

COL7655

Foundations of Visual Computing

Subodh Kumar

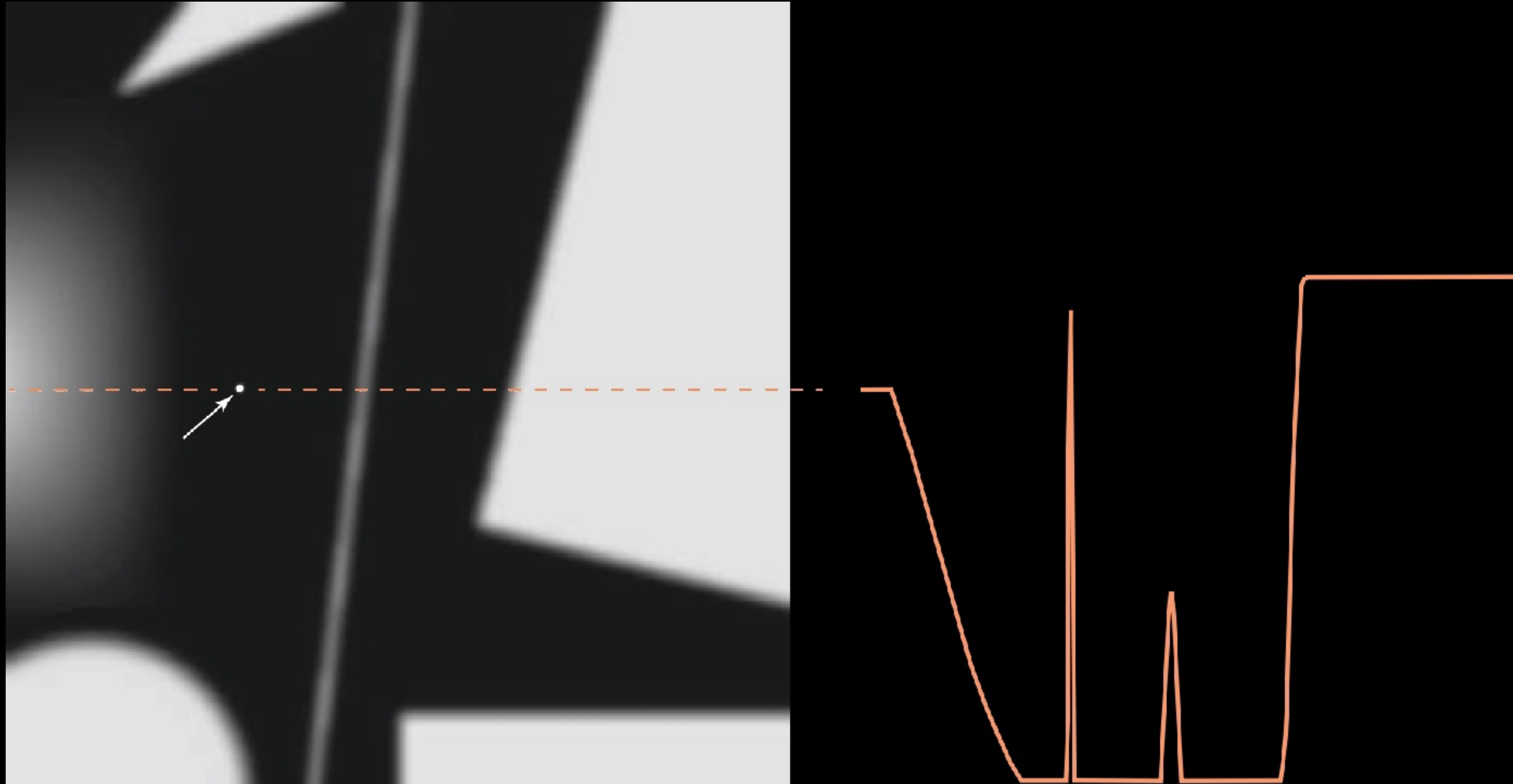
subodh@cse.iitd.ac.in

COL7655

Image Features

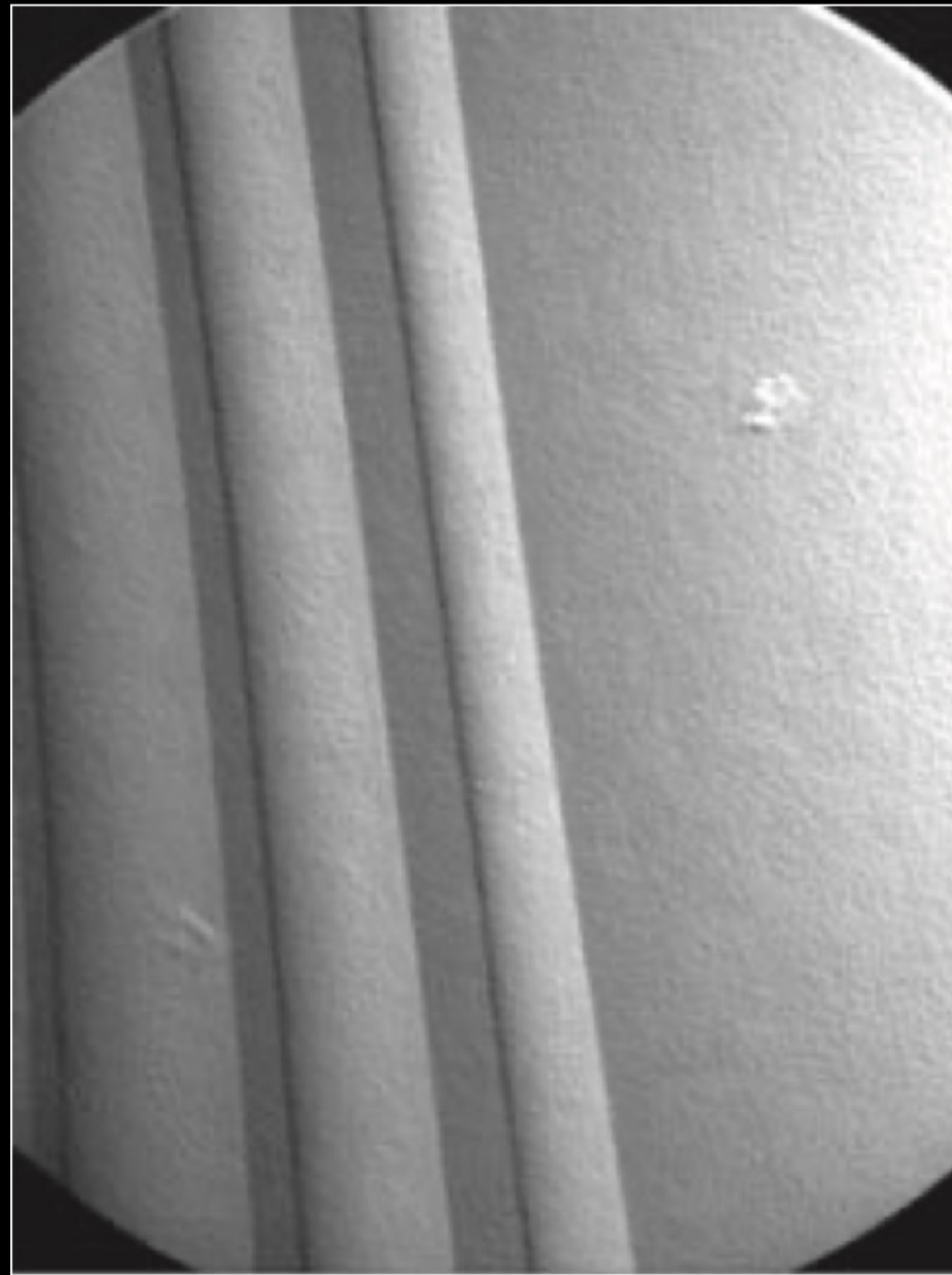
Thanks to Rahul Narain for help with slides

Feature: Corner, Edge

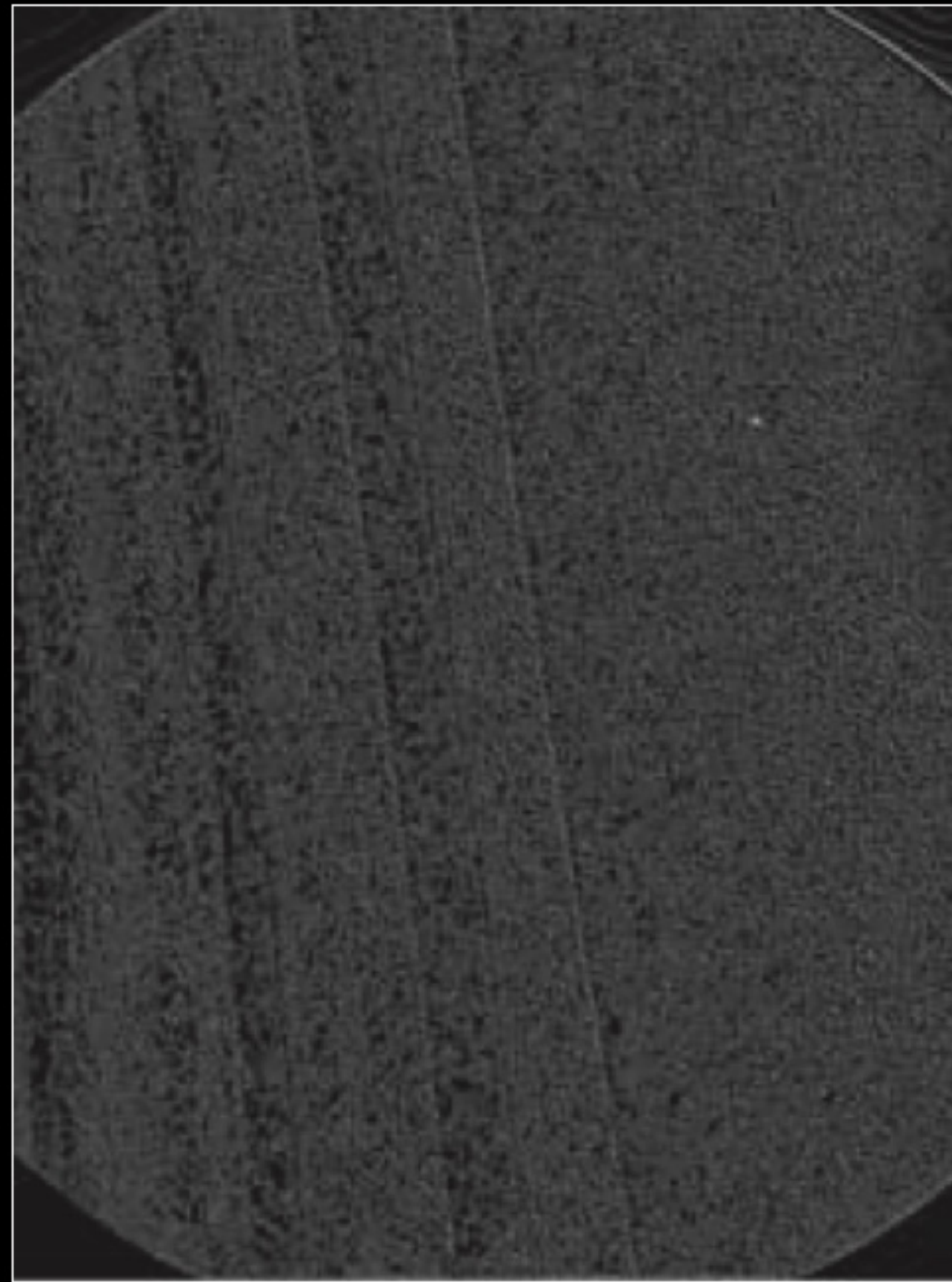


Detecting Isolated Points

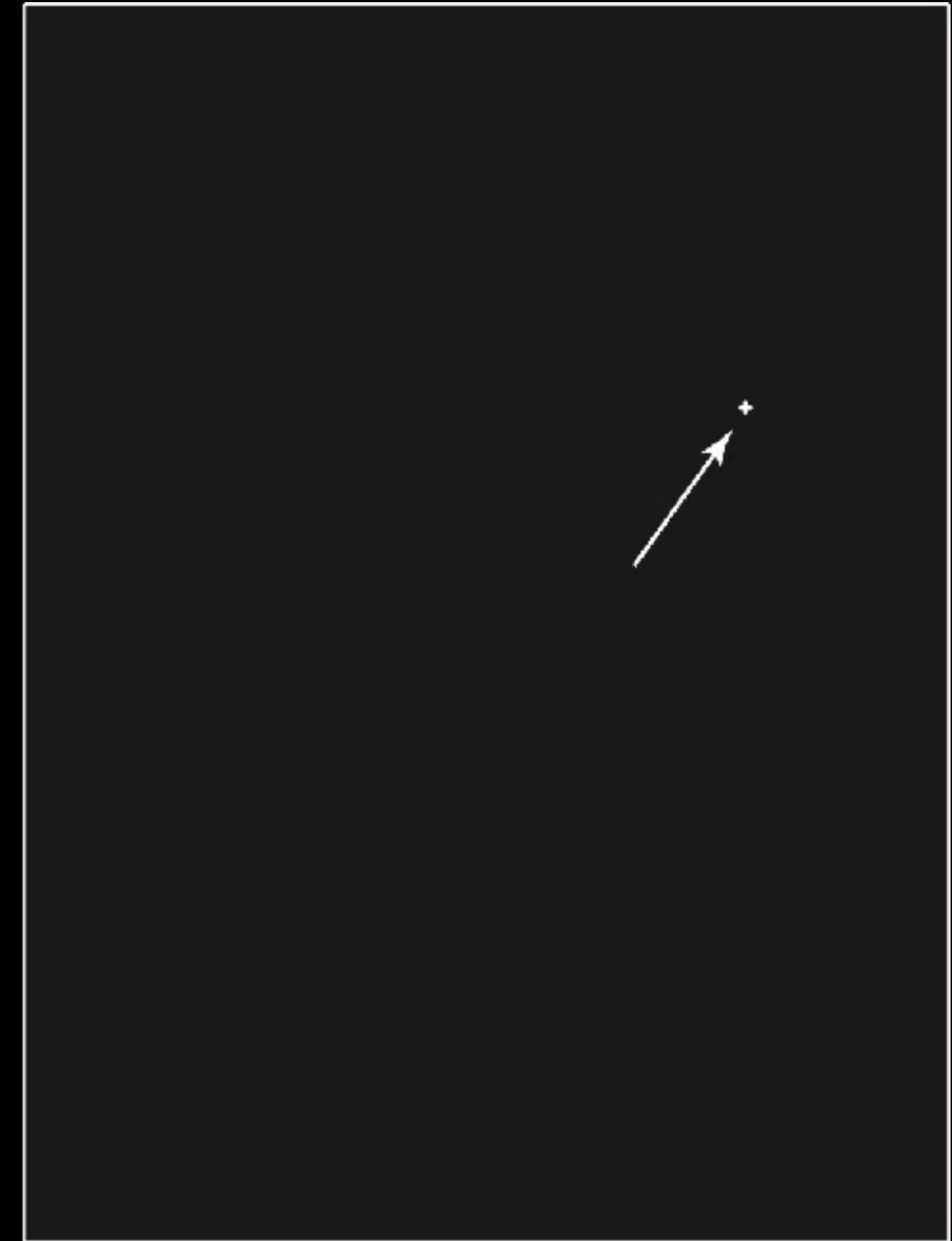
Using Laplacian Operator



Original



$|\nabla^2 f|$



Thresholded

Detecting Lines

Using Laplacian Operator

-1	-1	-1	2	-1	-1
2	2	2	-1	2	-1
-1	-1	-1	-1	-1	2

Horizontal

+45°

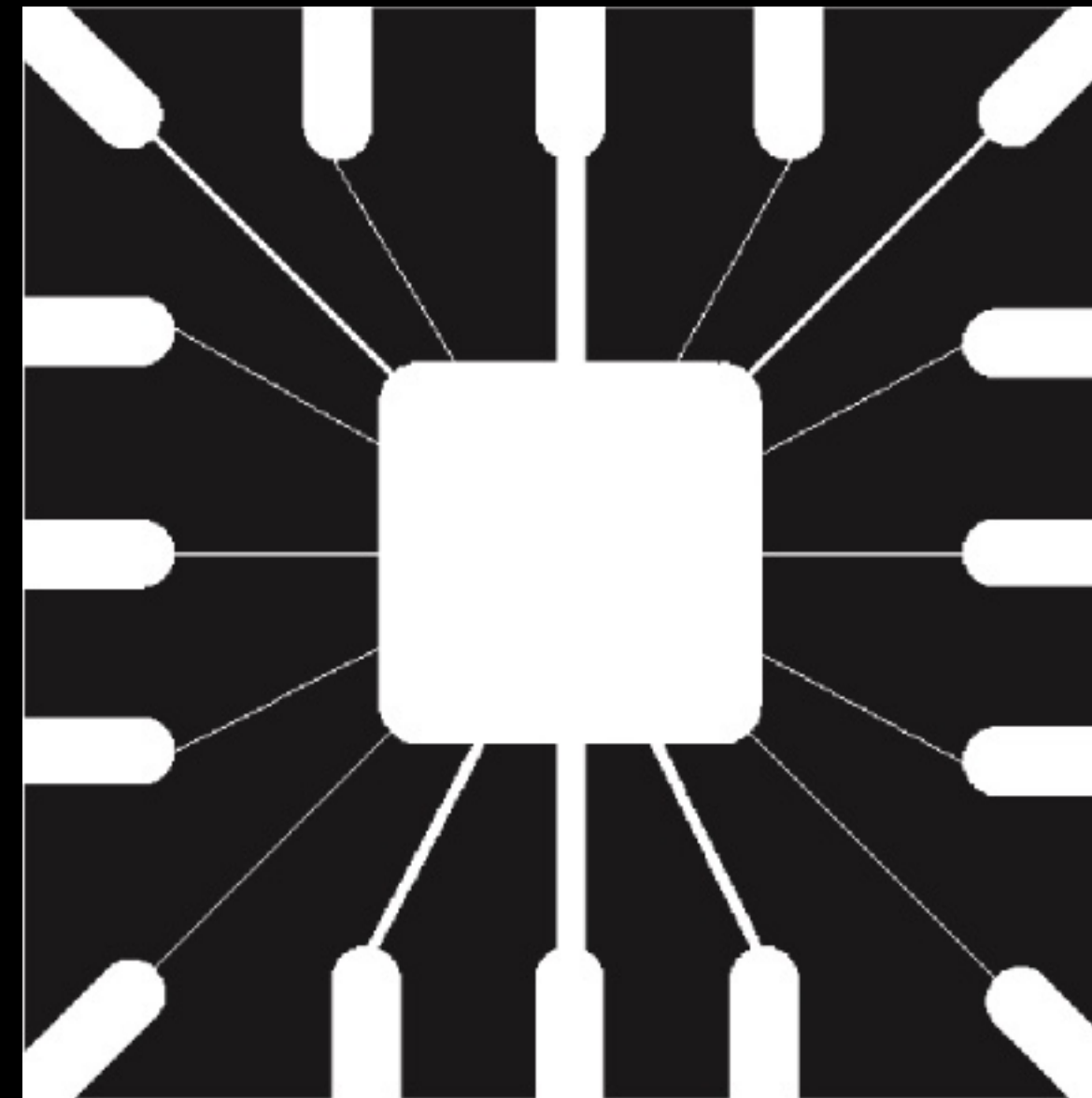
-1	2	-1	-1	-1	2
-1	2	-1	-1	2	-1
-1	2	-1	2	-1	-1

Vertical

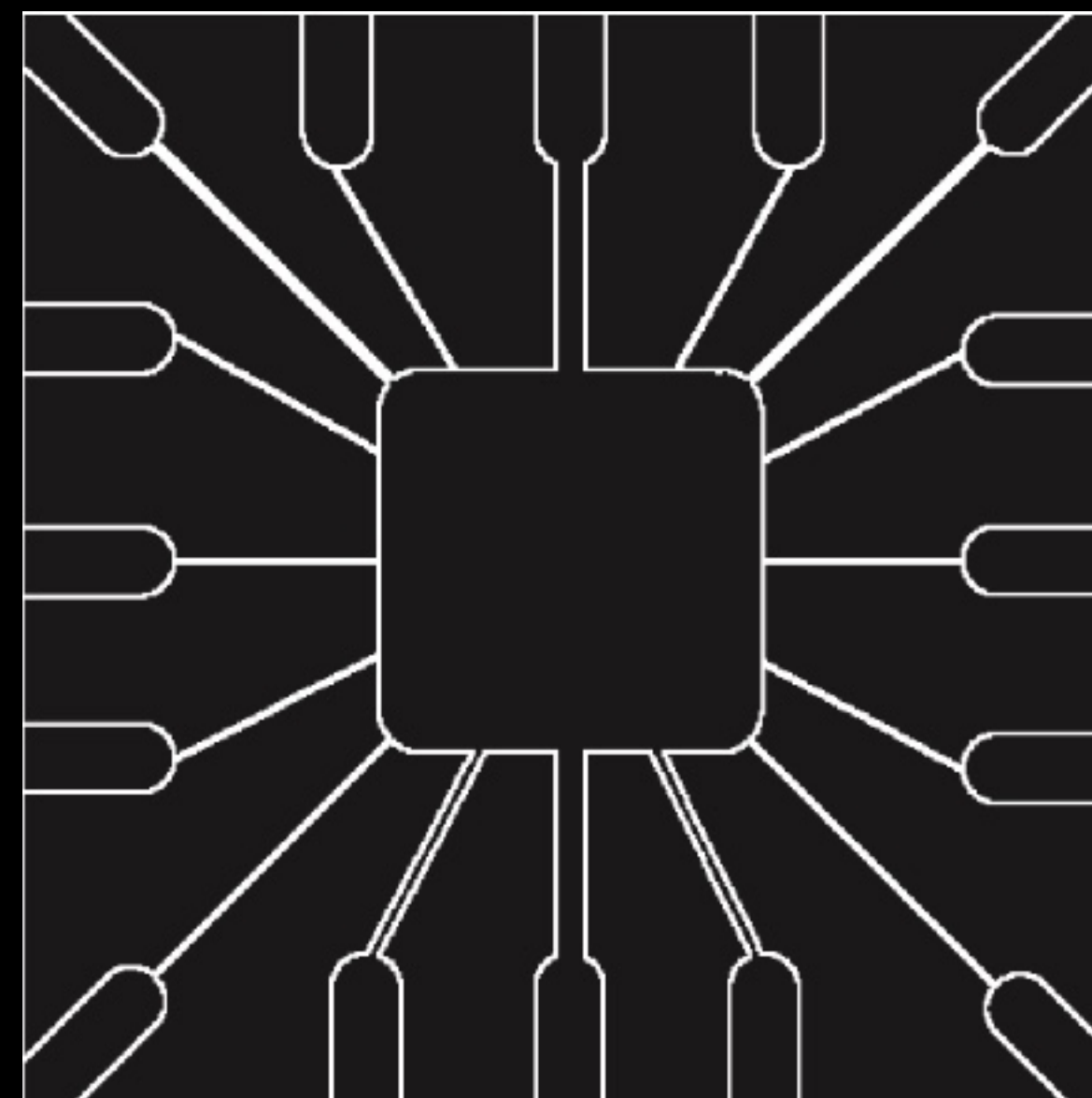
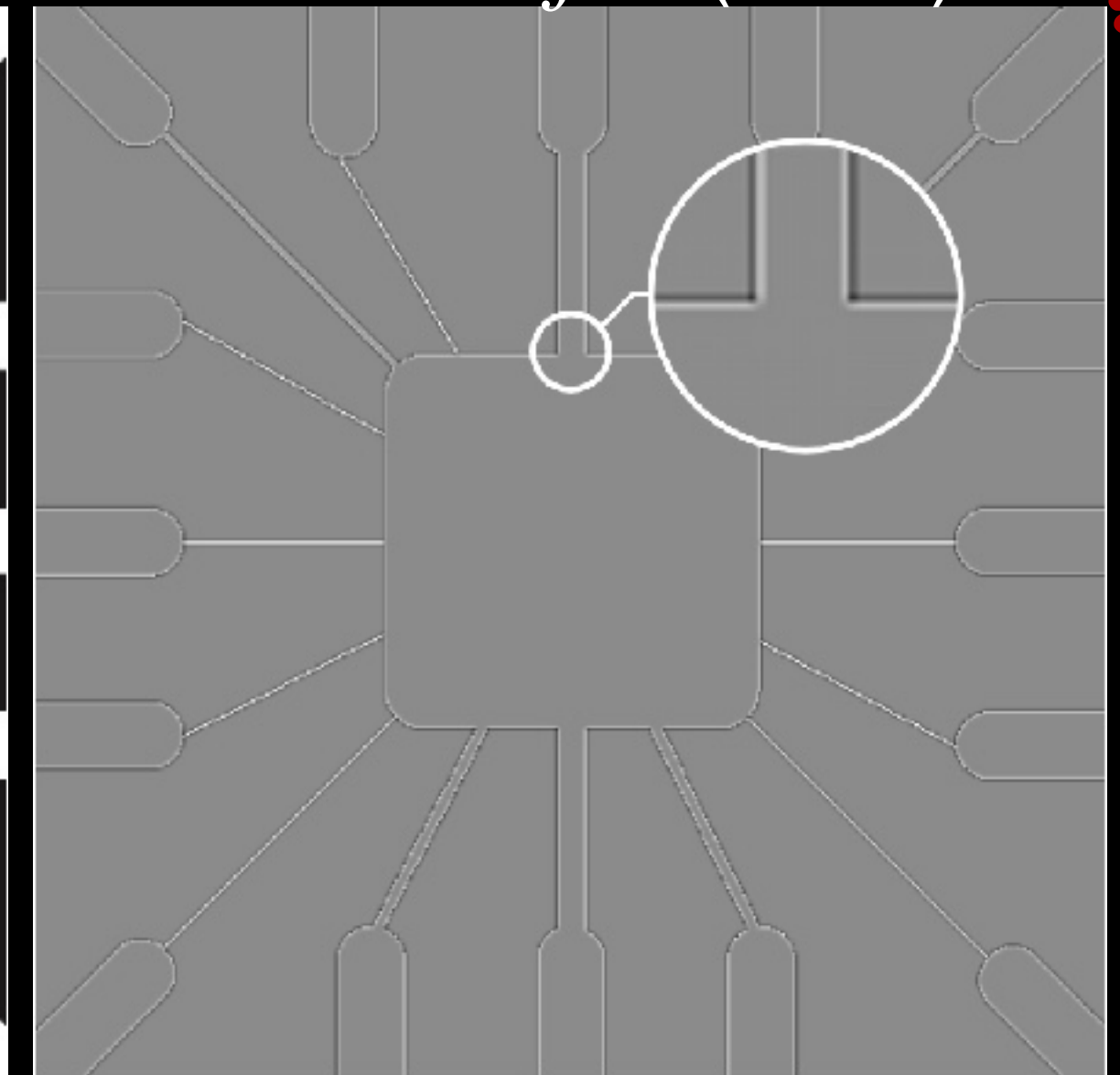
-45°

Directional Detectors

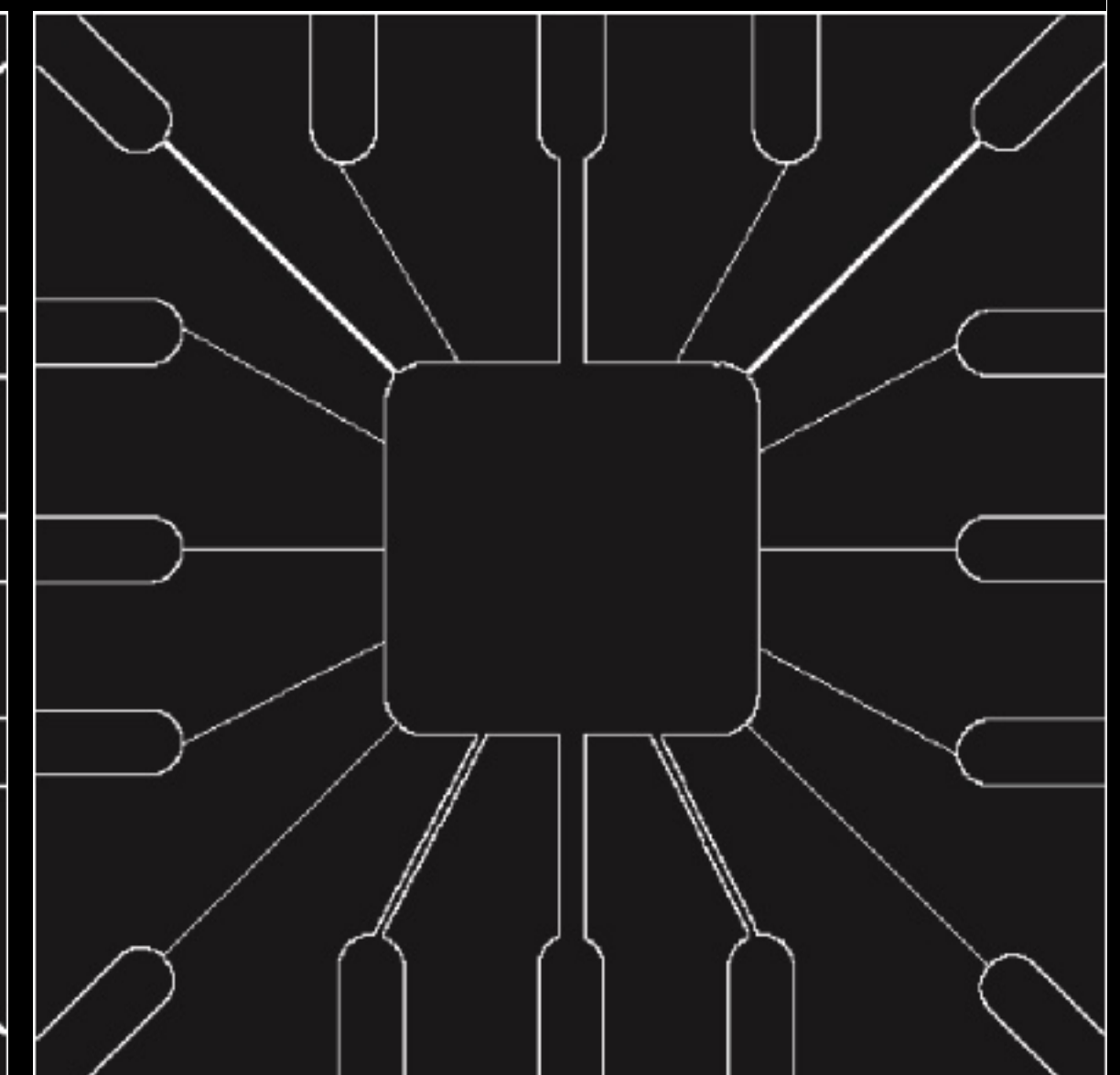
Original



$\nabla^2 f$ (+128)



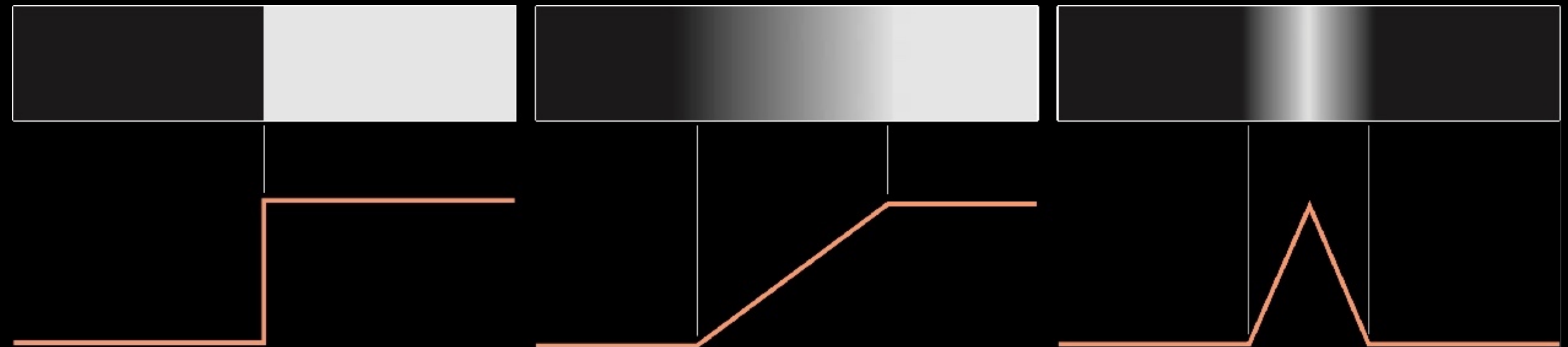
$|\nabla^2 f|$



$\max(\nabla^2 f, 0)$

Edge Models

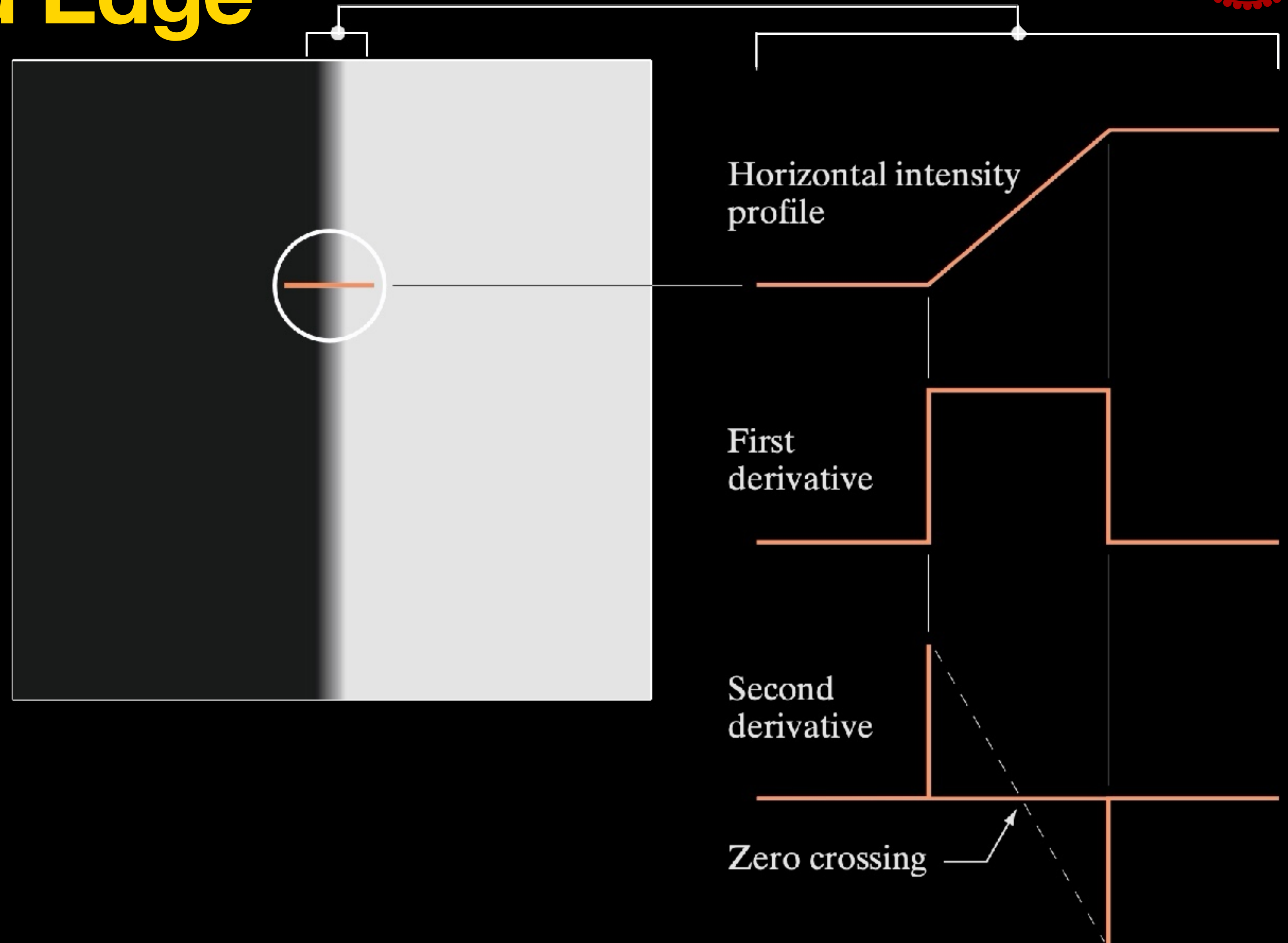
- **Step edge:** ideal case
- **Ramp edge:** more realistic (blurred step)
- **Roof edge:** actually a blurred line, or two ramp edges



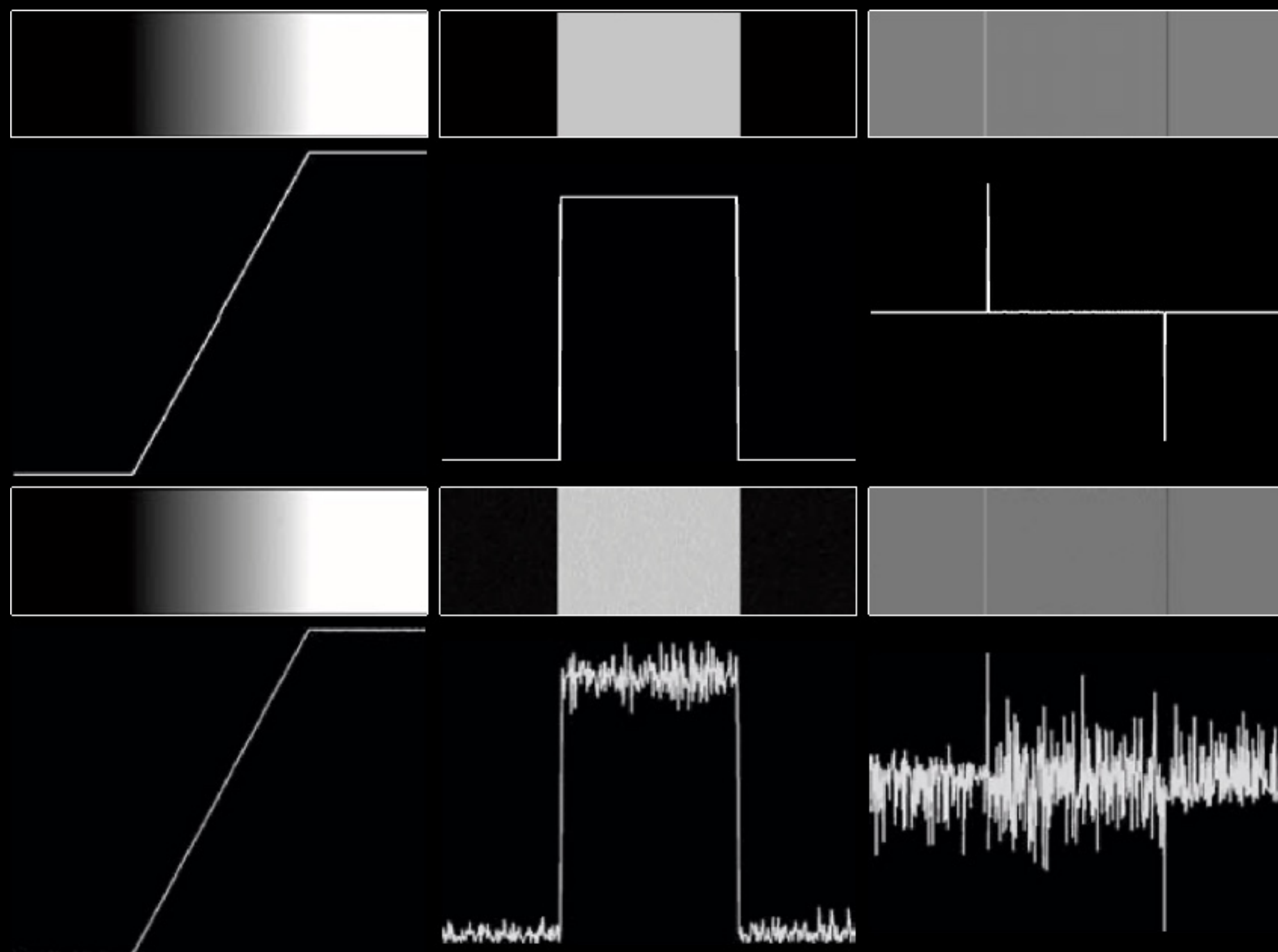


Derivatives and Edge

- Magnitude of 1st deriv
→ presence of edge
- Sign of 2nd deriv is
opposite on different
sides of edge
- Direction of gradient
vector $\perp$ edge direction
- Directional derivative of c
in direction $\mathbf{d} = \nabla c \cdot \mathbf{d}$

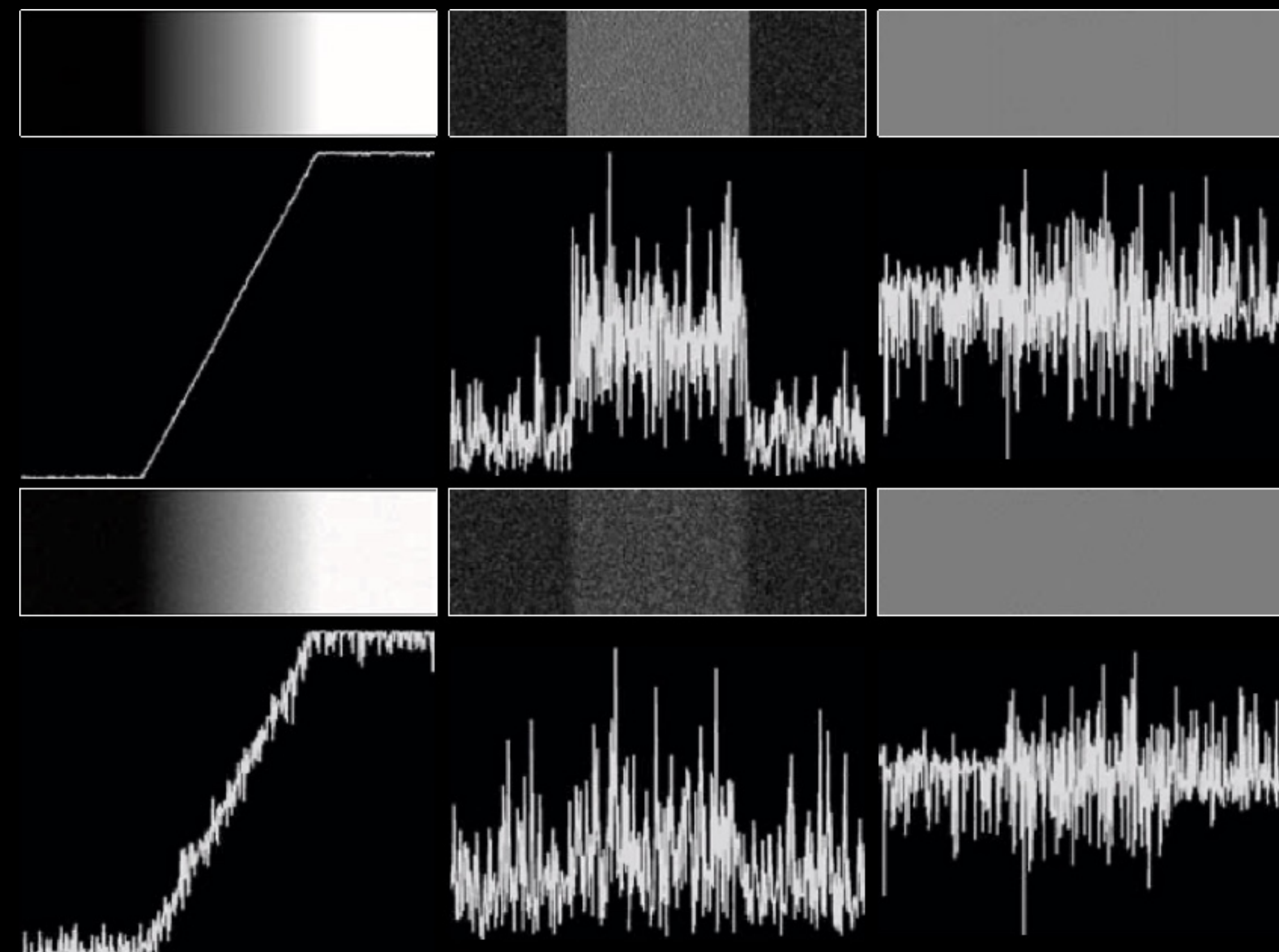


No noise



$\sigma = 0.1$

$\sigma = 1$



$\sigma = 10$

Edge Detection using Sobel Filter

Original image



Sobel in x



Sobel in y



Gradient
magnitude



Edge Detection using Sobel Filter

After 5x5
mean filter



Sobel in x



Sobel in y

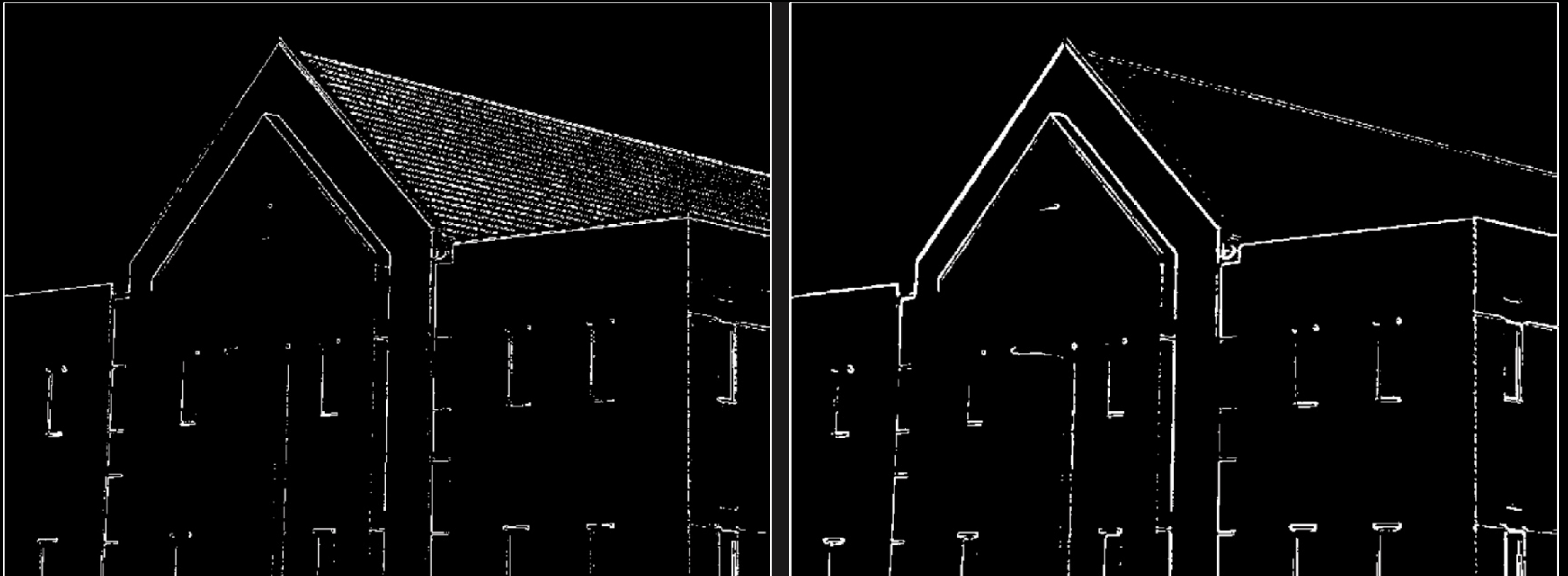


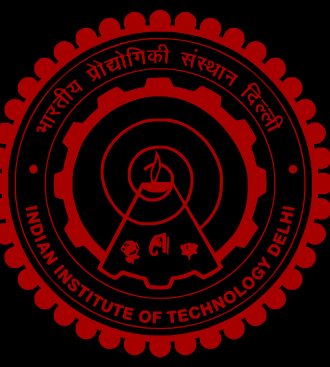
Gradient
magnitude



Edge Detection using Sobel Filter

- After thresholding at 33% of max intensity:





Marr-Hildreth Edge Detector

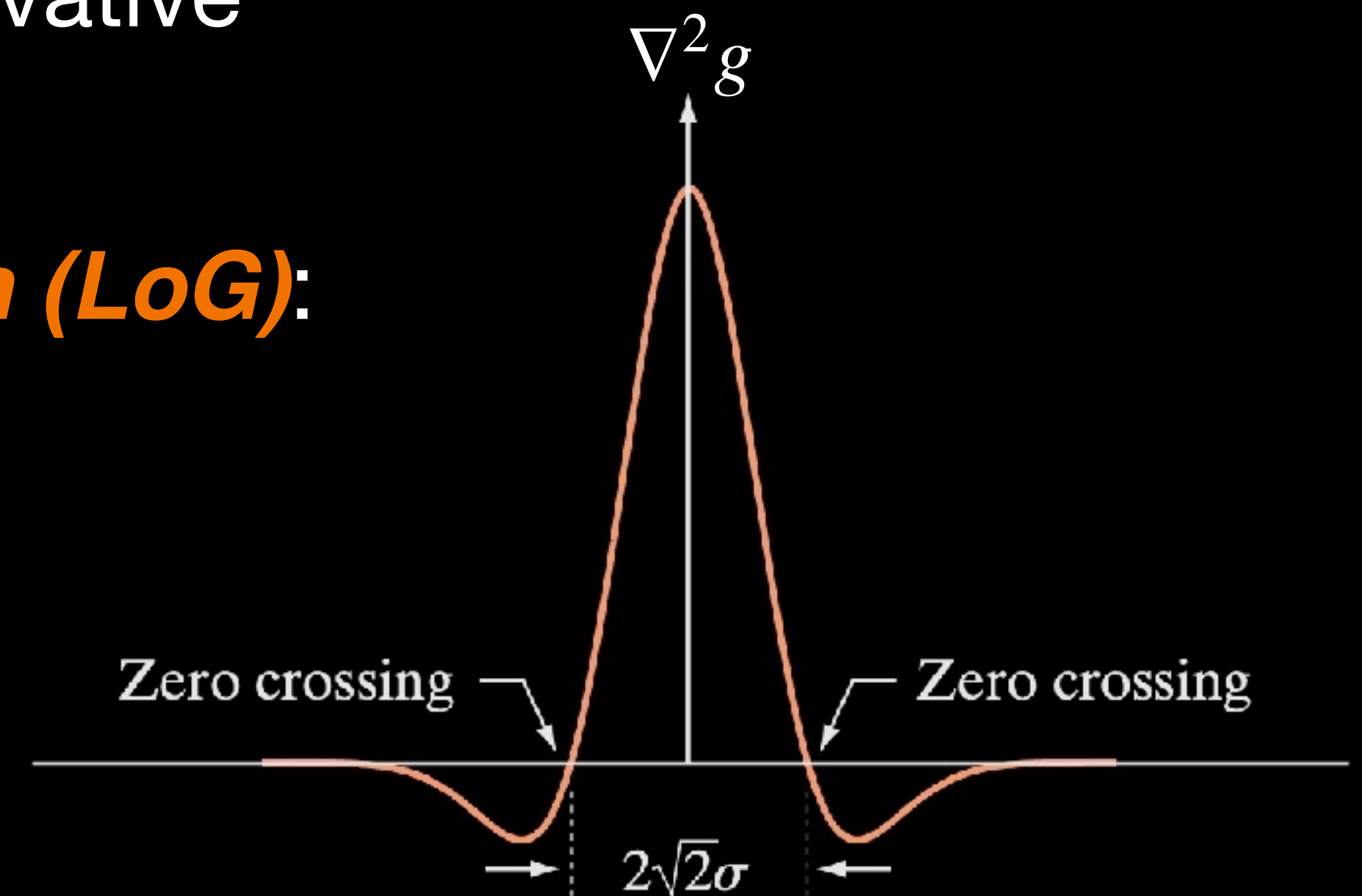
Key ideas:

1. Detect edges at a “tunable” scale
2. Localize edges at zero crossing of second derivative

Note: $(\nabla^2 G) * c = \nabla^2 (G * c)$

To do both, convolve with **Laplacian of Gaussian (LoG)**:

$$\nabla^2 G(x, y) = \left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right) \overbrace{\left(e^{-\frac{x^2 + y^2}{2\sigma^2}} \right)}^{\text{Standard Gaussian}}$$

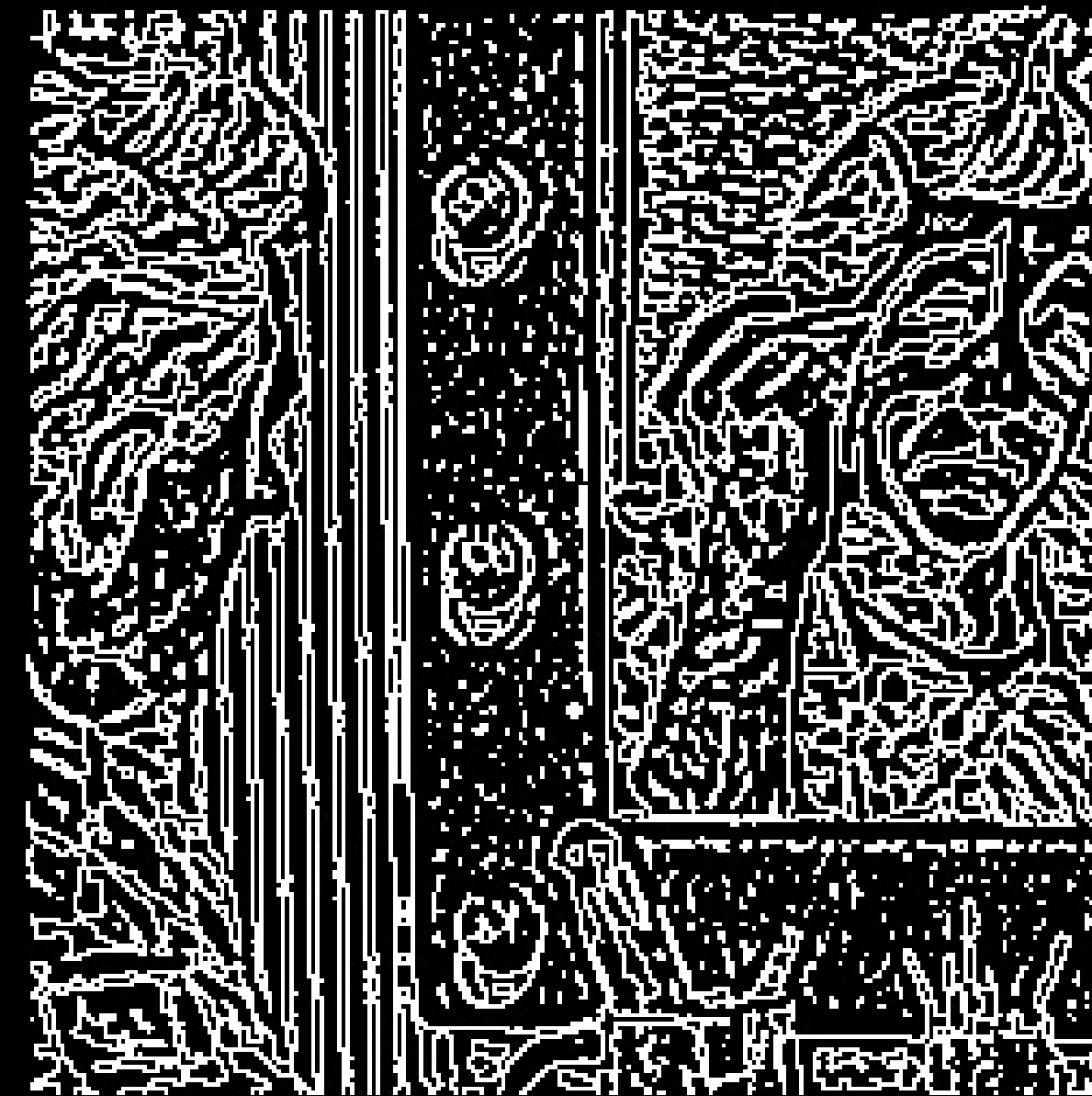




Original c



$$\hat{c} = c * \text{LoG}$$



→ All zero crossings



→ significant zero crossings

Next, detect *zero crossings*: pixels where $\text{sign}(\hat{c}(x+1, y)) \neq \text{sign}(\hat{c}(x-1, y))$, or similarly in other three pairs: vertically or diagonally

Keep only strong edges:

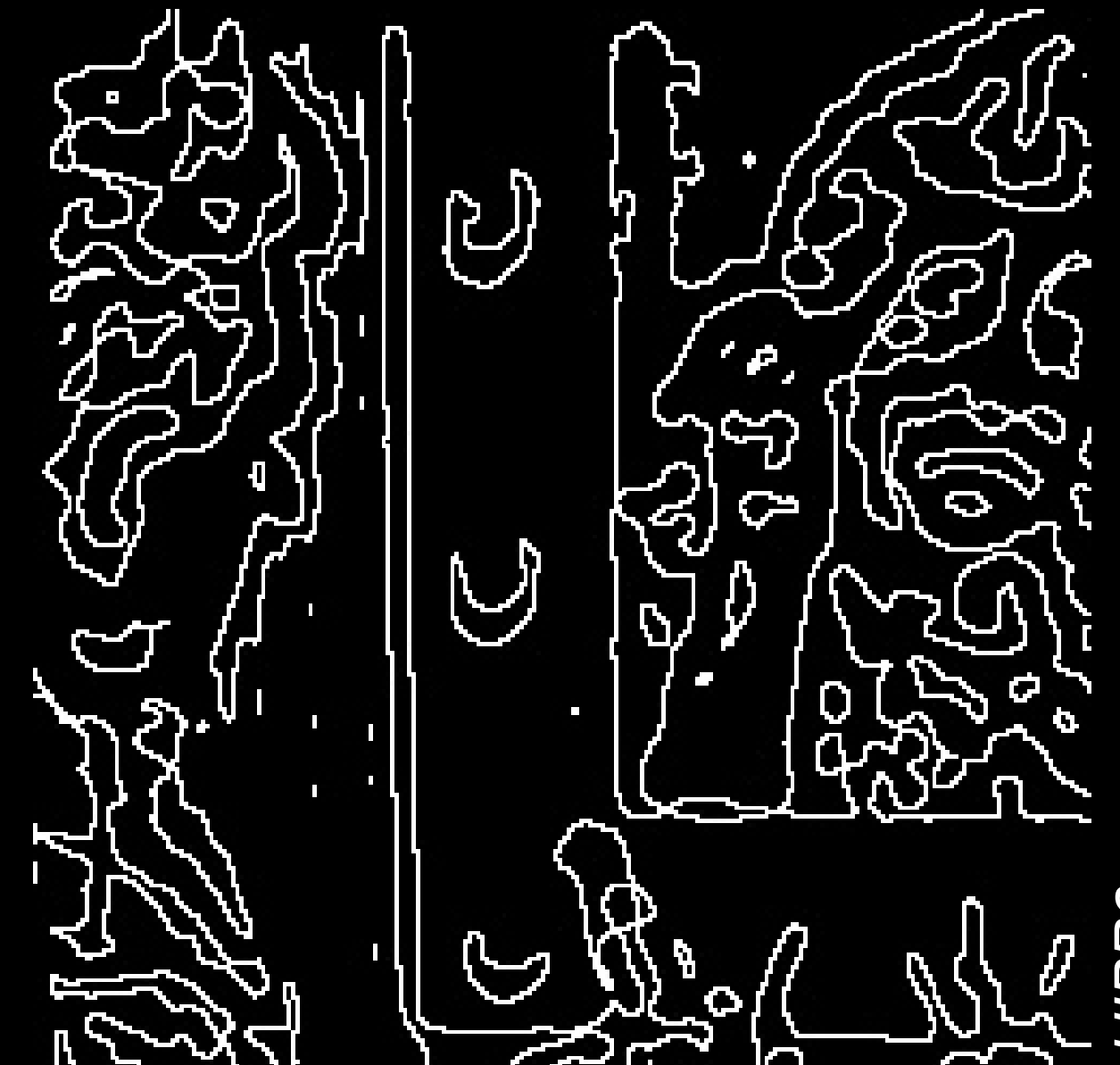
$$|\hat{c}(x+1, y) - \hat{c}(x-1, y)| > \text{some threshold}$$

(Check only in the direction of zero crossing)

Zero-crosses with a large difference; mitigates noise



Using larger σ :
(keeping all edges)





Canny Edge Detection

1. Applying smoothing filter to blur out noise
2. Compute Intensity Gradient: Magnitude and Direction
3. Suppress local Non-Maxima
4. Threshold in strong (sure edge) and threshold out non edges
 - Retaining weak edges
5. Hysteresis edge tracking
 - If a weak edge pixel has a strong edge neighbor, mark it as edge
 - If a weak edge pixel is isolated from any strong edges, do not mark as edge

Canny Edge Detector

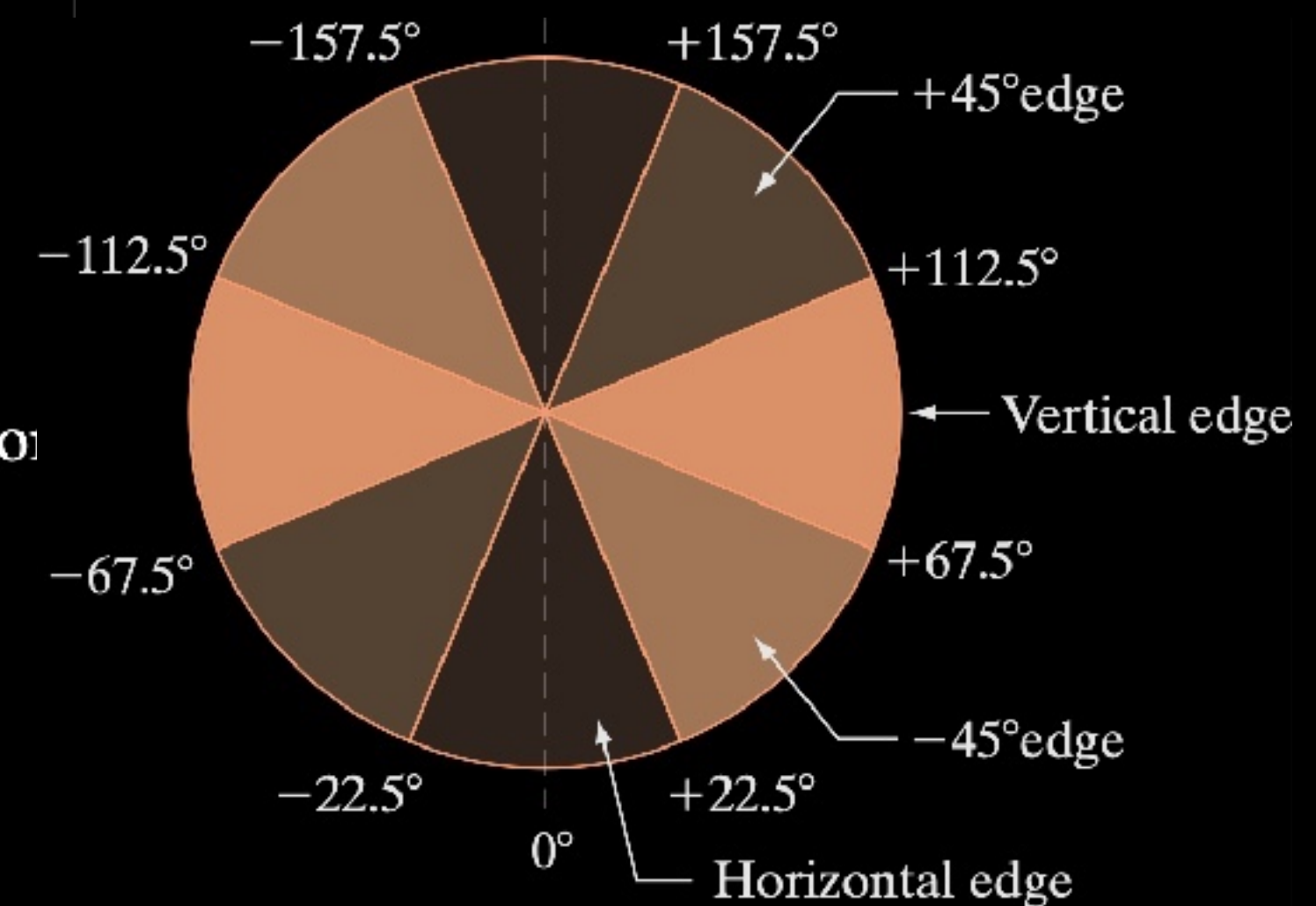
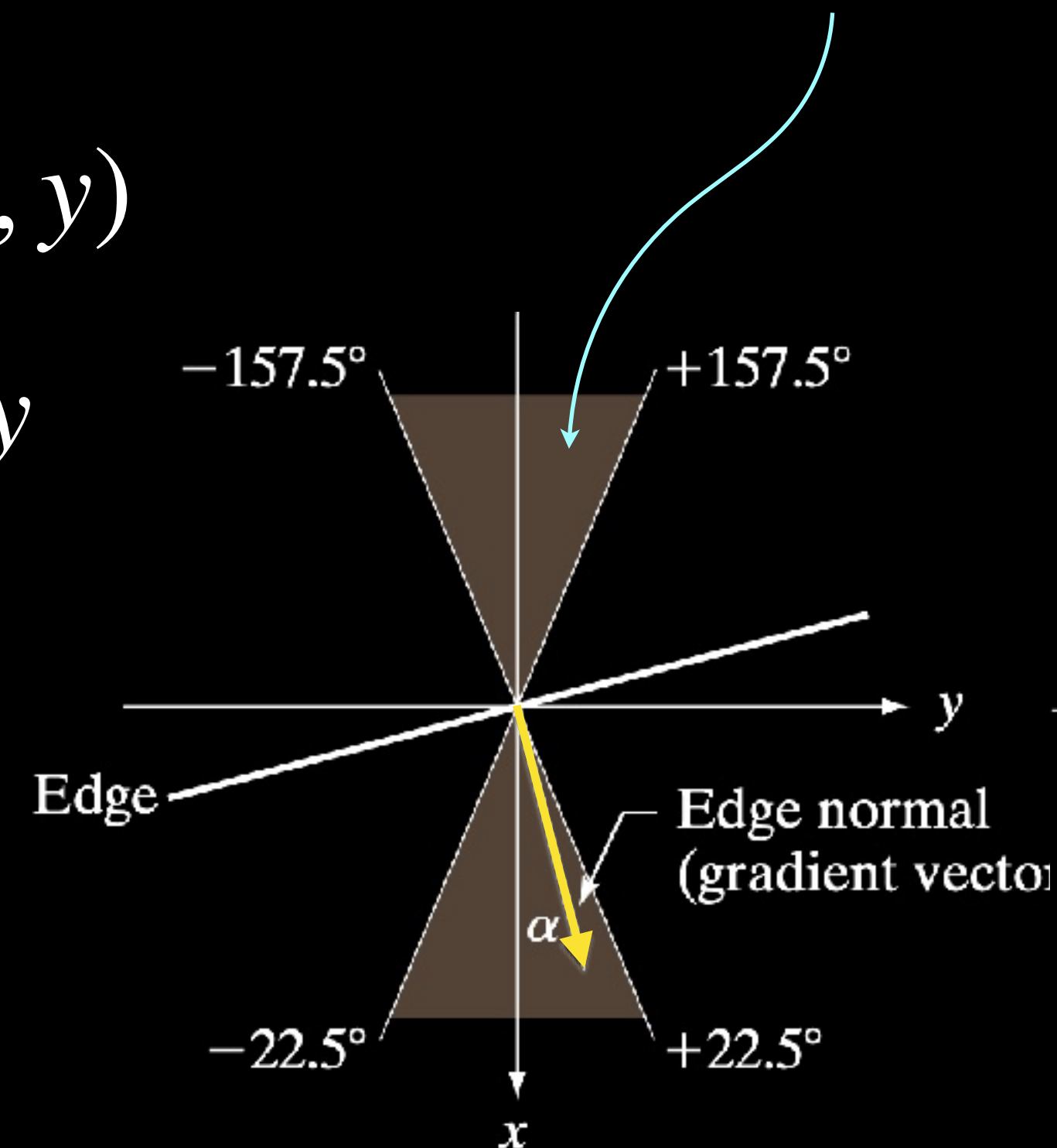
1. Compute $\hat{c}(x, y) = G(x, y) * c(x, y)$

2. Compute $\nabla \hat{c}_x(x, y)$ and $\nabla \hat{c}_y(x, y)$

3. **Non-maxima suppression:**

$\forall x, y$, check along its edge gradient.
If either neighbor has a larger $\|\nabla \hat{c}\|$,
set $\|\nabla \hat{c}(x, y)\| = 0$

Gradient Vector lies in this 'vertical' double-cone

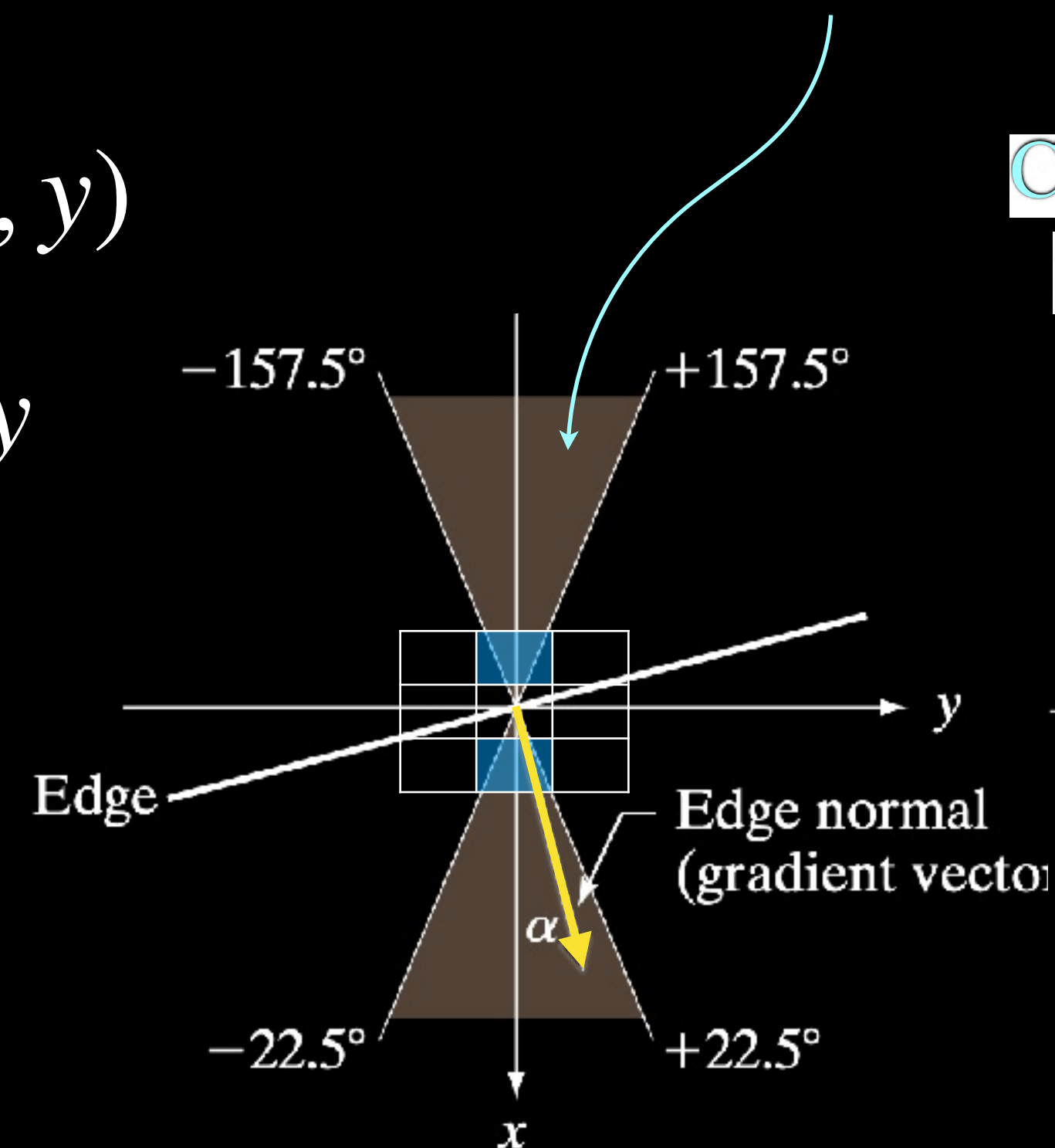


Four quantized double-cones

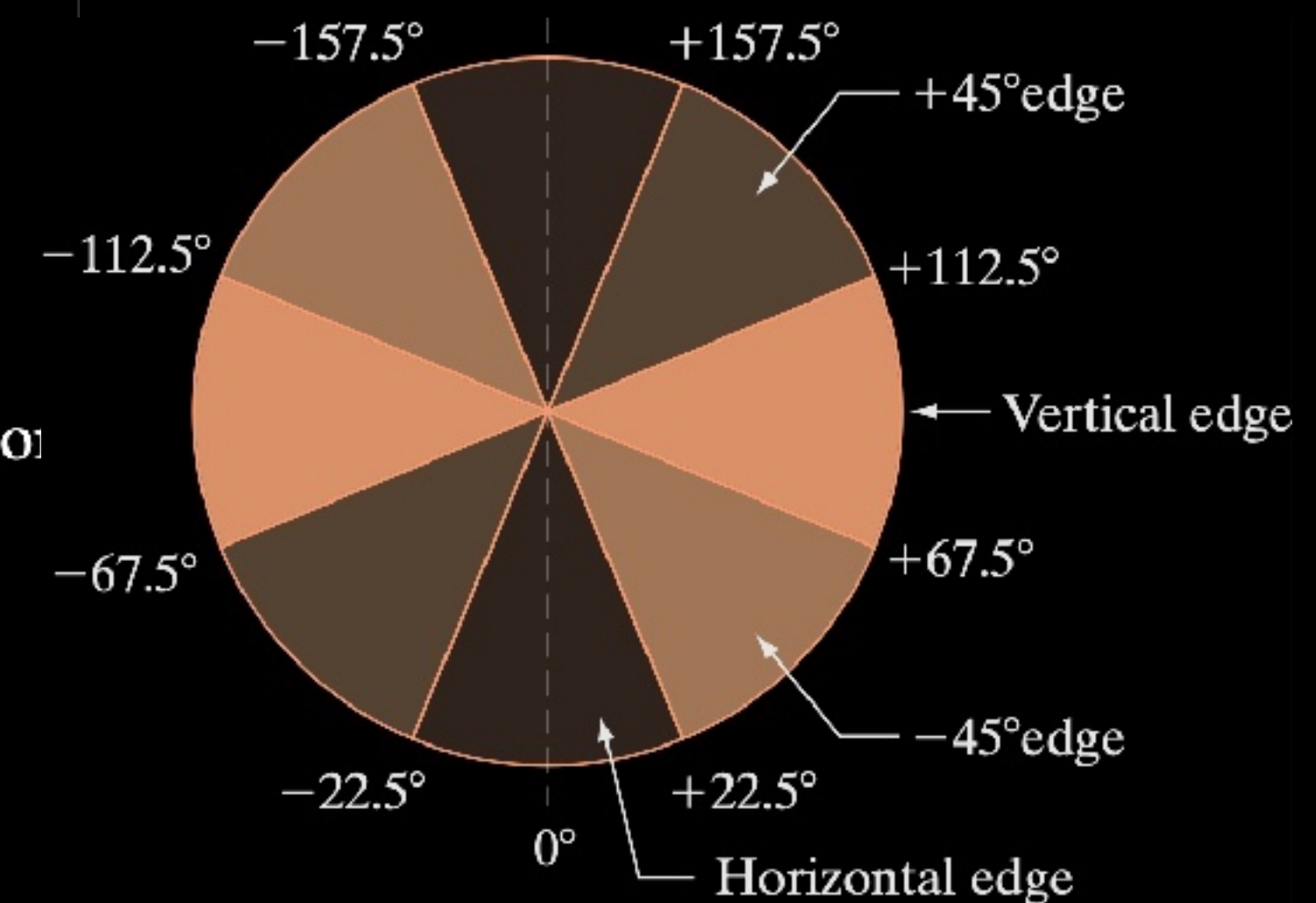
Canny Edge Detector

1. Compute $\hat{c}(x, y) = G(x, y) * c(x, y)$
2. Compute $\nabla \hat{c}_x(x, y)$ and $\nabla \hat{c}_y(x, y)$
3. **Non-maxima suppression:**
 $\forall x, y$, check along its edge gradient.
 If either neighbor has a larger $\|\nabla \hat{c}\|$,
 set $\|\nabla \hat{c}(x, y)\| = 0$

Gradient Vector lies in this 'vertical' double-cone

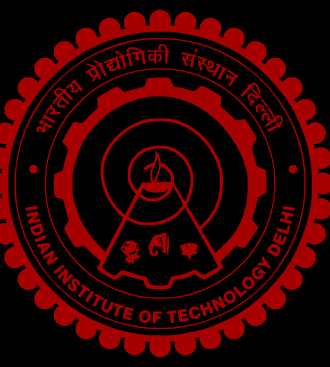


Check corresponding neighbor strengths



Four quantized double-cones

Canny Edge Detector

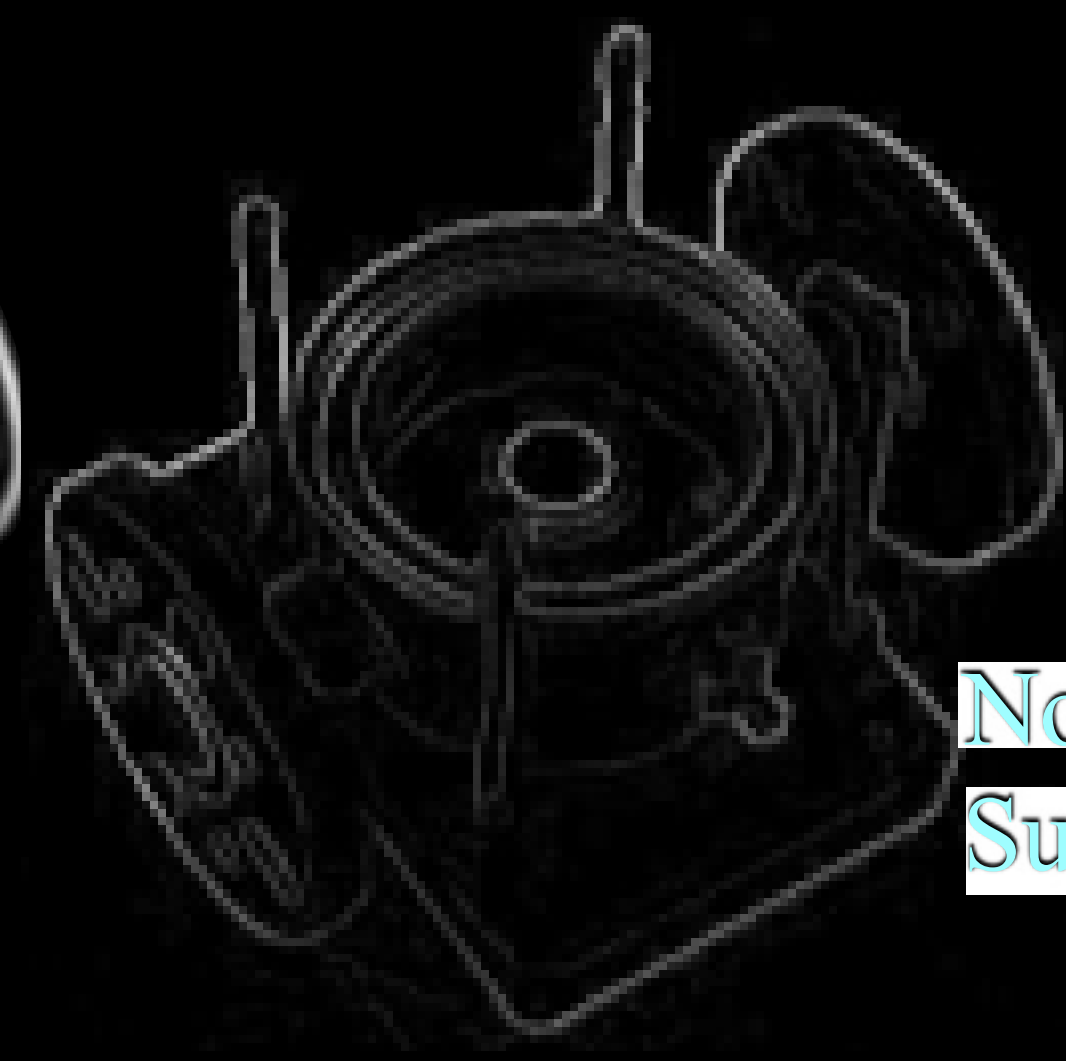


4. Double hysteresis thresholding

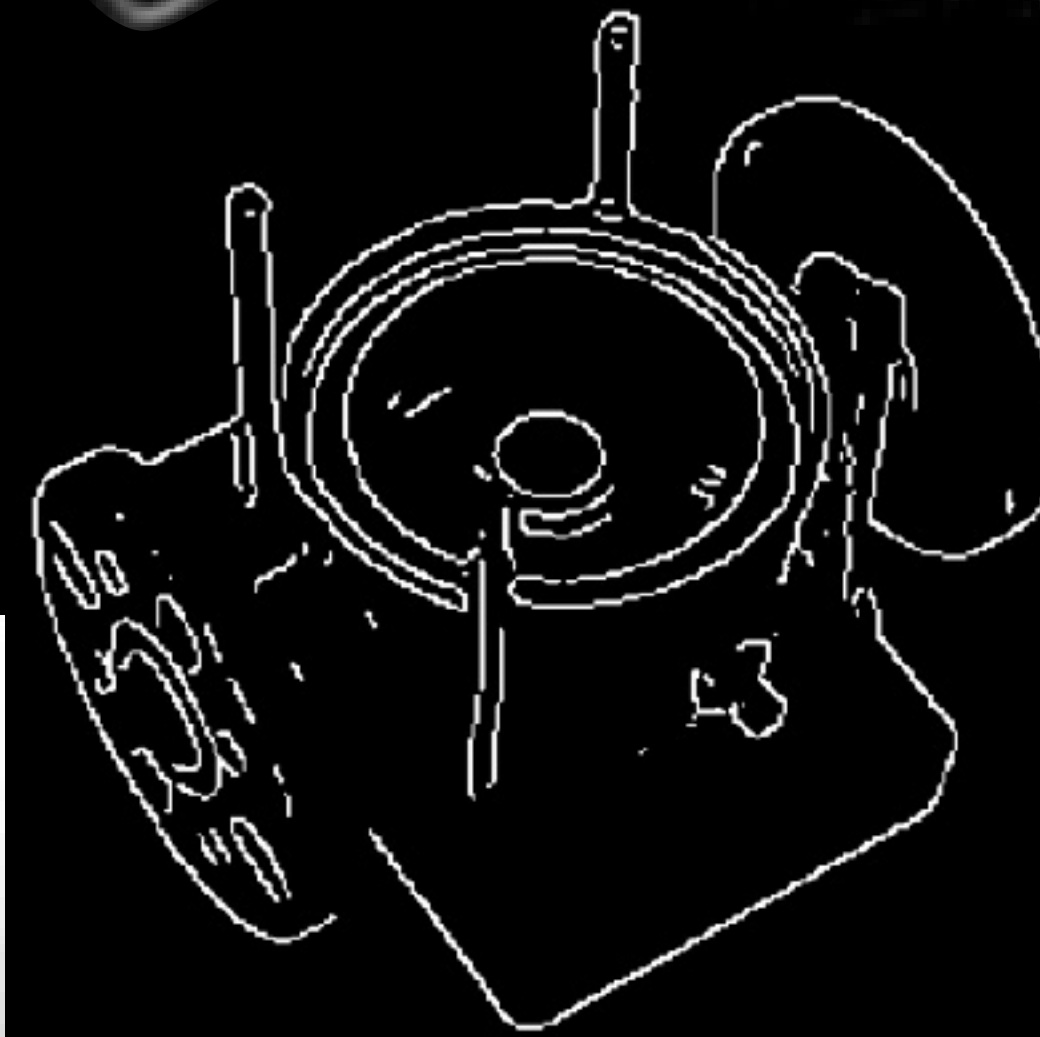
- $>$ High threshold $\Rightarrow$ strong edge pixels
- $<$ Low threshold $\Rightarrow$ weak edge pixels



Gradients



Non-maximal
Suppression

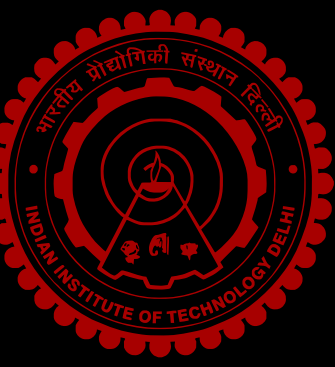


Strong Edges
(Thresholded)



Input Image

Canny Edge Detector



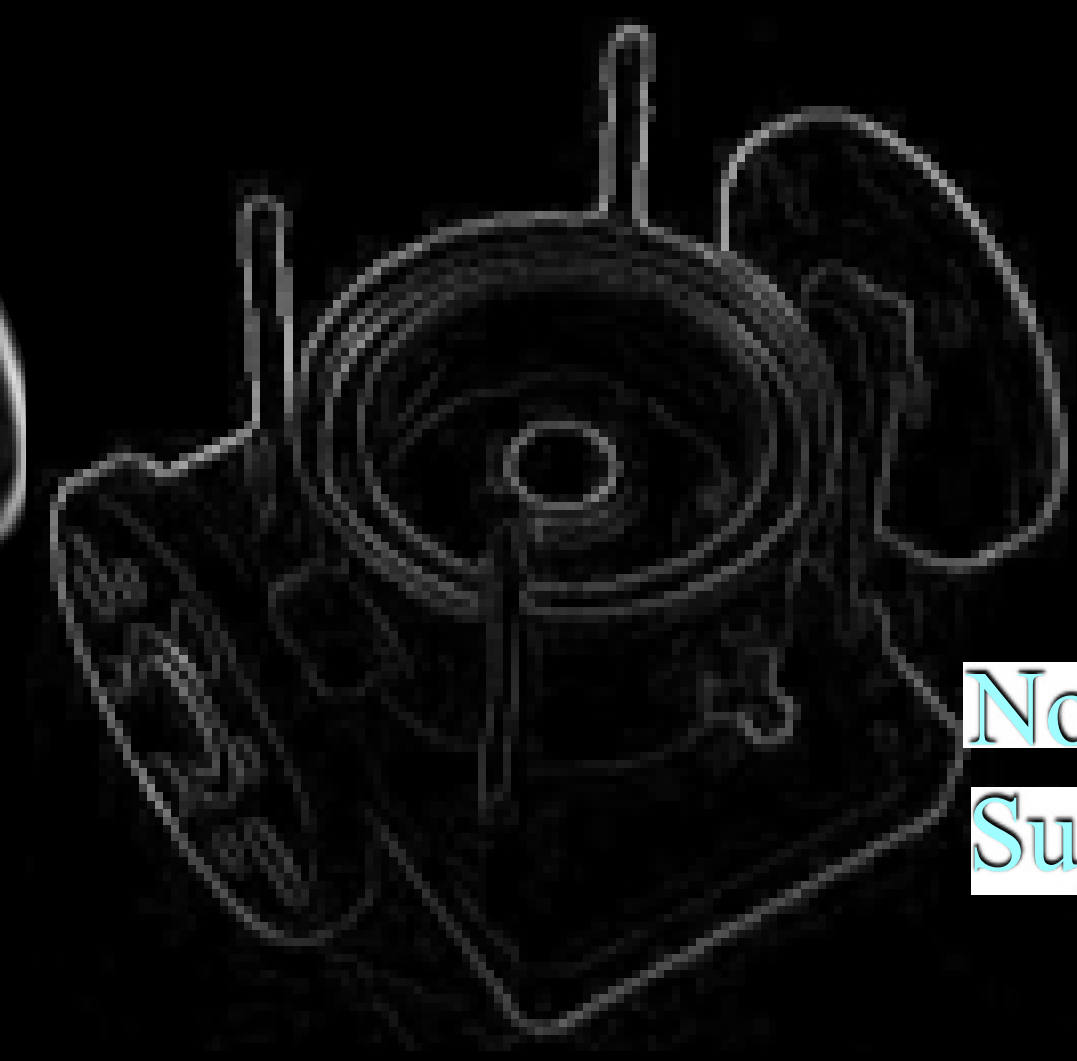
4. Double hysteresis thresholding

- $>$ High threshold $\Rightarrow$ strong edge pixels
- $<$ Low threshold $\Rightarrow$ weak edge pixels

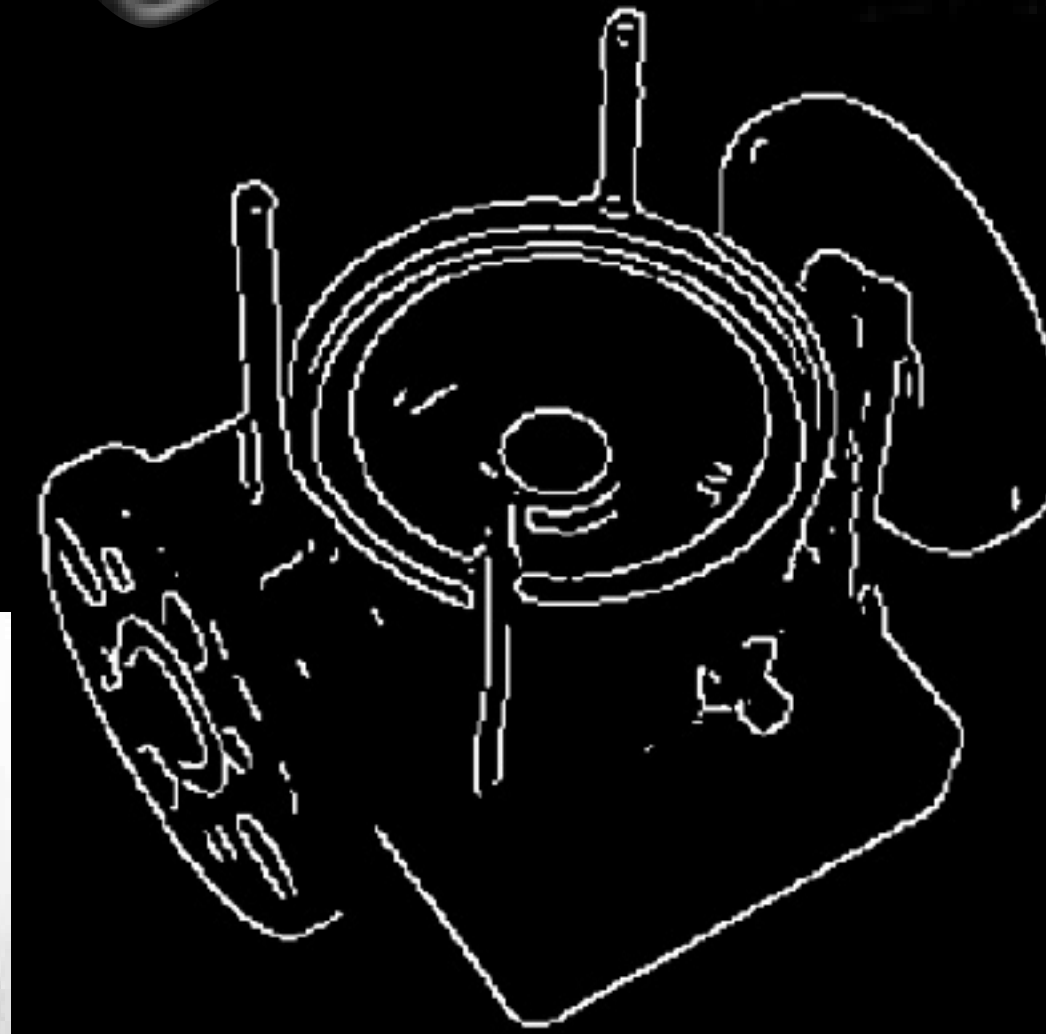
5. Mark weak edges connected to strong edges as strong edges



Gradients



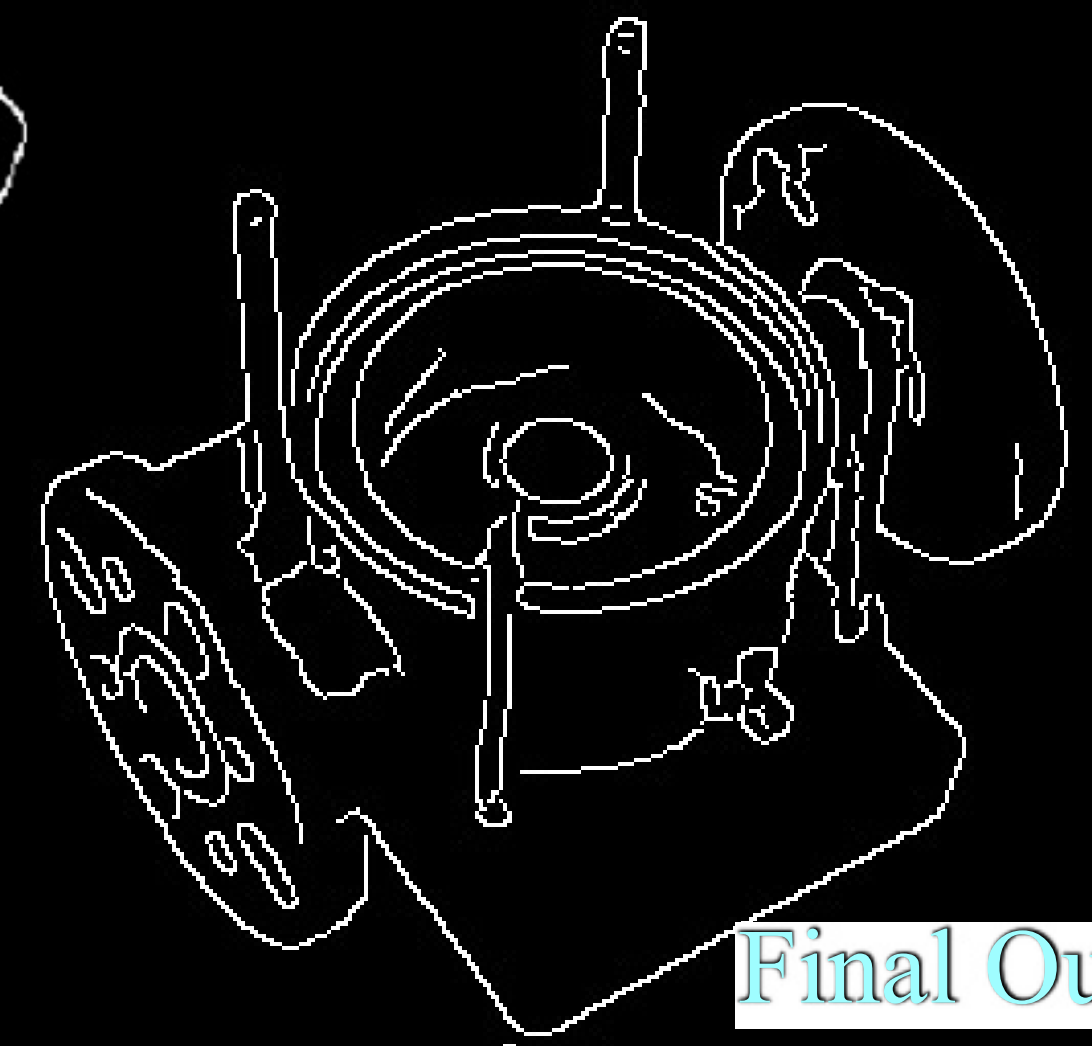
Non-maximal
Suppression



Strong Edges
(Thresholded)



Input Image



Final Output



Original



Marr-Hildreth



Canny

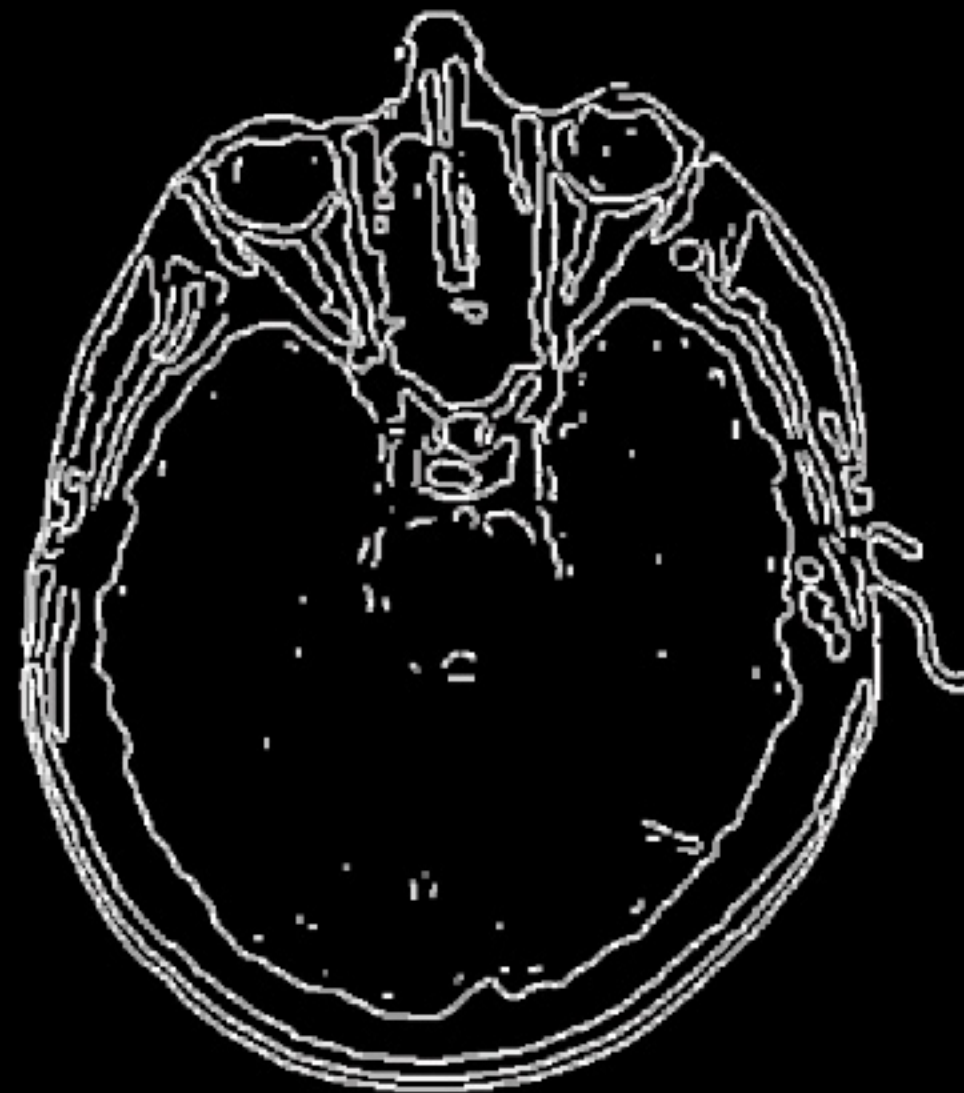
Original



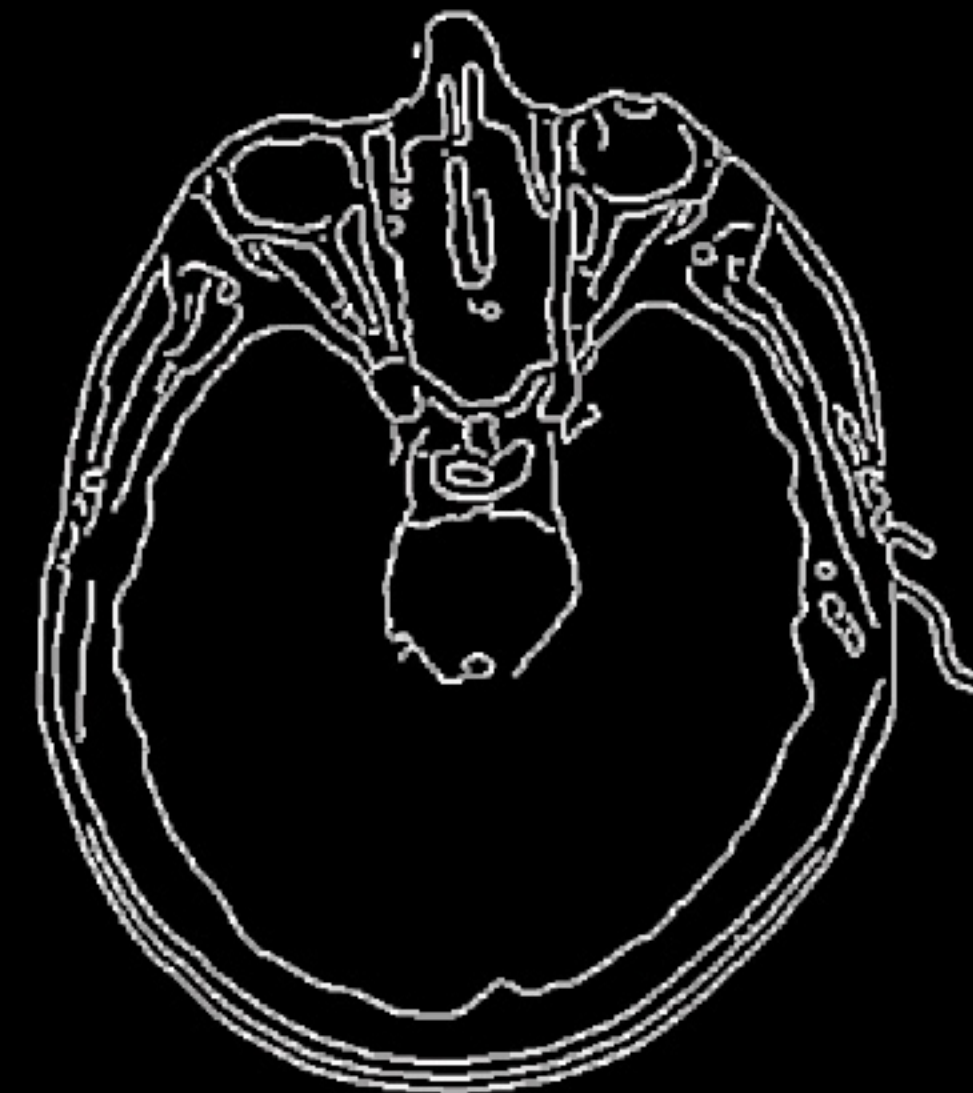
Threshold of
gradient magitude



Marr-Hildreth



Canny



Connecting “Edge” Pixels

Towards continuous lines/curves

- Local processing: close gaps between edge pixels that are “aligned”
- Gap filling: Closure — Dilate then Erode
- Global processing: **Hough transform**

