# CSCI 4830 / 5722
# Computer Vision

University of Colorado **Boulder**

# CSCI 4830 / 5722
# Computer Vision

Dr. Ioana Fleming
Spring 2019
Lecture 10

University of Colorado **Boulder**

# Reminders

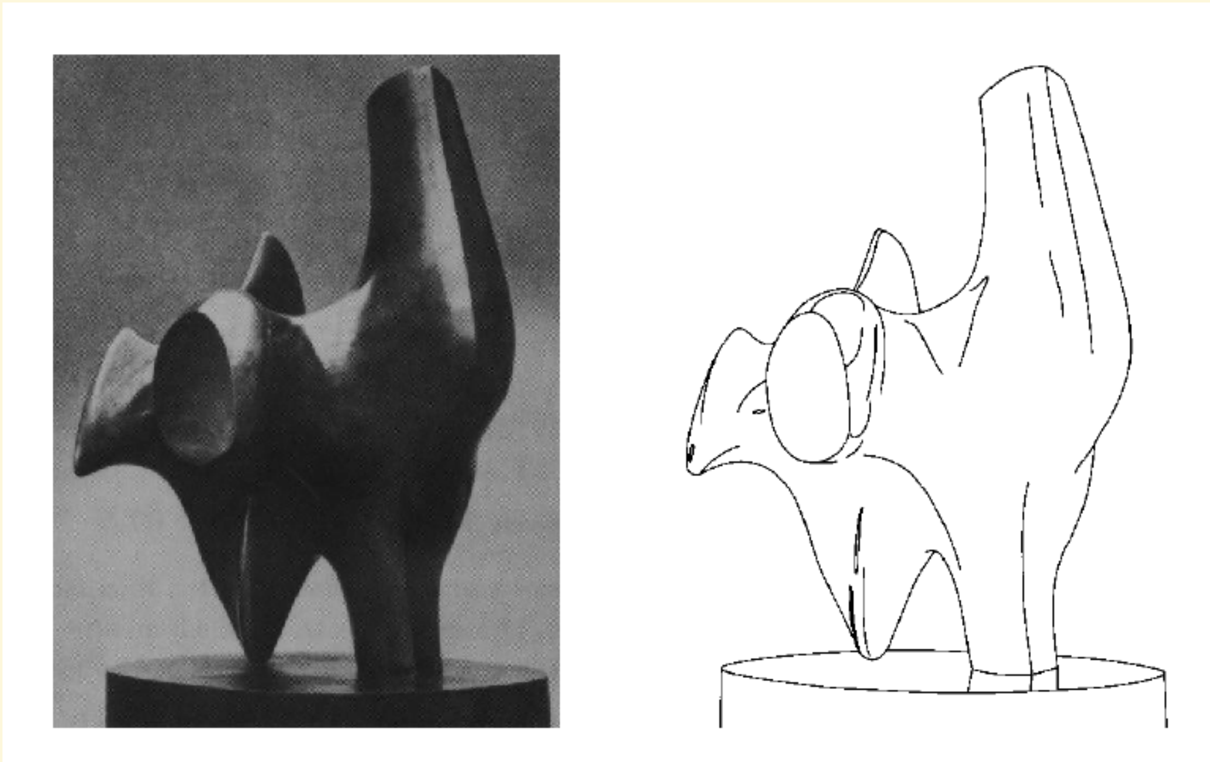

Submissions:
- Homework 2: due Wed 2/13 at 11 pm
- Homework 3: later this week

Readings:
- Szeliski:
  - chapter 3 (filters, changing resolution, Laplacian pyramids, warping)
  - chapter 4.1 (points) and 4.2 (edge detection)
- P&F Ch. 4,5

# Today

- Image filters as linear operators
- Edge detection

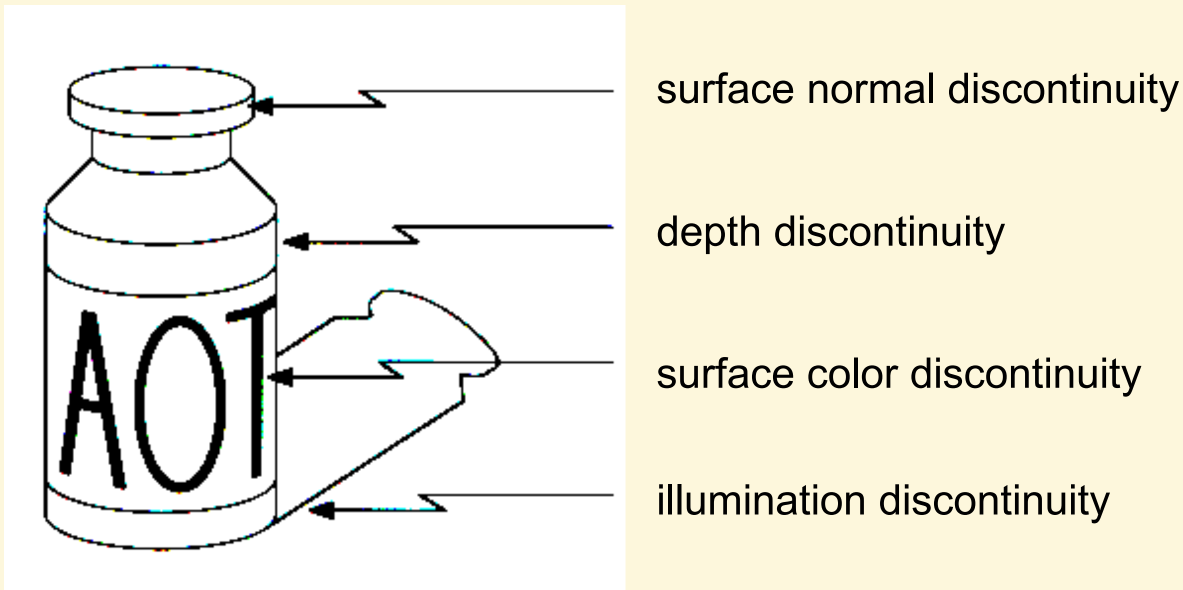
# Edge detection



- Convert a 2D image into a set of curves
  - Extracts salient features of the scene

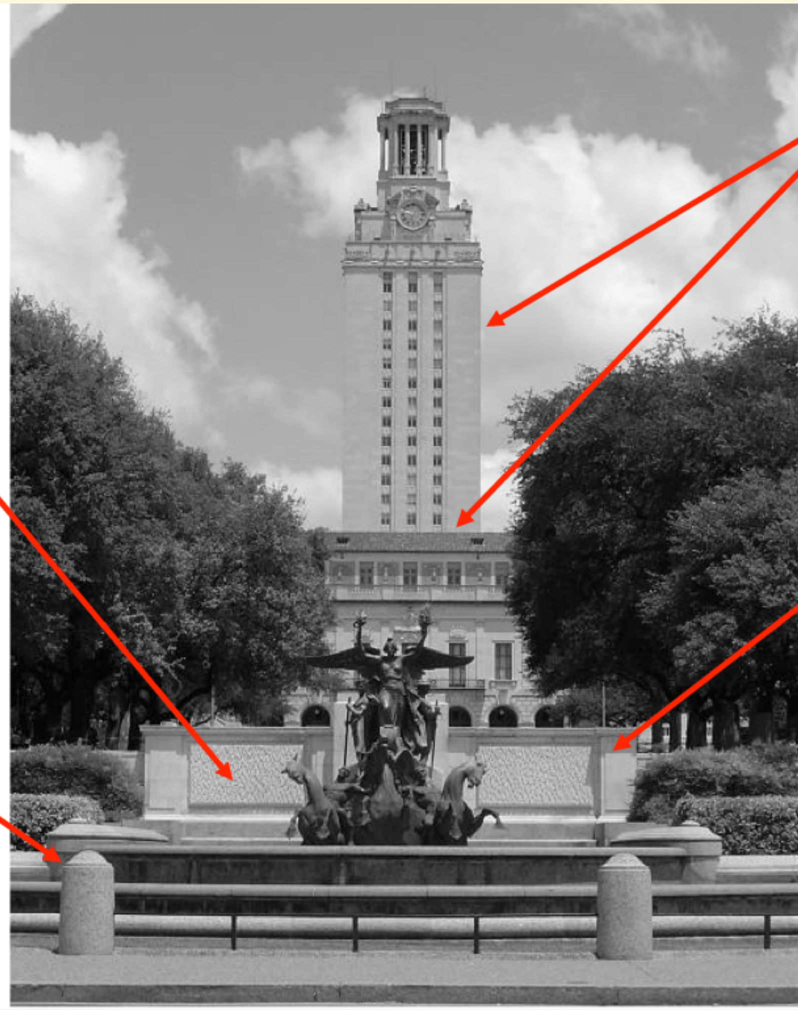
# Origin of Edges

- What can cause an edge?



# Why is edge detection useful?

Reflectance change: appearance information, texture

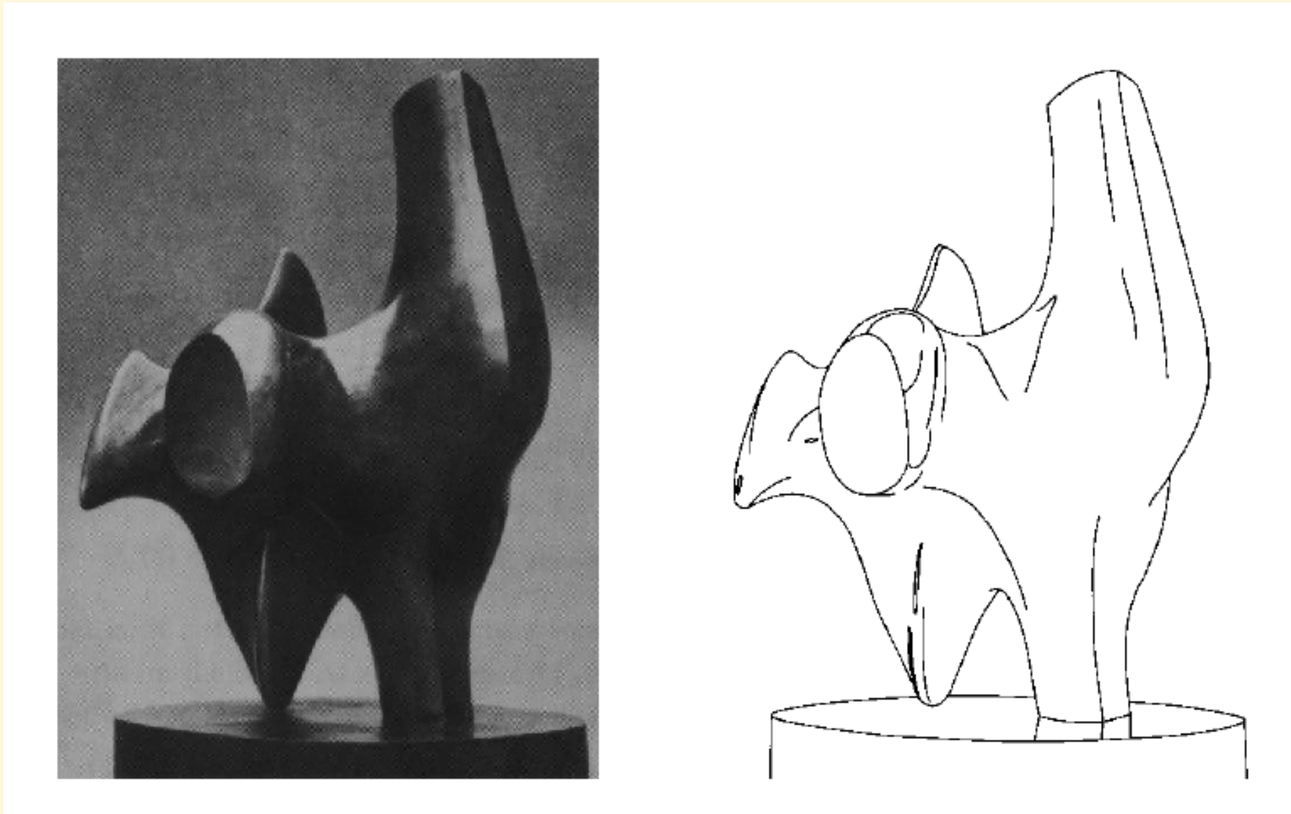


Depth discontinuity: object boundary

Cast shadows

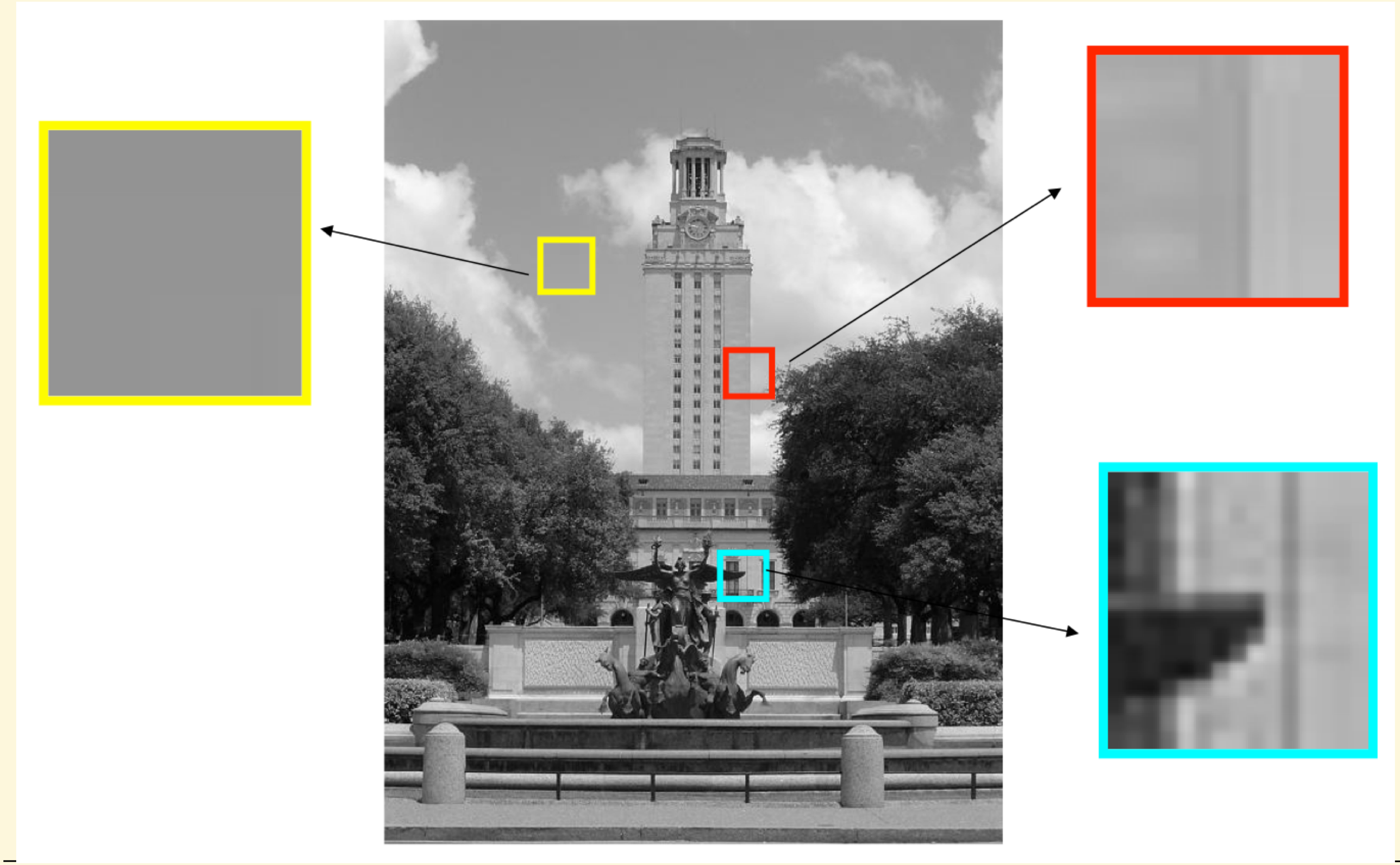
Change in surface orientation: shape

# Edge detection

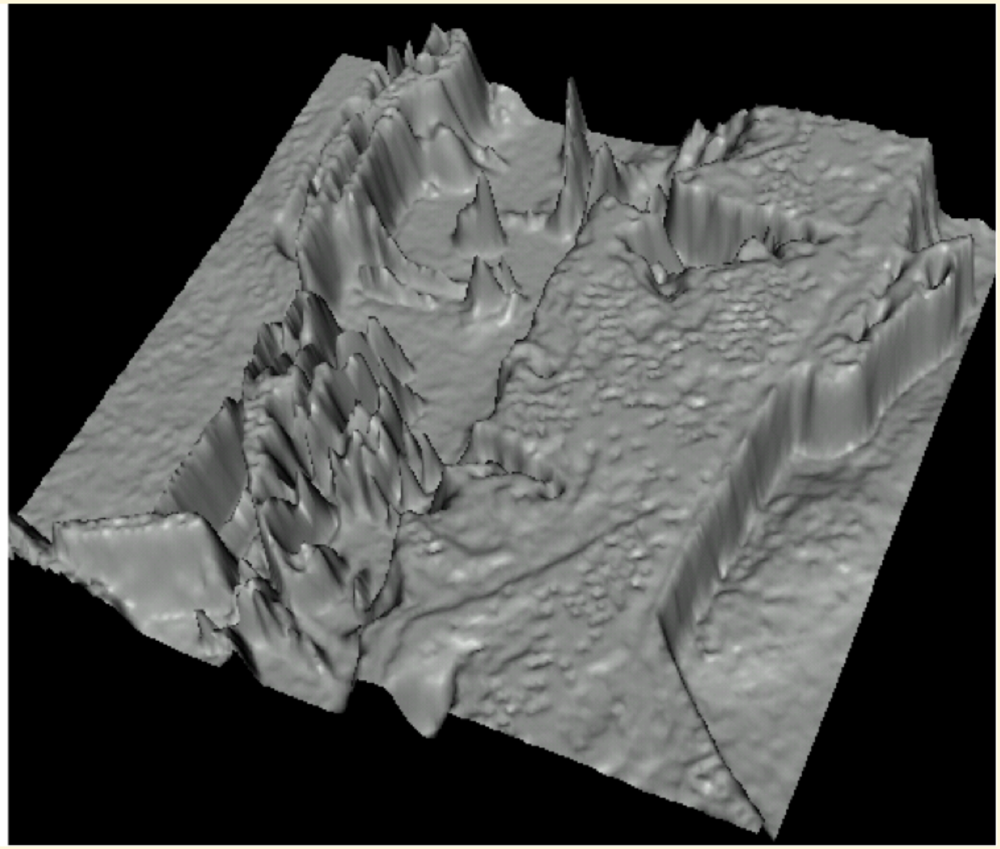


- How can you tell that a pixel is on an edge?

# Contrast and Invariance

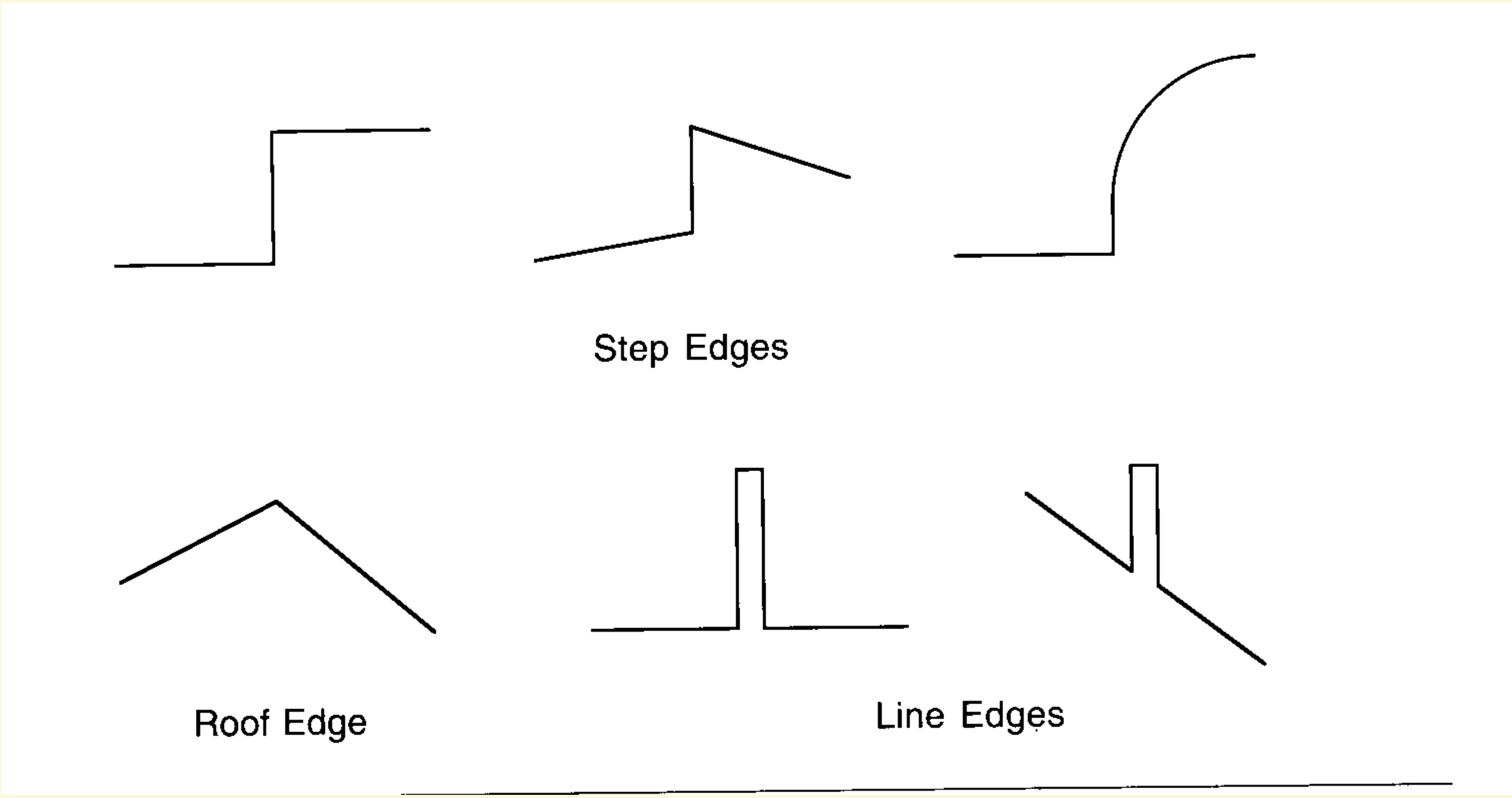


# Images as functions



Edges look like steep cliffs

# Profiles of image intensity edges

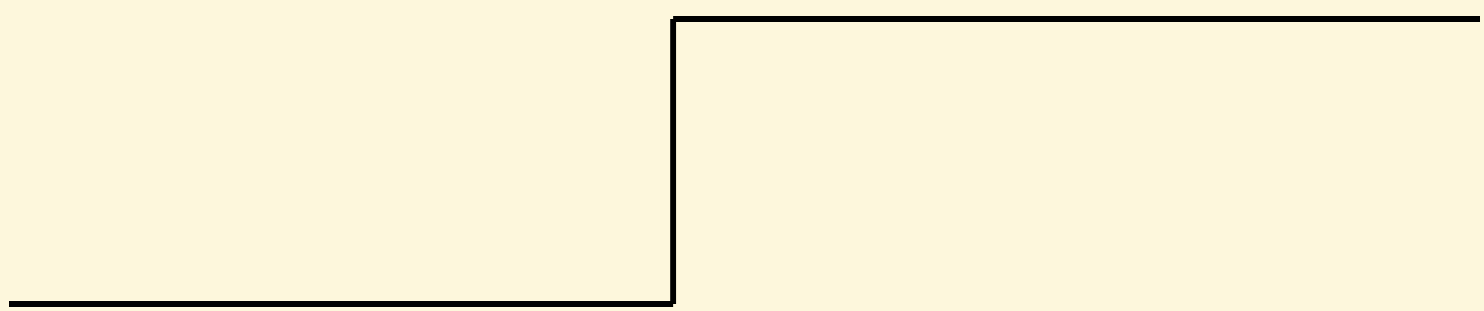


# Profiles of image intensity edges

**Step edge:** the image intensity abruptly changes from one value on one side of the discontinuity to a different value on the opposite side.
**Ramp edge:** a step edge where the intensity change is not instantaneous but occur over a finite distance.
**Ridge edge:** the image intensity abruptly changes value but then returns to the starting value within some short distance (i.e., usually generated by lines)
**Roof edge:** a ridge edge where the intensity change is not instantaneous but occurs over a finite distance (i.e., usually generated by the intersection of two surfaces)

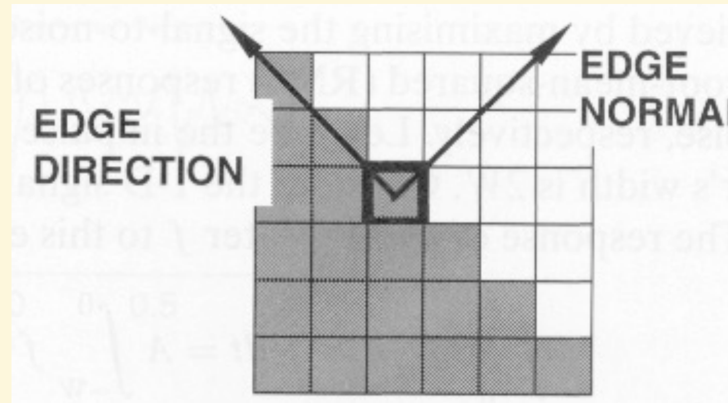
# Edge is Where Change Occurs

- Change is measured by derivative in 1D
- Biggest change is found where the derivative has maximum magnitude
- Or 2nd derivative is zero.



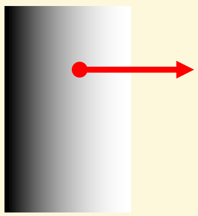
# Edge Descriptors

- **Edge direction:** perpendicular to the direction of maximum intensity change (i.e., edge normal)
- **Edge strength:** related to the local image contrast along the normal.
- **Edge position:** the image position at which the edge is located.

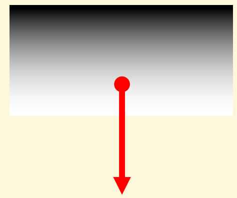


# Image gradient: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

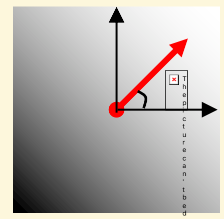
- The gradient points in the direction of most rapid change in intensity



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



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



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

The gradient direction is given by:

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

The *edge strength* is given by the gradient magnitude

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

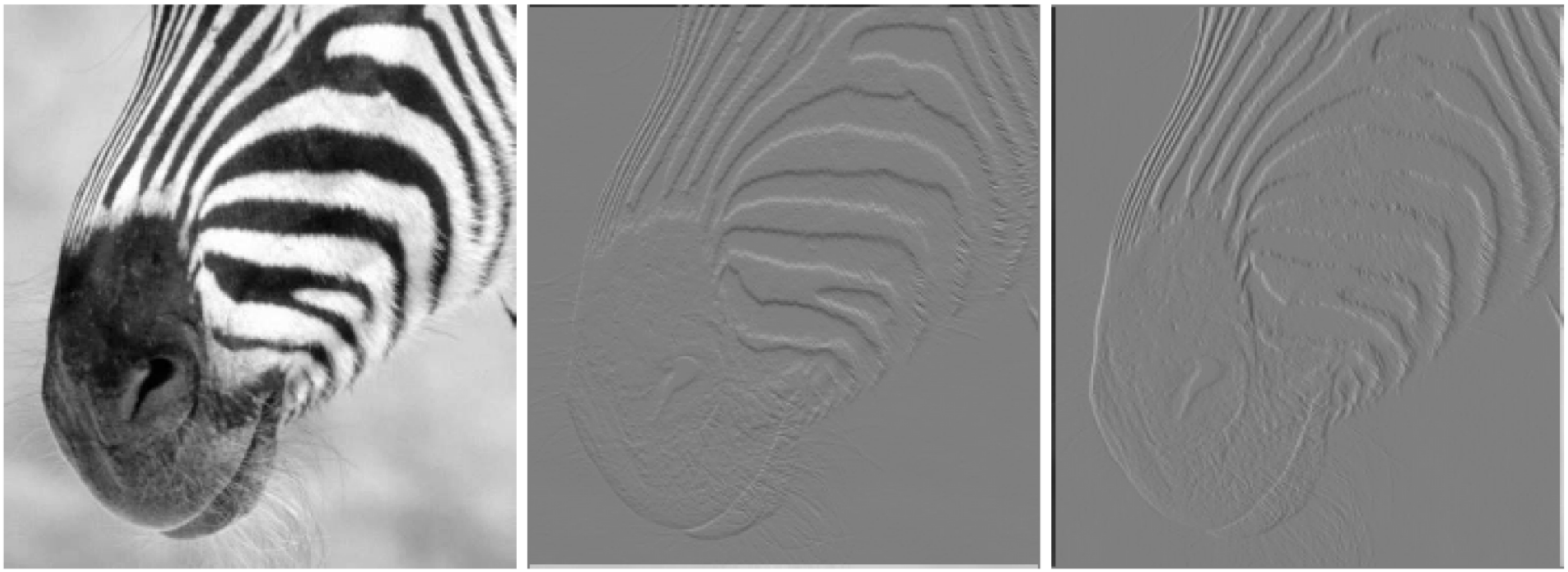
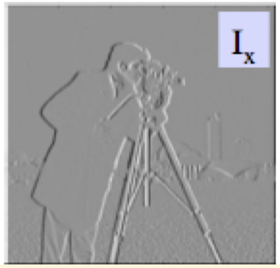
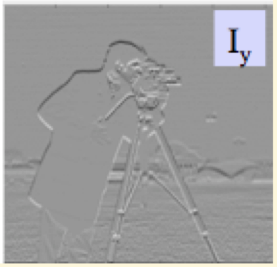


**Figure 8.1.** Finite differences are one way to obtain an estimate of a derivative. The image at the left shows a detail from a picture of a zebra. The center image shows the partial derivative in the $y$-direction — which responds strongly to horizontal stripes and weakly to vertical stripes — and the right image shows the partial derivative in the $x$-direction — which responds strongly to vertical stripes and weakly to horizontal stripes. In the derivative figures, a mid grey level is a zero value, a dark grey level is a negative value, and a light grey level is a positive value.

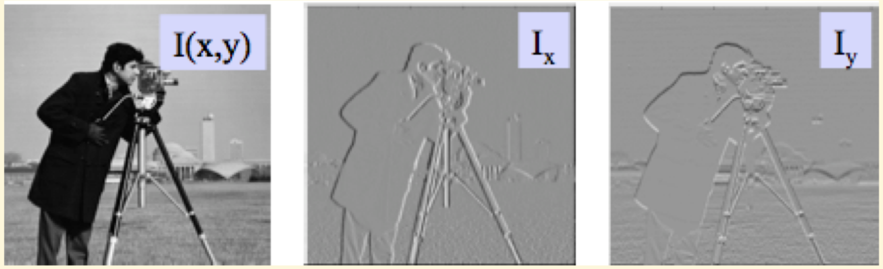
# Functions of Gradients

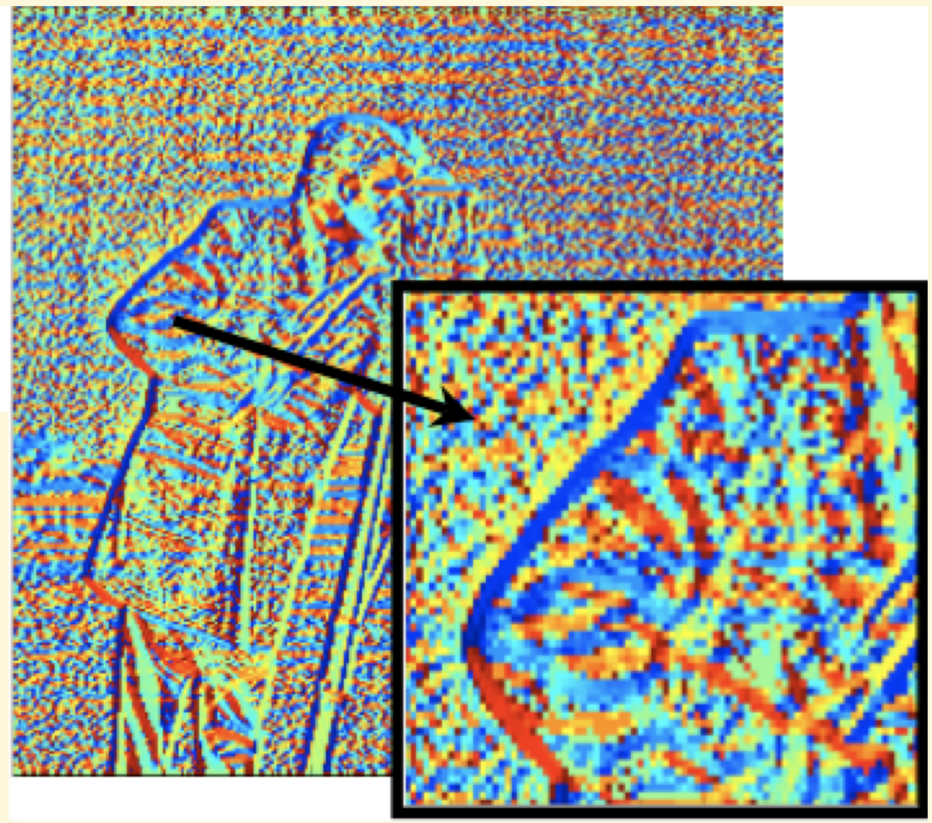




Magnitude of gradient
sqrt(Ix.^2 + Iy.^2)

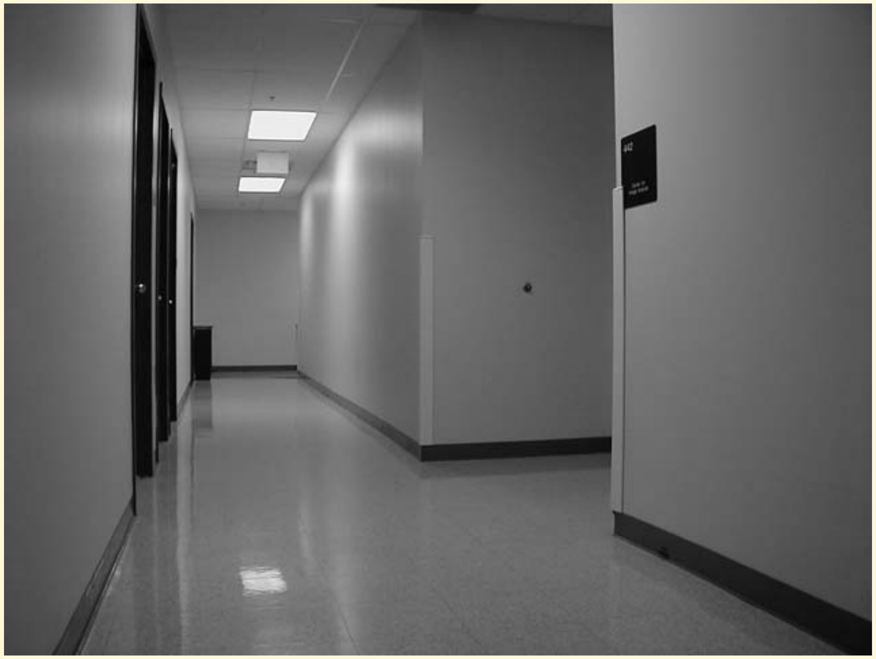
# Functions of Gradients

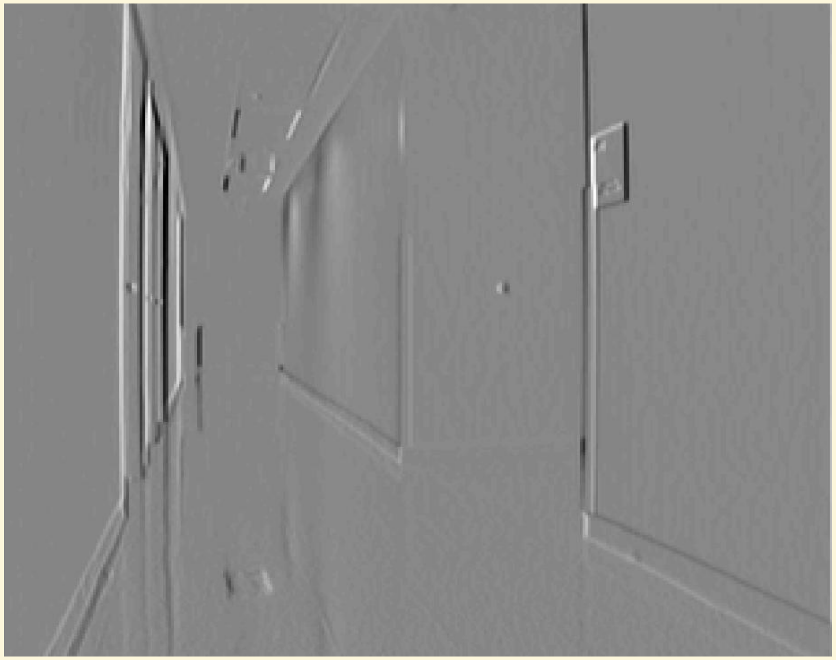


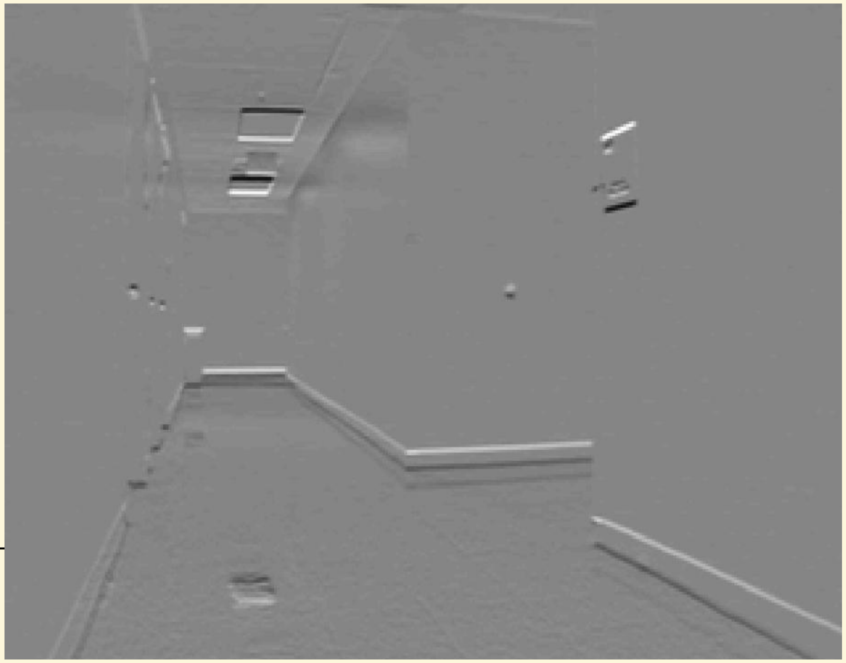

Angle of gradient
atan2(Iy, Ix)

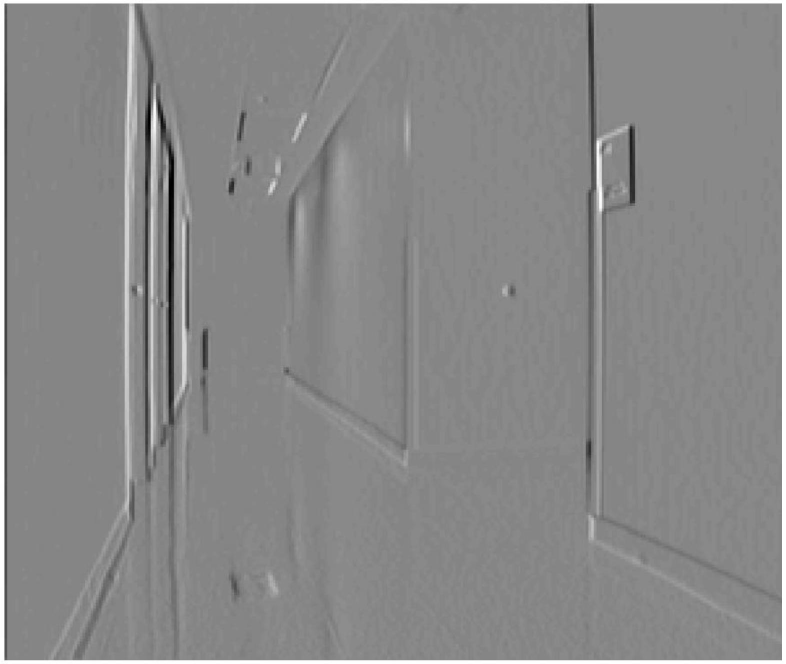


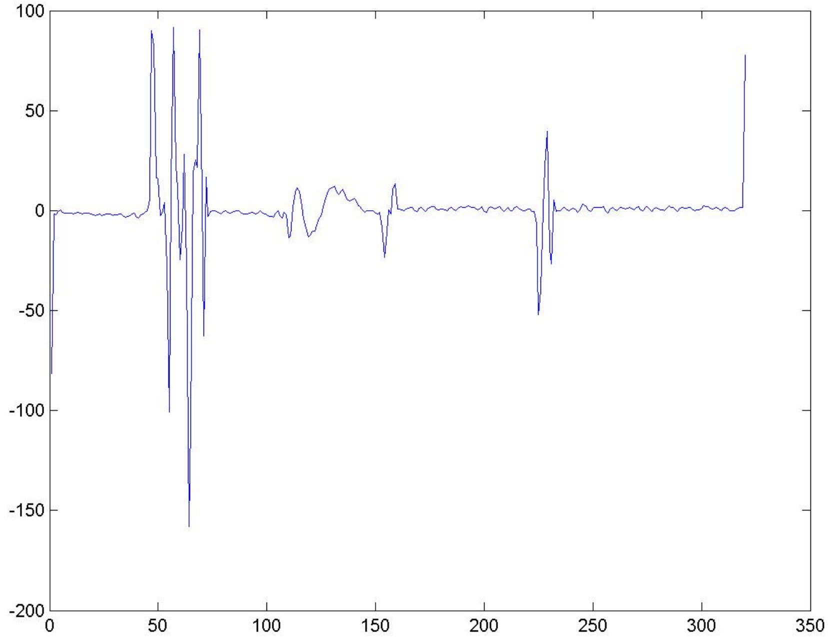
# Another example:







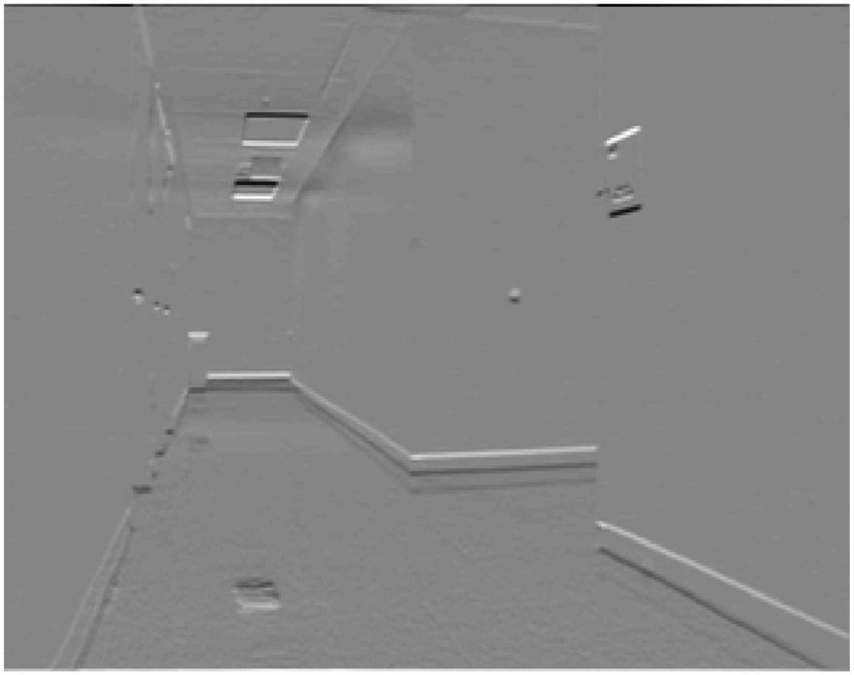





Vertical edges

First derivative

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

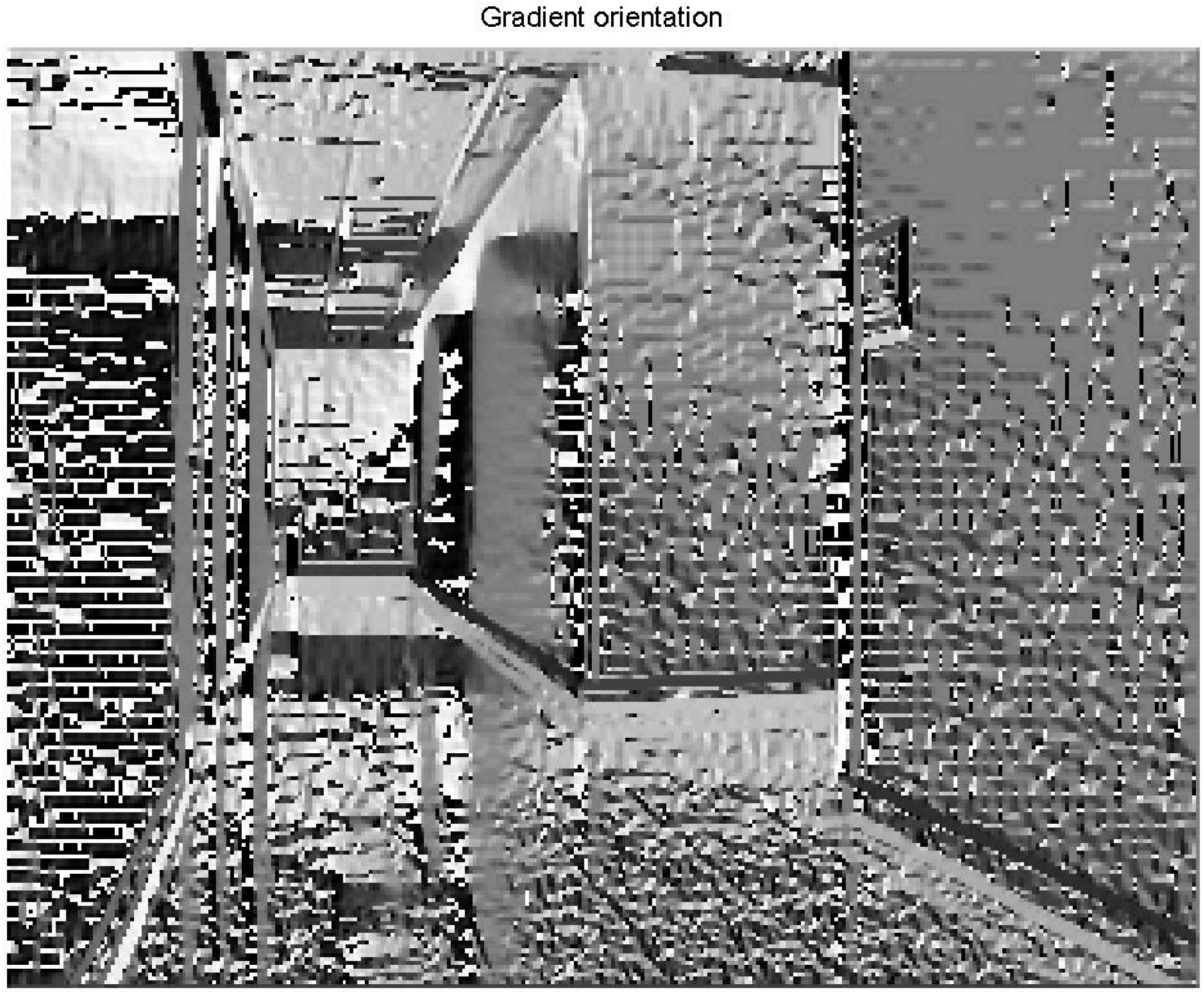
Horizontal edges

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

Gradient Magnitude

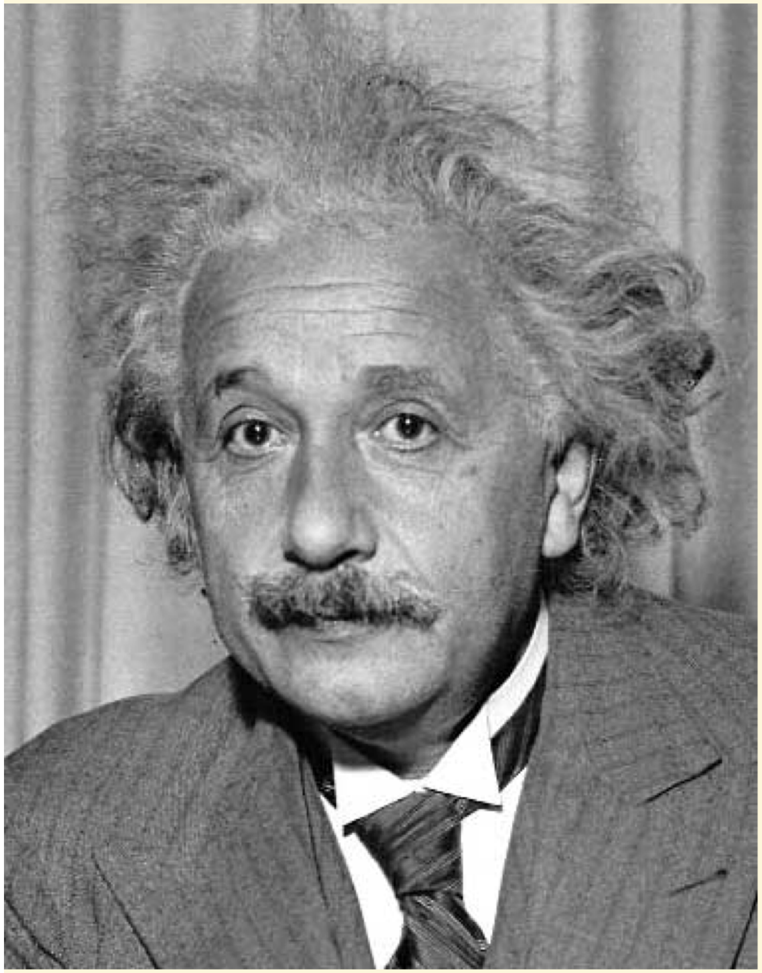
$$m = \sqrt{(\frac{\partial I}{\partial x})^2 + (\frac{\partial I}{\partial y})^2}$$

- Image Gradient $\nabla I = [\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}]$



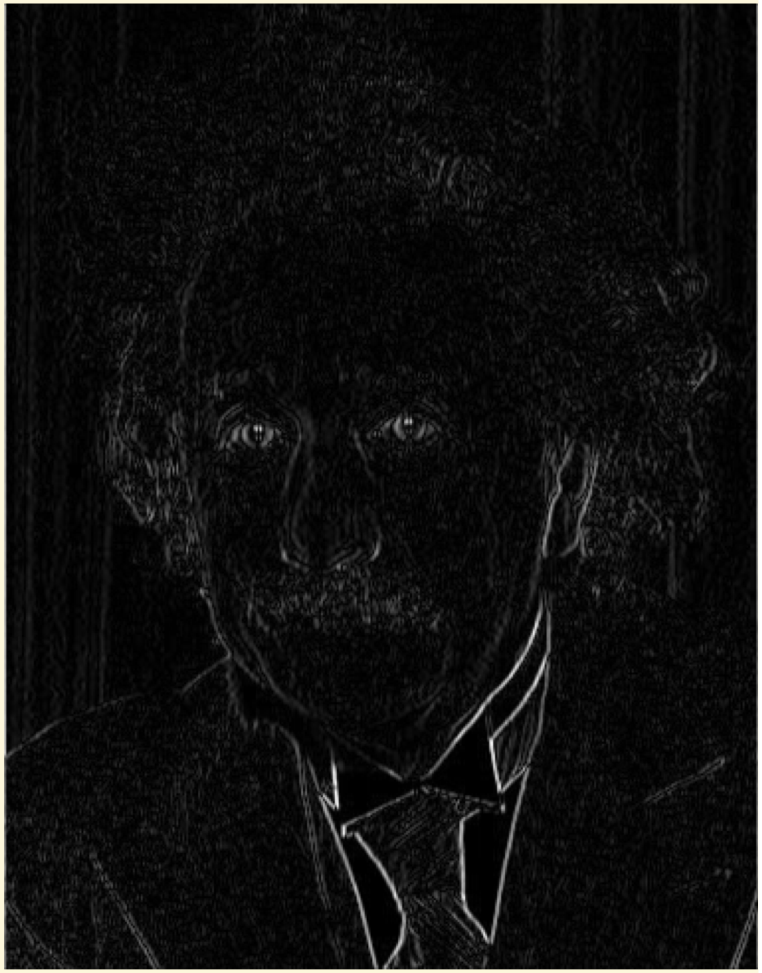

Gradient Orientation $\theta = \tan^{-1}(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$

# Other filters



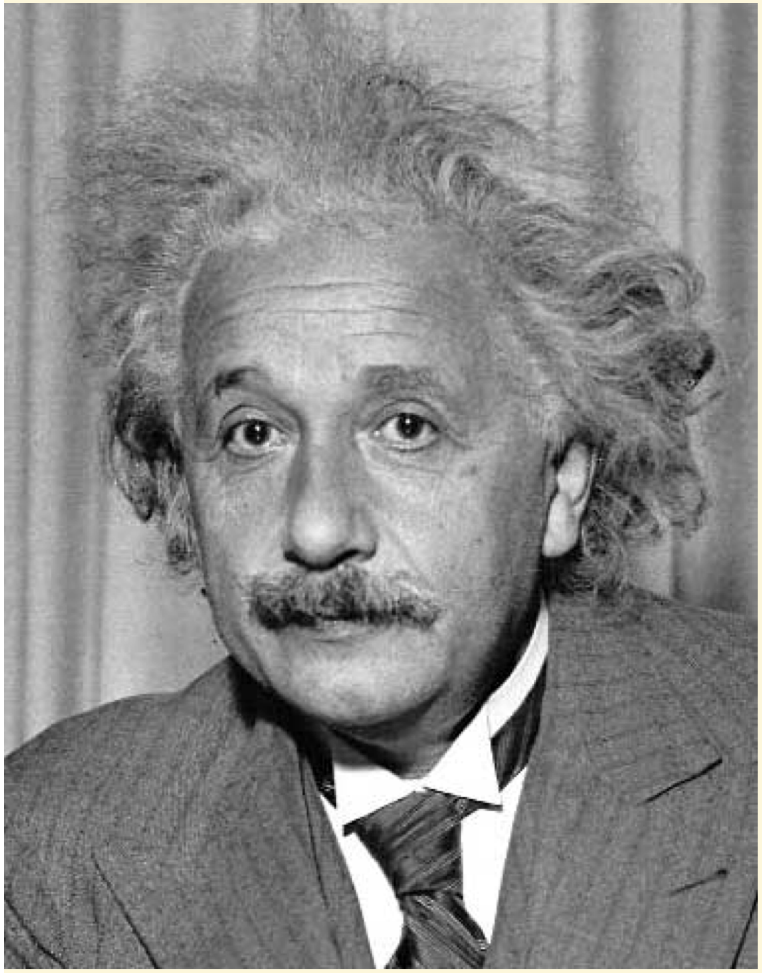
|  |  |  |
|---|---|---|
| 1 | 0 | -1 |
| 2 | 0 | -2 |
| 1 | 0 | -1 |

Sobel



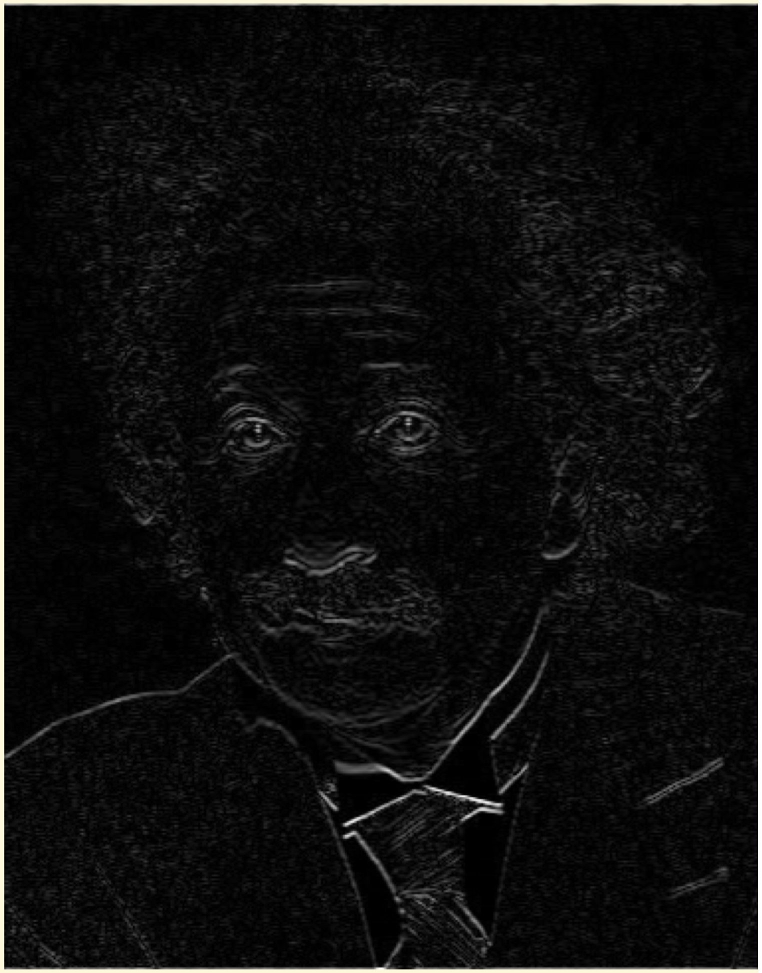
Vertical Edge
(absolute value)

# Other filters



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

Sobel



Horizontal Edge
(absolute value)

# Separable filters

A two-dimensional filter kernel is separable if it can be expressed as the outer product of two vectors.

Examples:

$$\frac{1}{3}\begin{bmatrix}1\\1\\1\end{bmatrix} * \frac{1}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$$

# Separable filters

A two-dimensional filter kernel is separable if it can be expressed as the outer product of two vectors.

Examples:

$$\frac{1}{4}\begin{bmatrix}1\\2\\1\end{bmatrix} * \frac{1}{4}\begin{bmatrix}1 & 2 & 1\end{bmatrix} = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$$

# Separable filters

Advantage: computational speedup

Example: averaging filter, kernel size = n

- for each pixel, we perform $n^2$ multiplications
- if we separate the filter into two 1-dimensional filters, then each 1-D filter would perform n multiplications
- $n^2$ vs. 2 *n
- for larger n and for larger images the comparison becomes significant

# Sobel Edge Detector

Convolve with:

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

and

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

Gives more weight to the center pixels

# Example – sobel_edge.m

Questions:

What values do we expect to get?

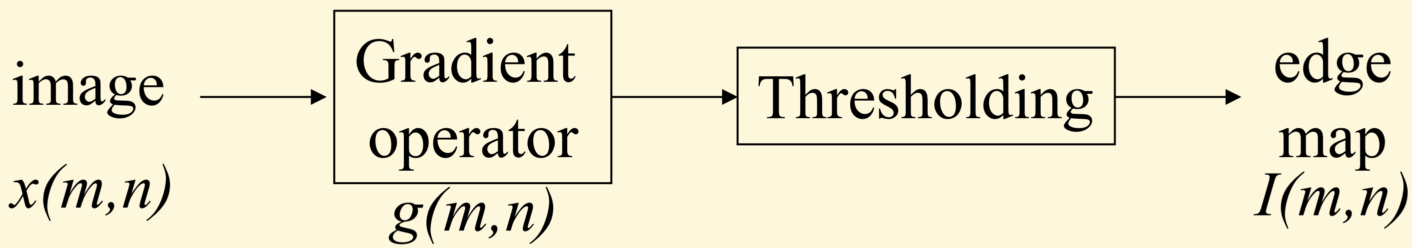
Can we use both Sobel filters?

What can we do with the resulting image?

# Gradient Operators

- Motivation: detect changes

change in the pixel value $\longrightarrow$ large gradient




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

# Edge Detection Using First Derivative

## 1D functions

(not centered at x)

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

$\Rightarrow$ mask: $[-1 \quad 1]$

$$\text{mask } \mathbf{M} = [-1, 0, 1]$$

(centered at x)

(upward) step edge

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | | | 12 | 12 | 12 | 12 | 12 | 24 | 24 | 24 | 24 | 24 |
| $S_1$ | $\otimes$ | $M$ | 0 | 0 | 0 | 0 | 12 | 12 | 0 | 0 | 0 | 0 |

ramp edge

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_3$ | | | 12 | 12 | 12 | 12 | 15 | 18 | 21 | 24 | 24 | 24 |
| $S_3$ | $\otimes$ | $M$ | 0 | 0 | 0 | 3 | 6 | 6 | 6 | 3 | 0 | 0 |

(downward) step edge

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_2$ | | | 24 | 24 | 24 | 24 | 24 | 12 | 12 | 12 | 12 | 12 |
| $S_2$ | $\otimes$ | $M$ | 0 | 0 | 0 | 0 | -12 | -12 | 0 | 0 | 0 | 0 |

roof edge

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_4$ | | | 12 | 12 | 12 | 12 | 24 | 12 | 12 | 12 | 12 | 12 |
| $S_4$ | $\otimes$ | $M$ | 0 | 0 | 0 | 12 | 0 | -12 | 0 | 0 | 0 | 0 |

# Common Operators (cont'd)

1. Prewitt operator

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

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

# Examples

original image

horizontal edge

vertical edge

Prewitt operator

# Edge Detection Using <u>Second</u> 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 Second Derivative (cont'd)

## 1D functions:

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

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

(centered at x+1)

Replace x+1 with x (i.e., centered at x):

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

$$\Downarrow$$

mask: $[1 \quad -2 \quad 1]$

# Common Operators (cont'd)

2. Sobel operator

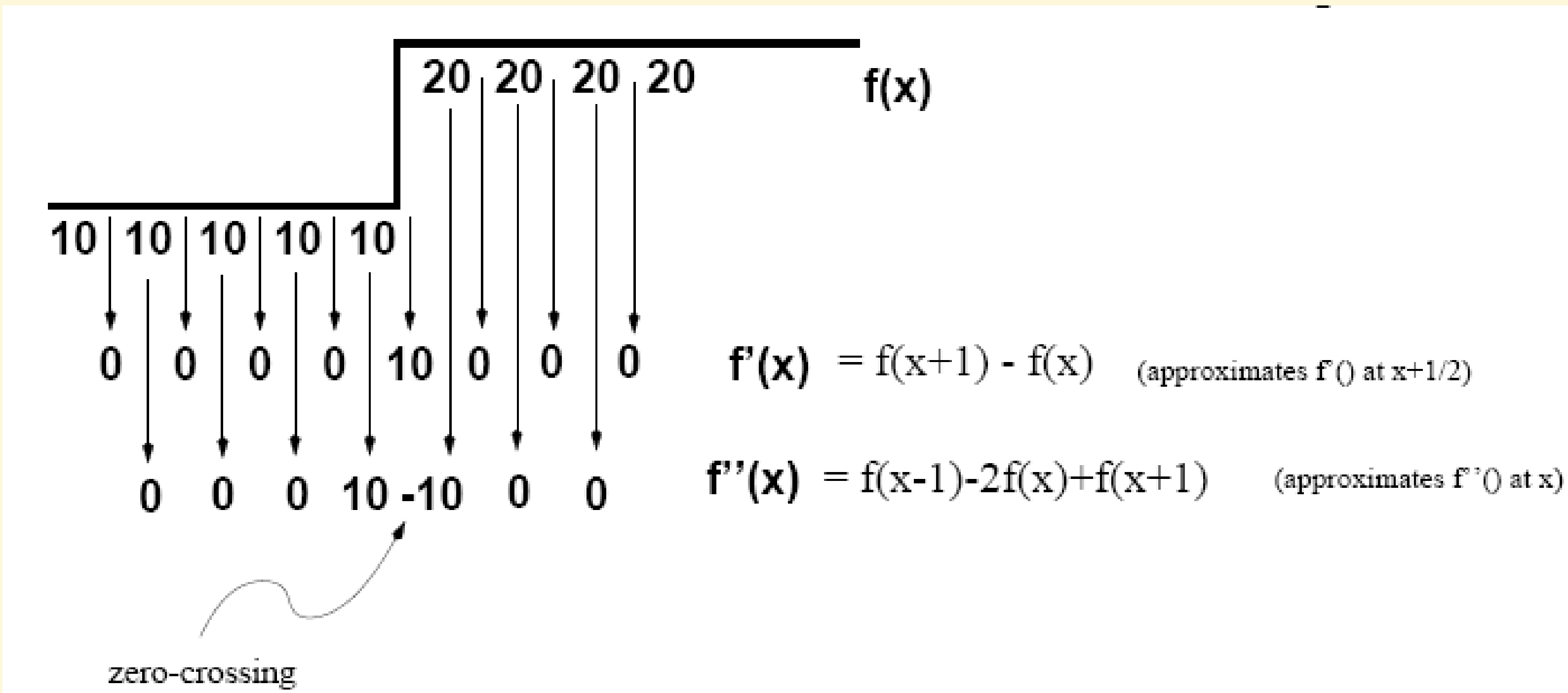
vertical

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

horizontal

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

# Edge Detection Using <u>Second</u> Derivative (cont'd)



# Edge Detection Using Second Derivative (cont'd)

(upward) step edge

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | | | 12 | 12 | 12 | 12 | 12 | 24 | 24 | 24 | 24 | 24 |
| $S_1$ | $\otimes$ | $M$ | 0 | 0 | 0 | 0 | -12 | 12 | 0 | 0 | 0 | 0 |

(downward) step edge

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_2$ | | | 24 | 24 | 24 | 24 | 24 | 12 | 12 | 12 | 12 | 12 |
| $S_2$ | $\otimes$ | $M$ | 0 | 0 | 0 | 0 | 12 | -12 | 0 | 0 | 0 | 0 |

ramp edge

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_3$ | | | 12 | 12 | 12 | 12 | 15 | 18 | 21 | 24 | 24 | 24 |
| $S_3$ | $\otimes$ | $M$ | 0 | 0 | 0 | -3 | 0 | 0 | 0 | 3 | 0 | 0 |

roof edge

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_4$ | | | 12 | 12 | 12 | 12 | 24 | 12 | 12 | 12 | 12 | 12 |
| $S_4$ | $\otimes$ | $M$ | 0 | 0 | 0 | -12 | 24 | -12 | 0 | 0 | 0 | 0 |

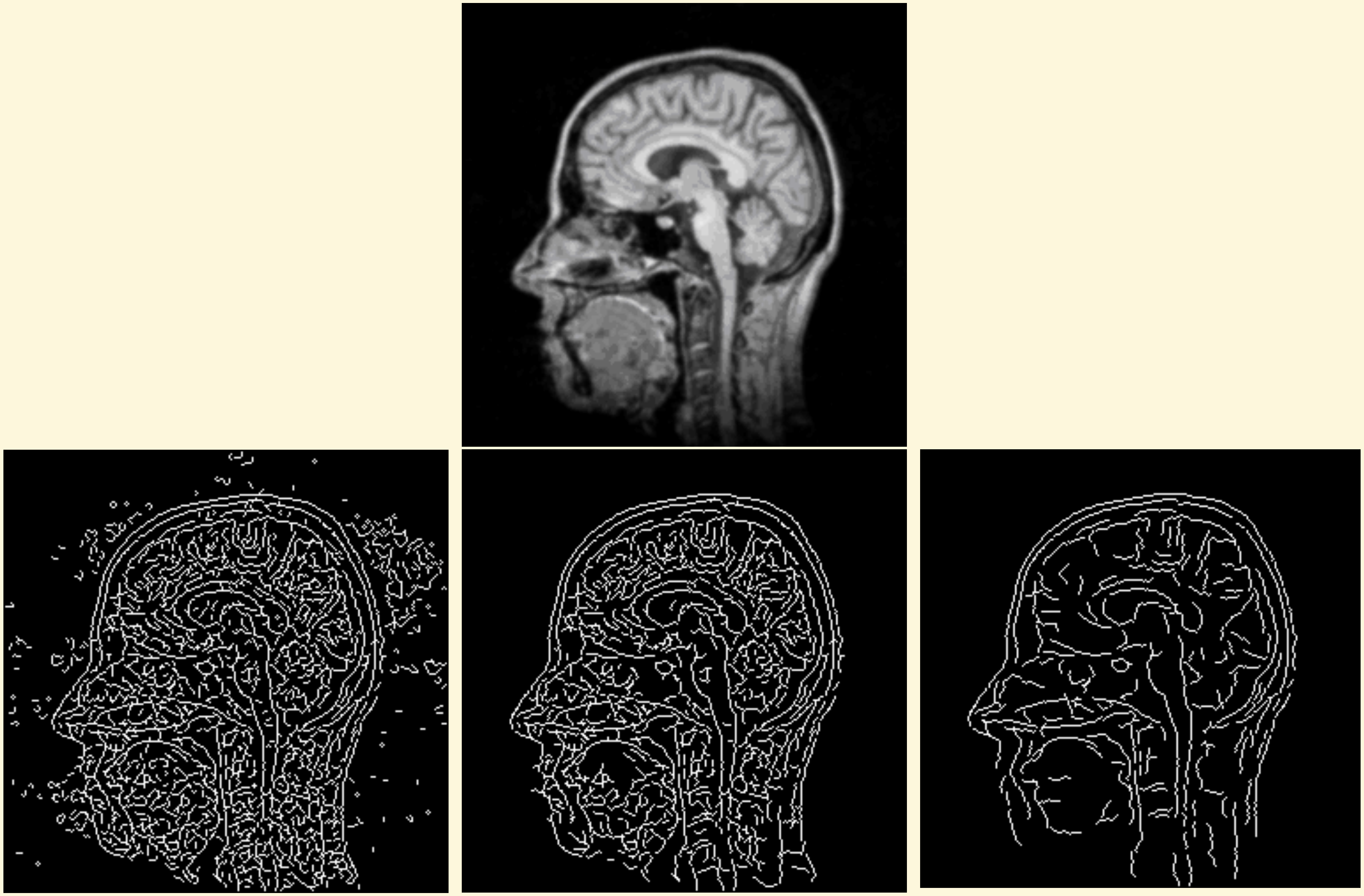
# Edge Detection Using Second Derivative (cont'd)

- Four cases of zero-crossings:

$$\{+,-\},\ \{+,0,-\},\{-,+\},\ \{-,0,+\}$$

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

# Effect of Thresholding Parameters



small → threshold → large

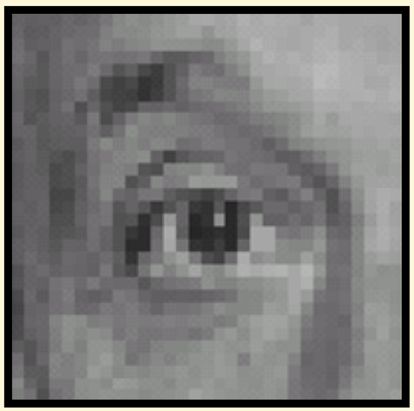
# Example – edge1.m and edge2.m

Using Matlab’s *edge*() built-in function

# Image filtering

- Image filtering: compute function of local neighborhood at each position
- Really important!
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching

# Practice with linear filters



Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

**Sharpening filter**

- Accentuates differences with local average



Source: D. Lowe

# Linear filters

Definition = an output pixel's value is determined as a weighted sum of input pixel values

Properties: Whatever the weights chosen, the output of this procedure is:

- *shift invariant*—meaning that the value of the output depends on the pattern in an image neighborhood, rather than the position of the neighborhood
- *linear*—meaning that the output for the sum of two images is the same as the sum of the outputs obtained for the images separately.

# Convolution

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- Each pixel in output image is
  - weighted average of window of pixels in input image
  - weights stay the same
  - window centered on pixel
  - window weights form

  the convolution **kernel**

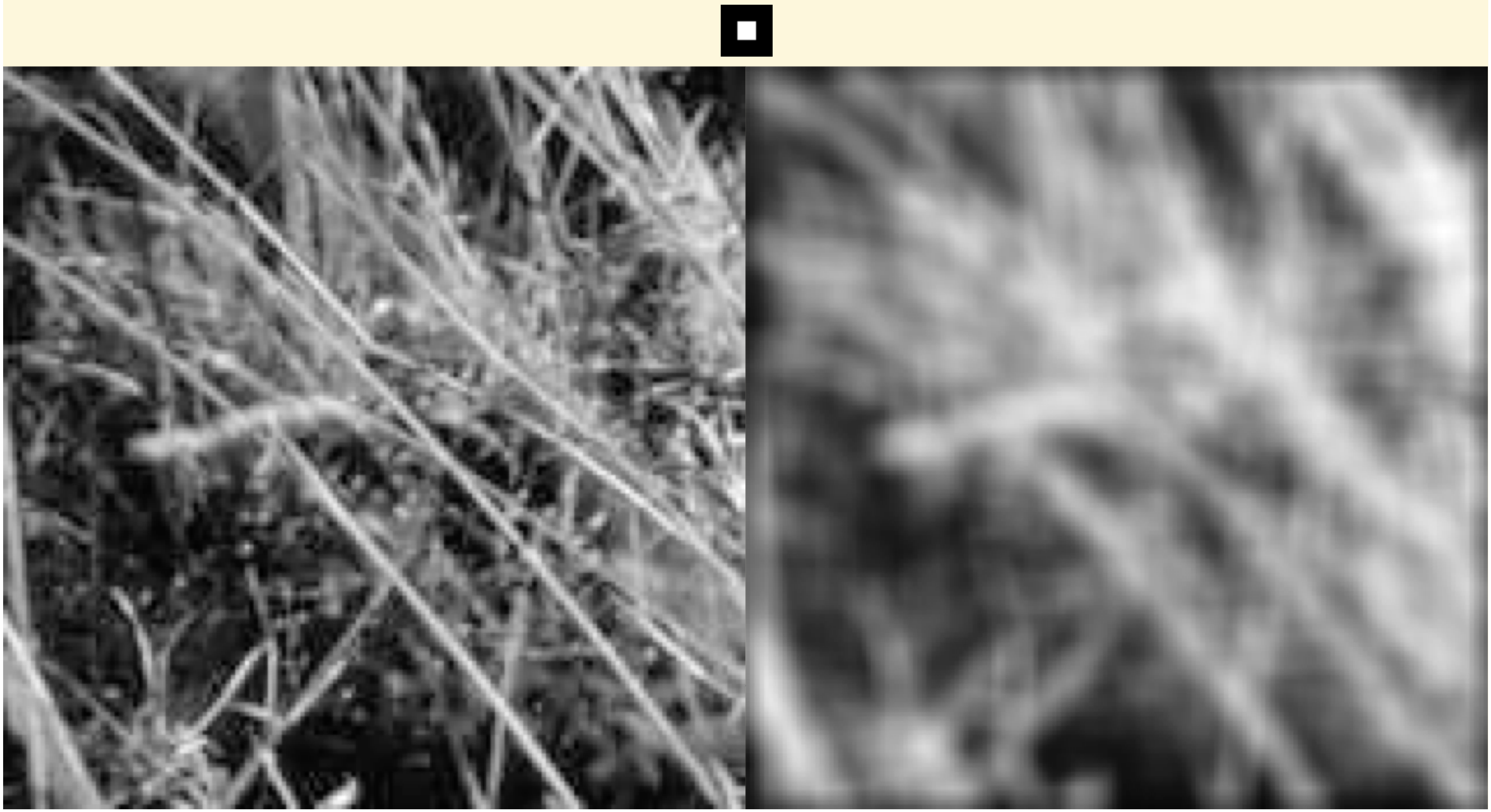
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Convolution

- Linear filtering as a pre-processing stage to edge extraction (Section 4.2 Szeliski) and interest point detection (Section 4.1) algorithms.

- Important operations
  - Example: smoothing by averaging
  - Example: smoothing by weighted average

# Smoothing with box filter



# Convolution - Example I

- Example: replace each pixel with an average of neighbors (mean filter or box filter)
  - (2k+1) by (2k+1) window centered at pixel

$$\mathcal{R}_{ij} = \frac{1}{(2k+1)^2} \sum_{u=i-k}^{u=i+k} \sum_{v=j-k}^{v=j+k} \mathcal{F}_{uv}.$$

i, j' th pixel in output image

u, v' th pixel in input image

If $k$ = 1, then Kernel_size = ?

If $k$ = 2, then Kernel_size = ?

# How big should the mask be?

## The bigger the mask,

- more neighbors contribute
- bigger noise spread.
- more blurring.
- more expensive to compute

## Limitations of averaging

- Signal frequencies shared with noise are lost
- Impulsive noise is diffused but not removed