中山大學
SUN YAT-SEN UNIVERSITY

Lecture 3.
Edge Detection

Pattern Recognition and Computer Vision

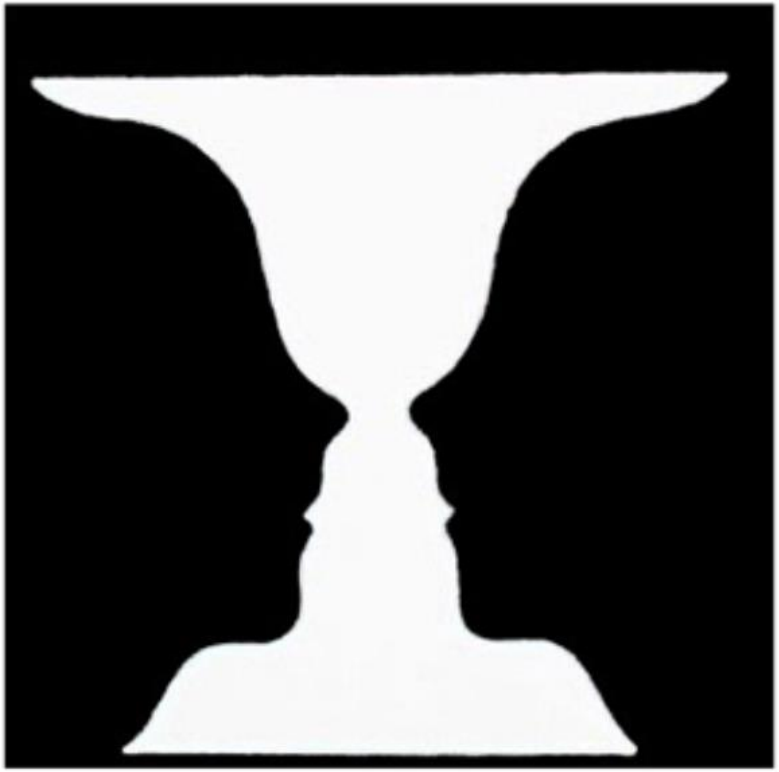
Guanbin Li,
School of Computer Science and Engineering, Sun Yat-Sen University

扫码签到

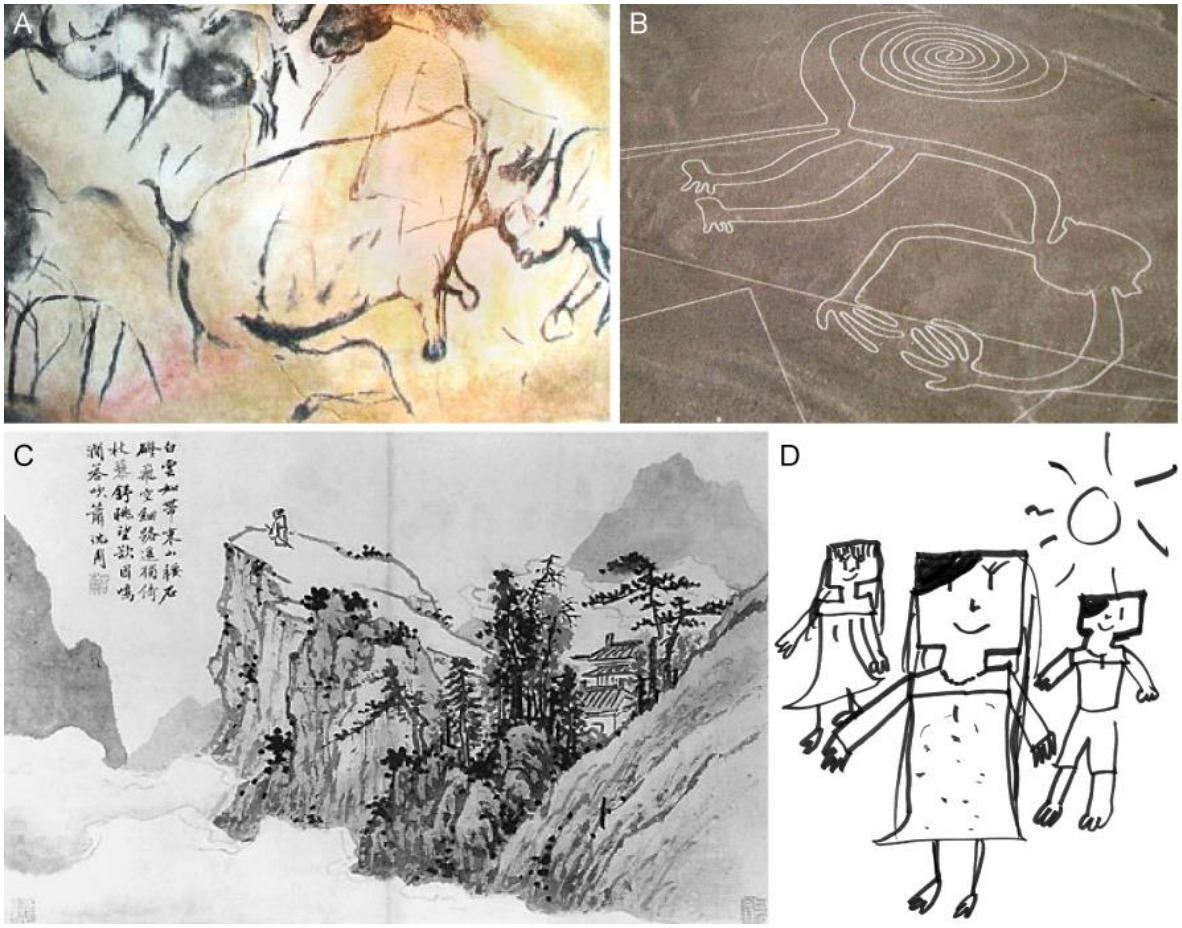
What we will learn today?

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel edge detector
- Canny edge detector
- Laplacian of Gaussian

Edge detection



Edge detection



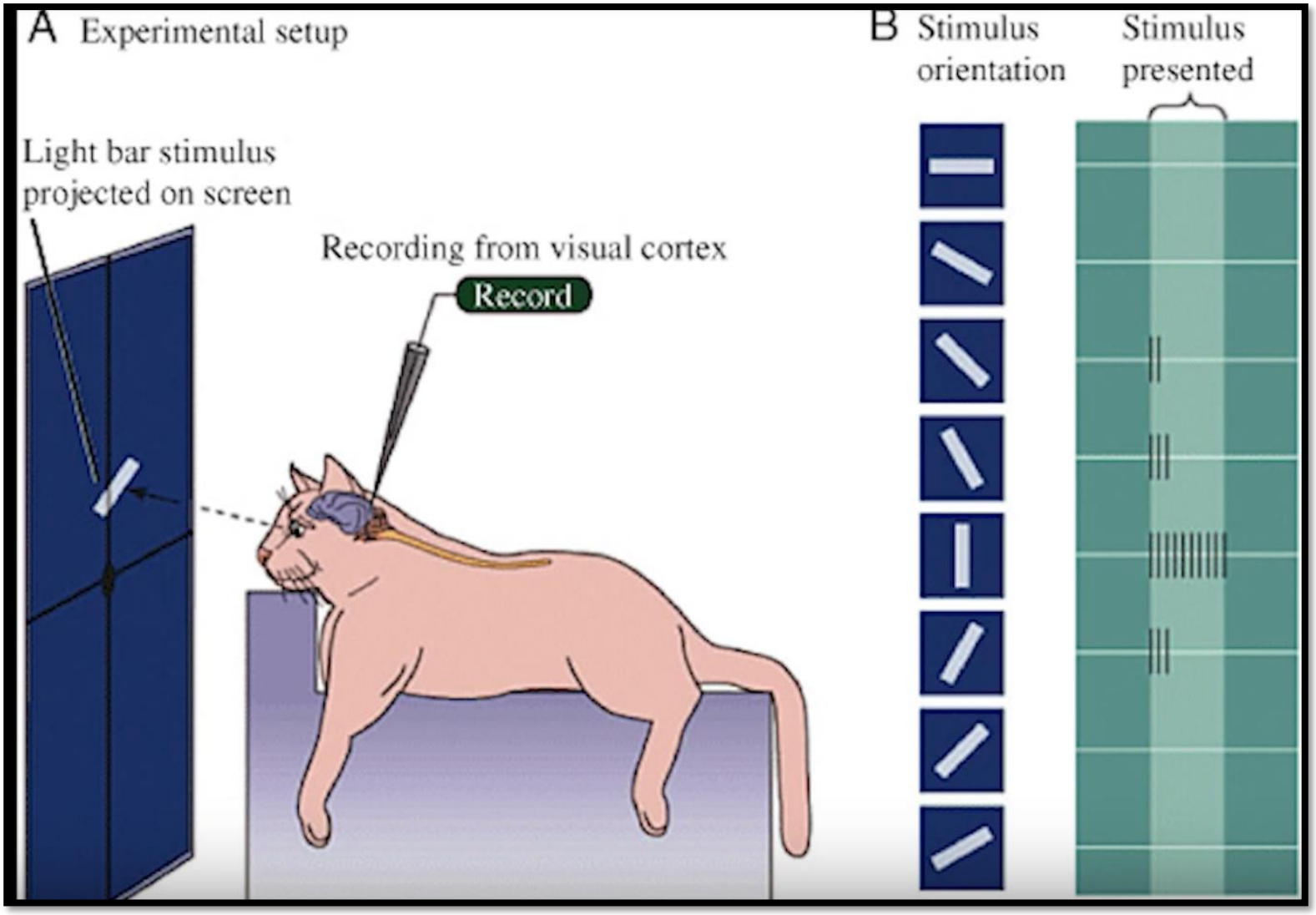
(A) Cave painting at Chauvet, France, about 30,000 B.C.;

(B) Aerial photograph of the picture of a monkey as part of the Nazca Lines geoglyphs, Peru, about 700 – 200 B.C.;

(C) Shen Zhou (1427-1509 A.D.): Poet on a mountain top, ink on paper, China;

(D) Line drawing by 7-year old I. Lleras (2010 A.D.)

Edge detection



Edge detection

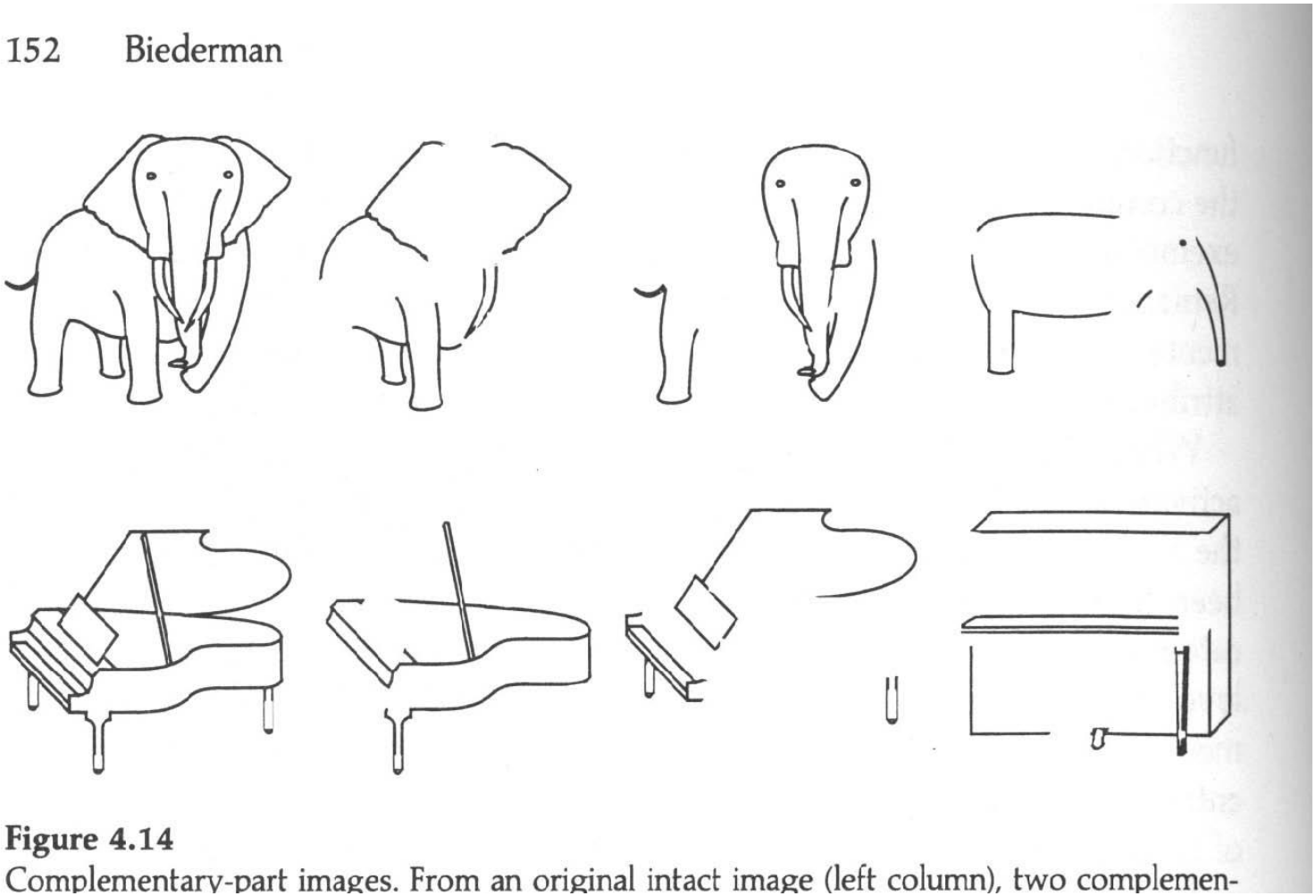
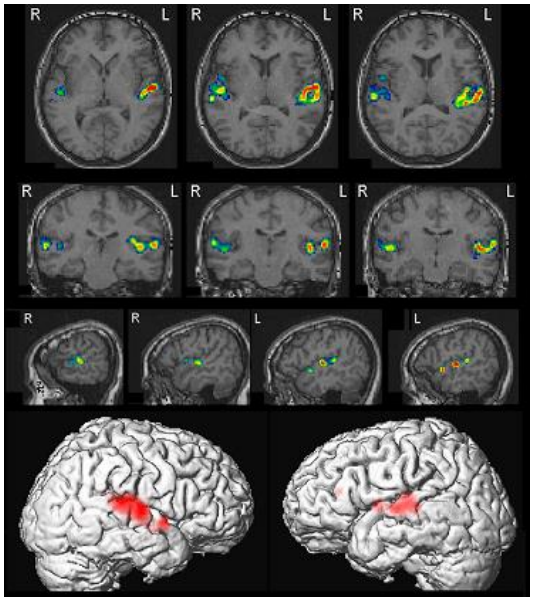
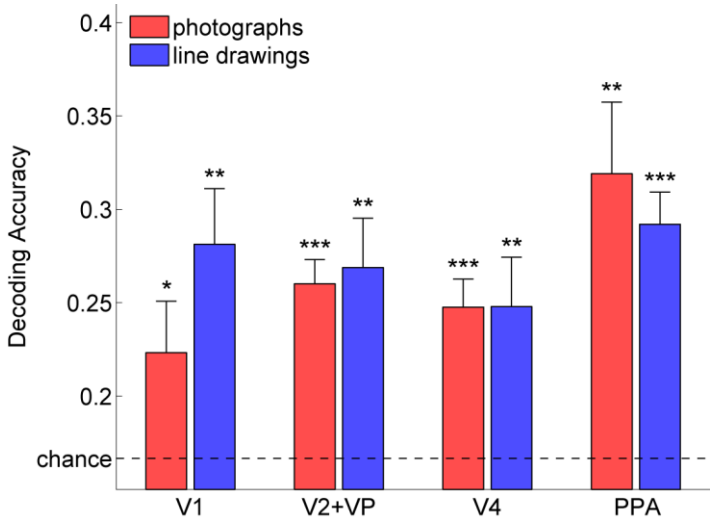


Figure 4.14
Complementary-part images. From an original intact image (left column), two complemen-

Edge detection





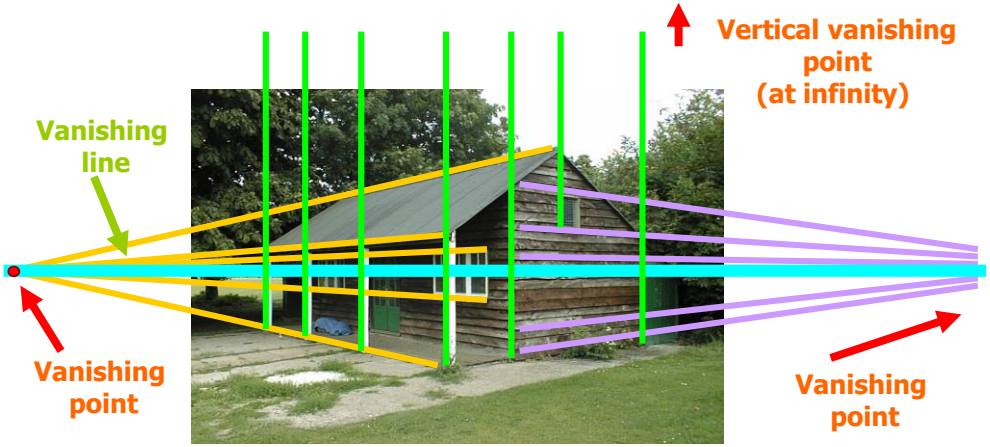
Edge detection

- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)

Edge detection

Why do we care about edges?

- Extract information, recognize objects
  - Image segmentation, region separation, object description and recognition
- Recover geometry and viewpoint
  - To extract information about the two-dimensional projection of a 3D scene



Edge detection

Origins of edges

surface normal discontinuity

depth discontinuity

surface color discontinuity

illumination discontinuity

Edge detection

Closeup of edges

Surface normal discontinuity

Edge detection

Closeup of edges

Edge detection

Closeup of edges

What we will learn today?

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel edge detector
- Canny edge detector
- Laplacian of Gaussian

Image Gradients

Derivatives in 1D

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

$$y = x^2 + x^4 \qquad y = \sin x + e^{-x}$$

$$\frac{dy}{dx} = 2x + 4x^3 \qquad \frac{dy}{dx} = \cos x + (-1)e^{-x}$$

Image Gradients

Discrete Derivative in 1D

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

$$\frac{df}{dx} = \frac{f(x) - f(x-1)}{1} = f'(x)$$

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

Image Gradients

Types of Discrete derivative in 1D

Backward $$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

Forward $$\frac{df}{dx} = f(x) - f(x+1) = f'(x)$$

Central $$\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$$

Image Gradients

1D discrete derivate filters

- Backward filter: [0 1 -1]

$$f(x) - f(x-1) = f'(x)$$

Image Gradients

1D discrete derivate filters

- Backward filter: [0 1 -1]

$$f(x) - f(x-1) = f'(x)$$

- Forward: [-1 1 0]

$$f(x) - f(x+1) = f'(x)$$

Image Gradients

1D discrete derivate filters

- Backward filter: [0 1 -1]

$$f(x) - f(x-1) = f'(x)$$

- Forward: [-1 1 0]

$$f(x) - f(x+1) = f'(x)$$

- Central: [1 0 -1]

$$f(x+1) - f(x-1) = f'(x)$$

Image Gradients

1D discrete derivate example

$$f(x) = 10 \quad 15 \quad 10 \quad 10 \quad 25 \quad 20 \quad 20 \quad 20$$

$$f'(x) = \quad 0 \quad 5 \quad -5 \quad 0 \quad 15 \quad -5 \quad 0 \quad 0$$

Image Gradients

Discrete derivate in 2D

Given function $f(x, y)$

Image Gradients

Discrete derivate in 2D

Given function $f(x, y)$

Gradient vector $$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

Image Gradients

Discrete derivate in 2D

Given function $f(x, y)$

Gradient vector $$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

Gradient magnitude $$\left|\nabla f(x, y)\right| = \sqrt{f_x^2 + f_y^2}$$

Gradient direction $$\theta = \tan^{-1} \frac{f_x}{f_y}$$

Image Gradients

2D discrete derivative filters

What does this filter do?

$$\frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Image Gradients

2D discrete derivative filters

$$\frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

What about this filter?

$$\frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Image Gradients

2D discrete derivative - example

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

Image Gradients

2D discrete derivative - example

What happens when we apply this filter?

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

$$\frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Image Gradients

2D discrete derivative - example

What happens when we apply this filter?

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

$$\frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$I_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Image Gradients

2D discrete derivative - example

Now let's try the other filter!

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

$$\frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Image Gradients

2D discrete derivative - example

What happens when we apply this filter?

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

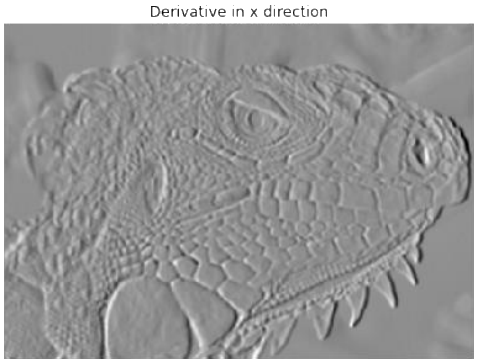
$$\frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

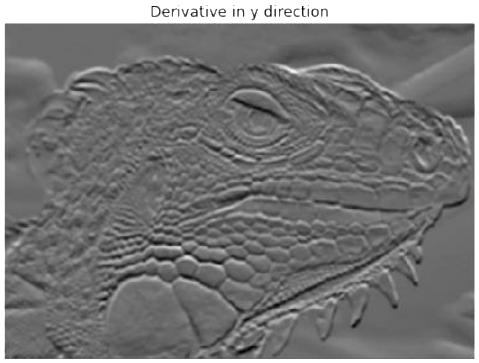
$$I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Image Gradients

3x3 image gradient filters

$$\frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \qquad \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$





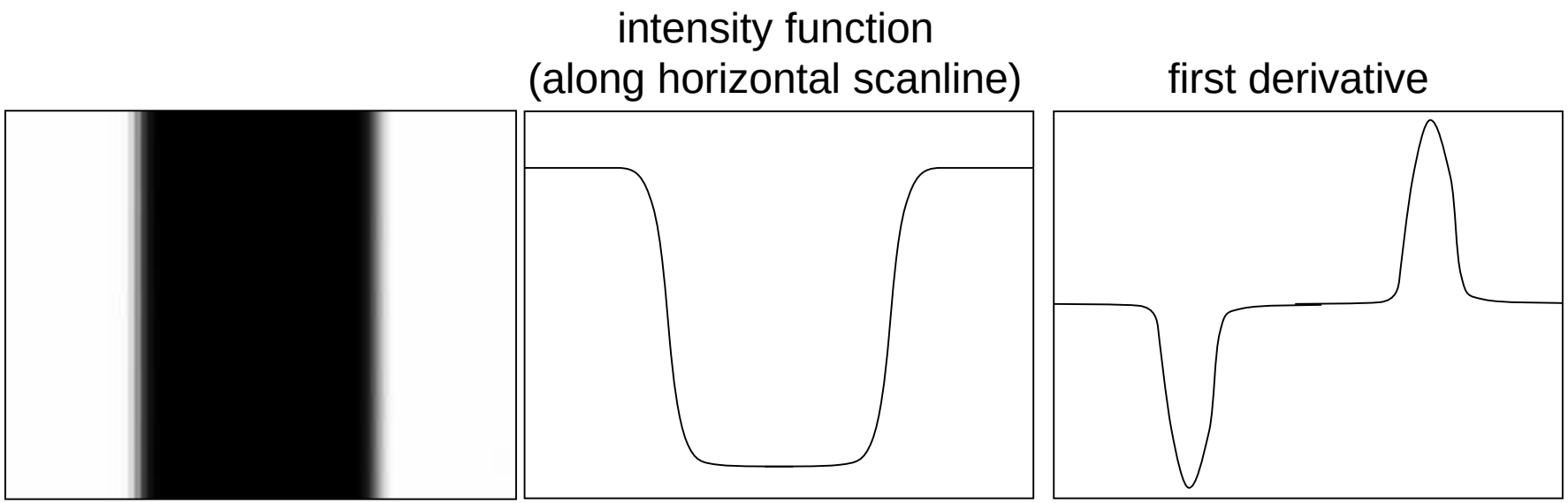
What we will learn today?

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel edge detector
- Canny edge detector
- Laplacian of Gaussian

A simple edge detector

Characterizing edges

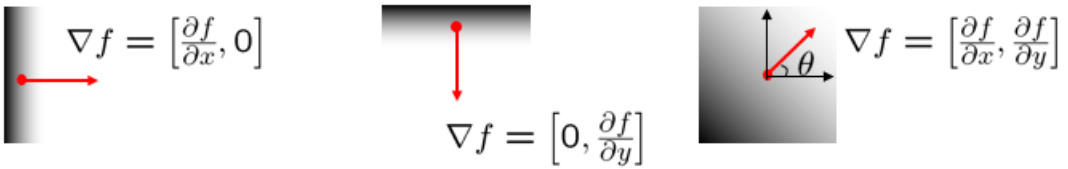
- An edge is a place of rapid change in the image intensity function



A simple edge detector

Image gradient

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



The gradient vector points in the direction of most rapid increase in intensity

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

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

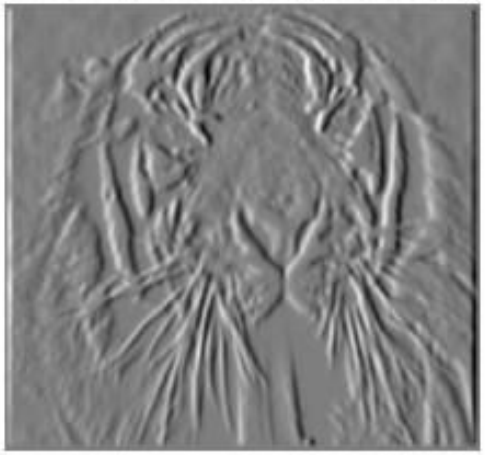
A simple edge detector

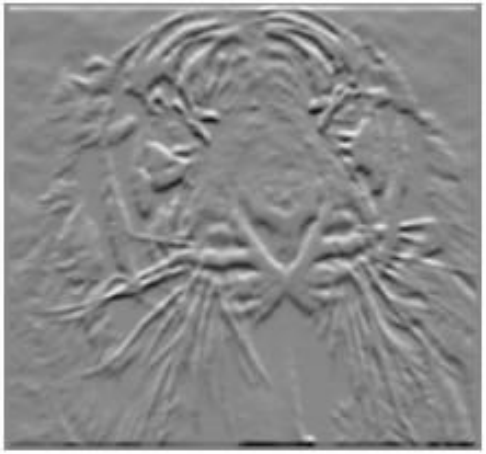
Finite differences: example

Original Image

Gradient magnitude

x-direction

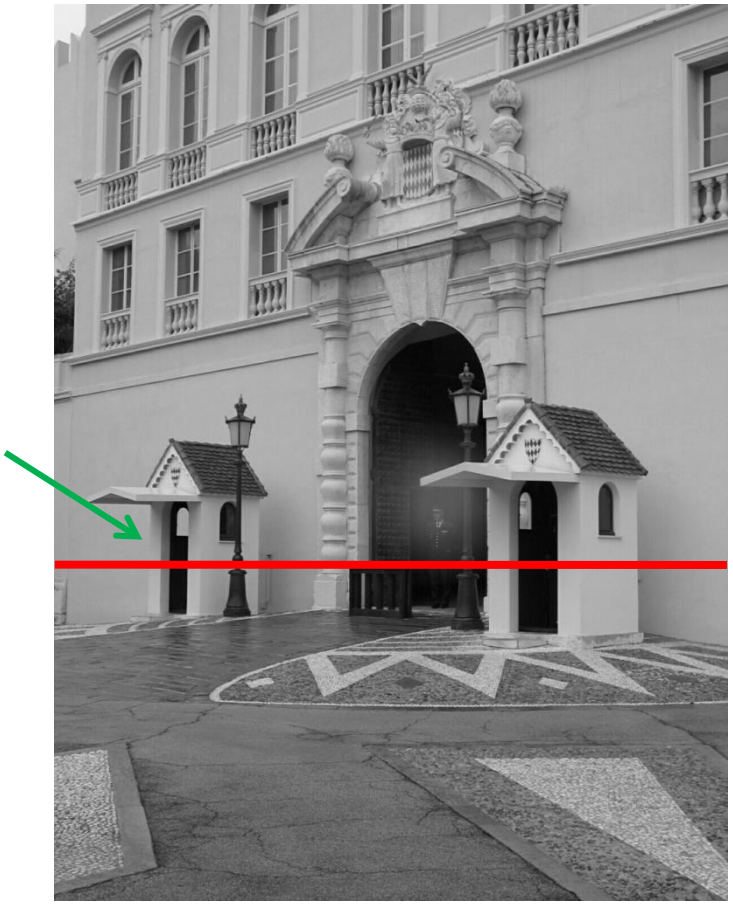


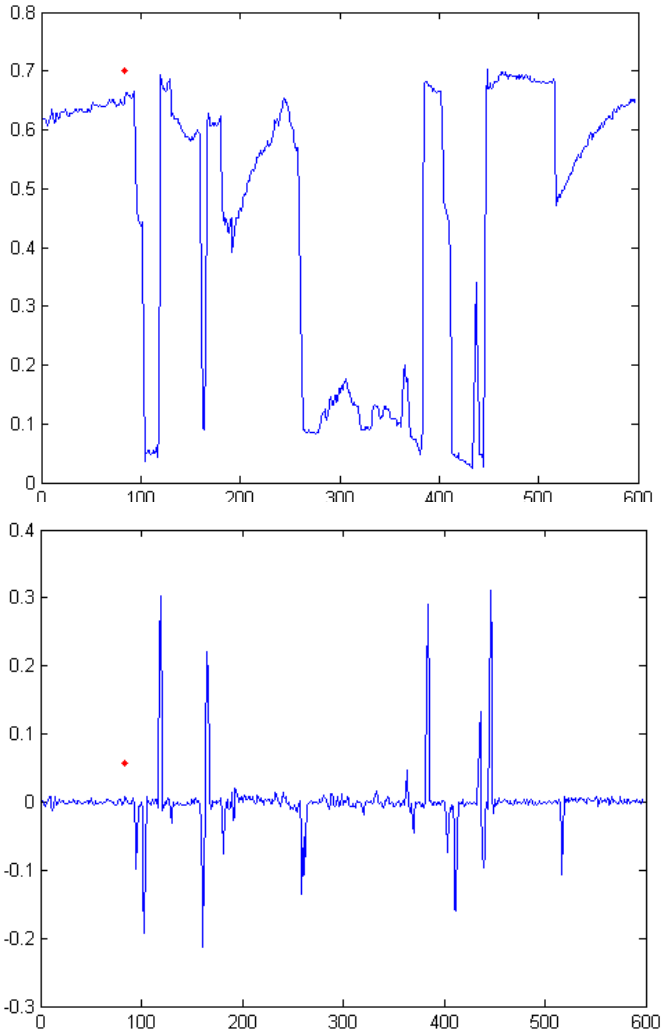


y-direction

A simple edge detector

Intensity profile

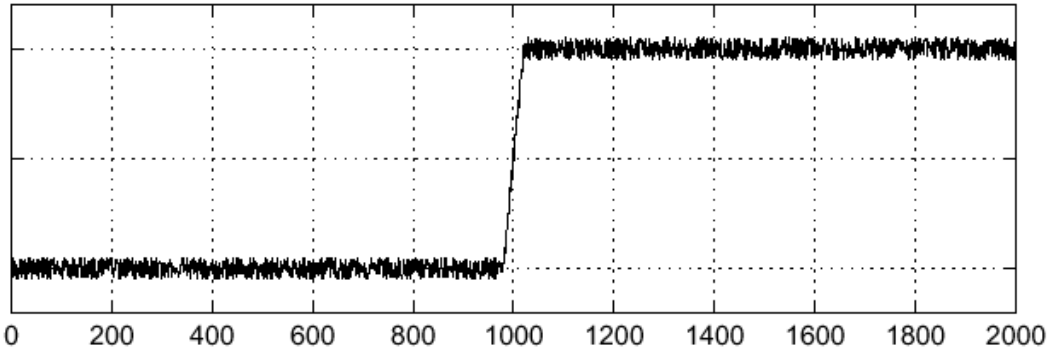




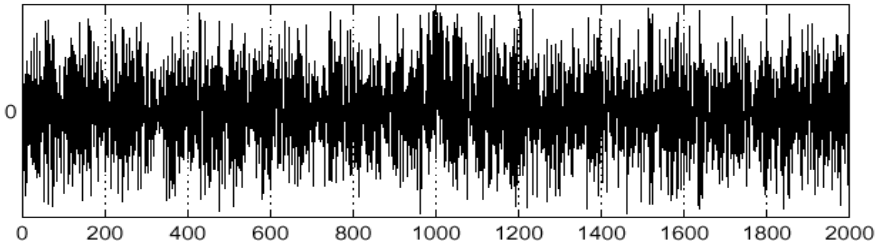
A simple edge detector

Effects of noise

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



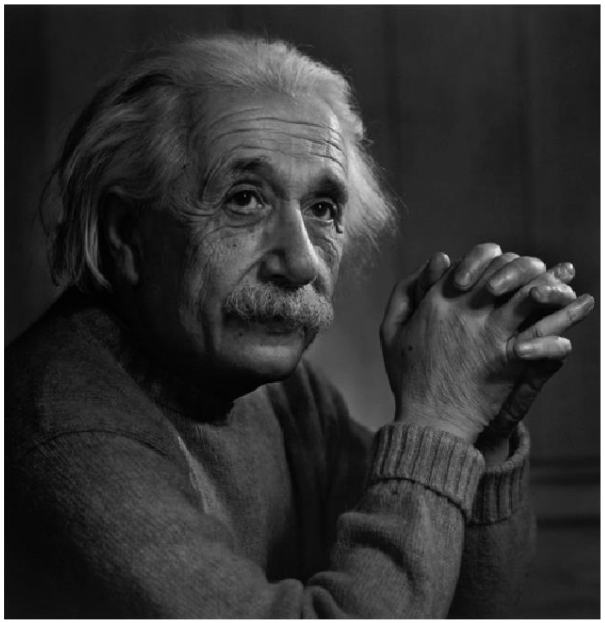
$\frac{d}{dx} f(x)$

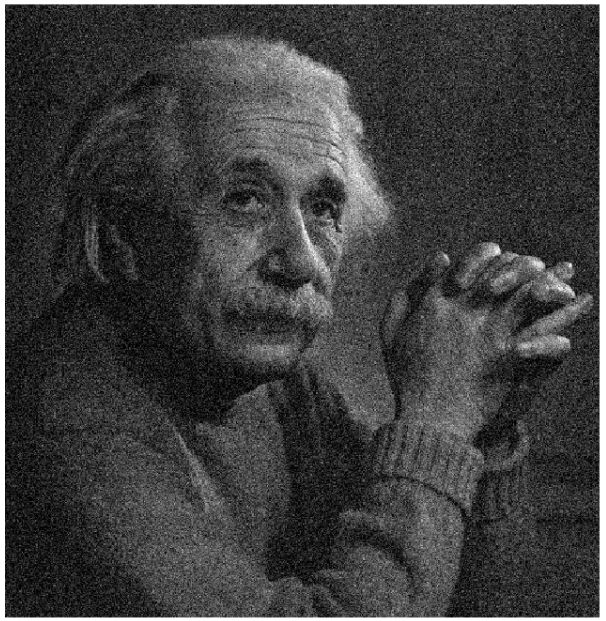


Where is the edge?

A simple edge detector

Effects of noise





A simple edge detector

Effects of noise

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What is to be done?
  - Smoothing the image should help, by forcing pixels different to their neighbors (=noise pixels?) to look more like neighbors

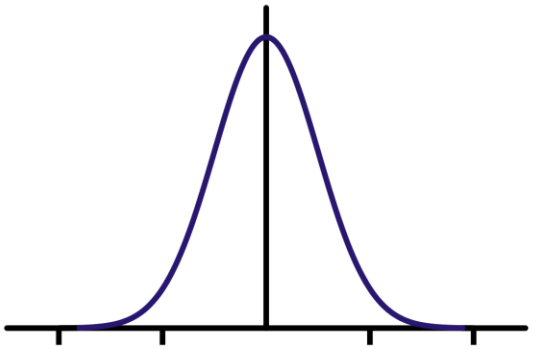
A simple edge detector

Smoothing with different filters

- Mean smoothing

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

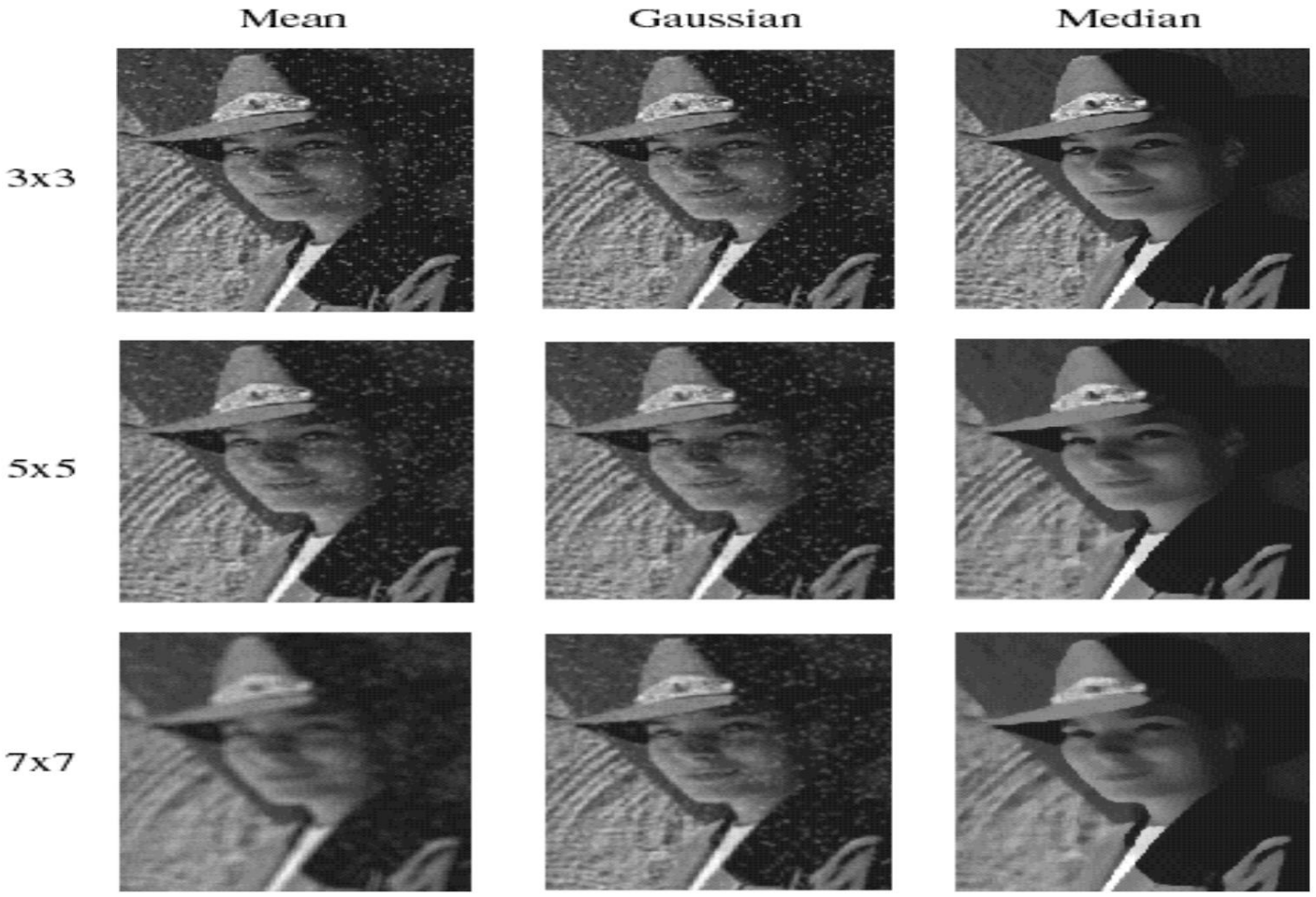
- Gaussian smoothing



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

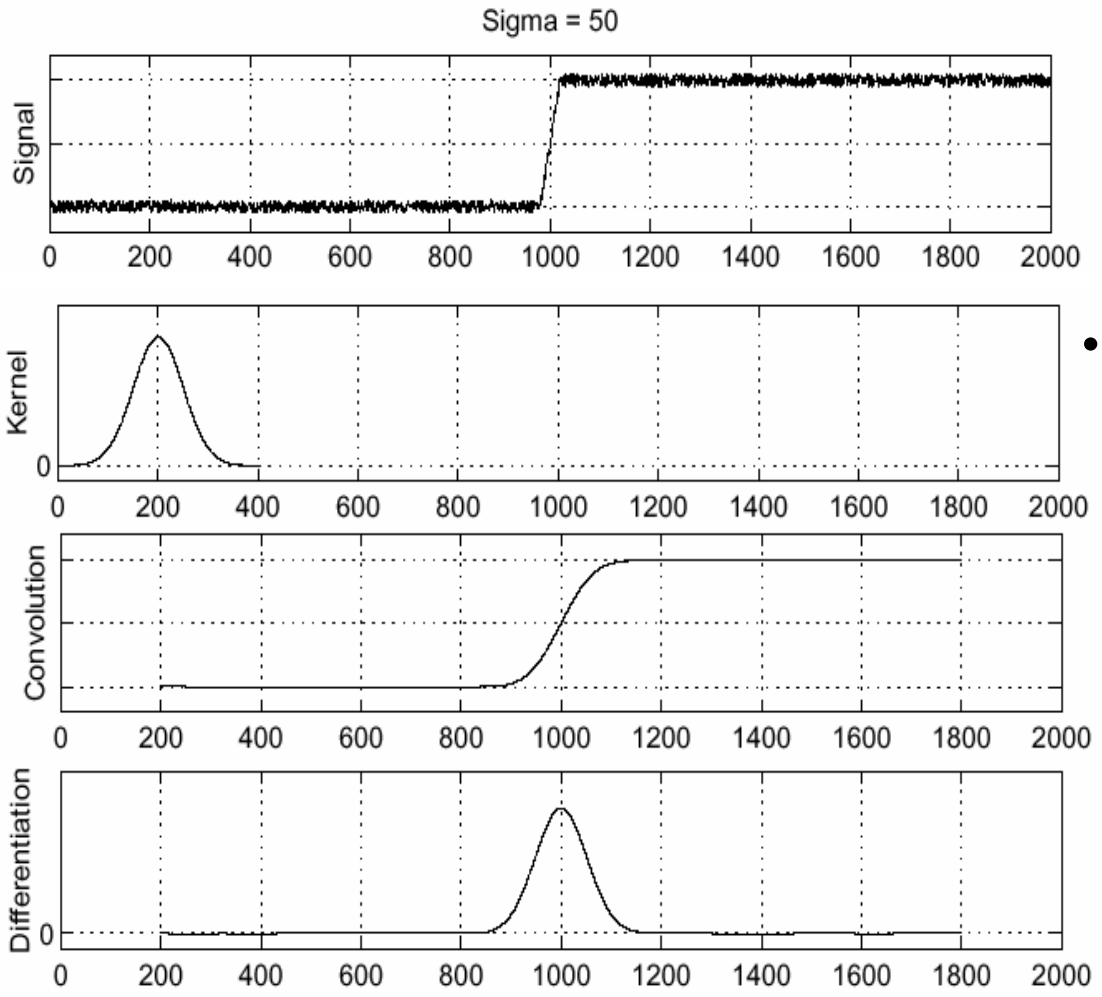
A simple edge detector

Smoothing with different filters



A simple edge detector

Solution: smooth first



- To find edges, look for peaks in

$$\frac{d}{dx}(f * g)$$

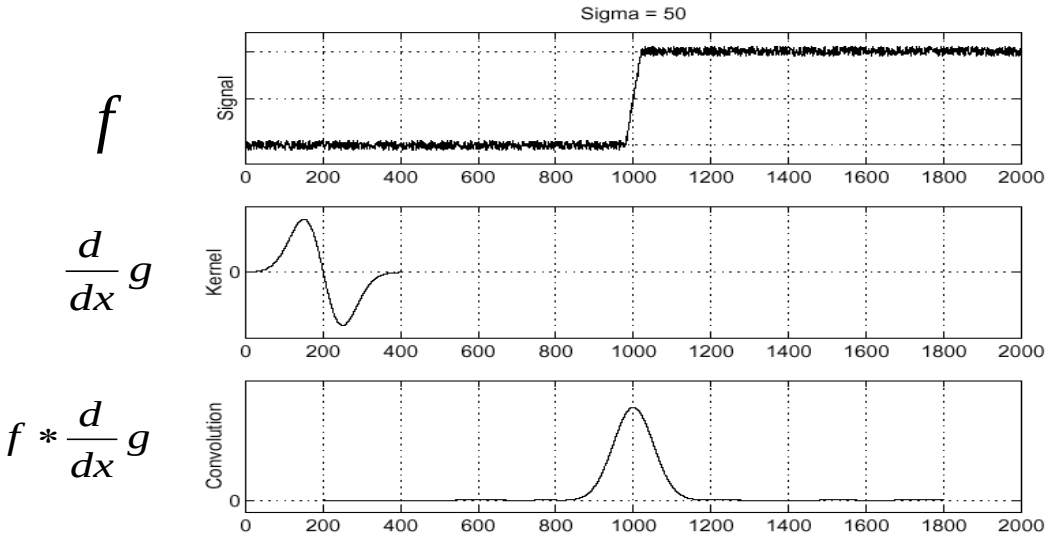
A simple edge detector

Derivative theorem of convolution

- This theorem gives us a very useful property:

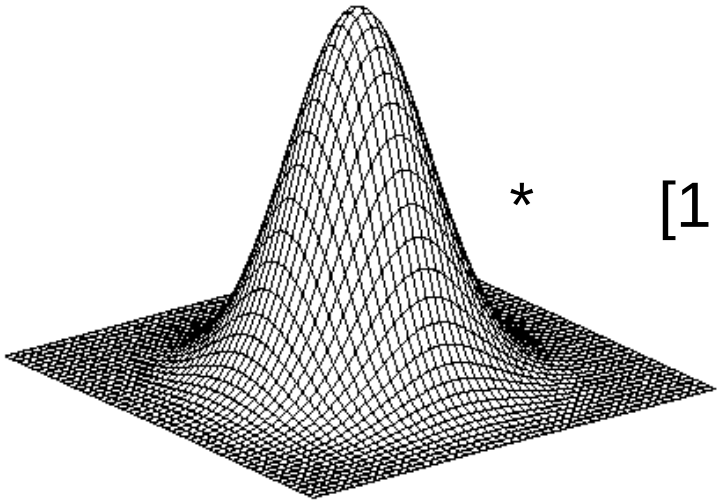
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

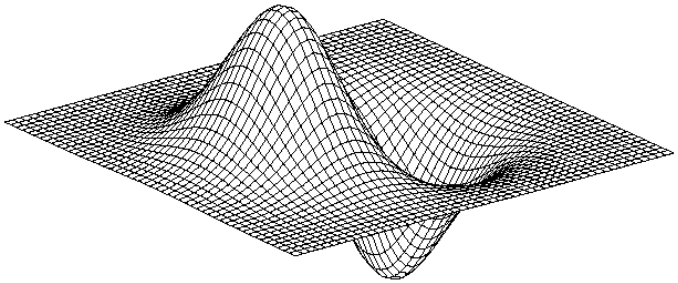


A simple edge detector

Derivative of Gaussian filter



* [1 0 -1] =

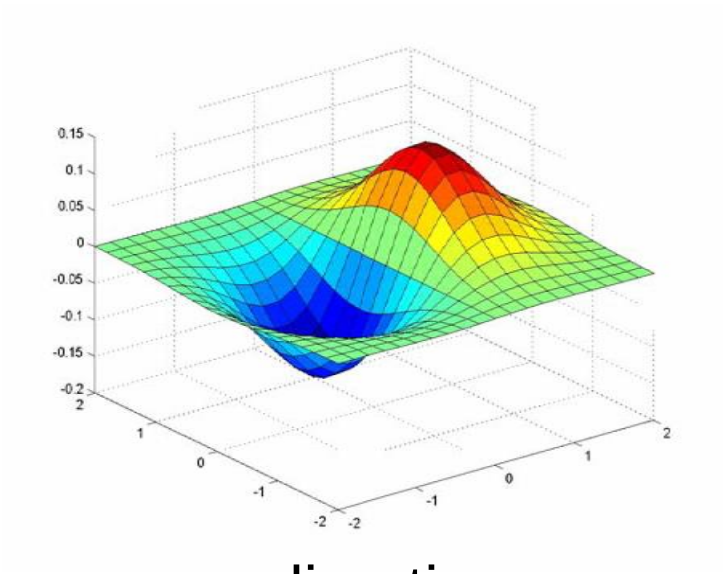


2D-gaussian

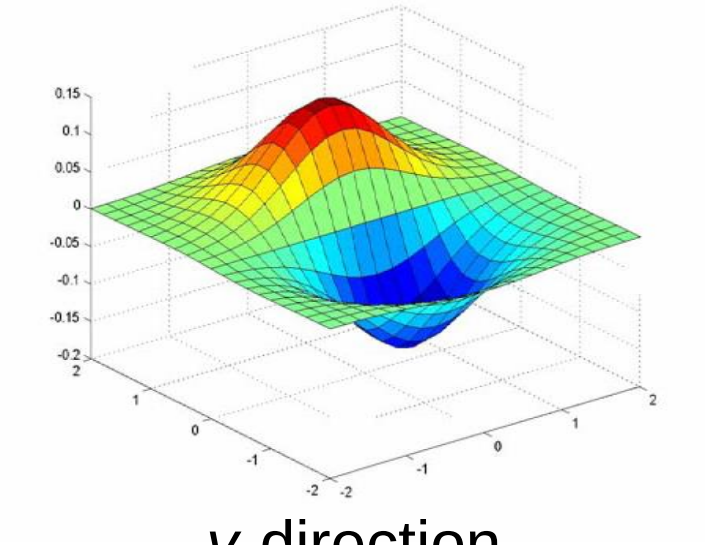
x - derivative

A simple edge detector

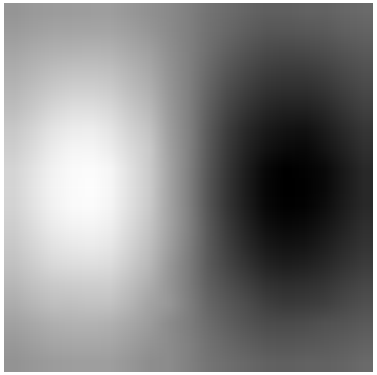
Derivative of Gaussian filter

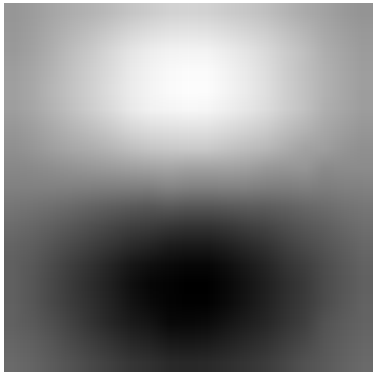


x-direction



y-direction



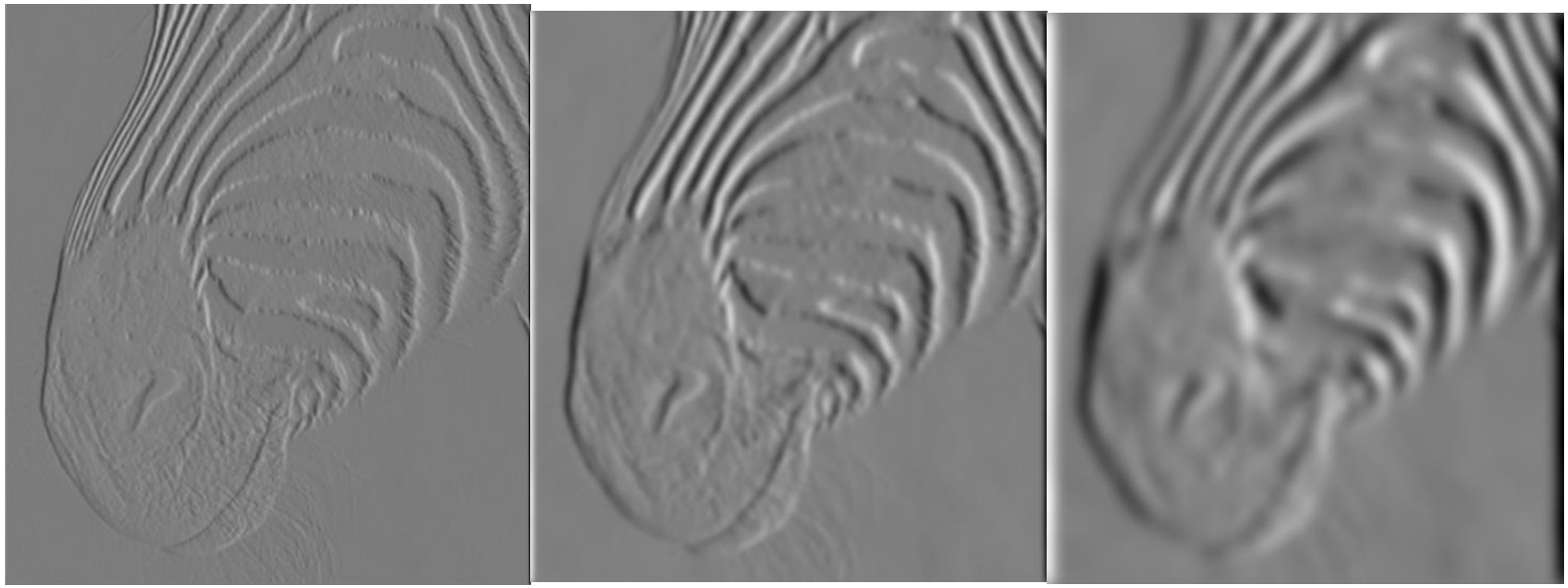


A simple edge detector

Derivative of Gaussian filter

A simple edge detector

Tradeoff between smoothing at different scales



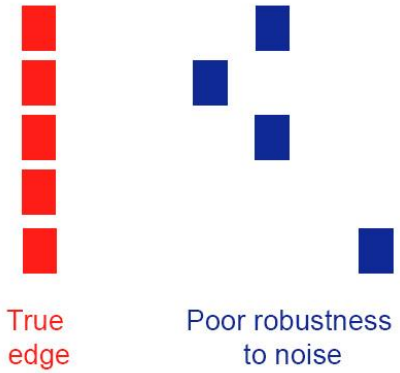
1 pixel 3 pixels 7 pixels

Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

A simple edge detector

Designing an edge detector

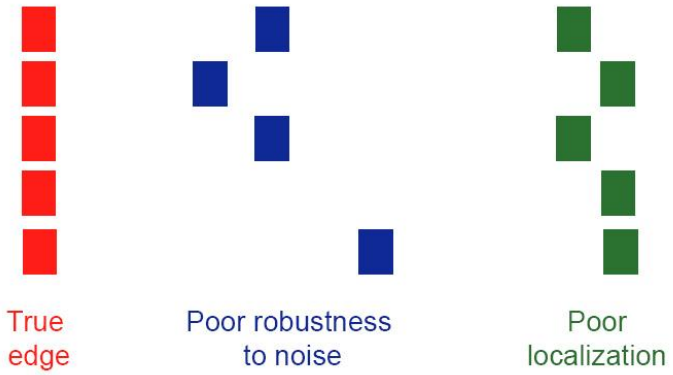
- Criteria for an “optimal” edge detector:
  - **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)



A simple edge detector

Designing an edge detector

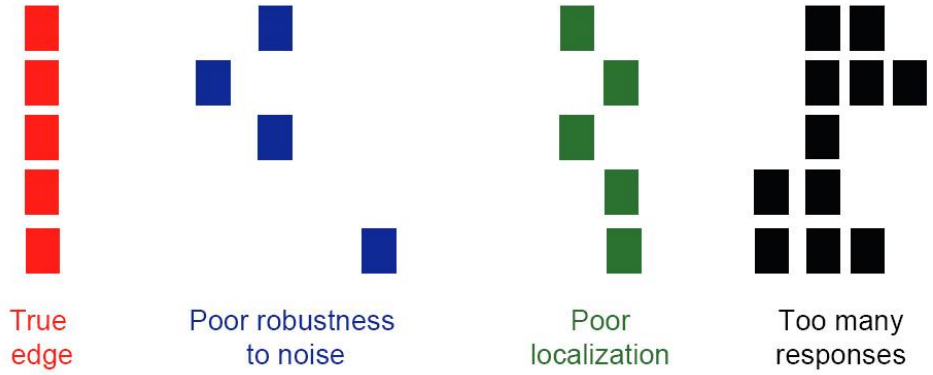
- Criteria for an “optimal” edge detector:
  - **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  - **Good localization:** the edges detected must be as close as possible to the true edges



A simple edge detector

Designing an edge detector

- Criteria for an “optimal” edge detector:
  - **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  - **Good localization:** the edges detected must be as close as possible to the true edges
  - **Single response:** the detector must return one point only for each true edge point; minimize the number of local maxima around the true edge



What we will learn today?

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel edge detector
- Canny edge detector
- Laplacian of Gaussian

Sobel edge detector

Sobel Operator

- uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives
- one for horizontal changes, and one for vertical

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \qquad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Sobel edge detector

Sobel Operator

- Smoothing + differentiation

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} +1 & 0 & -1 \end{bmatrix}$$

Gaussian smoothing

differentiation

Sobel edge detector

Sobel Operator

- Magnitude:

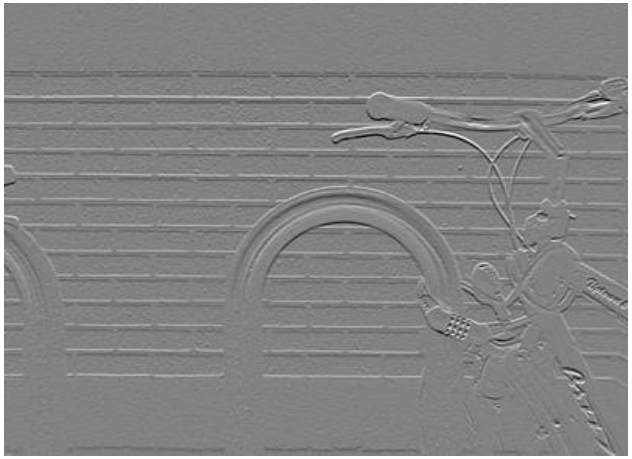
$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

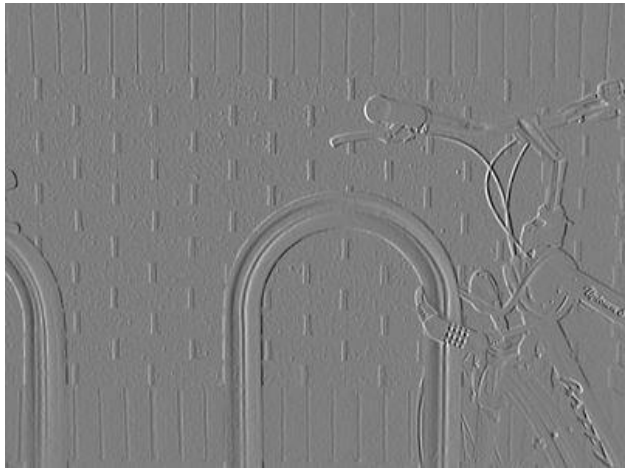
- Angle or direction of the gradient:

$$\mathbf{\Theta} = \operatorname{atan}\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

Sobel edge detector

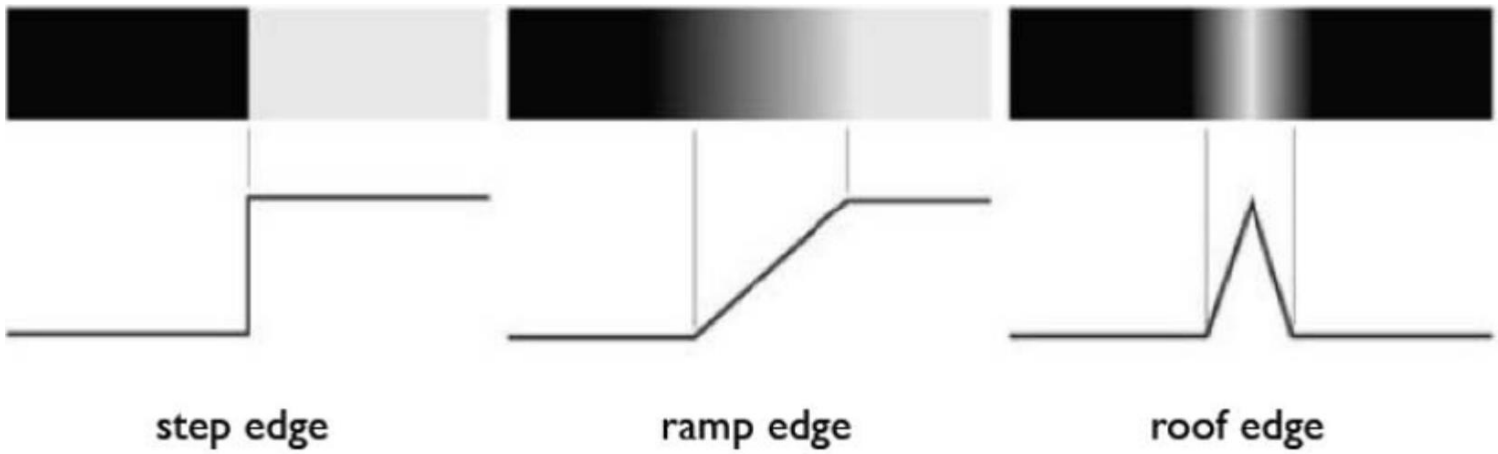
Sobel Filter example





Sobel edge detector

Sobel Filter Problems



- Poor Localization (Trigger response in multiple adjacent pixels)
- Thresholding value favors certain directions over others
  - Can miss oblique edges more than horizontal or vertical edges
  - False negatives

What we will learn today?

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel edge detector
- Canny edge detector
- Laplacian of Gaussian

Canny edge detector

- Suppress Noise
- Compute gradient magnitude and direction
- Apply Non-Maximum Suppression
  - Assures minimal response
- Use hysteresis and connectivity analysis to detect edges

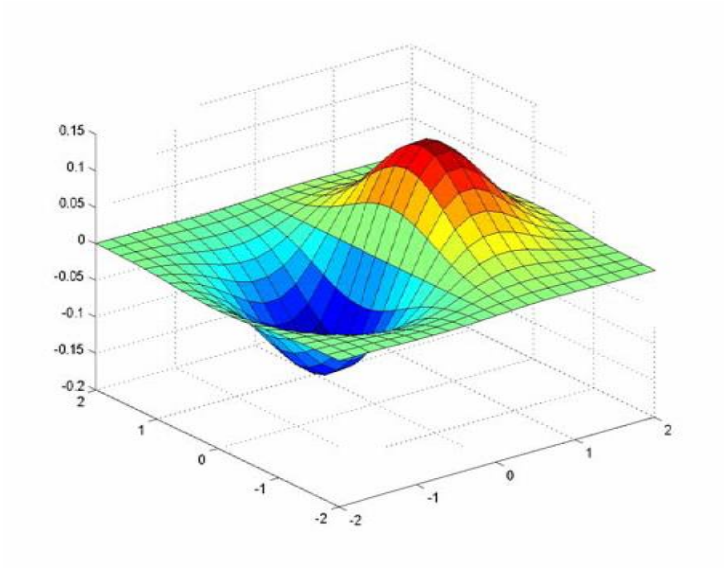
Canny edge detector

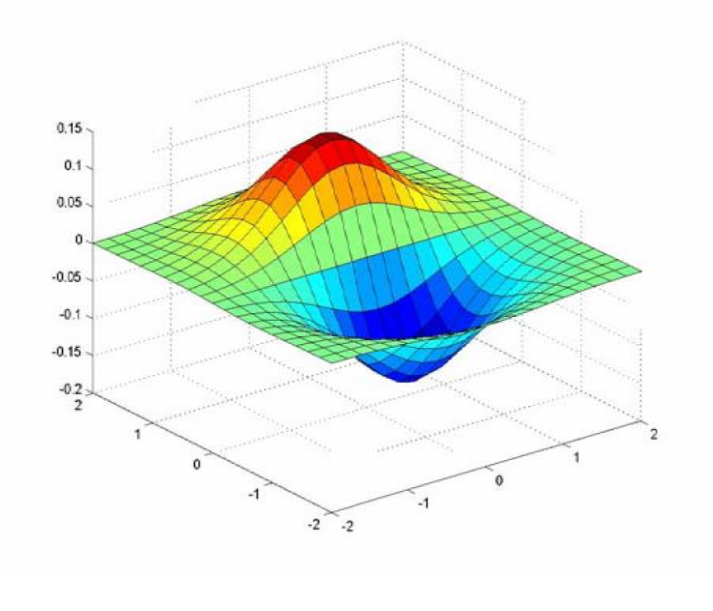
Example

original image

Canny edge detector

Derivative of Gaussian filter

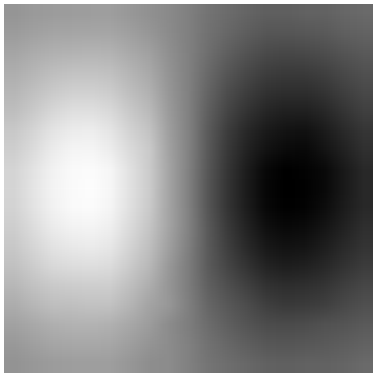


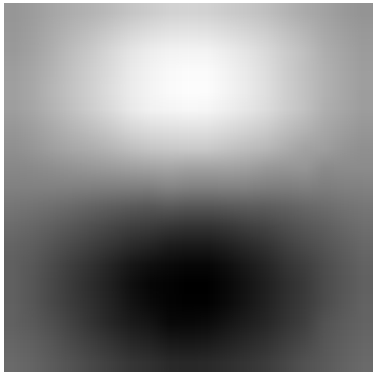


x-direction

y-direction

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

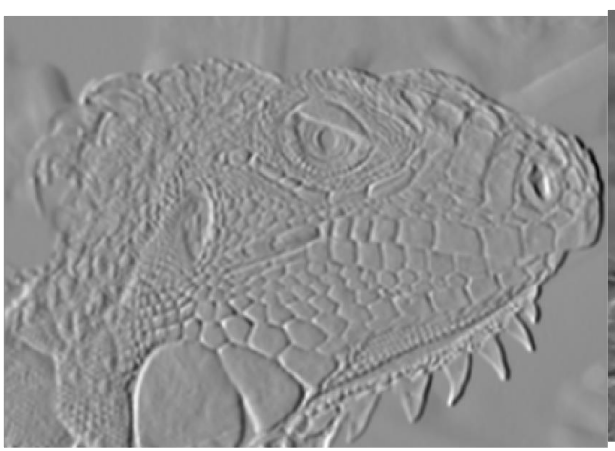




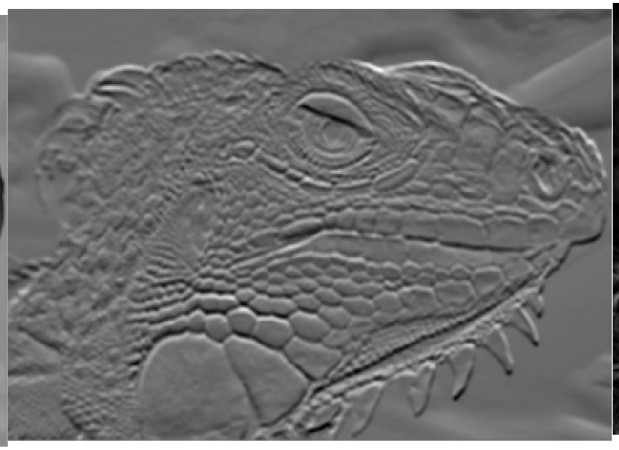
$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Canny edge detector

Compute gradients (DoG)



X-Derivative of Gaussian



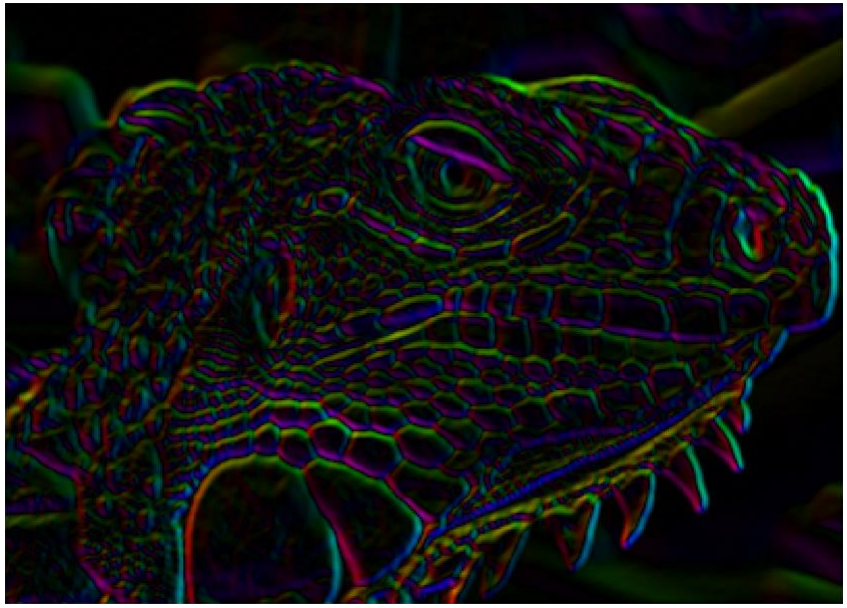
Y-Derivative of Gaussian

Gradient Magnitude

Canny edge detector

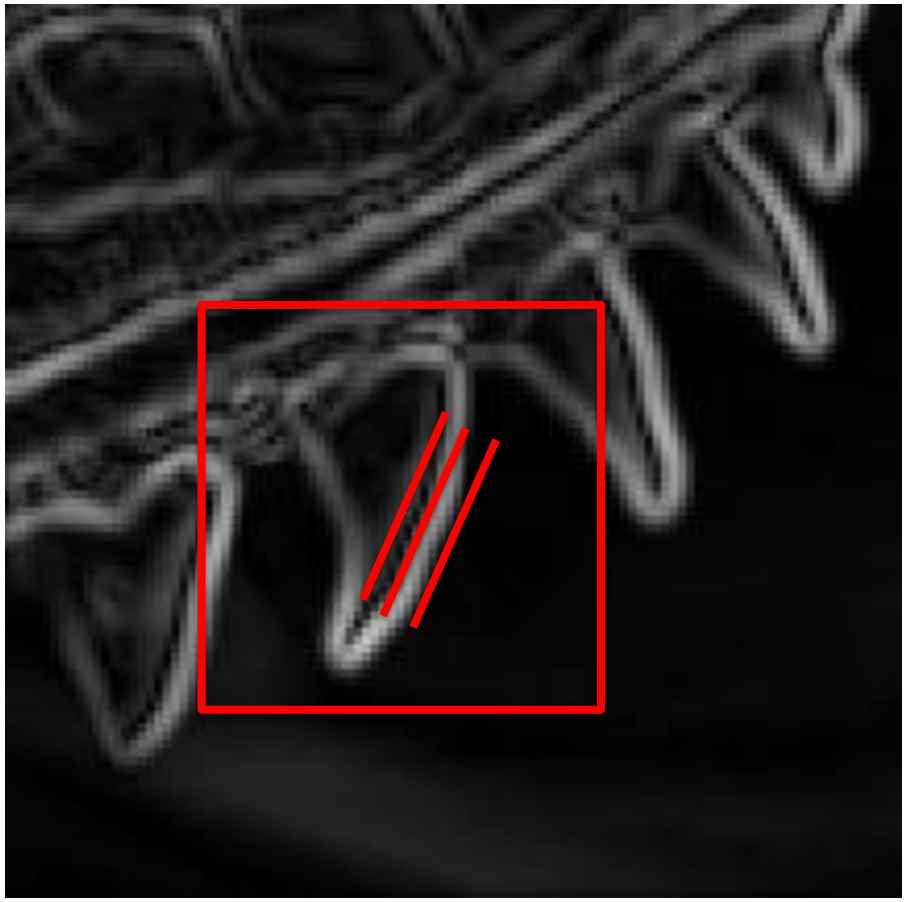
Get orientation at each pixel

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



Canny edge detector

Compute gradients (DoG)



Gradient Magnitude

Canny edge detector

- Suppress Noise
- Compute gradient magnitude and direction
- Apply Non-Maximum Suppression
  - Assures minimal response

Canny edge detector

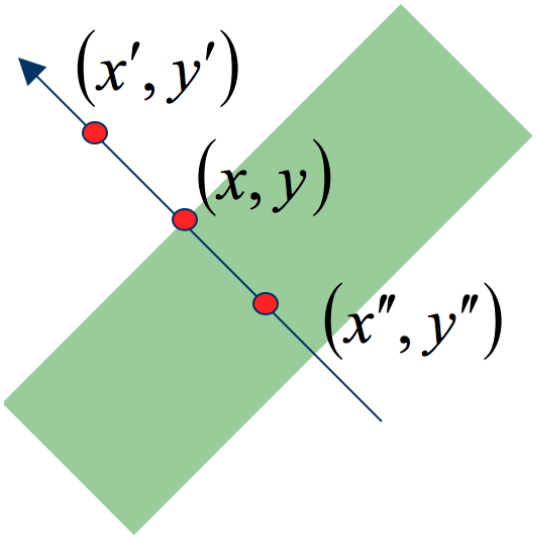
Non-maximum suppression

- To find the edge points, we need to find the local maxima of the gradient magnitude
- Suppress non-maxima gradient even if it passes threshold
- Only eight angle directions possible
  - Suppress all pixels in each direction which are not maxima
  - Do this in each marked pixel neighborhood

Canny edge detector

Remove spurious gradients

$|\nabla G|(x, y)$ is the gradient at pixel (x, y)



$$M(x,y) = \begin{cases} |\nabla G|(x,y) \text{ if } |\nabla G|(x,y) > |\nabla G|(x',y') \\ \quad\quad\quad\quad\quad\quad\quad\quad \& |\nabla G|(x,y) > |\nabla G|(x'',y'') \\ 0 \quad\quad\quad\quad\quad\quad\quad\quad \text{otherwise} \end{cases}$$

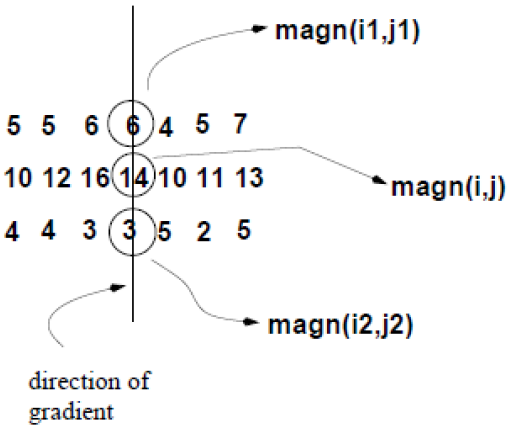
x' and **x**" are the neighbors of **x** along normal direction to an edge

Alper Yilmaz, Mubarak Shah Fall 2012, UCF

Canny edge detector

Non-maximum suppression

- To find the edge points, we need to find the local maxima of the gradient magnitude
- Suppress non-maxima gradient even if it passes threshold
- Only eight angle directions possible
  - Suppress all pixels in each direction which are not maxima
  - Do this in each marked pixel neighborhood



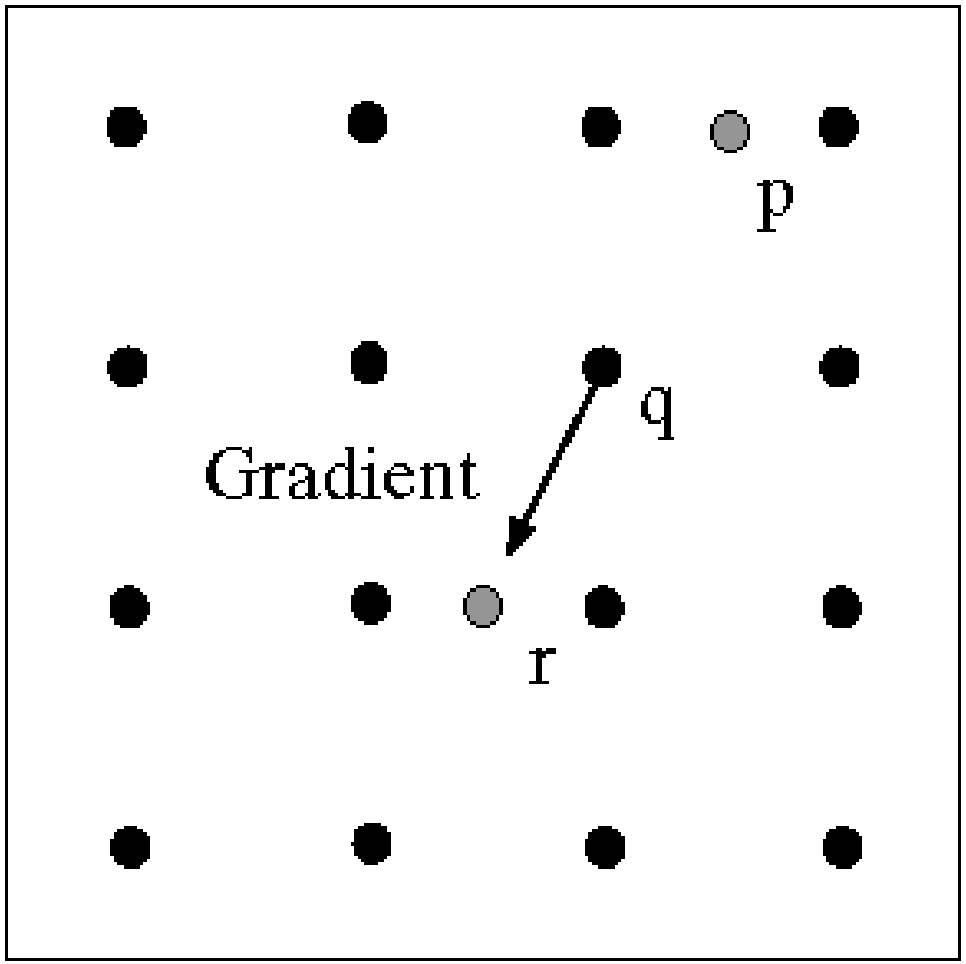
Algorithm

For each pixel (x,y) do:

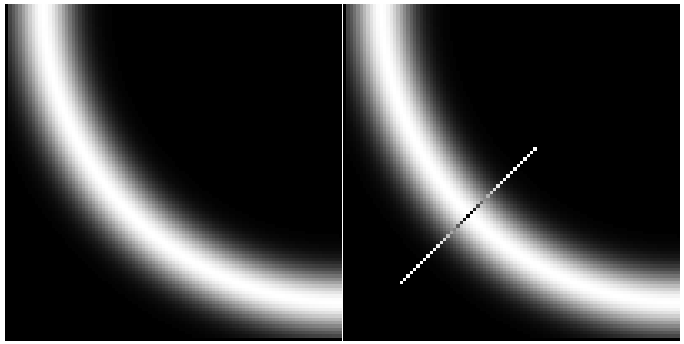
if $magn(i,j) < magn(i_1,j_1)$ or $magn(i,j) < magn(i_2,j_2)$
  then $I_N(i,j) = 0$
else $I_N(i,j) = magn(i,j)$

Canny edge detector

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.



Canny edge detector

Non-max Suppression

Before

After

Canny edge detector

- Suppress Noise
- Compute gradient magnitude and direction
- Apply Non-Maximum Suppression
  - Assures minimal response
- Use hysteresis and connectivity analysis to detect edges

Canny edge detector

Hysteresis thresholding

- Avoid streaking near threshold value
- Define two thresholds: Low and High
  - If less than Low, not an edge
  - If greater than High, strong edge
  - If between Low and High, weak edge

Canny edge detector

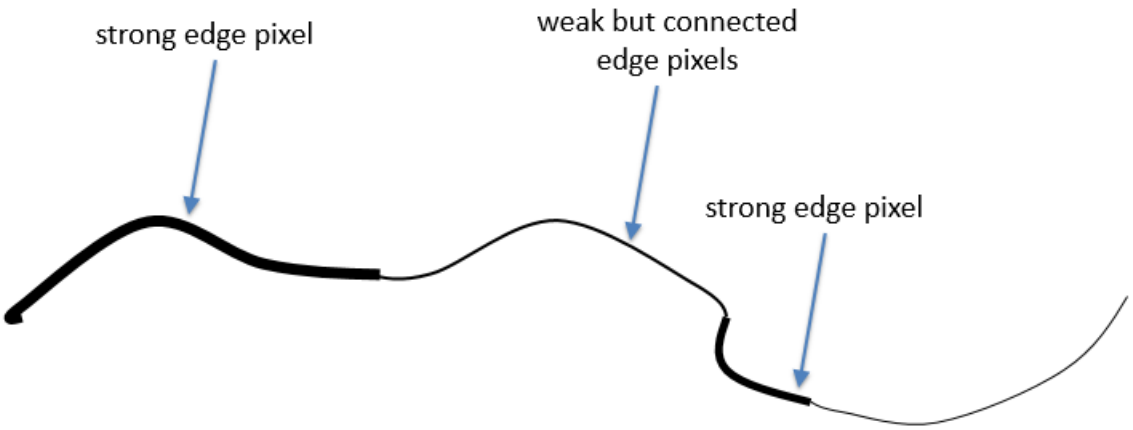
Hysteresis thresholding

If the gradient at a pixel is

- above High, declare it as an ‘strong edge pixel’
- below Low, declare it as a “non-edge-pixel”
- between Low and High
  - Consider its neighbors iteratively then declare it an “edge pixel” if it is connected to an ‘strong edge pixel’ directly or via pixels between Low and High

Canny edge detector

Hysteresis thresholding



Source: S. Seitz

Canny edge detector

Final Canny Edges

Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide “ridges” down to single pixel width
4. Thresholding and linking (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

Canny edge detector

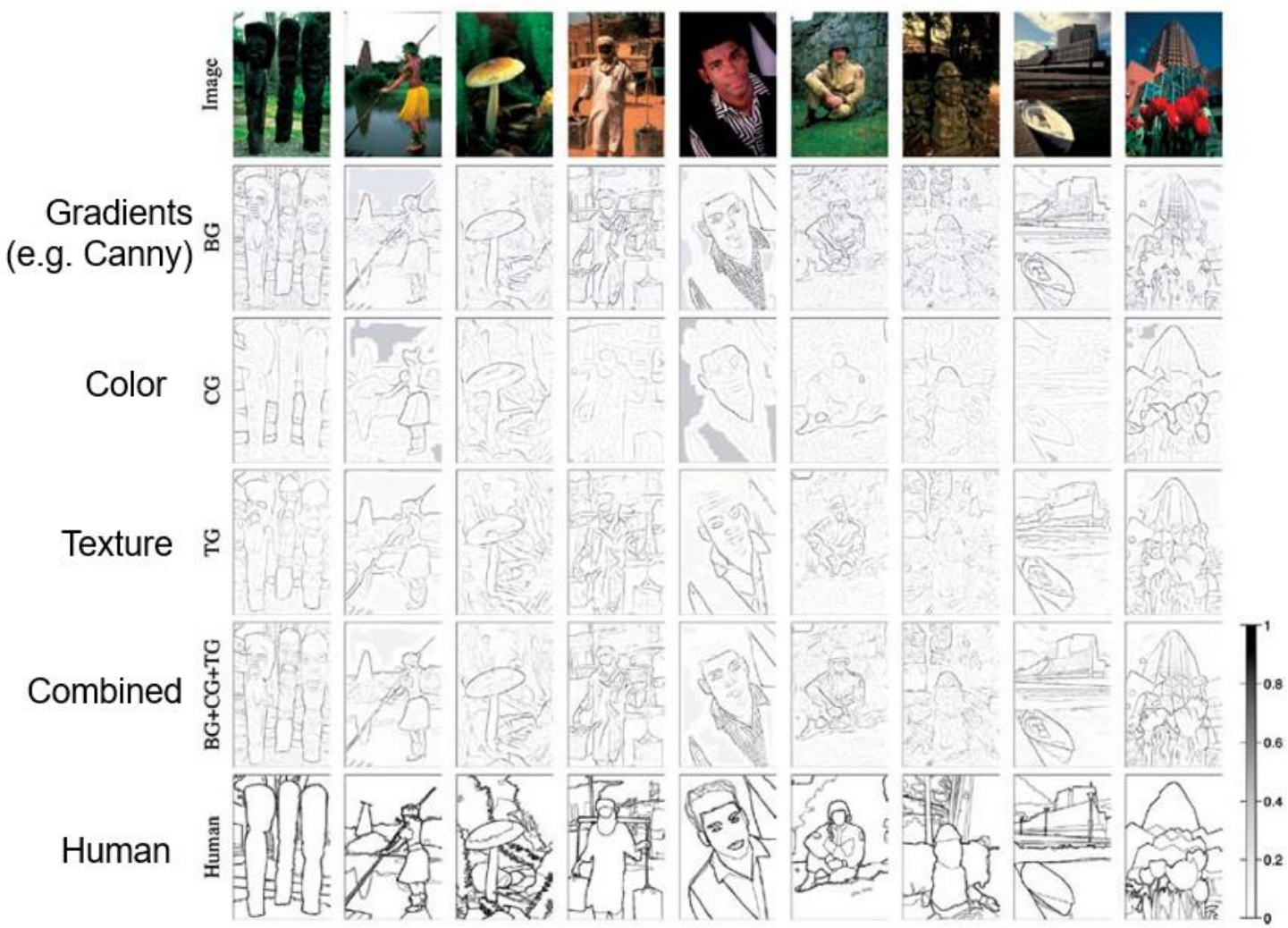
Effect of σ (Gaussian kernel spread/size)

original | Canny with $\sigma = 1$ | Canny with $\sigma = 2$

The choice of σ depends on desired behavior

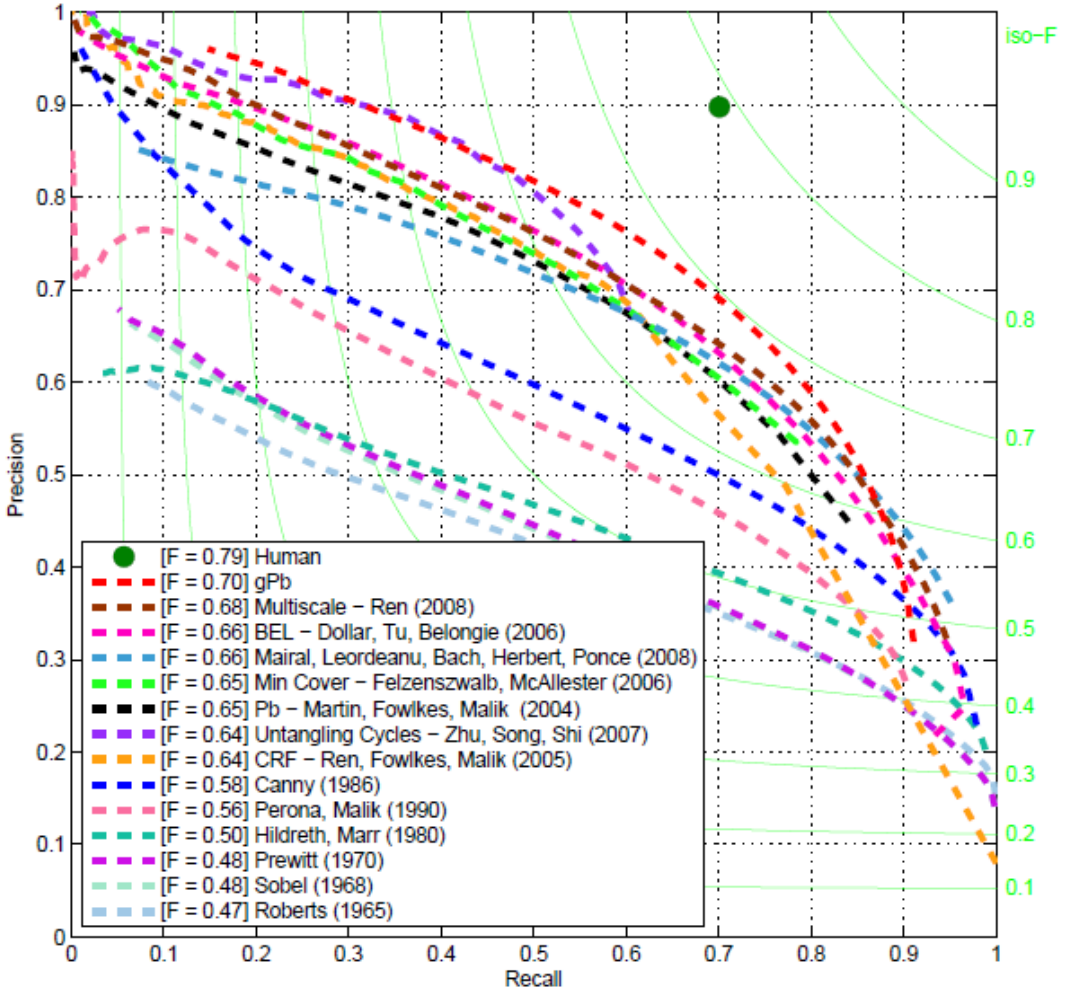
- large σ detects large scale edges
- small σ detects fine features

Canny edge detector



Canny edge detector

45 years of boundary detection

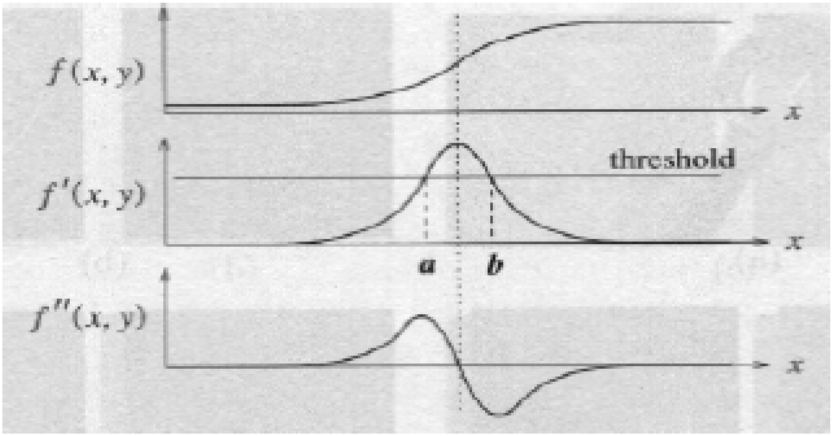


What will we learn today?

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel edge detector
- Canny edge detector
- Laplacian of Gaussian

Edge Detection Using the 2nd Derivative

- Edge points can be detected by finding the zero-crossings of the second derivative.



- There are two approaches that uses the second derivative to identify the edge presence
  - Smoothing then apply Gradient
  - Combine smoothing and Gradient Opertations

Edge Detection Using the 2nd Derivative

- Approximate finding maxima/minima of gradient magnitude by finding places where:

$$\frac{df^2}{dx^2}(x) = 0$$

- Can't always find discrete pixels where the second derivative is zero – look for zero-crossing instead.

Edge Detection Using the 2nd Derivative

- Computing the 2nd derivative:

$$f''(x) = \lim_{h \to 0} \frac{f'(x+h) - f'(x)}{h} \approx f'(x+1) - f'(x) =$$

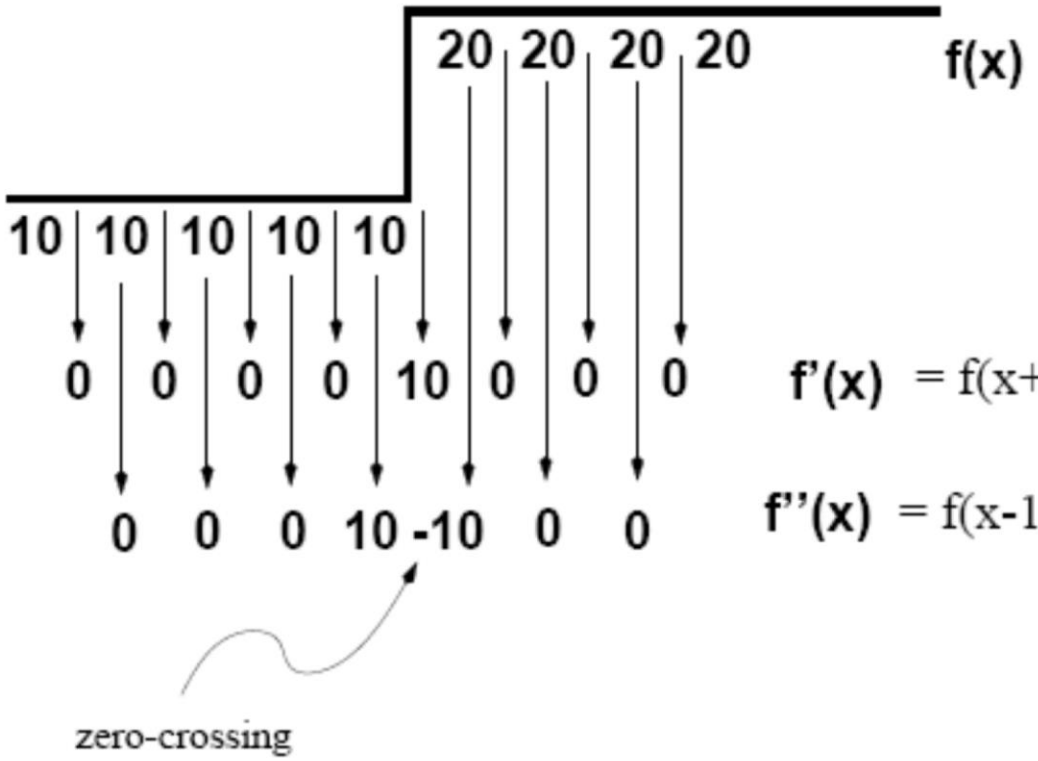
$$f(x+2) - 2f(x+1) + f(x) \quad (h=1)$$

  - This approximation is centered about $x + 1$
  - By replacing $x + 1$ by x we obtain:

$$f''(x) \approx f(x+1) - 2f(x) + f(x-1)$$

mask: [1 −2 1]

Edge Detection Using the 2nd Derivative



$f'(x) = f(x+1) - f(x)$ (approximates f'() at x+1/2)

$f''(x) = f(x-1)-2f(x)+f(x+1)$ (approximates f''() at x)

Edge Detection Using the 2nd Derivative

- Four cases of zero-crossings:

{+,-}, {+,0,-},{-,+}, {-,0,+}

- **Slope** of zero-crossing {a, -b} is: |a+b|.

- To detect "strong" zero-crossing, threshold the slope.

Second Derivative in 2D: Laplacian

The *Laplacian* is defined mathematically as

$$\nabla^2 = \nabla \cdot \nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

When we apply it to an image, we get

$$\nabla^2 f = \left(\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \right) I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Second Derivative in 2D: Laplacian

$$\frac{\partial^2 f}{\partial x^2} = f(i, j+1) - 2f(i, j) + f(i, j-1)$$

$$\frac{\partial^2 f}{\partial y^2} = f(i+1, j) - 2f(i, j) + f(i-1, j)$$

$$\nabla^2 f = -4f(i, j) + f(i, j+1) + f(i, j-1) + f(i+1, j) + f(i-1, j)$$

0	0	0
1	-2	1
0	0	0

+

0	1	0
0	-2	0
0	1	0

=

0	1	0
1	-4	1
0	1	0

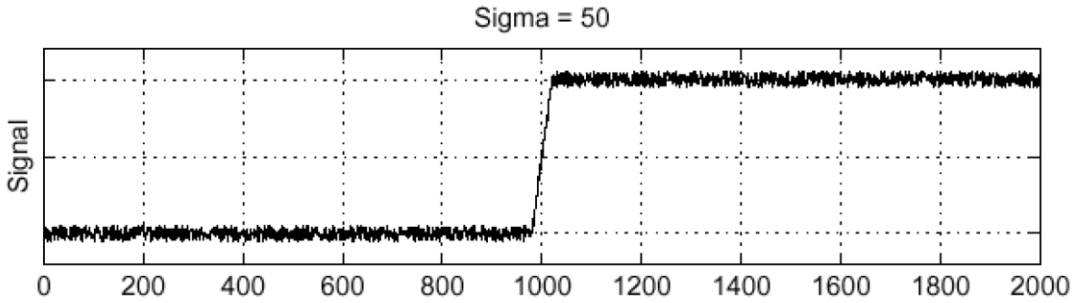
Properties of Laplacian

- It is cheaper to implement than the gradient (i.e., one mask only).
- It does not provide information about edge direction.
- It is more sensitive to noise (i.e., differentiates twice).
- It is an isotropic operator (equal weights in all directions)

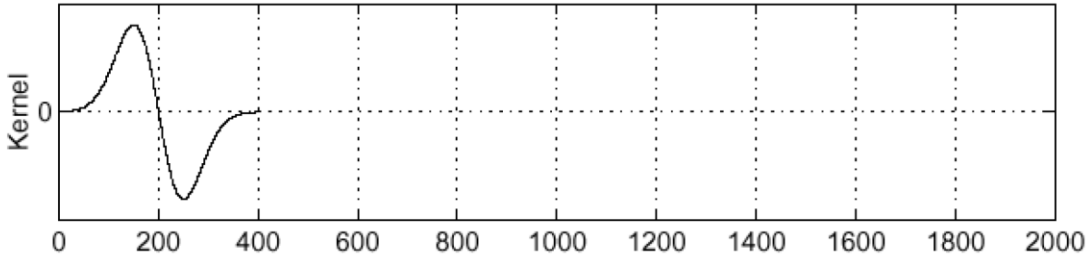
Combine Smoothing with Differentiation

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$ (i.e., saves one operation)

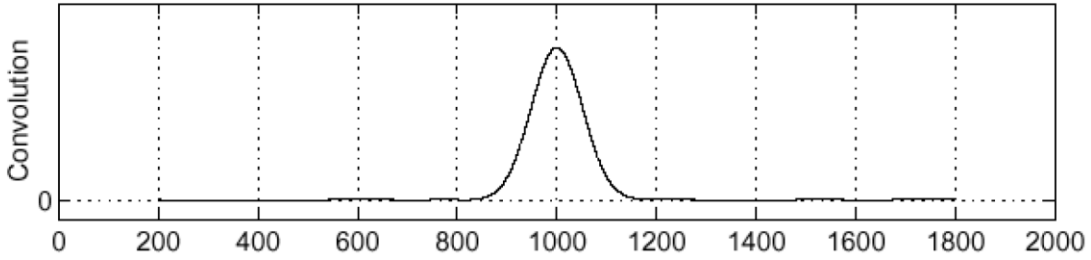
f



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



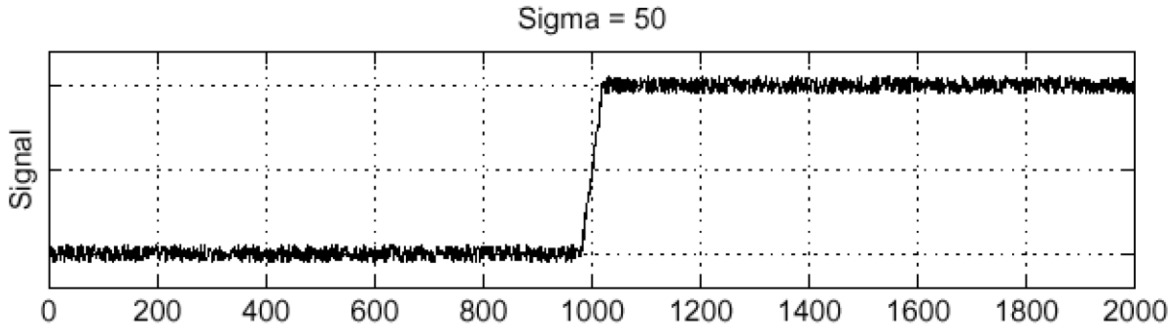
$(\frac{\partial}{\partial x}h) \star f$



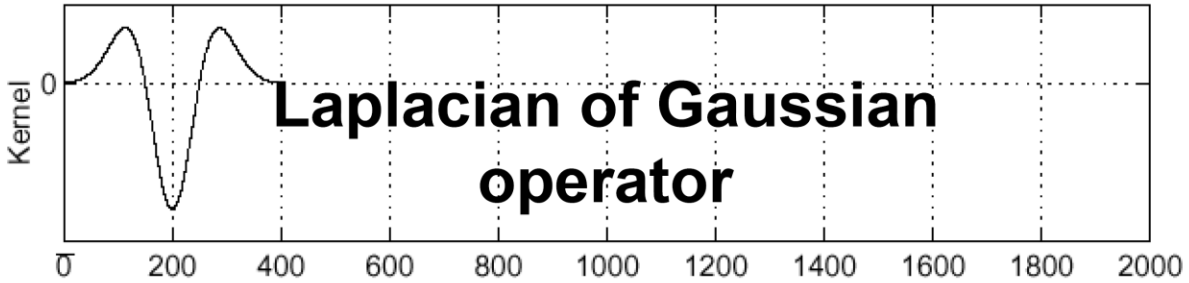
Laplacian of Gaussian

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

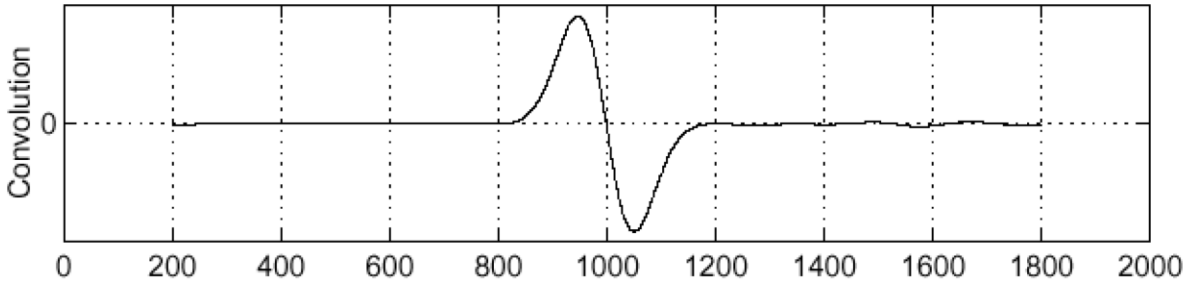
f



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



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



Laplacian of Gaussian (LoG)

- To reduce the noise effect, the image is first smoothed.
- When the filter chosen is a Gaussian, we call it the LoG edge detector. (Marr-Hildreth operator)

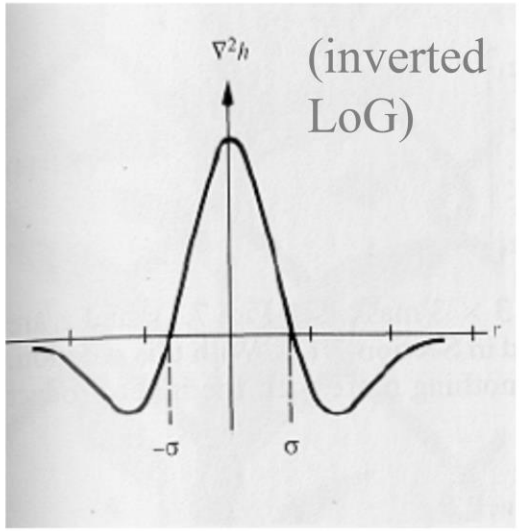
$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

σ controls smoothing

- It can be shown that:

$$\nabla^2[f(x,y) * G(x,y)] = \nabla^2 G(x,y) * f(x,y)$$

$$\nabla^2 G(x,y) = \left(\frac{r^2 - 2\sigma^2}{\sigma^4}\right) e^{-r^2/2\sigma^2}, \quad (r^2 = x^2 + y^2)$$



Laplacian of Gaussian (LoG)

(inverted LoG)

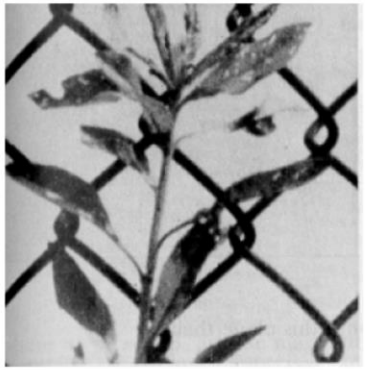
5 × 5 Laplacian of Gaussian mask

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

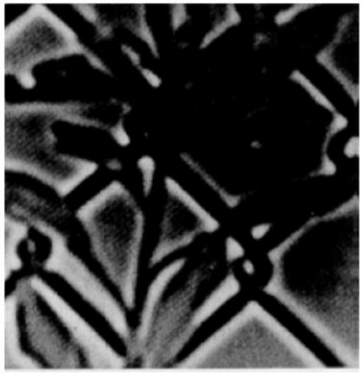
(inverted LoG)

17 × 17 Laplacian of Gaussian mask

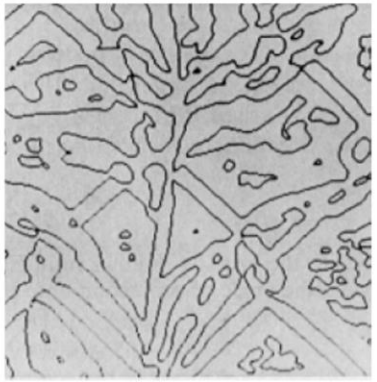
0	0	0	0	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	0
0	0	0	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	0	0
0	0	−1	−1	−1	−2	−3	−3	−3	−3	−3	−2	−1	−1	−1	0	0
0	0	−1	−1	−2	−3	−3	−3	−3	−3	−3	−3	−2	−1	−1	0	0
0	−1	−1	−2	−3	−3	−3	−2	−3	−2	−3	−3	−3	−2	−1	−1	0
0	−1	−2	−3	−3	−3	0	2	4	2	0	−3	−3	−3	−2	−1	0
−1	−1	−3	−3	−3	0	4	10	12	10	4	0	−3	−3	−3	−1	−1
−1	−1	−3	−3	−2	2	10	18	21	18	10	2	−2	−3	−3	−1	−1
−1	−1	−3	−3	−3	4	12	21	24	21	12	4	−3	−3	−3	−1	−1
−1	−1	−3	−3	−2	2	10	18	21	18	10	2	−2	−3	−3	−1	−1
−1	−1	−3	−3	−3	0	4	10	12	10	4	0	−3	−3	−3	−1	−1
0	−1	−2	−3	−3	−3	0	2	4	2	0	−3	−3	−3	−2	−1	0
0	−1	−1	−2	−3	−3	−3	−2	−3	−2	−3	−3	−3	−2	−1	−1	0
0	0	−1	−1	−2	−3	−3	−3	−3	−3	−3	−3	−2	−1	−1	0	0
0	0	−1	−1	−1	−2	−3	−3	−3	−3	−3	−2	−1	−1	−1	0	0
0	0	0	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	0	0
0	0	0	0	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	0



filtering



zero-crossings

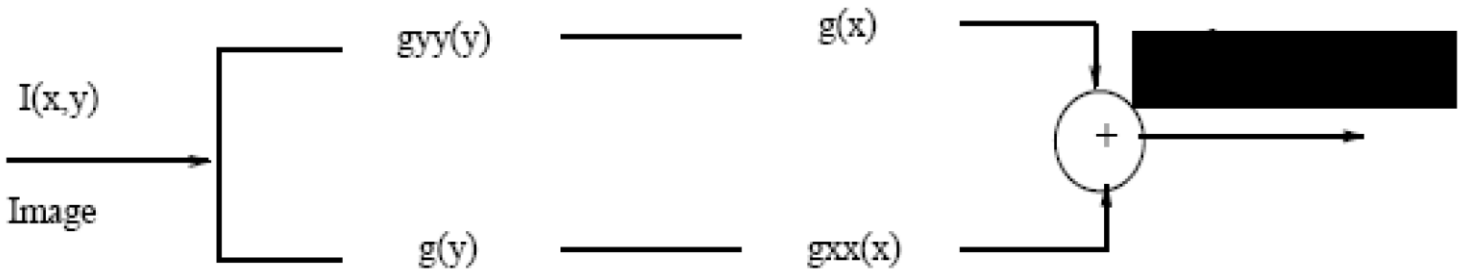


Decomposition of LoG

- It can be shown than LoG can be written as follows:

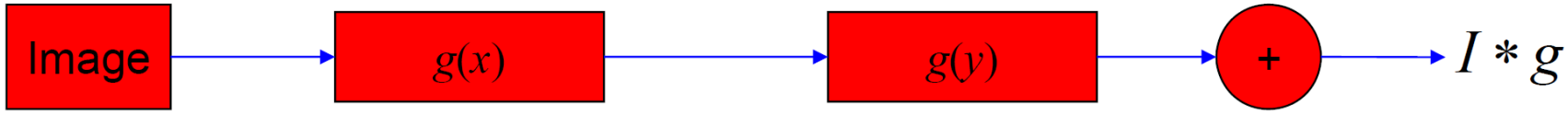
$$\nabla^2 g(x,y) = \frac{\partial}{\partial y^2} g(y) * g(x) + g(y) * \frac{\partial}{\partial x^2} g(x).$$

- 2D LoG convolution can be implemented using 4, 1D convolutions.

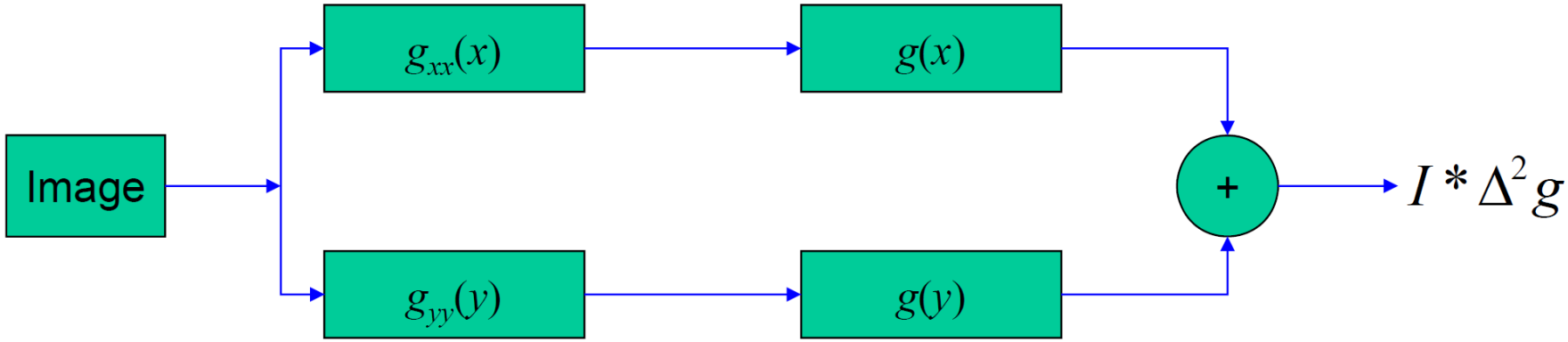


Decomposition of LoG

Gaussian Filtering



Laplacian-of-Gaussian Filtering



Edge Detection Using LOG

- Marr-Hildteth (LOG) Algorithm:
  - Compute LoG
    - Use one 2D filter: $\Delta^2 g(x,y)$
    - Use four 1D filters: $g(x),\ g_{xx}(x),\ g(y),\ g_{yy}(y)$
  - Find zero-crossings from each row and column
  - Find slope of zero-crossings
  - Apply threshold to slope and mark edges

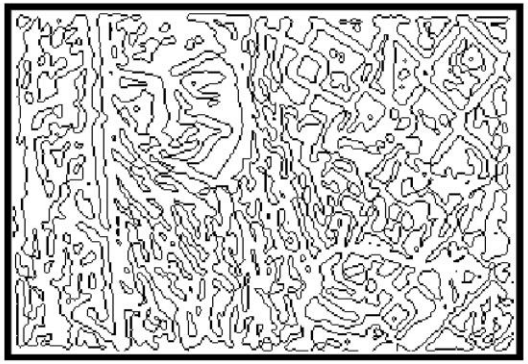
Edge Detection Using LOG

- The Laplacian-of-Gaussian (LOG) – cont.

I

$I * (\Delta^2 G)$

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



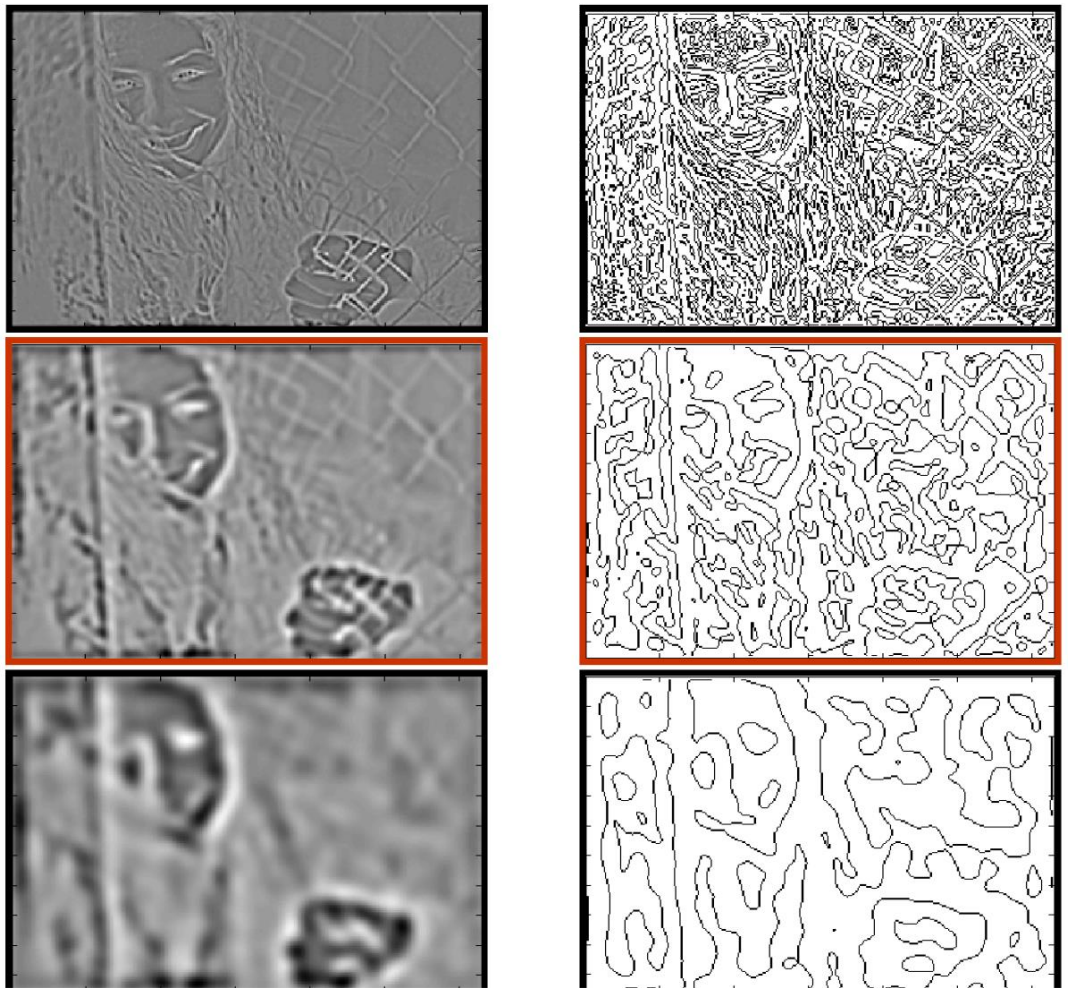
Edge Detection Using LOG

- The Laplacian-of-Gaussian (LOG) – cont.

$\sigma = 1$

$\sigma = 3$

$\sigma = 6$



What we have learned today?

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel Edge detector
- Canny edge detector