Edge Detection

Lecture for the course
Introduction to Intelligent Systems
at the University of Groningen

Enrique Alegre
Artificial Vision & Pattern Recognition Group
Dept. Electrical, Automatic and Systems Engineering
University of Leon (Spain)
Erasmus guest lecturer at the RUG

Modified by Nicolai Petkov (RUG)

Credits

Some of the material used in this lecture stems from:

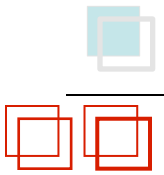
- S. Narasimhan. Computer Vision course. Spring 2006
- Longin Jan Latecki. Sliding Window Filters and Edge Detection. Computer Graphics and Image Processing
- Mircea Nicolescu. Computer Vision course (CS 791E)
- George Bebis. Computer Vision course (CS485/685)
- James Hays. Brown University

Contents

Why is edge detection necessary/useful?

Edges

- contain essential information (e.g. about object shape)
- present a more compact (i.e. sparse) image representation



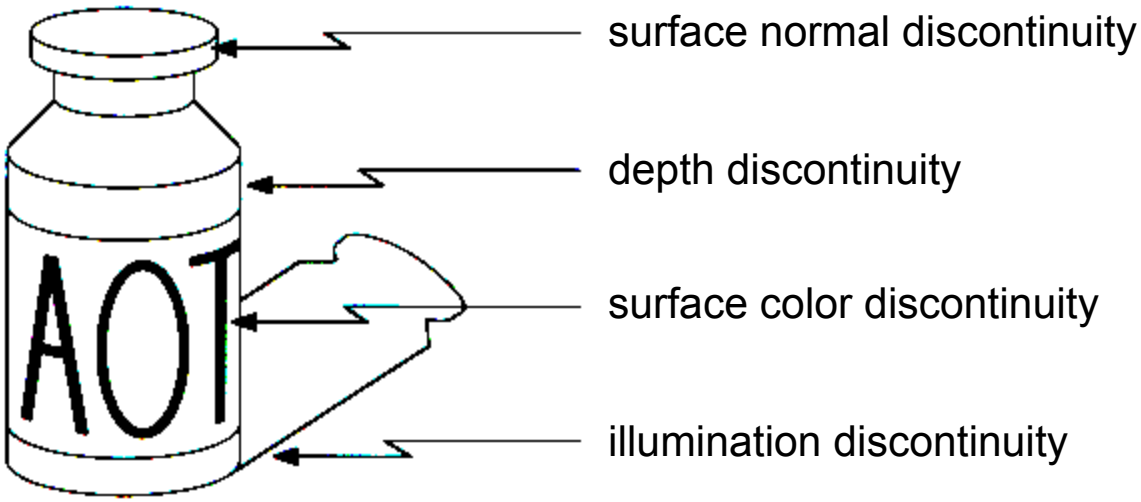
Some observations (not really a definition)

Edge points

- are pixels at or around which the image values (grey level or color) undergo a substantial variation
- are not spike points
- can be part of a boundary between two image regions that differ in luminance, color, texture, depth, surface orientation, movement (in video).
- can be part of a contour of an object (very useful if we are interested in object recognition)

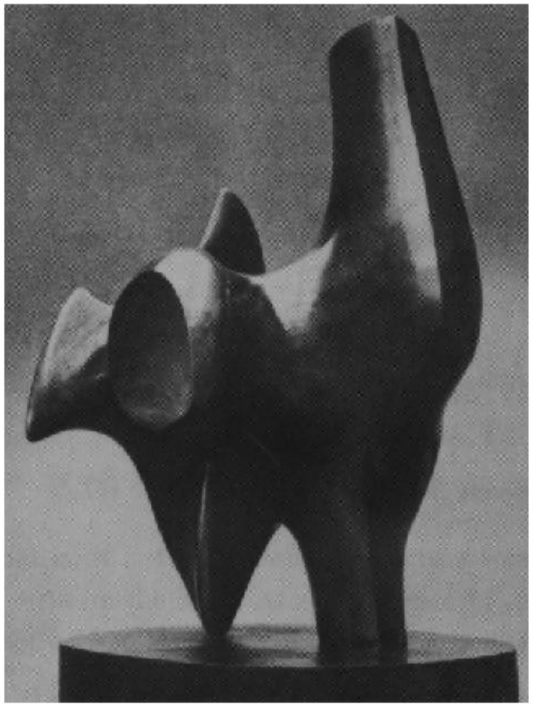
Origins of edges

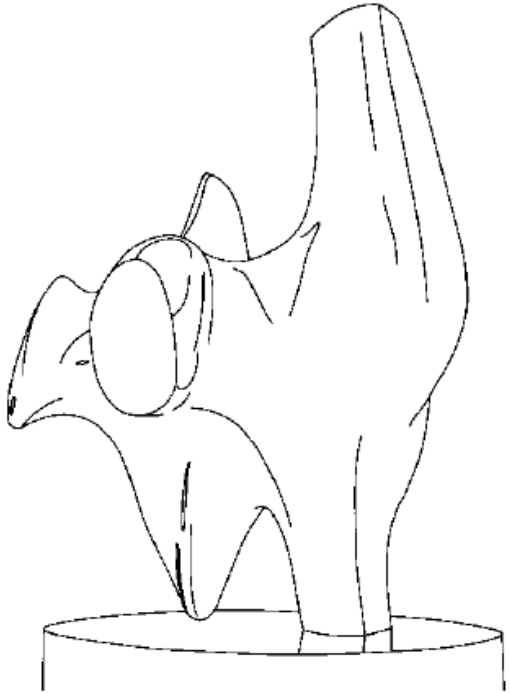
Edges can be caused by a variety of factors



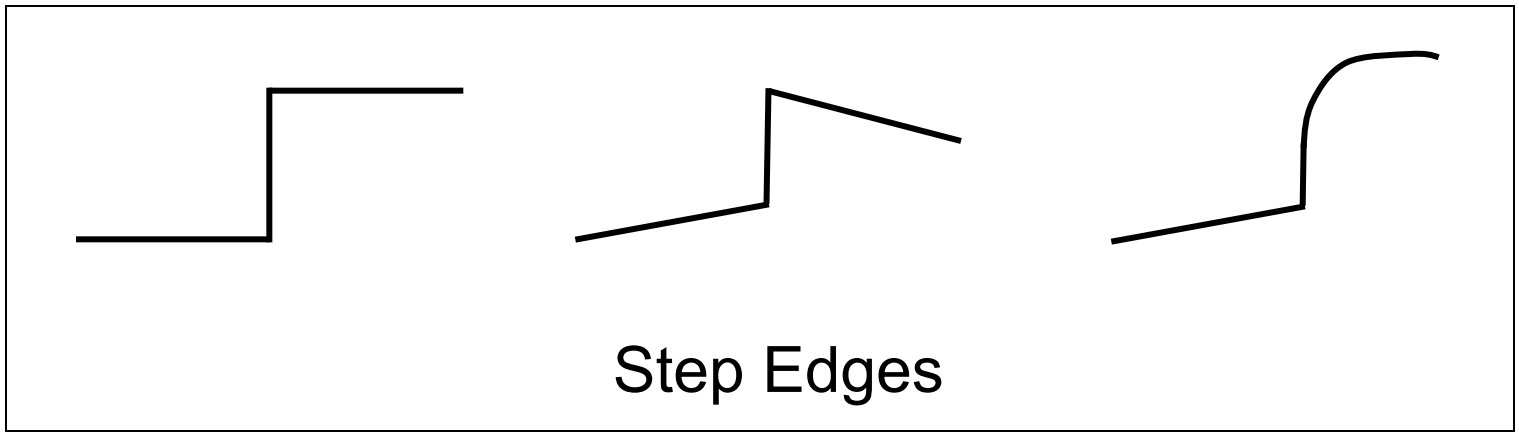
Edge pixel – a kind of a definition

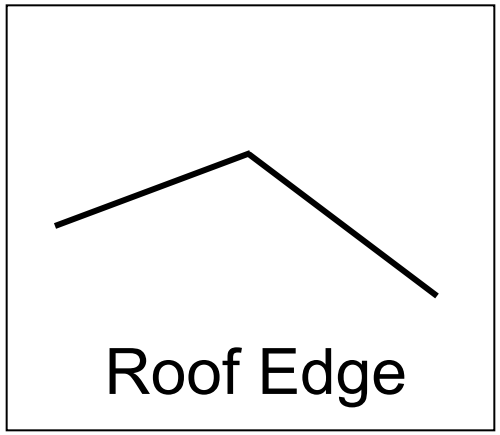
Edge pixel - a pixel that presents a *substantial value of the magnitude of the gradient* of some image attribute (most frequently this is the grey level intensity, but for color images it can be the color)

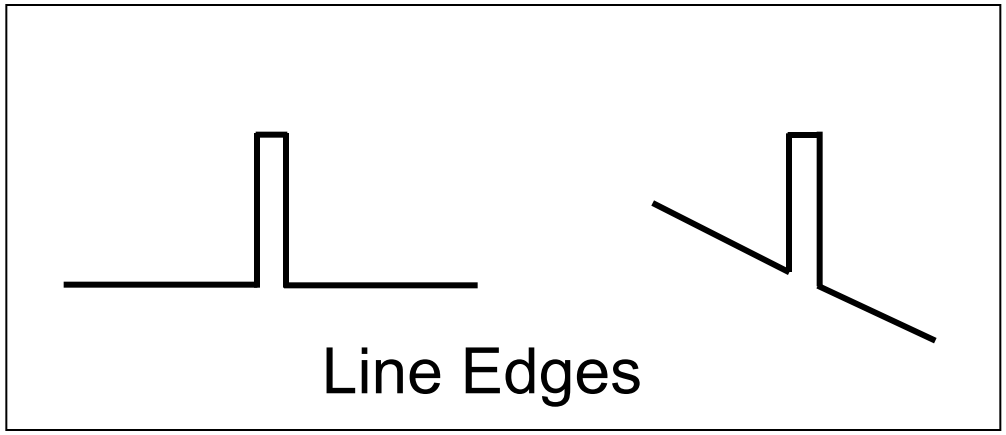




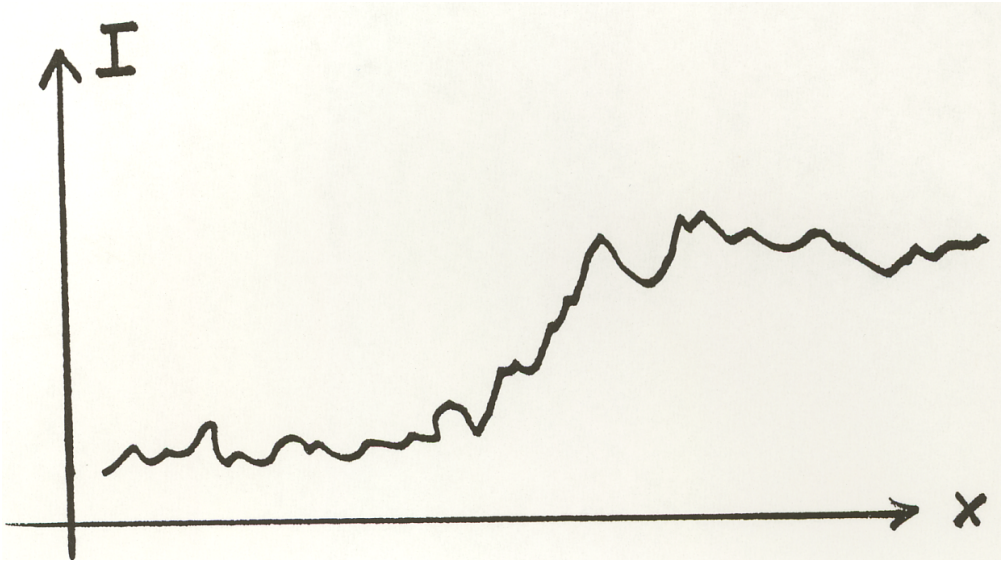
Types of edges







Real edges …

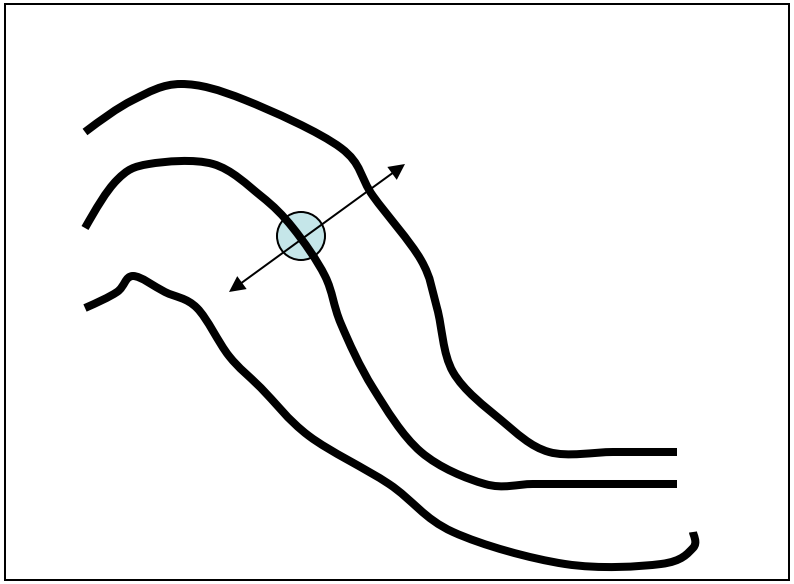


… are noisy and discrete!

2. Image gradient

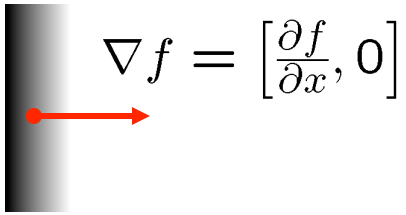
$$\nabla I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$$

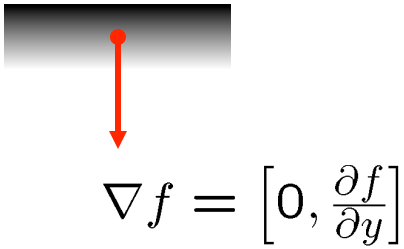
- Its **direction** is the one in which the values of the function change most rapidly.
- Its **magnitude** gives the rate of change.

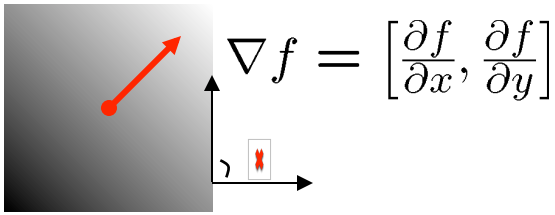


2.1 What the gradient tells us

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







- Gradient direction is the normal to an edge

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

- The gradient magnitude gives the *edge strength*

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

2.2 Gradient of a discrete function

How can we differentiate a *digital* (i.e. discrete) image f[x,y]?

Answer: **Approximate the derivatives by finite differences:**

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

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

3. Linear filters & Discrete convolution

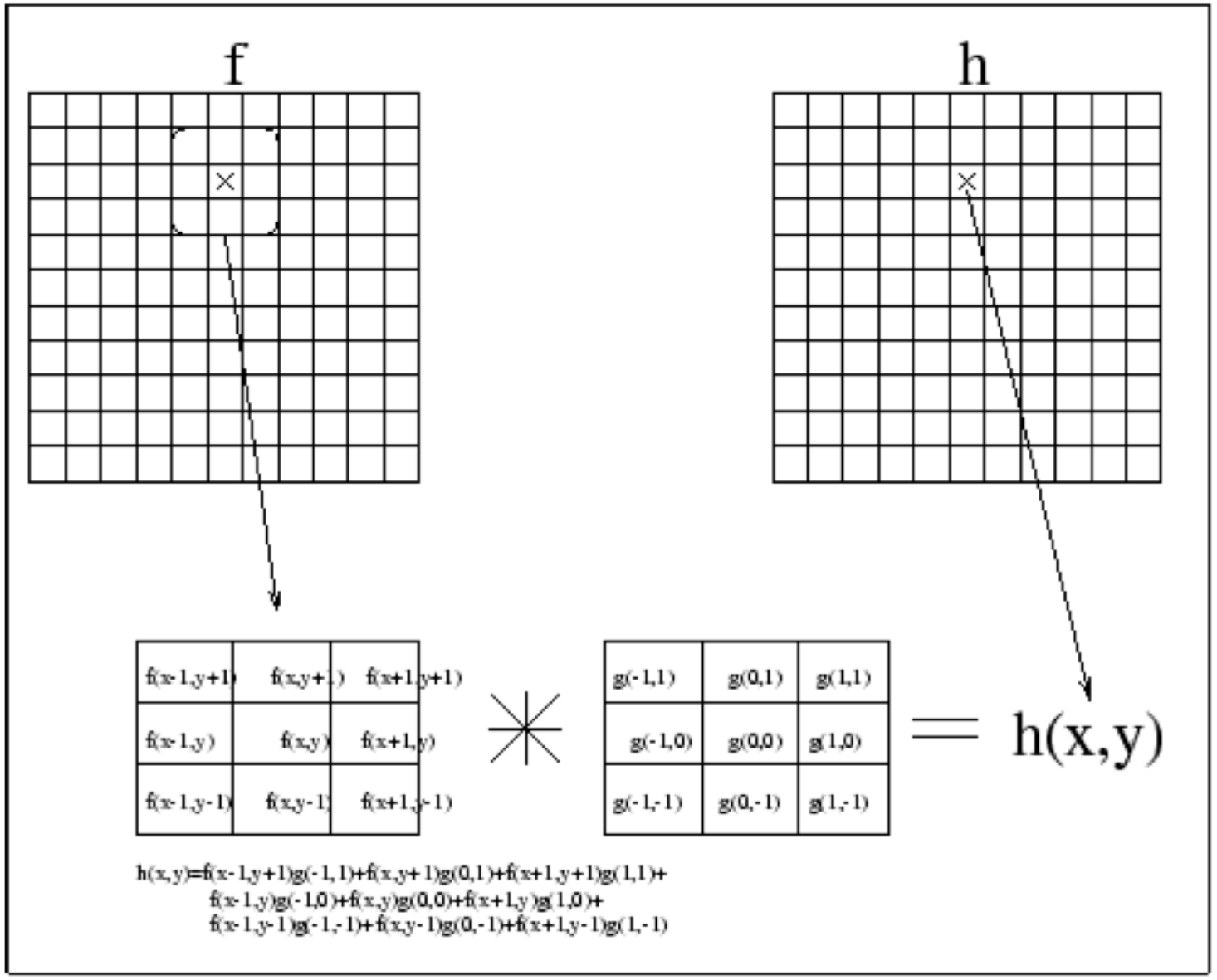
Convolution

Linear operations calculate a resulting value in an output image pixel f(i,j) as a linear combination of brightness in a local neighborhood of the pixel h(i,j) in the input image. **Discrete convolution**:

$$f(i,j) = w * h = \sum_{m=-a}^{a} \sum_{n=-b}^{b} w(m,n) h(i+m, j+n)$$

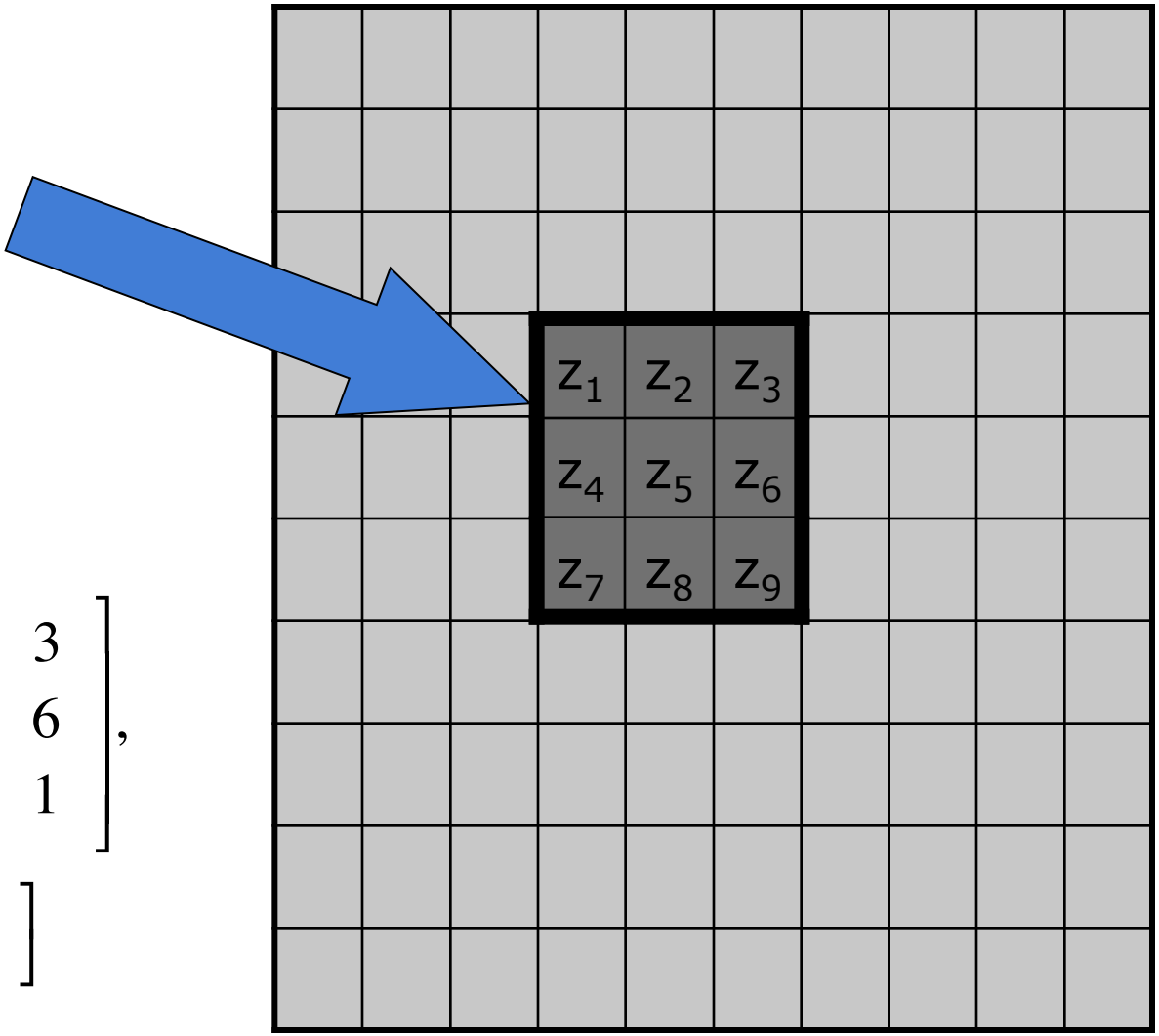
The function w is called a **convolution kernel** or a **filter mask**.
In our case it is a rectangle of size (2a+1)x(2b+1).

… 3. Linear filters & Discrete convolution



… 3. Linear filters & Discrete convolution

For each pixel the output is calculated using a nxn neighborhood around this pixel in the input.



Example

$$h(i,j) = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 3 & 4 & 1 \end{bmatrix},$$

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

… 3. Linear filters & discrete convolution

What happens near edges of signal (image)?

Possible solutions:

- **Pad with zeros** (edges get dark)
- **Replicate edge samples**
- Wrap around

$$h = f * w = \sum_i f_i \, w_{x-i}$$

Computational complexity: If ***f*** has *n* samples and ***w*** has *m* nonzero samples, straightforward *computation* takes *time* **O(*nm*)**

OK for small filter kernels, bad for large ones

… 3. Linear filters & Discrete convolution

Exercise: Compute the 2-D linear convolution of the following two signal ***h*** with mask ***w.***
Extend the signal ***h*** with 0's where needed.

$$h = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 3 & 4 & 1 \end{bmatrix},$$

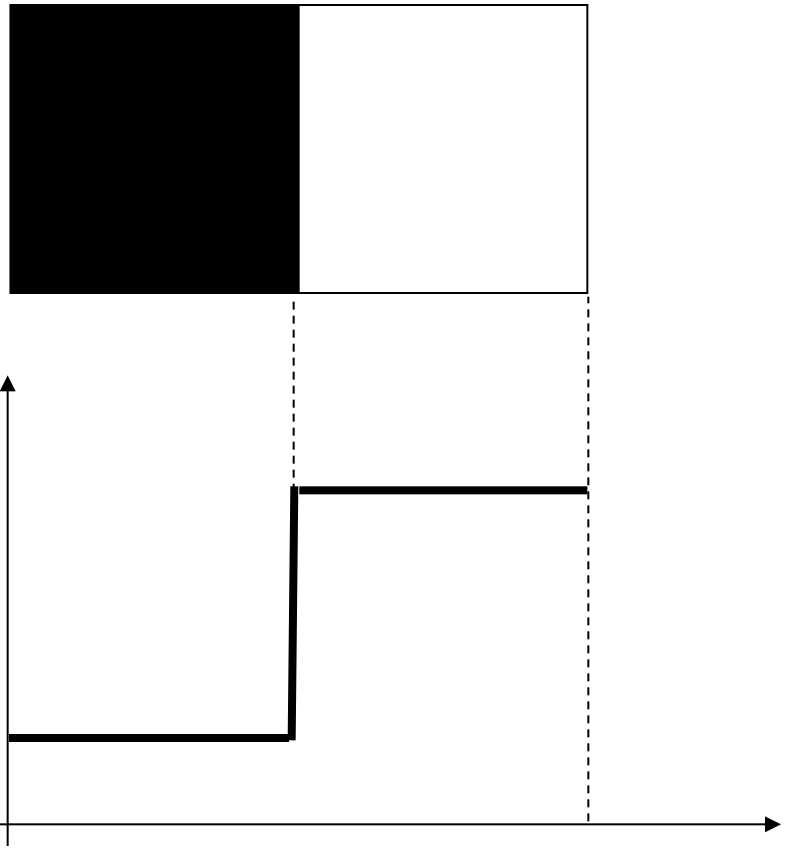
$$w = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

4. Detecting an edge

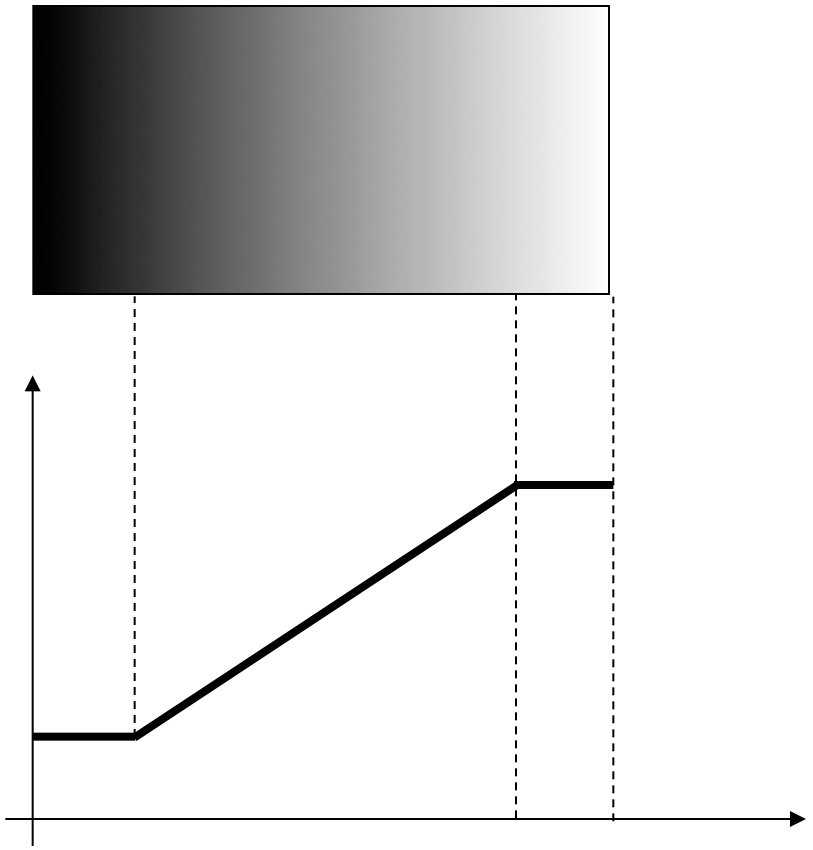
Gray-Level Transition

Ideal

Step

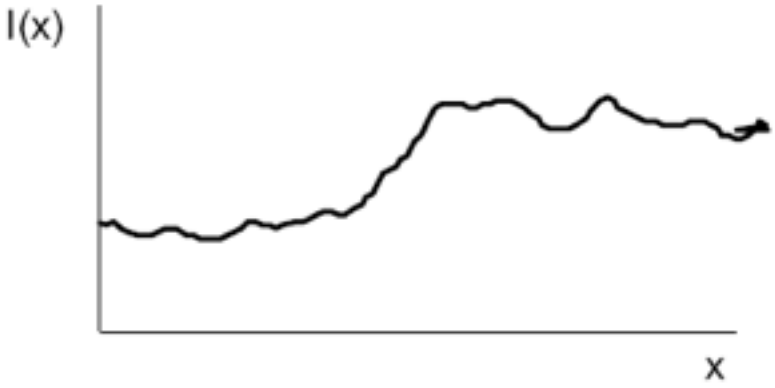


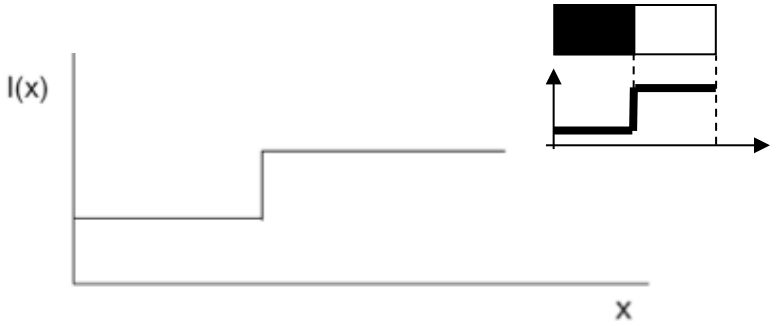
Ramp

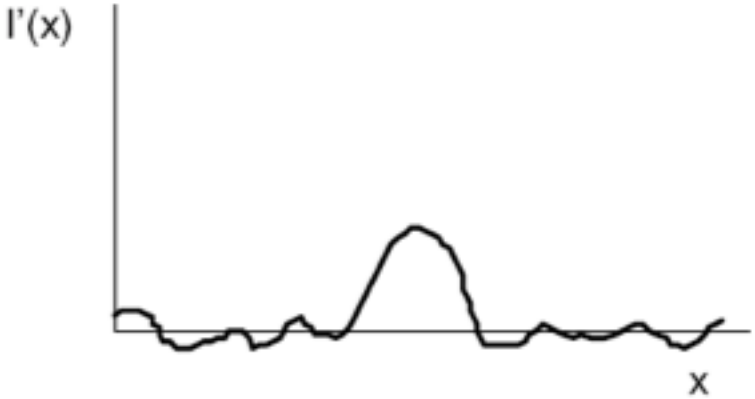


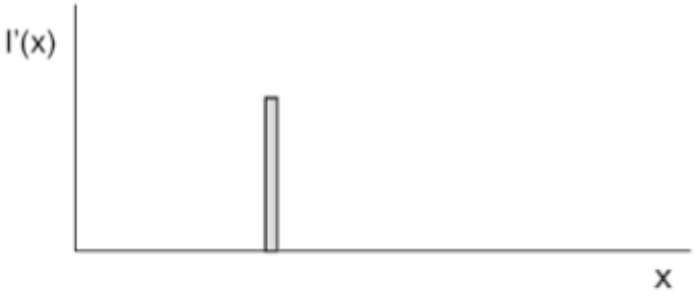
… 4. Detecting an edge

More realistically, due to blurring and noise, we generally have



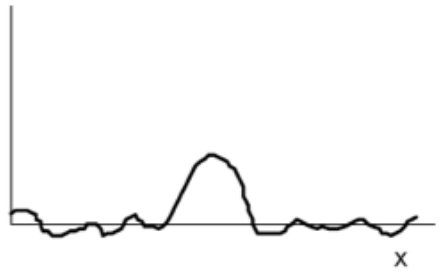




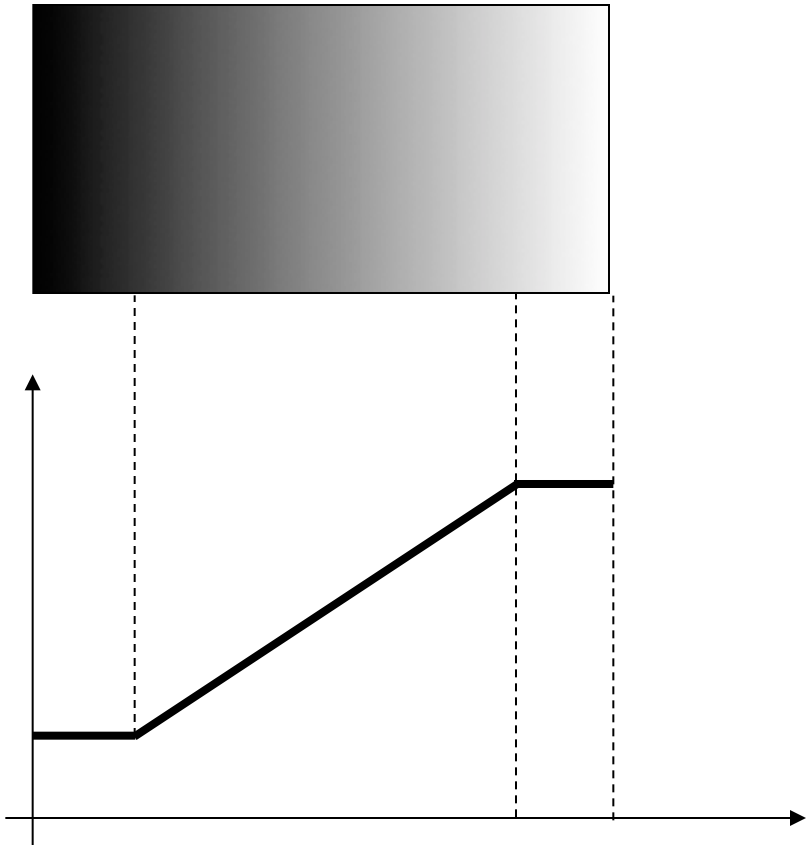


… 4. Detecting an edge

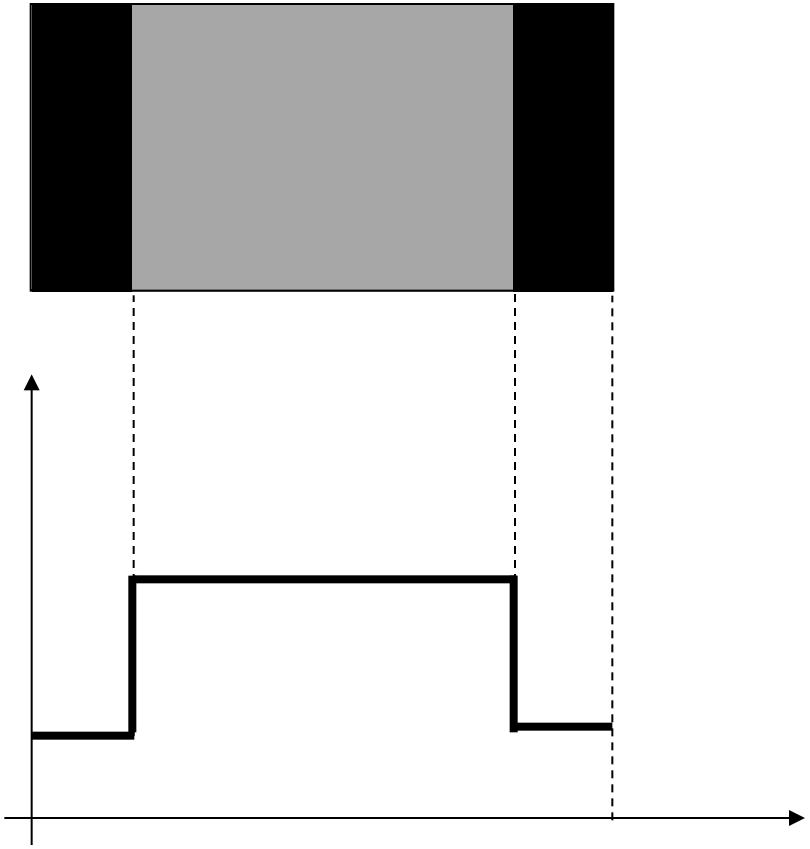
First Derivative



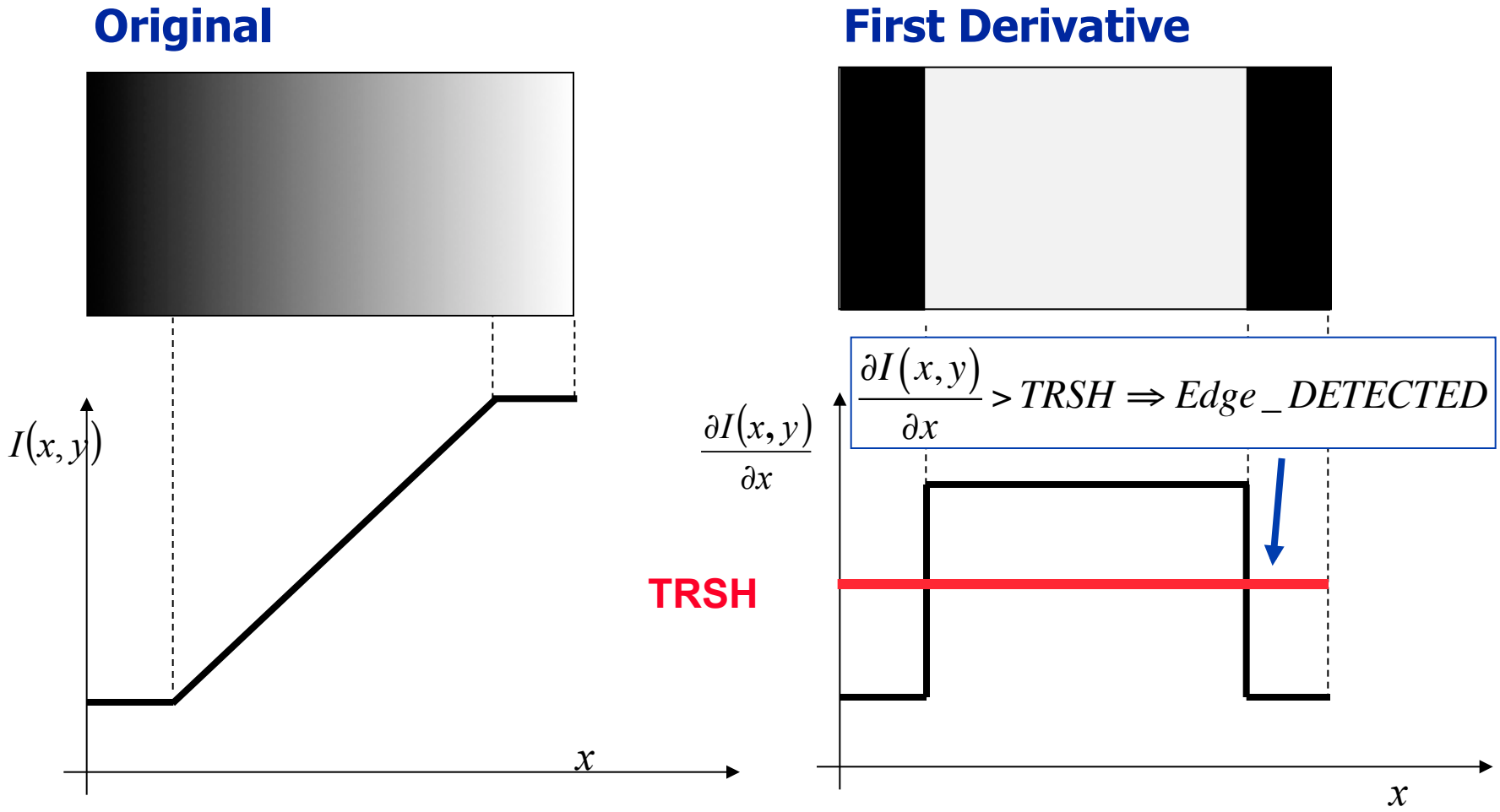
Original



First Derivative



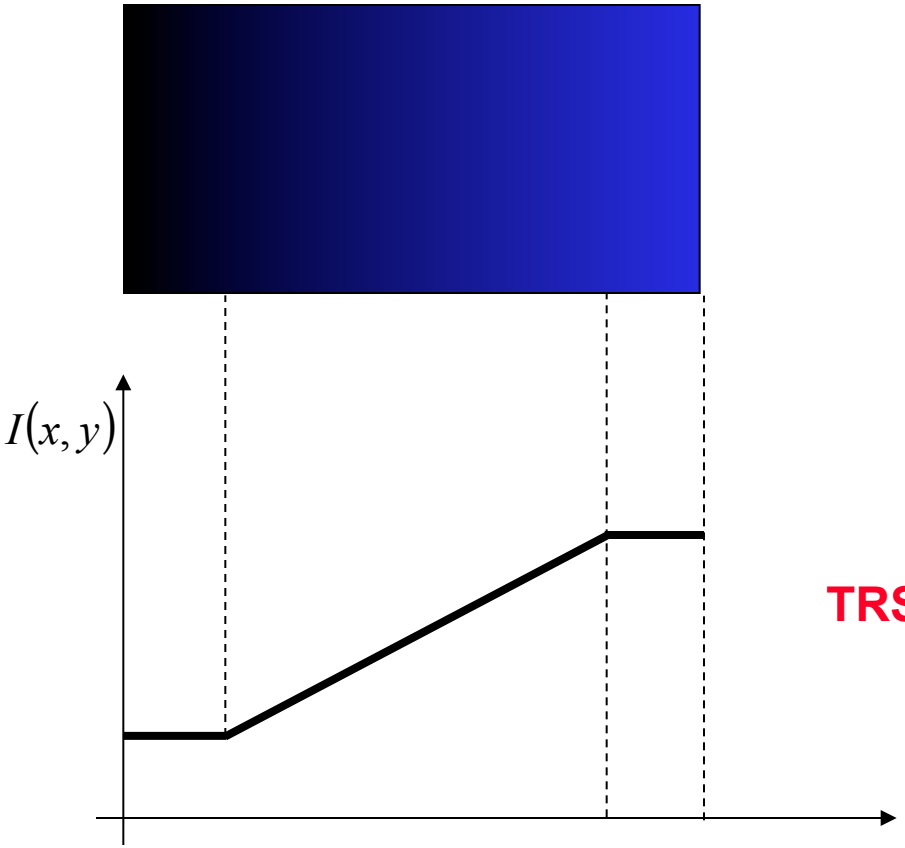
… 4. Detecting an edge

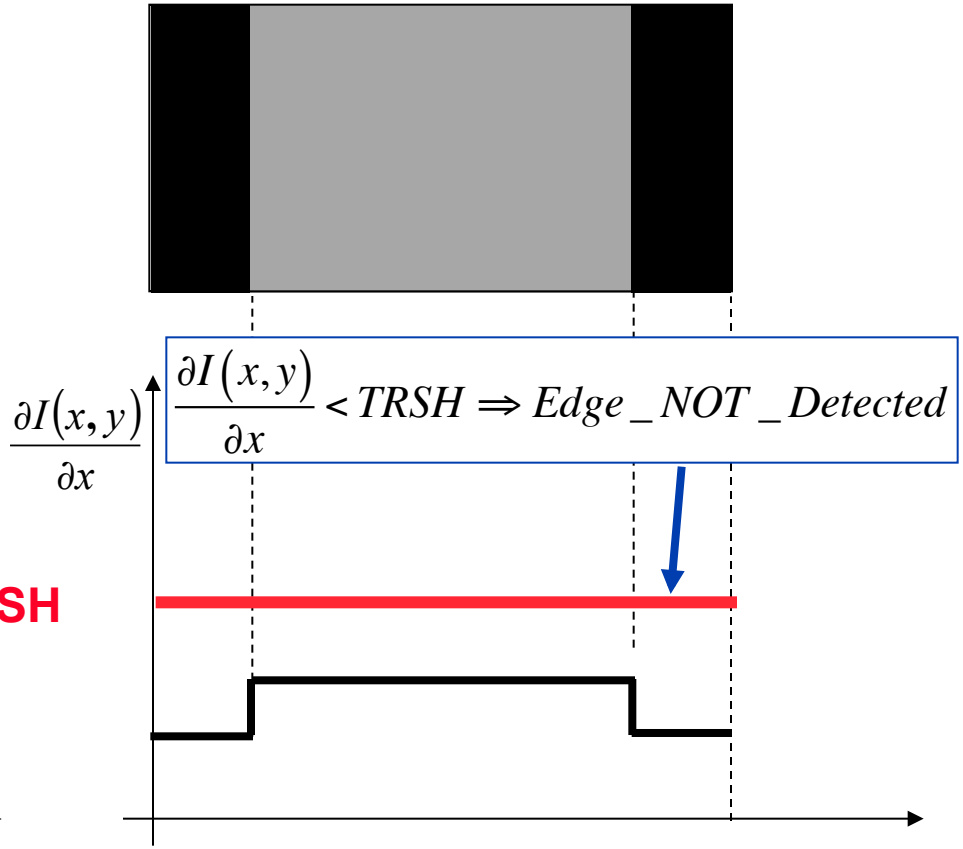


… 4. Detecting an edge

Original

First Derivative





5. Gradient –derivative- operators

Gradient $(\nabla \mathbf{f})$

The gradient is a vector that points in the direction of the greatest rate of increase.

In simple terms, the variation in space of any quantity can be represented (e.g. graphically) by a slope.

The gradient represents the steepness and direction of that slope.

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$\nabla f = mag(\nabla \mathbf{f}) = \left[G_x^2 + G_y^2\right]^{\frac{1}{2}} = \sqrt{\frac{\partial f^2}{\partial x} + \frac{\partial f^2}{\partial y}}$$

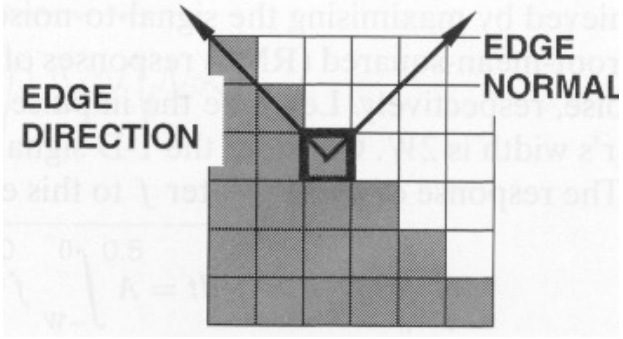
$$direction(\nabla \mathbf{f}) = \tan^{-1}\left(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right)$$

Or, the approximation: $\nabla f \approx |G_x| + |G_y| = \left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right|$

Magnitude: indicates edge strength

Direction: indicates edge direction

(i.e., perpendicular to edge direction)



5. Gradient-like operators

5.1 The Roberts edge detector

0	0	0
0	-1	0
0	0	1

0	0	0
0	0	-1
0	1	0

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

$$\Delta_{x-y} f = G_{x-y} \approx z_9 - z_5$$

$$\Delta_{x+y} f = G_{x+y} \approx z_8 - z_6$$

- Mark edge point only
- No information about edge orientation
- Primary disadvantage:
  - High sensitivity to noise
  - Few pixels are used to approximate the gradient

…5. Gradient operators

5.3 Prewitt edge detector

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

$$G_x \approx (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \qquad G_y \approx (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

…5. Gradient operators

5.4 Sobel Edge Detector

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

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

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

$$G_x \approx (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

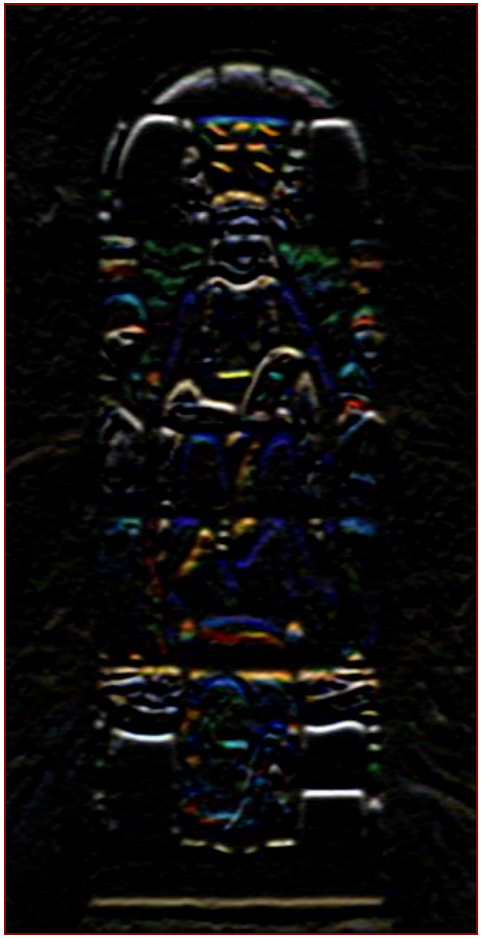
$$G_y \approx (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

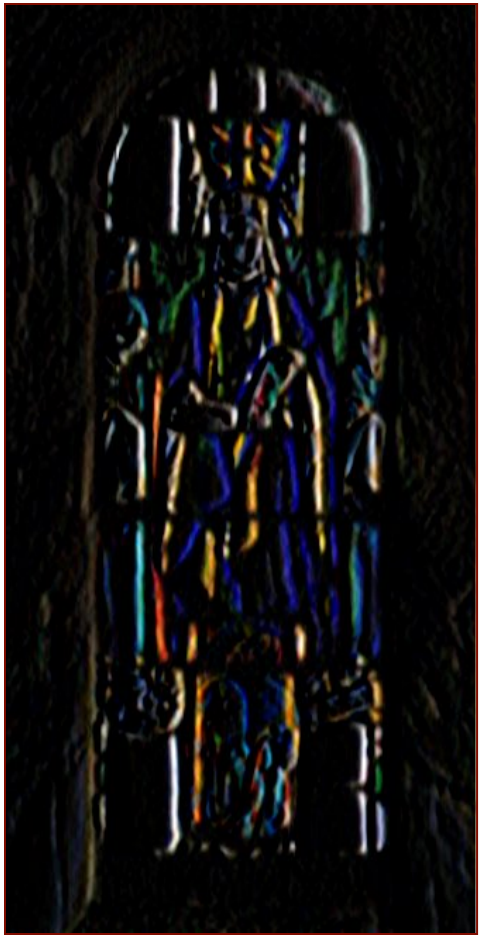
…5. Gradient operators

… Prewitt

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1



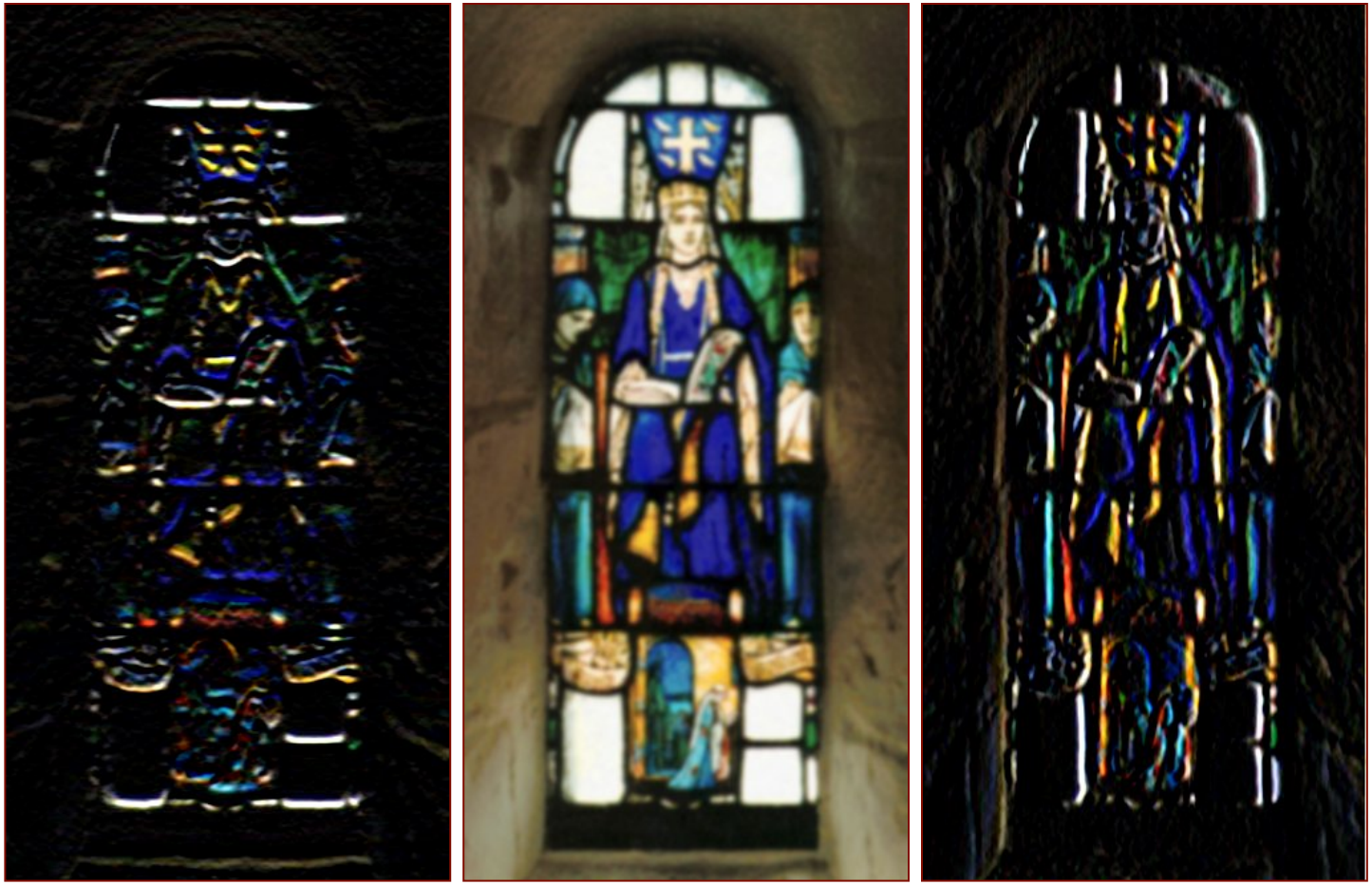


…5. Gradient operators

… Sobel

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

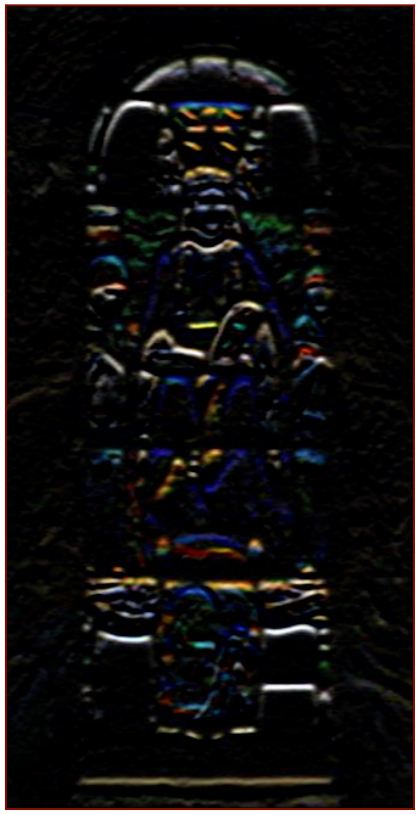
-1	0	1
-2	0	2
-1	0	1



…5. Gradient operators

… Prewitt vs Sobel

-1	-1	-1
0	0	0
1	1	1



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

-1	0	1
-1	0	1
-1	0	1

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

…5. Gradient-like operators

5.4 Robinson Compass Masks

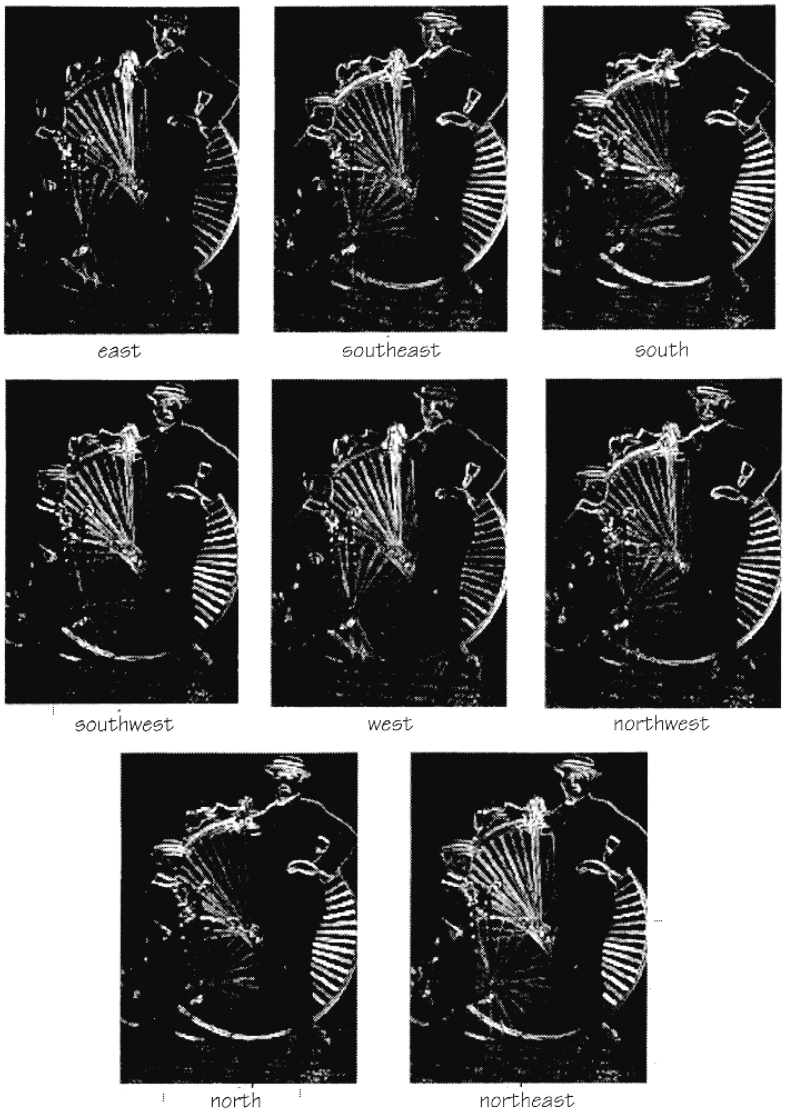
- Taking a single mask and rotating it to 8 major compass orientations: N, NW, W, SW, S, SE, E, and NE.
- The edge magnitude = The maximum value found by the convolution of each mask with the image.
- The edge direction is defined by the mask that produces the maximum magnitude.

$$E = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad NE = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \quad N = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad NW = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

$$W = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad SW = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix} \quad S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad SE = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

…5. Gradient-like operators

Robinson compass masks



…5. Gradient-like operators

5.5 Kirsch Compass

- Similar to the Robinson's but with different coefficients

$$E=\begin{bmatrix}-3 & -3 & 5\\ -3 & 0 & 5\\ -3 & -3 & 5\end{bmatrix}\quad NE=\begin{bmatrix}-3 & 5 & 5\\ -3 & 0 & 5\\ -3 & -3 & -3\end{bmatrix}\quad N=\begin{bmatrix}5 & 5 & 5\\ -3 & 0 & -3\\ -3 & -3 & -3\end{bmatrix}\quad NW=\begin{bmatrix}5 & 5 & -3\\ 5 & 0 & -3\\ -3 & -3 & -3\end{bmatrix}$$

$$W=\begin{bmatrix}5 & -3 & -3\\ 5 & 0 & -3\\ 5 & -3 & -3\end{bmatrix}\quad SW=\begin{bmatrix}-3 & -3 & -3\\ 5 & 0 & -3\\ 5 & 5 & -3\end{bmatrix}\quad S=\begin{bmatrix}-3 & -3 & -3\\ -3 & 0 & -3\\ 5 & 5 & 5\end{bmatrix}\quad SE=\begin{bmatrix}-3 & -3 & 5\\ -3 & 0 & 5\\ -3 & 5 & 5\end{bmatrix}$$

…5. Differential operators

5.6 Laplacian edge detector

Sometimes we are interested only in edge magnitudes without regard to their orientations.

- The **Laplacian** may be used.

- The Laplacian has the same properties in all directions and is therefore invariant to rotation in the image.

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

The Laplace operator is a very popular operator approximating the second derivative which gives the gradient magnitude only.

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8)$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

…5. Differential operators

Masks for 4 and 8 neighborhoods

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Mask with stressed significance of the central pixel or its neighborhood

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \qquad \begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix}$$

…5. Differential operators

5.7 Comparing operators

Gradient:

$$\nabla f \approx |G_x| + |G_y| = \left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right|$$

Good Localization
Noise Sensitive
Poor Detection

Roberts (2 x 2):

0	1
-1	0

1	0
0	-1

Sobel (3 x 3):

-1	0	1
-1	0	1
-1	0	1

1	1	1
0	0	0
-1	-1	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

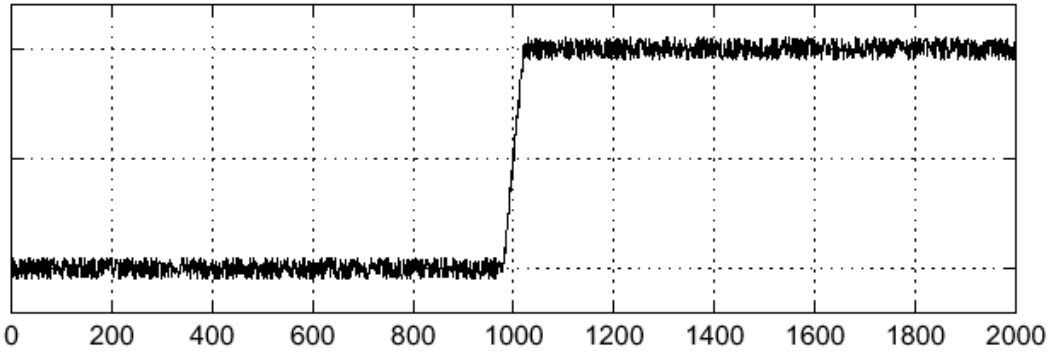
1	2	3	2	1
2	3	5	3	2
0	0	0	0	0
-2	-3	-5	-3	-2
-1	-2	-3	-2	-1

Poor Localization
Less Noise Sensitive
Good Detection

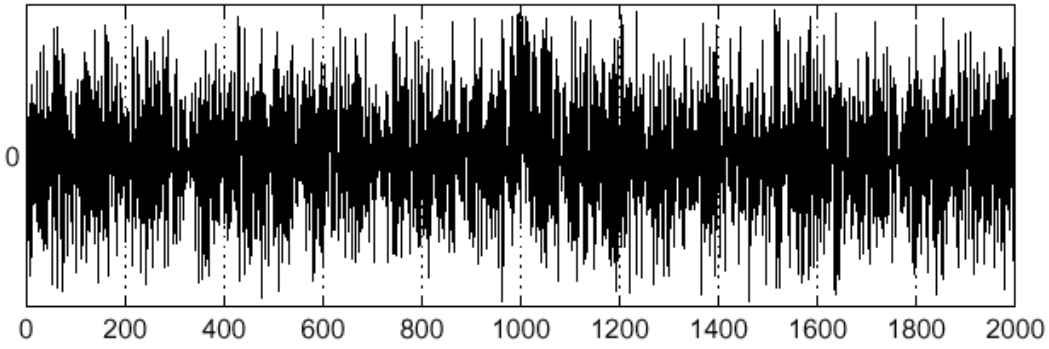
6. Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal



$$\frac{d}{dx} f(x)$$



Where is the edge?

6. Effects of noise

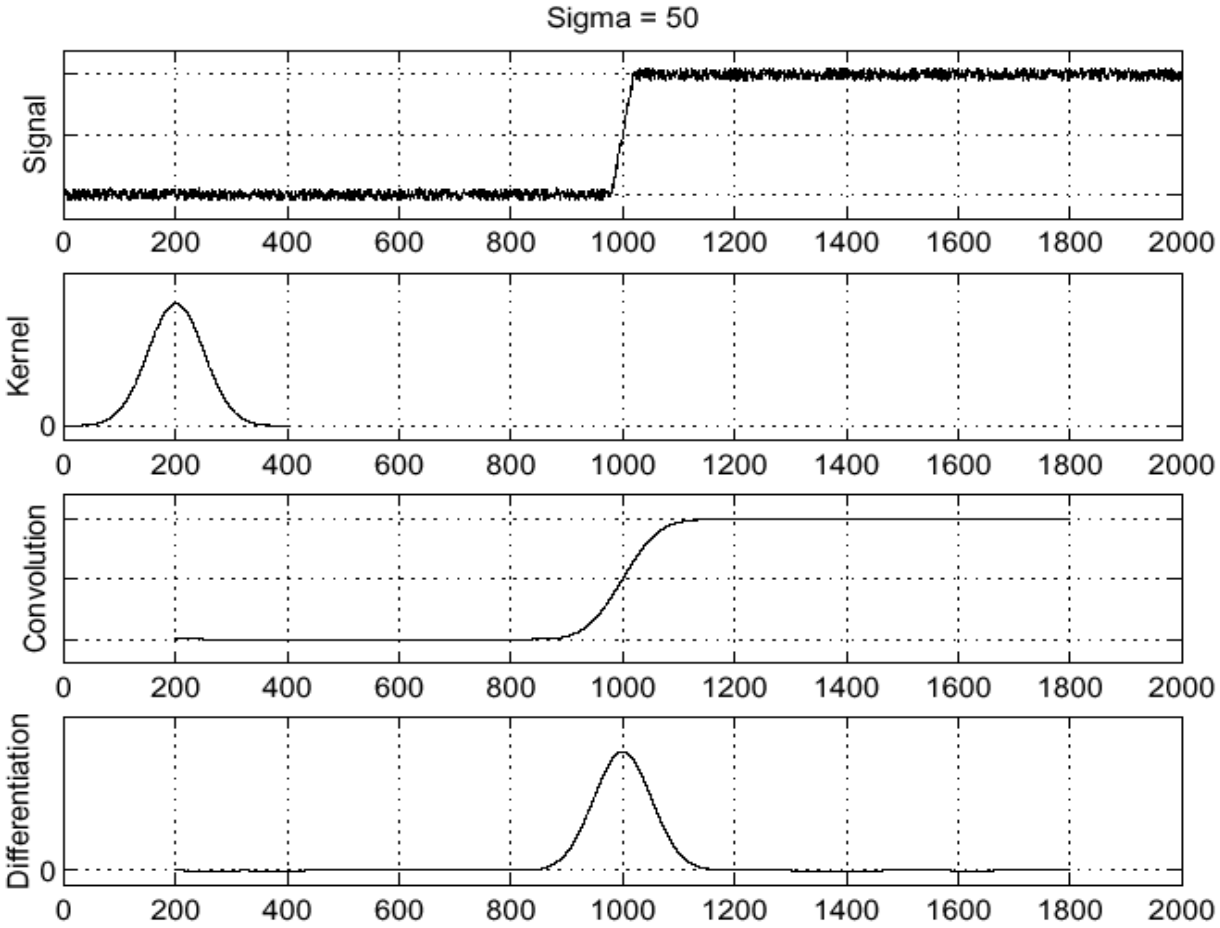
-> Smooth First

f

h

$(h * f)$

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



Where is the edge?

Look for peaks in $\frac{\partial}{\partial x}(h * f)$

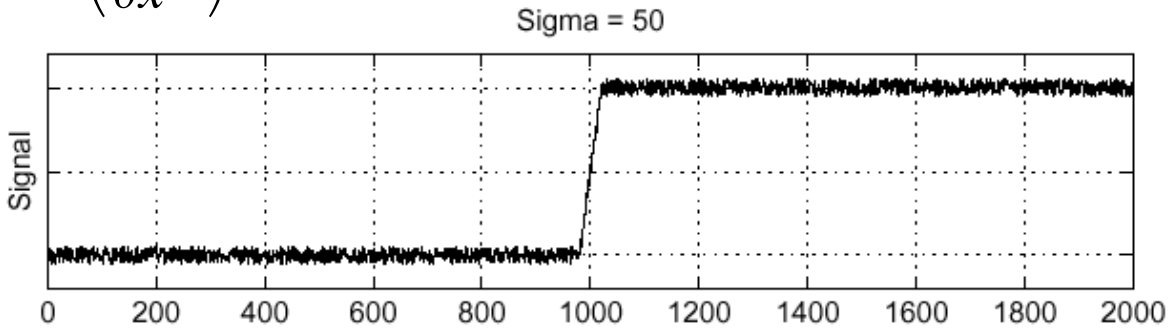
6. Effects of noise

6.2 Derivative theorem for Convolution

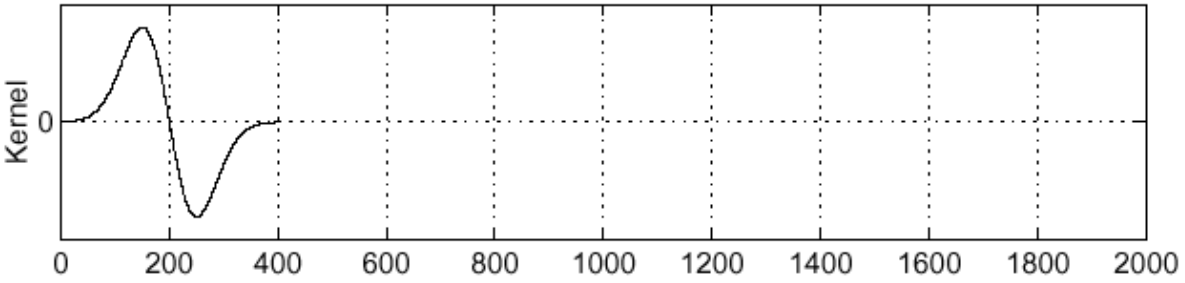
$$\frac{\partial}{\partial x}(h * f) = \left(\frac{\partial}{\partial x} h\right) * f$$

…saves us one operation.

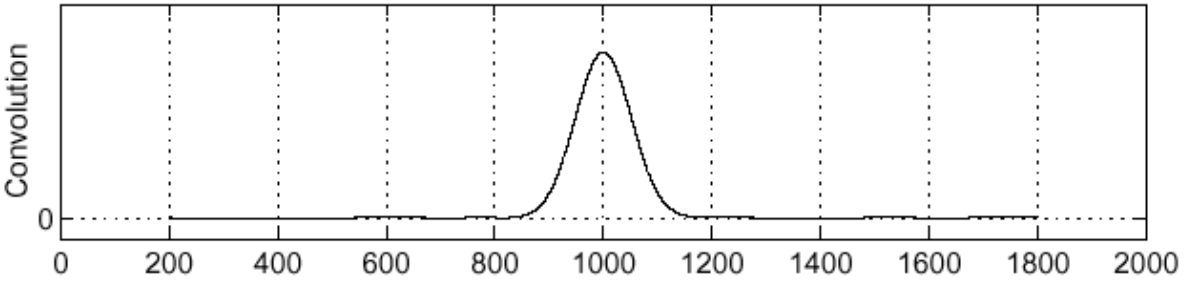
f



$\left(\frac{\partial}{\partial x} h\right)$

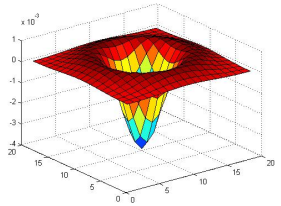


$\left(\frac{\partial}{\partial x} h\right) * f$



…6. Zero-crossing operators

6.3 Laplacian of Gaussian

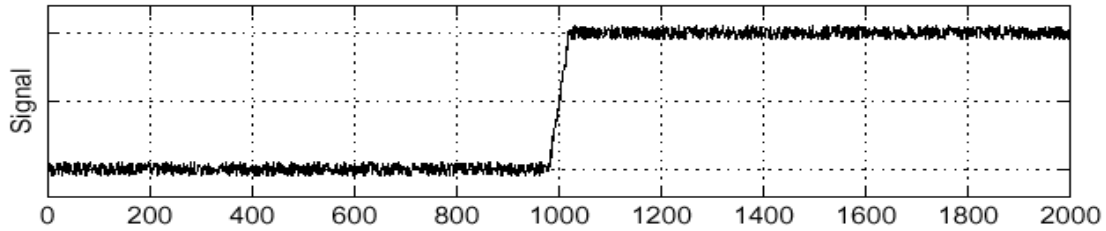


$$\frac{\partial^2}{\partial x^2}(h * f) = \left(\frac{\partial^2}{\partial x^2} h\right) * f$$

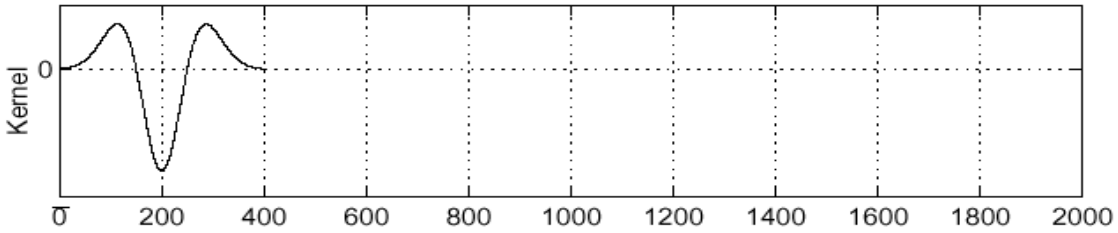
Laplacian of Gaussian

Sigma = 50

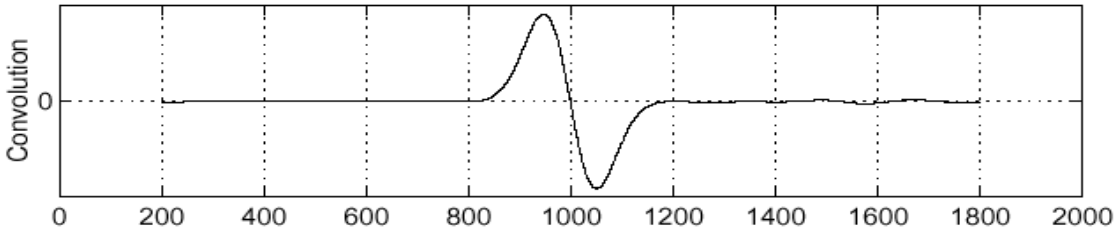
f



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



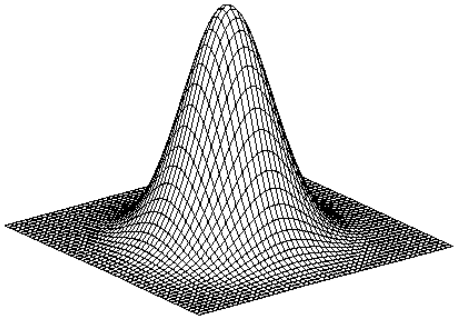
$\left(\frac{\partial^2}{\partial x^2} h\right) * f$



Where is the edge?

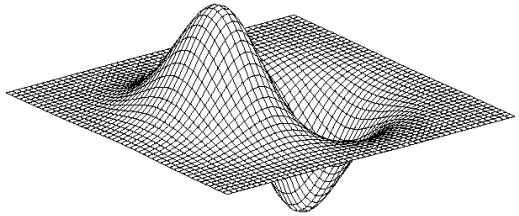
Zero-crossings of bottom graph !

6. Gaussian differential operators



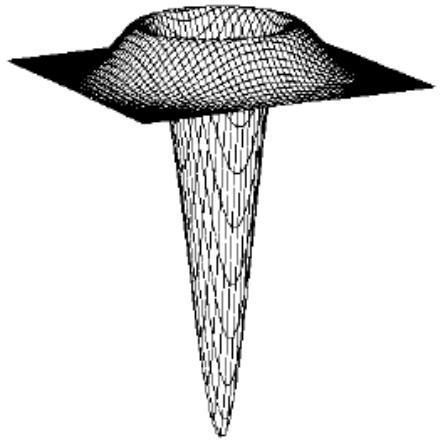
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

Gaussian



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

Derivative of Gaussian



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

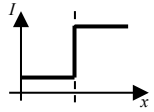
Laplacian of Gaussian

Mexican Hat (Sombrero)

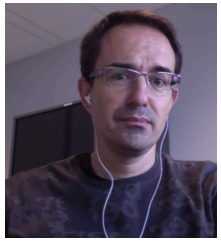
Where ∇^2 is the **Laplacian** operator:

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

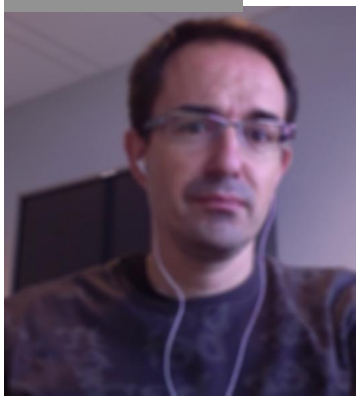
…6. Zero-crossing operators

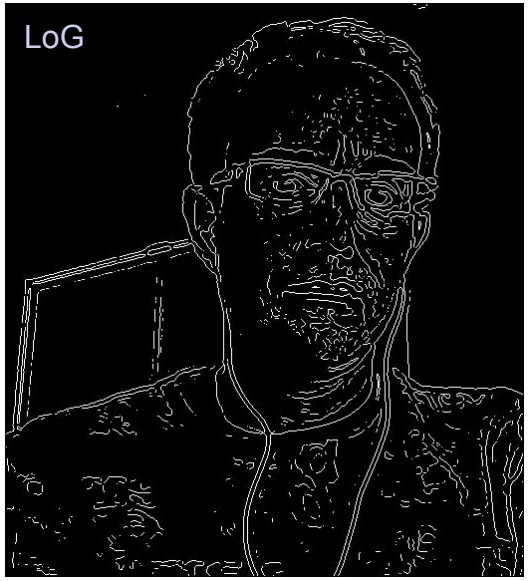


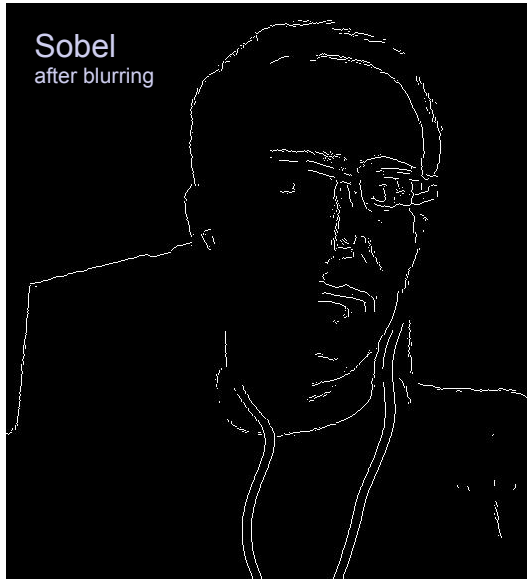
Examples of Edge Detection

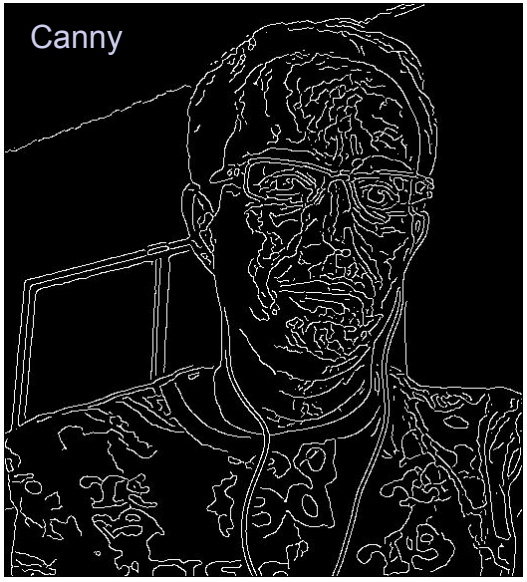


Gaussian blur





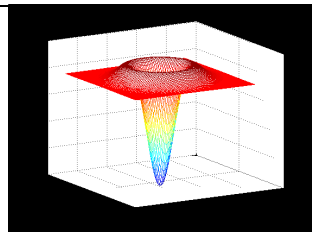




Canny ??

7. Edge Detectors

7.1. Marr-Hildreth edge detector LoG



The Marr-Hildreth Edge Detector was an early gradient-based edge detector.

It considers edges as zero-crossings in the second derivative of an image.

5x5 LoG mask

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

Algorithm for the Marr-Hildreth edge detector:

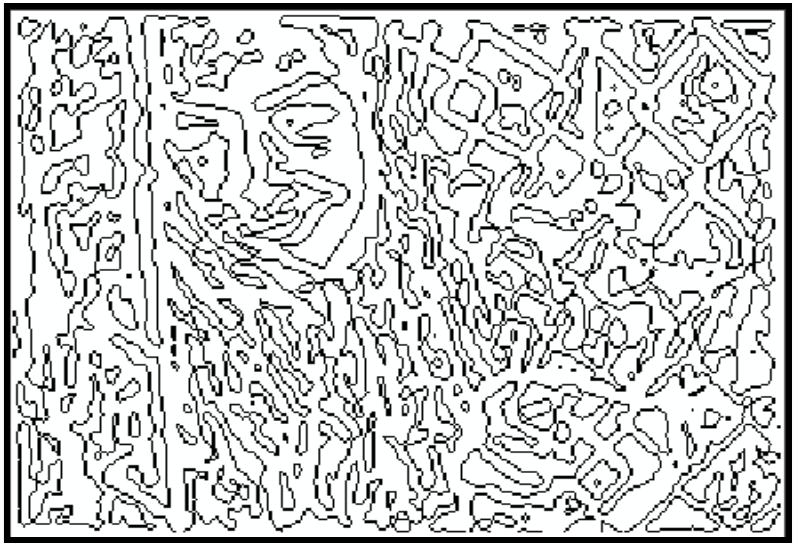
- **Reduce** the **noise** effect. Smooth the image using a Gaussian
- **Apply** a two-dimensional **Laplacian** to the smoothed image (often the first two steps are combined into a single operation)
- Loop through the result and look for sign changes.
  - Find zero-crossings from each row and column
  - Find slope of zero-crossings
  - Apply **threshold** to slope and **mark edges**

…7. Edge Detectors … 7.1. Marr-Hildreth edge detector (LoG)

Example of zero-crossings

$I * \left(\nabla^2 G\right)$

I

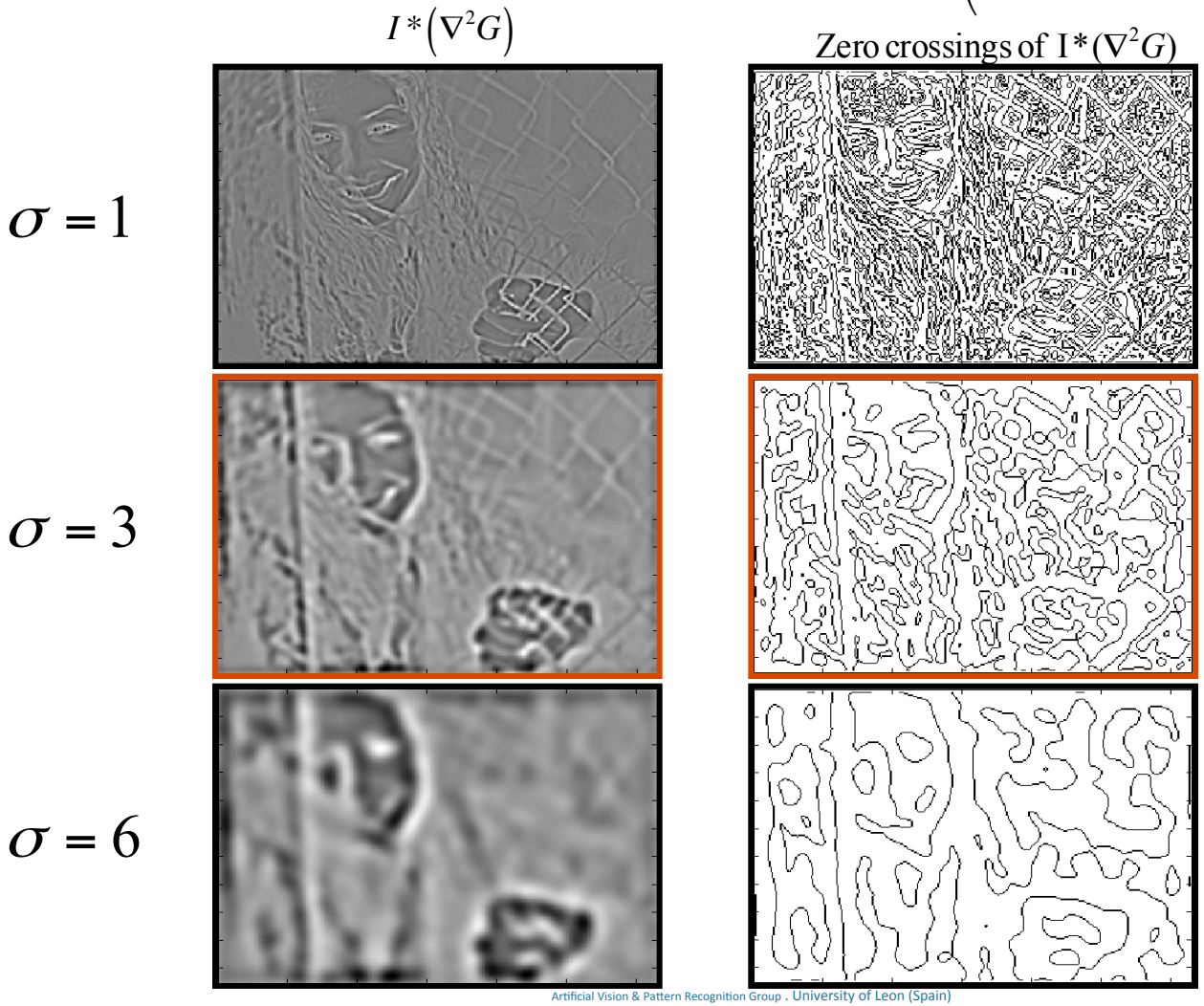


Zero crossings of $I * (\nabla^2 G)$

…7. Edge Detectors … 7.1. Marr-Hildreth edge detector (LoG)

Effect of Gaussian parameters

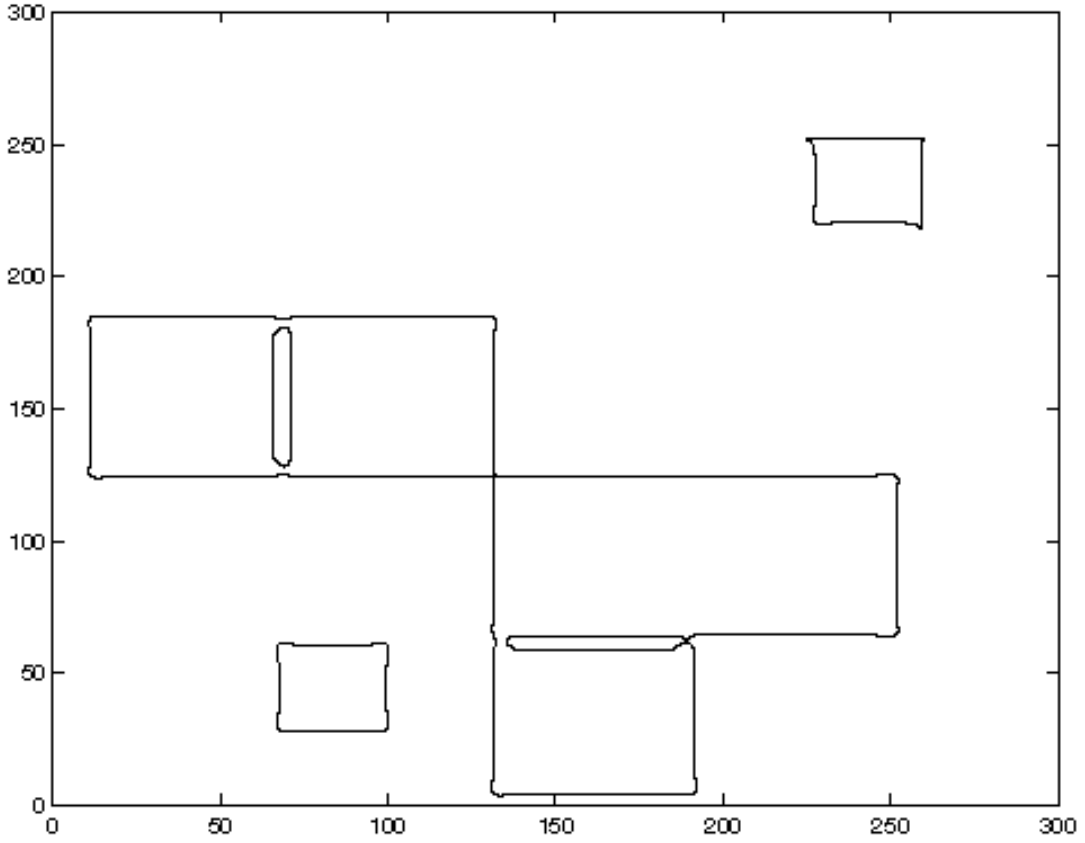
$$\nabla^2[f(x,y)*G(x,y)] = \nabla^2 G(x,y)*f(x,y) \qquad \nabla^2 G(x,y) = \left(\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$



…7. Edge Detectors … 7.1. Marr-Hildreth edge detector (LoG)

Disadvantage of LoG edge detection

Does not handle corners well

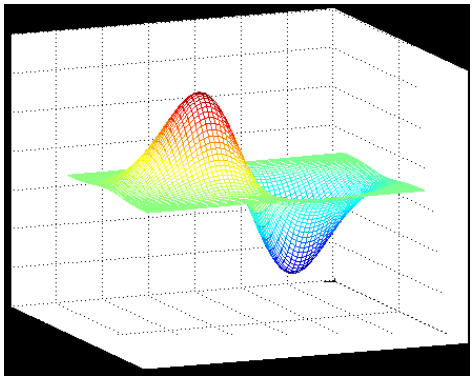


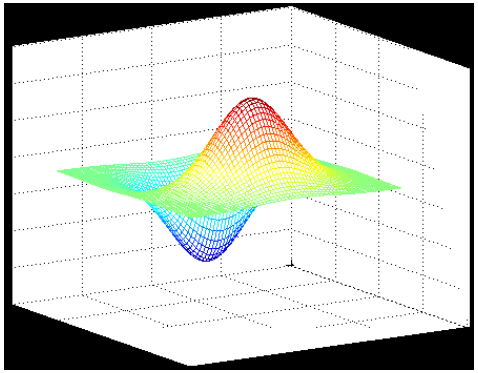
…7. Edge Detectors … 7.1. Marr-Hildreth edge detector (LoG)

Disadvantage of LoG edge detection

- Does not handle corners well
- Why?

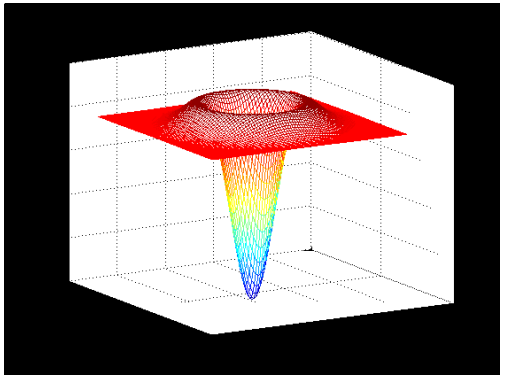
The derivative of the Gaussian:





The Laplacian of the Gaussian:

(unoriented)



…7. Edge Detectors … 7.1. Marr-Hildreth edge detector (LoG)

Difference of Gaussians (DoG) -intead of LoG

- The Difference-of-Gaussians (DOG)
  - Approximates the LOG filter with a filter that is the difference of two differently sized Gaussians – a *DOG* filter ("Difference of Gaussians").
  - The image is first smoothed by convolution with a Gaussian kernel of scale σ_1

$$g_1(x,y) = G_{\sigma 1}(x,y) * f(x,y)$$

  - A second image is obtained by smoothing with a Gaussian kernel of scale σ_2

$$g_2(x,y) = G_{\sigma 2}(x,y) * f(x,y)$$

…7. Edge Detectors … 7.1. Marr-Hildreth edge detector (LoG)

Difference of Gaussians (DoG) -intead of LoG

Their difference is:

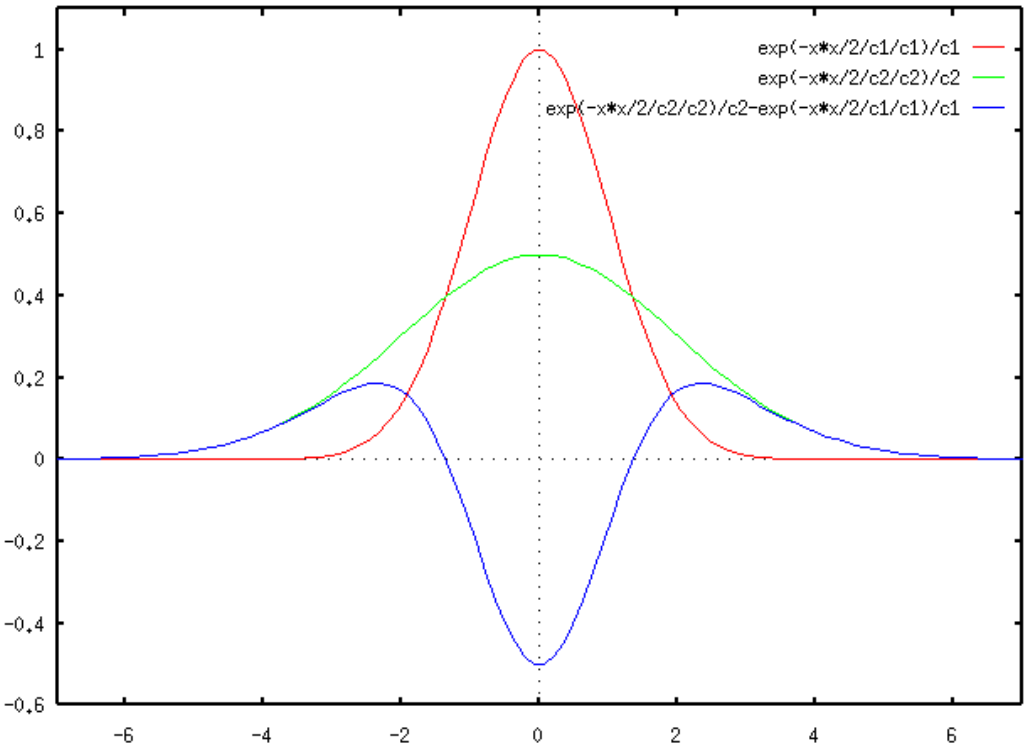
$$
\begin{aligned}
& g_1(x,y) - g_2(x,y) \\
&= G_{\sigma 1}(x,y) * f(x,y) - G_{\sigma 2}(x,y) * f(x,y) \\
&= (G_{\sigma 1} - G_{\sigma 2}) * f(x,y) = DoG * f(x,y)
\end{aligned}
$$

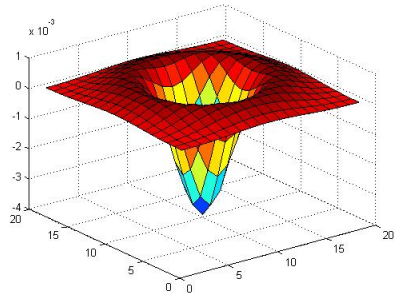
The DOG as an operator or convolution kernel is defined as:

$$
DoG = G_{\sigma 1} - G_{\sigma 2} = e^{\frac{-(x^2+y^2)}{2\sigma_1^2}} - e^{\frac{-(x^2+y^2)}{2\sigma_2^2}}
$$

…7. Edge Detectors … 7.1. Marr-Hildreth edge detector (LoG)

Difference of Gaussians (DoG) -intead of LoG





…7. Edge Detectors … 7.1. Marr-Hildreth edge detector (LoG)

Example DoG

(a) $\sigma = 1$ (b) $\sigma = 2$ (b)-(a)

…7. Edge Detectors

7.2. Canny

Edge detector based on:

1. **Smooth** image $I(x,y)$ with 2D Gaussian: $G * I$
2. Compute **edge strength** and **orientation** at all pixels

   Find Local edge normal directions for each pixel

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

   Compute edge magnitudes

   $$|\nabla(G * I)|$$

3. **"Non-maxima supression".** Locate edges by finding zero-crossings along the edge normal directions

   Reduce thick edge strength responses

   around true edges

4. **Link** and threshold using "**hysteresis**"

   Simple method of "contour completion"

…7. Edge Detectors … 7.2. Canny Edge Detector

Example

Smoothed Gradient Magnitude

Original Image

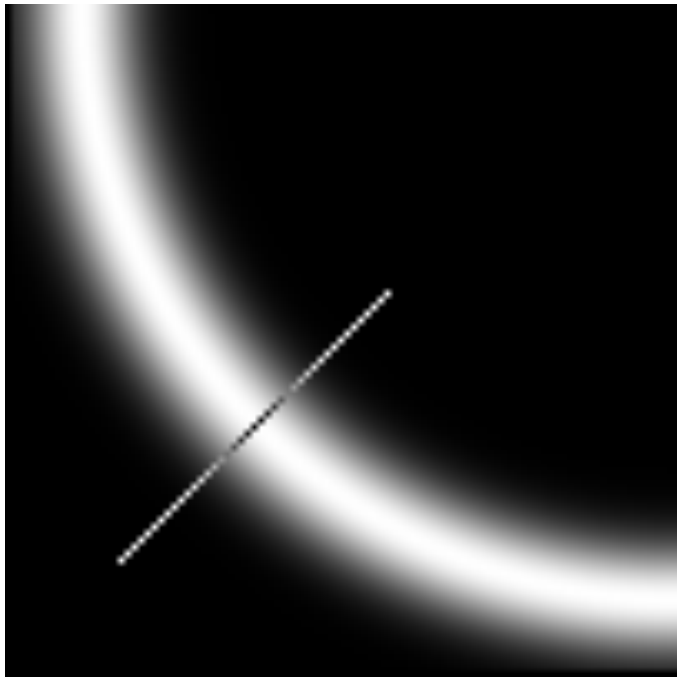
Thinning
(non-maxima suppression)

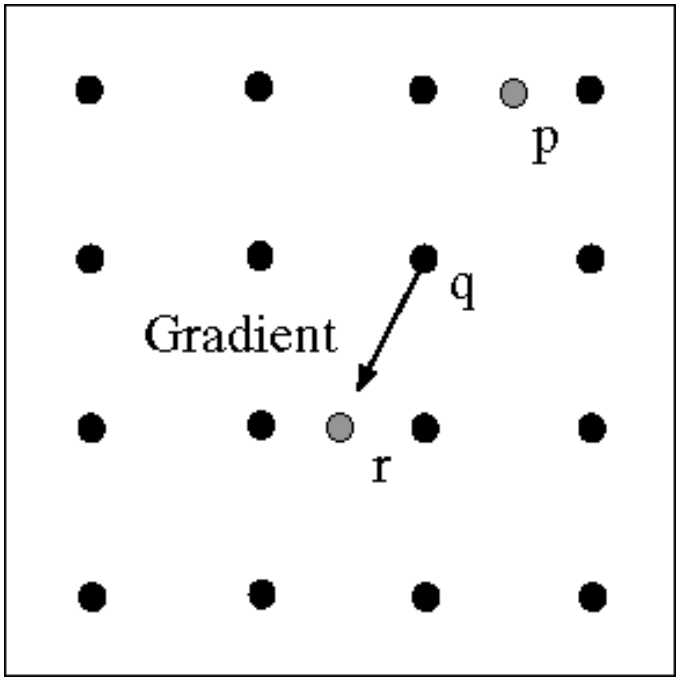
Thresholded gradient magnitude

…7. Edge Detectors … 7.2. Canny Edge Detector

Non-maximum suppression

At q, the value must be larger than values interpolated at p or r





Check if pixel is local maximum along gradient direction

…7. Edge Detectors … 7.2. Canny Edge Detector

Example. Non-maximum suppression

Original image

Gradient magnitude

courtesy of G. Loy

Non-maxima suppressed

Slide credit: Christopher Rasmussen

…7. Edge Detectors … 7.2. Canny Edge Detector

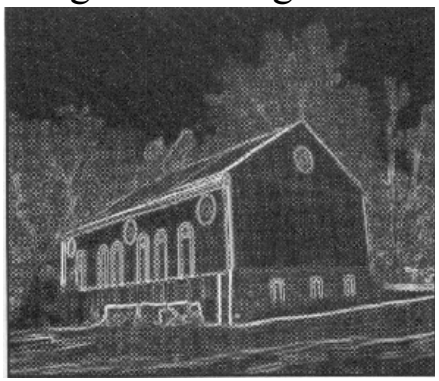
Hysteresis based Thresholding

- Standard Thresholding:

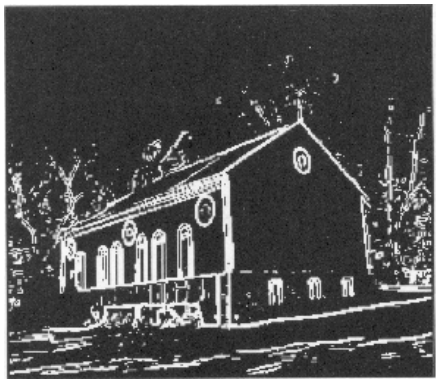
$$E(x,y) = \begin{cases} 1 & \text{if } \|\nabla f(x,y)\| > T \text{ for some threshold } T \\ 0 & \text{otherwise} \end{cases}$$

  - Can only select “strong” edges.
  - Does not guarantee “continuity”.

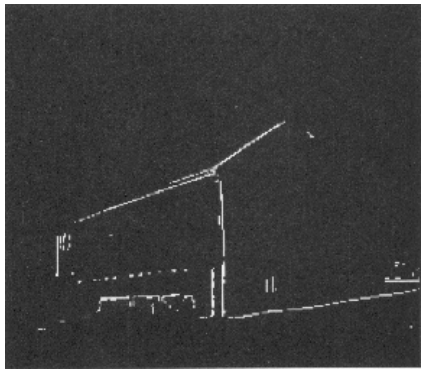
gradient magnitude



low threshold



high threshold



…7. Edge Detectors … 7.2. Canny Edge Detector

Hysteresis based Thresholding

Hysteresis based Thresholding (use two thresholds)

- Low threshold t_{low} (t_l or t_0)
- High threshold t_{high} (t_h or t_1; usually, $t_h=2t_l$)

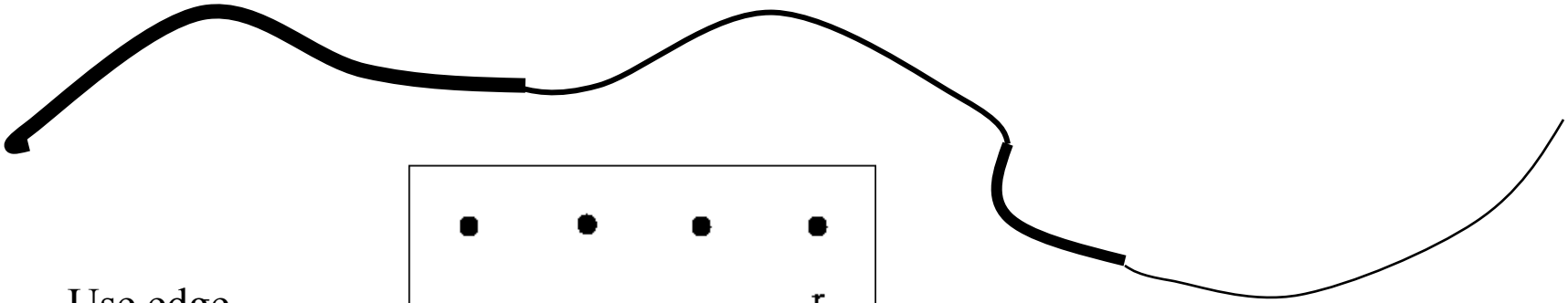
$$
\begin{array}{rll}
 & \|\nabla f(x,y)\| \geq t_1 & \text{definitely an edge} \\
t_0 \geq & \|\nabla f(x,y)\| < t_1 & \text{maybe an edge, depends on context} \\
 & \|\nabla f(x,y)\| < t_0 & \text{definitely not an edge}
\end{array}
$$

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

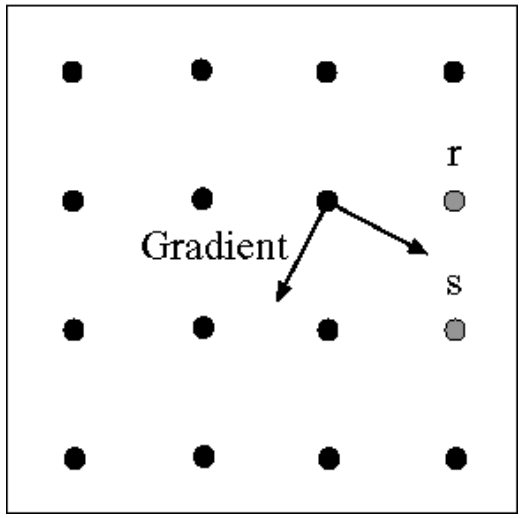
…7. Edge Detectors … 7.2. Canny Edge Detector

Hysteresis Thresholding – Edge Linking

Idea: use a **high** threshold to start edge curves and a **low** threshold to continue them.



Use edge "direction" for linking edges



…7. Edge Detectors … 7.2. Canny Edge Detector

Algorithm

A. Canny Enhancer

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

Given image I

1. Apply Gaussian Smoothing to I.

$$J = G * I$$

2. For each pixel (i, j):

   1. Compute the gradient components

$$\nabla I_x = \frac{\partial I}{\partial x} \quad \nabla I_y = \frac{\partial I}{\partial y}$$

   2. **(Es)** Estimate the edge strength

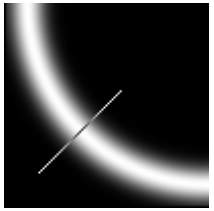
$$\nabla I = mag(\nabla \mathbf{I}) = \sqrt{\nabla I_x^2 + \nabla I_y^2}$$

   3. **(Eo)** Estimate the orientation of the edge normal

$$direction(\nabla \mathbf{f}) = \tan^{-1}\left(\frac{\nabla I_y}{\nabla I_x}\right)$$

…7. Edge Detectors … 7.2. Canny Edge Detector

… Algorithm



B. Canny Edge Detector: Non-maxima suppression

The input is the output of CANNY_ENHANCER.

We need to **thin** the edges.

Given Es, Eo, the **E**dge **s**trength and **E**dge **o**rientation images. For each pixel (i, j),

1. Find the direction best approximate the direction Eo(i, j).
2. If Es (i, j) is smaller than at least one of its two neighbors along this direction, suppress this pixel.

The output is an image of the thinned edge points after suppressing non-maxima edge points.

…7. Edge Detectors … 7.2. Canny Edge Detector

… Algorithm

C. Canny Edge Detector: Hysteresis Thesholding

Performs edge tracking and reduces the probability of false contours.

Use two threshold parameters, t_l (Threshold low), and t_h (Threshold high)

1. Produce two thresholded images $I_1(i,j)$ and $I_h(i,j)$, using t_l and t_h

   Since $I_h(i,j)$ was formed with a high threshold, it will contain fewer false edges but there might be gaps in the contours.

2. Link the edges in $I_h(i,j)$ into contours

   1. Look in $I_l(i,j)$ when a gap is found
   2. By examining the 8 neighbors in $I_l(i,j)$, gather edge points from $I_l(i,j)$ until the gap has been bridged to an edge in $I_h(i,j)$

Note: Large gaps are difficult to bridge with this algorithm.

Example: Canny Edge Detection

Original image

gap is gone

Strong + connected weak edges

Strong edges only

Weak edges

courtesy of G. Loy

Summary of concepts

- Concept of Edge
- Gradient, concept and discrete gradient
- Discrete convolution
- Convolution mask or filter
- Detecting edges: first derivative and Threshold
- Gradient operators: mask and concepts
- Smoothing for noise removal
- Detecting the edge in the zero-crossings
- Edge Detectors: Marr-Hildreth and Canny