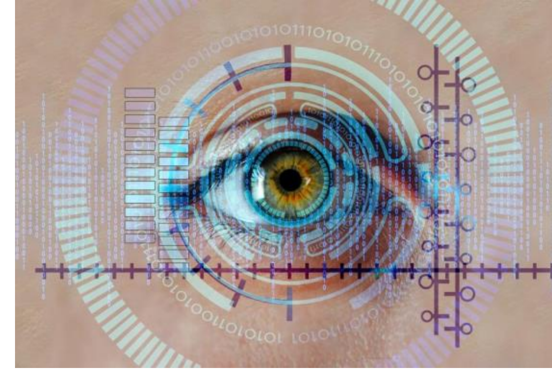
北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY



Computer Vision

Lecture 5 Edge Detection

School of Computer Science and Technology

Ying Fu

fuying@bit.edu.cn

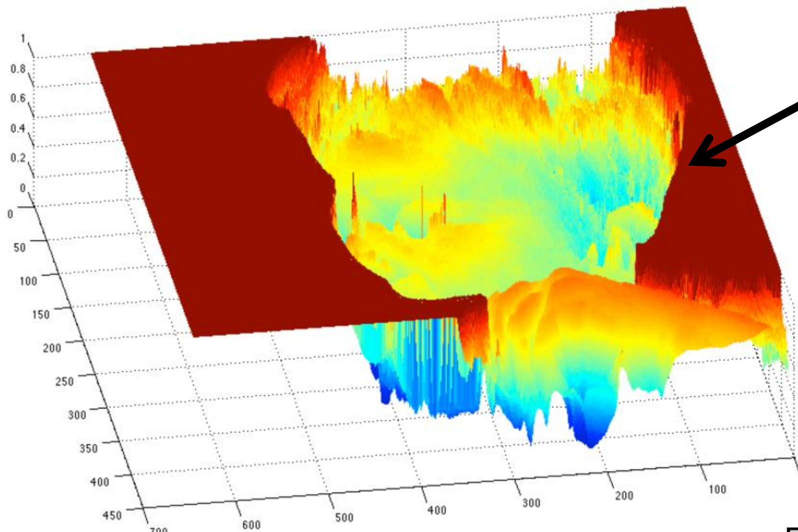
Outline

- Overview
- Image gradients
- A simple edge detector
- Sobel edge detector
- Canny edge detector

What are image edges?

grayscale image

$f(\boldsymbol{x})$

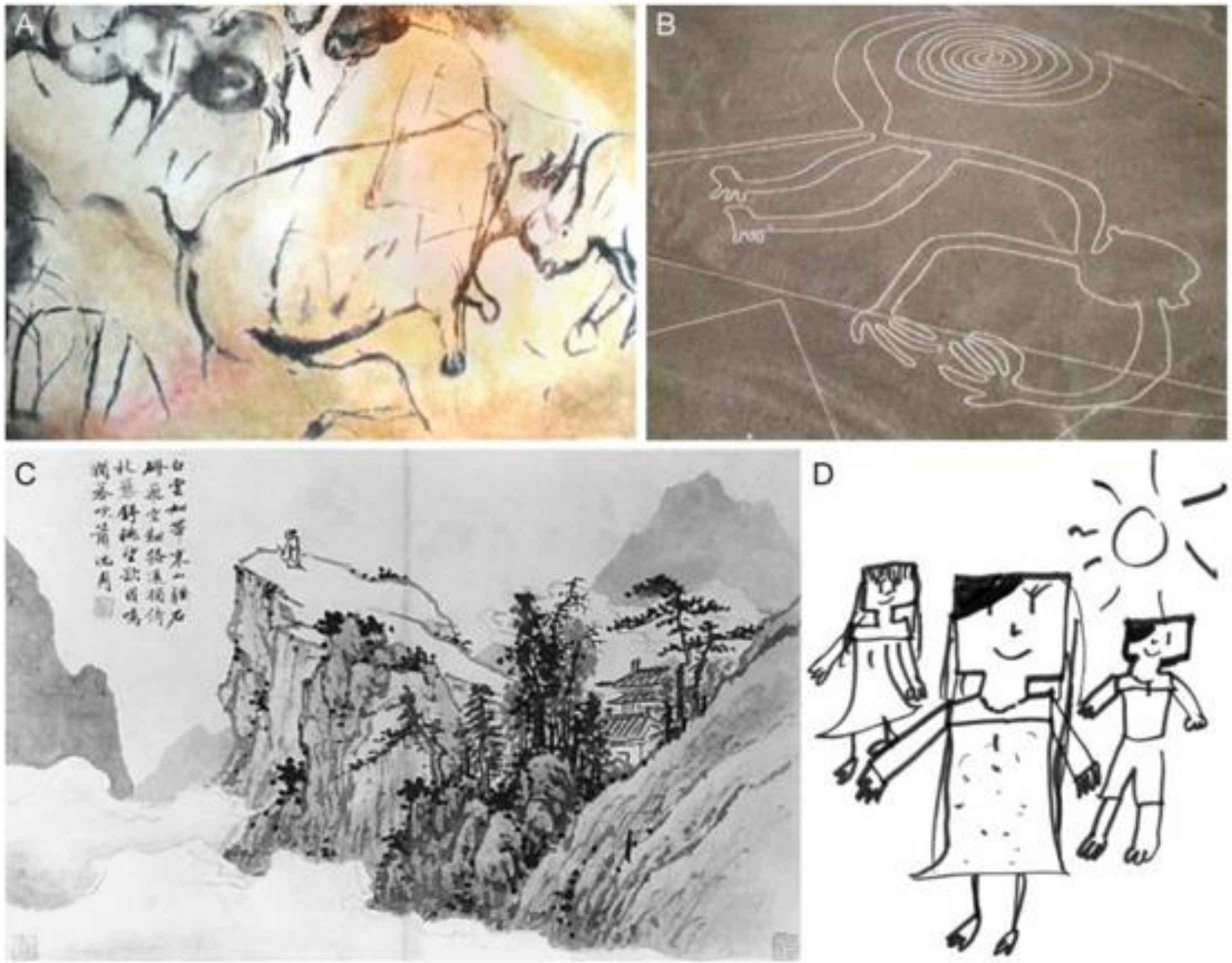


Very sharp discontinuities in intensity.

domain $\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

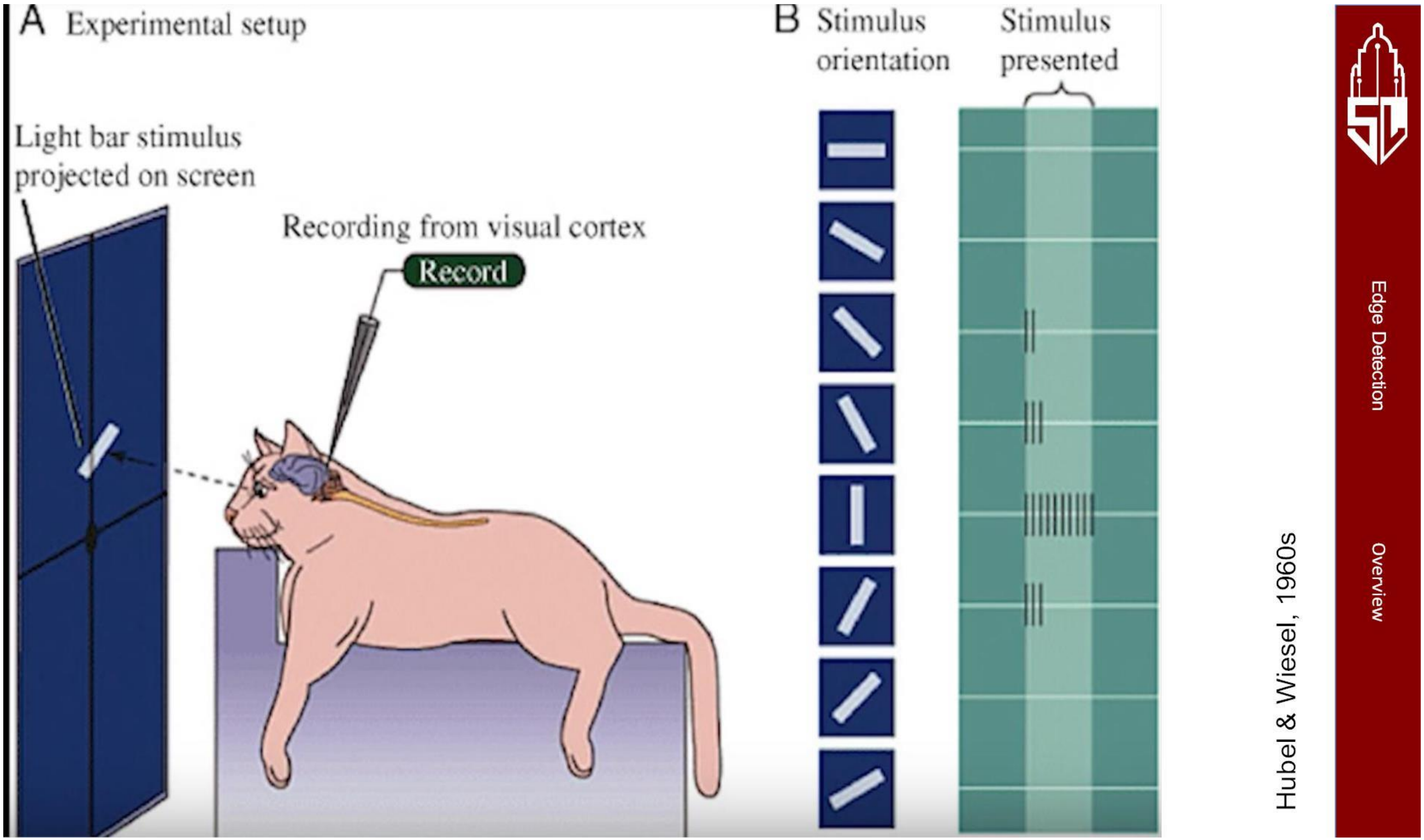
Edges

Edges



(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.).

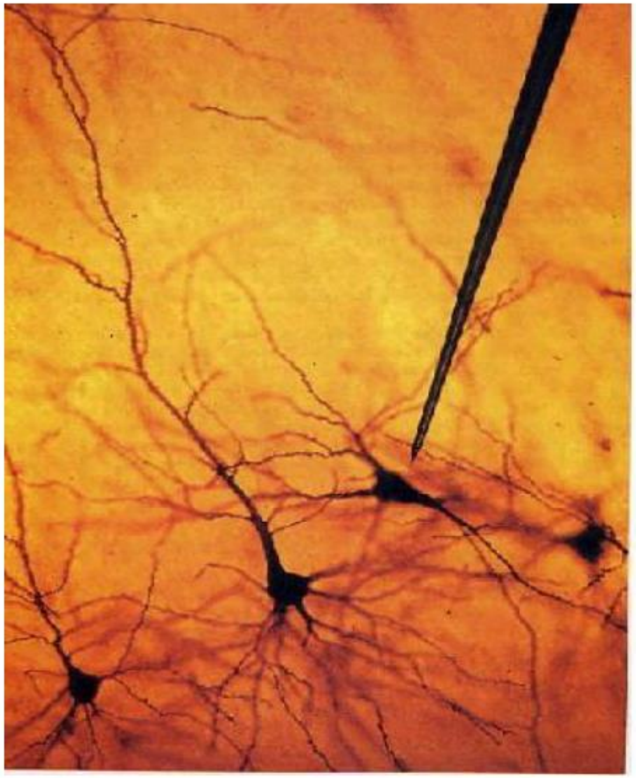
Edges

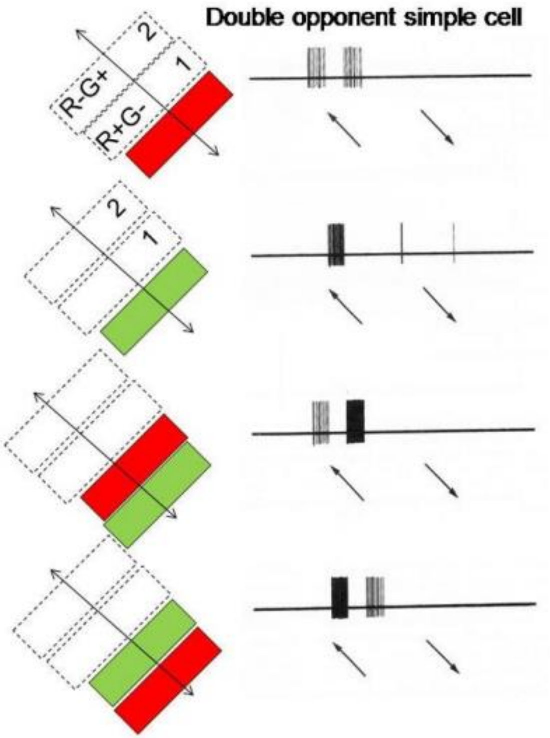


Hubel & Wiesel, 1960s

Edges

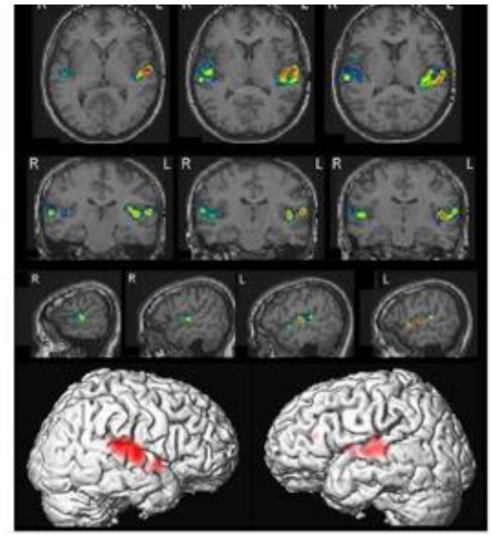
- We know edges are special from human (mammalian) vision studies

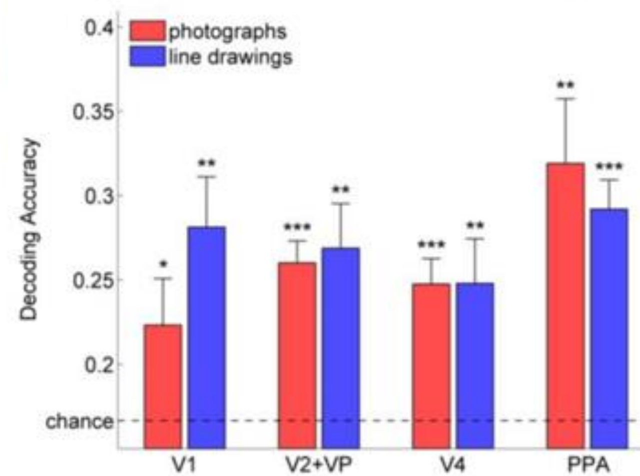




Hubel & Wiesel, 1960s

Edges





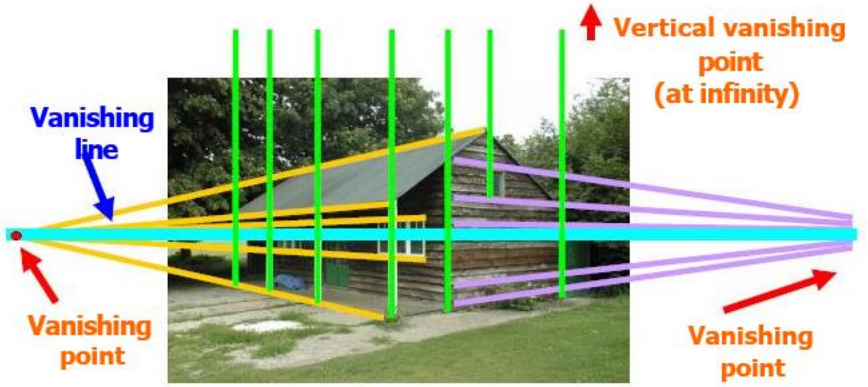
Walther, Chai, Caddigan, Beck & Fei-Fei, *PNAS, 2011*

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
  - Recover geometry and viewpoint



Origin of edges

- surface normal discontinuity
- depth discontinuity
- surface color discontinuity
- Illumination discontinuity

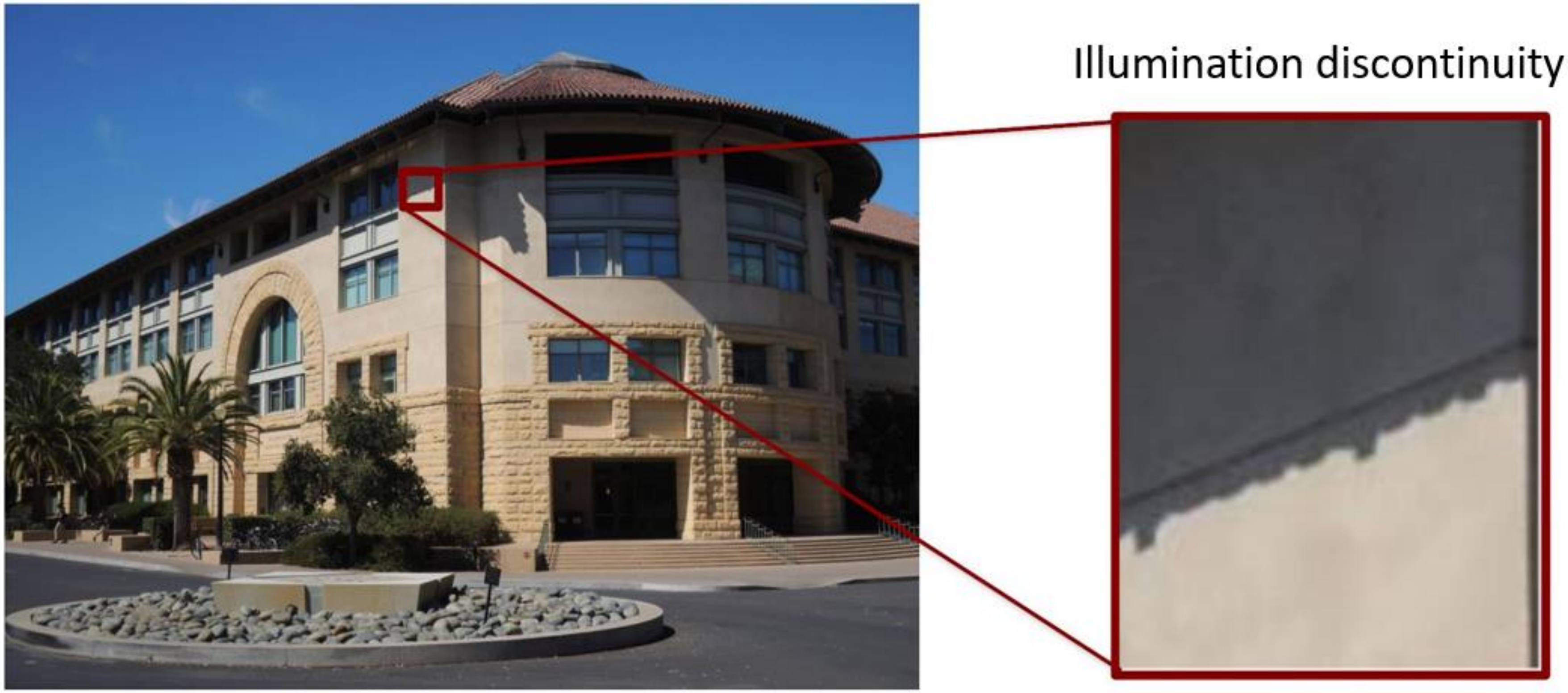
Closeup of edges

Surface normal discontinuity

Closeup of edges

Closeup of edges

Closeup of edges



Edge detection

- How would you go about detecting edges in an image (i.e., discontinuities in a function)?

✓ You take derivatives: derivatives are large at discontinuities.

- How do you differentiate a discrete image (or any other discrete signal)?

✓ You use (finite) differences.

Outline

- Overview
- Image gradients
- A simple edge detector
- Sobel edge detector
- Canny edge detector

Image gradients

- Derivatives in 1D

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

Image gradients

- Derivatives in 1D — example

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

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

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

$$f(x) - f(x-1) = f'(x)$$ [0 1 -1]

  - Forward filter

$$f(x) - f(x+1) = f'(x)$$ [-1 1 0]

  - Central filter

$$f(x+1) - f(x-1) = f'(x)$$ [1 0 -1]

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

Image gradients

- 2D discrete derivative filters

What about this filter?

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

- in what direction do x and y increase?

Image gradients

- 2D discrete derivative — example

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

What happens when we apply this filter?

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

Image gradients

- 2D discrete derivative — example

What happens when we apply this filter?

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \Rightarrow I_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

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

Image gradients

- 2D discrete derivative — example

Now let's try the other filter!

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

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

Image gradients

- 2D discrete derivative — example

What happens when we apply this filter?

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

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

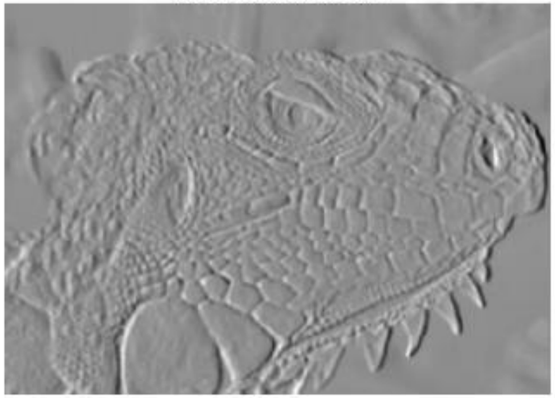
$$\Rightarrow \quad I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Image gradients

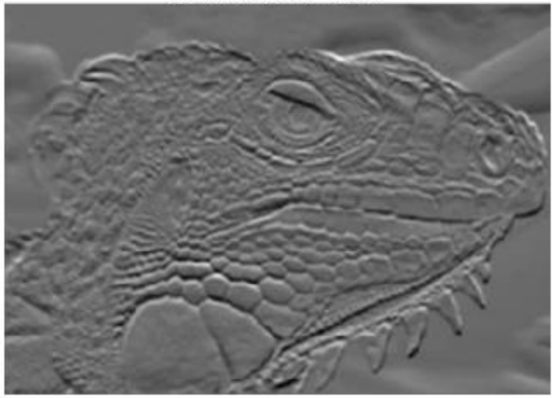
- 3x3 image gradient filters

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

Derivative in x direction



Derivative in y direction

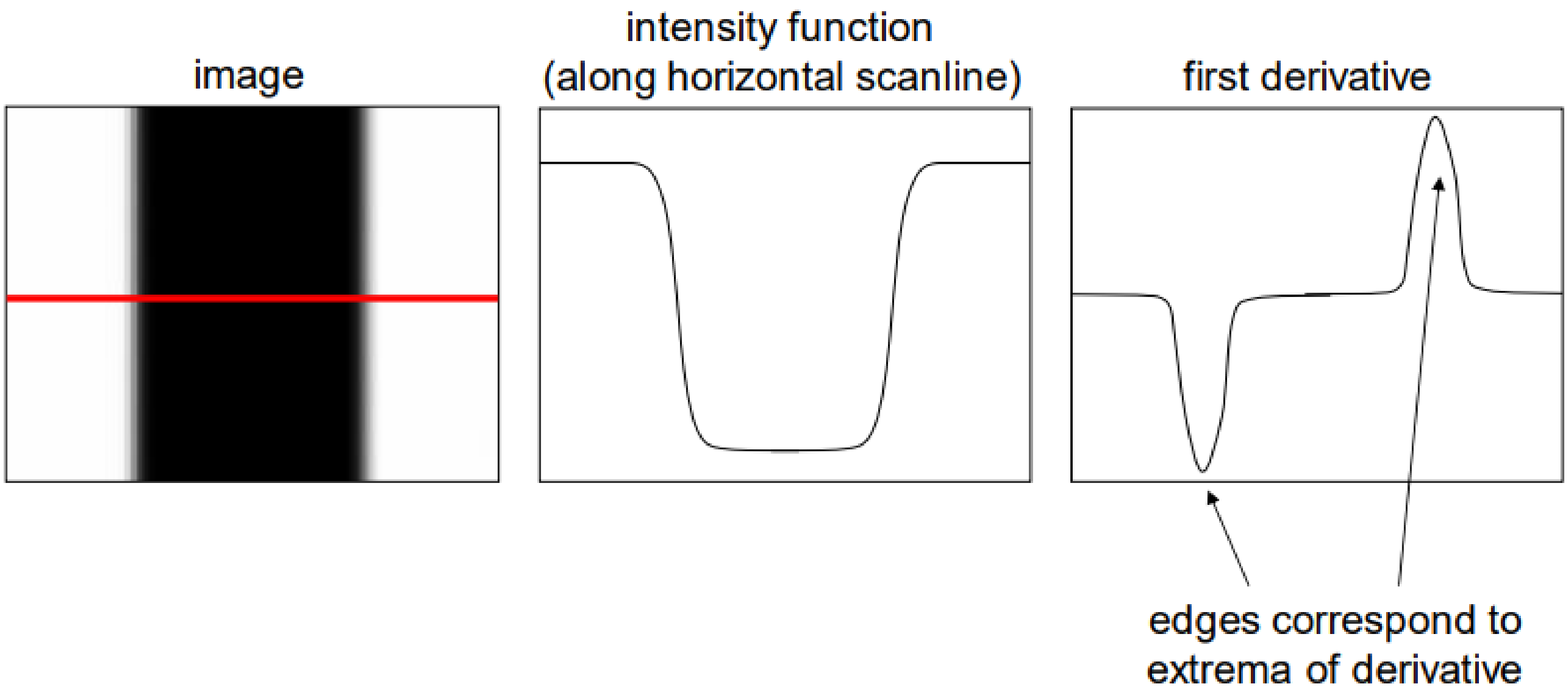


Outline

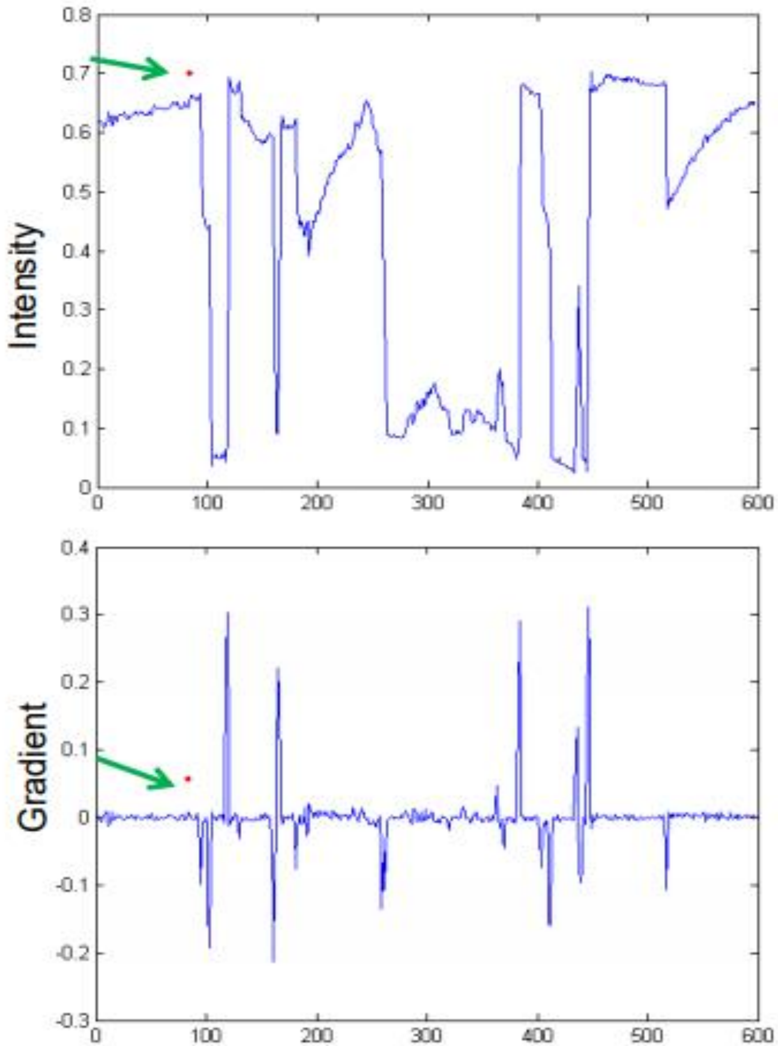
- Overview
- Image gradients
- A simple edge detector
- Sobel edge detector
- Canny edge detector

Characterizing edges

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



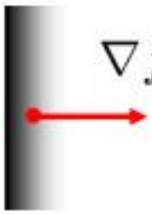
Intensity profile



Source: D. Hoiem

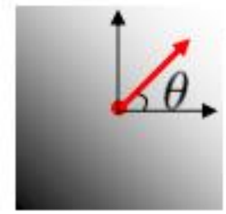
Image gradient

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



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

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



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

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

The gradient direction is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

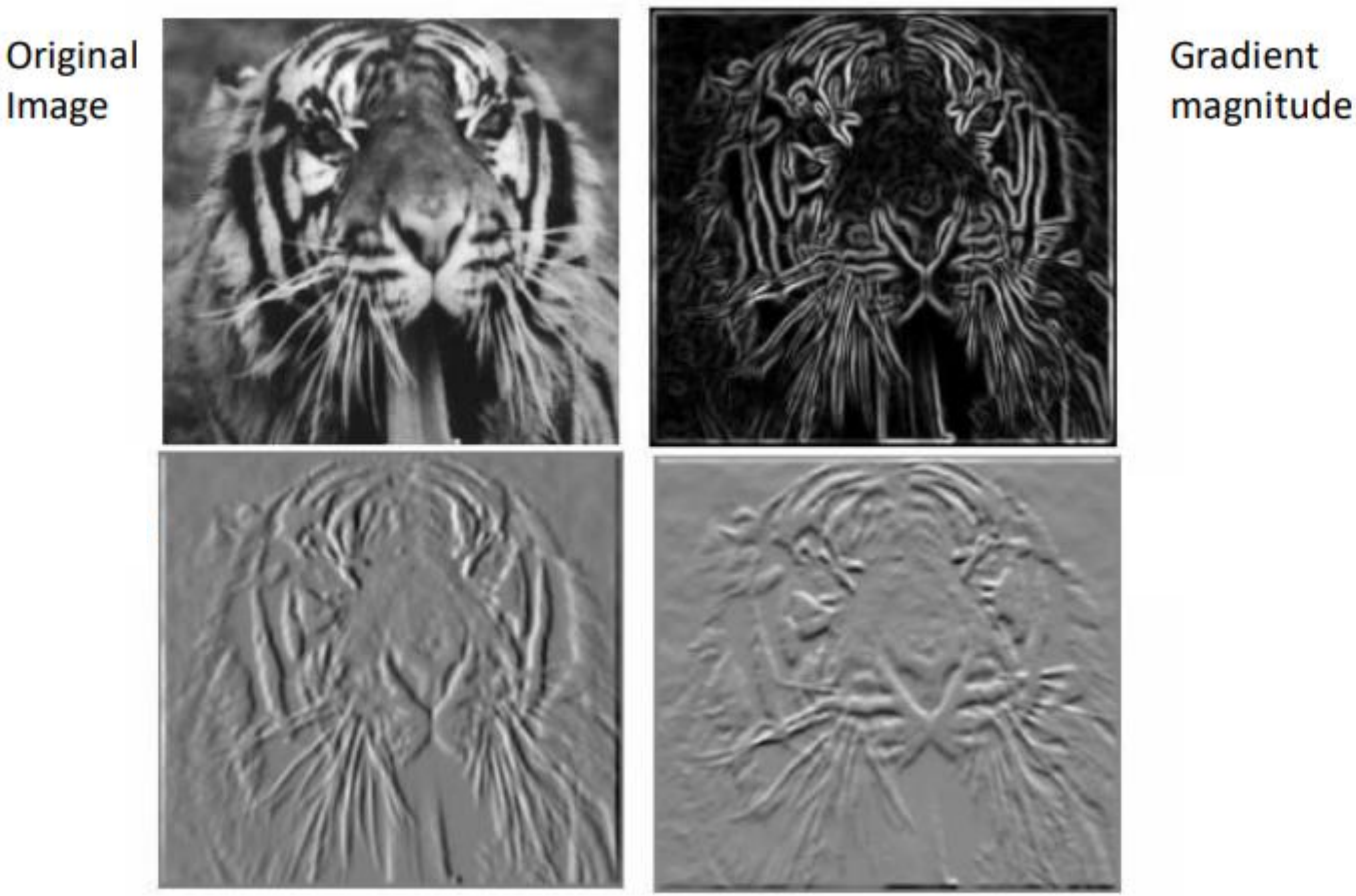
- how does this relate to the direction of the edge?

The *edge strength* is given by the gradient magnitude

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

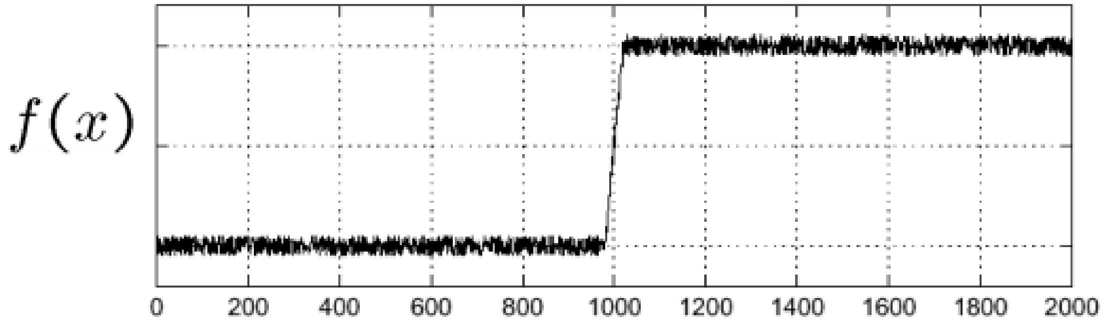
Discrete derivative/gradient: example

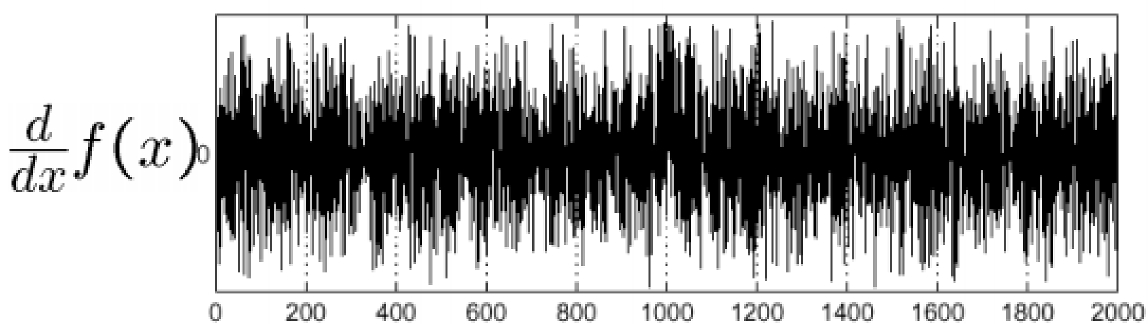
- Which one is the gradient in the x-direction? How about y-direction?



Effects of noise

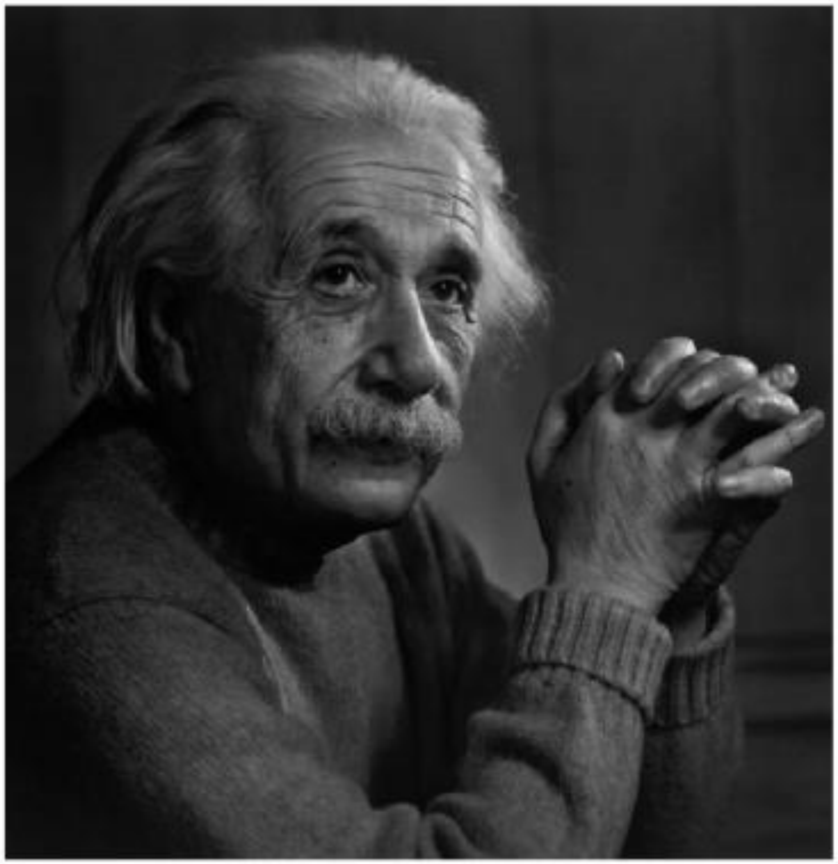
- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

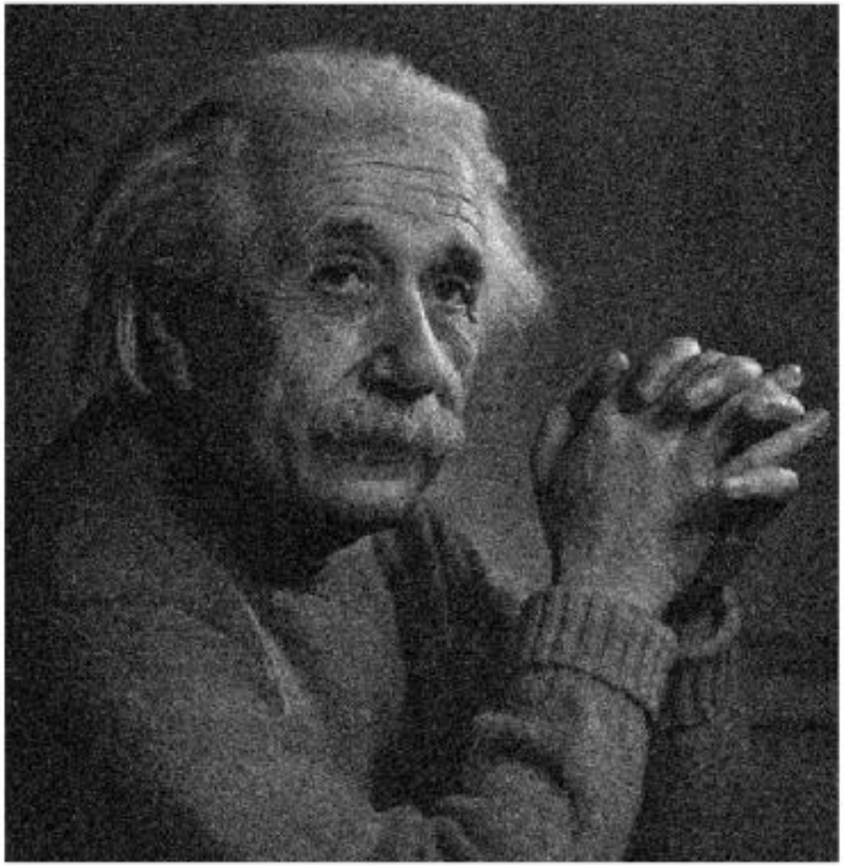




Where is the edge?

Effects of noise





Effects of noise

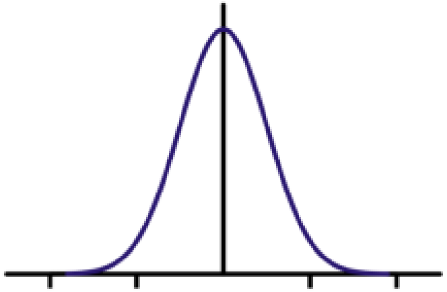
- Discrete gradient filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What is to be done?
  - Smoothing the image should help, by forcing pixels different to their neighbors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

Smoothing filters

- Mean smoothing

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

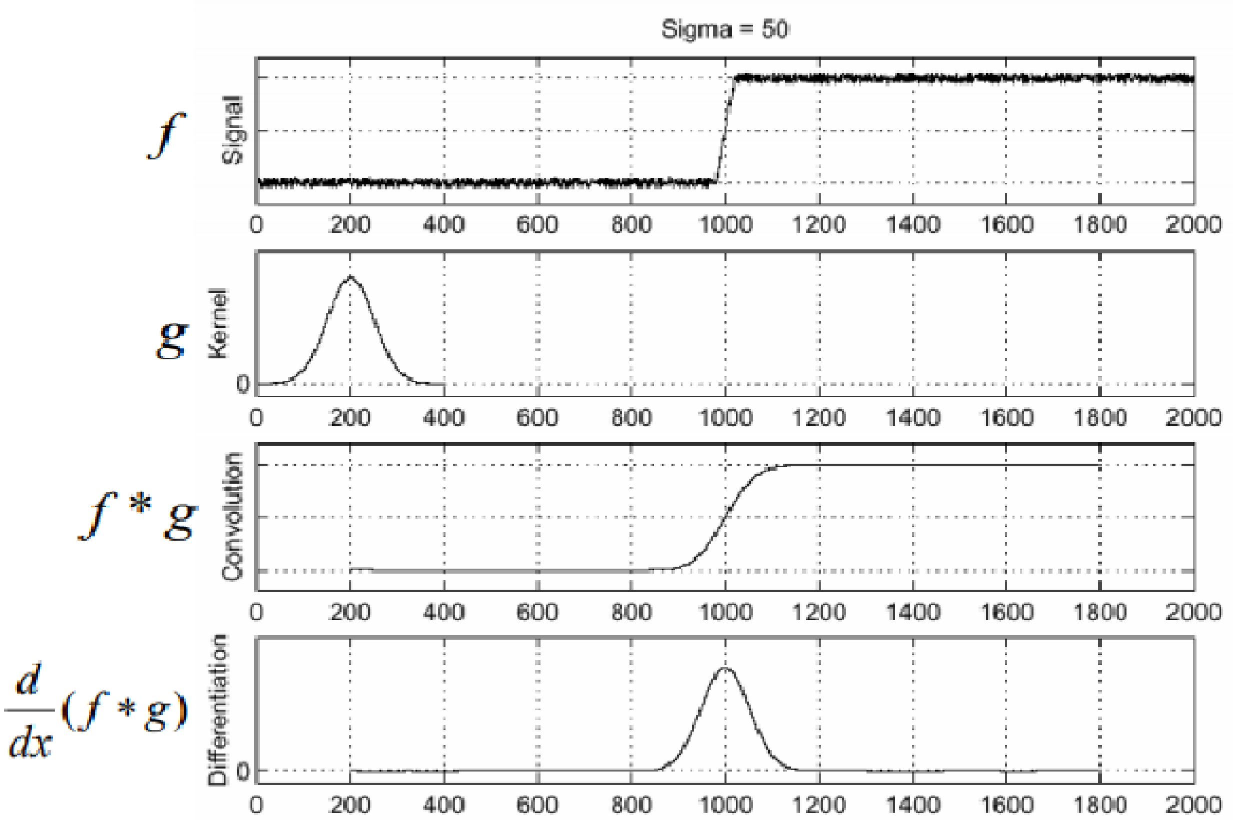
- Gaussian smoothing



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

A simple edge detector

- Smoothing + peaks



- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Derivative theorem of convolution

- This theorem gives us a very useful property

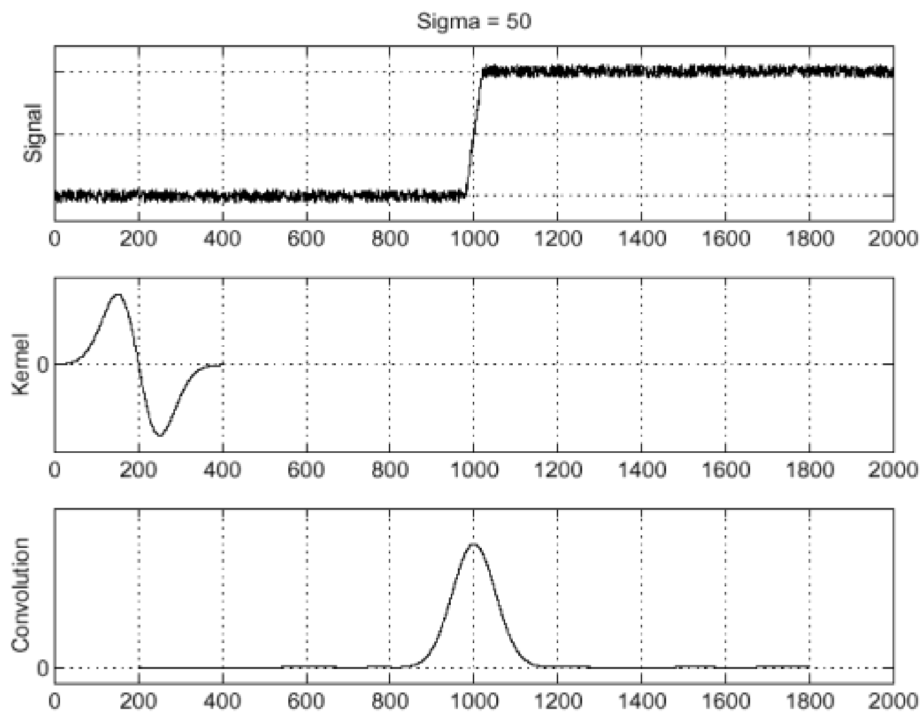
$$\frac{d}{dx}(f*g) = f*\frac{d}{dx}g$$

- This saves us one operation

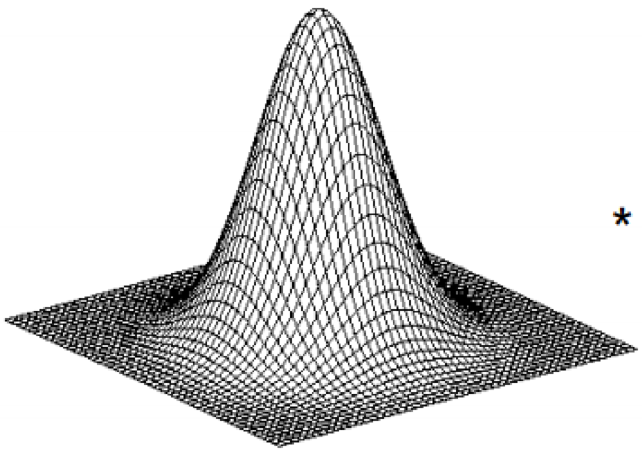
f

$\frac{d}{dx}g$

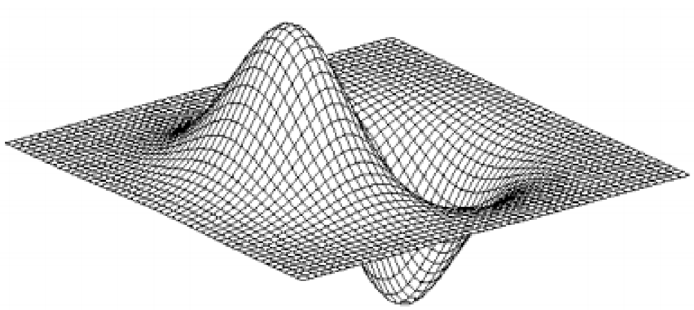
$f*\frac{d}{dx}g$



Derivative of Gaussian (DOG) filter



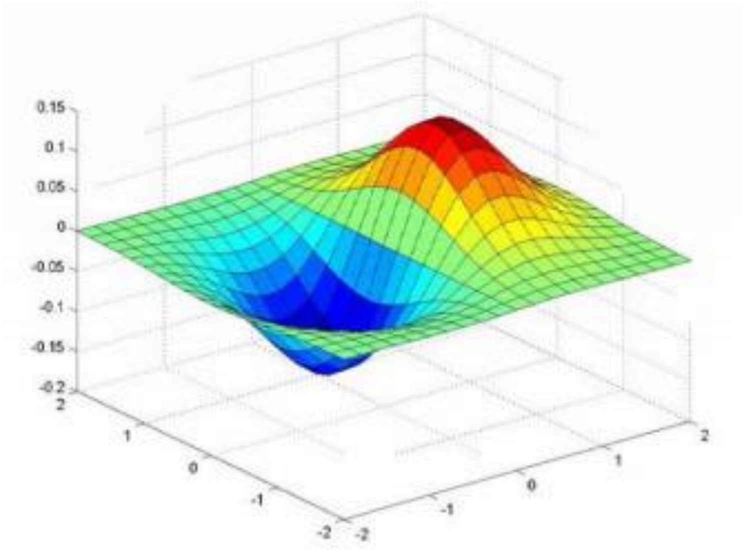
* [1 0 -1] =



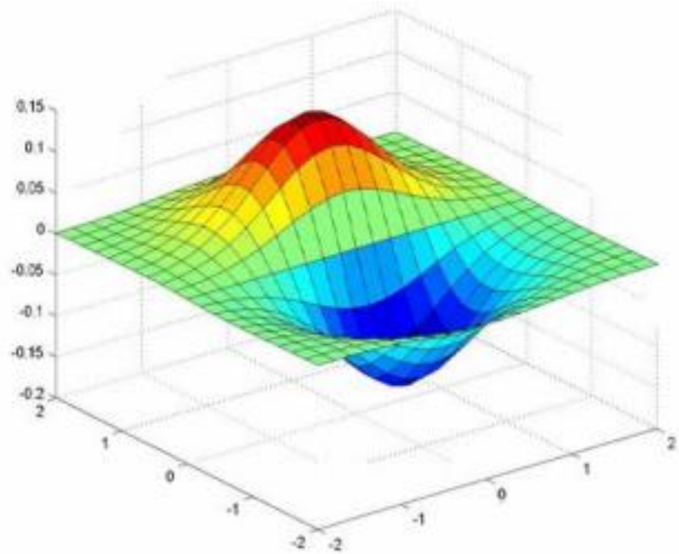
2D-gaussian

x - derivative

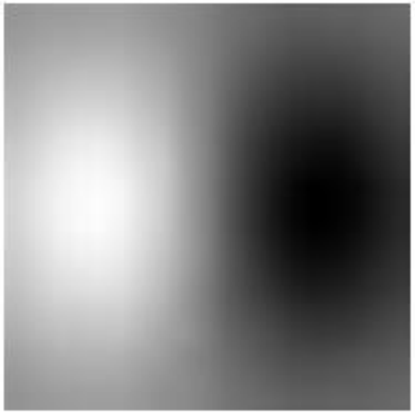
Derivative of Gaussian filter

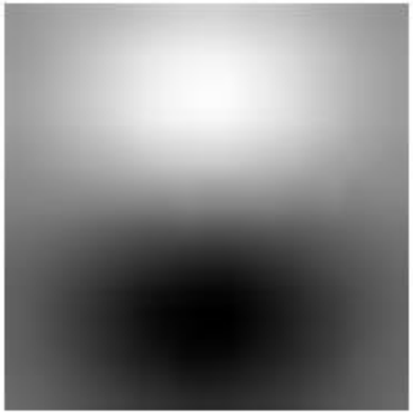


x-direction

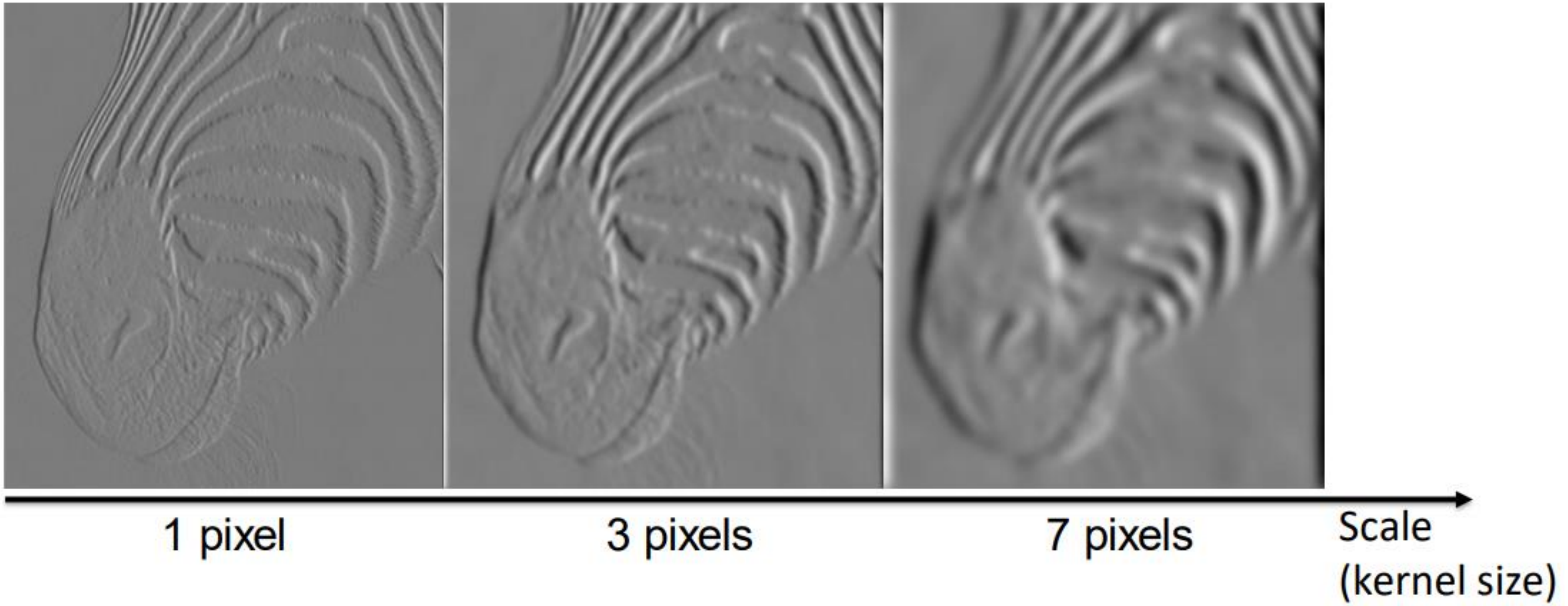


y-direction





Tradeoff between smoothing and localization



- Stronger smoothing removes noise, but blurs edges.
- Finds edges at different “scales”.

Laplace filter

Basically a second derivative filter.

- We can use finite differences to derive it, as with first derivative filter.

first-order finite difference

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h}$$

1D derivative filter

1	0	-1

second-order finite difference

$$f''(x) = \lim_{h \to 0} \frac{f(x + h) - 2f(x) + f(x - h)}{h^2}$$

Laplace filter

1	-2 ?	1

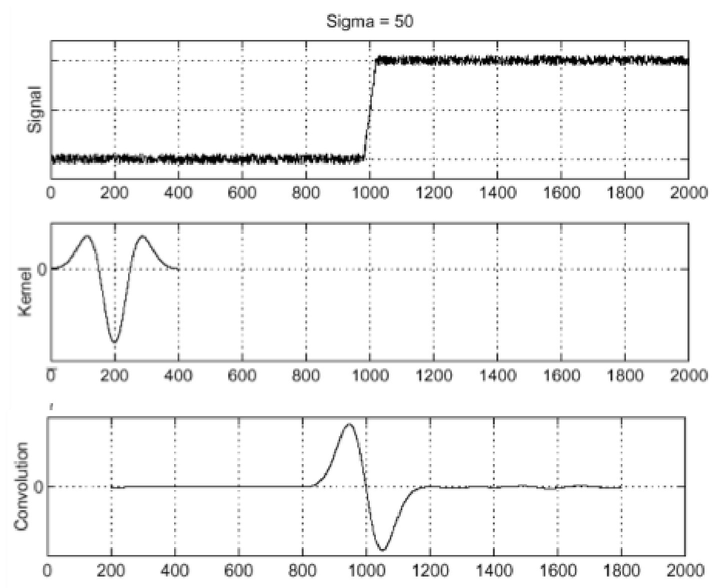
Laplacian of Gaussian (LoG) filter

- As with derivative, we can combine Laplace filtering with Gaussian filtering

input

Laplacian of Gaussian

output



“zero crossings” at edges

Laplace and LoG filtering examples


Laplacian of Gaussian filtering


Laplace filtering

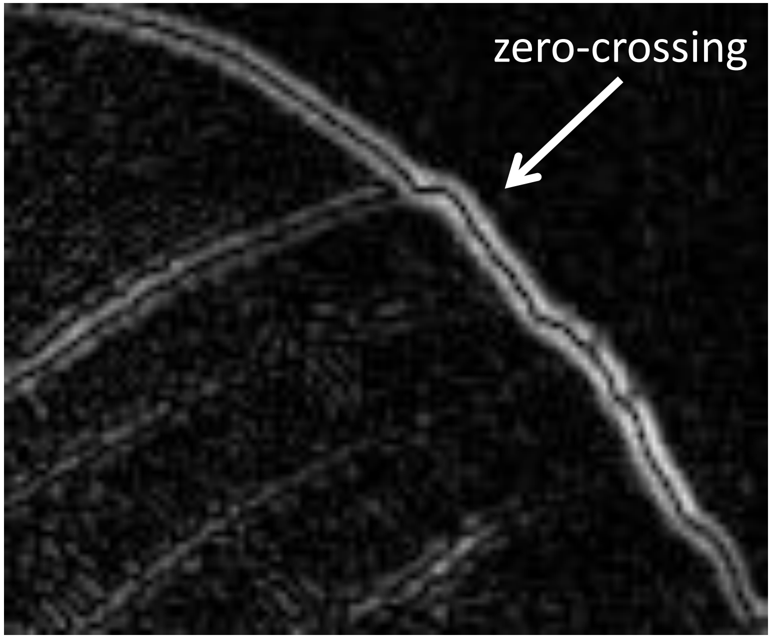
Laplacian of Gaussian vs Derivative of Gaussian

Laplacian of Gaussian filtering

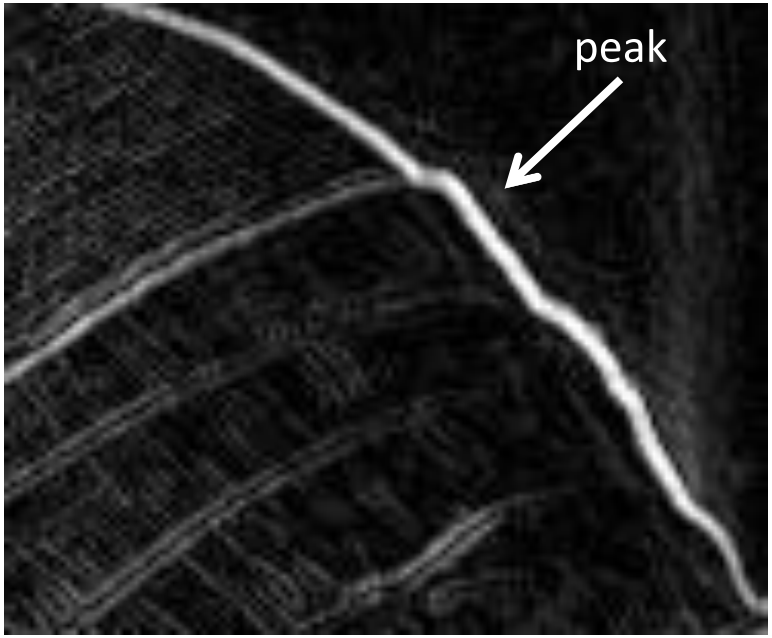
Derivative of Gaussian filtering

Laplacian of Gaussian vs Derivative of Gaussian

- Zero crossings are more accurate at localizing edges (but not very convenient)

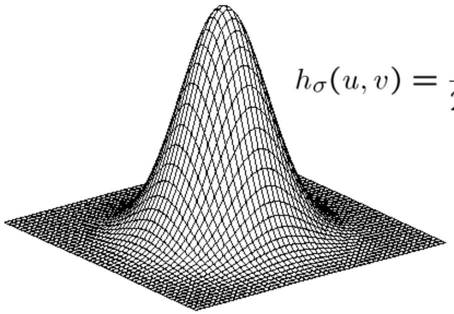


Laplacian of Gaussian filtering



Derivative of Gaussian filtering

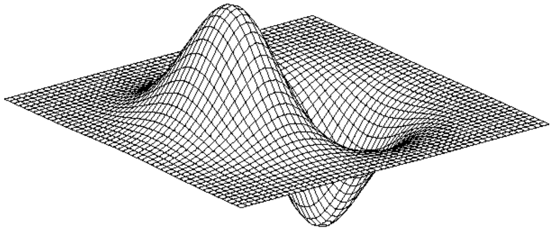
2D Gaussian filters



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

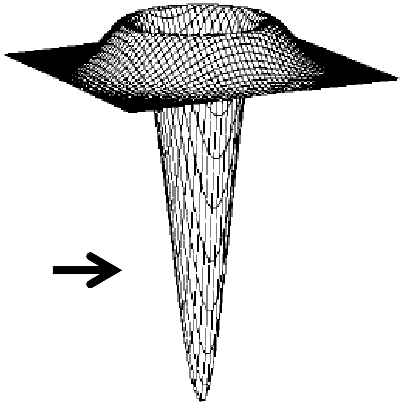
Gaussian

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



Derivative of Gaussian

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

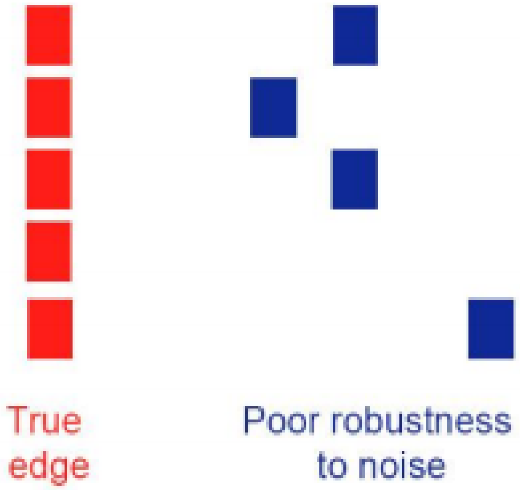


Laplacian of Gaussian

Designing an edge detector

- Criteria for an “optimal” edge detector:

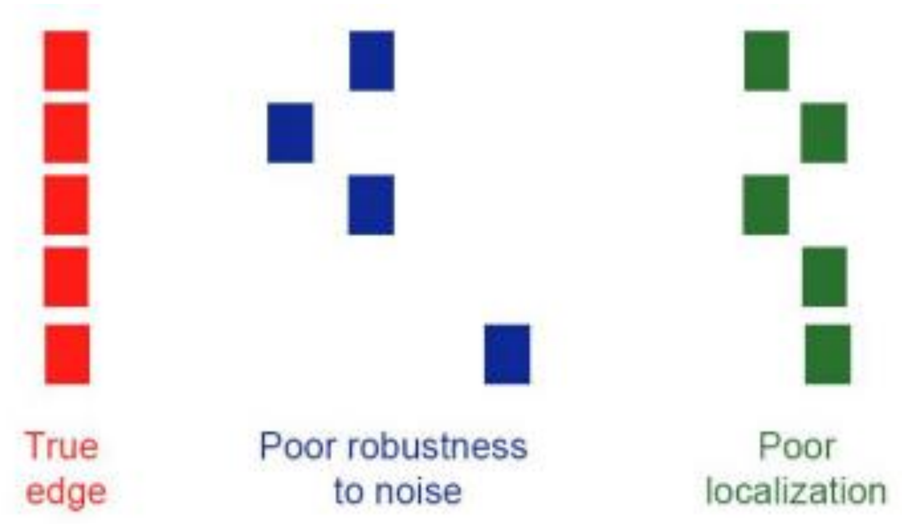
  (1) Good detection: the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)



Designing an edge detector

- Criteria for an “optimal” edge detector:

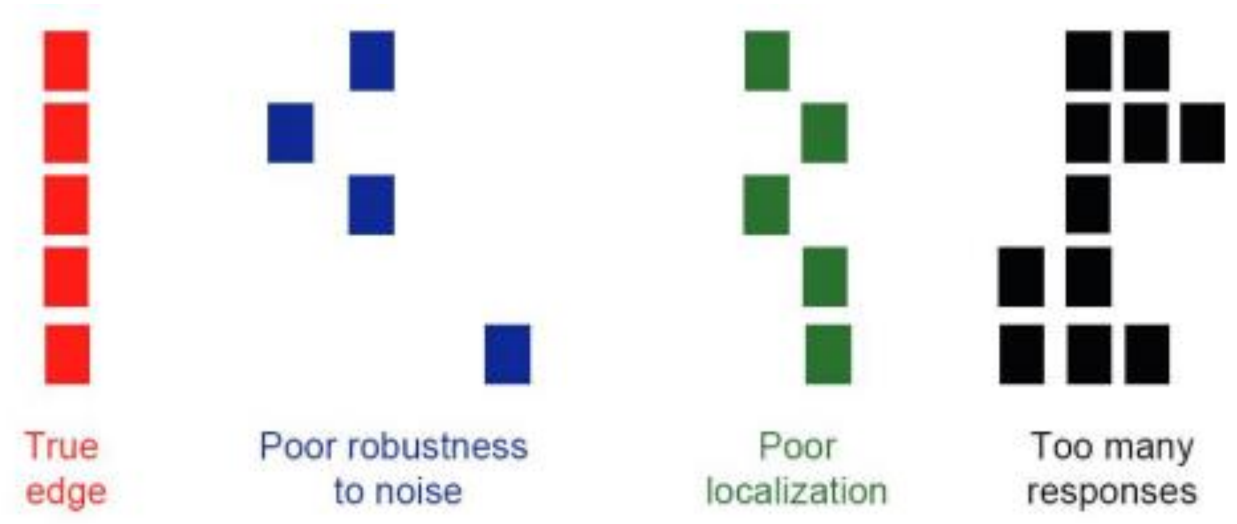
  (2) Good localization: the edges detected must be as close as possible to the true edges



Designing an edge detector

- Criteria for an “optimal” edge detector:

  (3) Single response: the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge



Outline

- Overview
- Image gradients
- A simple edge detector
- Sobel edge detector
- Canny edge detector
- Hough transform for line detection

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

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

Sobel Operation

- Smoothing + differentiation

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

Gaussiansmoothing differentiation

The Sobel filter

- Horizontal Sober filter:

$$\begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline \end{array}$$

- Vertical Sobel filter:

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array} = \begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline -1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array}$$

Sobel Operation

- Magnitude:

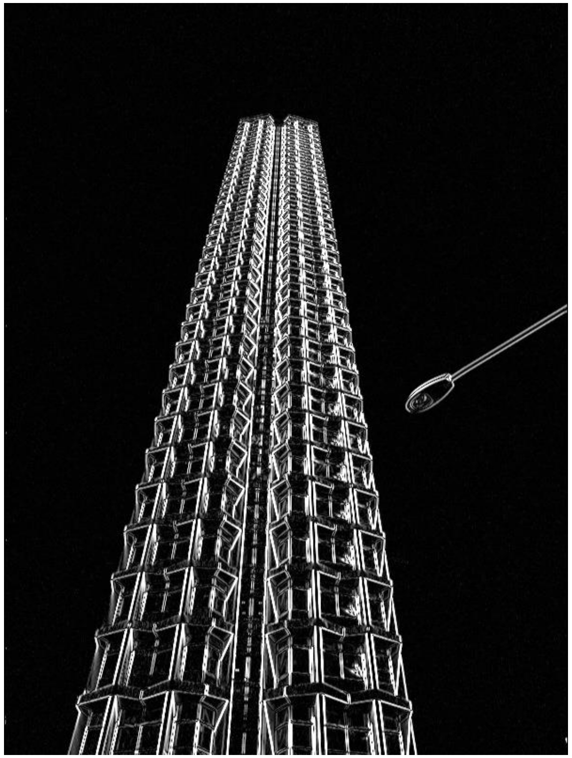
$$\mathbf{G} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2}$$

- Angle or direction of the gradient:

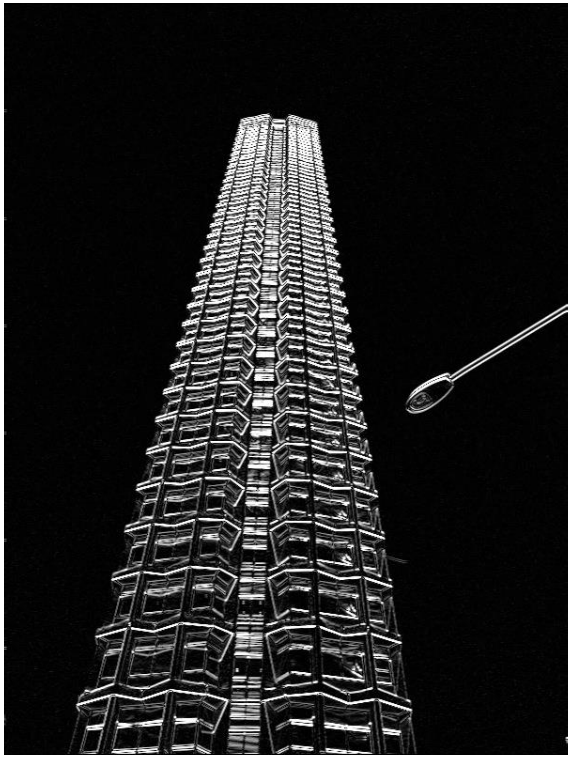
$$\mathbf{\Theta} = \operatorname{atan}\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

Sobel filter example

original



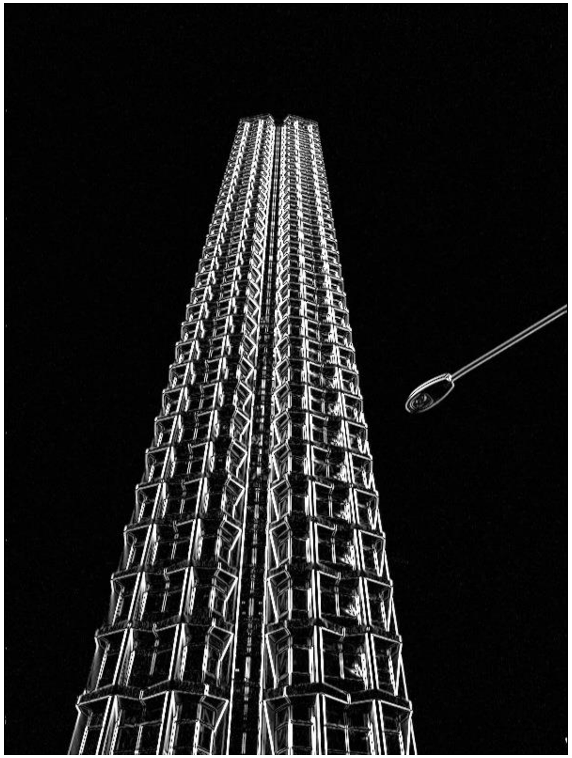
which Sobel filter?



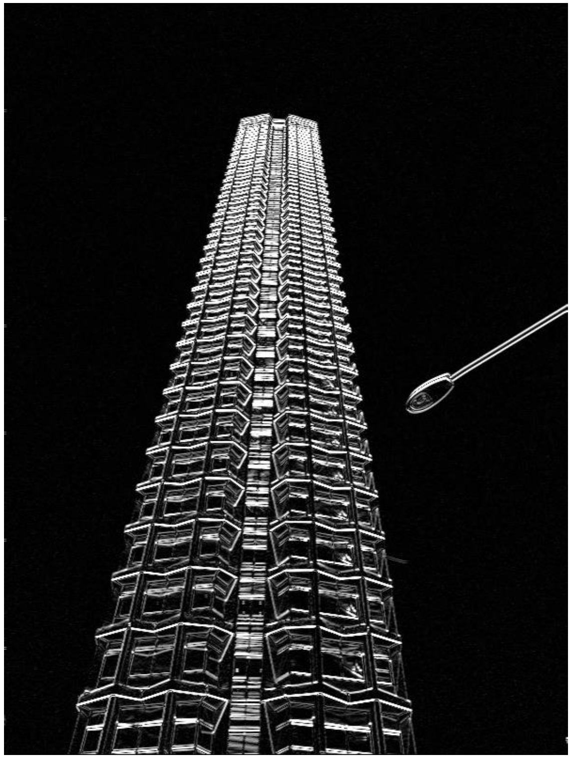
which Sobel filter?

Sobel filter example

original

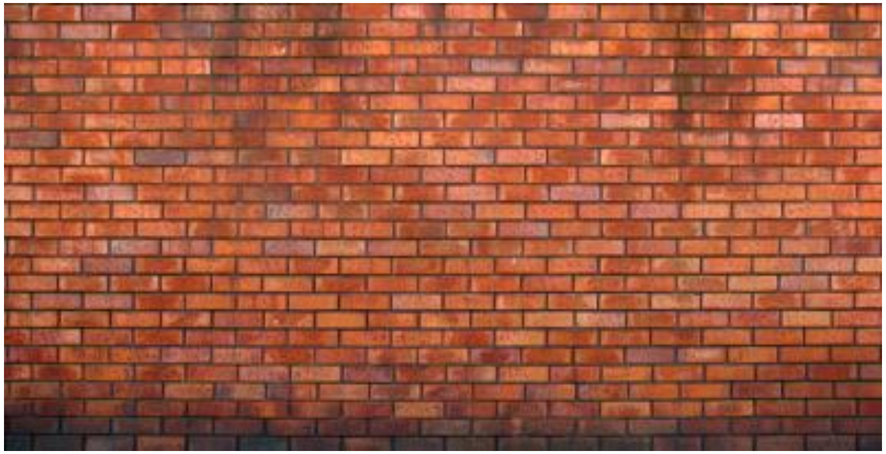


horizontal Sobel filter



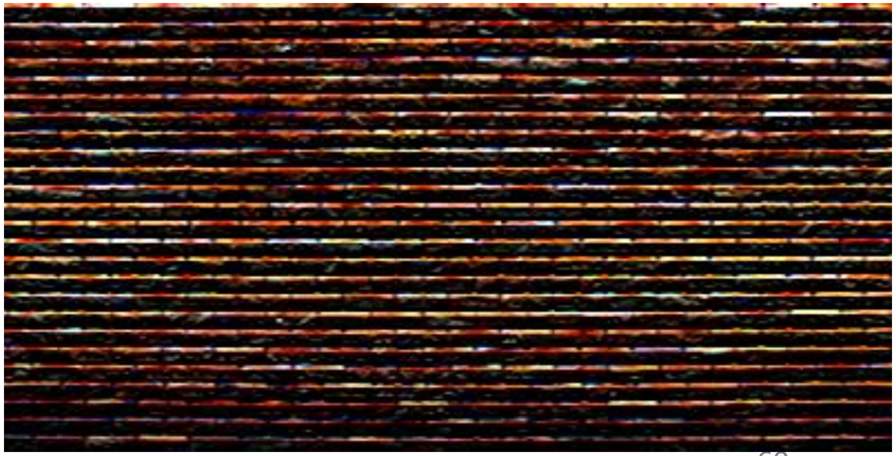
vertical Sobel filter

Sobel filter example



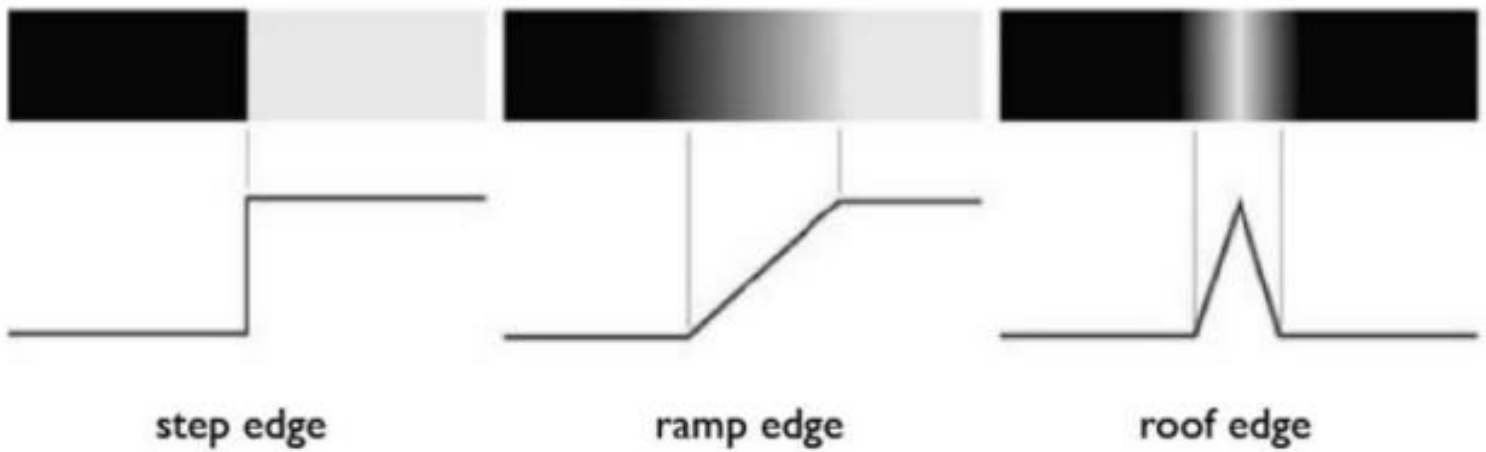
original

horizontal Sobel filter



vertical Sobel filter

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 (missing real edges)

Several derivative filters

Sobel

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

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

Scharr

3	0	-3
10	0	-10
3	0	-3

3	10	3
0	0	0
-3	-10	-3

Prewitt

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

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

Roberts

0	1
-1	0

1	0
0	-1

- How are the other filters derived and how do they relate to the Sobel filter?
- How would you derive a derivative filter that is larger than 3x3?

★ https://blog.csdn.net/m0_38007695/article/details/112796821

Outline

- Overview
- Image gradients
- A simple edge detector
- Sobel edge detector
- Canny edge detector

Canny edge detector

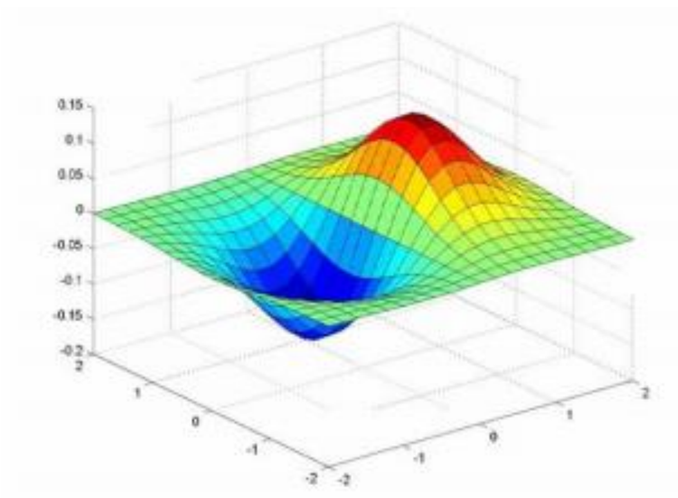
- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of signal-to-noise ratio and localization

J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

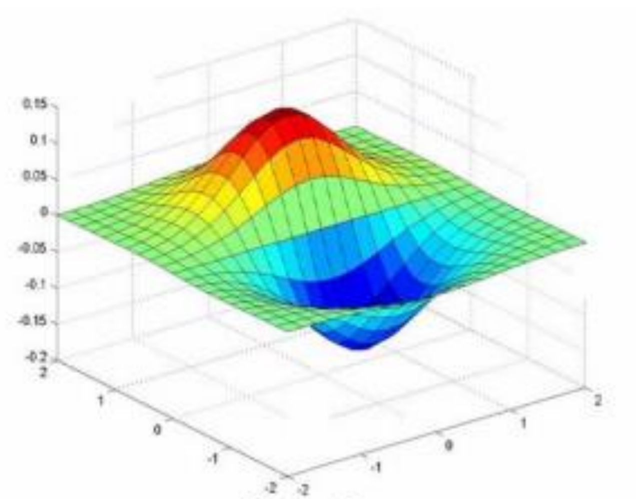
Canny edge detector

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

Derivative of Gaussian (DOG) filter

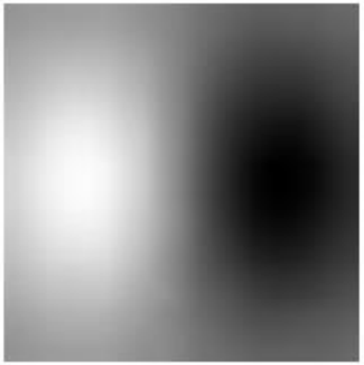


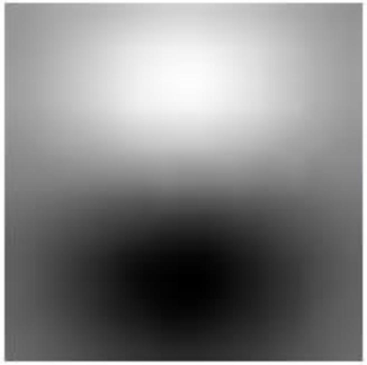
x-direction



y-direction

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



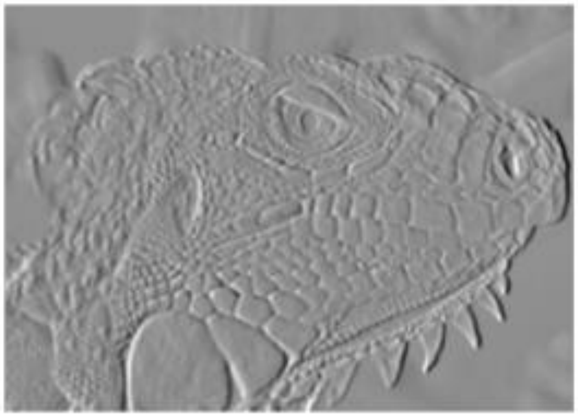


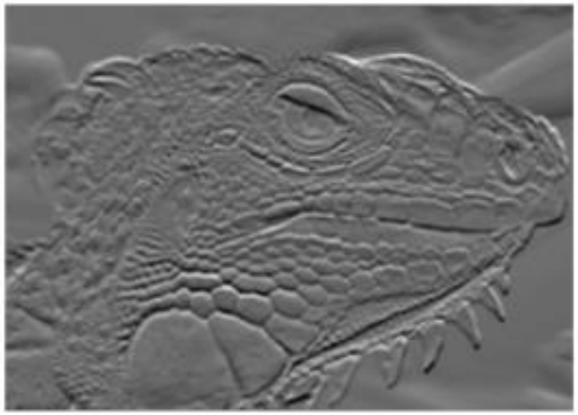
$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Example

Original image

Computing gradients



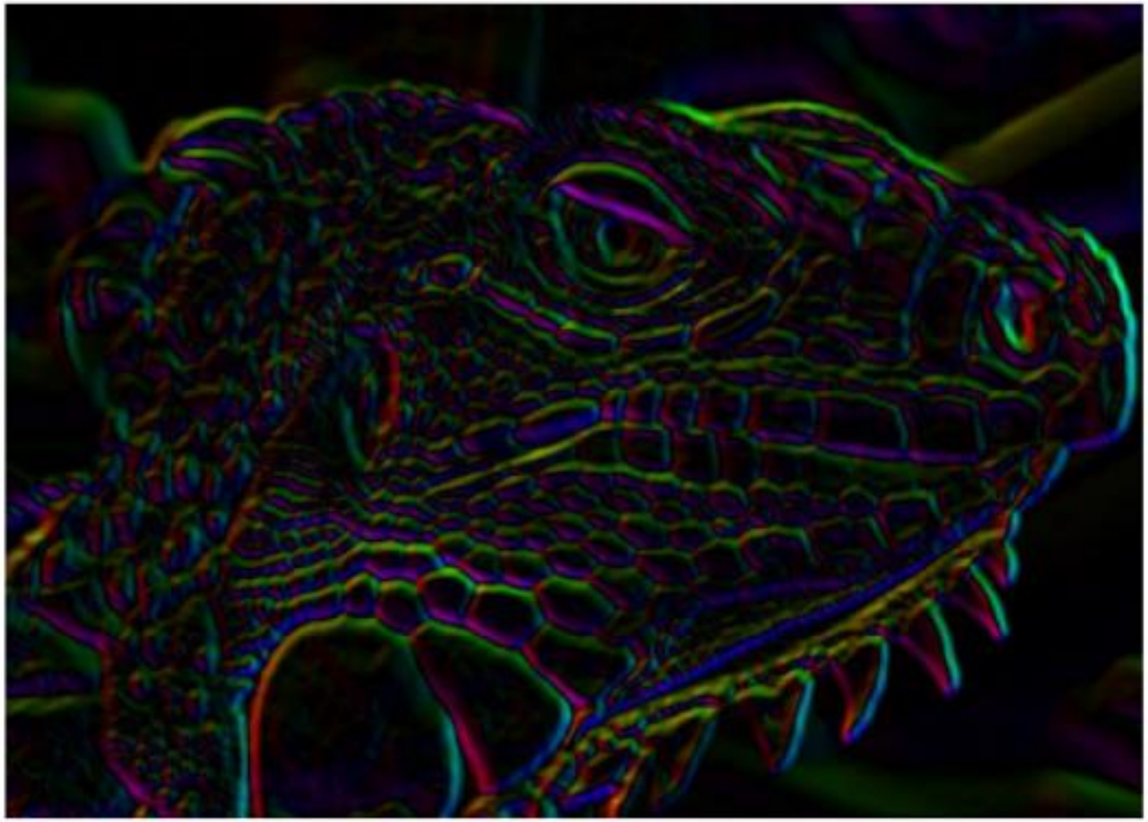


X-Derivative of Gaussian

Y-Derivative of Gaussian

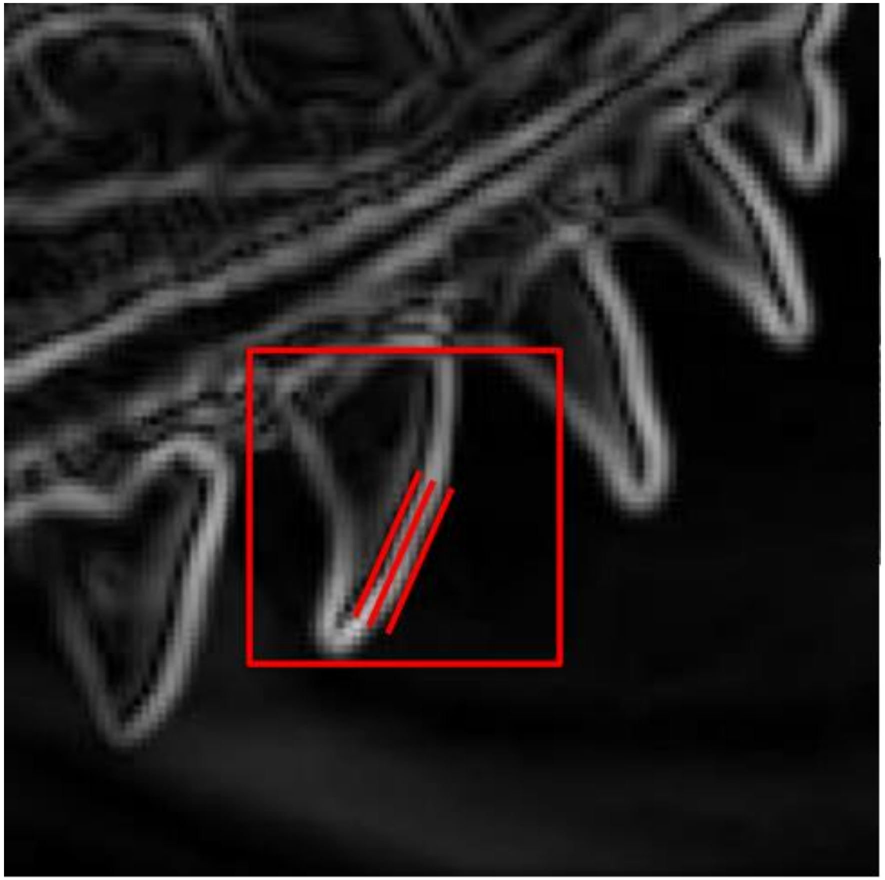
Gradient Magnitude

Get orientation at each pixel



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

Compute gradients



Gradient
Magnitude

Canny edge detector

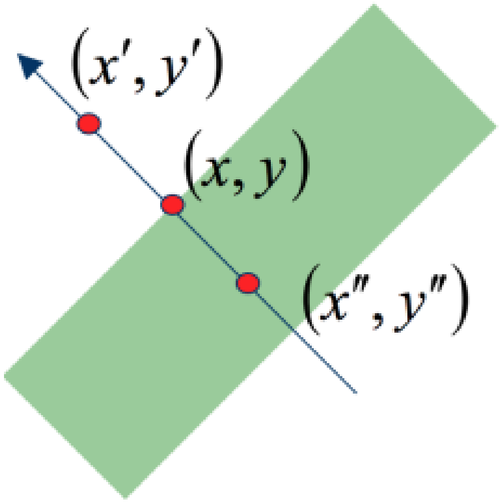
- Suppress Noise
- Compute gradient magnitude and direction
- Apply Non-Maximum Suppression

Non-maximum suppression

- Edge occurs where gradient reaches a maxima
- Suppress non-maxima gradient even if it passes threshold
- Compare current pixel *vs* neighbors along direction of gradient
  - Remove if not maximum

Remove spurious gradients

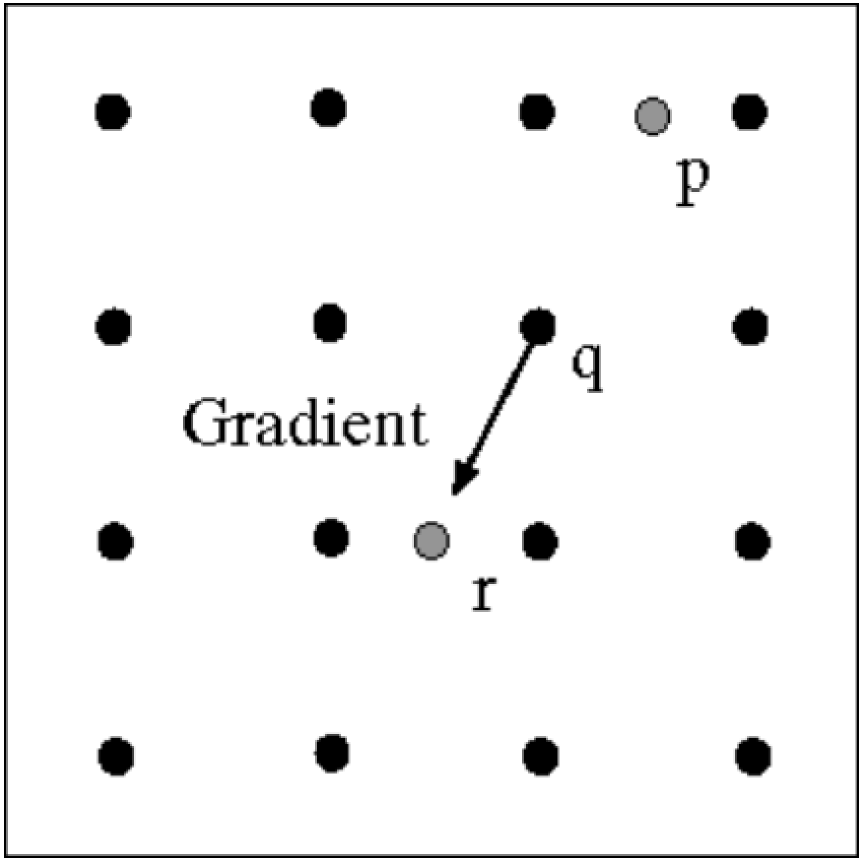
$|\nabla G|(x, y)$ is the gradient at pixel (x, y)



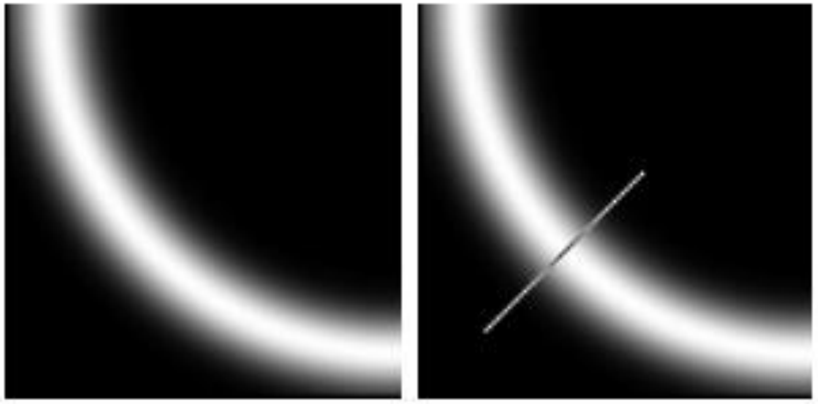
$$M(x, y) = \begin{cases} |\nabla G|(x, y) & \text{if } |\nabla G|(x, y) > |\nabla G|(x', y') \\ & \& |\nabla G|(x, y) > |\nabla G|(x'', y'') \\ 0 & \text{otherwise} \end{cases}$$

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

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.



Non-maximum suppression

Before After

Canny edge detector

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

Detecting edges with a single threshold

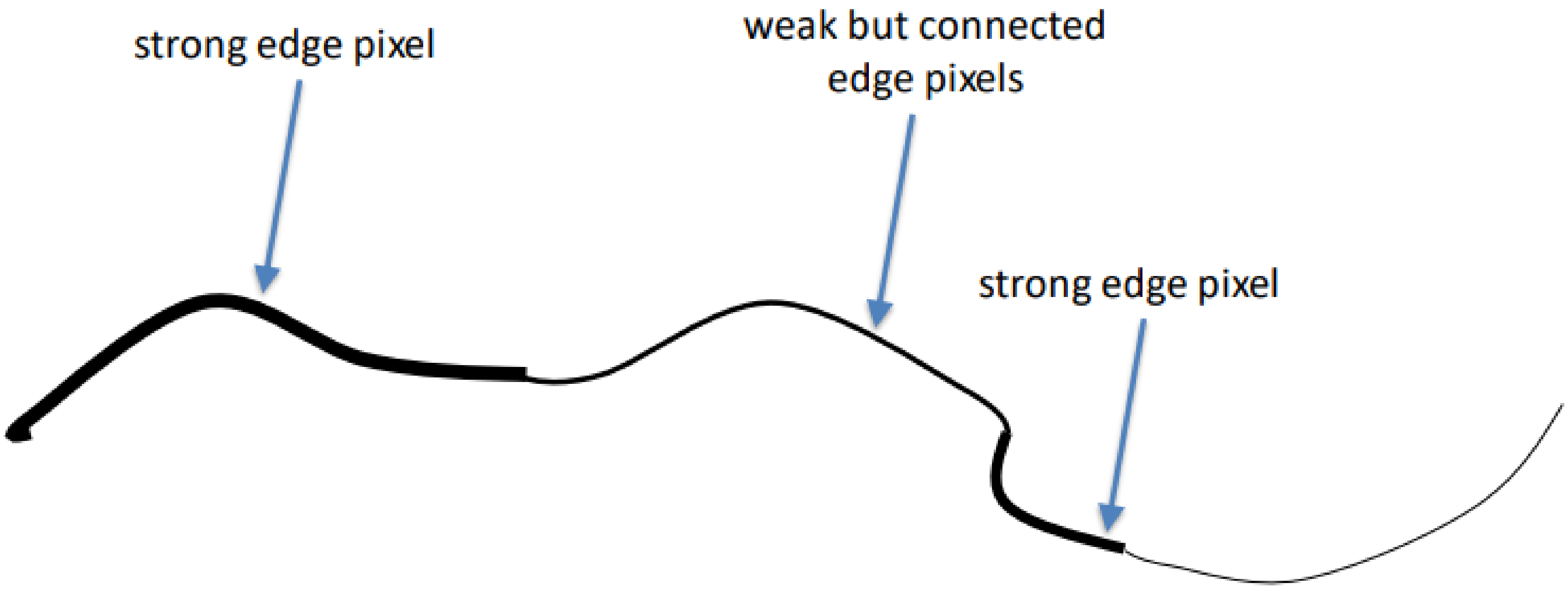
Threshold too high

Threshold too low

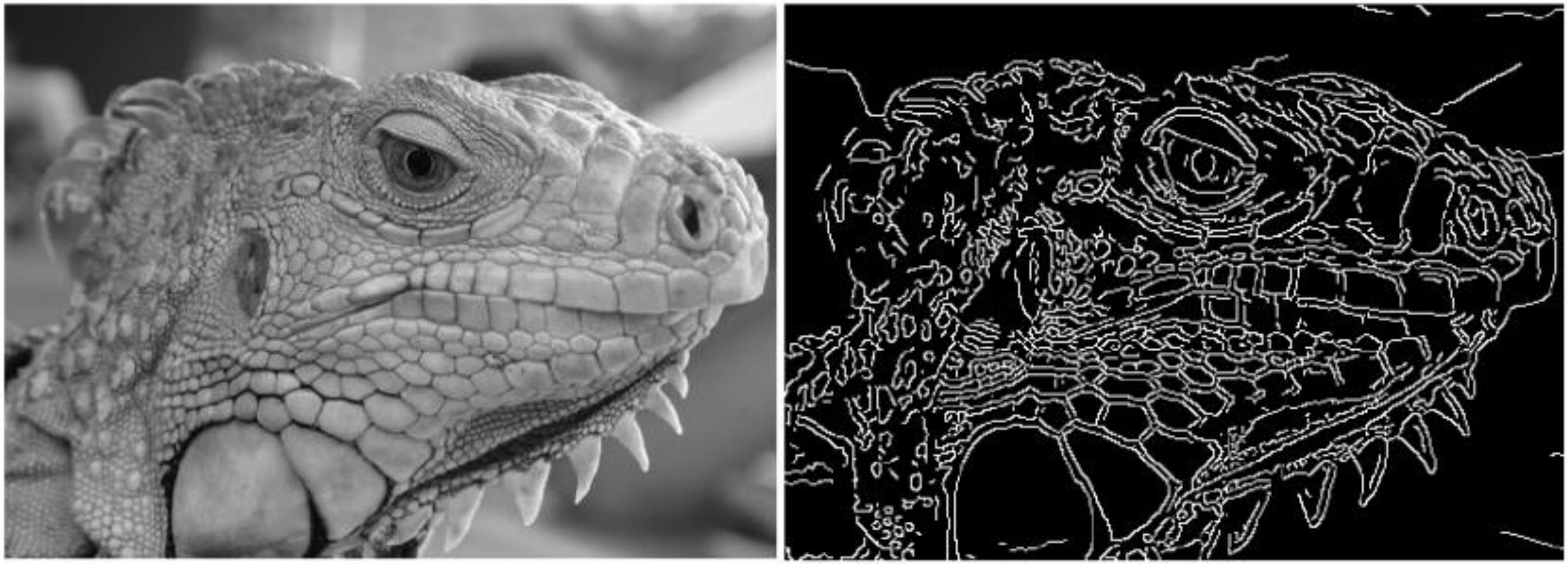
Hysteresis thresholding

- 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
  - Consider its neighbors iteratively then declare it an "edge pixel" if it is connected to an 'strong edge pixel' directly and via pixels between Low and High

Hysteresis thresholding



Final Canny Edges



Canny edge detector

1. Filter image with a filter
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

Effect of σ(Gaussian kernel spread/size)

- The choice of σ depends on desired behavior
  - Large σ detects large scale edges
  - Small σ detects fine features

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

References

- Basic reading:
  - Szeliski textbook, Chapter 3.2, 4,1