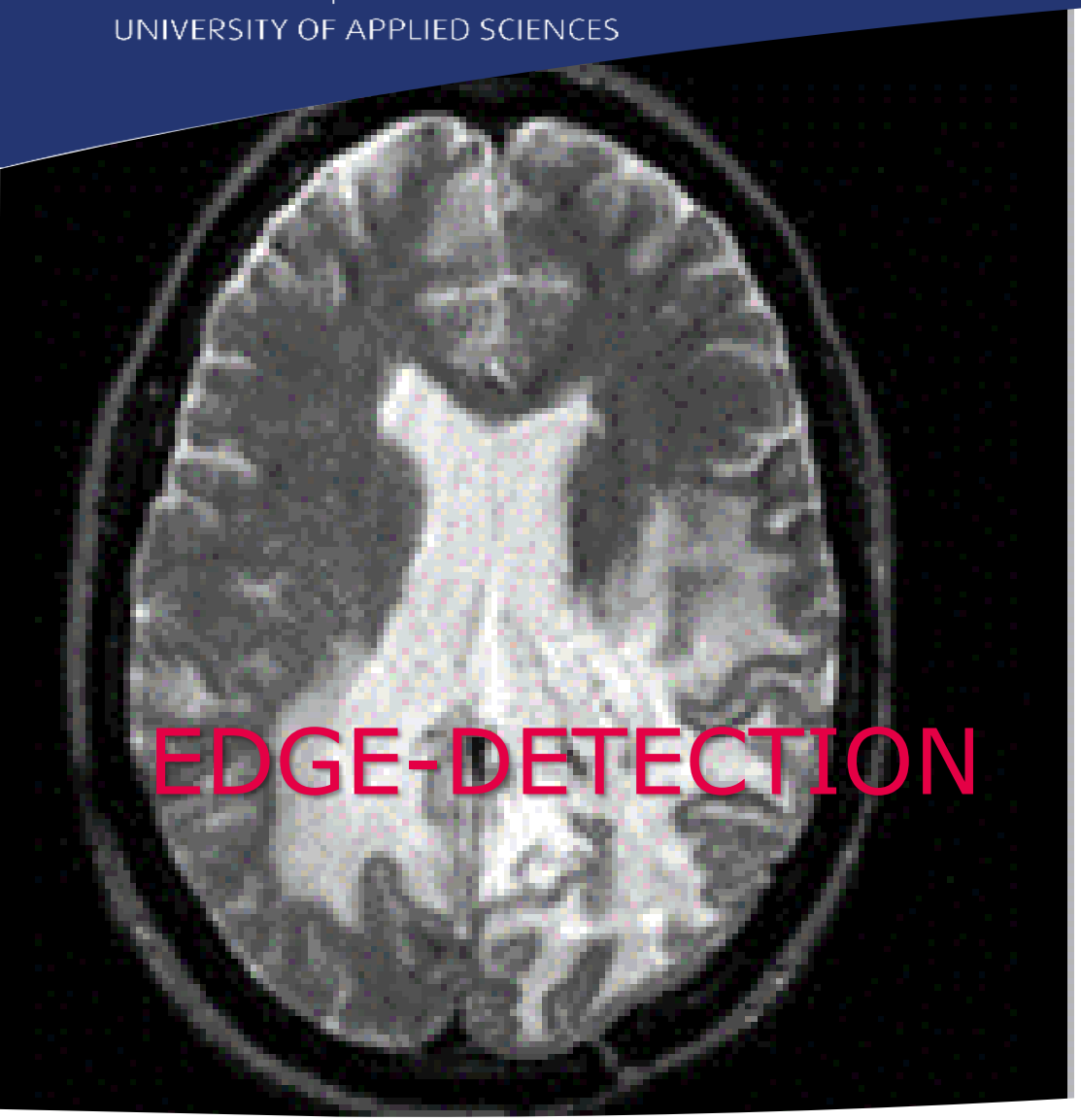


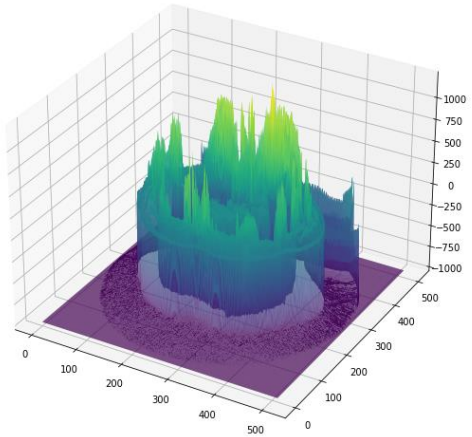
EDGE-DETECTION

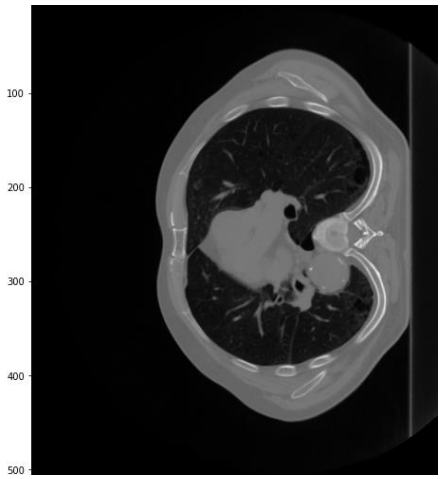
Edge Detection

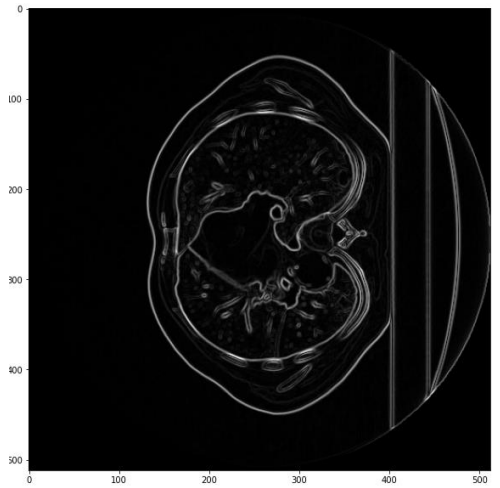
How can we detect changes?

- Edges: **Boundary** (steep-changes) between two regions with distinct gray-level properties.

- Most important visual information.



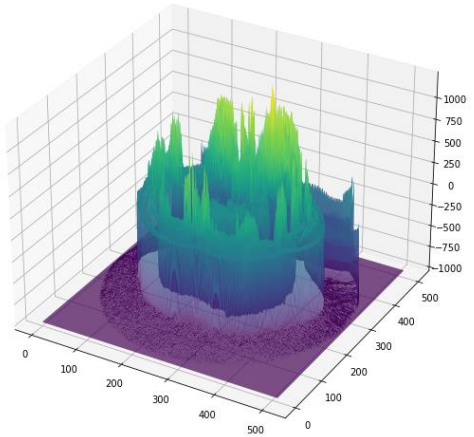


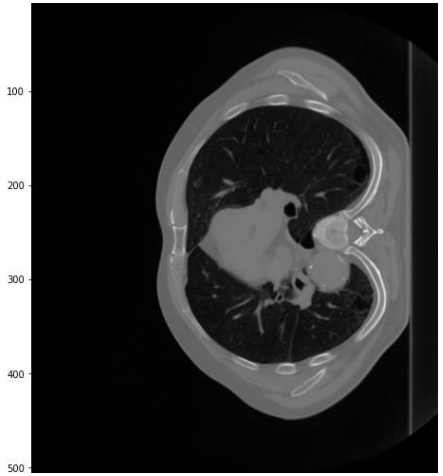


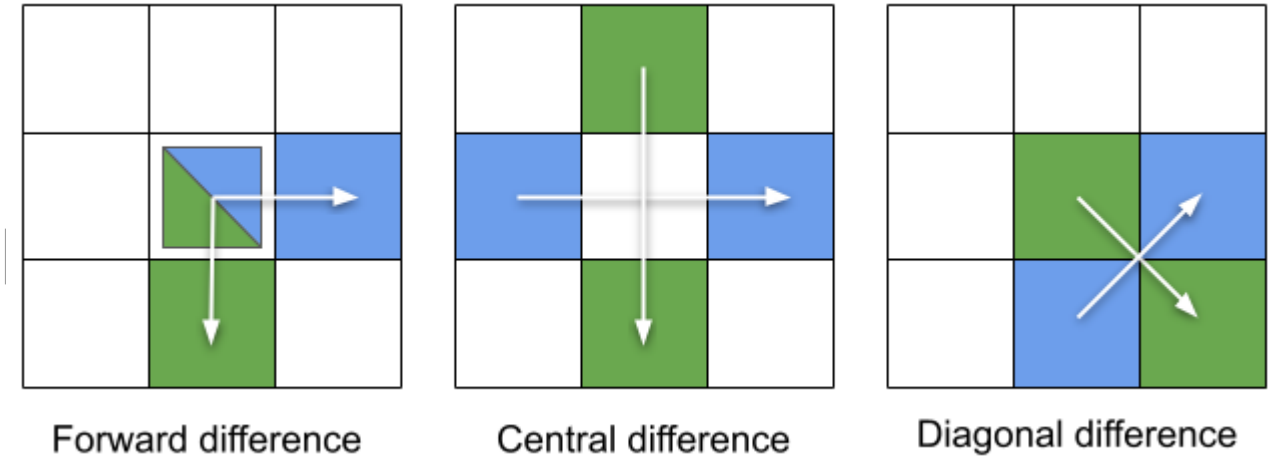
Edge Detection – Derivatives!

- Steepness is defined by spatial derivatives: $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ (and $\frac{\partial I}{\partial z}$ in 3D)

- Image edges (gradients) have direction and magnitude

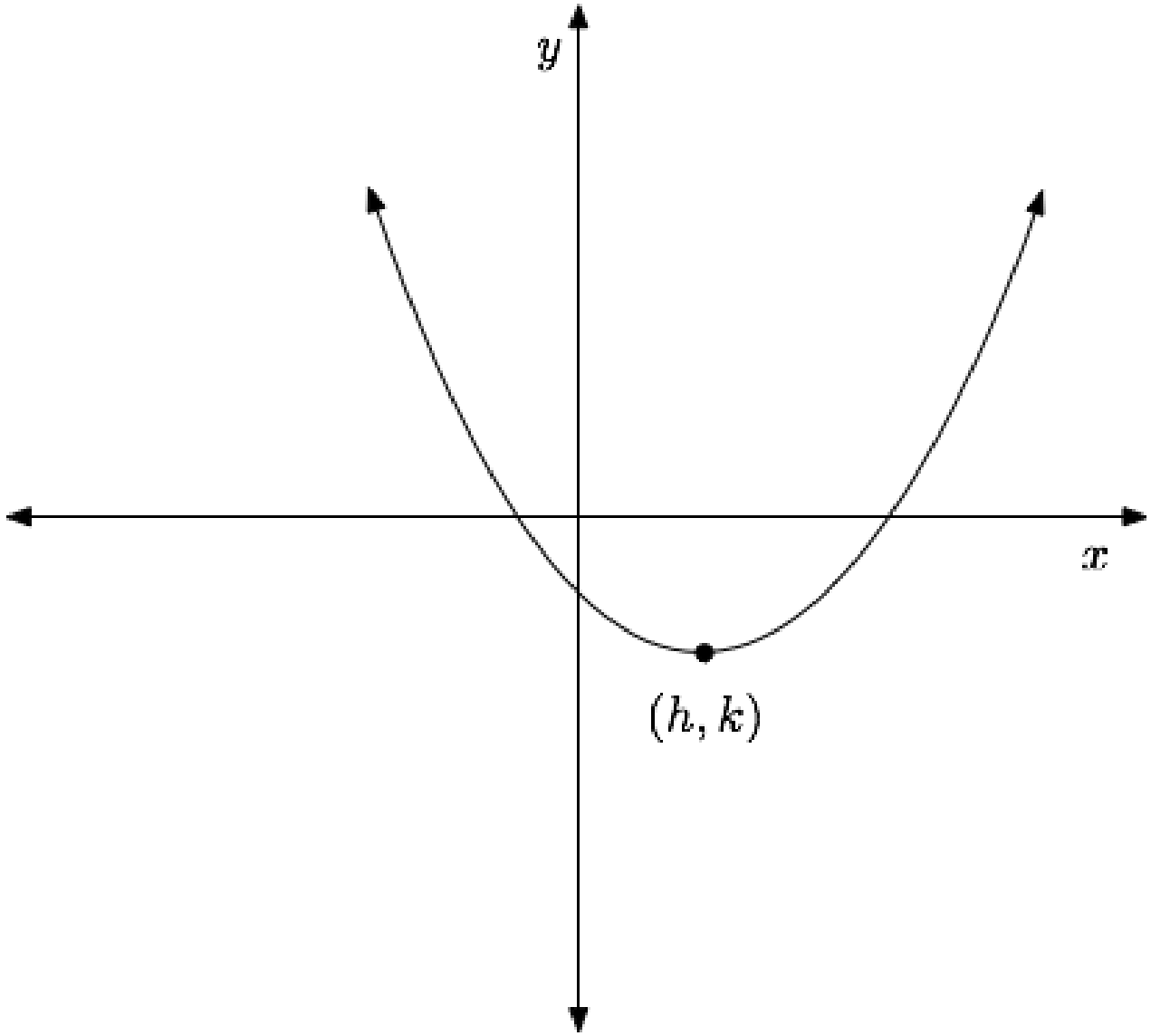






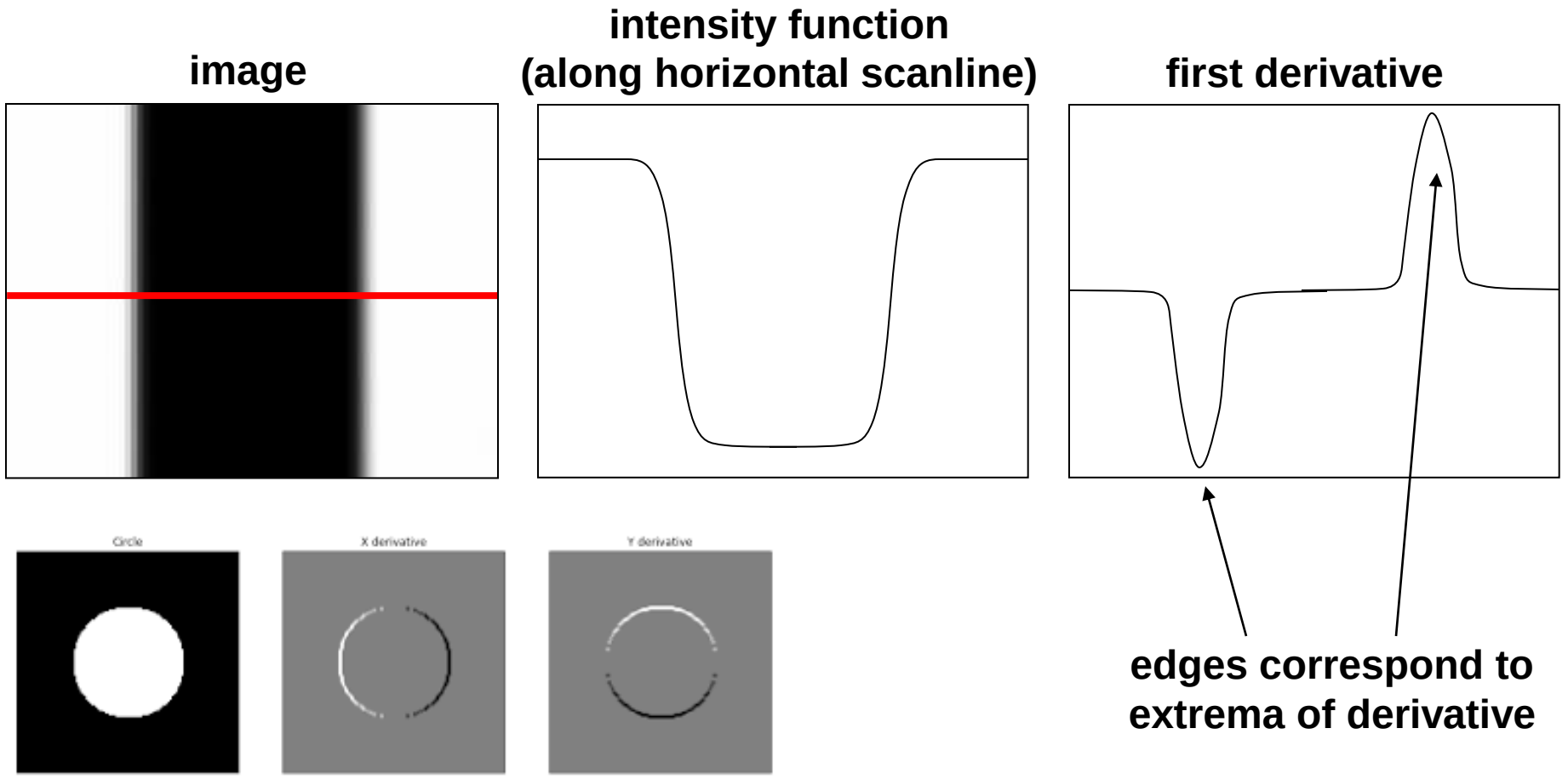
Derivatives

Reminder



Derivatives and edges

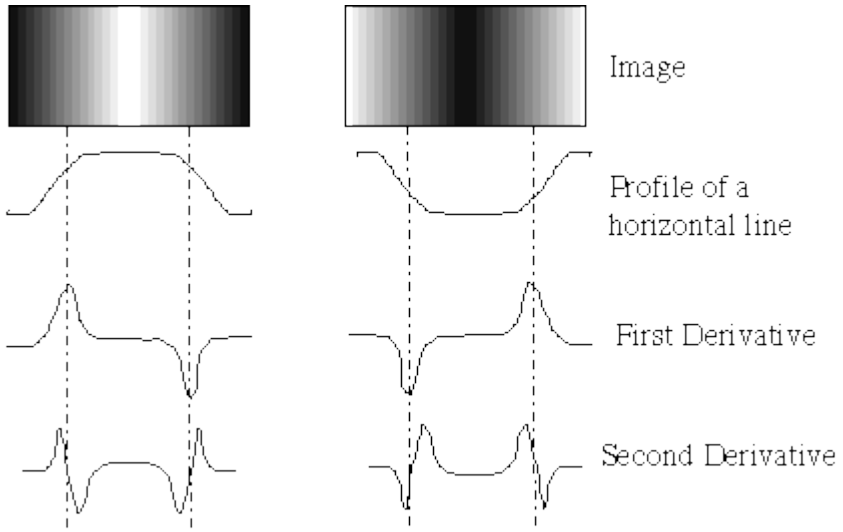
An edge is a place of ***rapid change*** in the image intensity function.

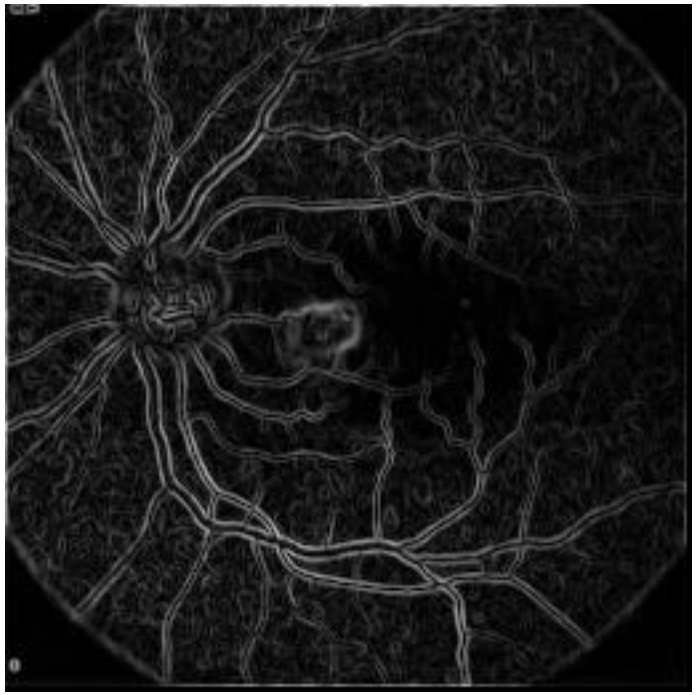


Source: L. Lazebnik

Edge Detection

Using derivatives





Derivatives with convolution

Where would the function "go" if we give it a little "nudge"?

For 2D function, $f(x, y)$, the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon, y) - f(x,y)}{\epsilon}$$

For discrete data, we can approximate using finite differences:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

Edge Detection as Derivatives

Where would the function "go" if we give it a little "nudge"?

How would you implement this as a filter?

- For continuous functions: $f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$
- Approximate partial derivatives for both x and y:

$$\frac{\partial}{\partial x} f(i,j) \cong f(i,j) - f(i-1,j)$$

$$\frac{\partial}{\partial y} f(i,j) \cong f(i,j) - f(i,j-1)$$

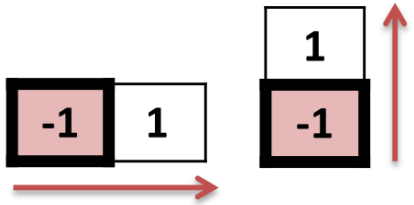
- Approximate derivatives by convolution with $[1 \quad -1]$

These kernels perform well on very clean images. They don't work as well if there's noise on the images (kernels are too small!)

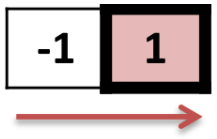
Edge Detection Convolution Kernels

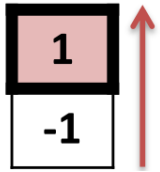
Calculating Gradients

- Forward Finite Difference: $f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$

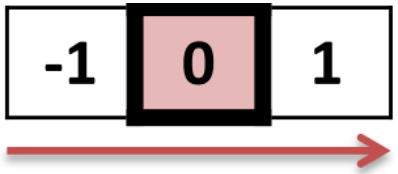


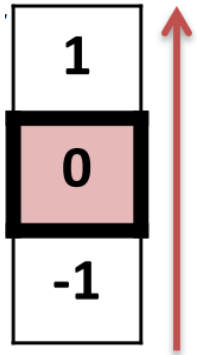
- Backward Finite Difference: $f'(x) = \lim_{\Delta x \to 0} \frac{f(x) - f(x-\Delta x)}{\Delta x}$





- Central Finite Difference: $f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}$





Derivatives Approximation with Kernel Operations

- Roberts Operator
- Prewitt Operator
- Sobel Operator

+1	0	-1
+1	0	-1
+1	0	-1

Gx

-1	-1	-1
0	0	0
+1	+1	+1

Gy

Prewitt Operator

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

Gx

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

Gy

Sobel Operator

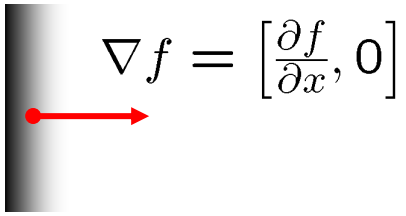
Gradient Operators

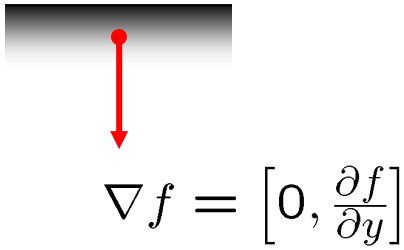
- 2D first derivatives - $\frac{\partial}{\partial x} f(i,j) \cong f(i,j) - f(i-1,j)$
- $\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$
- Gradient Magnitude: $\nabla f = \sqrt{G_x^2 + G_y^2} \approx |G_x| + |G_y|$
- Gradient Direction: $\alpha(x,y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$

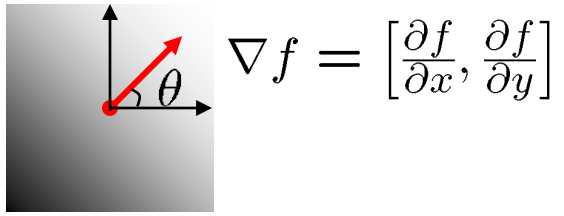
Image Gradients

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

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





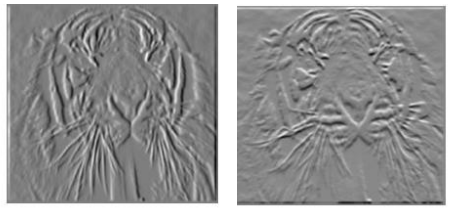


The **gradient direction** (orientation of edge normal) is given by:

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

The **edge strength** is given by the gradient magnitude

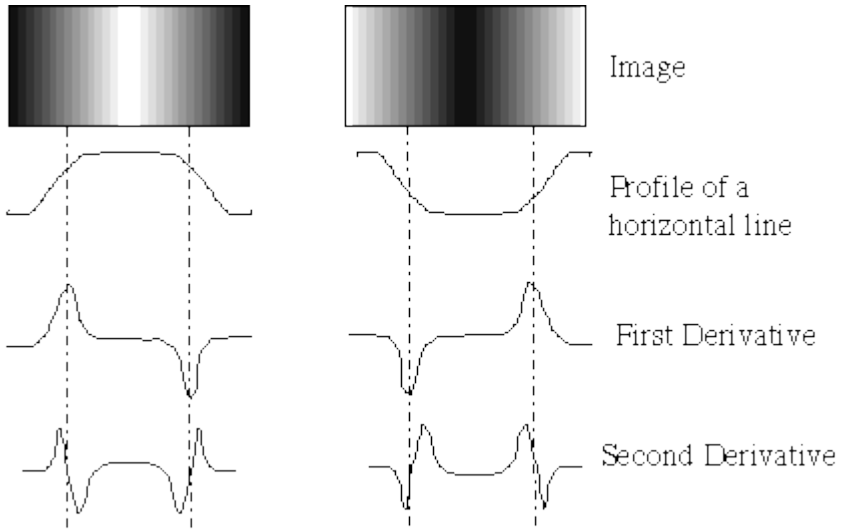
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$



Slide credit Steve Seitz

Edge Detection

Using derivatives for a discrete function



Gradient vector:

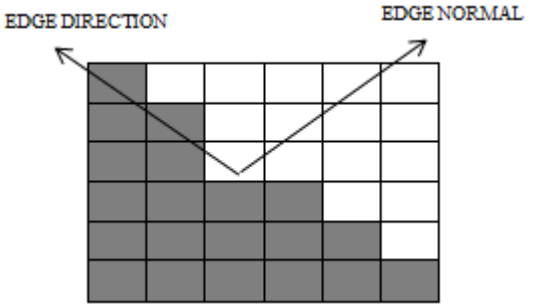
$$\nabla I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]^T$$

Magnitude:

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

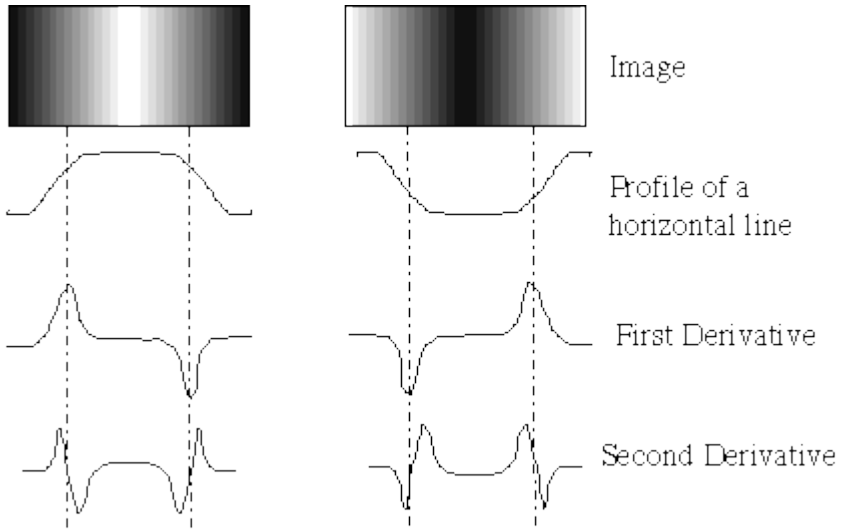
Orientation:

$$\theta = \text{atan}^2\left(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x}\right)$$



Edge Detection

Using derivatives for a discrete function through filters



+1	0	-1
+1	0	-1
+1	0	-1

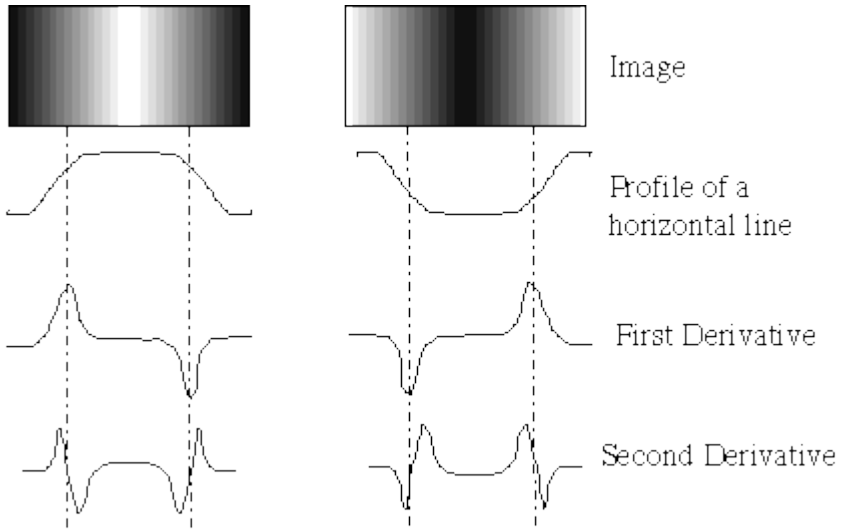
Gx

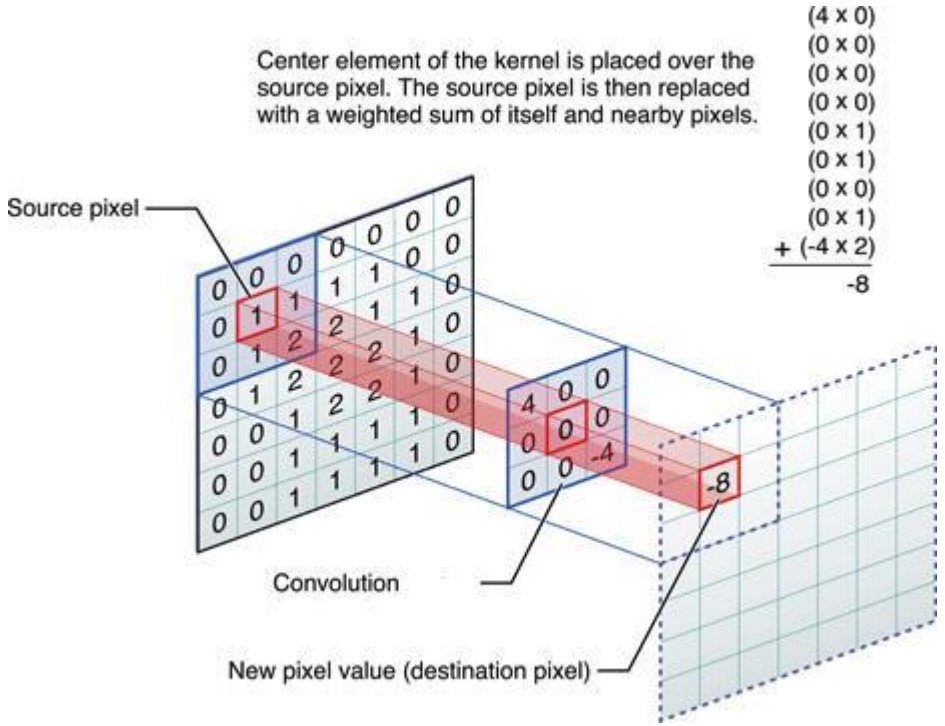
-1	-1	-1
0	0	0
+1	+1	+1

Gy

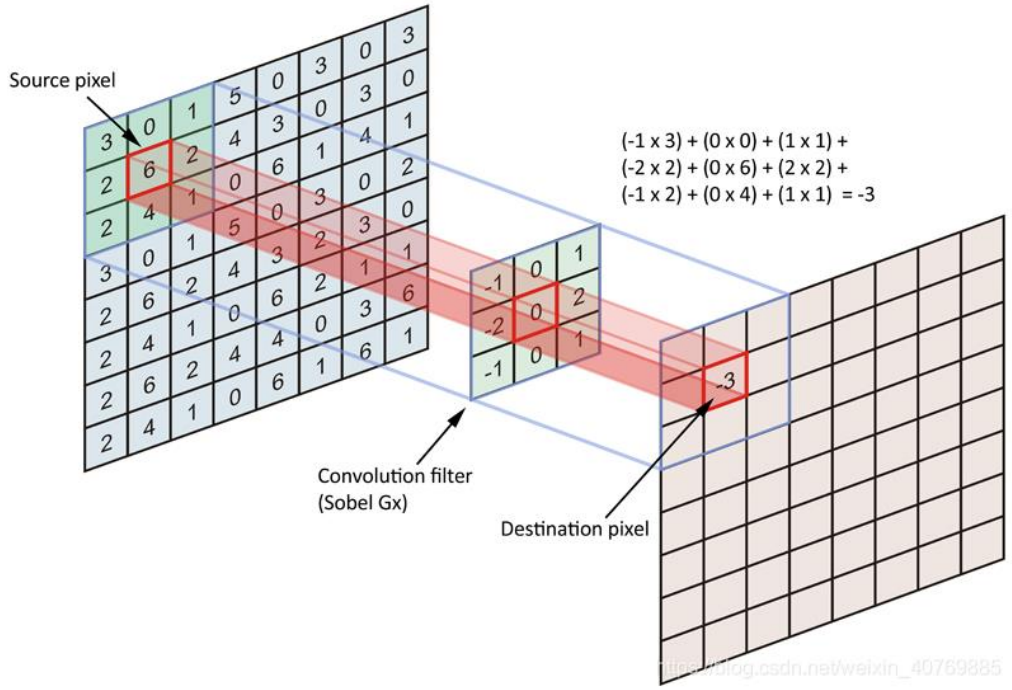
Edge Detection

Using derivatives for a discrete function through filters convolutions





Sobel Operator



- Approximate gradient calculation (the image derivatives)
- The final edge is received by L2-Distance:

$$\sqrt{G_x^2 + G_y^2}$$

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

Gx

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

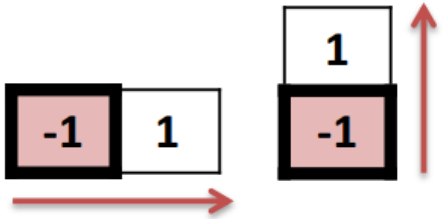
Gy

Edge Detection Convolution Kernels

Prewitt and Sobel compute derivatives in one direction while smoothing in the other direction

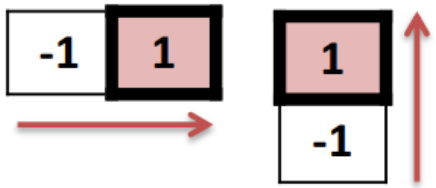
Calculating Gradients

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$



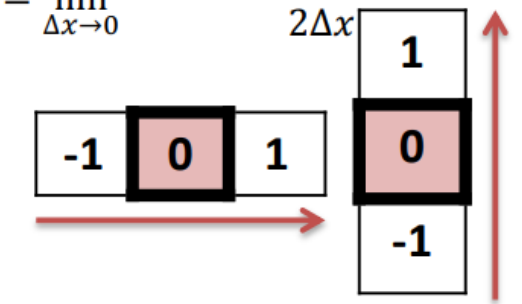
Forward Finite Difference

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x}$$

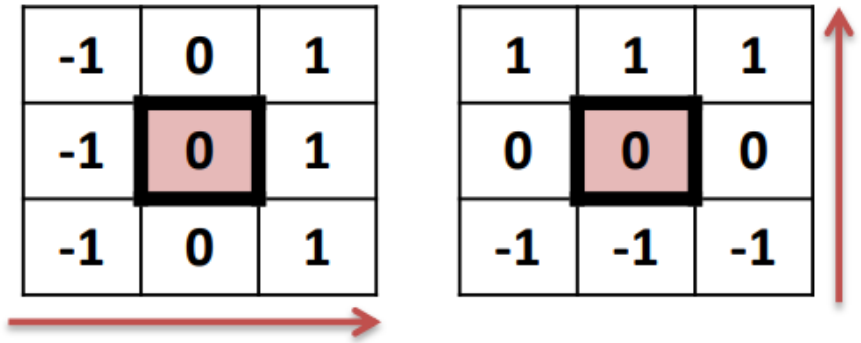


Backward Finite Difference

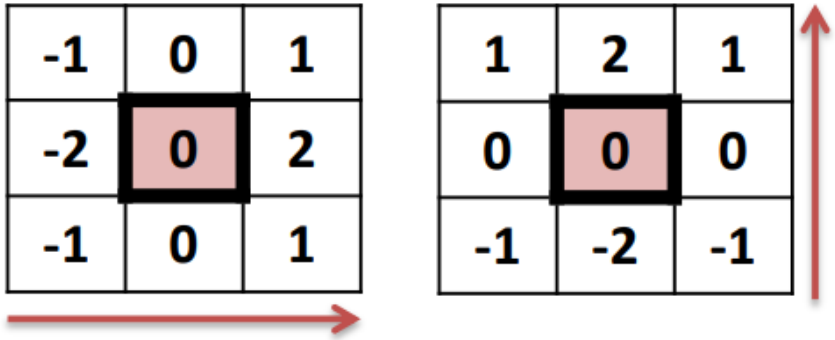
$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x}$$



Central Finite Difference

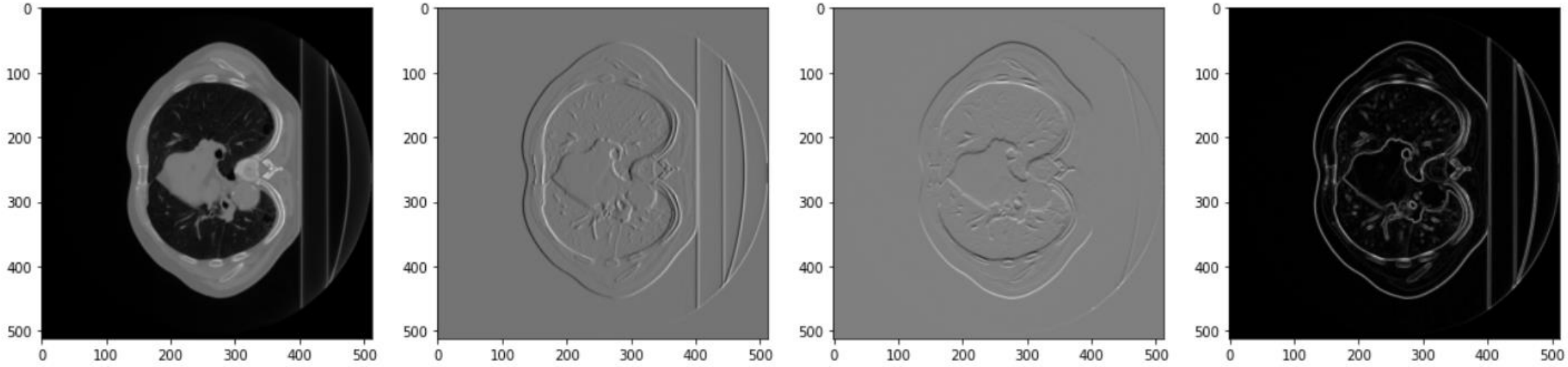


Prewitt Operator



Sobel Operator

Sobel Operator



imc FH KREMS
UNIVERSITY OF APPLIED SCIENCES

CANNY EDGE DETECTION

When noise is on the way

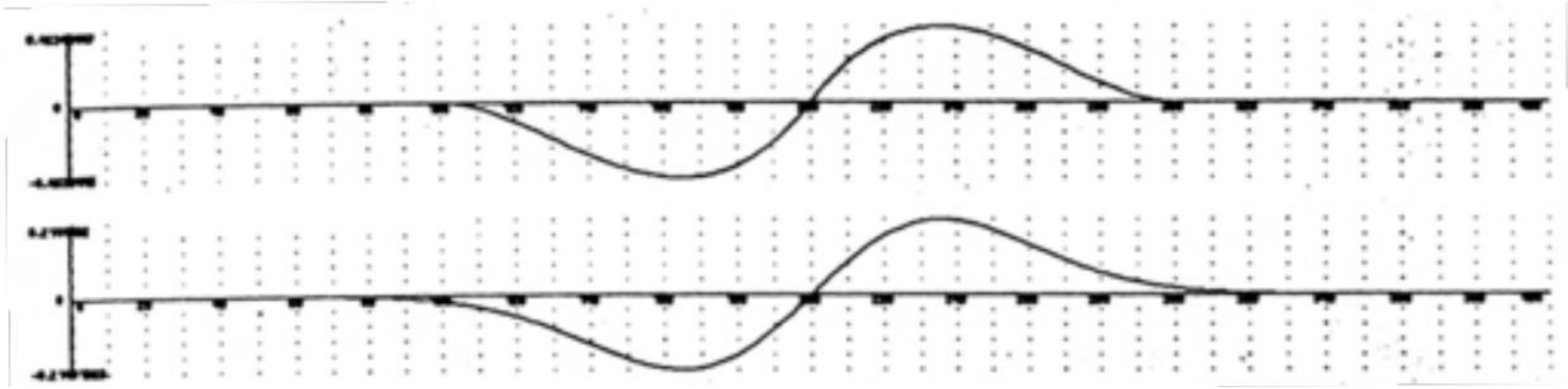
Canny Edge Detector

Classic Algorithm for Finding Edges in Images

- We want a kernel to simultaneously maximize SNR and localization for a step edge under white *Gaussian* noise
- Solution by numerical optimization is very close to derivative of Gaussian kernel

Numerically Optimized

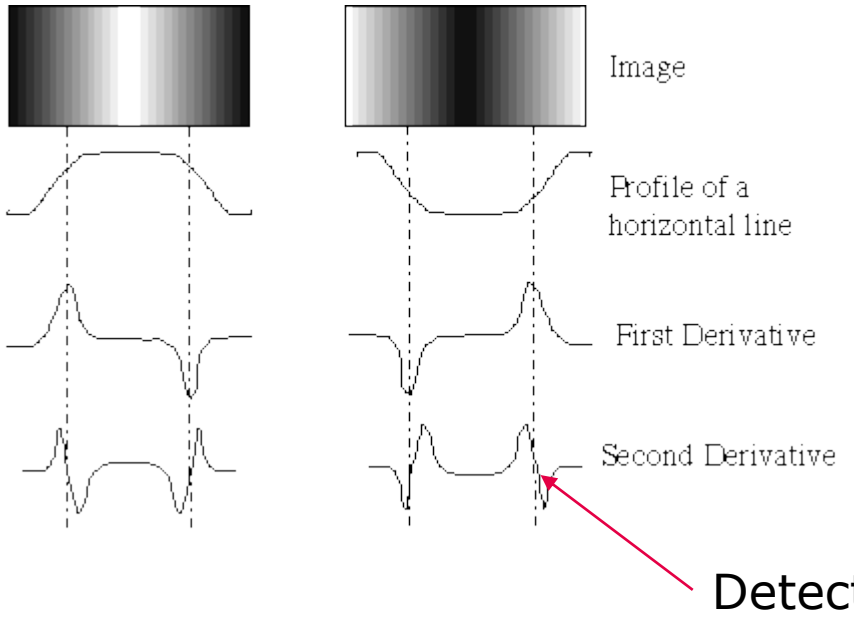
Derivative of Gaussian



Canny, IEEE TPAMI 1986

Edge Detection - Laplacian

Second Derivative



- Laplacians: sum of second derivatives:

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

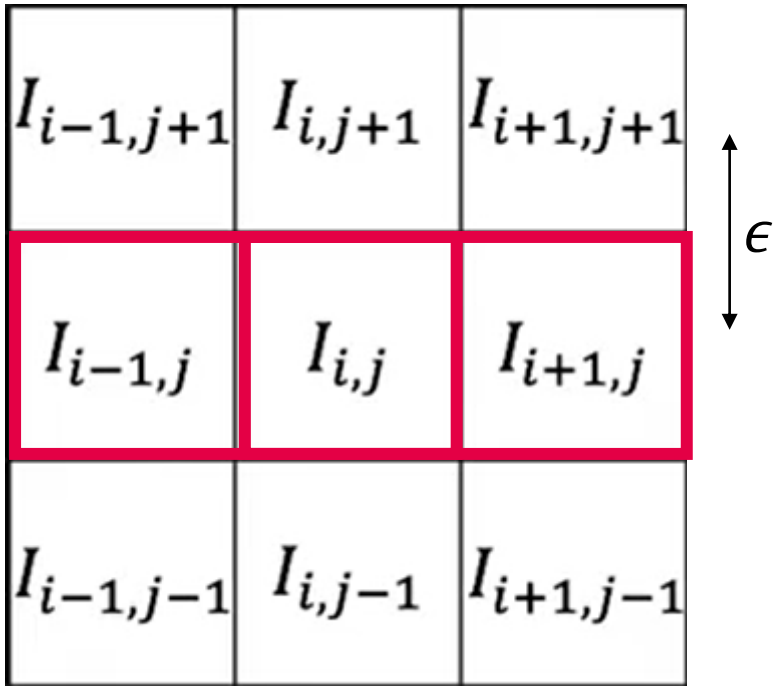
- Zero-crossings mark edge location
- Do not provide directions of edges

Edge Detection - Laplacian

Second Derivative – the difference of the difference

- $\frac{\partial^2 I}{\partial x^2} \approx \frac{1}{\epsilon^2}\left(I_{i-1,j} - 2I_{i,j} + I_{i+1,j}\right)$
- $\frac{\partial^2 I}{\partial y^2} \approx \frac{1}{\epsilon^2}\left(I_{i,j-1} - 2I_{i,j} + I_{i,j+1}\right)$
- $\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$

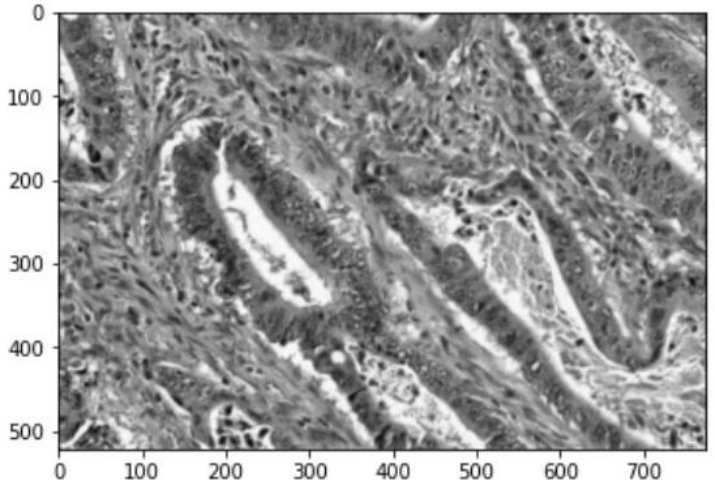
- $\nabla^2 = \frac{1}{\epsilon^2}\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ or (better) $\nabla^2 = \frac{1}{6\epsilon^2}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$

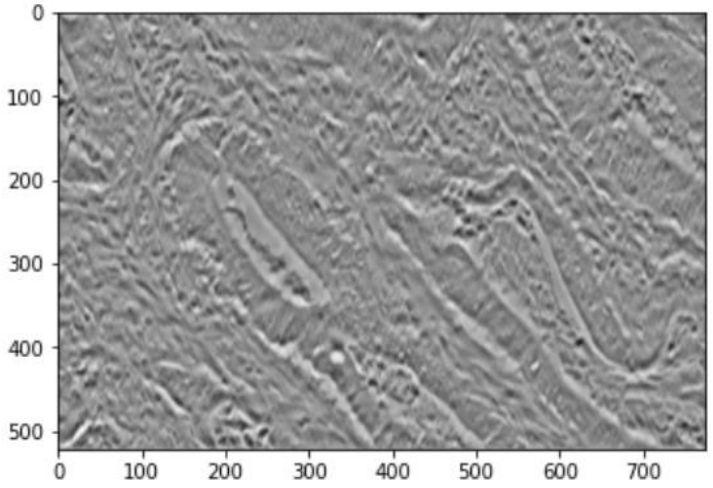


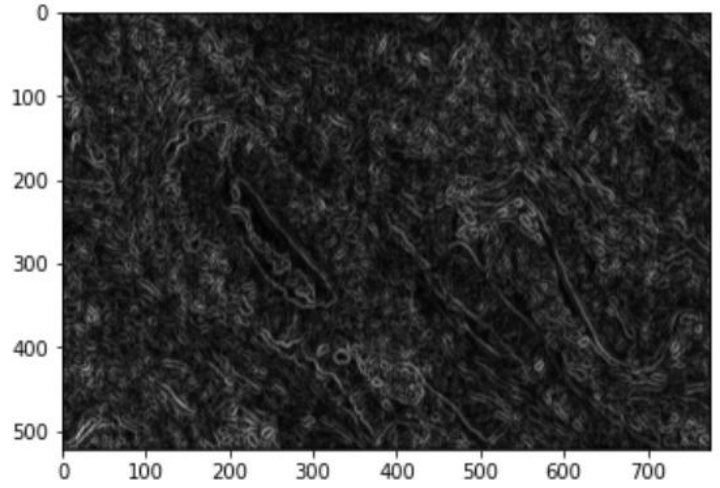
Edge Detection - Laplacian

Laplacian of a gaussian

$$\begin{pmatrix} 0 & 0 & 3 & 2 & 2 & 2 & 3 & 0 & 0 \\ 0 & 2 & 3 & 5 & 5 & 5 & 3 & 2 & 0 \\ 3 & 3 & 5 & 3 & 0 & 3 & 5 & 3 & 3 \\ 2 & 5 & 3 & -12 & -23 & -12 & 3 & 5 & 2 \\ 2 & 5 & 0 & -23 & -40 & -23 & 0 & 5 & 2 \\ 2 & 5 & 3 & -12 & -23 & -12 & 3 & 5 & 2 \end{pmatrix}$$

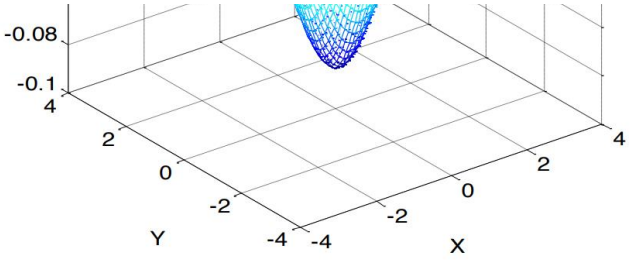




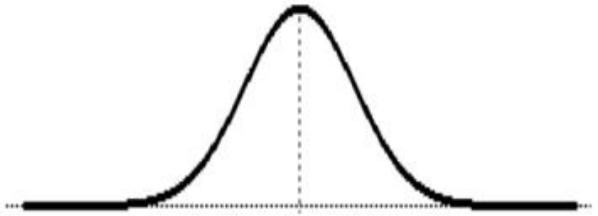
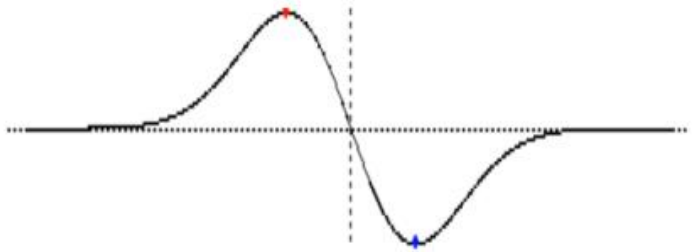
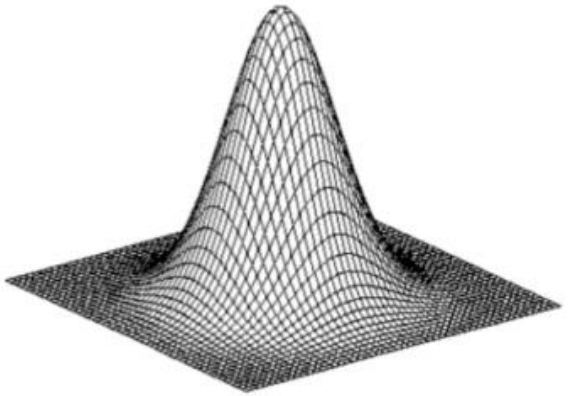
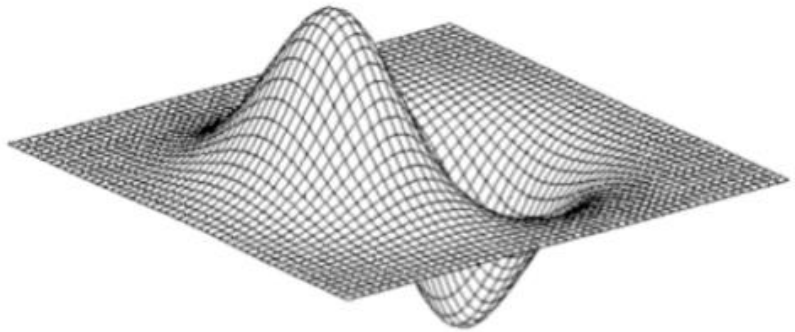


$\sqrt{2}$

$$Log(x,y) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$



(Partial) Derivative of Gaussian Kernel

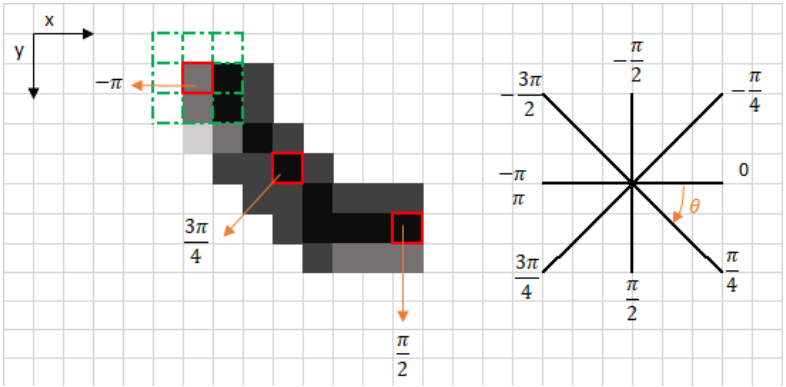
	$G(\cdot)$	$\frac{\partial G(\cdot)}{\partial x}$
1D		
2D		
3D	$\frac{1}{\left(\sigma\sqrt{2\pi}\right)^3} e^{-\frac{\left(x^2+y^2+z^2\right)}{2\sigma^2}}$	$-\frac{x}{\sigma^2\left(\sigma\sqrt{2\pi}\right)^3} e^{-\frac{\left(x^2+y^2+z^2\right)}{2\sigma^2}}$

A Computational Approach to Edge Detection

John Canny

Canny Edge Detection

Commonly used edge-detection filter



- Noise smoothing with a gaussian filter → $n_\sigma * I$
- Edge gradients (e.g., with Sobel operator) → $\nabla n_\sigma * I$
- Find Gradient Magnitude for each pixel → $\|\nabla n_\sigma * I\|$
- Find Gradient Orientation for each pixel → $\hat{n} = \frac{\nabla n_\sigma * I}{\|\nabla n_\sigma * I\|}$
- Compute the Laplacian along the gradient direction $\hat{n}$ at each pixel → $\frac{\partial^2 n_\sigma * I}{\partial \hat{n}^2}$

Canny Edge Detection

In other words

- Convolving with a derivative of Gaussian is the same as convolving with a Gaussian and then taking a derivative
  (…however, taking the *exact* derivative of an image is not well defined)
- It is (more or less) equivalent to blurring first, to reduce noise, then taking an exact derivative.

Canny Edge Detection

Final steps:

- Hysteresis thresholding:
  - First, global threshold with an upper threshold (on $|\nabla I|$)
  - Then, region grow starting from those regions using a lower threshold
- Non-maximum suppression
  - Eliminate edges that are not local maxima (along the direction of the gradient)

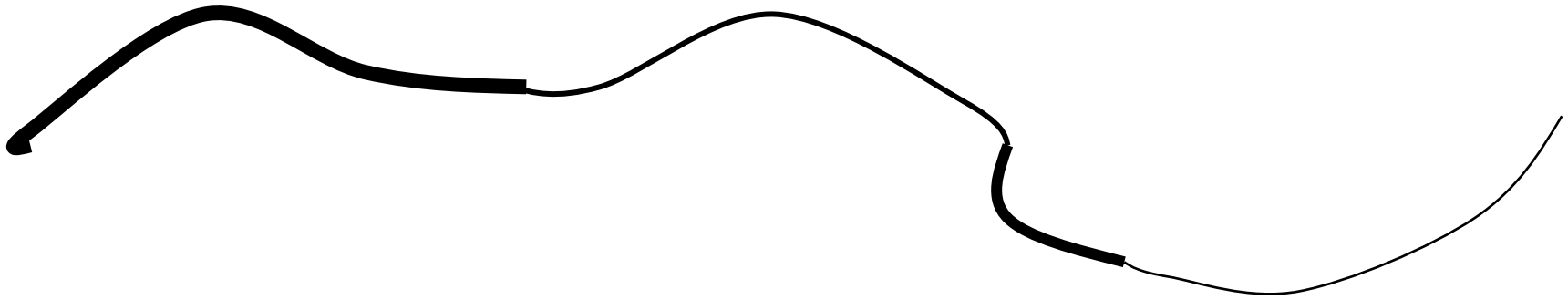
Hysteresis thresholding

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

Credit: James Hays

Hysteresis thresholding

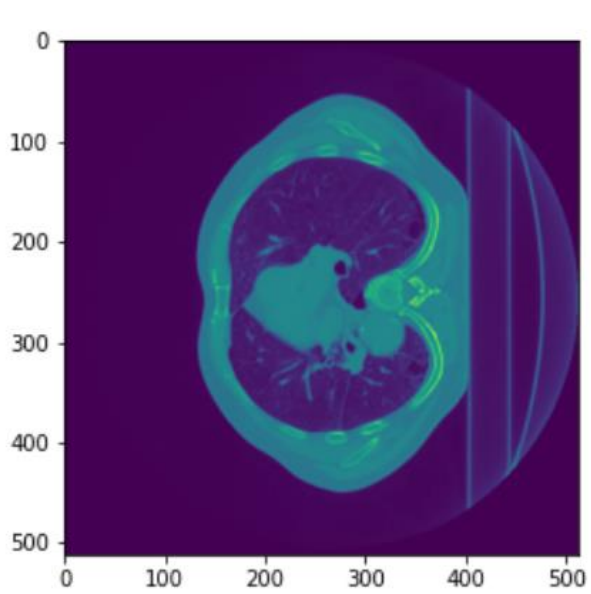
- Use a high threshold to start edge curves, and a low threshold to continue them.

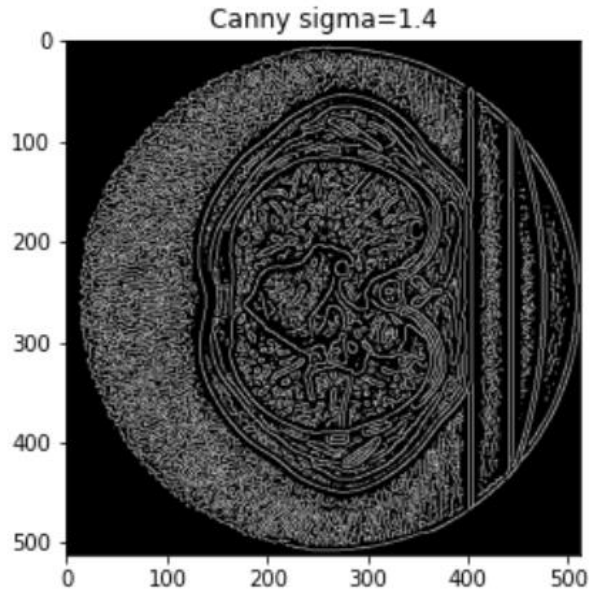


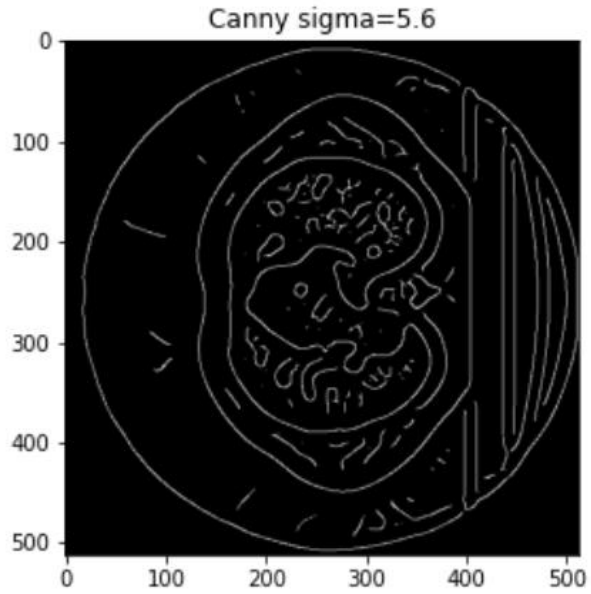
Source: Steve Seitz

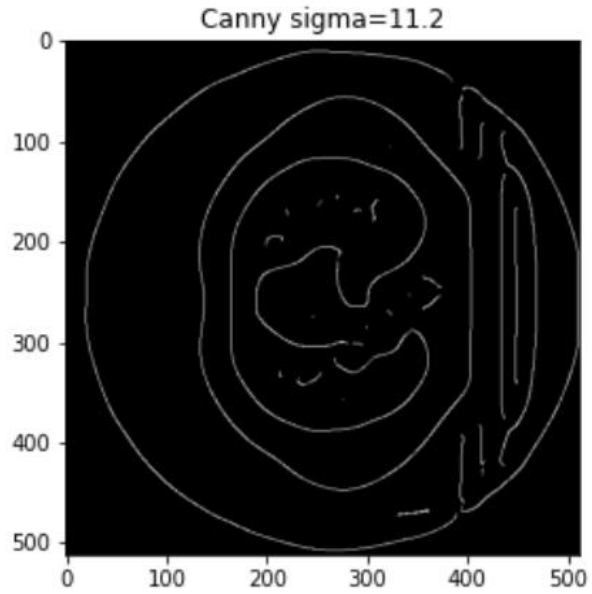
Canny Edge Detection

The filtering’s sigma value affects the results









Take-Aways

- There are many different segmentation methods
  - Thresholding
  - Region Growing
  - ML-Based
  - And even more is coming up soon:
    - Level-sets,
    - Fast-Marching,
    - Graph-based
- Edge Detection
  - Can produce disconnected boundaries
  - Over/Under Segmentation
  - Harder in 3D
  - Canny smooths before applying gradients
- No one algorithm is the “best”.
  - Must match to the problem and data