Computer Graphics -Edge Detection

Junjie Cao @ DLUT

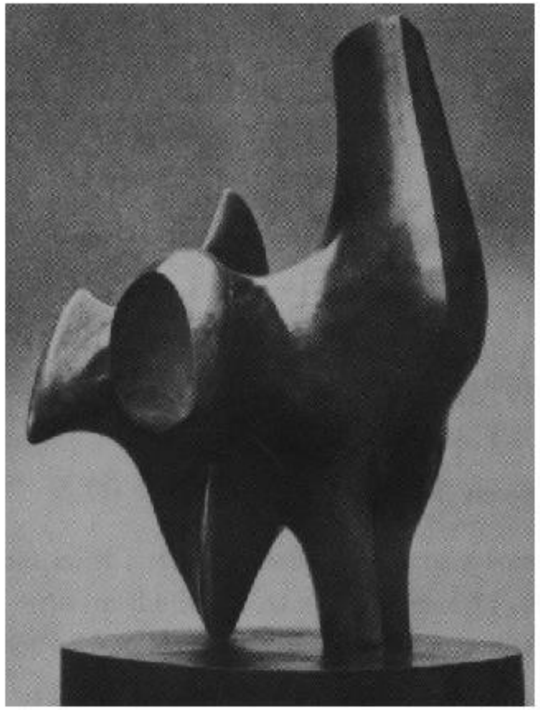
Spring 2016

http://jjcao.github.io/ComputerGraphics/

Agenda

- What is an edge?
- Type of edges
- Edge detection methods

Edge Detection

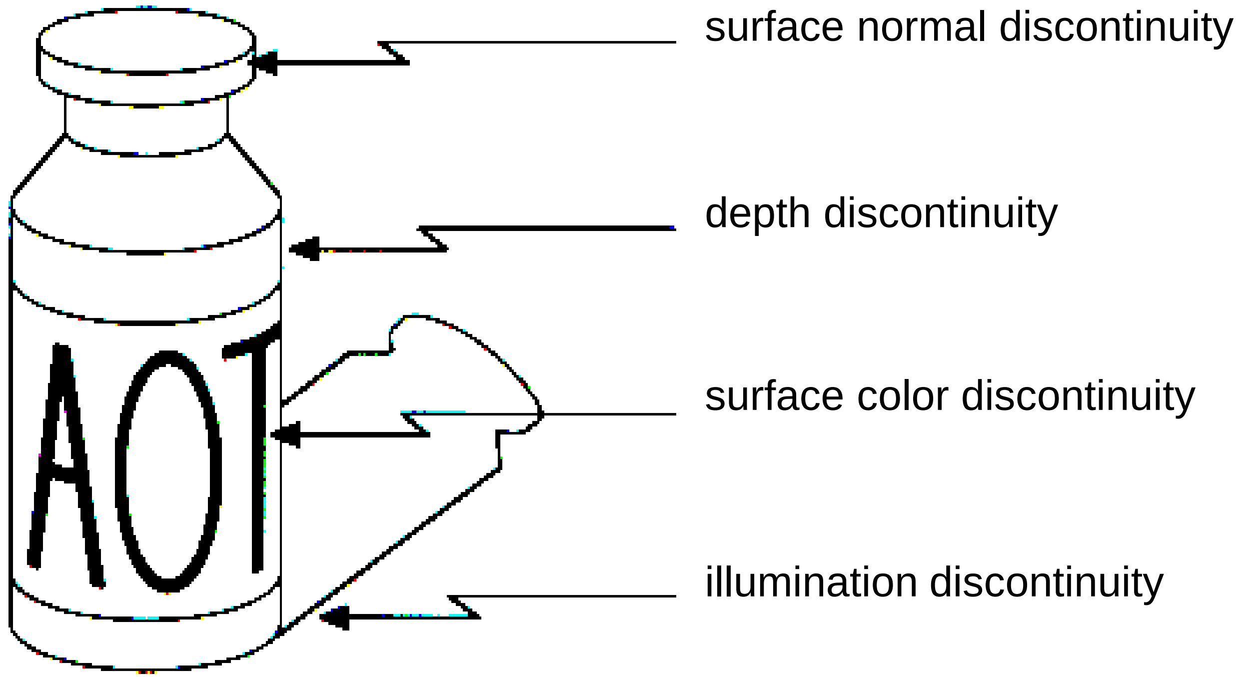


- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - They usually correspond to object boundaries - segmentation.
  - More compact than pixels while retaining most of the image information.

Origin of Edges

- Geometric events
  - surface orientation (boundary) discontinuities
  - depth discontinuities
  - color and texture discontinuities

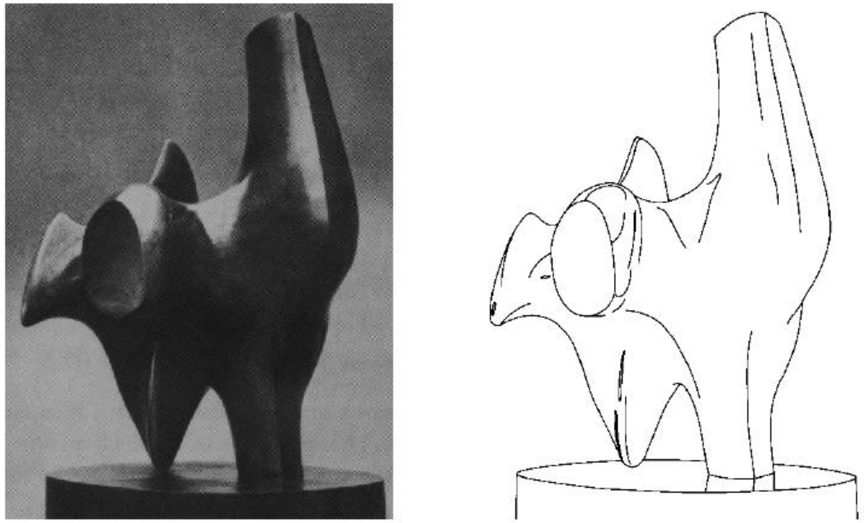
- Non-geometric events
  - illumination changes
  - specularities
  - shadows
  - inter-reflections

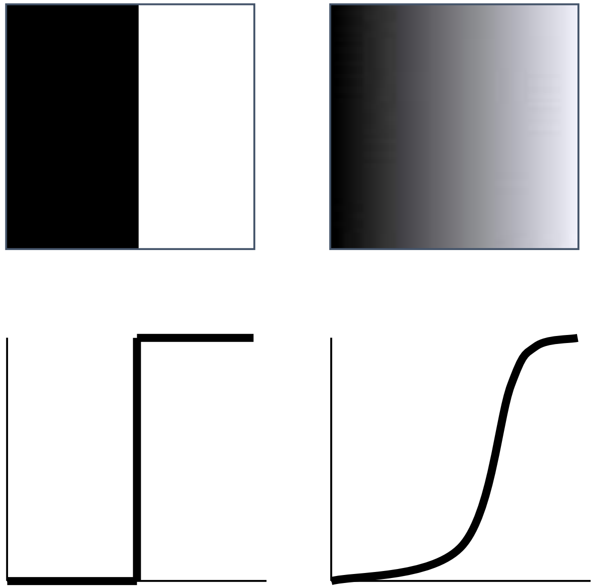


Edges are caused by a variety of factors

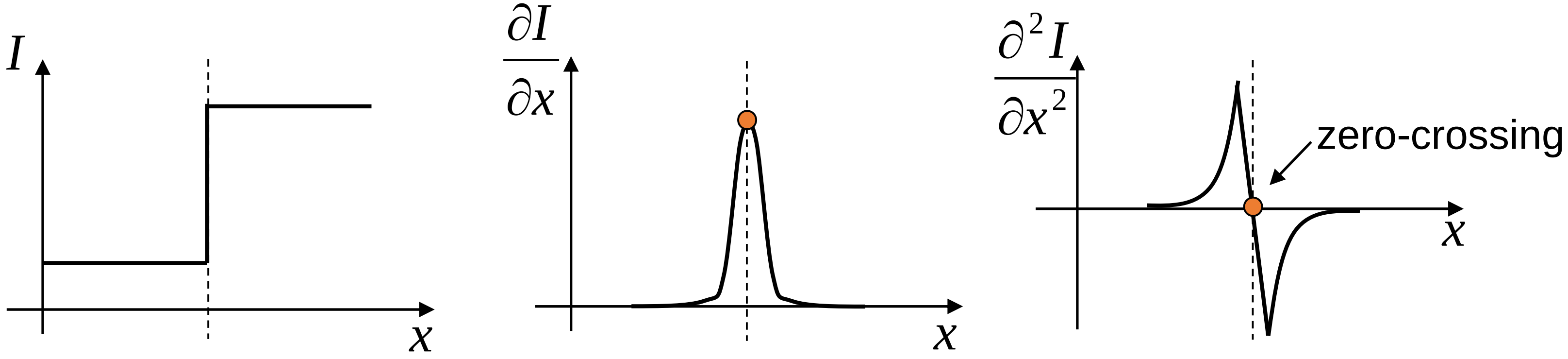
How can you tell that a pixel is on an edge?

- Biggest local change, derivative has maximum magnitude
- Or 2nd derivative is zero.

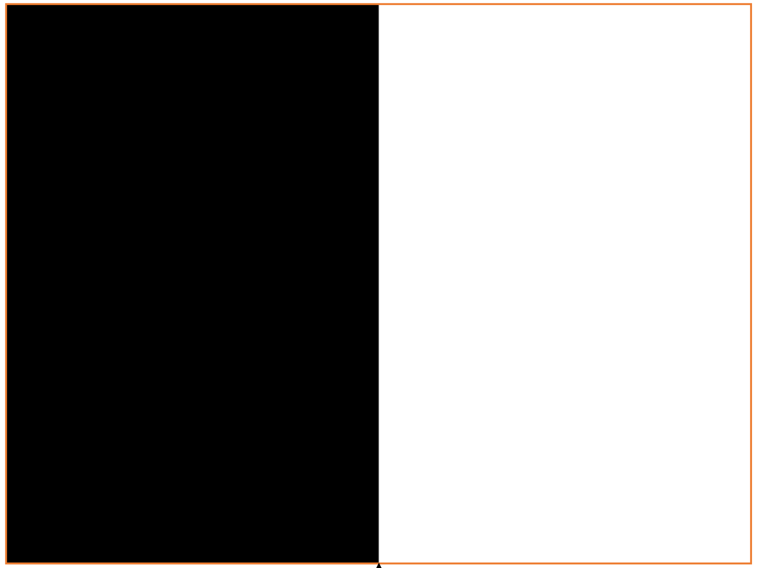




Brightness vs. Spatial Coordinates

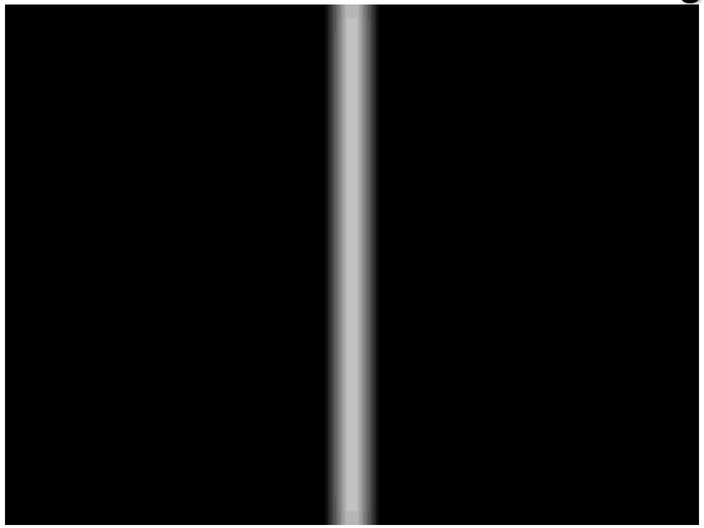


What is an Edge?



Edge easy to find

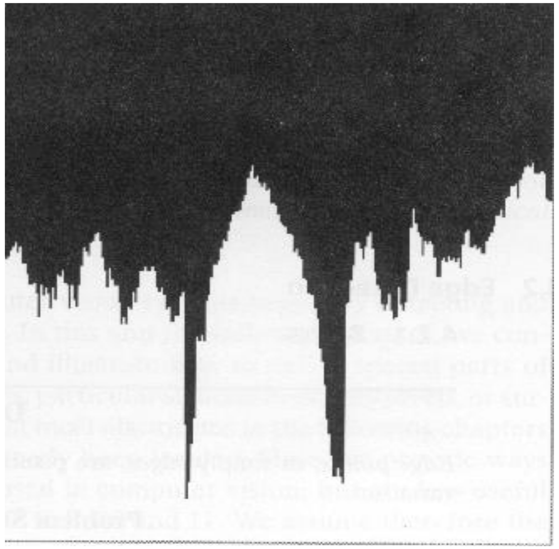
Where is edge? Single pixel wide or multiple pixels?



Is this one edge or two?

Real Edges

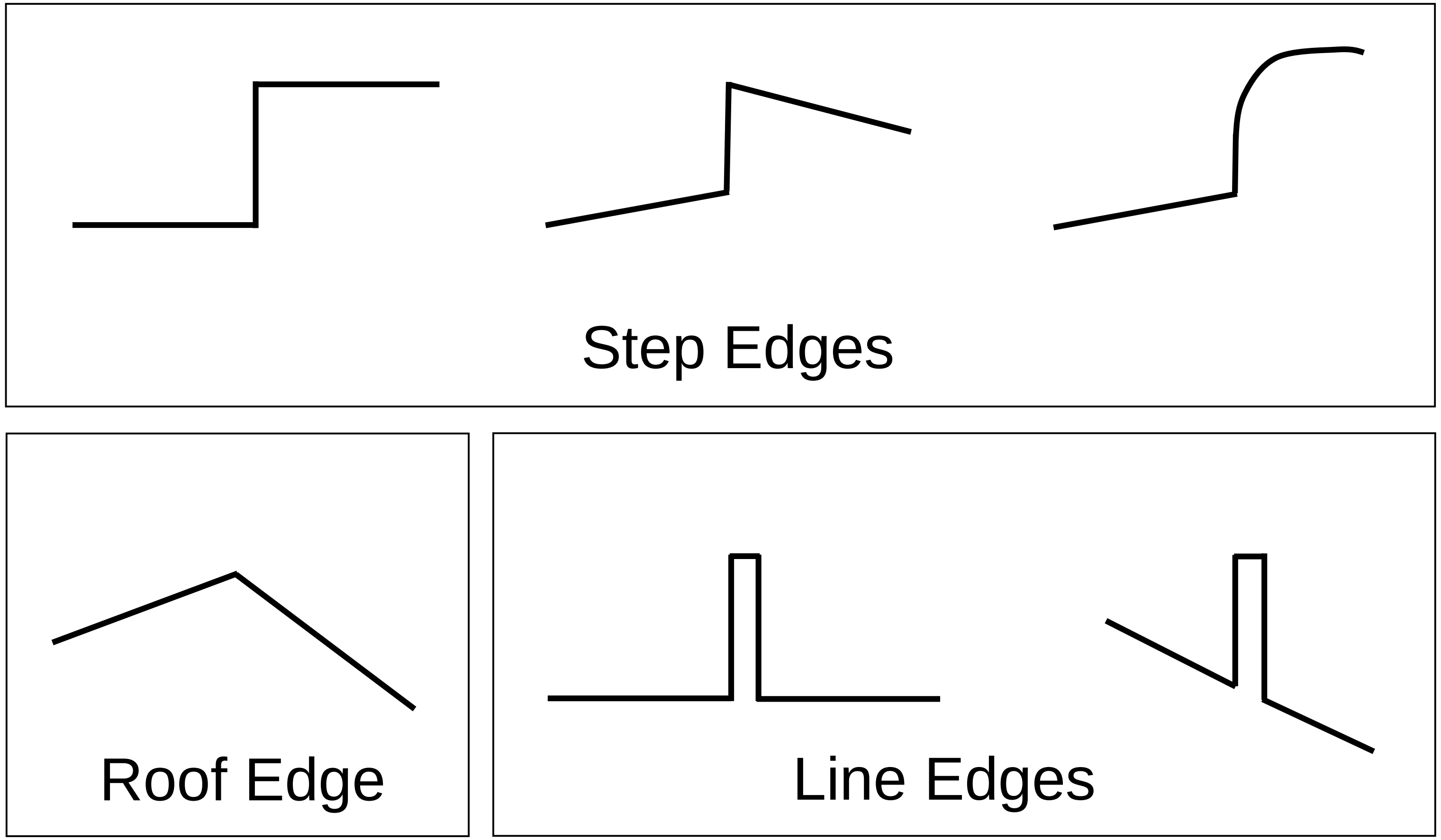
- Edge operator produces
  - Edge Magnitude
  - Edge Orientation



- Gradient is high everywhere.
- Must smooth before taking gradient.

- Edge detector should have:
  - High **Detection** Rate. Filter responds to edge, not noise.
  - Good **Localization**: detected edge near true edge.
  - Single Response: one per edge.

Edge Types



Gradient/Edge Operators

- Motivation: detect changes

change in the pixel value $\longrightarrow$ large gradient

$$\underset{x(m,n)}{\text{image}} \longrightarrow \boxed{\text{Gradient operator}} \xrightarrow{g(m,n)} \boxed{\text{Thresholding}} \longrightarrow \underset{I(m,n)}{\text{edge map}}$$

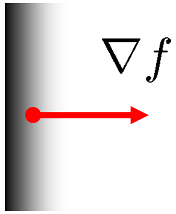
$$I(m,n) = \begin{cases} 1 & |g(m,n)| > th \\ 0 & otherwise \end{cases}$$

Image gradient

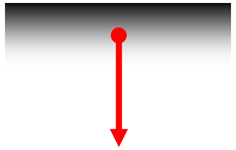
- The gradient of an image:

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

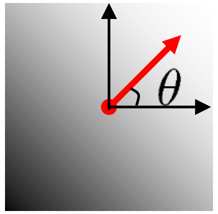
- The gradient points in the direction of most rapid change in intensity



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



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



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

The gradient direction is given by:

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

- how does this relate to the direction of the edge?

The *edge strength* is given by the gradient magnitude

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

The discrete gradient

- How can we differentiate a *digital* image f[x,y]?
  - Option 1: reconstruct a continuous image, then take gradient
  - Option 2: take discrete derivative (finite difference)

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

The Sobel operator

- Better approximations of the derivatives exist
  - The *Sobel* operators below are very commonly used

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{8}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

$$s_x \qquad\qquad s_y$$

- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn’t make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient value, however

Prewitt operator

vertical $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ horizontal $\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

Examples

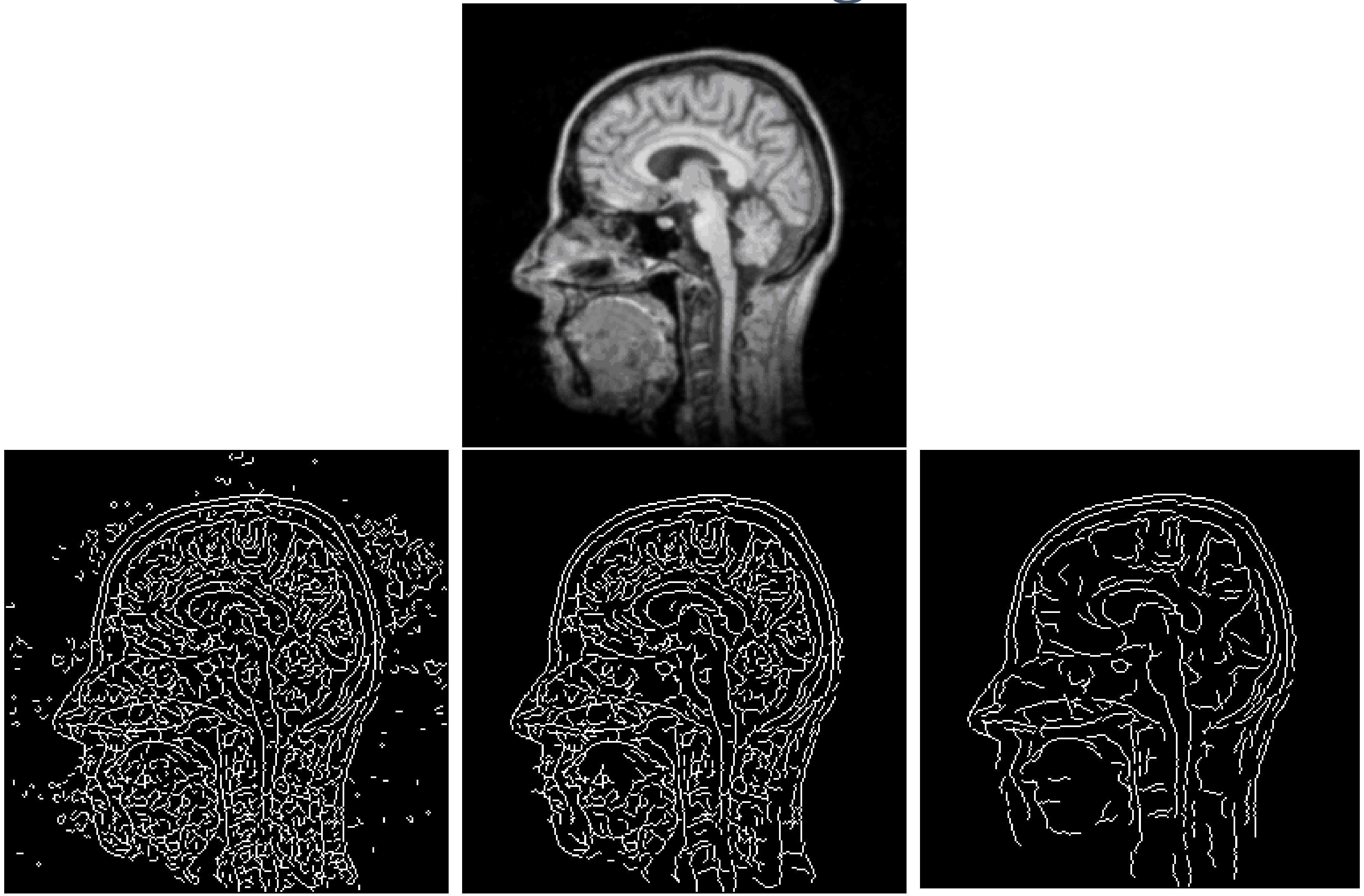

original image


horizontal edge


vertical edge

Prewitt operator (*th=48*)

Effect of Thresholding Parameters



small → threshold → large

Some Edge Operators

- Kirsch

+5	+5	+5
-3	0	-3
-3	-3	-3

-3	+5	+5
-3	0	+5
-3	-3	-3

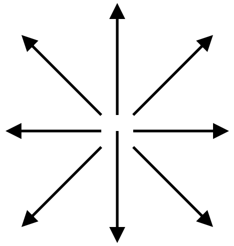
-3	-3	+5
-3	0	+5
-3	-3	+5

-3	-3	-3
-3	0	+5
-3	+5	+5

Compass Operators

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$



$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$g(m,n) = \max_k \{ | g_k(m,n) | \}$$

Examples

Compass operator (*th=48*)

Prewitt operator (*th=48*)

horizontal edge

vertical edge

Comparing Edge Operators

Gradient:

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

Roberts (2 x 2):

0	**1**
-1	**0**

1	**0**
0	**-1**

Sobel (3 x 3):

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

1	2	1
0	0	0
-1	-2	1

Sobel (5 x 5):

-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**

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

Laplacian Operators

- Gradient operator: first-order derivative

sensitive to abrupt change, but not slow change

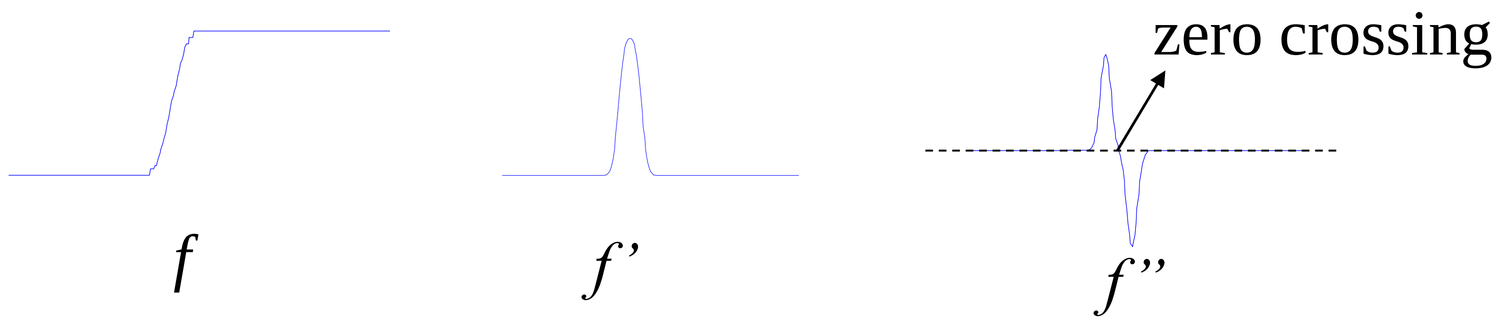
second-order derivative: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$ (Laplacian operator)

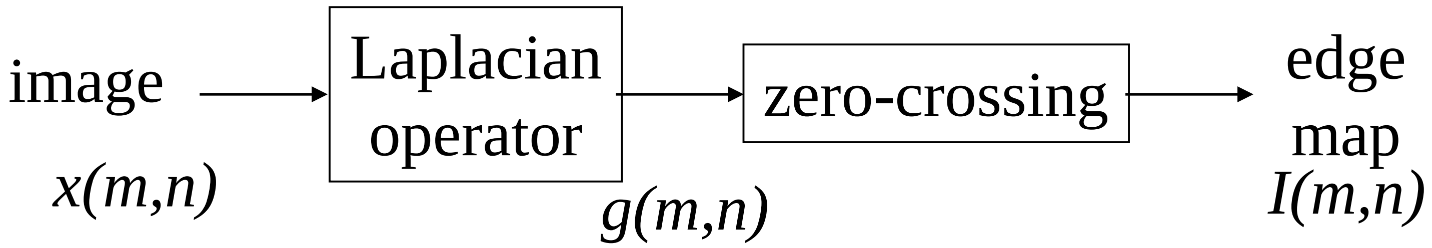
$\frac{\partial^2 f}{\partial x^2} = 0 \longrightarrow$ local extreme in f'

- Discrete Laplacian operator

$$\frac{1}{1+a}\begin{bmatrix} a & 1-a & a \\ 1-a & -4 & 1-a \\ a & 1-a & a \end{bmatrix} \quad \underset{a=0}{\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}} \quad \underset{a=0.5}{\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}}$$

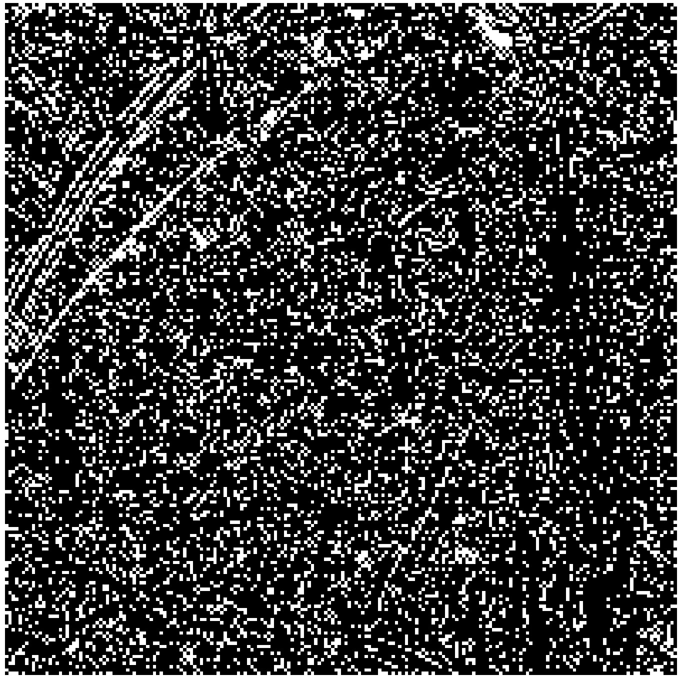
Zero Crossings





Examples

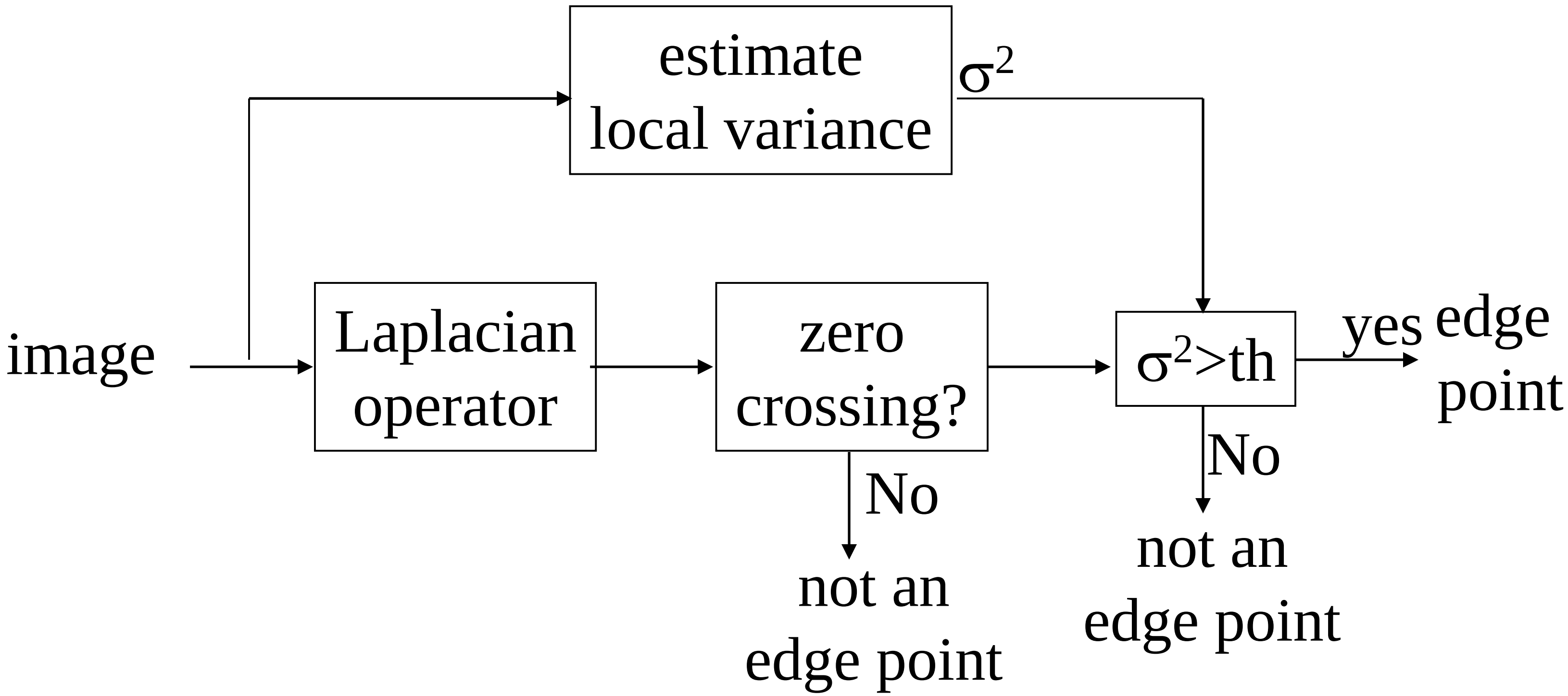
original image



zero-crossings

Question: why is it so sensitive to noise (many false alarms)?
Answer: a sign flip from 0.01 to -0.01 is treated the same as from 100 to -100

Robust Laplacian-based Edge Detector



Examples

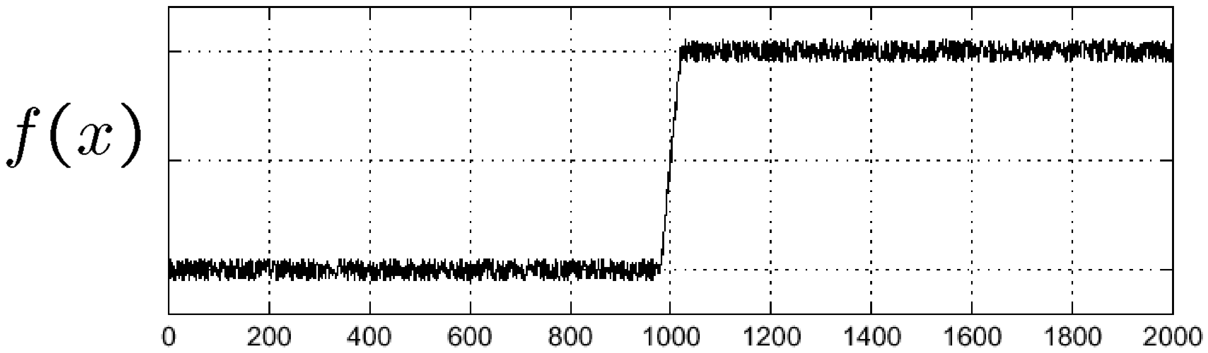
More robust but return multiple edge pixels (poor localization)

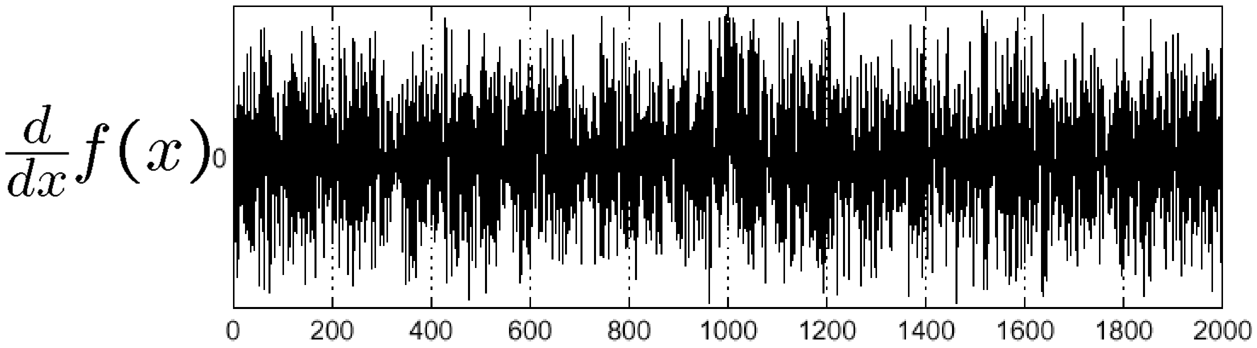
Canny Edge Detector*

- Low error rate of detection
  - Well match human perception results
- Good localization of edges
  - The distance between actual edges in an image and the edges found by a computational algorithm should be minimized
- Single response
  - The algorithm should not return multiple edges pixels when only a single one exists

Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

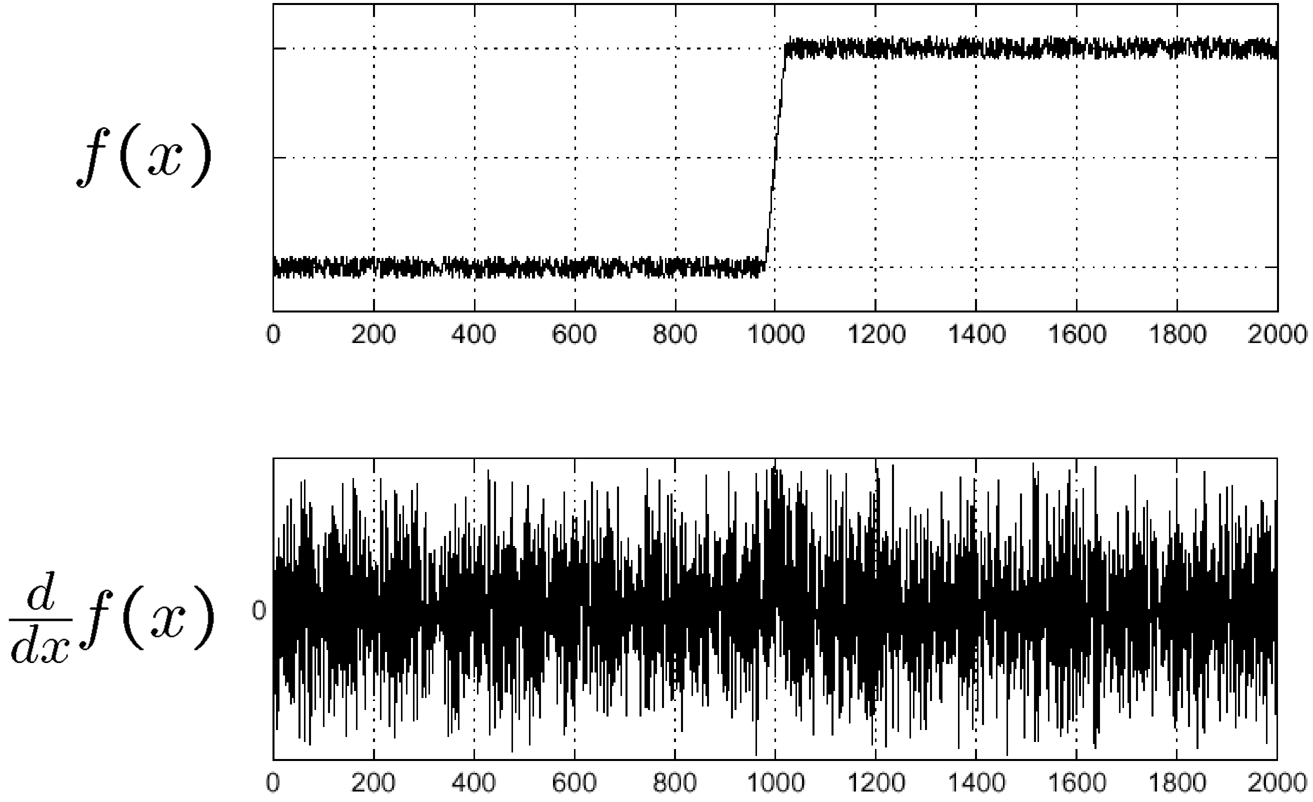




Where is the edge?

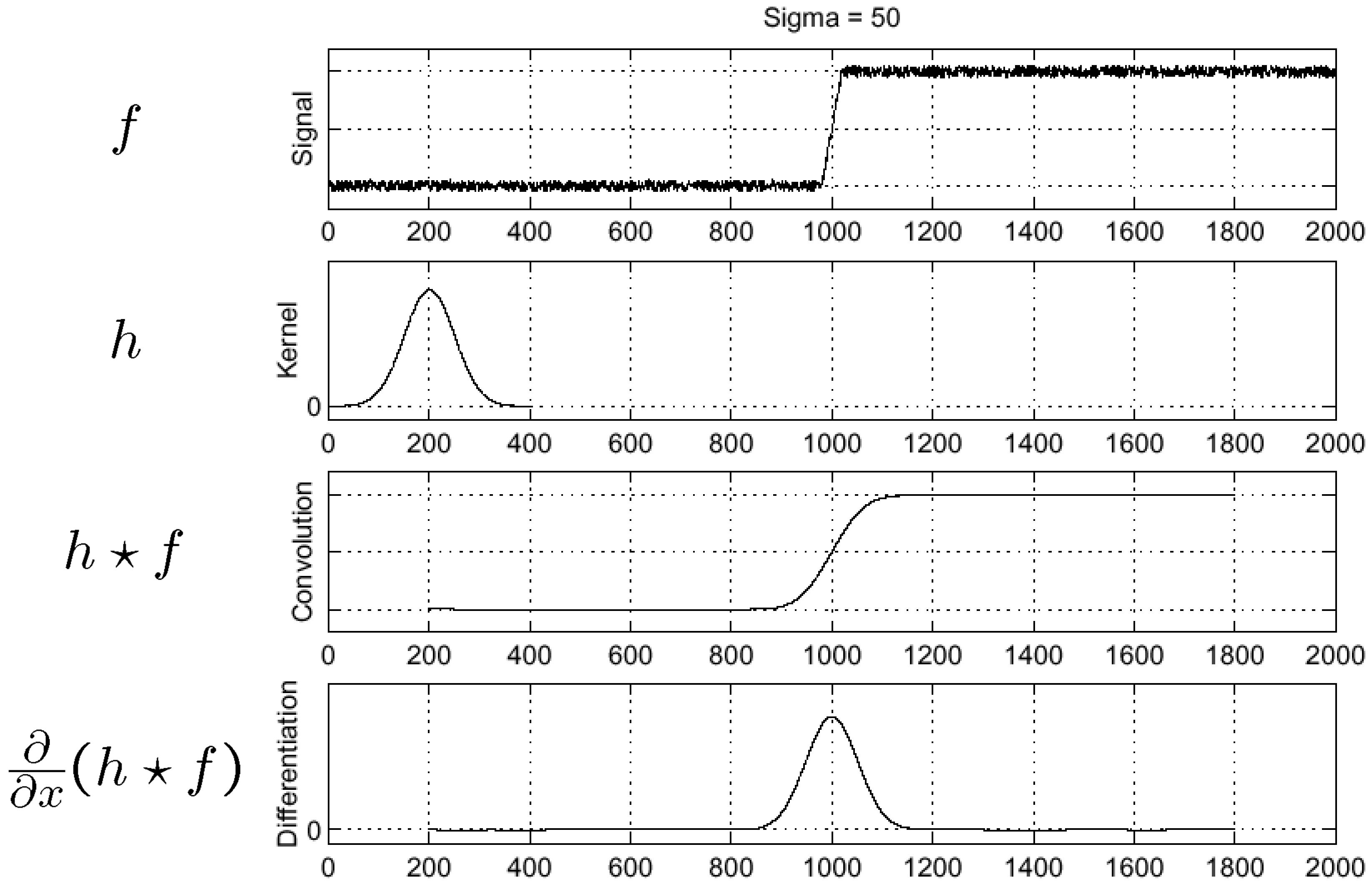
Effects of Noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal



Where is the edge??

Solution: smooth first



Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Smoothing and Differentiation

- Derivative theorem of convolution:
  - because differentiation is convolution, and convolution is associative

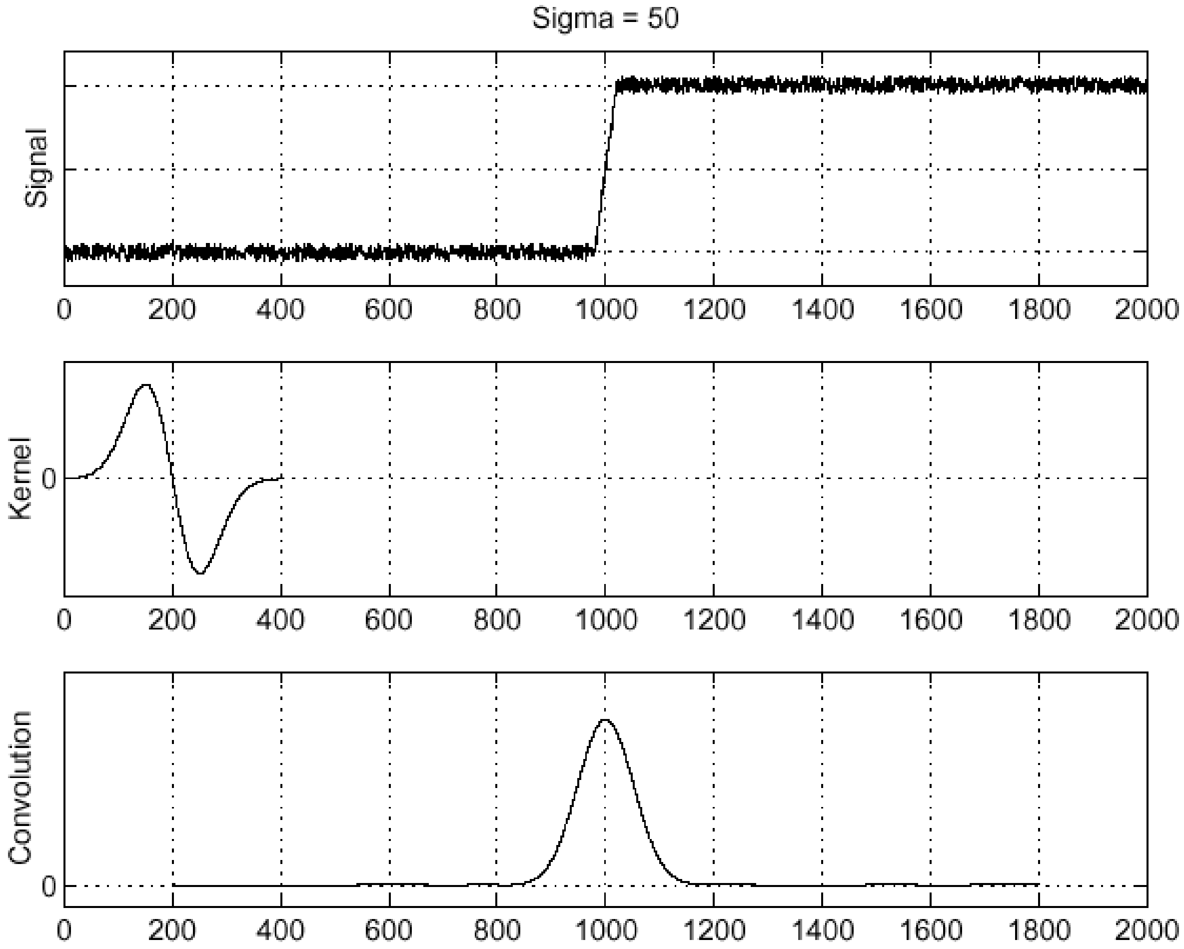
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

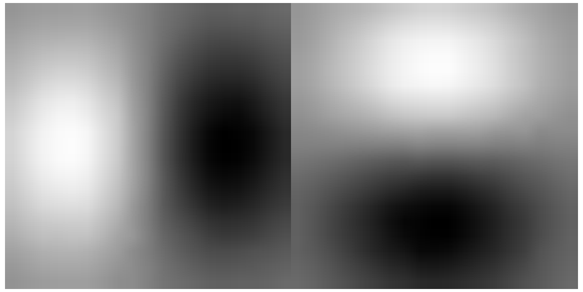
- This saves us one operation:

f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$





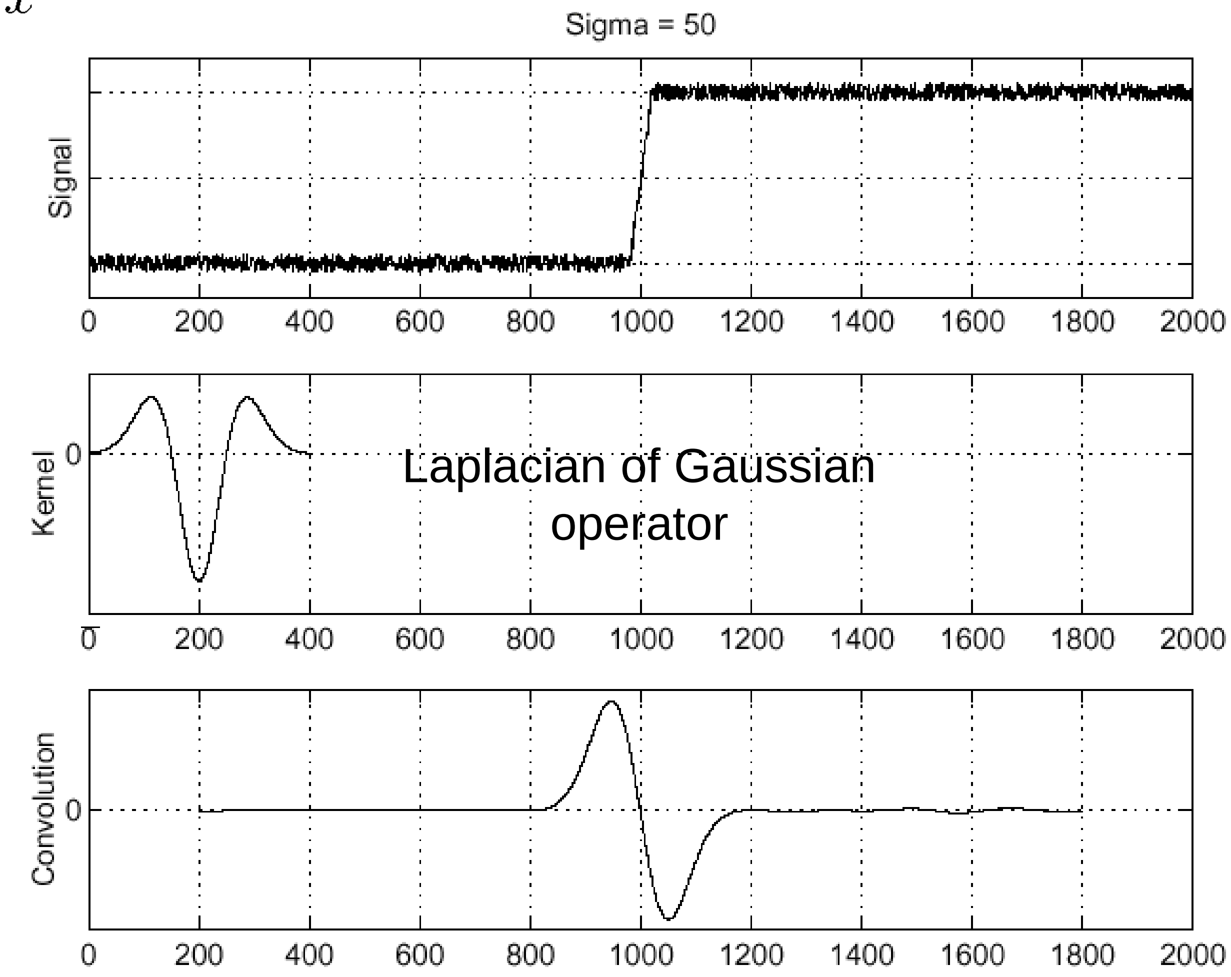
Laplacian of Gaussian

- Conside$\frac{\partial^2}{\partial x^2}(h \star f)$

f

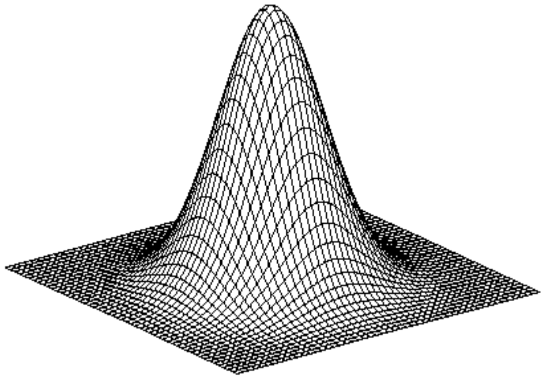
$\frac{\partial^2}{\partial x^2}h$

$(\frac{\partial^2}{\partial x^2}h) \star f$



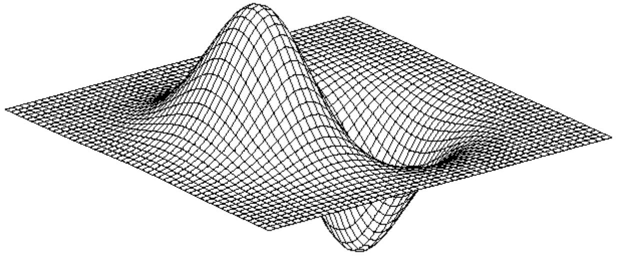
Where is the edge? Zero-crossings of bottom graph

2D edge detection filters



Gaussian

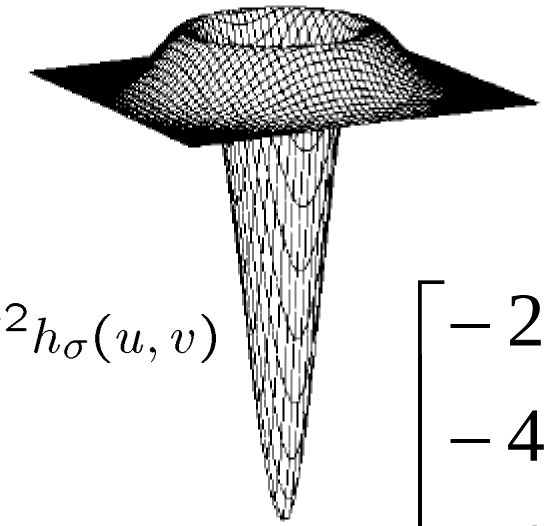
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian (DoG)

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

Laplacian of Gaussian (LoG)
Mexican Cap



$$\nabla^2 h_\sigma(u,v)$$

$$\begin{bmatrix} -2 & -4 & -4 & -4 & -2 \\ -4 & 0 & 8 & 0 & -4 \\ -4 & 8 & 24 & 8 & -4 \\ -4 & 0 & 8 & 0 & -4 \\ -2 & -4 & -4 & -4 & -2 \end{bmatrix}$$

∇^2 is the **Laplacian** operator:

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

Flow-chart of Canny Edge Detector*

(*J. Canny'1986*)

Original image

↓

Smoothing by Gaussian convolution

↓

Differential operators along x and y axis

↓

Non-maximum suppression
finds peaks in the image gradient

↓

Hysteresis thresholding locates edge strings

↓

Edge map

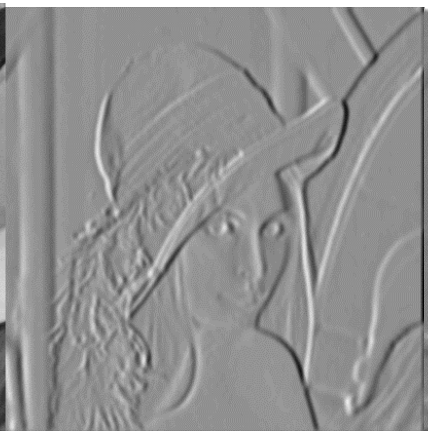
- Assume:
  - Linear filtering
  - Additive iid Gaussian noise
- Prons:
  - High **Detection** Rate. Filter responds to edge, not noise.
  - Good **Localization**: detected edge near true edge.
  - Single Response: one per edge.

Optimal Edge Detection: Canny (continued)

- Optimal Detector is approximately Derivative of Gaussian.
- Detection/Localization trade-off
  - More smoothing improves detection
  - And hurts localization.
- This is what you might guess from (detect change) + (remove noise)

Canny Edge Detector Example

original image



vertical edges

horizontal edges

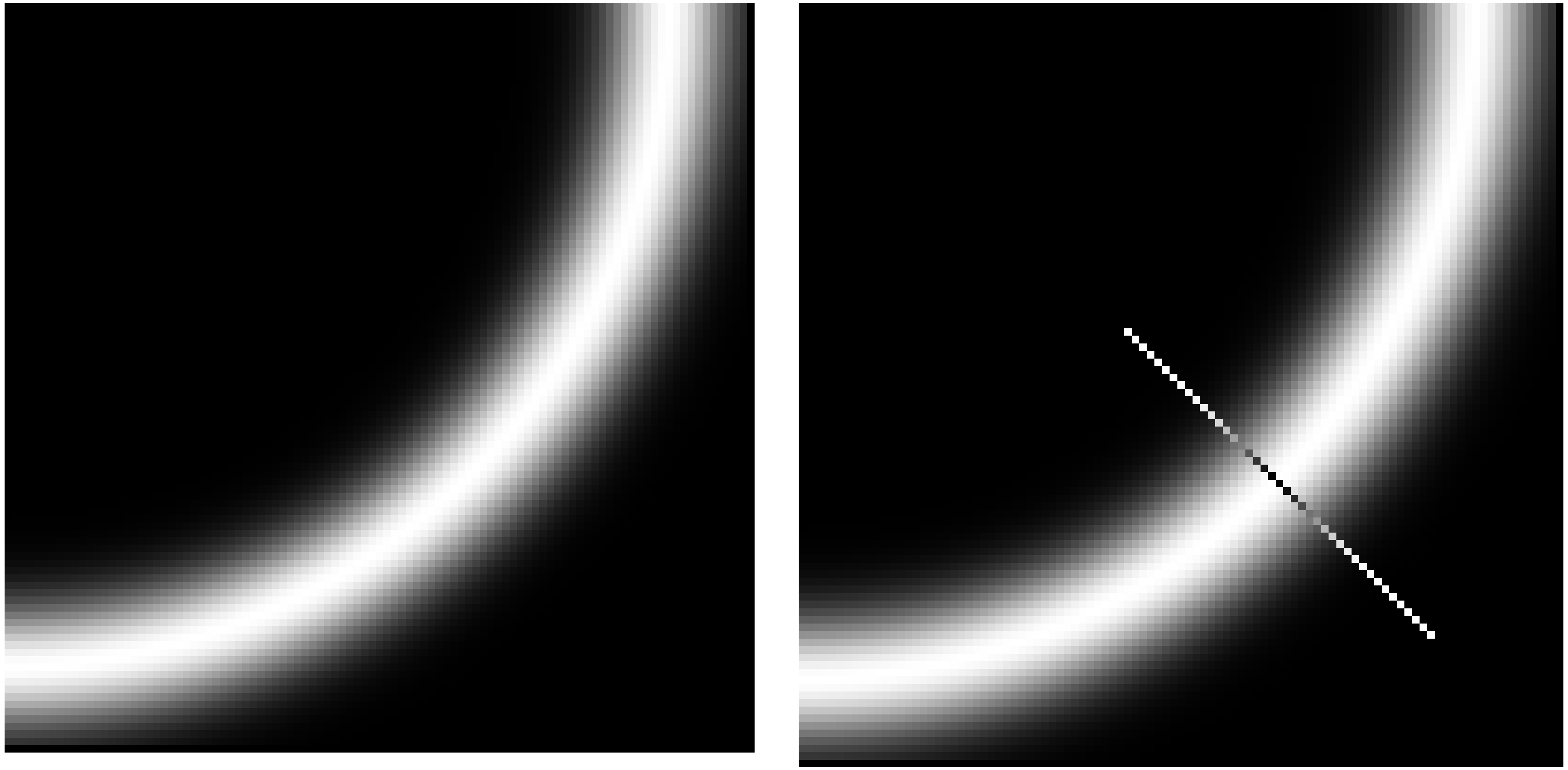
norm of the gradient

thresholding

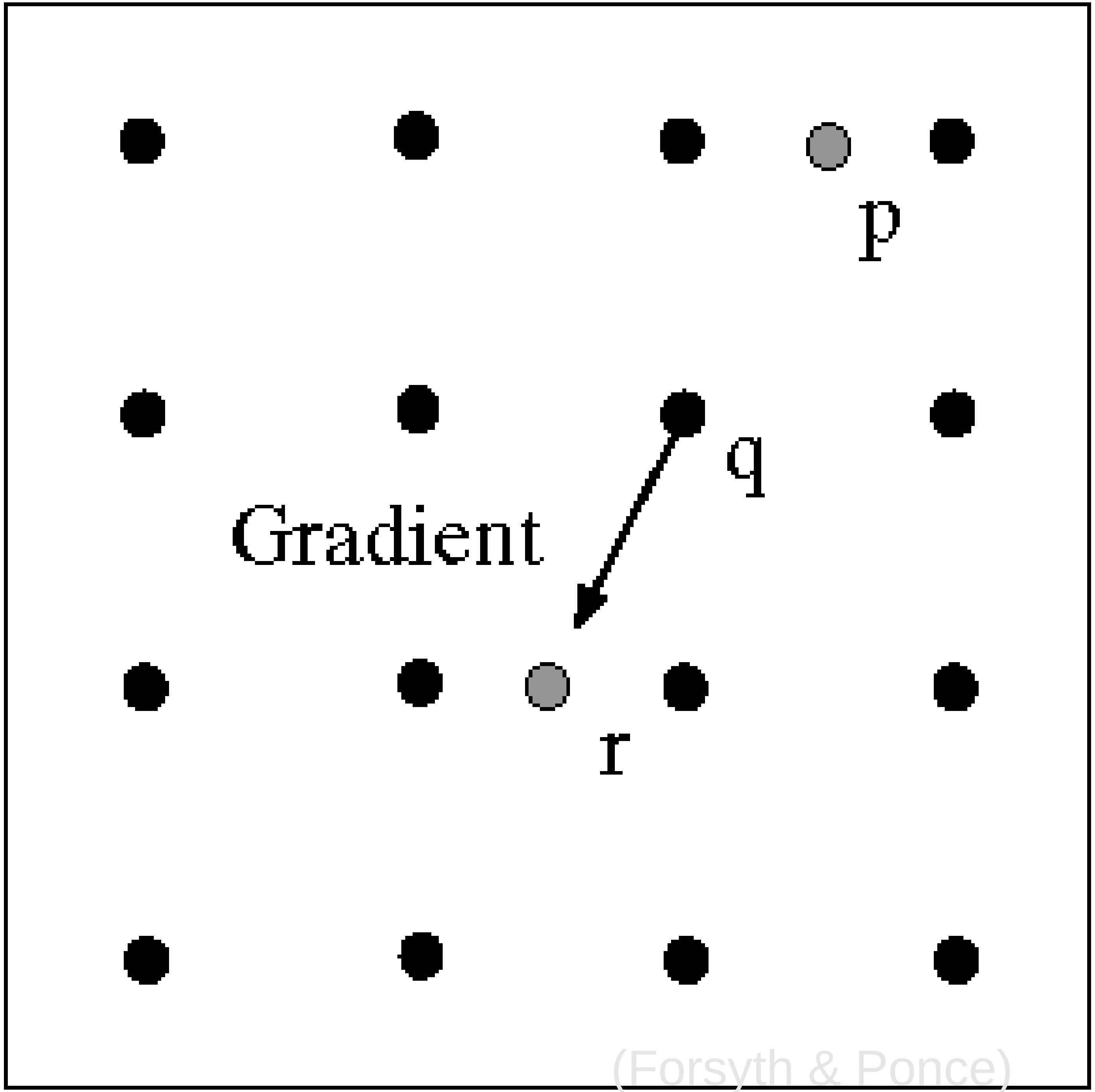
Thinning
(non-maximum suppression)

Finding the Peak

1) The gradient magnitude is large along thick trail; how do we identify the significant points?
2) How do we link the relevant points up into curves?

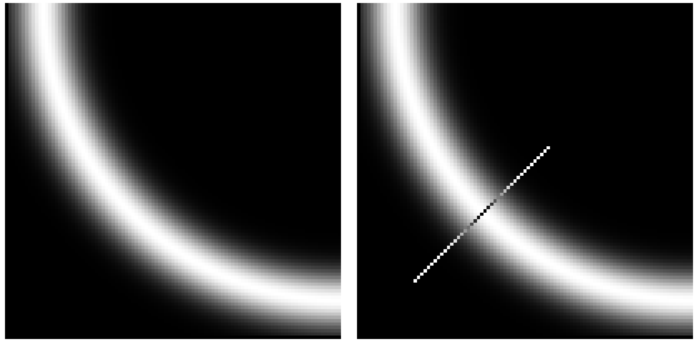


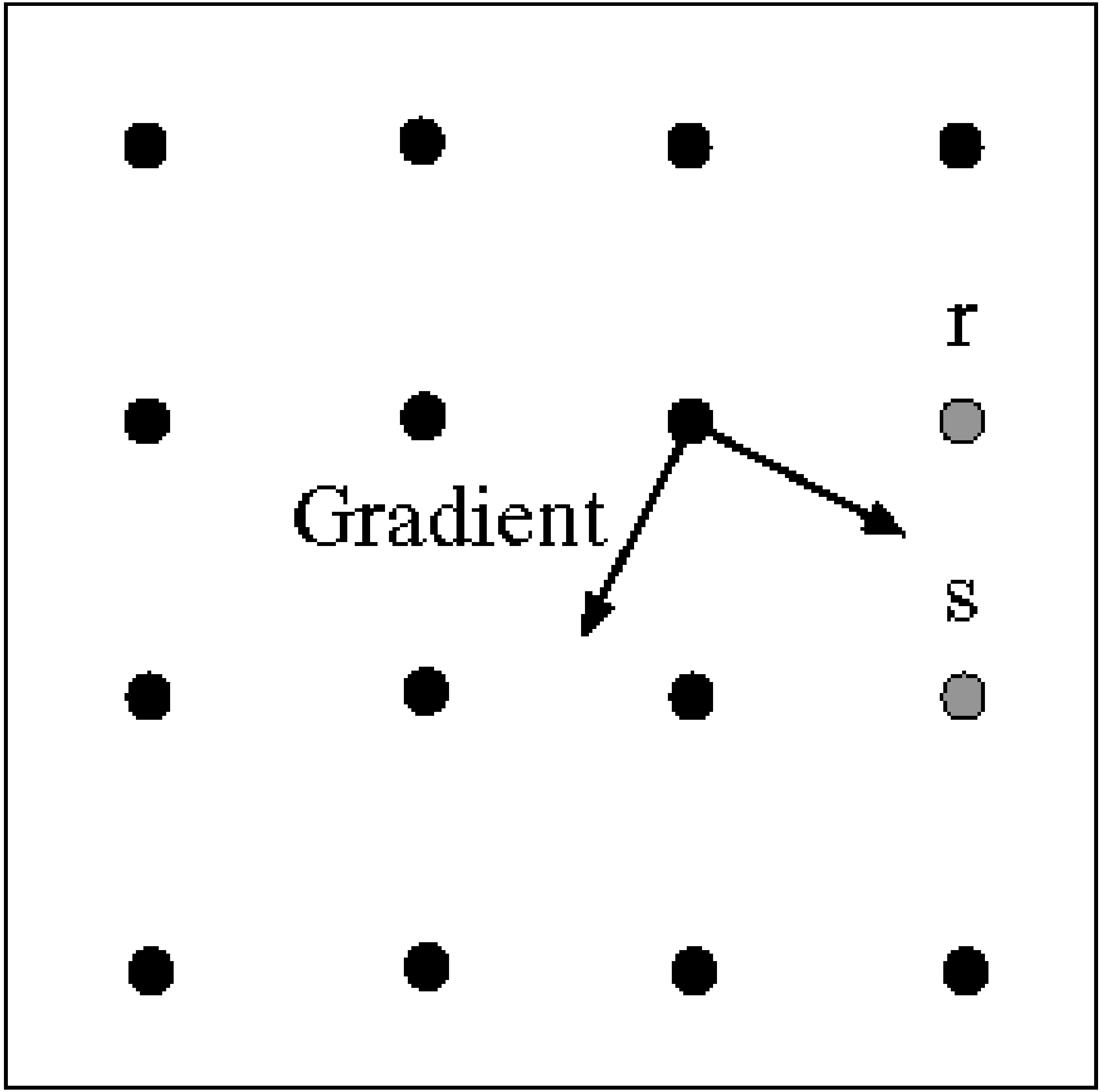
We wish to mark points along the curve where the magnitude is biggest.
We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues: at which point is the maximum, and where is the next one?



Non-maximum suppression

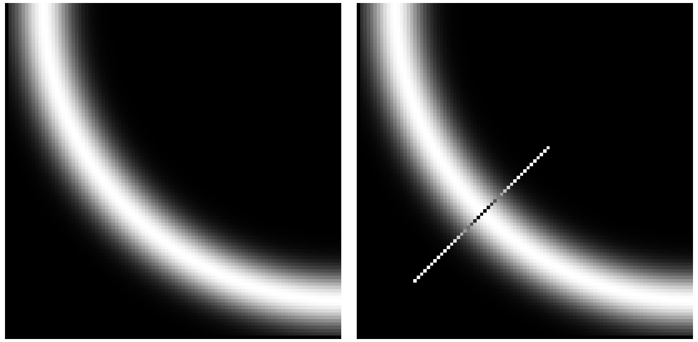
- At q, we have a maximum if the value is larger than those at both p and at r.
- Interpolate to get these values.





Predicting the next edge point

- Assume the marked point is an edge point.
- Then we construct the tangent to the edge curve (which is normal to the gradient at that point)
- and use this to predict the next points (here either r or s).

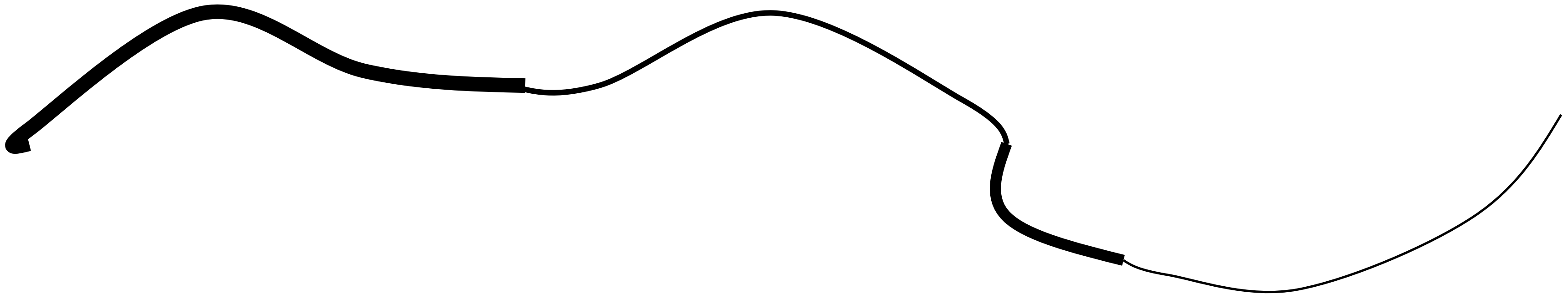


Edge Thresholding & Hysteresis

- Standard Thresholding: $E(x,y) = \begin{cases} 1 & \text{if } \|\nabla f(x,y)\| > T \text{ for some threshold } T \\ 0 & \text{otherwise} \end{cases}$
  - Can only select "strong" edges.
  - Does not guarantee "continuity".
- Hysteresis based Thresholding (use two thresholds)

$$\begin{array}{rl} \|\nabla f(x,y)\| \geq t_1 & \text{definitely an edge} \\ t_0 \geq \|\nabla f(x,y)\| < t_1 & \text{maybe an edge, depends on context} \\ \|\nabla f(x,y)\| < t_0 & \text{definitely not an edge} \end{array}$$

Example: For "maybe" edges, decide on the edge if neighboring pixel is a strong edge.



Canny Edge Operator

- Smooth image *I* with 2D Gaussian: $G * I$

- Find local edge normal directions for each pixel

$$\overline{\mathbf{n}} = \frac{\nabla(G * I)}{\left|\nabla(G * I)\right|}$$

- Compute edge magnitudes $\left|\nabla(G * I)\right|$

- Locate edges by finding zero-crossings along the edge normal directions (**non-maximum suppression**)

$$\frac{\partial^2 (G * I)}{\partial \overline{\mathbf{n}}^2} = 0$$

Effect of σ (Gaussian kernel size)

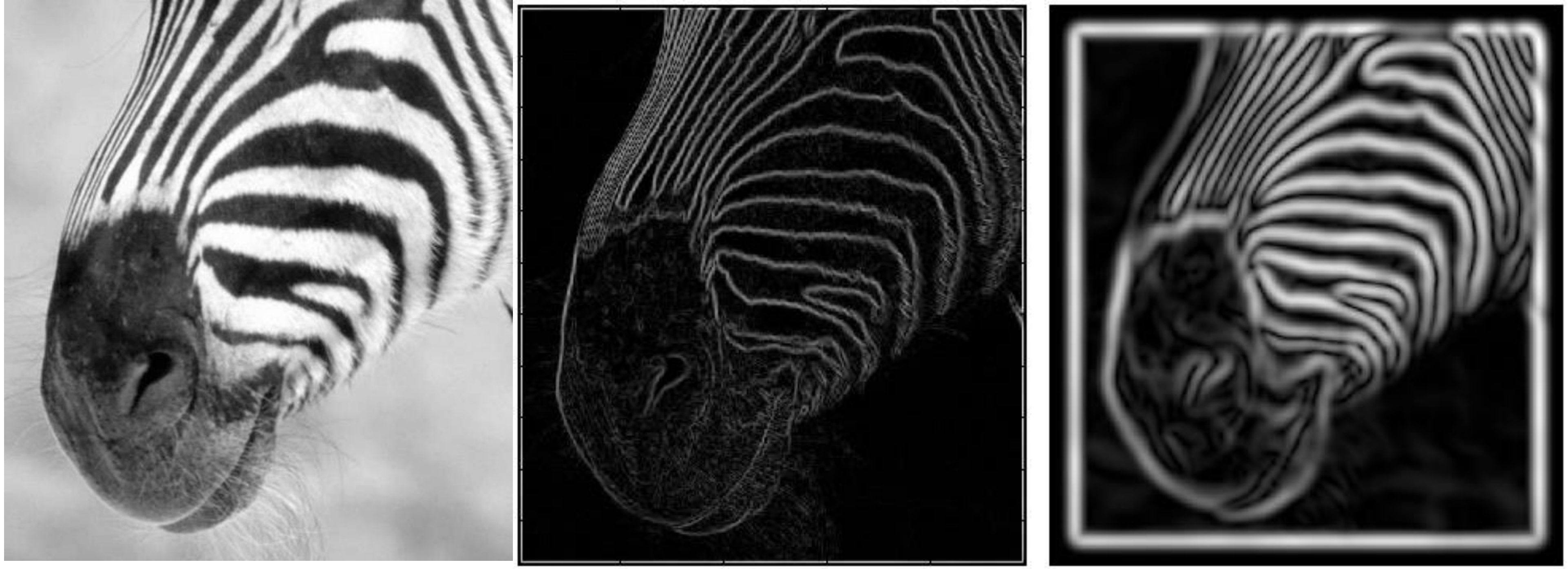
original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

The choice of σ depends on desired behavior

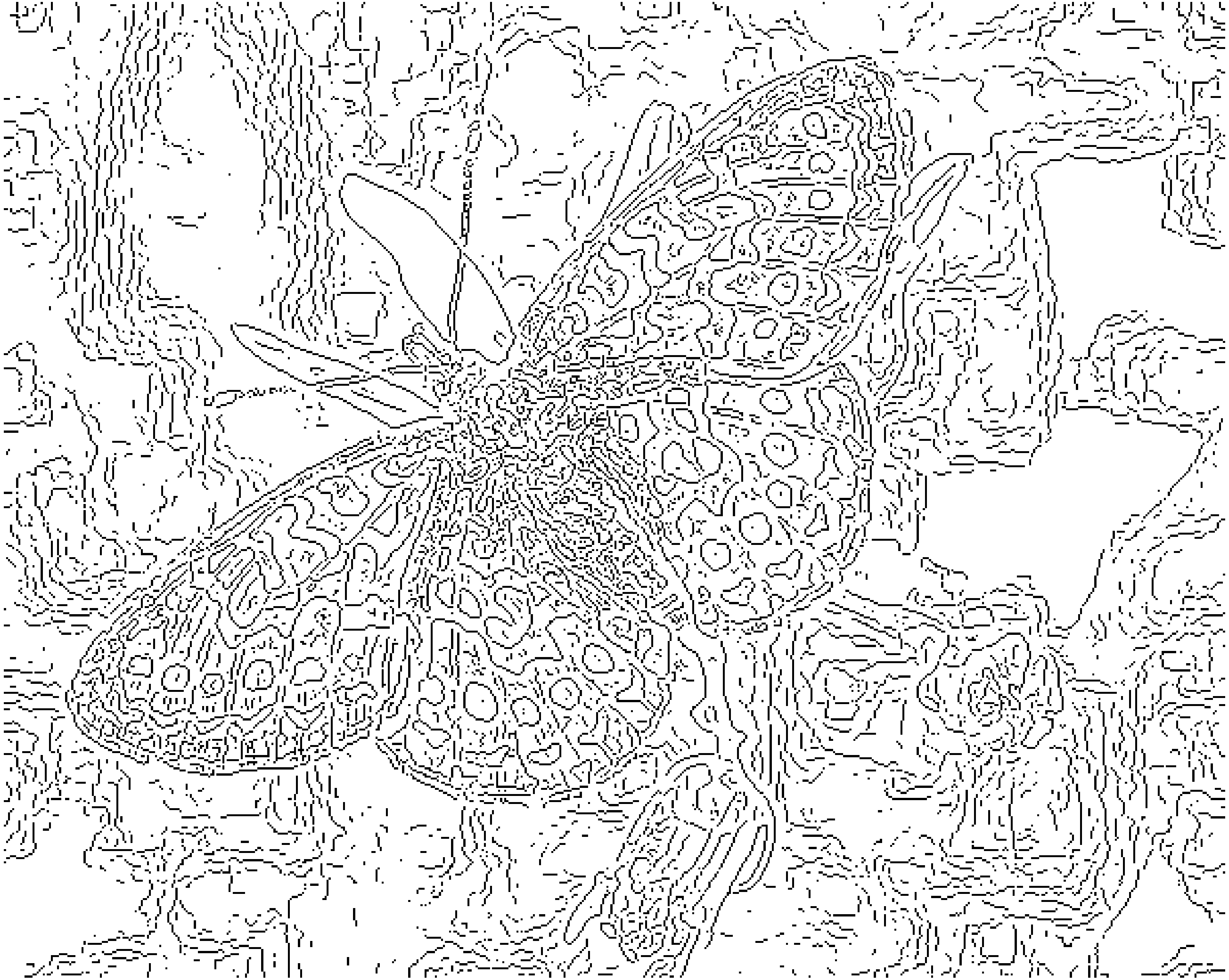
- large σ detects large scale edges
- small σ detects fine features



Scale

- Smoothing
- Eliminates noise edges.
- Makes edges smoother.
- Removes fine detail.

(Forsyth & Ponce)

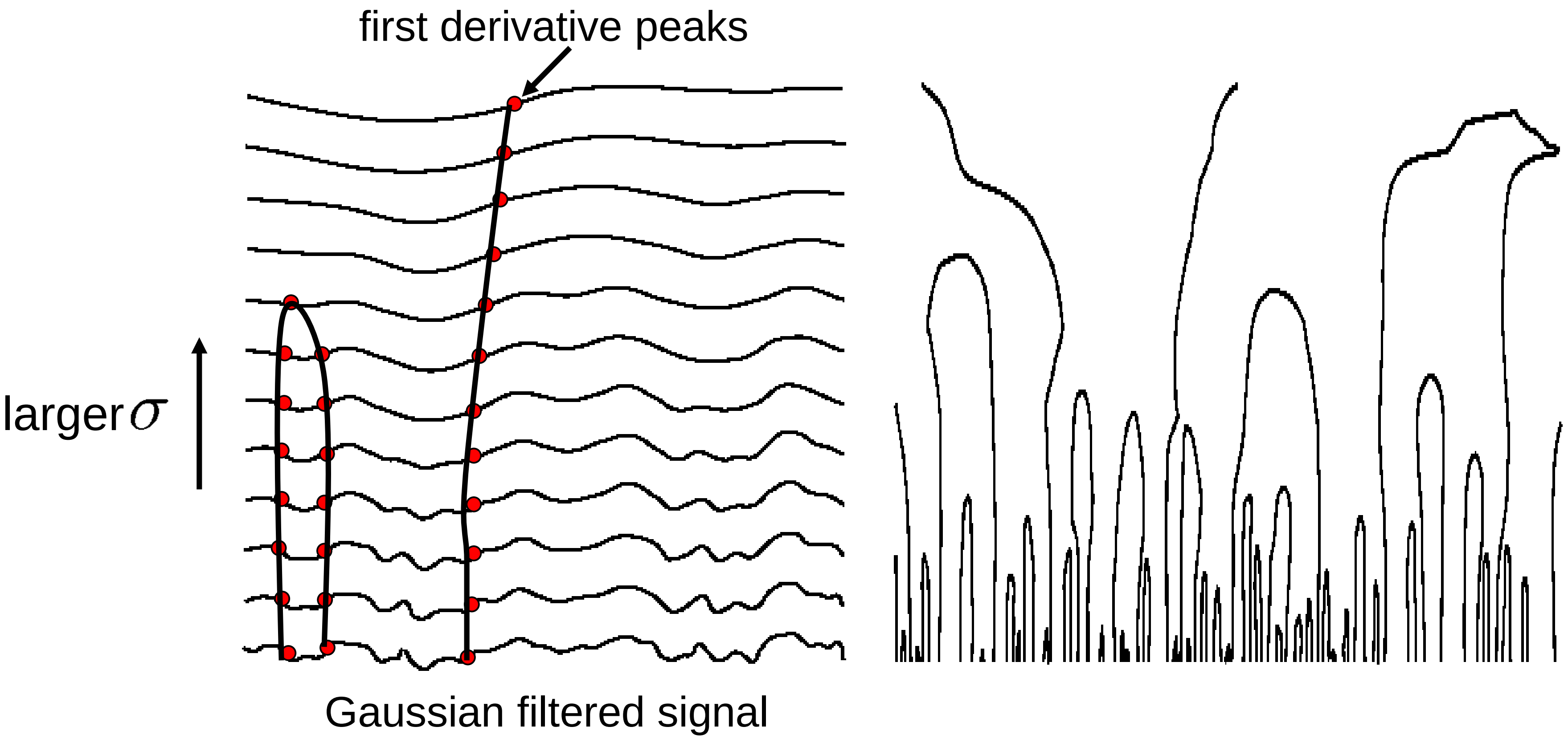


fine scale
high
threshold

coarse
scale,
high
threshold

coarse
scale
low
threshold

Scale space (Witkin 83)



- Properties of scale space (w/ Gaussian smoothing)
  - edge position may shift with increasing scale (σ)
  - two edges may merge with increasing scale
  - an edge may ***not*** split into two with increasing scale

Why is Canny so Dominant

- Still widely used after 20 years.
  1. Theory is nice (but end result same).
  2. Details good (magnitude of gradient).
  3. Hysteresis an important heuristic.
  4. Code was distributed.
  5. Perhaps this is about all you can do with linear filtering.

Difference of Gaussians (DoG)

- Laplacian of Gaussian can be approximated by the difference between two different Gaussians

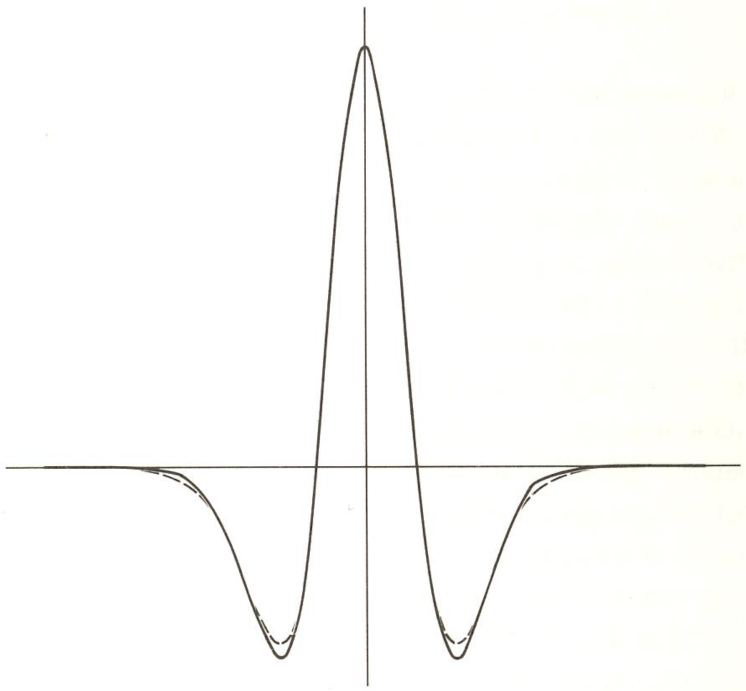


Figure 2–16. The best engineering approximation to $\nabla^2 G$ (shown by the continuous line), obtained by using the difference of two Gaussians (DOG), occurs when the ratio of the inhibitory to excitatory space constraints is about 1:1.6. The DOG is shown here dotted. The two profiles are very similar. (Reprinted by permission from D. Marr and E. Hildreth, "Theory of edge detection, " *Proc. R. Soc. Lond. B 204,* pp. 301–328.)

DoG Edge Detection

(a) $\sigma = 1$

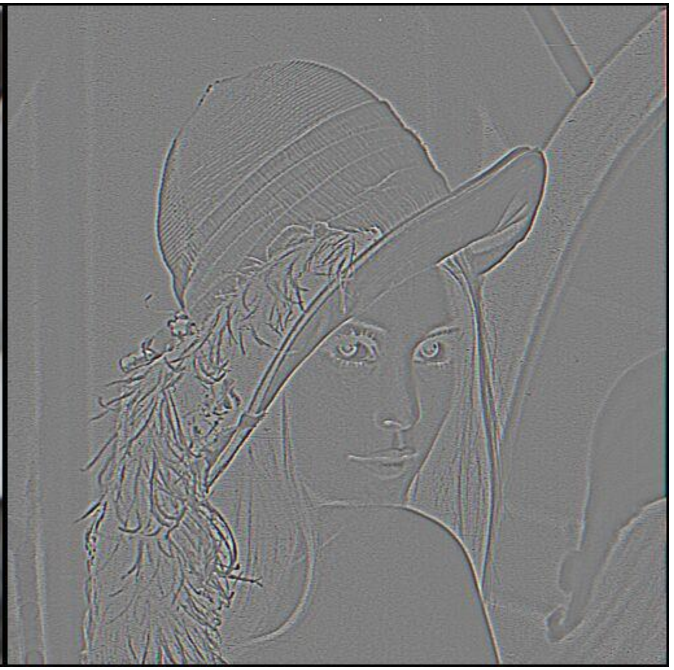
(b) $\sigma = 2$

(b)-(a)

Edge detection by subtraction

original

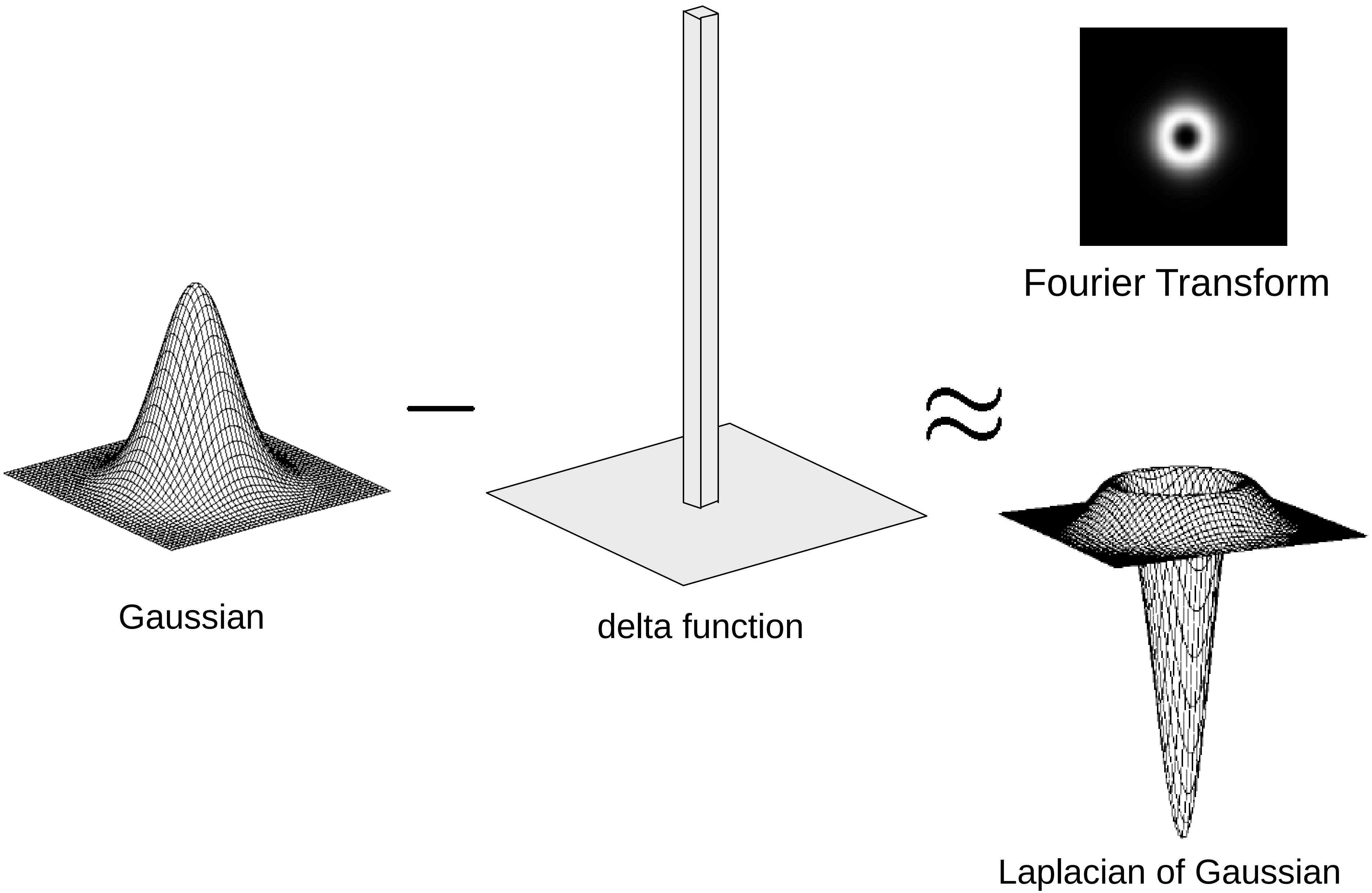
smoothed (5x5 Gaussian)



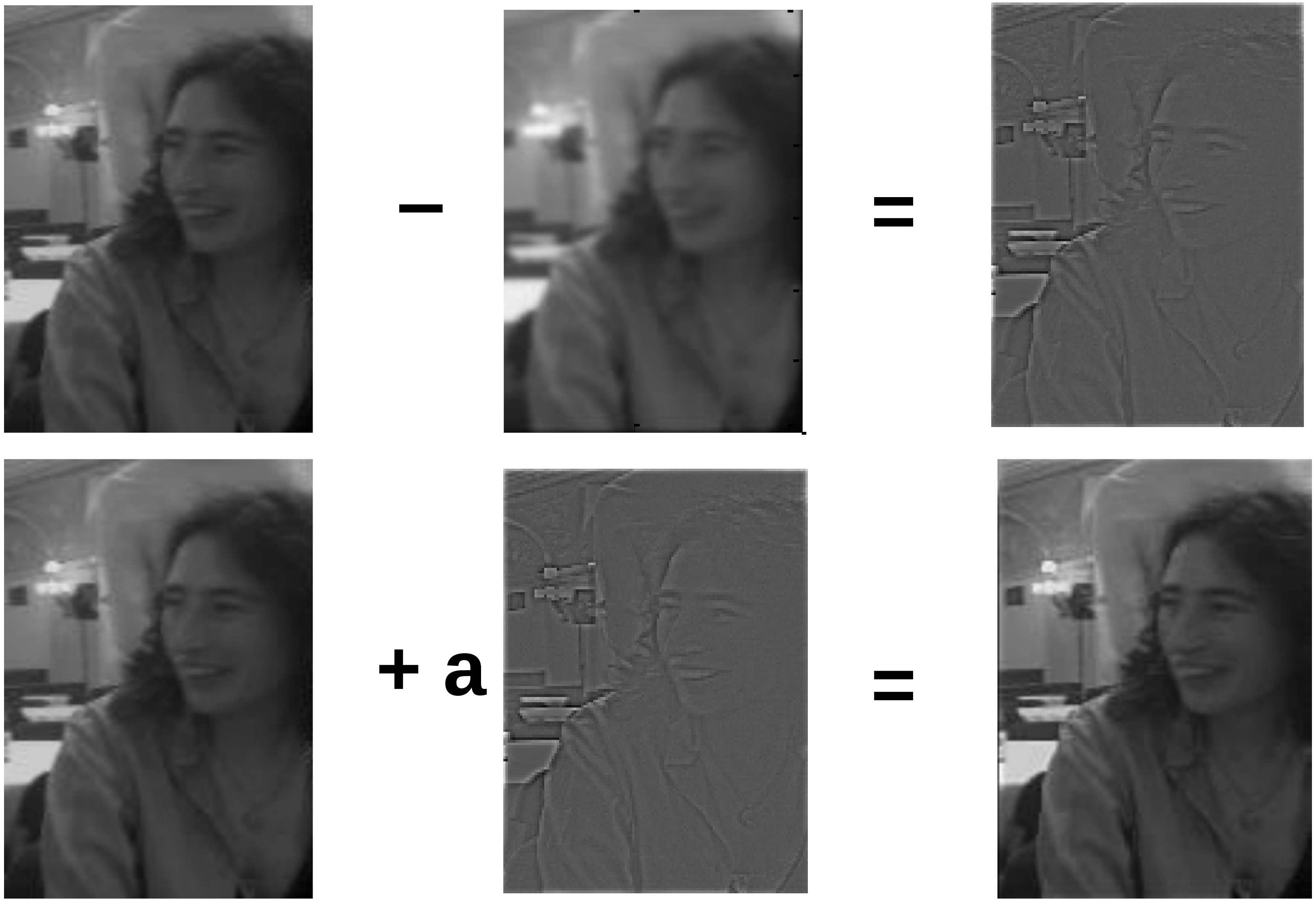
smoothed – original
(scaled by 4, offset +128)

Why does
this work?

Gaussian - image filter



Unsharp Masking



An edge is not a line...

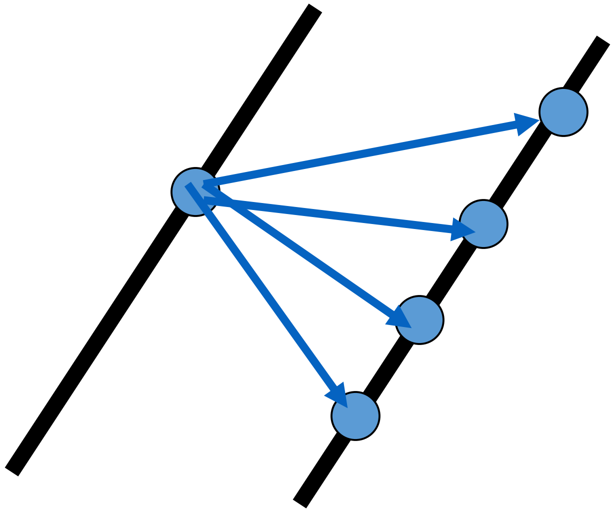
How can we detect ***lines*** ? HoughTransformation

Corners

- Why are they important?

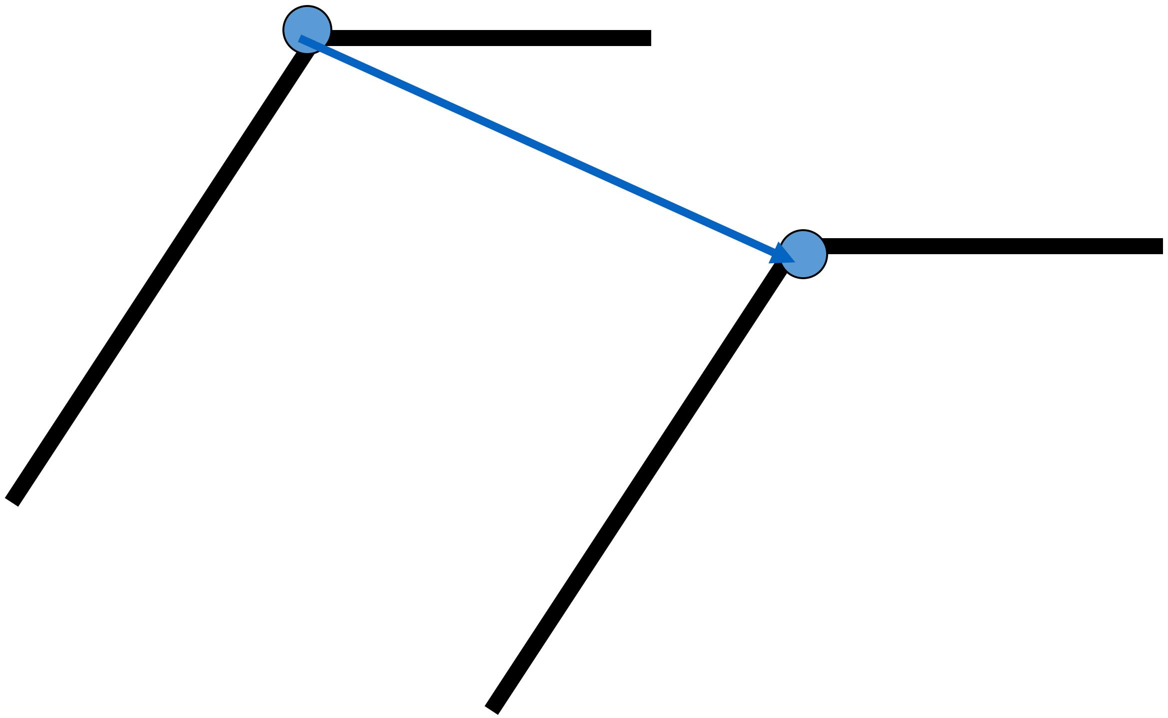
Corners contain more edges than lines

- A point on a line is hard to match.

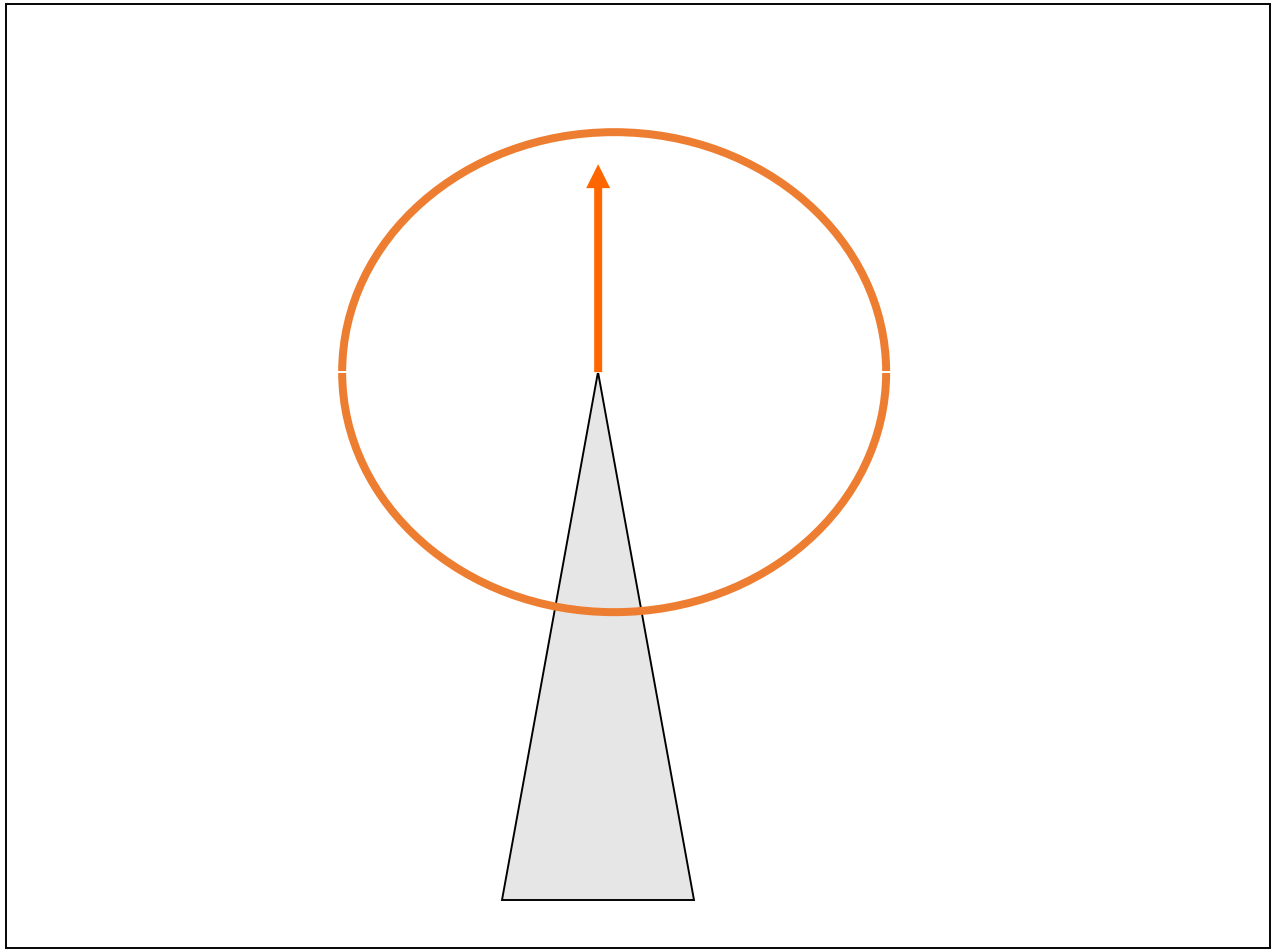


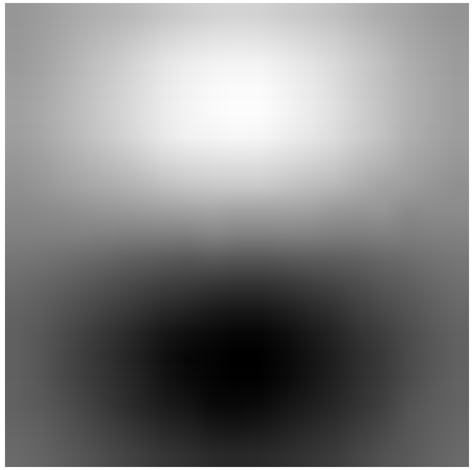
Corners contain more edges than lines

- A corner is easier



Edge Detectors Tend to Fail at Corners





Finding Corners

Intuition:

- Right at corner, gradient is ill defined.
- Near corner, gradient has two different values.

Formula for Finding Corners

We look at matrix:

Sum over a small region, the hypothetical corner

Gradient with respect to x, times gradient with respect to y

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

Matrix is symmetric

WHY THIS?

First, consider case where:

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

This means all gradients in neighborhood are:

(k,0) or (0, c) or (0, 0) (or off-diagonals cancel).

What is region like if:

1. $\lambda 1 = 0$?
2. $\lambda 2 = 0$?
3. $\lambda 1 = 0$ and $\lambda 2 = 0$?
4. $\lambda 1 > 0$ and $\lambda 2 > 0$?

General Case:

From Linear Algebra we haven't talked about it follows that since C is symmetric:

$$C = R^{-1}\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

So every case is like one on last slide.

So, to detect corners

- Filter image.
- Compute magnitude of the gradient everywhere.
- We construct C in a window.
- Use Linear Algebra to find $\lambda 1$ and $\lambda 2$.
- If they are both big, we have a corner.