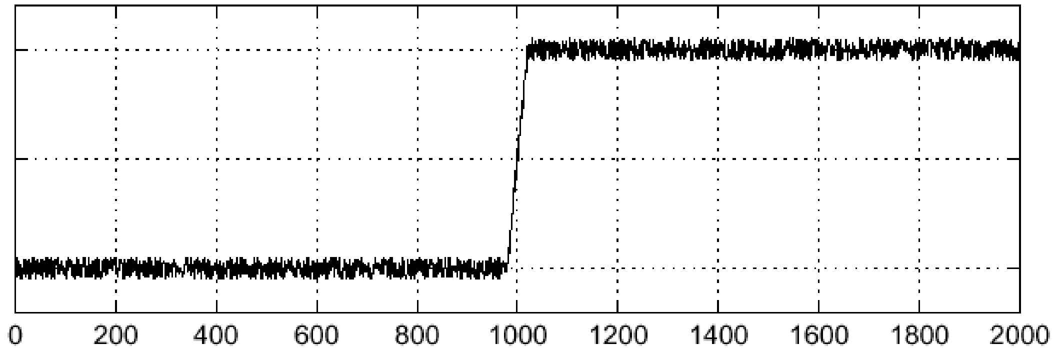


Source: D. Hoiem

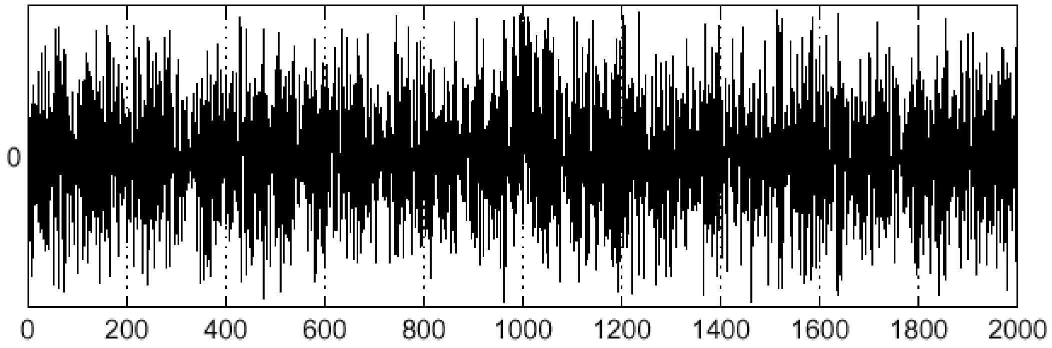
Effects of Noise

- 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?

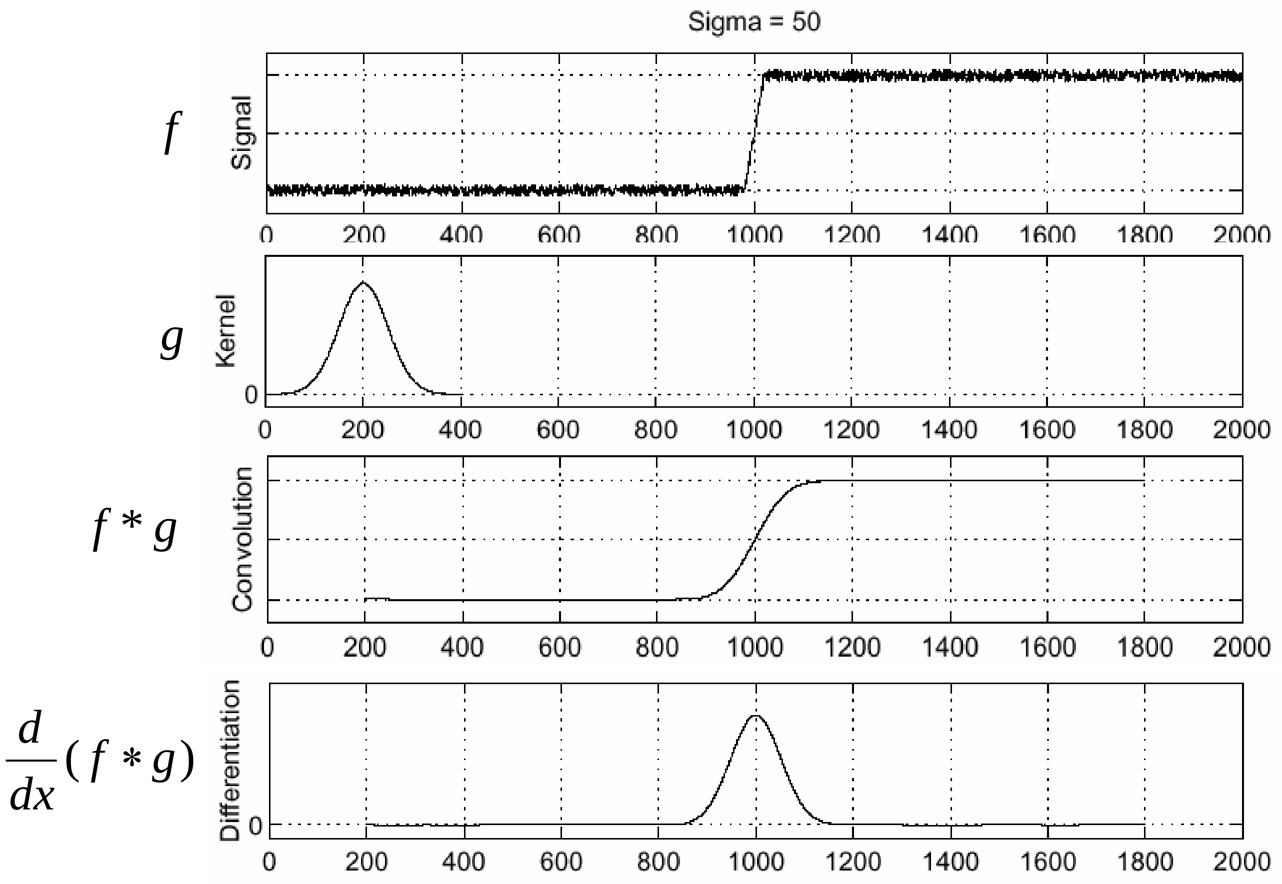
Credit to Seitz.

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?

Source: D. Forsyth

Solution: smooth first



To find edges, look for peaks in $$\frac{d}{dx}(f * g)$$

Source: S. Seitz

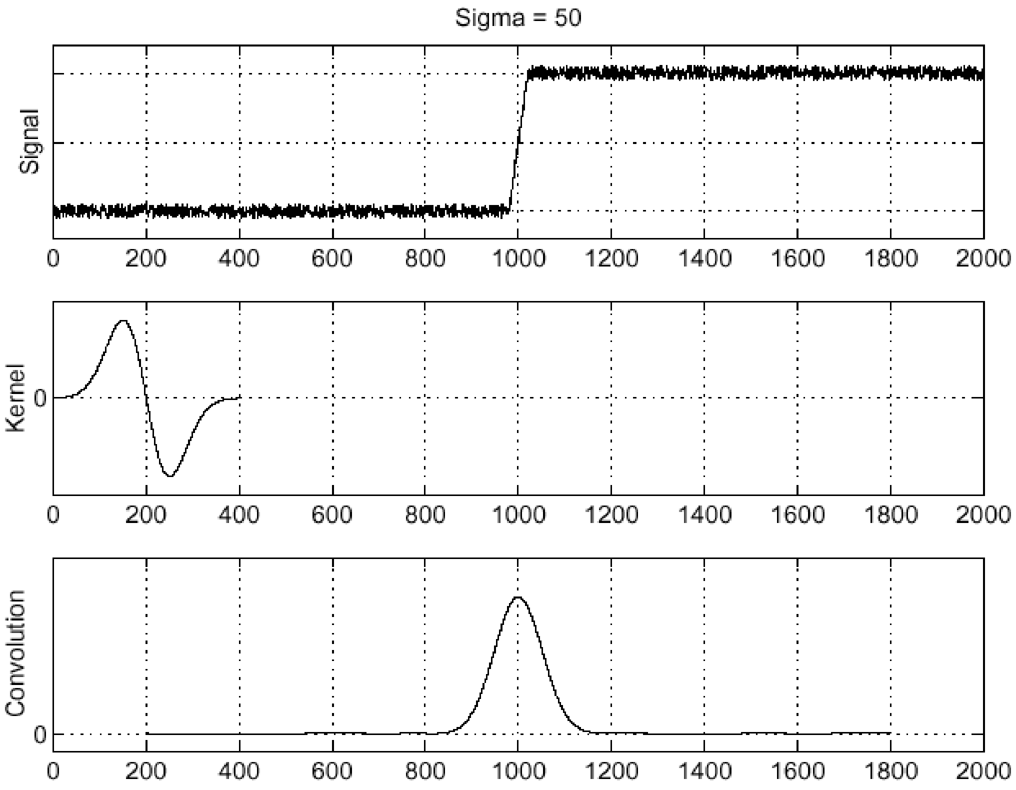
Derivative theorem of convolution

- Convolution is differentiable: $$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$
- This saves us one operation:

f

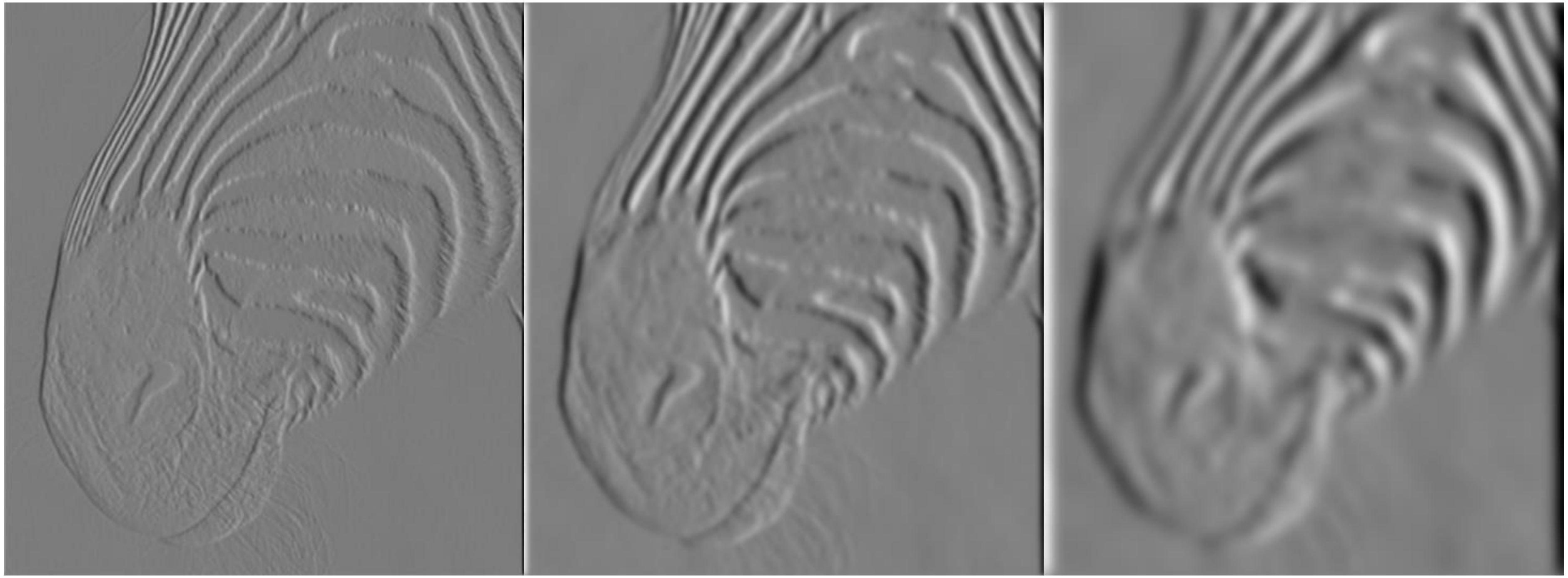
$\frac{d}{dx} g$

$f * \frac{d}{dx} g$



Source: S. Seitz

Solution: Smoothing



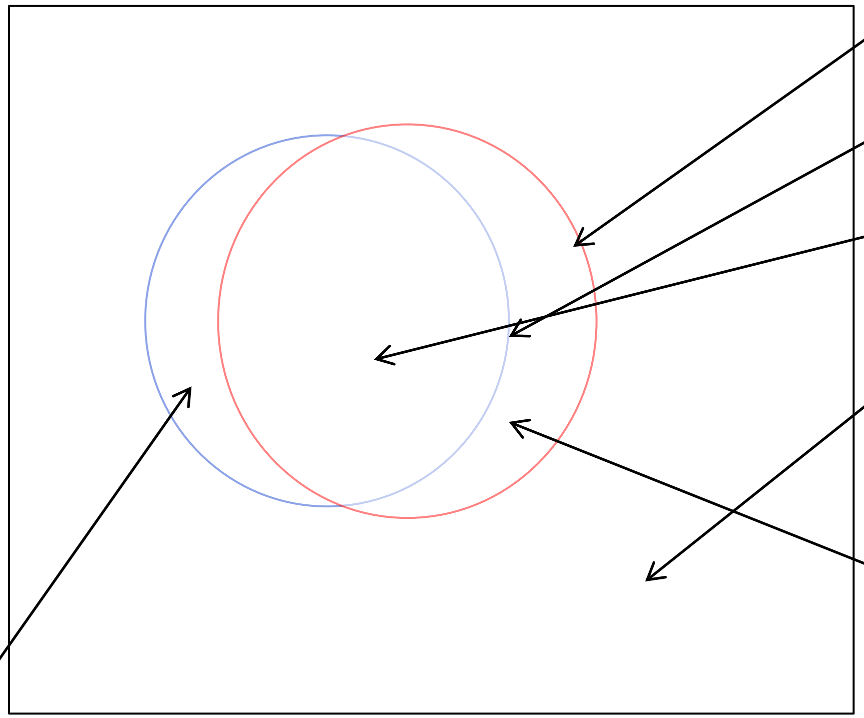
1 pixel 3 pixels 7 pixels

- Smoothing removes noise, but blurs edge.

Evaluate Edge Detection

$$\text{precision} = \frac{GT \bigcap RM}{RM} = \frac{TP}{RM}$$

$$\text{recall} = \frac{GT \bigcap RM}{GT} = \frac{TP}{GT}$$



Ground Truth (GT)

Results of Method (RM)

True Positives (TP)

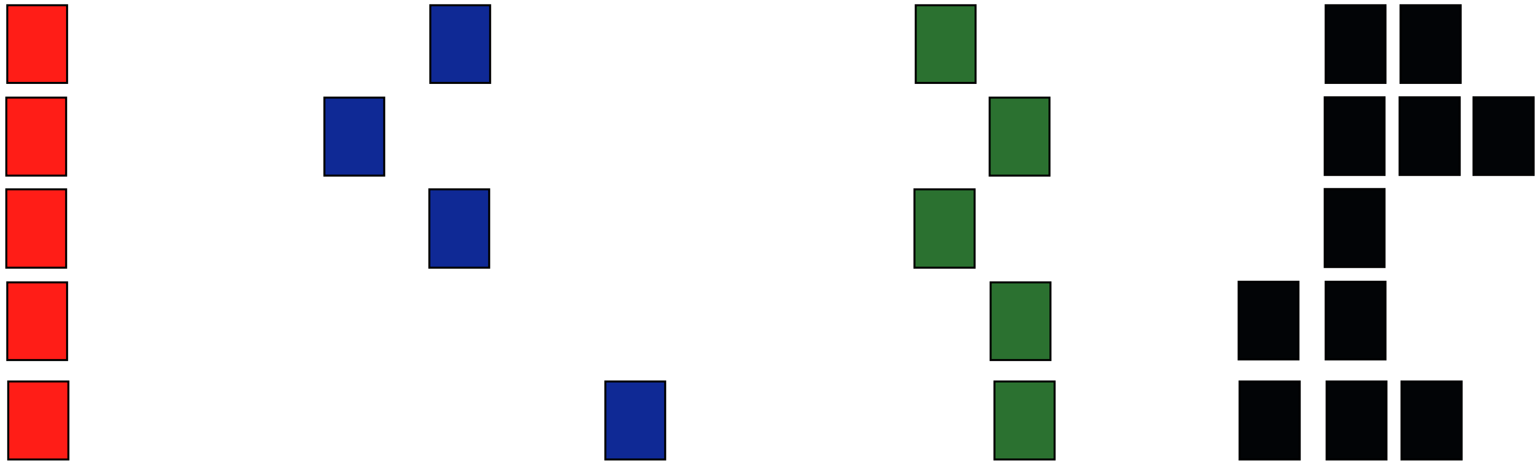
True Negatives (TN)

False Negatives (FN)

False Positives (FP)

Design Criteria for Edge Detection

- Good detection: find all real edges, ignoring noise or other artifacts
- Good localization
  - as close as possible to the true edges
  - one point only for each true edge point



True edge

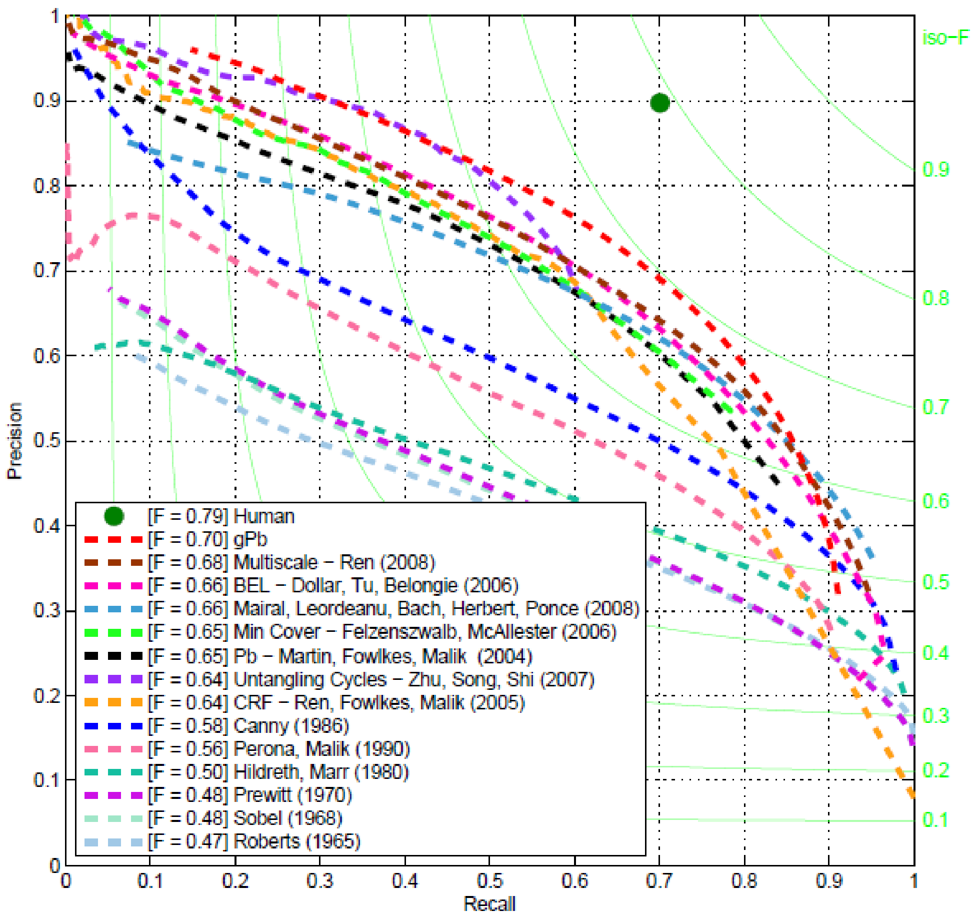
Poor robustness to noise

Poor localization

Too many responses

45 years of boundary detection

[Pre deep learning]



Source: Arbelaez, Maire, Fowlkes, and Malik. TPAMI 2011 (pdf)

Questions?

CAP5415 Computer Vision

Yogesh S Rawat

yogesh@ucf.edu

HEC-241

Questions?

Edge Detection

Lecture 4

Prewitt and Sobel edge detection

Prewitt and Sobel Edge Detector

- Compute derivatives
  - In x and y directions
- Find gradient magnitude
- Threshold gradient magnitude

Discrete derivative - revisit

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

Backward difference

$$\frac{df}{dx} = f(x) - f(x+1) = f'(x)$$

Forward difference

$$\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$$

Central difference

Derivative Masks

Backward difference	[-1 1]
Forward difference	[1 -1]
Central difference	[-1 0 1]

Image derivative

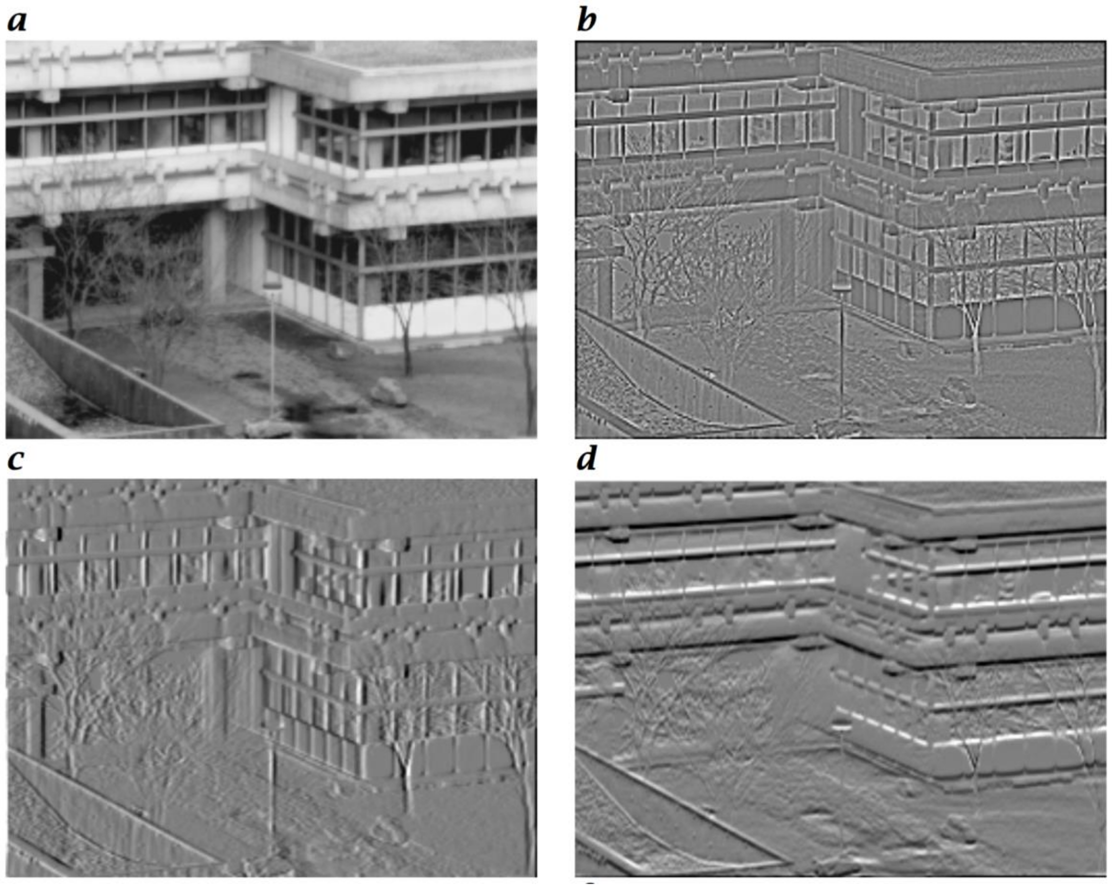
Given function $f(x, y)$

Gradient vector $\nabla f(x, y) = \begin{bmatrix} \dfrac{\partial f(x, y)}{\partial x} \\ \dfrac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$

Gradient magnitude $|\nabla f(x, y)| = \sqrt{f_x^2 + f_y^2}$

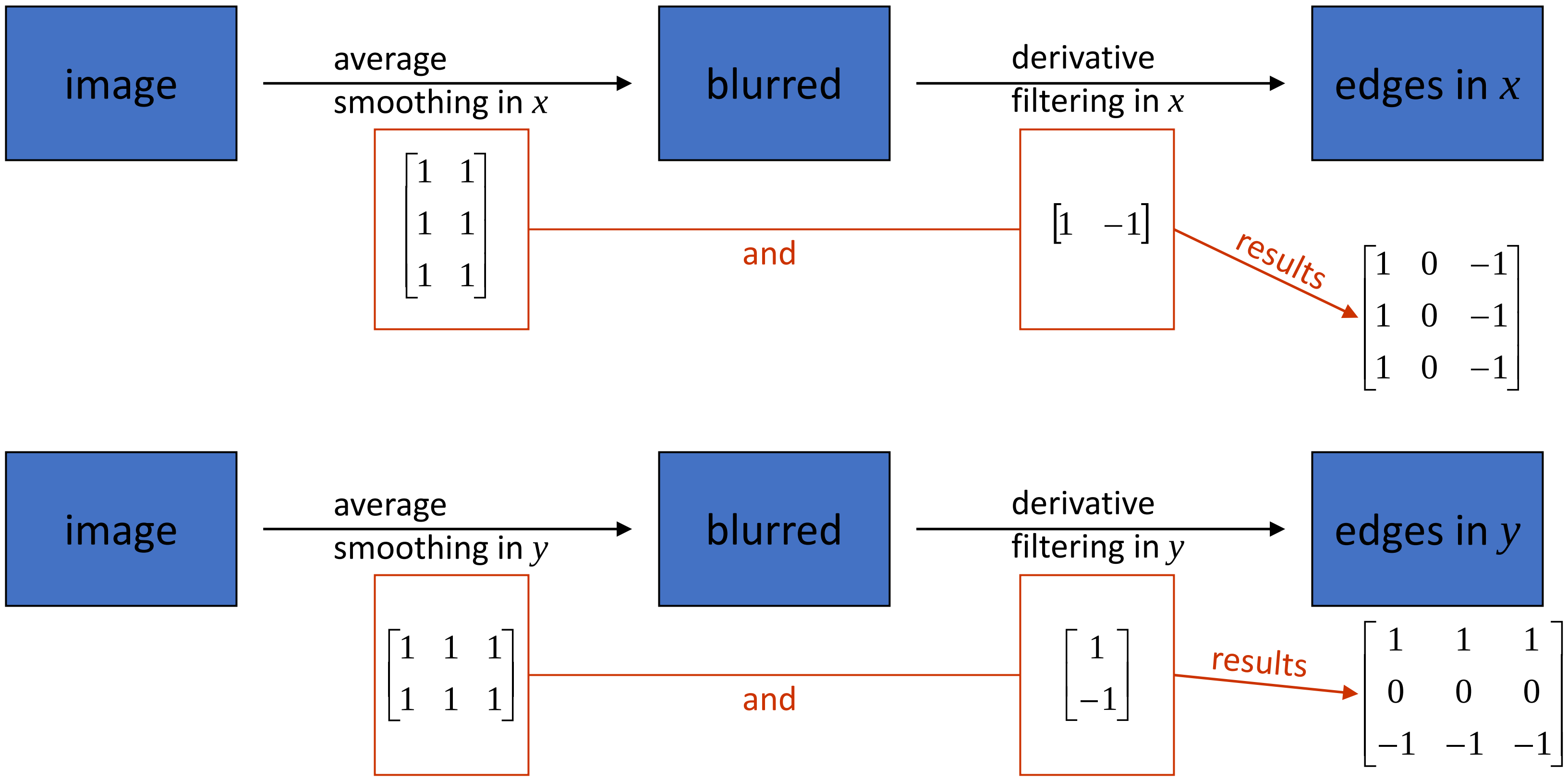
Gradient direction $\theta = \tan^{-1} \dfrac{f_x}{f_y}$

Example



a. Original image
b. Laplacian operator
c. Horizontal derivative
d. Vertical derivative

Prewitt Edge Detector



Sobel Edge Detector

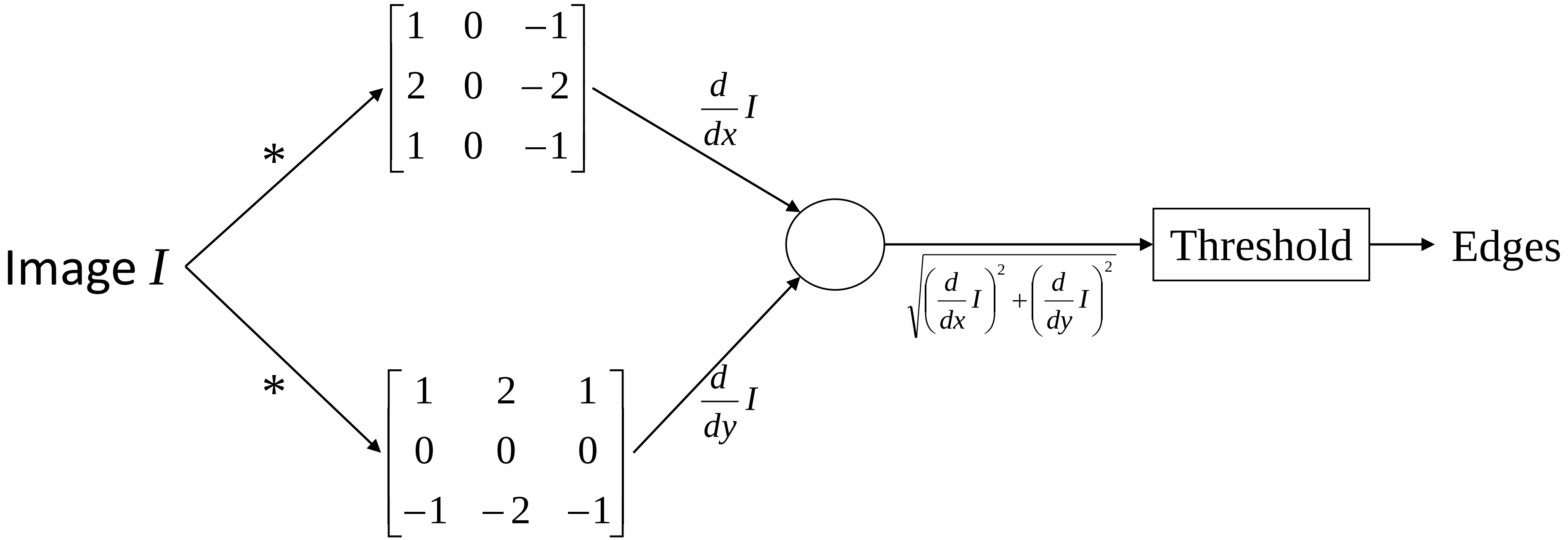
image $\xrightarrow[\text{smoothing in } x]{\text{average}}$ blurred $\xrightarrow[\text{filtering in } x]{\text{derivative}}$ edges in x

$$\begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 1 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & -1 \end{bmatrix} \quad \xrightarrow{\text{results}} \quad \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

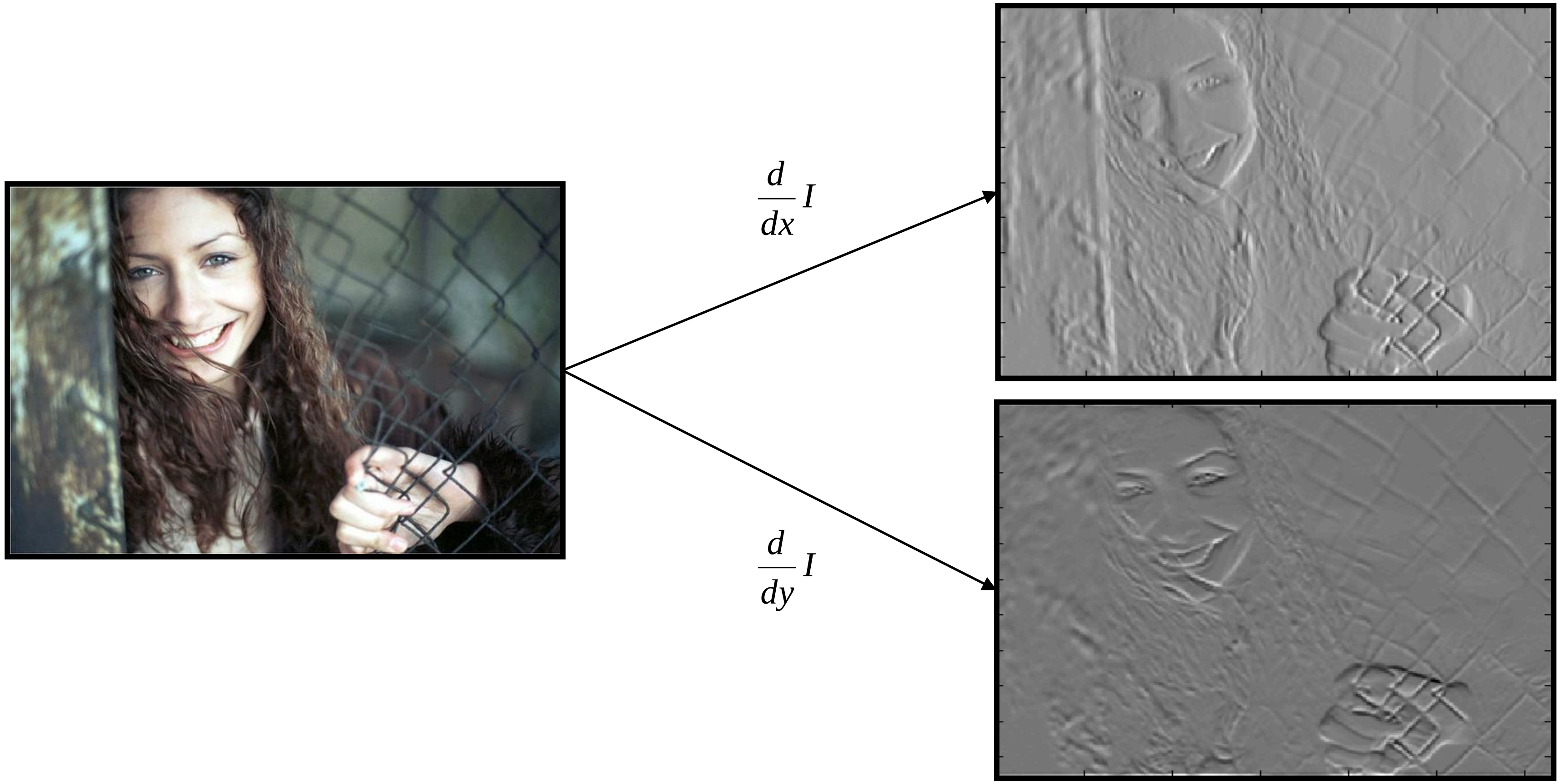
image $\xrightarrow[\text{smoothing in } y]{\text{average}}$ blurred $\xrightarrow[\text{filtering in } y]{\text{derivative}}$ edges in y

$$\begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad \xrightarrow{\text{results}} \quad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Sobel Edge Detector



Sobel Edge Detector



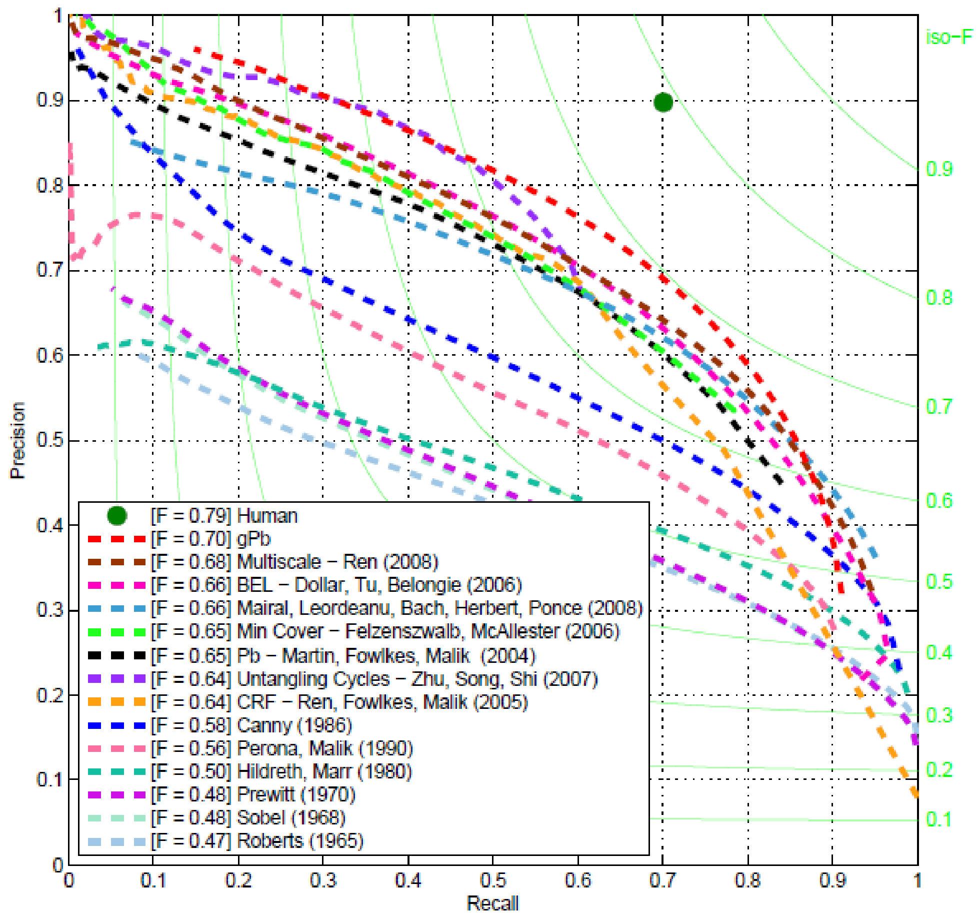
Sobel Edge Detector

$$\Delta = \sqrt{\left(\frac{d}{dx} I\right)^2 + \left(\frac{d}{dy} I\right)^2}$$

$$\Delta \geq Threshold = 100$$

UCF Center for Research in Computer Vision

Sobel vs Prewitt



Source: Arbelaez, Maire, Fowlkes, and Malik. TPAMI 2011 (pdf)

Questions?

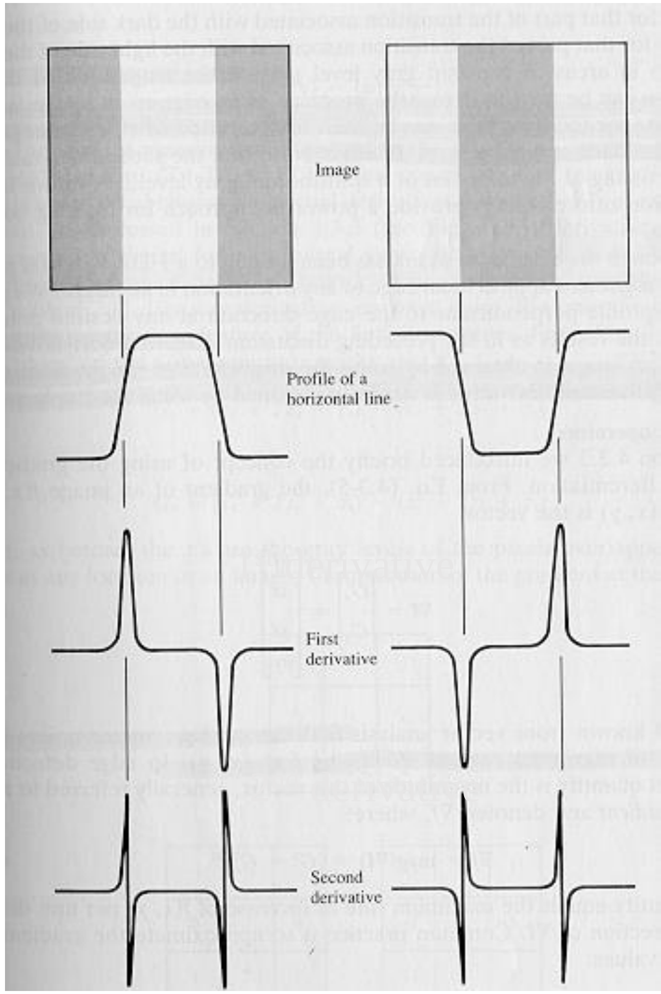
Edge Detection

Lecture 4

Marr Hildreth edge detection

Second derivate

- Maxima minima of first derivative
- Zero-crossings of second derivative



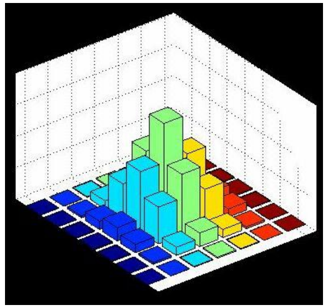
Marr Hildreth Edge Detector

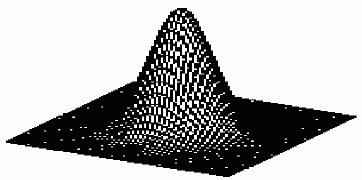
- Smooth image by Gaussian filter
- Apply Laplacian
  - Widely used operator
- Find zero crossings
  - Scan along each row, record an edge point at the location of zero-crossing.
  - Repeat above step along each column

Marr Hildreth Edge Detector

- Gaussian smoothing

$$\underbrace{S}_{\text{smoothed image}} = \underbrace{g}_{\text{Gaussian filter}} * \underbrace{I}_{\text{image}}$$





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

- Find Laplacian

$$\Delta^2 S = \underbrace{\frac{\partial^2}{\partial x^2} S}_{\text{second order derivative in } x} + \underbrace{\frac{\partial^2}{\partial y^2} S}_{\text{second order derivative in } y}$$

- ∇ is used for gradient (derivative)
- Δ is used for Laplacian

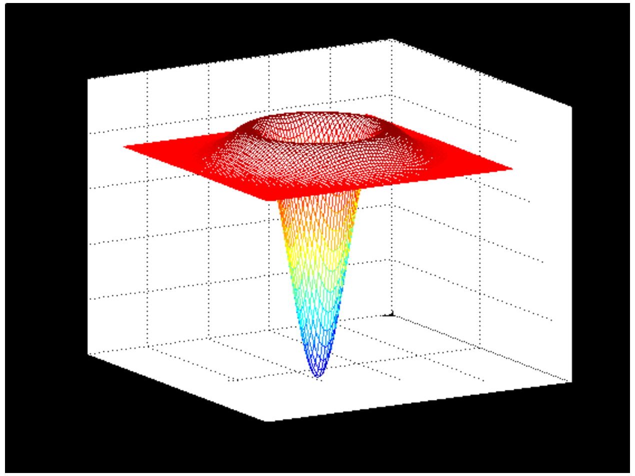
Finding Zero Crossings

- Four cases of zero-crossings :
  - {+,-}
  - {+,0,-}
  - {-,+}
  - {-,0,+}
- Slope of zero-crossing {a, -b} is |a+b|.
- To mark an edge
  - Compute slope of zero-crossing
  - Apply a threshold to slope

Marr Hildreth Edge Detector

- Deriving the Laplacian of Gaussian (LoG)

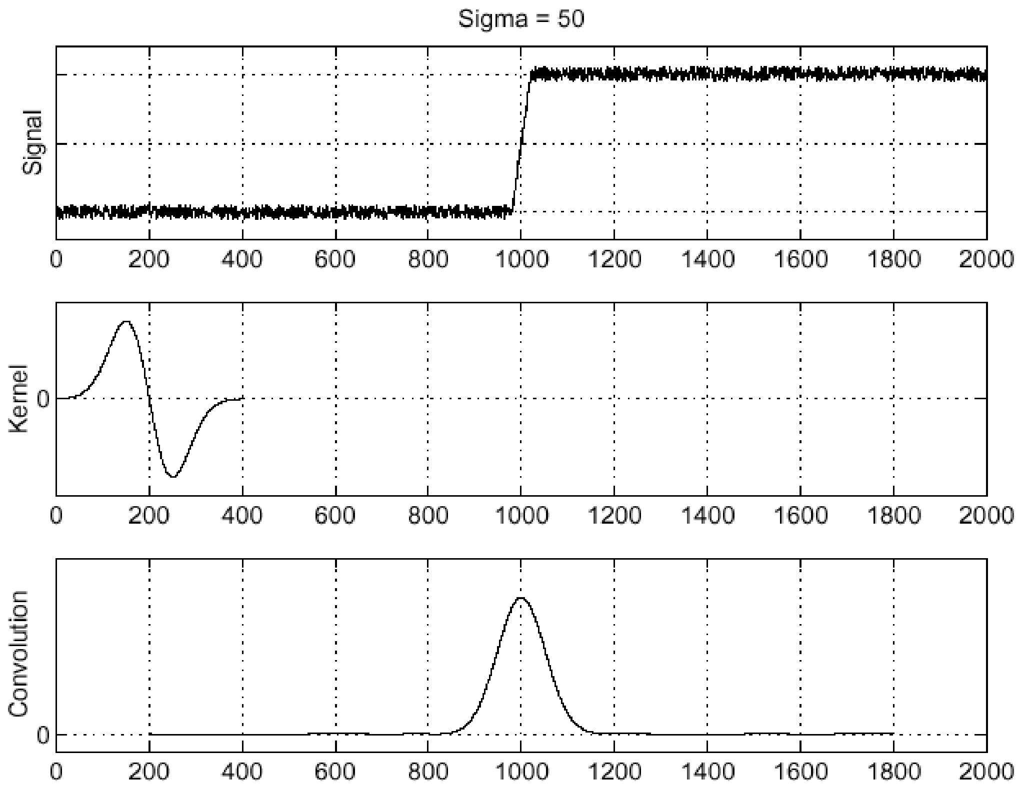
$$\Delta^2 S = \Delta^2 (g * I) = (\Delta^2 g) * I$$

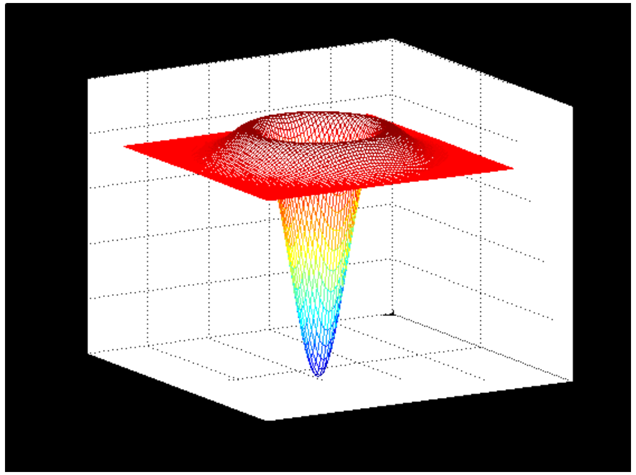


$$g(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\Delta^2 g(x,y) = -\frac{1}{\sqrt{2\pi}\sigma^3}\left(2 - \frac{x^2+y^2}{\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Marr Hildreth Edge Detector





LoG Filter

$$\Delta^2 G_\sigma = -\frac{1}{\sqrt{2\pi}\sigma^3}\left(2-\frac{x^2+y^2}{\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$

			Y				
0.0008	0.0066	0.0215	0.031	0.0215	0.0066	0.0008	
0.0066	0.0438	0.0982	0.108	0.0982	0.0438	0.0066	
0.0215	0.0982	0	-0.242	0	0.0982	0.0215	
0.031	0.108	-0.242	-0.7979	-0.242	0.108	0.031	X
0.0215	0.0982	0	-0.242	0	0.0982	0.0215	
0.0066	0.0438	0.0982	0.108	0.0982	0.0438	0.0066	
0.0008	0.0066	0.0215	0.031	0.0215	0.0066	0.0008	

UCF Center for Research in Computer Vision

On the Separability of LoG

$$g(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$
$$= \frac{1}{\sqrt{2\pi}\sigma} \underbrace{\left(e^{-\frac{x^2}{2\sigma^2}}\right)}_{g_1(x)} \underbrace{\left(e^{-\frac{y^2}{2\sigma^2}}\right)}_{g_2(y)}$$

- Similar to separability of Gaussian filter
  - 2D Gaussian can be separated into 2 one-dimensional Gaussians

$$h(x,y) = I(x,y) * g(x,y)$$ n^2 multiplications

$$h(x,y) = (I(x,y) * g_1(x)) * g_2(y)$$ $2n$ multiplications

$$g_1 = \begin{bmatrix} .011 & .13 & .6 & 1 & .6 & .13 & .011 \end{bmatrix}$$

$$g_2 = \begin{bmatrix} .011 \\ .13 \\ .6 \\ 1 \\ .6 \\ .13 \\ .011 \end{bmatrix}$$

On the Separability of LoG

$$\Delta^2 S = \Delta^2 (g * I) = (\Delta^2 g) * I = I * (\Delta^2 g)$$

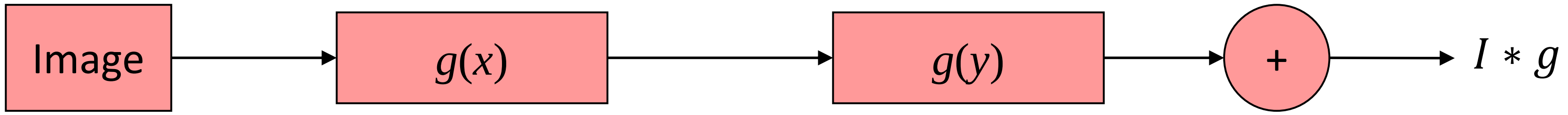
Requires n^2 multiplications

$$\Delta^2 S = \left(I * g_{yy}(y)\right) * g(x) + (I * g_{xx}(x)) * g(y)$$

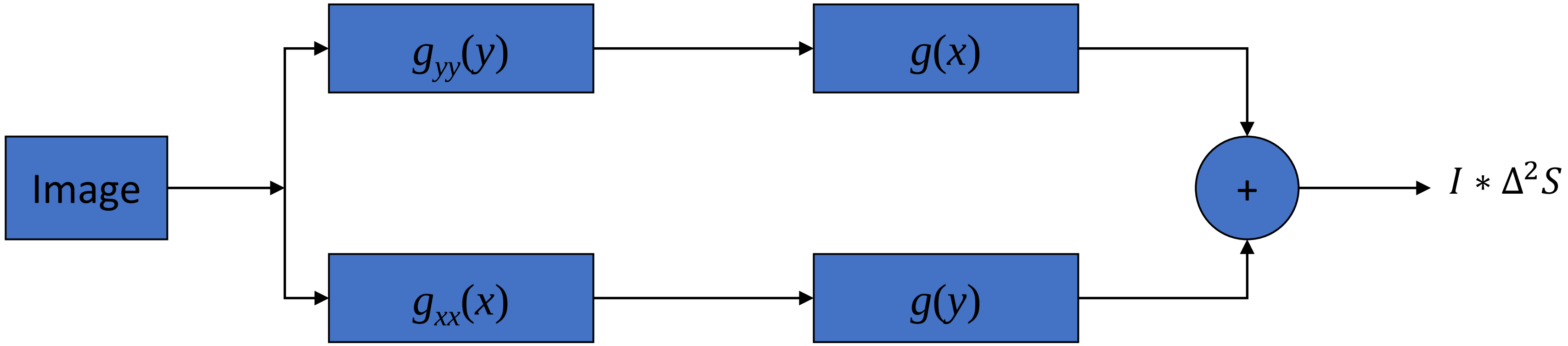
Requires $4n$ multiplications

Separability

Gaussian Filtering



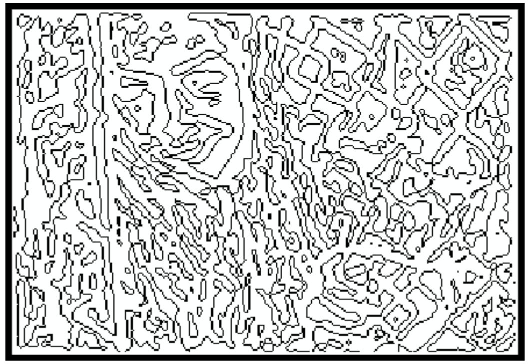
Laplacian of Gaussian Filtering



Algorithm

- Compute LoG
  - Use 2D filter $\Delta^2 g(x, y)$
  - Use 4 1D filters $g(x),\ g_{xx}(x),\ g(y),\ g_{yy}(y)$
- Find zero-crossings from each row
- Find slope of zero-crossings
- Apply threshold to slope and mark edges

Example

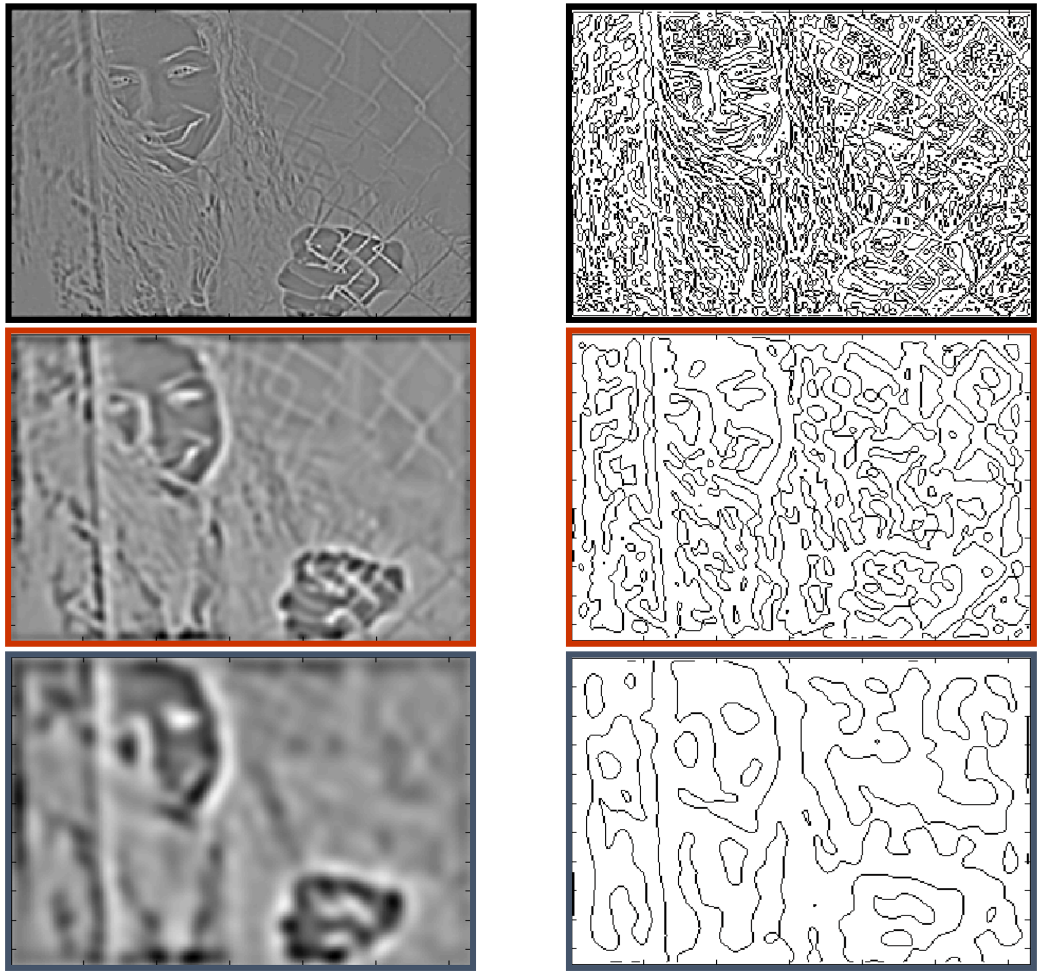
I	$I * \left(\Delta^2 g\right)$	Zero crossings of $\Delta^2 S$
		

Example

$\sigma = 1$

$\sigma = 3$

$\sigma = 6$



Questions?

Edge Detection

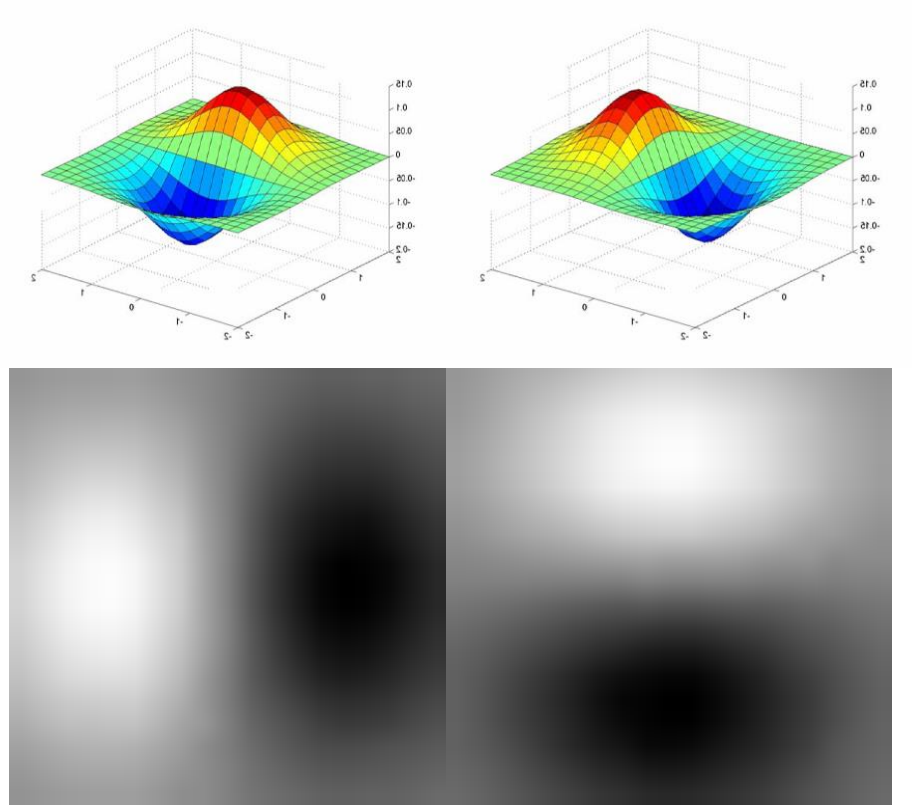
Lecture 4

Canny edge detection

Canny Edge Detector

- Smooth Image with Gaussian filter
- Compute Derivative of filtered image
- Find Magnitude and Orientation of gradient
- Apply Non-max suppression
- Apply Thresholding (Hysteresis)

Canny



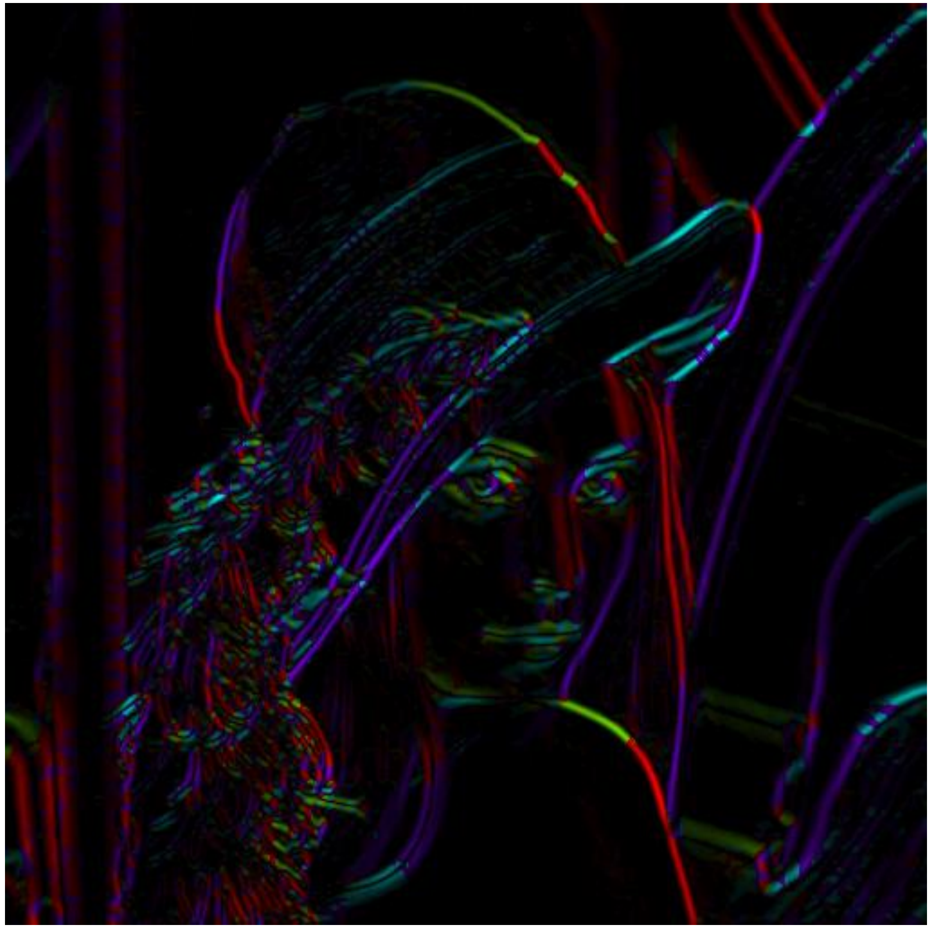
Canny-Gradients

X-Derivative of Gaussian

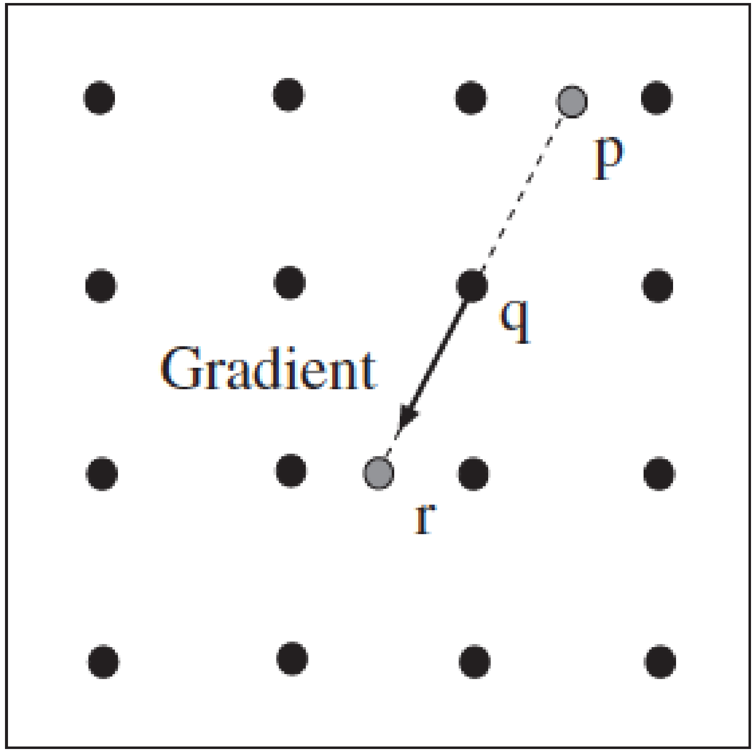
Y-Derivative of Gaussian

Gradient Magnitude

Gradient Orientation

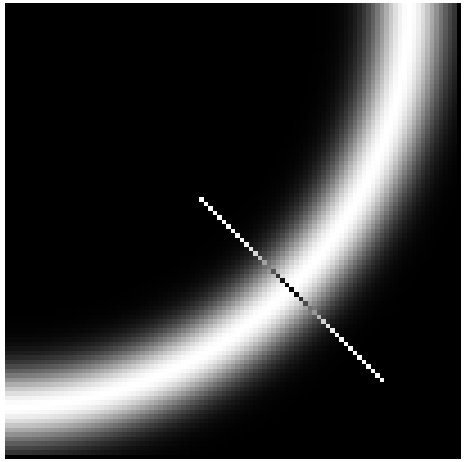


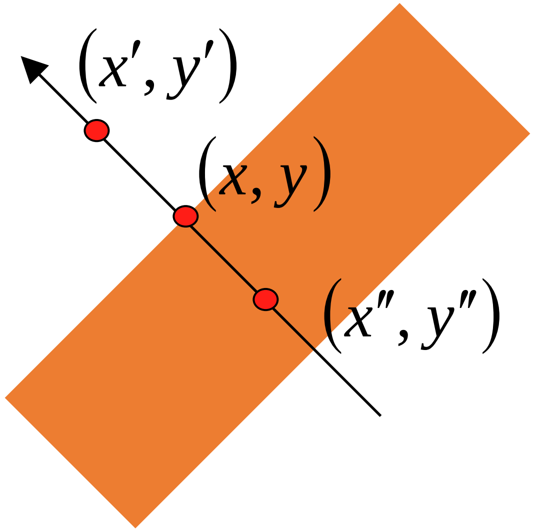
Non-maximum suppression



If gradient responses at r and p are smaller than q, q is an edge

Non-maximum suppression





$$M(x,y)=\begin{cases} |\nabla S|(x,y) & \text{if } |\nabla S|(x,y) > |\Delta S|(x',y') \\ & \& \, |\Delta S|(x,y) > |\Delta S|(x'',y'') \\ 0 & \text{otherwise} \end{cases}$$

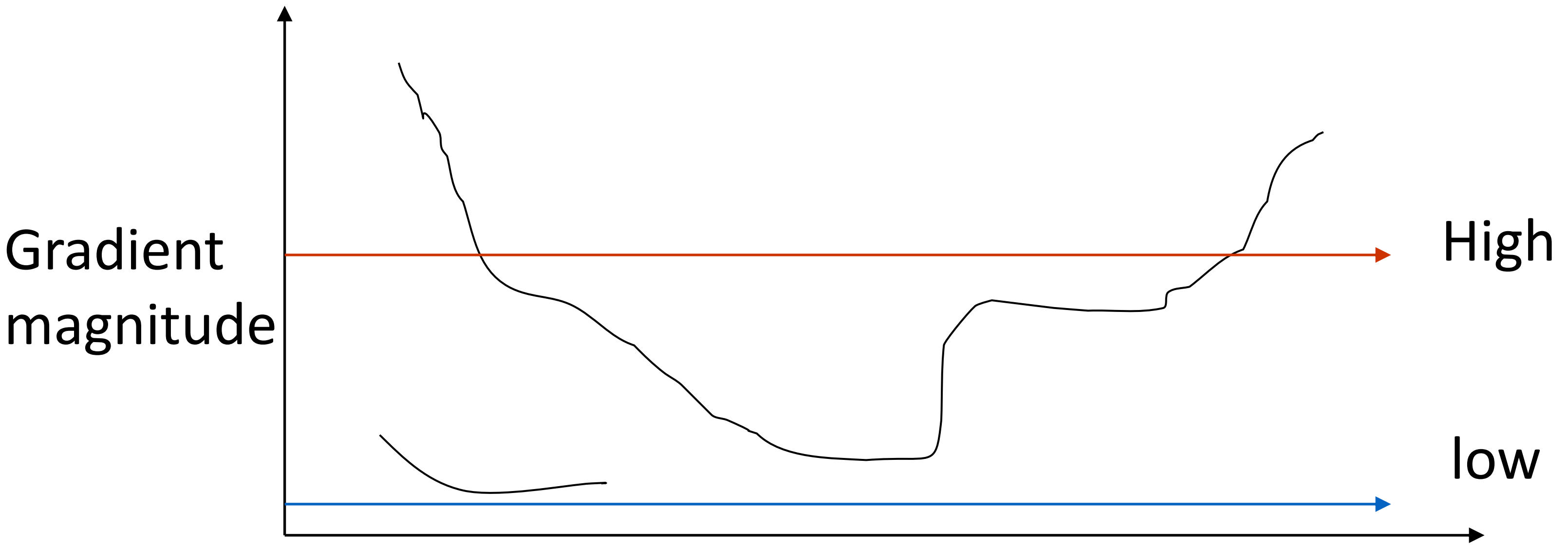
x' and **x**'' are the neighbors of **x** along normal direction to an edge

Non-maximum suppression

Before Non-Max Suppression

After Non-Max Suppression

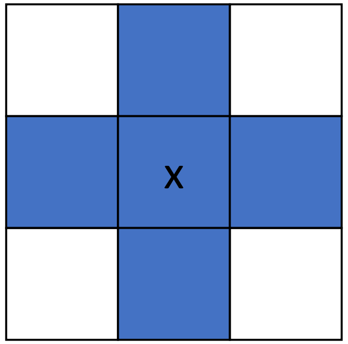
Hysteresis Thresholding [L, H]



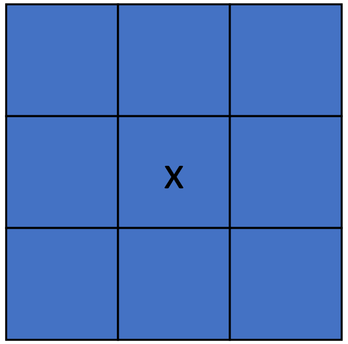
Hysteresis Thresholding [L, H]

- If the gradient at a pixel is
  - above “**High**”, declare it as an ‘**edge pixel**’
  - below “**Low**”, declare it as a “**non-edge-pixel**”
  - **between** “low” and “high”
    - Consider its neighbors iteratively then declare it an “edge pixel” if it is **connected** to an ‘edge pixel’ **directly** or via pixels **between** “low” and “high”.

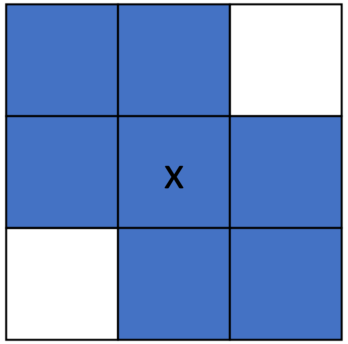
Hysteresis Thresholding [L, H]



4 connected



8 connected



6 connected

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

Final Canny Edges

Effect of Gaussian Kernel (smoothing)

original

Canny with $\sigma = 1$

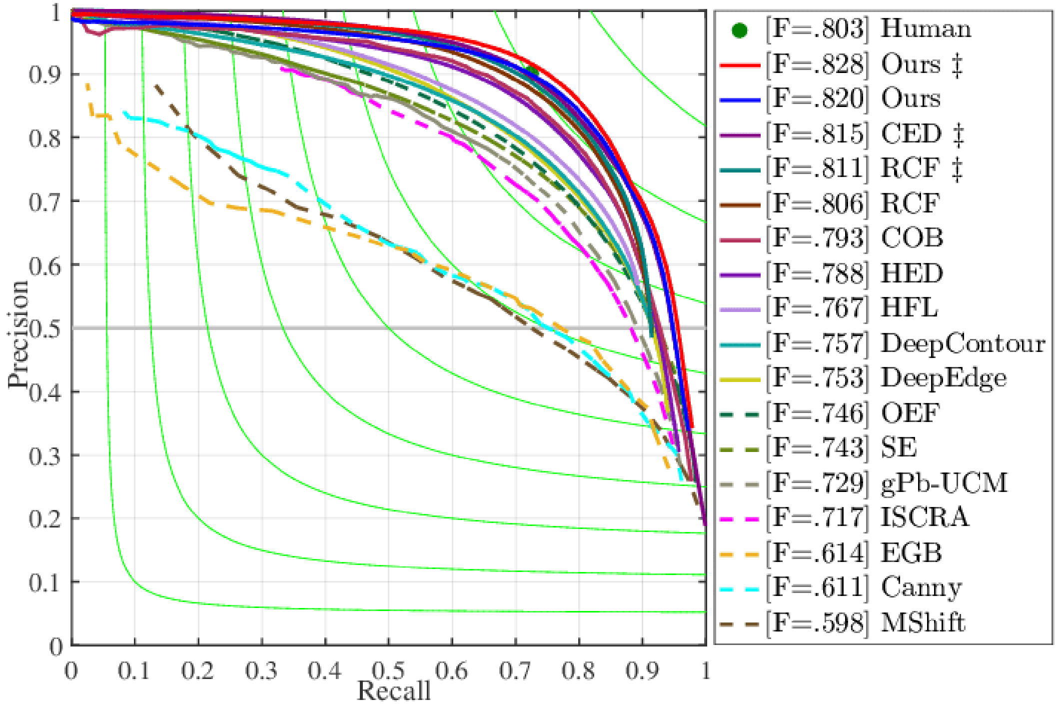
Canny with $\sigma = 2$

The choice of σ depends on desired behavior

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

Edge Detection with Deep Learning

- We will revisit edge detection
  - After Deep Learning tutorial lectures
  - If time permits



Questions?

Sources for this lecture include materials from works by Mubarak Shah, Abhijit Mahalanobis, and D. Lowe

Other sources from James Hays, Lana Lazebnik, Steve Seitz, David Forsyth, David Lowe, Fei-Fei Li, and Derek Hoiem