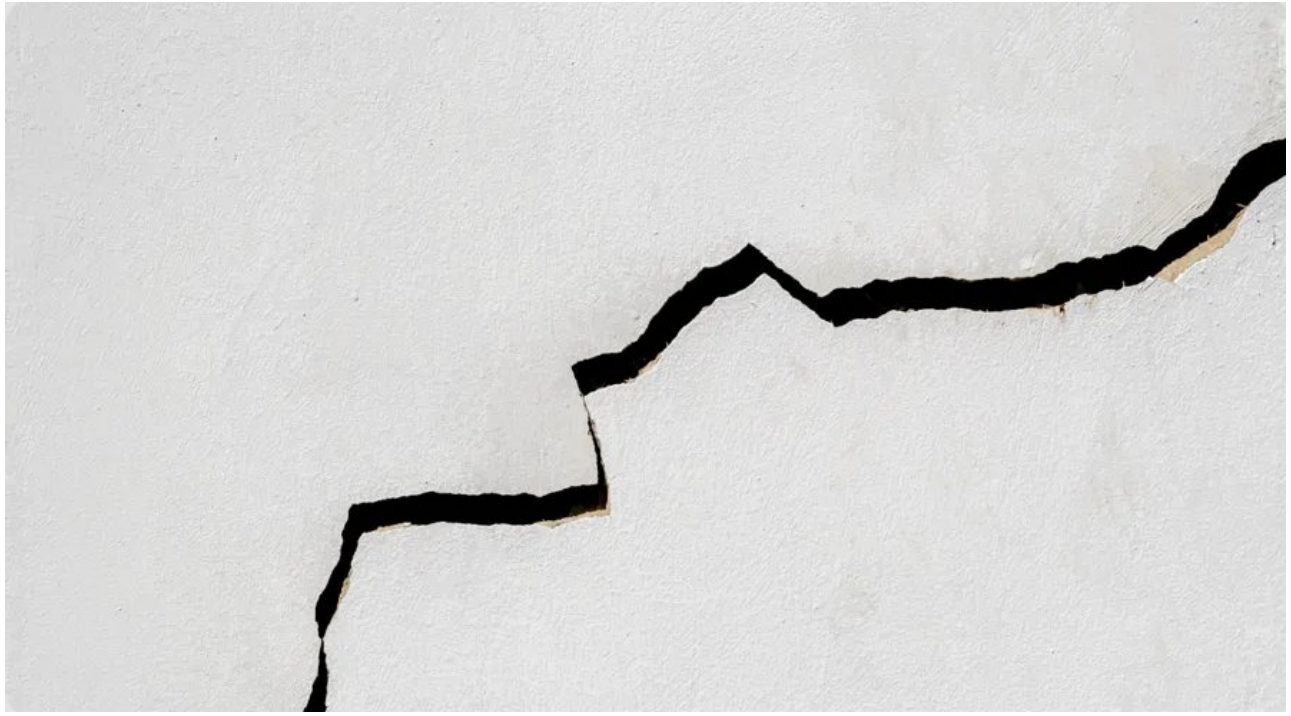
Edge Detection



https://liaoux.github.io/uir-cv-2026/

Introduction to Computer Vision
Spring 2026, Lecture 3

Edge Detection

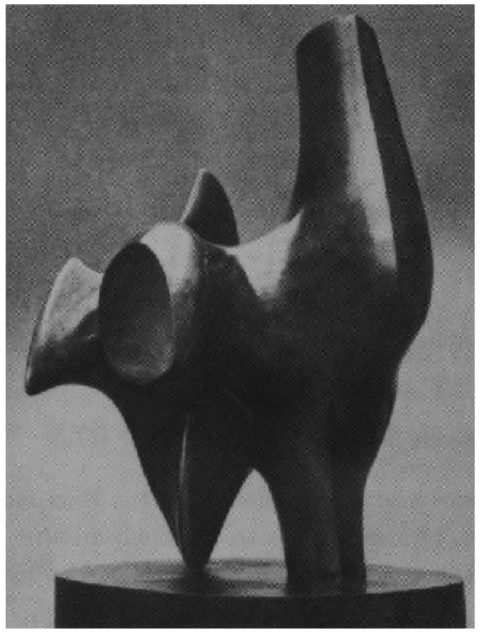
Convert a 2D image into a set of points where image intensity changes rapidly.

Topics:

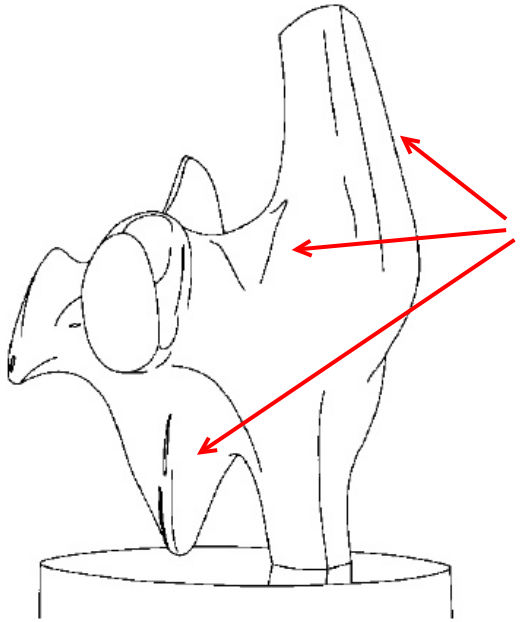
1. What is an Edge?
2. Edge Detection Using Gradients.
3. Edge Detection Using Laplacian.
4. Canny Edge Detector.

What is an Edge?

Rapid change in image intensity within small region

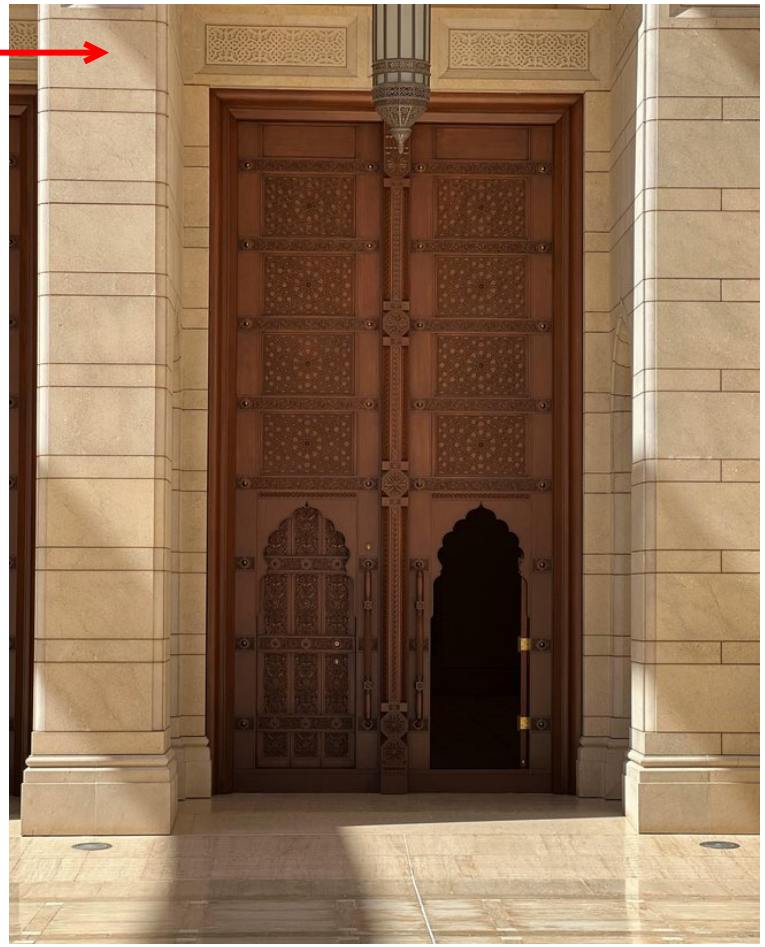


The Real Photograph



The Sketch by an Artist

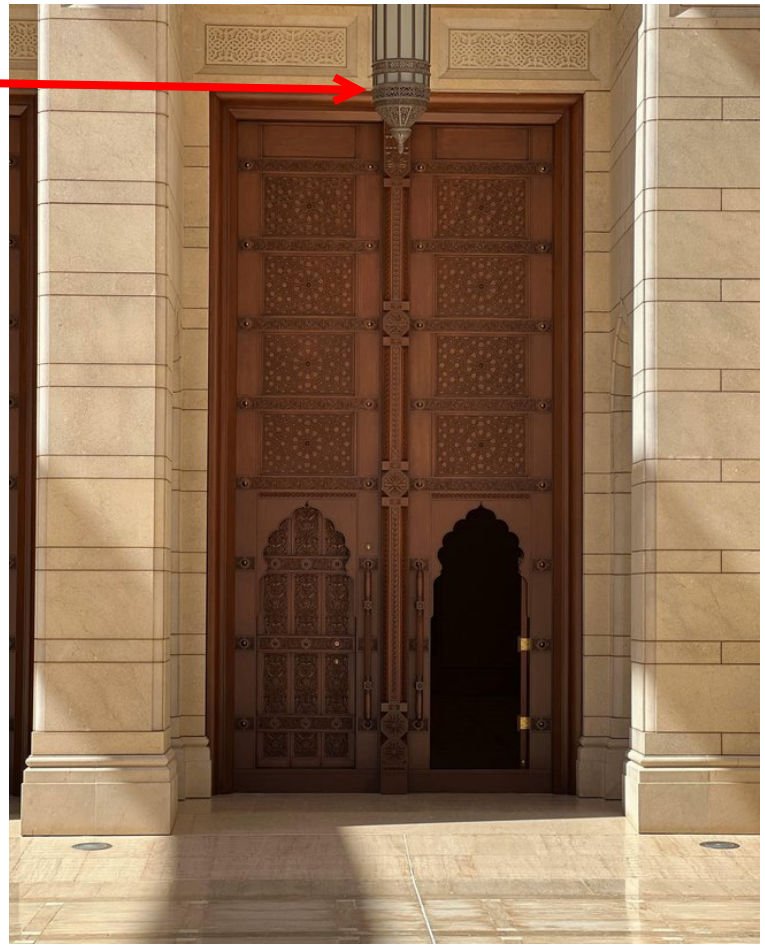
Causes of Edges



Edges created by Surface normal discontinuity

e.g.: Two surfaces made of the same material, having different surface orientations where they meet, they will likely receive different amounts of light from the light sources in the scene, and hence will have different brightness value

Causes of Edges



Edges created by depth discontinuity

e.g.: If one object is located in front of another, there will likely be a sudden change in intensity along the boundary between the two objects

Causes of Edges

Edges created by surface color discontinuity

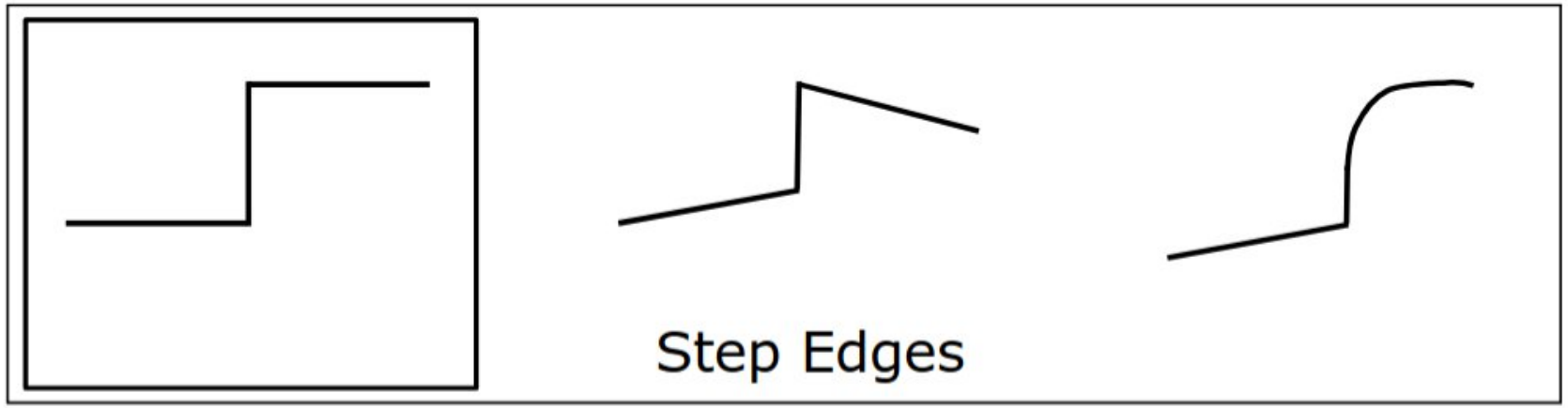
e.g.: Happens when a single surface changes reflectance properties, like texture or material.

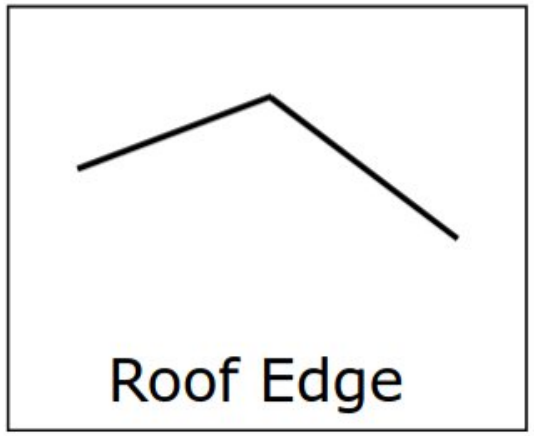
Causes of Edges

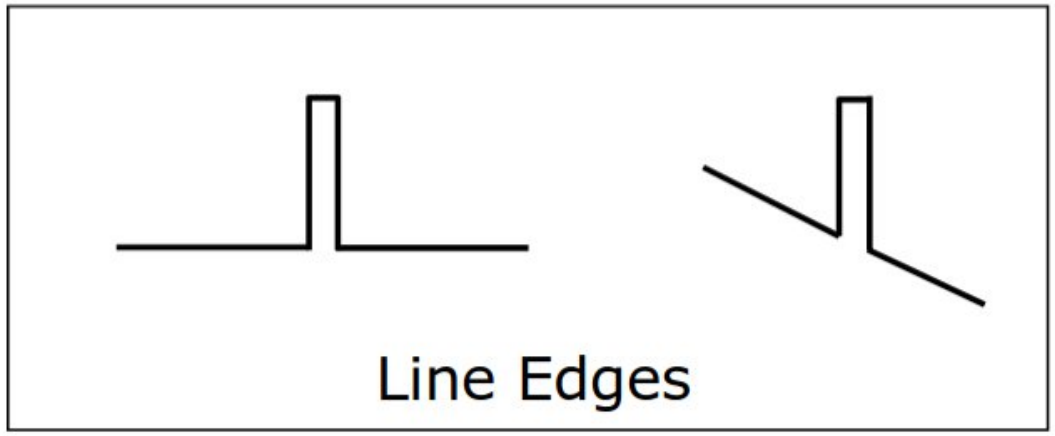
Edges created by
illumination discontinuity

e.g.: The wall casts a sharp shadow on the background, resulting in a significant difference in the amount of light falling within and outside the shadow

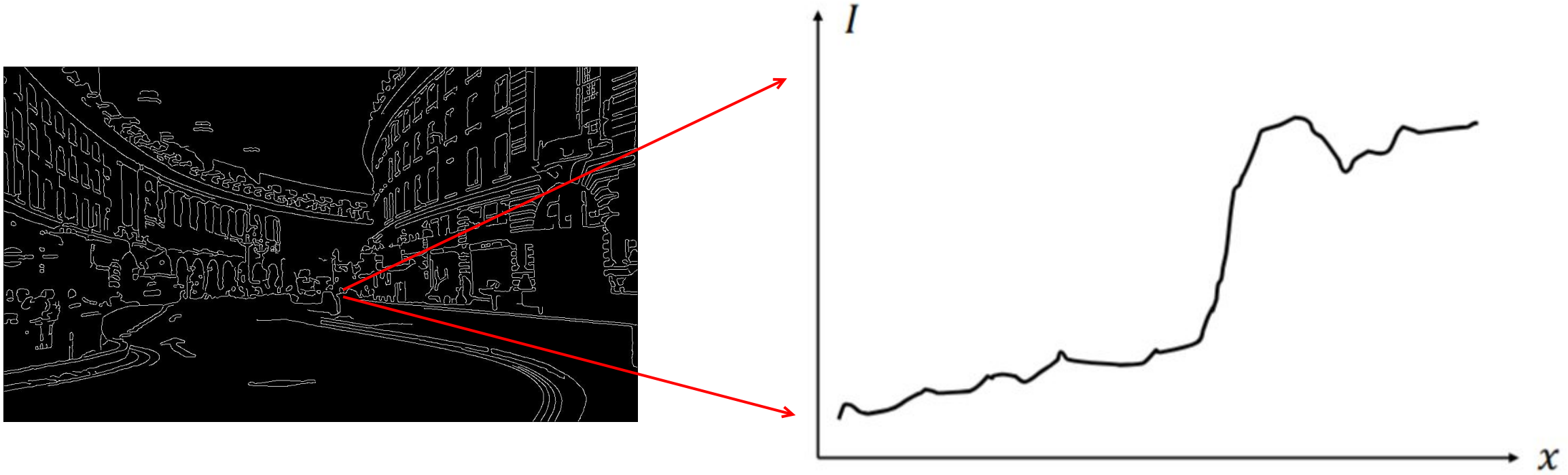
Types of edges







Real Edges



Problems: Noisy Images and Discrete Images

Edge Detector

We want an Edge Operator that produces:

- Edge Position
- Edge Magnitude (Strength)
- Edge Orientation (Direction)

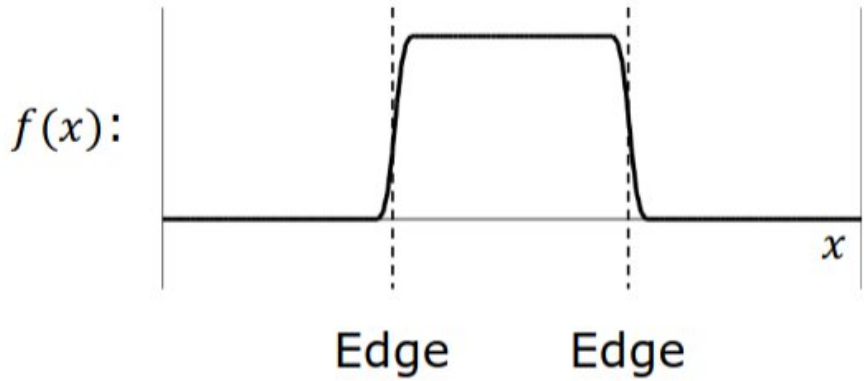
Performance Requirements:

- High Detection Rate
- Good Localization
- Low Noise Sensitivity

Edge Detection Using Gradients

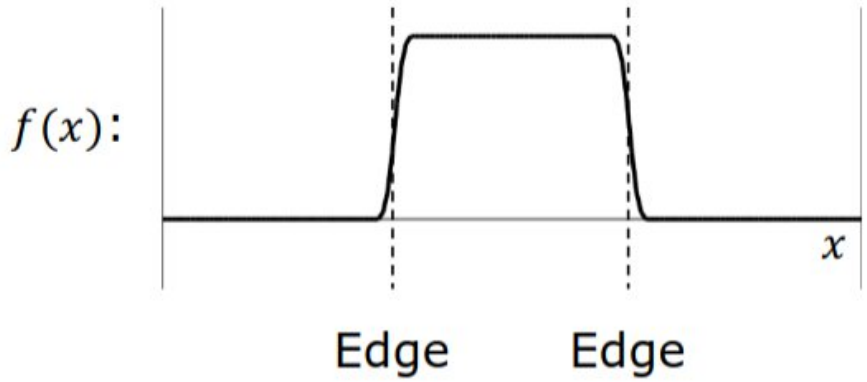
1D Edge Detection

Edge is a rapid change in image intensity in a small region.



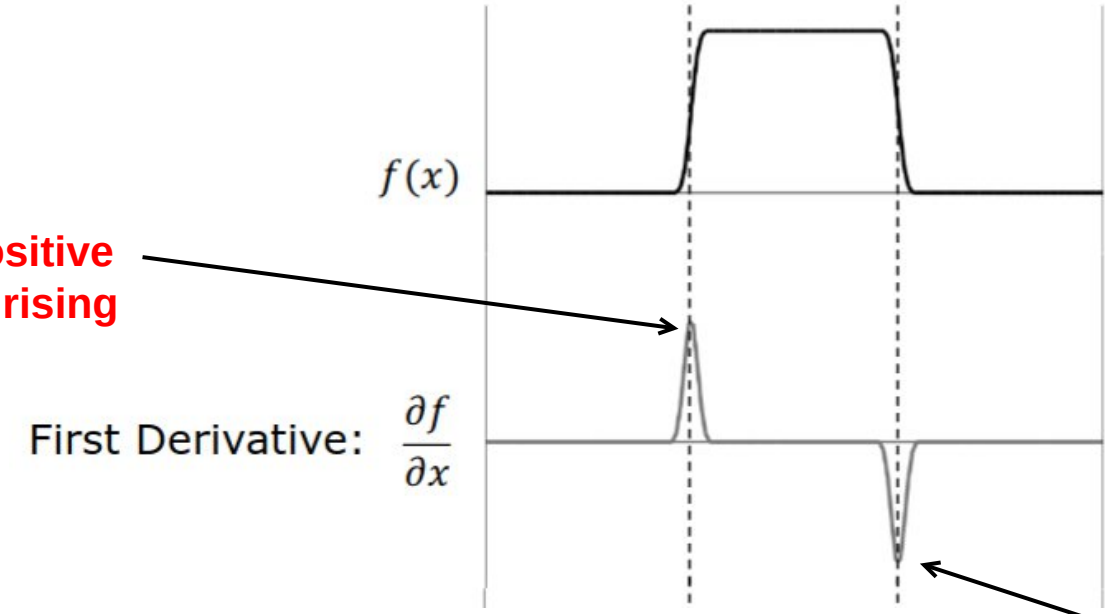
1D Edge Detection

Edge is a rapid change in image intensity in a small region.



Basic Calculus: Derivative of a continuous function represents the amount of change in the function.

Edge Detection Using 1^{st} Derivative

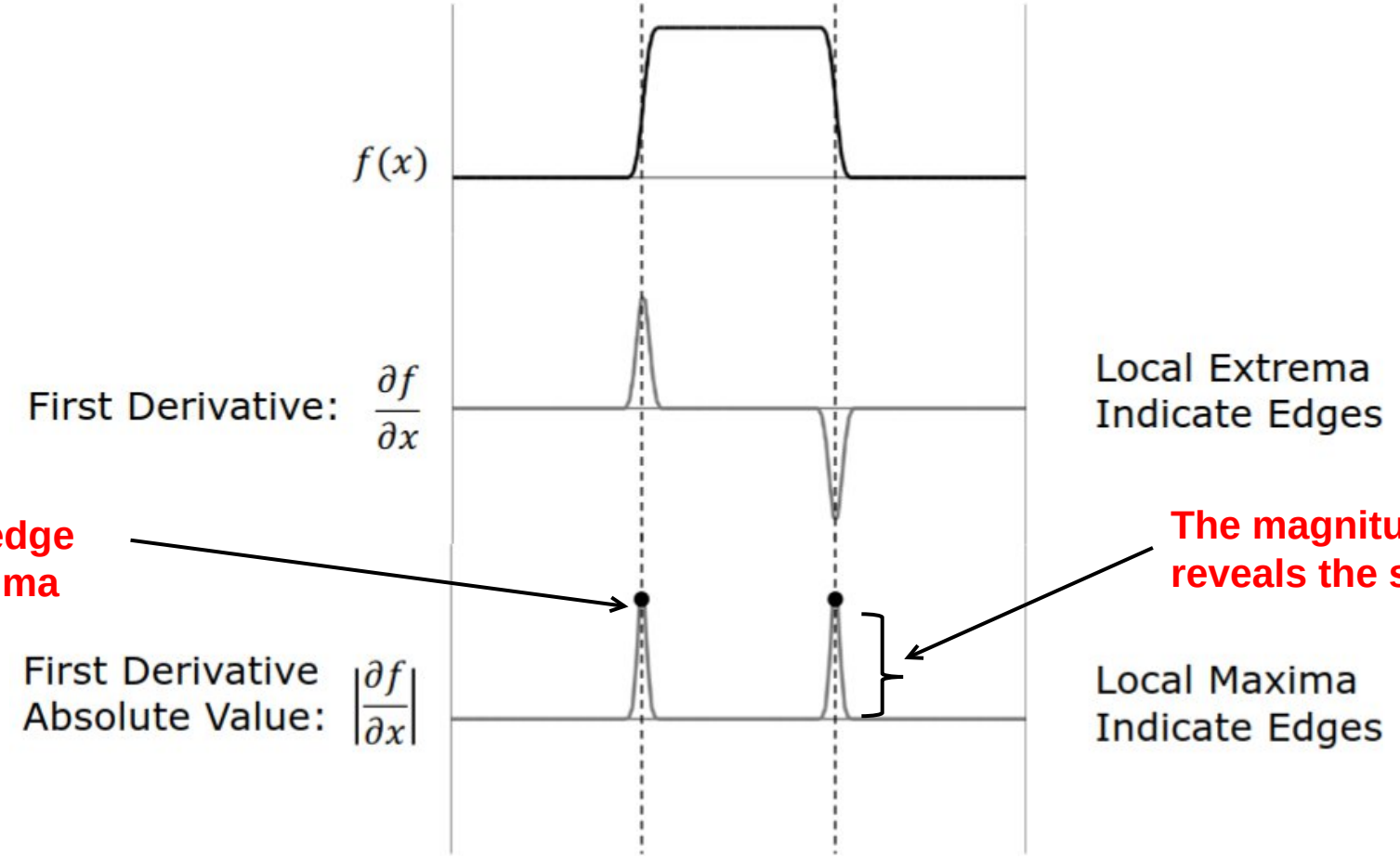


The derivative is positive because we have a rising edge

Local Extrema Indicate Edges

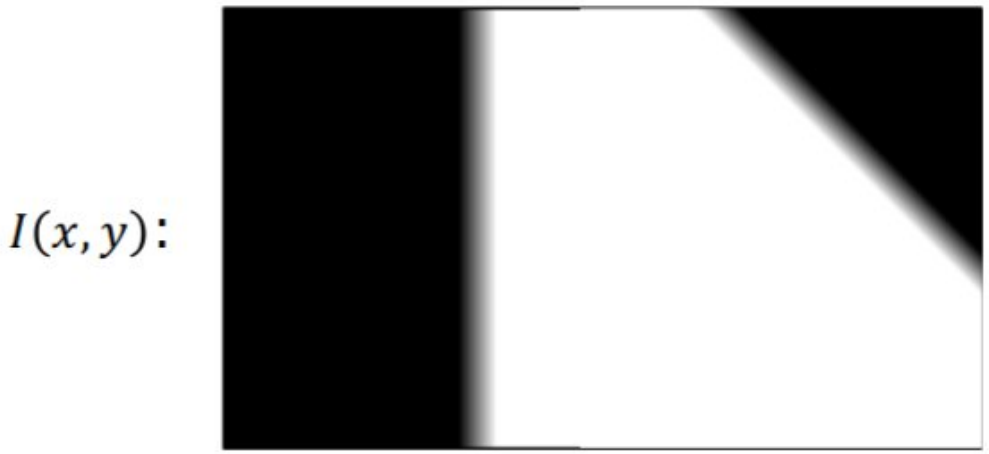
The derivative is negative because we have a falling edge

Edge Detection Using 1^{st} Derivative



Provides Both Location and Strength of an Edge

2D Edge Detection



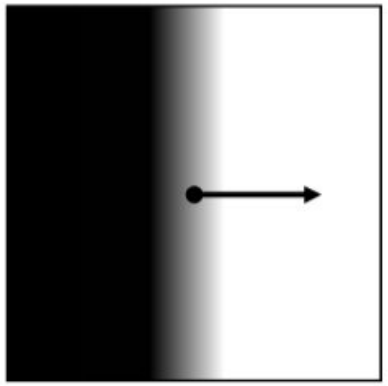
Basic Calculus: Partial Derivatives of a 2D continuous function represents the amount of change along each dimension.

Gradient (∇)

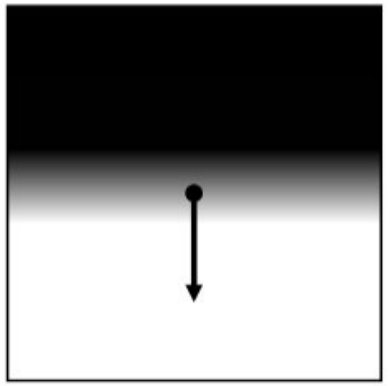
Gradient (Partial Derivatives) represents the direction of most rapid change in intensity

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

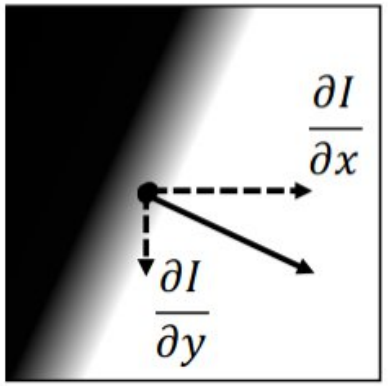
Pronounced as "Del I"



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



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



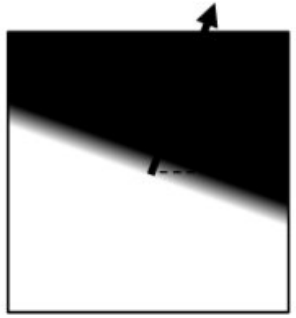
$$\nabla I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]$$

Gradient (∇) as Edge Detector

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

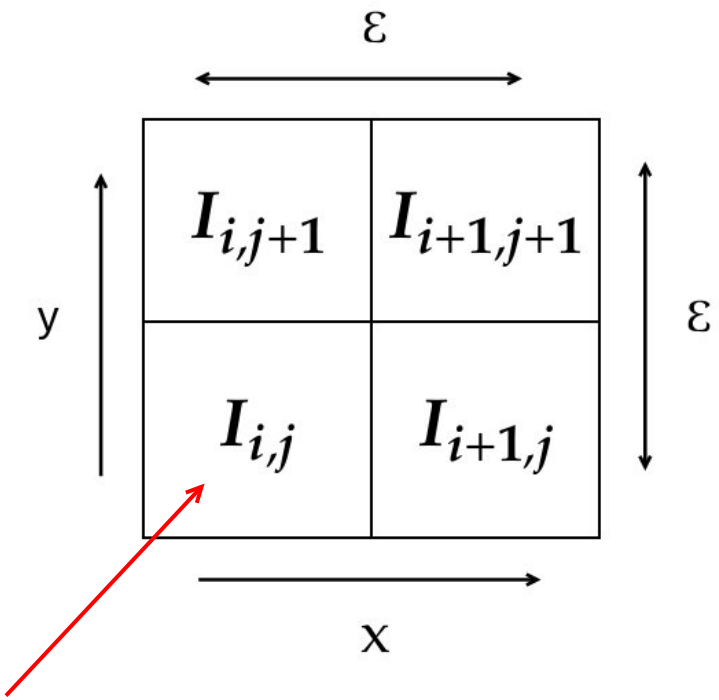
Gradient Orientation $\theta = \tan^{-1}\left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x}\right)$

Note: I represents the image intensity function



Discrete Gradient (∇) Operator

1. Moving **right** from $I_{i,j}$ to $I_{i+1,j}$:
   - i increases, j fixed, this is the x direction
   - i is the $x - index$
2. Moving **up** from $I_{i,j}$ to $I_{i,j+1}$:
   - j increases, i fixed, this is the y direction
   - j is the $y - index$
3. ε is the vertical and the horizontal spacing



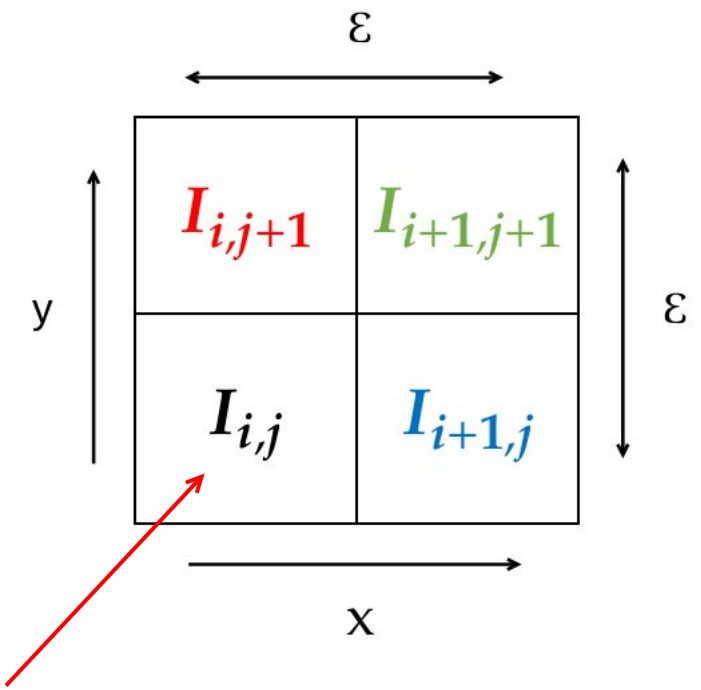
The pixel intensity at position (i,j)

Discrete Gradient (∇) Operator

Finite difference approximations:

$$\frac{\partial \mathrm{I}}{\partial x} = \frac{1}{2\varepsilon}\left[\left(I_{i+1,j+1} - I_{i,j+1}\right) + \left(I_{i+1,j} - I_{i,j}\right)\right]$$

$$\frac{\partial \mathrm{I}}{\partial \mathrm{y}} = \frac{1}{2\varepsilon}\left[\left(I_{i+1,j+1} - I_{i+1,j}\right) + \left(I_{i,j+1} - I_{i,j}\right)\right]$$



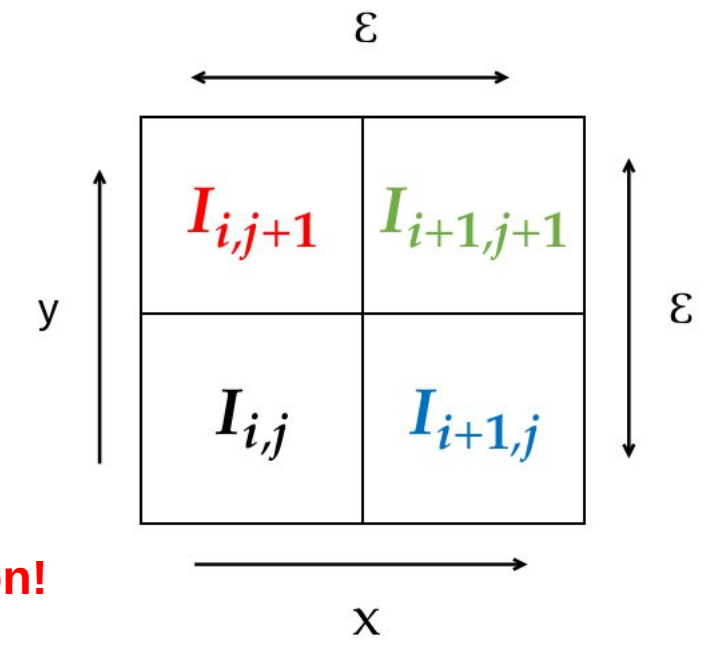
The pixel intensity at position (i,j)

Discrete Gradient (∇) Operator

Finite difference approximations:

$$\frac{\partial I}{\partial x} = \frac{1}{2\varepsilon}\left[\left(I_{i+1,j+1} - I_{i,j+1}\right) + \left(I_{i+1,j} - I_{i,j}\right)\right]$$

$$\frac{\partial I}{\partial y} = \frac{1}{2\varepsilon}\left[\left(I_{i+1,j+1} - I_{i+1,j}\right) + \left(I_{i,j+1} - I_{i,j}\right)\right]$$



Can be implemented as Convolution!

$$\frac{\partial I}{\partial x} = \frac{1}{2\varepsilon}\begin{bmatrix} -1 & +1 \\ -1 & +1 \end{bmatrix} \Rightarrow H_x = \begin{bmatrix} +1 & -1 \\ +1 & -1 \end{bmatrix}$$

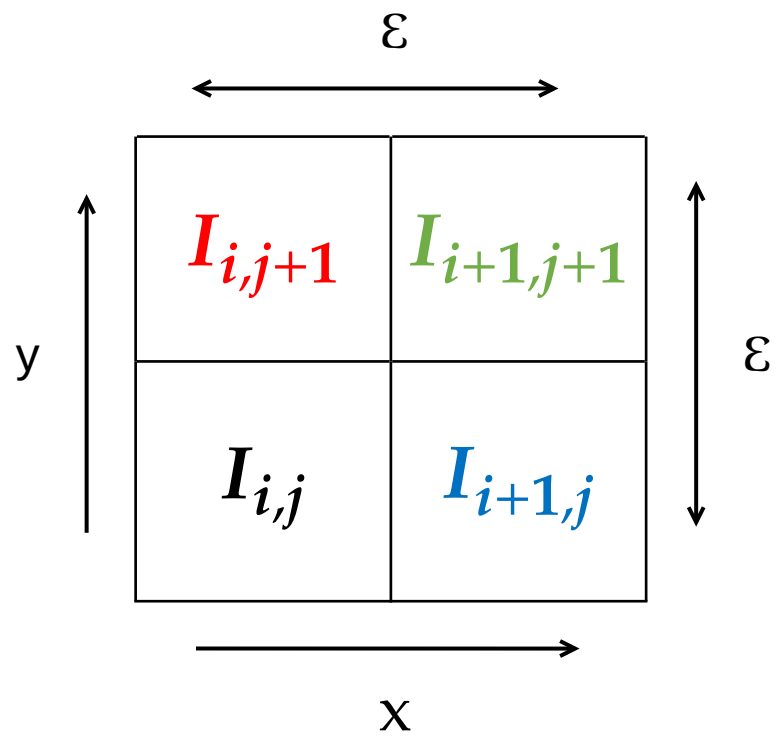
$$\frac{\partial I}{\partial y} = \frac{1}{2\varepsilon}\begin{bmatrix} +1 & +1 \\ -1 & -1 \end{bmatrix} \Rightarrow H_y = \begin{bmatrix} -1 & -1 \\ +1 & +1 \end{bmatrix}$$

Discrete Gradient (∇) Operator

Finite difference approximations:

$$\frac{\partial \mathrm{I}}{\partial x} = \frac{1}{2\mathcal{E}}\left[\left(I_{i+1,j+1} - I_{i,j+1}\right) + \left(I_{i+1,j} - I_{i,j}\right)\right]$$

$$\frac{\partial \mathrm{I}}{\partial \mathrm{y}} = \frac{1}{2\mathcal{E}}\left[\left(I_{i+1,j+1} - I_{i+1,j}\right) + \left(I_{i,j+1} - I_{i,j}\right)\right]$$



$$\frac{\partial \mathrm{I}}{\partial x} = \frac{1}{2\mathcal{E}} \begin{bmatrix} -1 & +1 \\ -1 & +1 \end{bmatrix} \Rightarrow H_x = \begin{bmatrix} +1 & -1 \\ +1 & -1 \end{bmatrix}$$

$$\frac{\partial \mathrm{I}}{\partial \mathrm{y}} = \frac{1}{2\mathcal{E}} \begin{bmatrix} +1 & +1 \\ -1 & -1 \end{bmatrix} \Rightarrow H_y = \begin{bmatrix} -1 & -1 \\ +1 & +1 \end{bmatrix}$$

Comparing Gradient (∇) Operators

Gradient	Roberts	Prewitt	Sobel (3x3)	Sobel (5x5)
$\frac{\partial I}{\partial x}$	0 1 -1 0	-1 0 1 -1 0 1 -1 0 1	-1 0 1 -2 0 2 -1 0 1	-1 -2 0 2 1 -2 -3 0 3 2 -3 -5 0 5 3 -2 -3 0 3 2 -1 -2 0 2 1
$\frac{\partial I}{\partial y}$	1 0 0 -1	1 1 1 0 0 0 -1 -1 -1	1 2 1 0 0 0 -1 -2 -1	1 2 3 2 1 2 3 5 3 2 0 0 0 0 0 -2 -3 -5 -3 -2 -1 -2 -3 -2 -1

- Good Localization
- Noise Sensitive
- Poor Detection

- Poor Localization
- Less Noise Sensitive
- Good Detection

Gradient (∇) Using Sobel Filter

The Lena Image

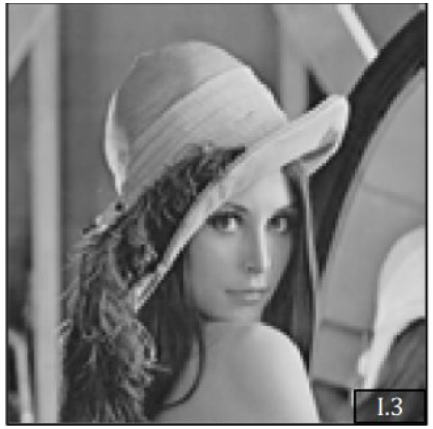


Image (I)

$\partial I / \partial x$

$\partial I / \partial y$

Gradient Magnitude

Gradient (∇) Using Sobel Filter

The Lena Image

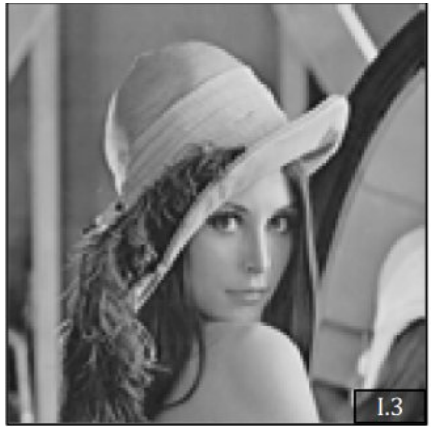


Image (I)

How to declare that a pixel is an edge ?

$\partial I / \partial x$

$\partial I / \partial y$

Gradient Magnitude

Edge Thresholding

Standard: (Single Threshold T)

$\|\nabla I(x,y)\| < T$ Definitely Not an Edge

$\|\nabla I(x,y)\| \geq T$ Definitely an Edge

Hysteresis Based: (Two Thresholds $T_0 < T_1$)

$\|\nabla I(x,y)\| < T_0$ Definitely Not an Edge

$\|\nabla I(x,y)\| \geq T_1$ Definitely an Edge

$T_0 \leq \|\nabla I(x,y)\| < T_1$ Is an Edge if a Neighboring Pixel is Definitely an Edge

Sobel Edge Detector

Image (I)

$\partial I/\partial x$

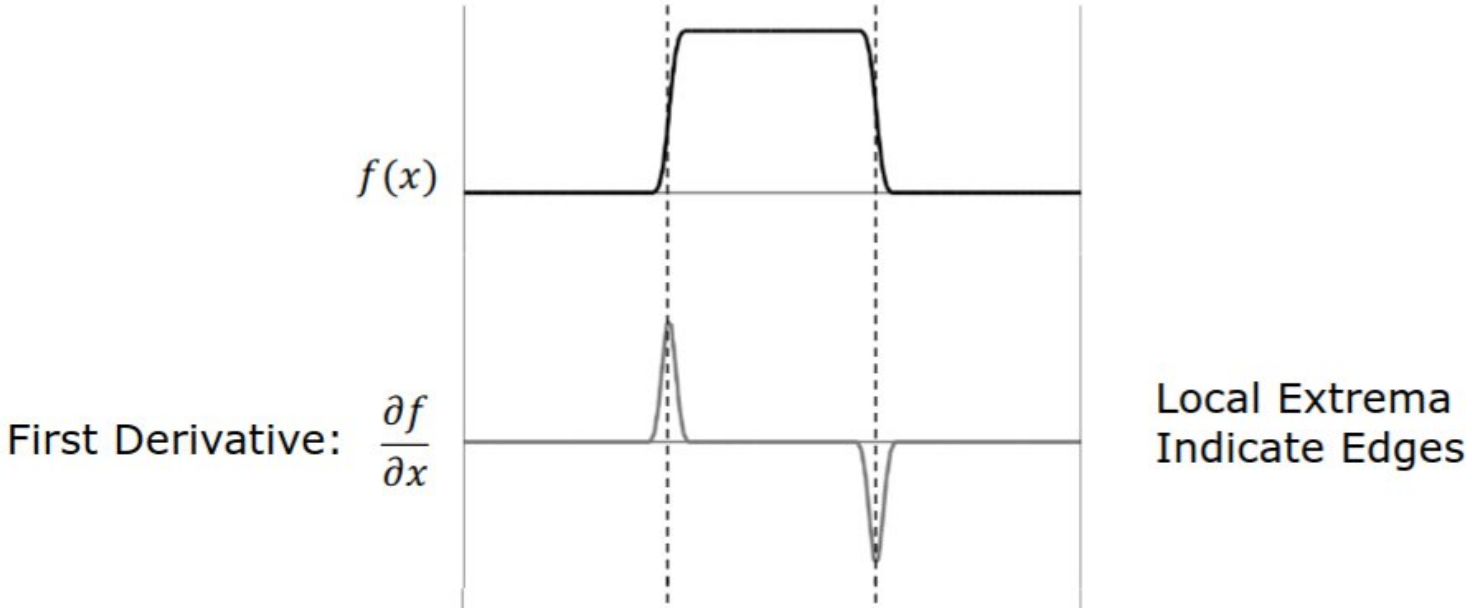
$\partial I/\partial y$

Gradient Magnitude

Thresholded Edge

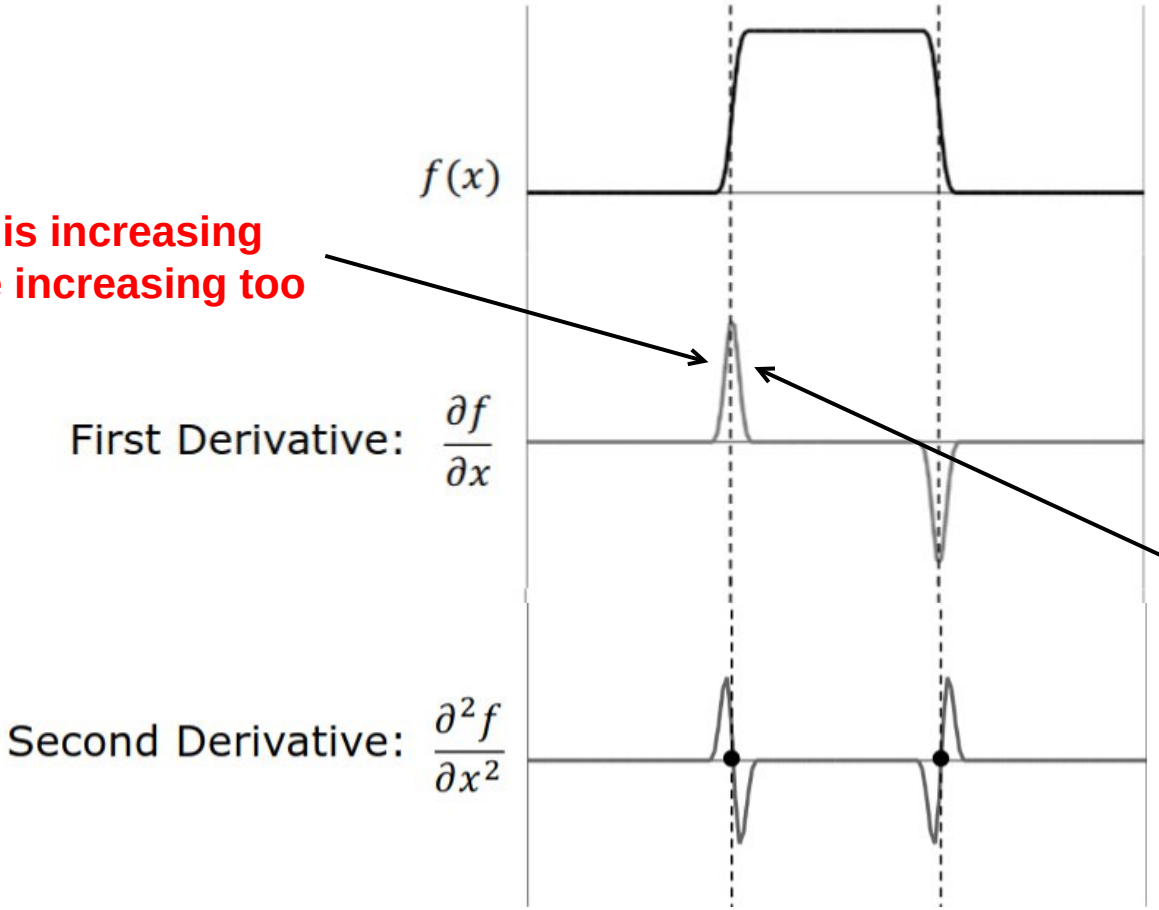
Edge Detection Using Laplacian

Edge Detection Using 1^{st} Derivative



What is the derivative of the derivative ?

Edge Detection Using 2^{nd} Derivative



When the derivative is increasing it's derivative will be increasing too

Local Extrema Indicate Edges

When the derivative is decreasing it's decreasing will be increasing too

Zero-Crossings Indicate Edges

Laplacian (∇^2) as Edge Detector

Laplacian: Sum of Pure Second Derivatives

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

Pronounced as "Del Square I"

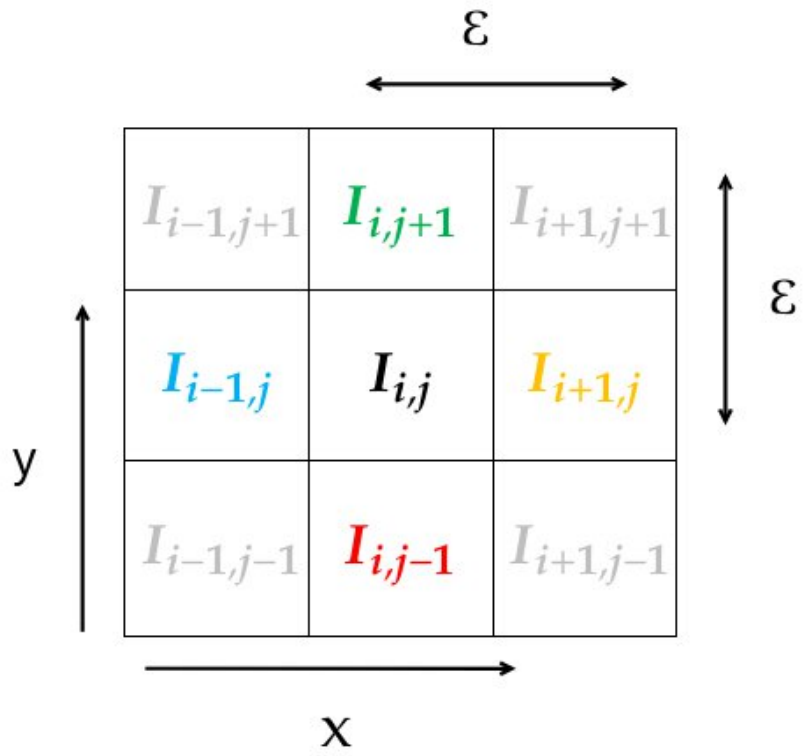
- Edges are "zero-crossings" in Laplacian of image
- Laplacian does not provide directions of edges

Discrete Laplacian (∇^2) Operator

Finite difference approximations:

$$\frac{\partial^2 \mathbf{I}}{\partial x^2} = \frac{1}{\varepsilon^2}\left[I_{i-1,j} - 2I_{i,j} + I_{i+1,j}\right]$$

$$\frac{\partial^2 \mathbf{I}}{\partial y^2} = \frac{1}{\varepsilon^2}\left[I_{i,j-1} - 2I_{i,j} + I_{i,j+1}\right]$$



$$\textit{Laplacian} = \nabla^2 I = \frac{\partial^2 \mathbf{I}}{\partial x^2} + \frac{\partial^2 \mathbf{I}}{\partial y^2} = \frac{1}{\varepsilon^2}\left[I_{i-1,j} + I_{i,j-1} - 4I_{i,j} + I_{i+1,j} + I_{i,j+1}\right]$$

Discrete Laplacian (∇^2) Operator

Finite difference approximations:

$$\frac{\partial^2 \mathbf{I}}{\partial x^2} = \frac{1}{\varepsilon^2}\left[I_{i-1,j} - 2I_{i,j} + I_{i+1,j}\right]$$

$$\frac{\partial^2 \mathbf{I}}{\partial y^2} = \frac{1}{\varepsilon^2}\left[I_{i,j-1} - 2I_{i,j} + I_{i,j+1}\right]$$

Convolution Mask:

$$\frac{1}{\varepsilon^2}\begin{bmatrix} 0 & +1 & 0 \\ +1 & -4 & +1 \\ 0 & +1 & 0 \end{bmatrix}$$

$$Laplacian = \nabla^2 I = \frac{\partial^2 \mathbf{I}}{\partial x^2} + \frac{\partial^2 \mathbf{I}}{\partial y^2} = \frac{1}{\varepsilon^2}\left[I_{i-1,j} + I_{i,j-1} - 4I_{i,j} + I_{i+1,j} + I_{i,j+1}\right]$$

Discrete Laplacian (∇^2) Operator

More Accurate Version — Idea:

- The basic kernel only measures curvature along x and y
- Extend to all 4 directions: horizontal, vertical, and both diagonals
- Apply the same 3-point formula along each diagonal (step = $\varepsilon\sqrt{2}$)
- Average all four estimates
- Scale to integer weights, axis neighbors get weight 4, diagonal neighbors get weight 1, center = −20

Convolution Mask:

$$\frac{1}{6\varepsilon^2}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

Laplacian Edge Detector

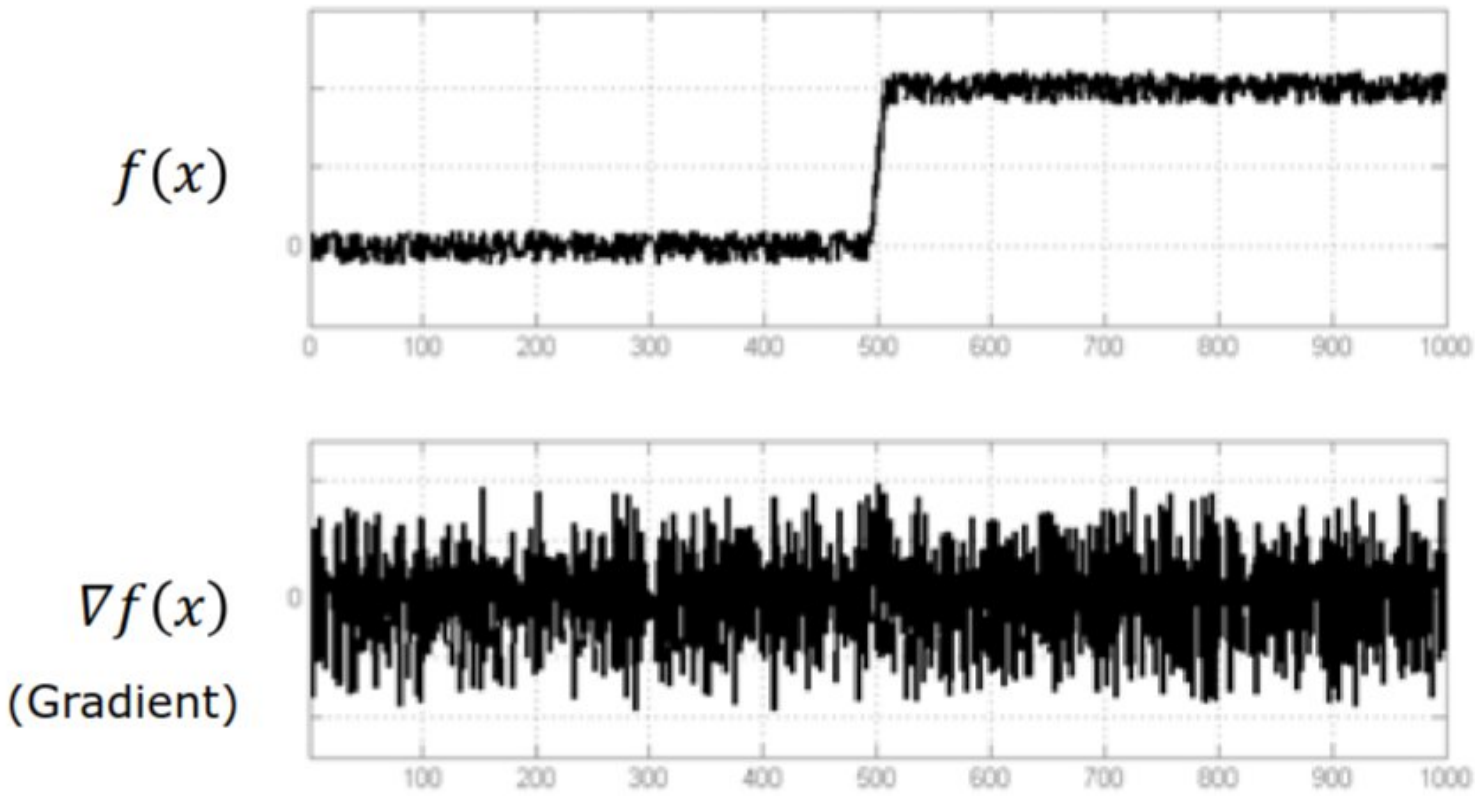
Image (I)


Laplacian
(0 maps to 128)


Laplacian
"Zero Crossings"

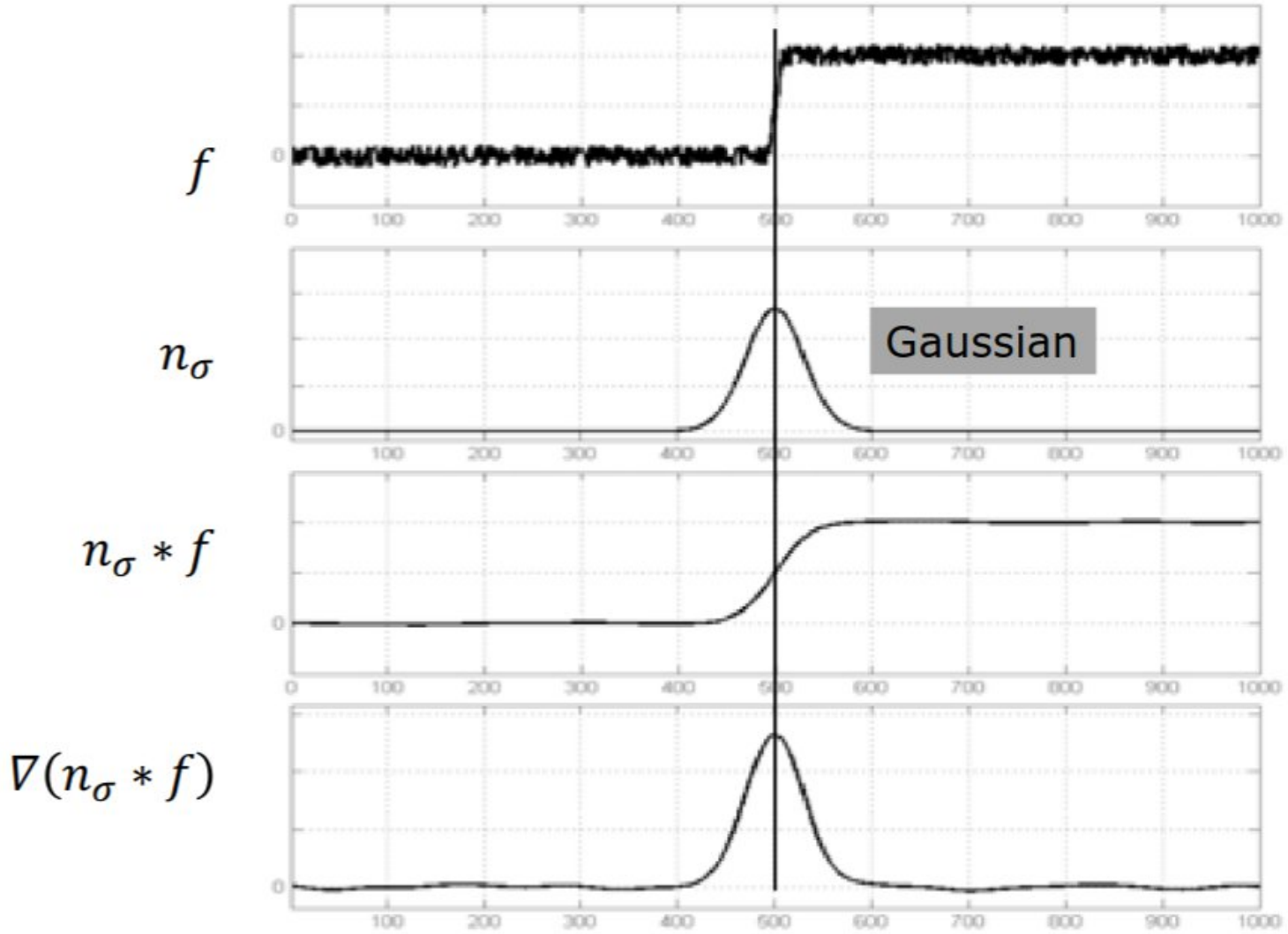
- Since the output contains **negative values**, a level of **128 is used as zero** — pixels below 128 are negative, above 128 are positive
- At an edge, the Laplacian crosses from positive to negative (or vice versa) — this is called a **zero-crossing**
- At a zero-crossing, the pixel value is **exactly 128**, surrounded by large positive and negative values
- These **zero-crossings are detected** to produce the final binary edge map

Effects of Noise



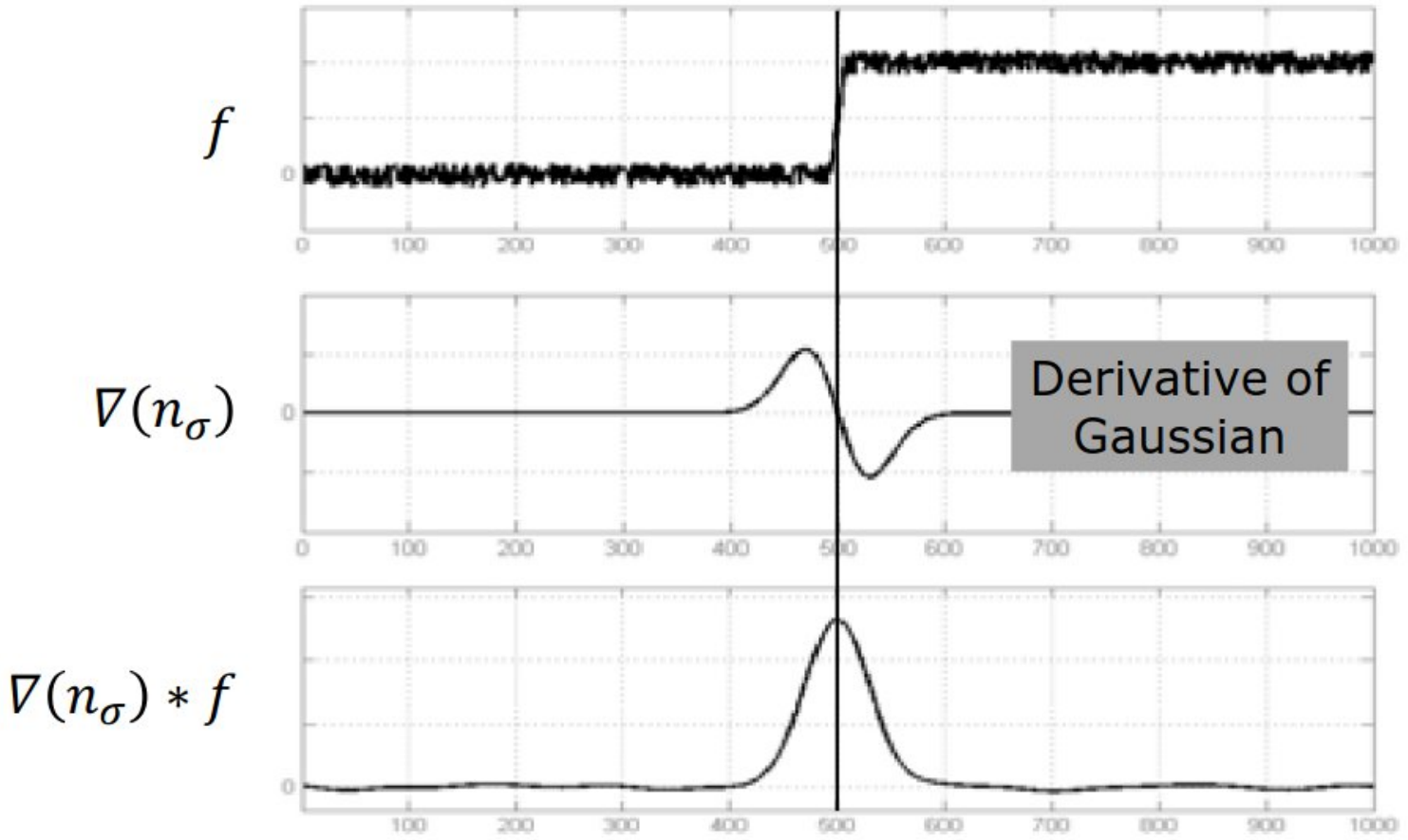
Where is the edge??

Solution: Gaussian Smooth First



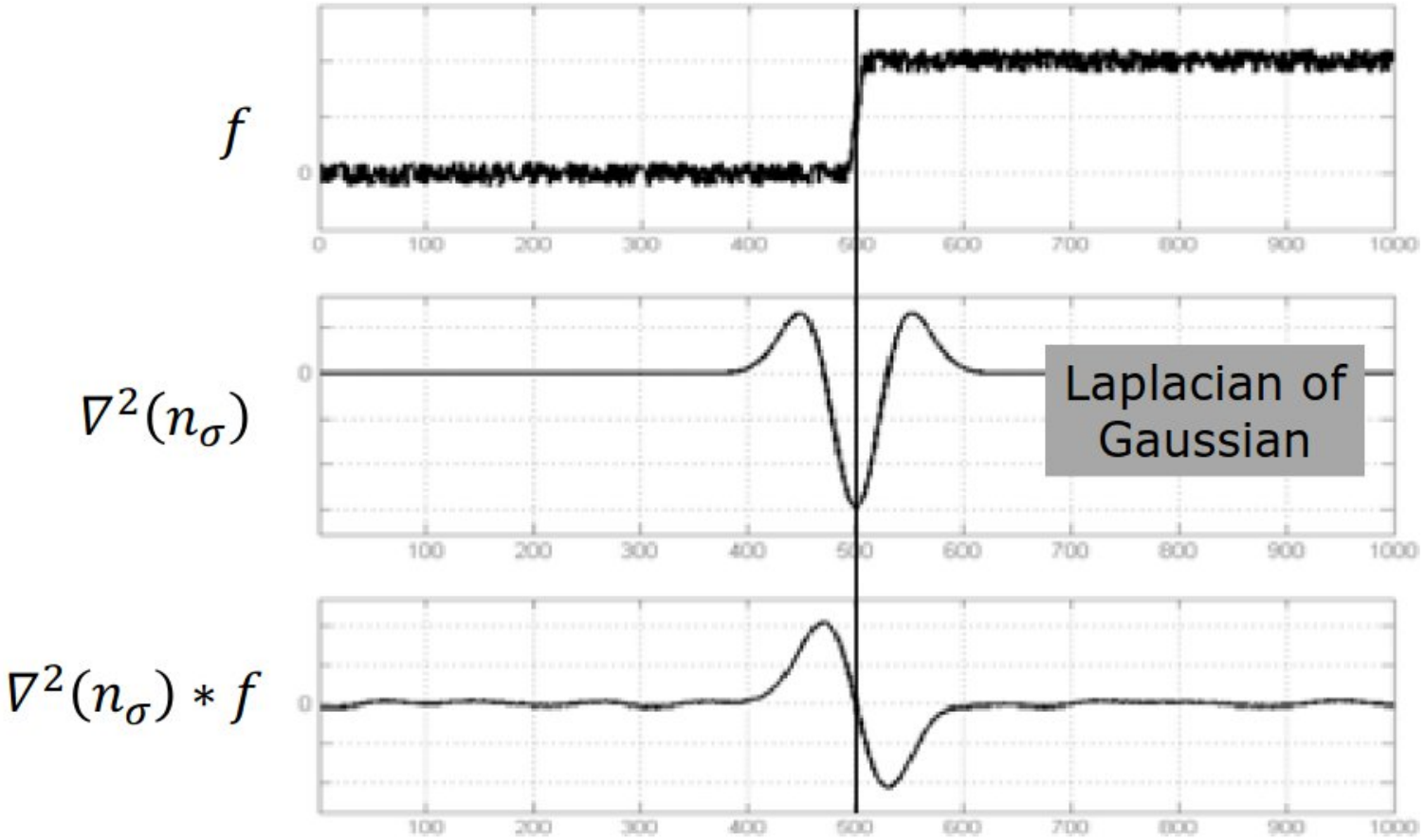
Derivative of Gaussian $\nabla(\eta_\sigma)$

$\nabla(n_\sigma * f) = \nabla(n_\sigma) * f$ ...saves us one operation.

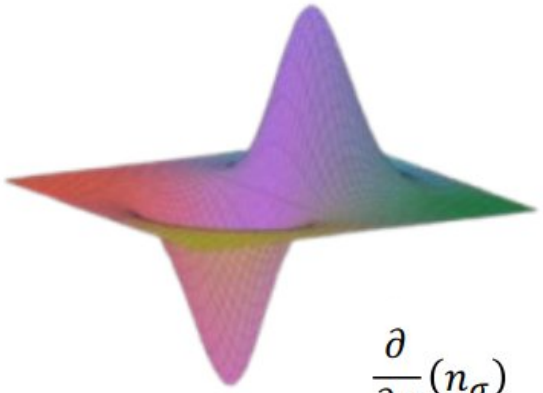
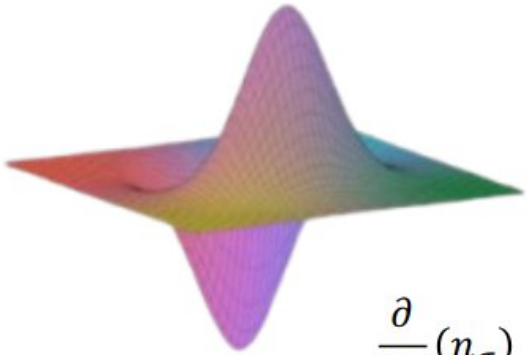
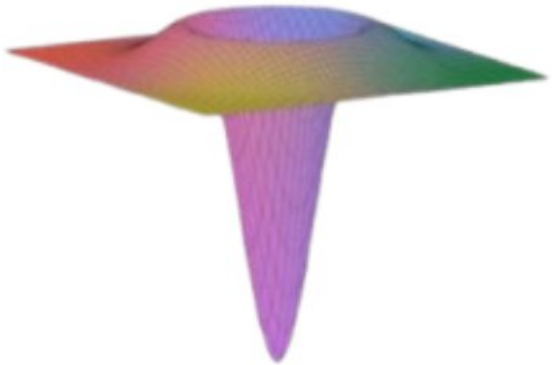


Derivative of Laplacian $\nabla^2(\eta_\sigma)$

$\nabla^2(n_\sigma * f) = \nabla^2(n_\sigma) * f$ ...saves us one operation.



Gradient vs. Laplacian

Derivative of Gaussian (∇G)	Laplacian of Gaussian ($\nabla^2 G$)
 $\frac{\partial}{\partial x}(n_\sigma)$  $\frac{\partial}{\partial y}(n_\sigma)$	 $\frac{\partial^2}{\partial x^2}(n_\sigma) + \frac{\partial^2}{\partial y^2}(n_\sigma)$ Inverted "Sombrero" (Mexican Hat)

Gradient vs. Laplacian

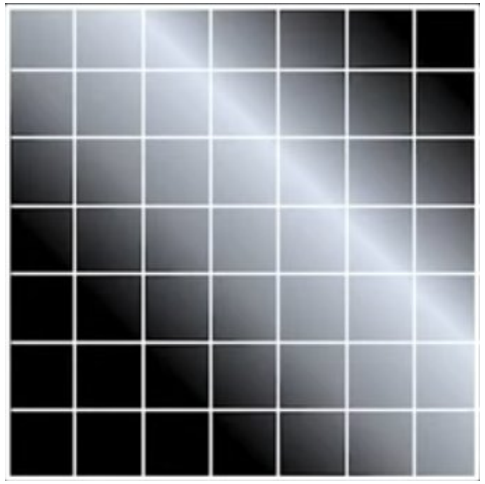
Gradient	Laplacian
Provides location, magnitude and direction of the edge.	Provides only location of the edge.
Detection using Maxima Thresholding.	Detection based on Zero-Crossing.
Non-linear operation. Requires two convolutions.	Linear Operation. Requires only one convolution.

An operator that has the best of both?

Canny Edge Detector

Canny Edge Detector

- Smooth Image with 2D Gaussian: $n_\sigma * I$
- Compute Image Gradient using Sobel Operator: $\nabla n_\sigma * I$
- Find Gradient Magnitude at each pixel: $\|\nabla n_\sigma * I\|$

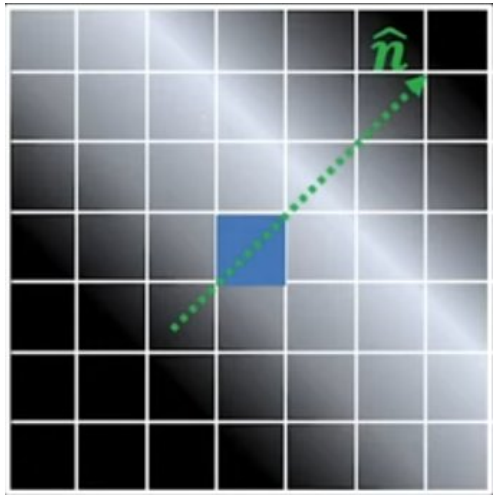


$\|\nabla n_\sigma * I\|$

Canny Edge Detector

- Smooth Image with 2D Gaussian: $n_\sigma * I$
- Compute Image Gradient using Sobel Operator: $\nabla n_\sigma * I$
- Find Gradient Magnitude at each pixel: $\|\nabla n_\sigma * I\|$
- Find Gradient Orientation at each Pixel:

$$\hat{\boldsymbol{n}} = \frac{\nabla n_\sigma * I}{\|\nabla n_\sigma * I\|}$$



$\|\nabla n_\sigma * I\|$

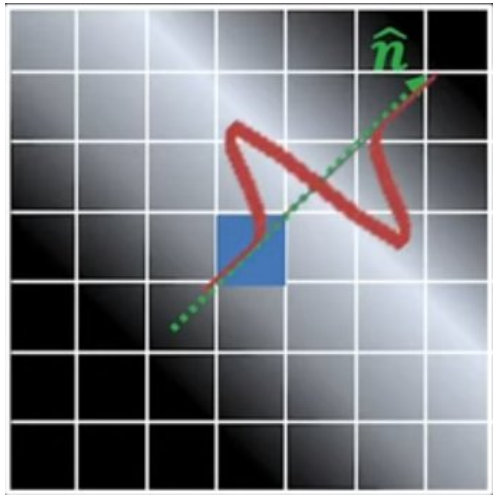
Canny Edge Detector

- Smooth Image with 2D Gaussian: $n_\sigma * I$
- Compute Image Gradient using Sobel Operator: $\nabla n_\sigma * I$
- Find Gradient Magnitude at each pixel: $\|\nabla n_\sigma * I\|$
- Find Gradient Orientation at each Pixel:

$$\hat{\boldsymbol{n}} = \frac{\nabla n_\sigma * I}{\|\nabla n_\sigma * I\|}$$

- Compute Laplacian along the Gradient Direction $\hat{\boldsymbol{n}}$ at each pixel

$$\frac{\partial^2 (n_\sigma * I)}{\partial \hat{\boldsymbol{n}}^2}$$



$\|\nabla n_\sigma * I\|$

Canny Edge Detector

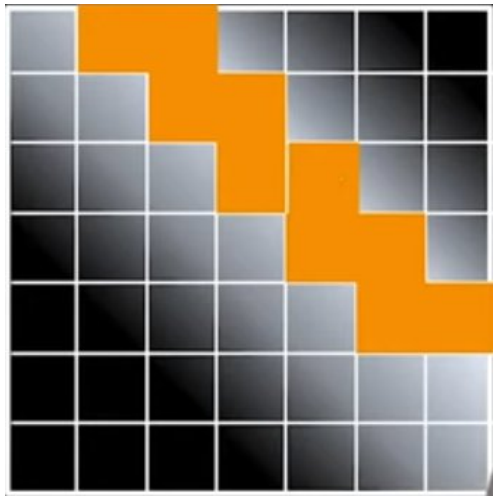
- Smooth Image with 2D Gaussian: $n_\sigma * I$
- Compute Image Gradient using Sobel Operator: $\nabla n_\sigma * I$
- Find Gradient Magnitude at each pixel: $\|\nabla n_\sigma * I\|$
- Find Gradient Orientation at each Pixel:

$$\hat{\boldsymbol{n}} = \frac{\nabla n_\sigma * I}{\|\nabla n_\sigma * I\|}$$

- Compute Laplacian along the Gradient Direction $\hat{\boldsymbol{n}}$ at each pixel

$$\frac{\partial^2 (n_\sigma * I)}{\partial \hat{\boldsymbol{n}}^2}$$

- Find Zero Crossings in Laplacian to find the edge location



$\|\nabla n_\sigma * I\|$

Canny Edge Detector Results: Lena

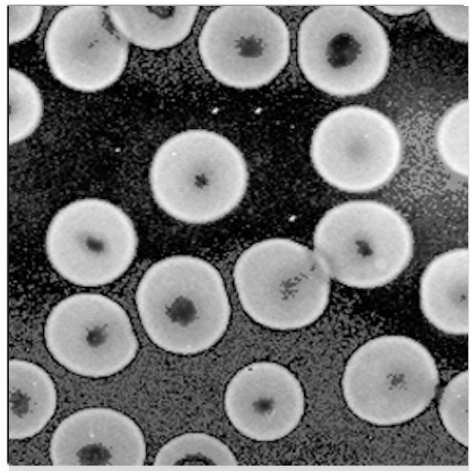
Image

$\sigma = 1$

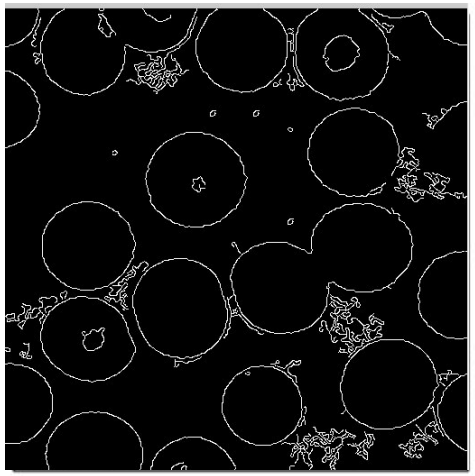
$\sigma = 2$

$\sigma = 4$

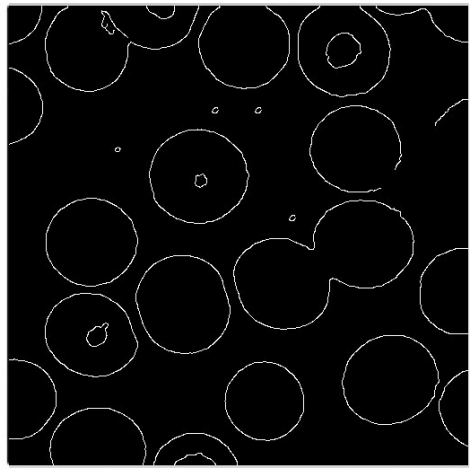
Canny Edge Detector Results: Cells



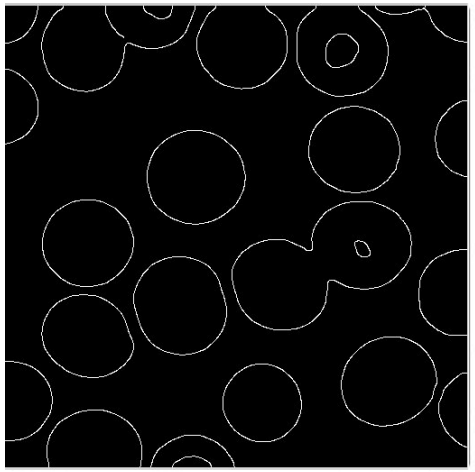
Image



$\sigma = 1$



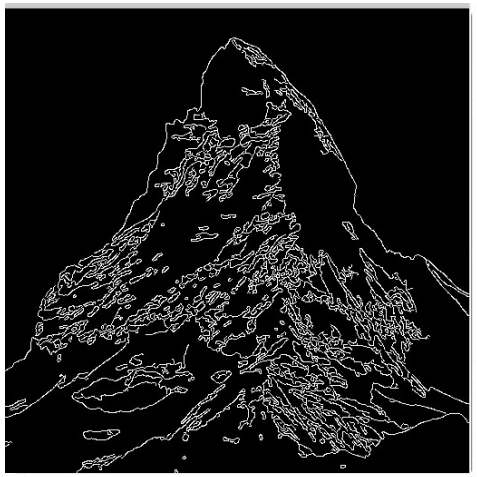
$\sigma = 2$



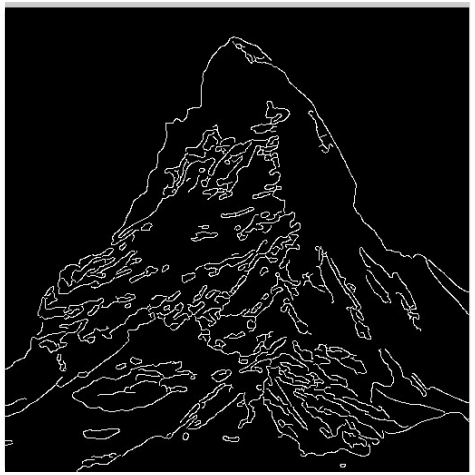
$\sigma = 4$

Canny Edge Detector Results: Mont Cervin

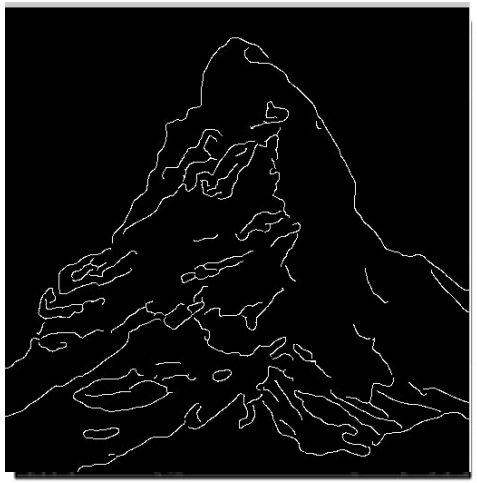
Image



$\sigma = 1$

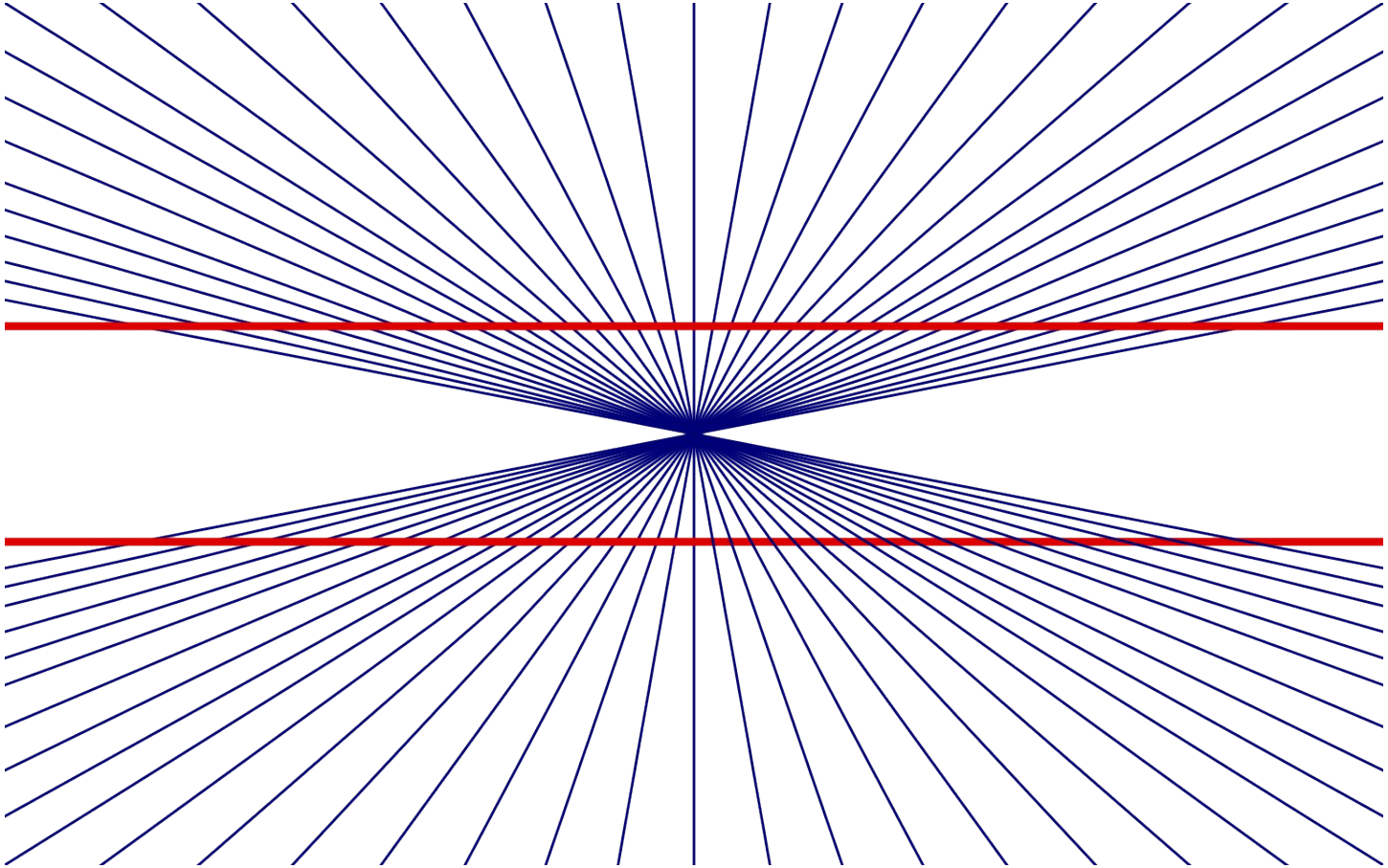


$\sigma = 2$



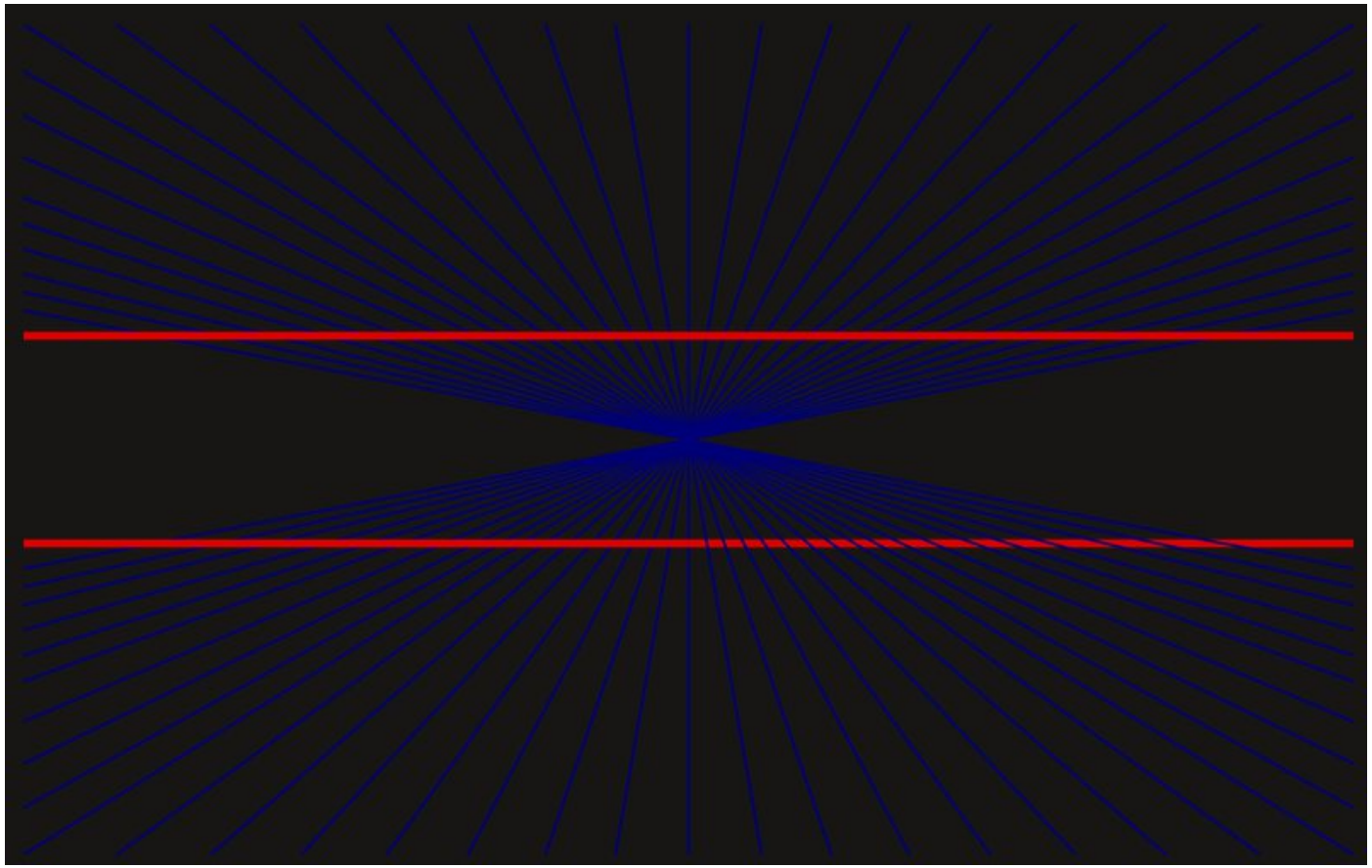
$\sigma = 4$

Edge Illusions: Hering Illusion



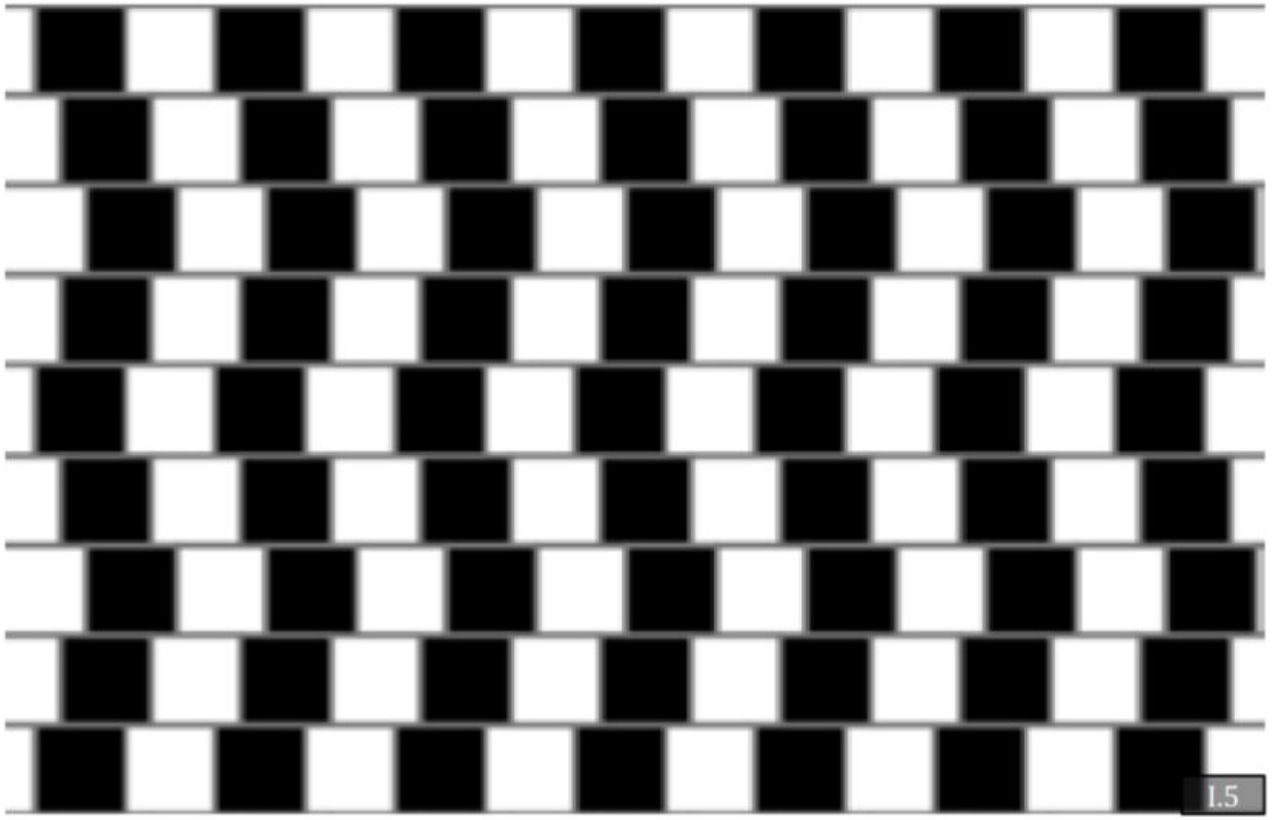
Ewald Hering, 1861

Edge Illusions: Hering Illusion



Ewald Hering, 1861

Edge Illusions: Café Wall Illusion

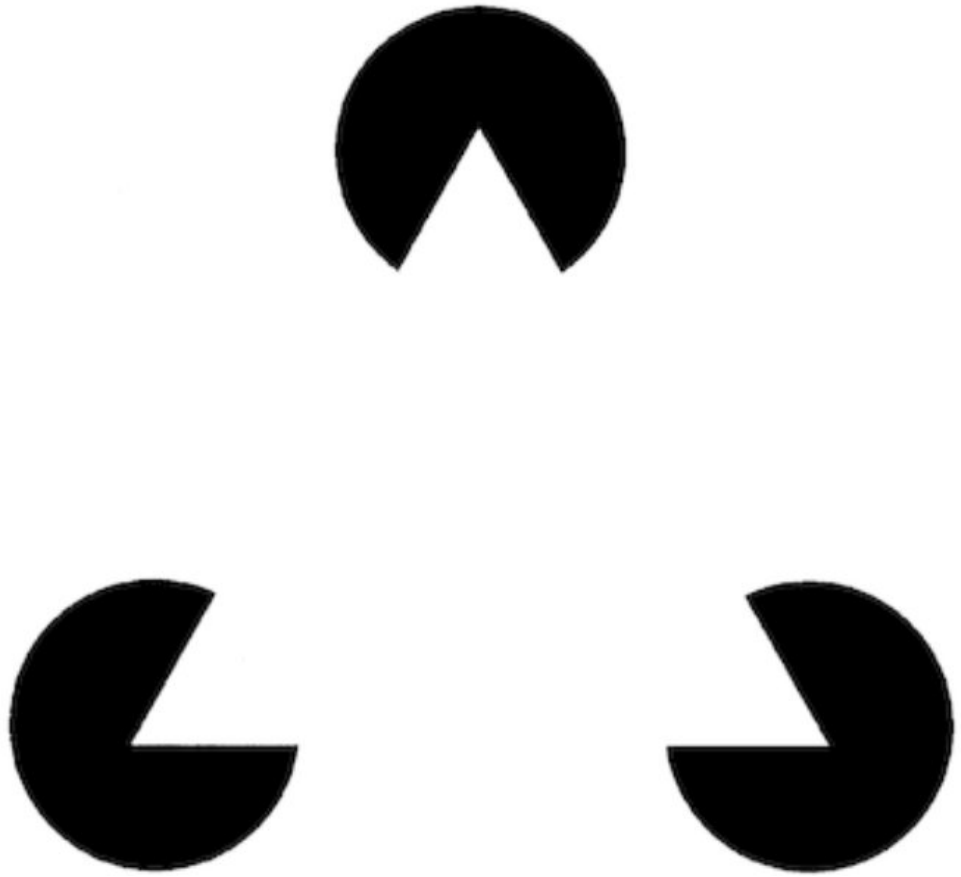


Gregory and Heard, 1979

Edge Illusions: Café Wall Illusion

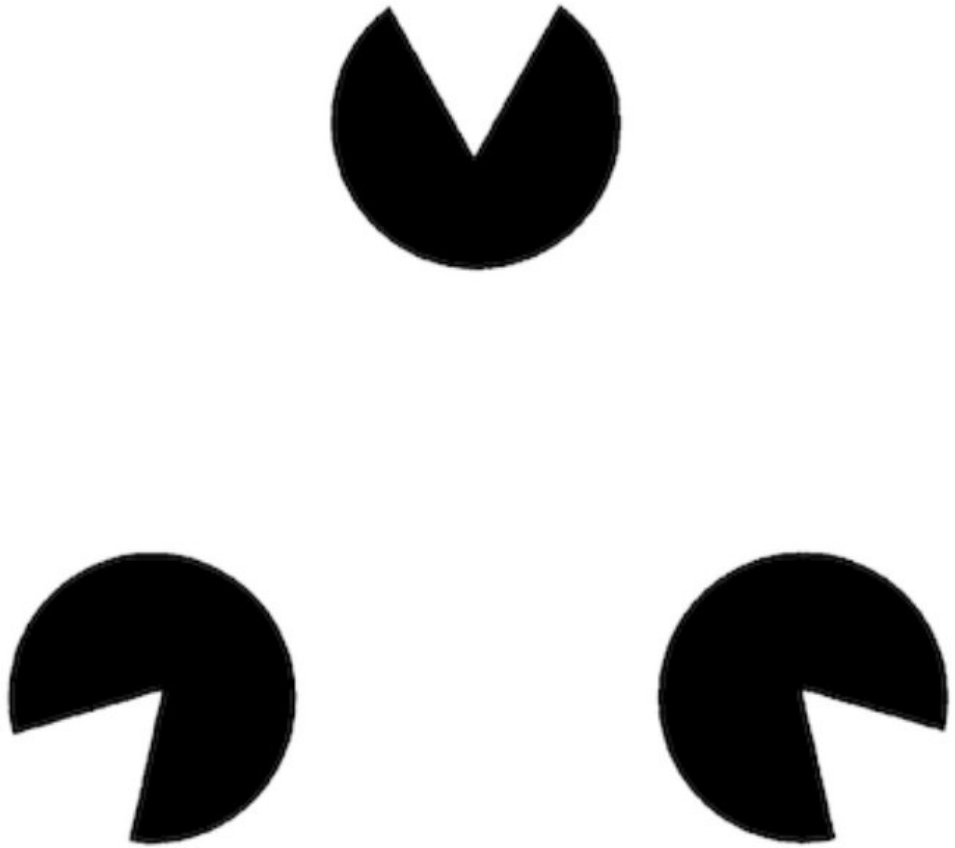
Gregory and Heard, 1979

Edge Illusions: The Kanizsa Triangle



Kanizsa, 1955

Edge Illusions: The Kanizsa Triangle



Kanizsa, 1955

Edge Illusions vs Edge Detectors

Illusion	What Fools Human Vision	What Edge Detectors Do
Hering	Acute angle expansion distorts straight lines	Correctly detect straight lines (gradient-based)
Café Wall	Context-dependent border locking warps gray lines	Detect gray lines as perfectly horizontal
Kanizsa Triangle	Top-down completion creates edges from nothing	Detect **no edge** — no gradient to compute