CAP 5415 Computer Vision

Dr. Mubarak Shah
Univ. of Central Florida

Filtering

Lecture-2

Contents

Contents

- Filtering/Smoothing/Removing Noise

Contents

- Filtering/Smoothing/Removing Noise
- Convolution/Correlation

Contents

- Filtering/Smoothing/Removing Noise
- Convolution/Correlation
- Image Derivatives

Contents

- Filtering/Smoothing/Removing Noise
- Convolution/Correlation
- Image Derivatives
- Histogram

Contents

- Filtering/Smoothing/Removing Noise
- Convolution/Correlation
- Image Derivatives
- Histogram
- Some Matlab Functions

Images: General

Images: General

Binary

Images: General

Binary

Gray Scale

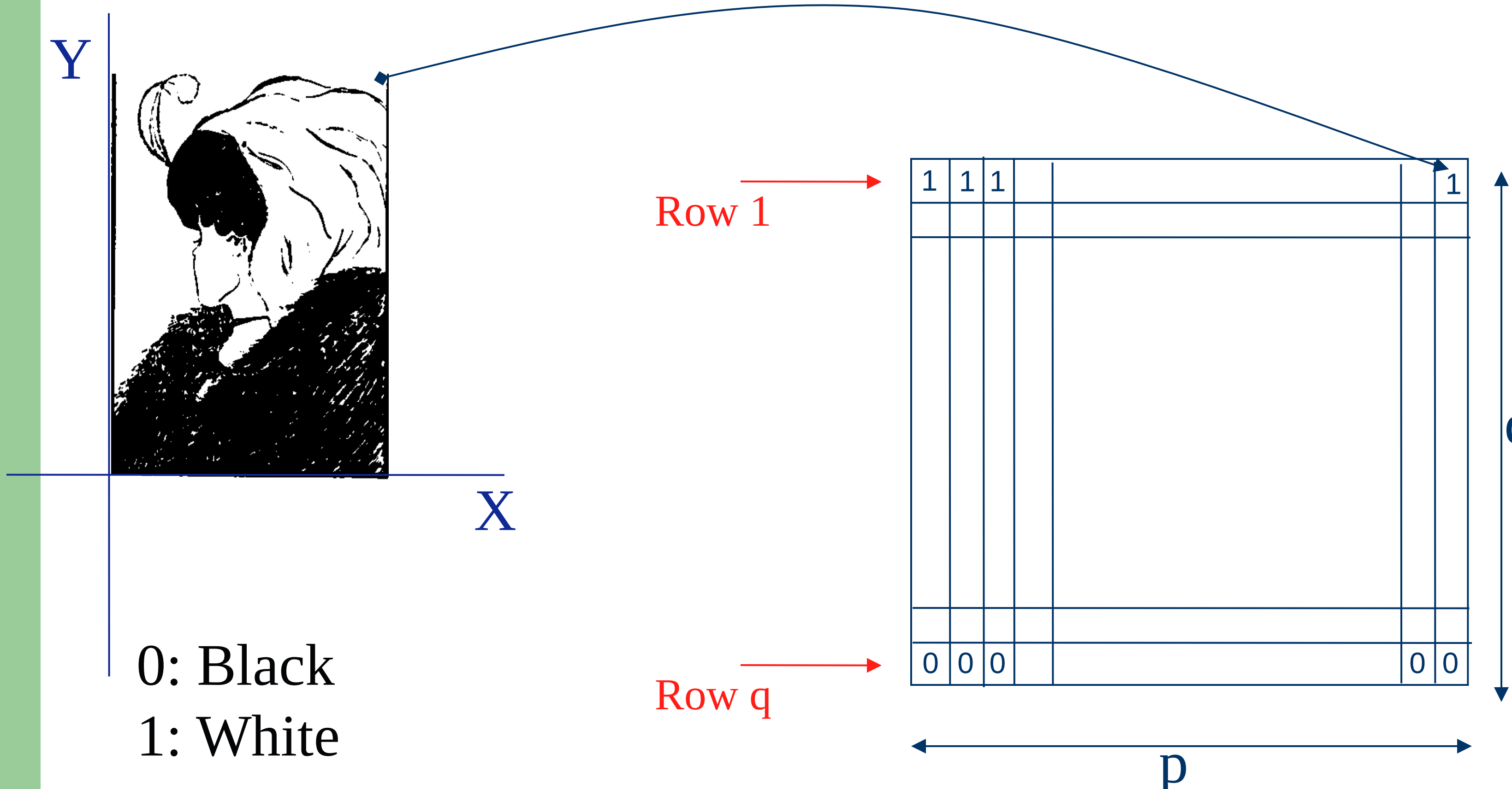
Images: General

Binary

Gray Scale

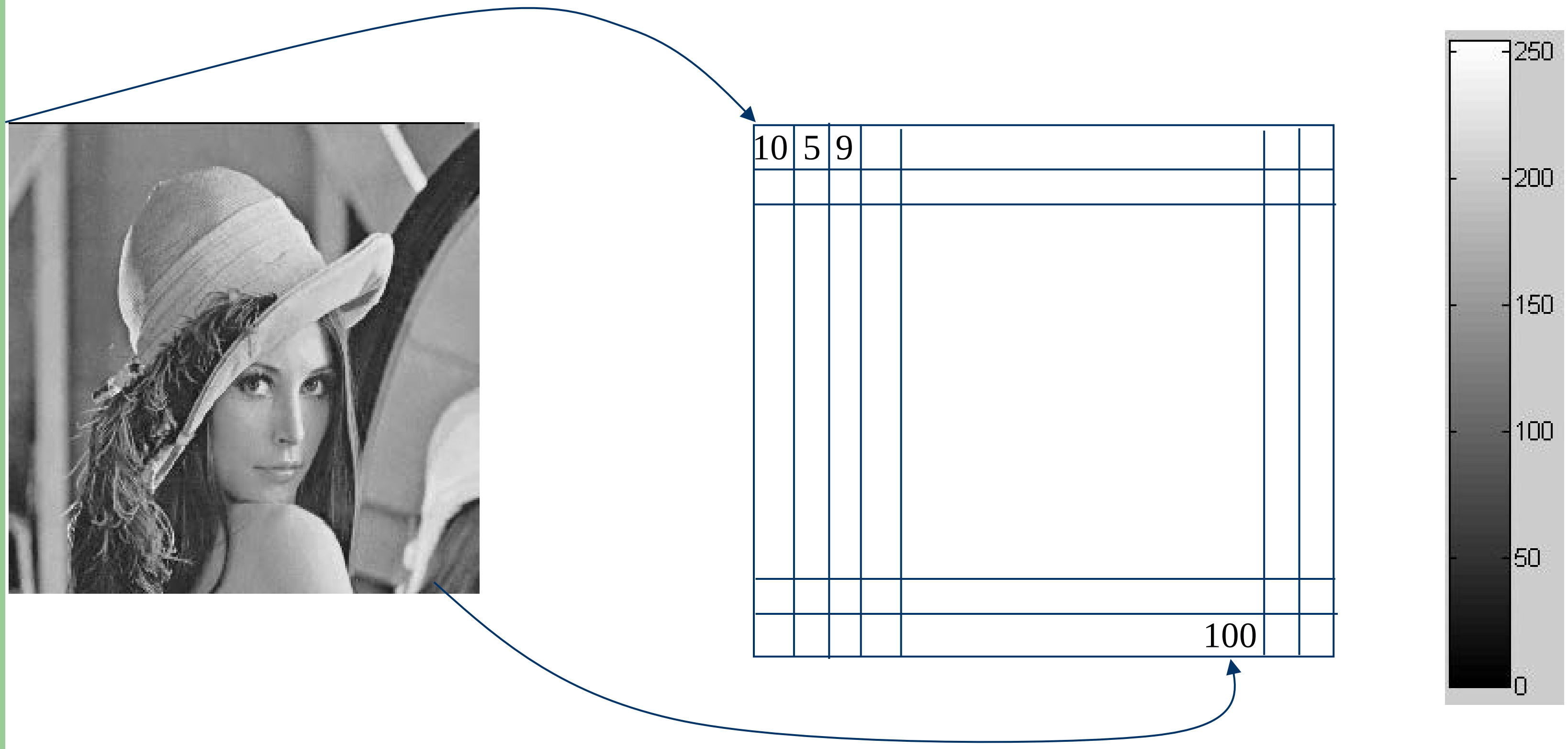
Color

Binary Images

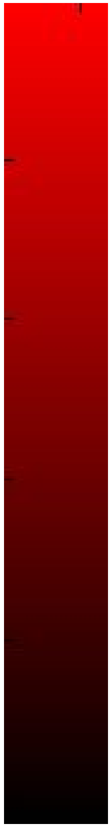


0: Black
1: White

Gray Level Image

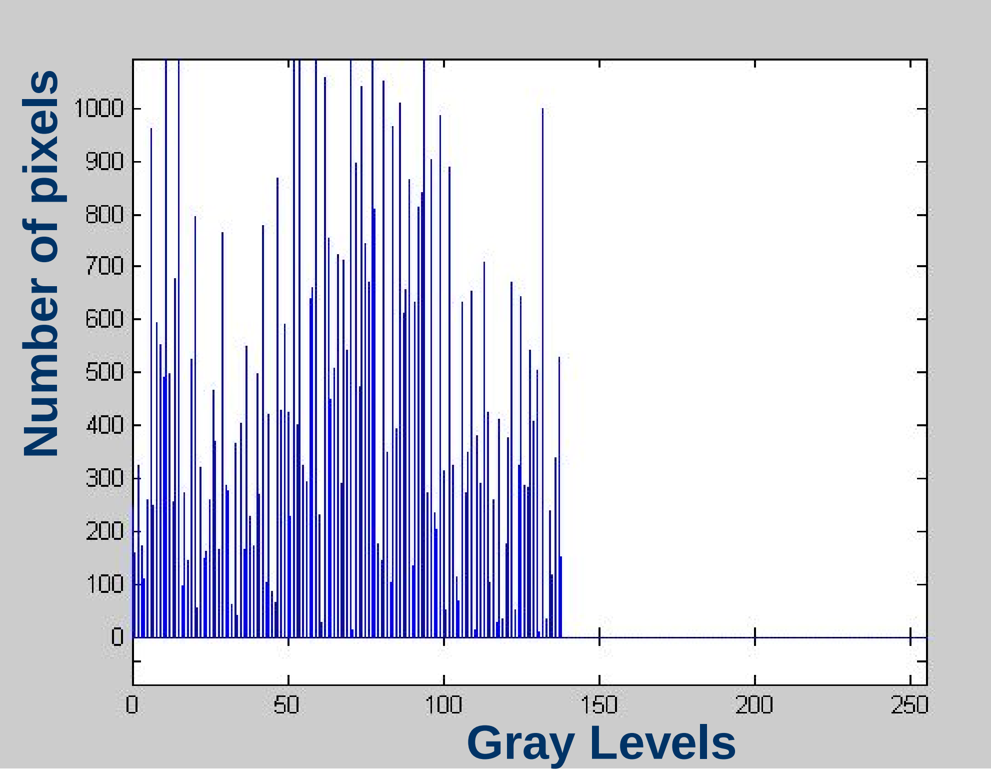


Gray Scale Image



Color Image
Red, Green, Blue Channels

Image Histogram



- Histogram captures the distribution of gray levels in the image.
- How frequently each gray level occurs in the image

Histogram Code

Histogram Code

C Code:

```
for (i=0;i<m,i++)
    for (j=0;j<n,j++)
        hist[I[i,j]]++;
```

Histogram Code

C Code:

```
for (i=0;i<m,i++)
    for (j=0;j<n,j++)
        hist[I[i,j]]++;
```

MATLAB: imhist(I)

Image Noise

Image Noise

- Light Variations

Image Noise

- Light Variations
- Camera Electronics

Image Noise

- Light Variations
- Camera Electronics
- Surface Reflectance

Image Noise

- Light Variations
- Camera Electronics
- Surface Reflectance
- Lens

Image Noise

- Light Variations
- Camera Electronics
- Surface Reflectance
- Lens

- Noise is random, it occurs with some probability

Image Noise

- Light Variations
- Camera Electronics
- Surface Reflectance
- Lens

- Noise is random, it occurs with some probability
- It has a distribution

Image Noise

- $I(x,y)$: the true pixel values

Image Noise

- $I(x,y)$: the true pixel values
- $n(x,y)$: the noise at pixel (x,y)

Image Noise

- $I(x,y)$: the true pixel values
- $n(x,y)$: the noise at pixel (x,y)

Additive noise

Image Noise

- $I(x,y)$: the true pixel values
- $n(x,y)$: the noise at pixel (x,y)

$$\hat{I}(x,y) = I(x,y) + n(x,y)$$ Additive noise

Image Noise

- $I(x,y)$: the true pixel values
- $n(x,y)$: the noise at pixel (x,y)

$$\hat{I}(x,y) = I(x,y) + n(x,y)$$ Additive noise

➔

Image Noise

Image Noise

- $I(x,y)$: the true pixel values

Image Noise

- $I(x,y)$: the true pixel values
- $n(x,y)$: the noise at pixel (x,y)

Image Noise

- $I(x,y)$: the true pixel values
- $n(x,y)$: the noise at pixel (x,y)

Multiplicative noise

Image Noise

- $I(x,y)$: the true pixel values
- $n(x,y)$: the noise at pixel (x,y)

$$\hat{I}(x,y) = I(x,y) \times n(x,y)$$ Multiplicative noise

Image Noise

- $I(x,y)$: the true pixel values
- $n(x,y)$: the noise at pixel (x,y)

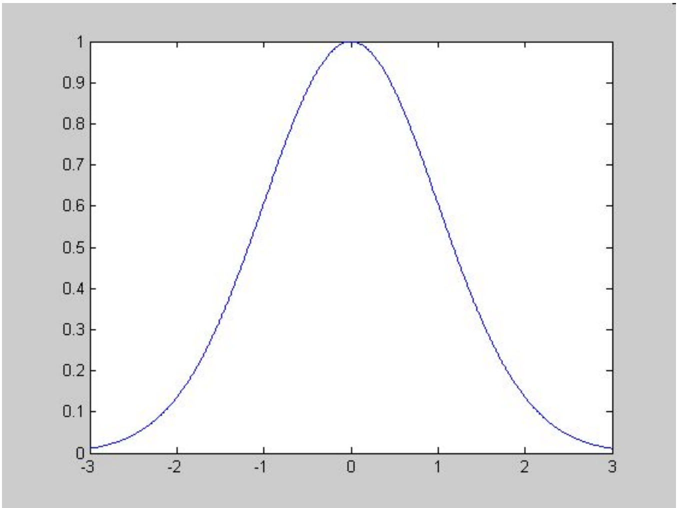
$$\hat{I}(x,y) = I(x,y) \times n(x,y)$$ Multiplicative noise

➔

Gaussian Noise

$$n(x,y) \approx g(n) = e^{\frac{-n^2}{2\sigma^2}}$$

$g(n)$

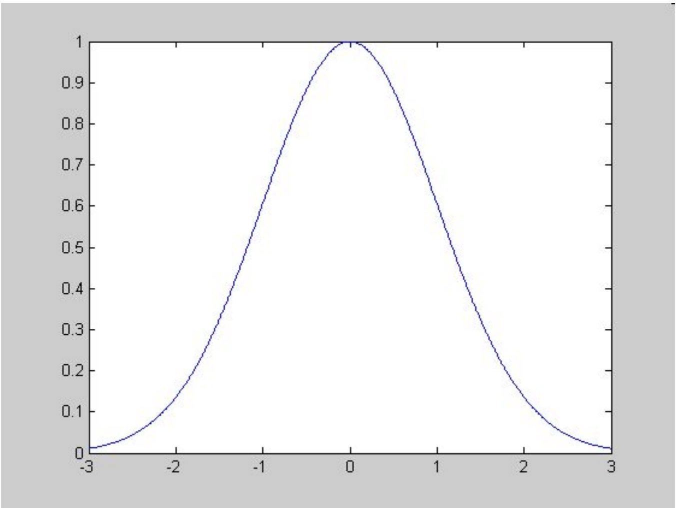


n

Gaussian Noise

$$n(x, y) \approx g(n) = e^{\frac{-n^2}{2\sigma^2}}$$

$g(n)$



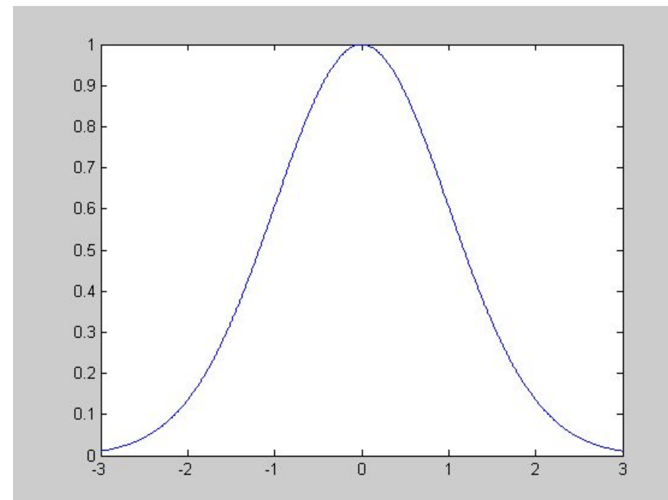
n

Probability Distribution
n is a random variable

Gaussian Noise

$$n(x,y) \approx g(n) = e^{\frac{-n^2}{2\sigma^2}}$$

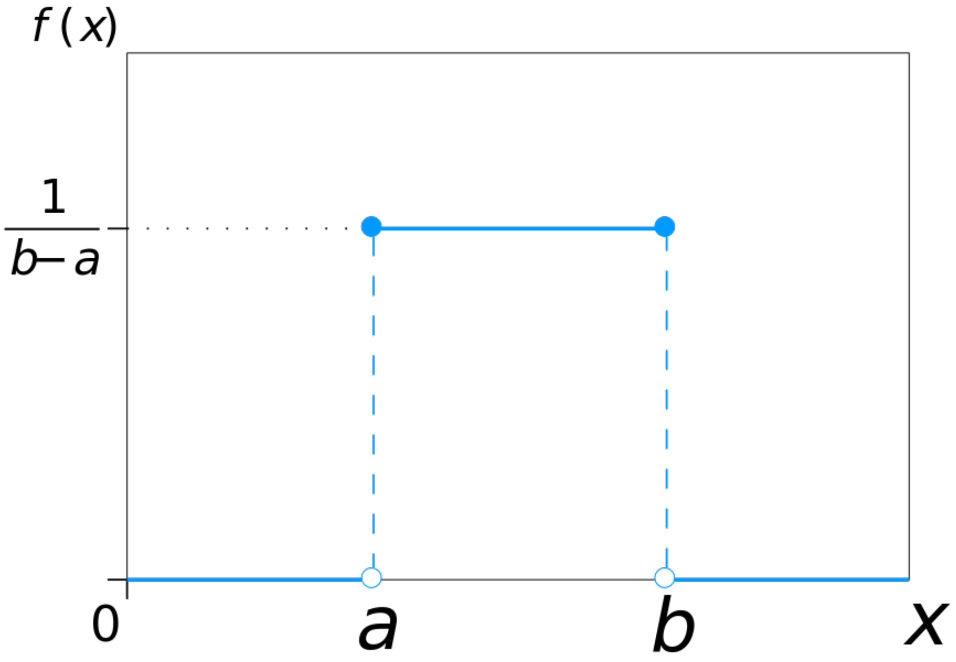
$g(n)$



n

Probability Distribution
n is a random variable

Uniform Distribution



Salt and Pepper Noise

- Each pixel is randomly made black or white with a uniform probability distribution

➔

Image filtering

- Image filtering: compute function of local neighborhood at each position

Slide Credit: James Hays, Cornell U.

Image filtering

- Image filtering: compute function of local neighborhood at each position

- Really important!
  - Enhance images
    - Denoise, resize, increase contrast, etc.

Slide Credit: James Hays, Cornell U.

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.

Slide Credit: James Hays, Cornell U.

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

Slide Credit: James Hays, Cornell U.

Image Derivatives & Averages

Definitions

Definitions

- Derivative: Rate of change
  - *Speed* is a rate of change of a *distance*
  - *Acceleration* is a rate of change of *speed*

Definitions

- Derivative: Rate of change
  - *Speed* is a rate of change of a *distance*
  - *Acceleration* is a rate of change of *speed*
- Average (Mean)
  - Dividing the sum of N values by N

Derivative

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

Derivative

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

$v = \dfrac{ds}{dt}$ speed $\qquad$ $a = \dfrac{dv}{dt}$ acceleration

Examples: Analytic Derivatives

Examples: Analytic Derivatives

$$y = x^2 + x^4$$

$$\frac{dy}{dx} = 2x + 4x^3$$

Examples: Analytic Derivatives

$$y = x^2 + x^4$$

$$\frac{dy}{dx} = 2x + 4x^3$$

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

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

Discrete Derivative

Discrete Derivative

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

Discrete Derivative

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

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

Discrete Derivative

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

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

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

Discrete Derivative
Finite Difference

Discrete Derivative
Finite Difference

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

Discrete Derivative
Finite Difference

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

Backward difference

Discrete Derivative
Finite Difference

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

Backward difference

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

Discrete Derivative
Finite Difference

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

Backward difference

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

Forward difference

Discrete Derivative
Finite Difference

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

Backward difference

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

Forward difference

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

Discrete Derivative
Finite Difference

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

Backward difference

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

Forward difference

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

Central difference

Example

Example

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

Example

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

Example

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

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

Example

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

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

Example

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

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

$$f''(x) = 0 \quad 5 \quad 10 \quad 5 \quad 15 \quad -20 \quad 5 \quad 0$$

Example

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

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

$$f''(x) = 0 \quad 5 \quad 10 \quad 5 \quad 15 \quad -20 \quad 5 \quad 0$$

Derivative Masks

Example

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

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

$$f''(x) = 0 \quad 5 \quad 10 \quad 5 \quad 15 \quad -20 \quad 5 \quad 0$$

Derivative Masks

Backward difference	[-1 1]
Forward difference	[1 -1]
Central difference	[-1 0 1]

Derivatives in 2 Dimensions

Derivatives in 2 Dimensions

Given function $f(x, y)$

Derivatives in 2 Dimensions

Given function $$f(x, y)$$

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

Derivatives in 2 Dimensions

Given function $f(x, y)$

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

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

Derivatives in 2 Dimensions

Given function $$f(x, y)$$

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

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

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

Derivatives of Images

Derivative masks

Derivatives of Images

Derivative masks

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

Derivatives of Images

Derivative masks

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

Derivatives of Images

Derivative masks

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

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

Derivatives of Images

Derivative masks

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

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

Derivatives of Images

Derivative masks

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

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Derivatives of Images

Derivative masks

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

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \boxed{10} & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Derivatives of Images

Derivative masks $\quad f_x \Rightarrow \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad f_y \Rightarrow \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \boxed{10} & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Derivatives of Images

Derivative masks $$f_x \Rightarrow \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad f_y \Rightarrow \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \boxed{10} & \boxed{10} & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Derivatives of Images

Derivatives of Images

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

Derivatives of Images

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

Correlation

Correlation

$$f \otimes h = \sum_k \sum_l f(k,l)\,h(k,l)$$

Correlation

$$f \otimes h = \sum_k \sum_l f(k,l)\,h(k,l)$$

f = Image

h = Kernel

Correlation

$$f \otimes h = \sum_{k} \sum_{l} f(k,l)h(k,l)$$

f = Image

h = Kernel

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

Correlation

$$f \otimes h = \sum_k \sum_l f(k,l)h(k,l)$$

f = Image

h = Kernel

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

$\otimes$

Correlation

$$f \otimes h = \sum_k \sum_l f(k,l)h(k,l)$$

f = Image

h = Kernel

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

$\otimes$

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

Correlation

$$f \otimes h = \sum_{k} \sum_{l} f(k,l)h(k,l)$$

f = Image

h = Kernel

$$f \qquad\qquad\qquad h$$

$$\begin{array}{|c|c|c|}\hline f_1 & f_2 & f_3 \\ \hline f_4 & f_5 & f_6 \\ \hline f_7 & f_8 & f_9 \\ \hline \end{array} \otimes \begin{array}{|c|c|c|}\hline h_1 & h_2 & h_3 \\ \hline h_4 & h_5 & h_6 \\ \hline h_7 & h_8 & h_9 \\ \hline \end{array}$$

Correlation

$$f \otimes h = \sum_k \sum_l f(k,l)h(k,l)$$

f = Image

h = Kernel

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

$\otimes$

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$$\begin{aligned} f \otimes h = {} & f_1 h_1 + f_2 h_2 + f_3 h_3 \\ & + f_4 h_4 + f_5 h_5 + f_6 h_6 \\ & + f_7 h_7 + f_8 h_8 + f_9 h_9 \end{aligned}$$

Convolution

Convolution

$$f * h = \sum_{k} \sum_{l} f(k,l)\, h(-k,-l)$$

Convolution

$$f * h = \sum_k \sum_l f(k,l)\,h(-k,-l)$$

f = Image

h = Kernel

Convolution

$$f * h = \sum_k \sum_l f(k,l)\,h(-k,-l)$$

f = Image

h = Kernel

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

Convolution

$$f * h = \sum_k \sum_l f(k,l)\, h(-k,-l)$$

f = Image

h = Kernel

$X - flip$

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

Convolution

$$f * h = \sum_{k} \sum_{l} f(k,l)\,h(-k,-l)$$

f = Image

h = Kernel

$$\begin{bmatrix} h_7 & h_8 & h_9 \\ h_4 & h_5 & h_6 \\ h_1 & h_2 & h_3 \end{bmatrix} \xleftarrow{\;X - flip\;} \overset{h}{\begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}}$$

Convolution

$$f * h = \sum_{k} \sum_{l} f(k,l)\, h(-k,-l)$$

f = Image

h = Kernel

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$X - flip$

h_7	h_8	h_9
h_4	h_5	h_6
h_1	h_2	h_3

$Y - flip$

Convolution

$$f * h = \sum_k \sum_l f(k,l)\,h(-k,-l)$$

f = Image

h = Kernel

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$X-flip$

h_7	h_8	h_9
h_4	h_5	h_6
h_1	h_2	h_3

$Y-flip$

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

Convolution

$$f * h = \sum_k \sum_l f(k,l)\,h(-k,-l)$$

f = Image

h = Kernel

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$X - flip$

h_7	h_8	h_9
h_4	h_5	h_6
h_1	h_2	h_3

$Y - flip$

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

Convolution

$$f * h = \sum_k \sum_l f(k,l)\,h(-k,-l)$$

f = Image

h = Kernel

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$X - flip$

h_7	h_8	h_9
h_4	h_5	h_6
h_1	h_2	h_3

$Y - flip$

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

$*$

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

Convolution

$$f * h = \sum_k \sum_l f(k,l)\,h(-k,-l)$$

f = Image

h = Kernel

$$\begin{bmatrix} h_7 & h_8 & h_9 \\ h_4 & h_5 & h_6 \\ h_1 & h_2 & h_3 \end{bmatrix} \xleftarrow{X - flip} h = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

$Y - flip$

$$f = \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} * \begin{bmatrix} h_9 & h_8 & h_7 \\ h_6 & h_5 & h_4 \\ h_3 & h_2 & h_1 \end{bmatrix} \rightarrow$$

$$\begin{aligned} f * h = {} & f_1 h_9 + f_2 h_8 + f_3 h_7 \\ & + f_4 h_6 + f_5 h_5 + f_6 h_4 \\ & + f_7 h_3 + f_8 h_2 + f_9 h_1 \end{aligned}$$

Correlation and Convolution

- Convolution is associative

Correlation and Convolution

- Convolution is associative

$$F * (G * I) = (F * G) * I$$

Averages

- Mean

Averages

- Mean

$$I = \frac{I_1 + I_2 + \ldots I_n}{n} = \frac{\sum_{i=1}^{n} I_i}{n}$$

Averages

- Mean

$$I = \frac{I_1 + I_2 + \ldots I_n}{n} = \frac{\sum_{i=1}^{n} I_i}{n}$$

- Weighted mean

Averages

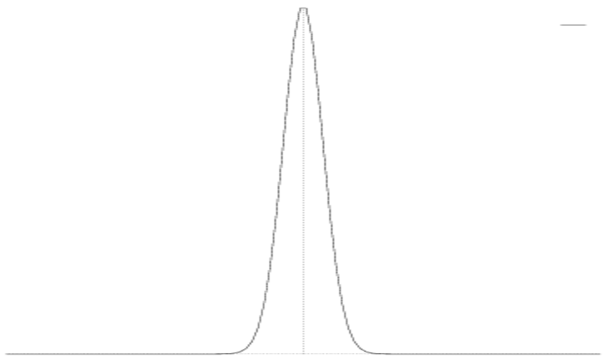
- Mean

$$I = \frac{I_1 + I_2 + \ldots I_n}{n} = \frac{\sum_{i=1}^{n} I_i}{n}$$

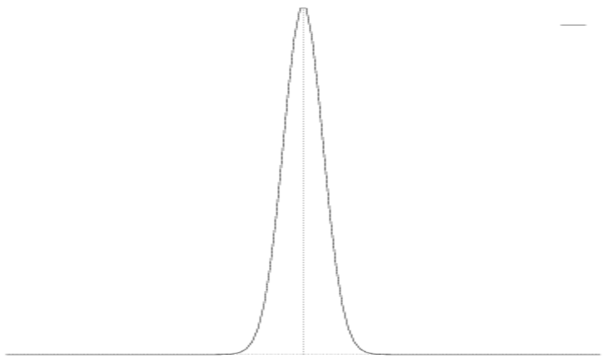
- Weighted mean

$$I = \frac{w_1 I_1 + w_2 I_2 + \ldots + w_n I_n}{n} = \frac{\sum_{i=1}^{n} w_i I_i}{n}$$

Gaussian Filter

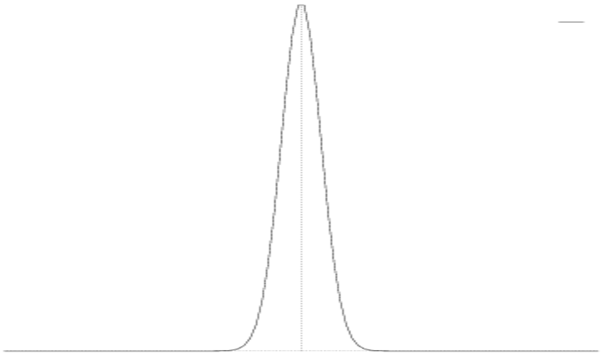


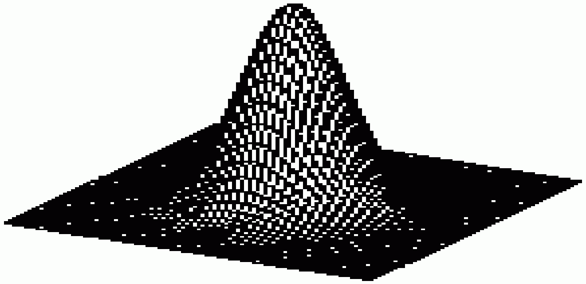
Gaussian Filter



$$g(x) = e^{\frac{-x^2}{2o^2}}$$

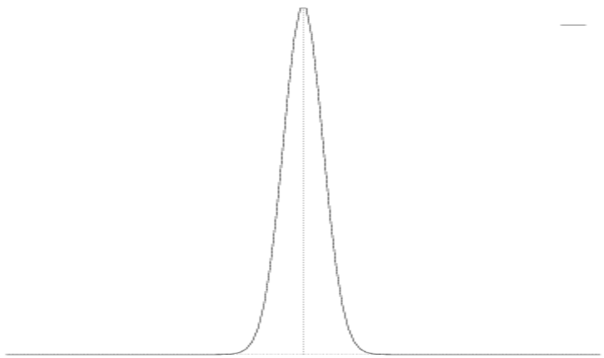
Gaussian Filter

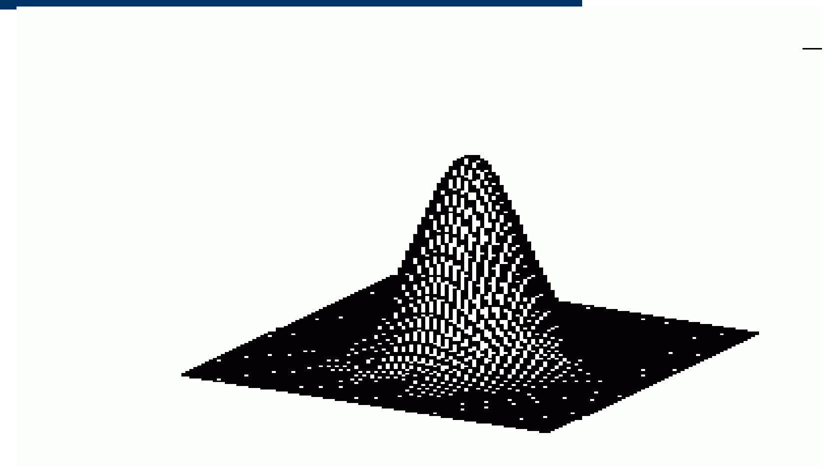




$$g(x) = e^{\frac{-x^2}{2\sigma^2}}$$

Gaussian Filter

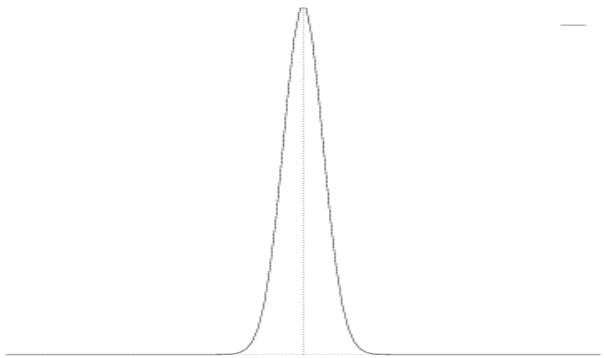


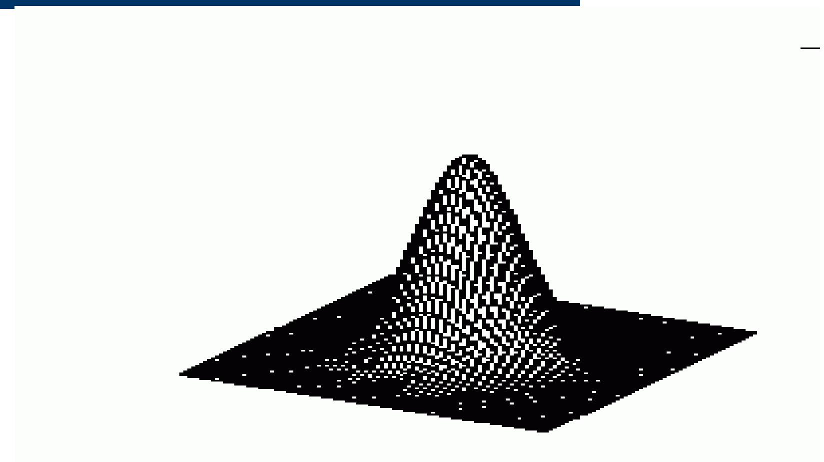


$$g(x) = e^{\frac{-x^2}{2o^2}}$$

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

Gaussian Filter



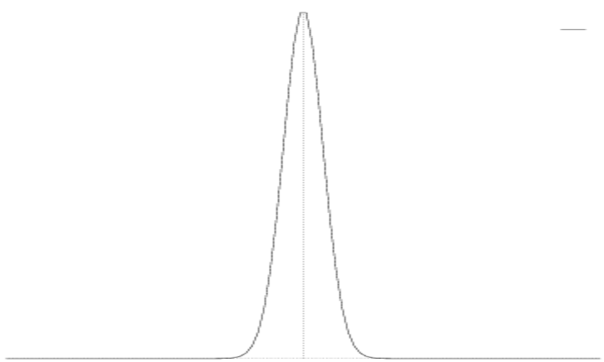


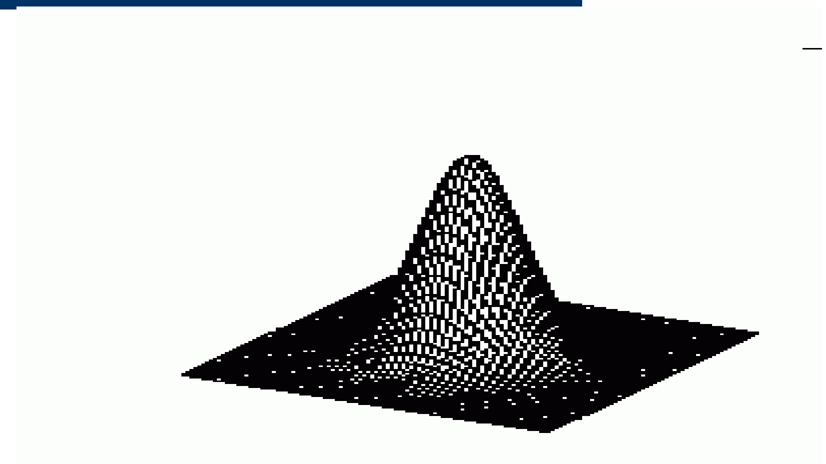
$$g(x) = e^{\frac{-x^2}{2o^2}}$$

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

$$g(x) = [.011 \quad .13 \quad .6 \quad 1 \quad .6 \quad .13 \quad .011]$$

Gaussian Filter





$$g(x) = e^{\frac{-x^2}{2o^2}}$$

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

$$g(x) = \begin{bmatrix} .011 & .13 & .6 & 1 & .6 & .13 & .011 \end{bmatrix}$$

$$\sigma = 1$$

Properties of Gaussian

Properties of Gaussian

- Most common natural model

Properties of Gaussian

- Most common natural model
- Smooth function, it has infinite number of derivatives

Properties of Gaussian

- Most common natural model
- Smooth function, it has infinite number of derivatives
- It is Symmetric

Properties of Gaussian

- Most common natural model
- Smooth function, it has infinite number of derivatives
- It is Symmetric
- Fourier Transform of Gaussian is Gaussian.

Properties of Gaussian

- Most common natural model
- Smooth function, it has infinite number of derivatives
- It is Symmetric
- Fourier Transform of Gaussian is Gaussian.
- Convolution of a Gaussian with itself is a Gaussian.

Properties of Gaussian

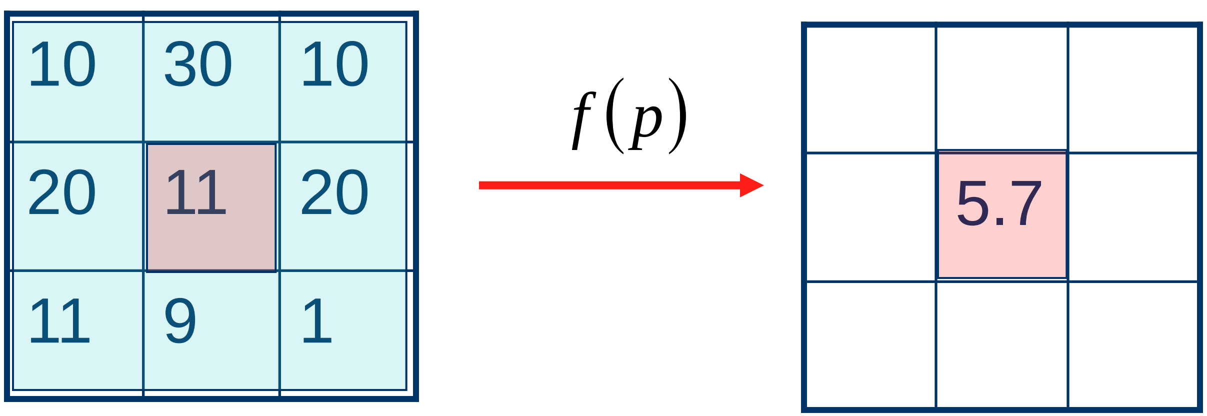
- Most common natural model
- Smooth function, it has infinite number of derivatives
- It is Symmetric
- Fourier Transform of Gaussian is Gaussian.
- Convolution of a Gaussian with itself is a Gaussian.
- Gaussian is separable; 2D convolution can be performed by two 1-D convolutions

Properties of Gaussian

- Most common natural model
- Smooth function, it has infinite number of derivatives
- It is Symmetric
- Fourier Transform of Gaussian is Gaussian.
- Convolution of a Gaussian with itself is a Gaussian.
- Gaussian is separable; 2D convolution can be performed by two 1-D convolutions
- There are cells in eye that perform Gaussian filtering.

Filtering

- Modify pixels based on some function of the neighborhood



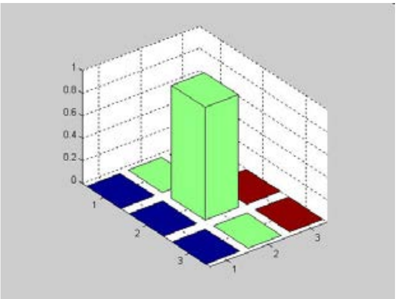
Linear Filtering

- The output is the linear combination of the neighborhood pixels

$$\begin{bmatrix} 1 & 3 & 0 \\ 2 & 10 & 2 \\ 4 & 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0.1 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} & & \\ & 5 & \\ & & \end{bmatrix}$$

Image — Kernel — Filter Output

Filtering Examples



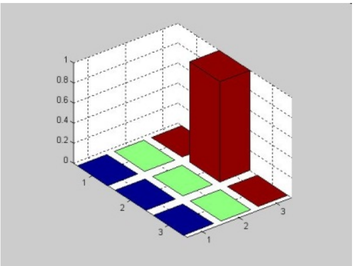
*

0	0	0
0	1	0
0	0	0

=

Filtering Examples

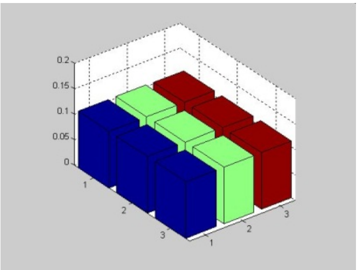
*



0	0	0
0	0	1
0	0	0

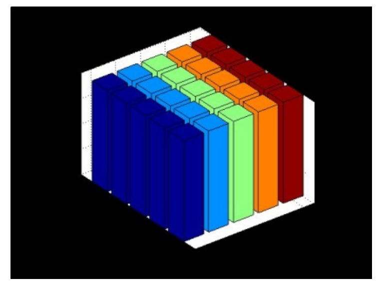
=

Filtering Examples

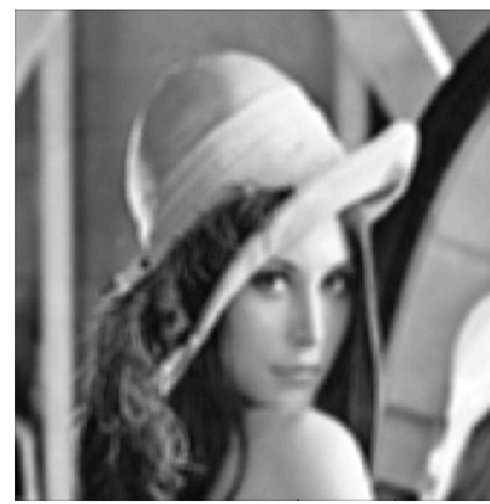


$$* \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

Filtering Examples

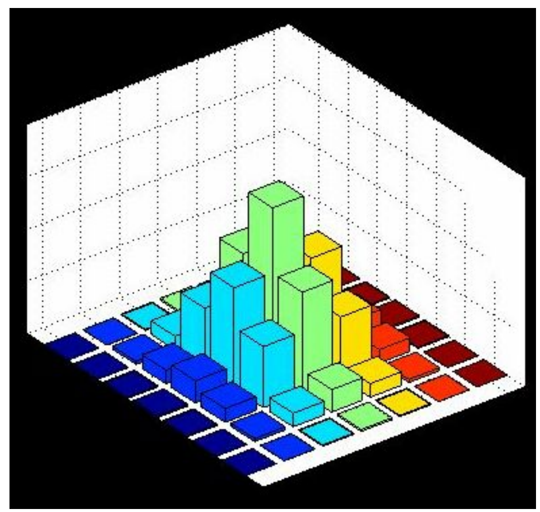


$$* \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$



Filtering Gaussian

*



=

Gaussian vs. Averaging

Gaussian Smoothing

Smoothing by Averaging

Noise Filtering

After additive Gaussian Noise

After Averaging

After Gaussian Smoothing

Example: box filter

$$g[\cdot\,,\cdot\,]$$

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

Slide credit: David Lowe (UBC)

Image filtering

$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$f[.,.]$

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

$h[.,.]$

	0								

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$f[.,.]$

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

$h[.,.]$

	0	10							

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[.,.]$

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

$h[.,.]$

	0	10	20						

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[.,.]$

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

$h[.,.]$

	0	10	20	30					

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

	0	10	20	30	30				

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot\,,\cdot\,]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

	0	10	20	30	30				
				?					

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$f\,[.,.]$

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

$h[.,.]$

	0	10	20	30	30				
						$\boxed{?}$			
				50					

$$h[m,n]=\sum_{k,l} g[k,l]\ f\,[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot,\cdot] \quad \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$$f[.,.]$$

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

$$h[.,.]$$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

Credit: S. Seitz

Box Filter

What does it do?

$$g[\cdot,\cdot]$$

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

Slide credit: David Lowe (UBC)

Box Filter

What does it do?

•Replaces each pixel with an average of its neighborhood

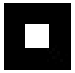
•Achieve smoothing effect (remove sharp features)

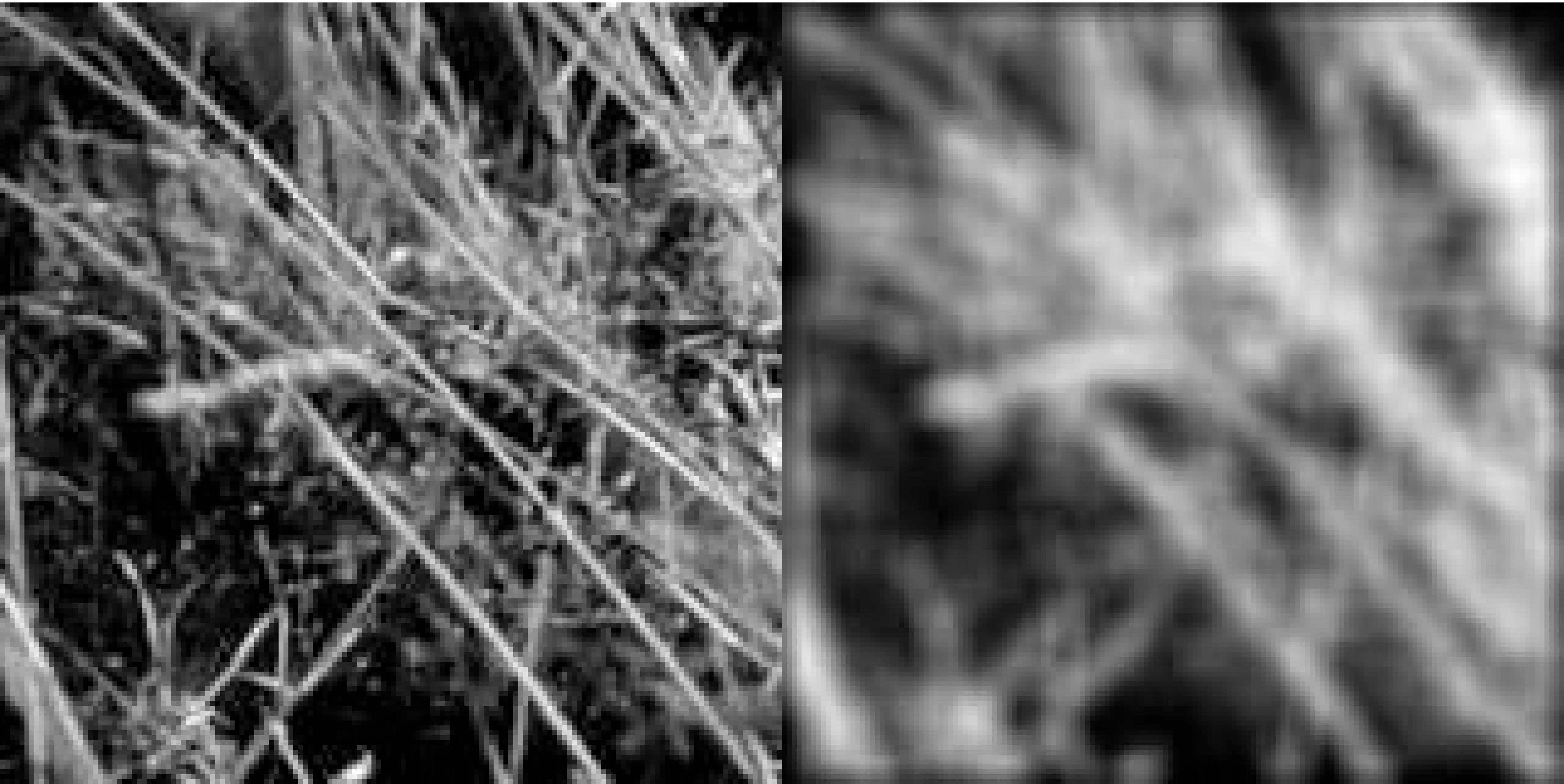
$$g[\cdot\,,\cdot\,]$$

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Slide credit: David Lowe (UBC)

Smoothing with box filter





Practice with linear filters

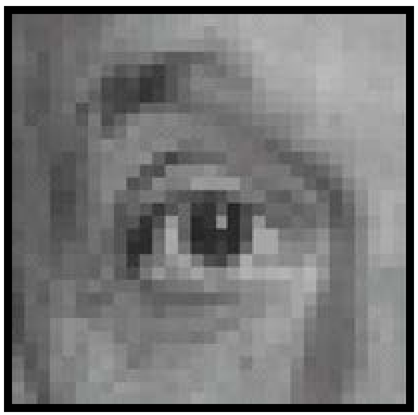
Original

0	0	0
0	1	0
0	0	0

?

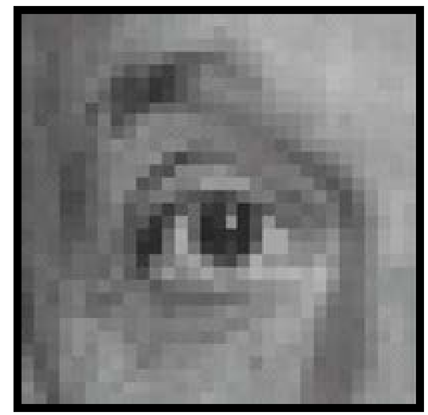
Source: D. Lowe

Practice with linear filters



Original

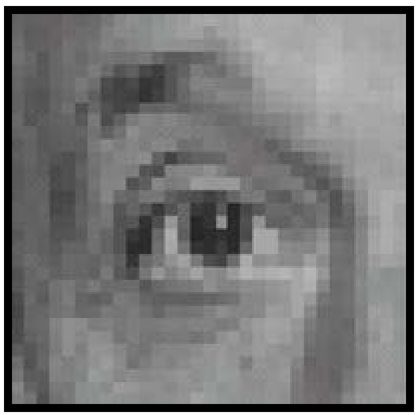
0	0	0
0	1	0
0	0	0



Filtered
(no change)

Source: D. Lowe

Practice with linear filters



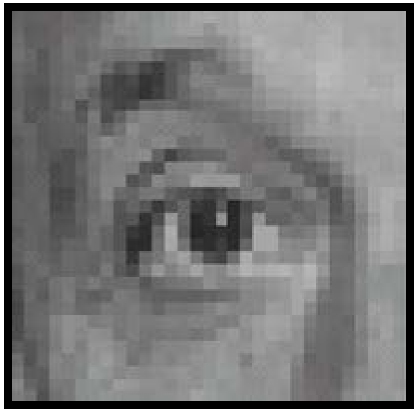
Original

0	0	0
0	0	1
0	0	0

?

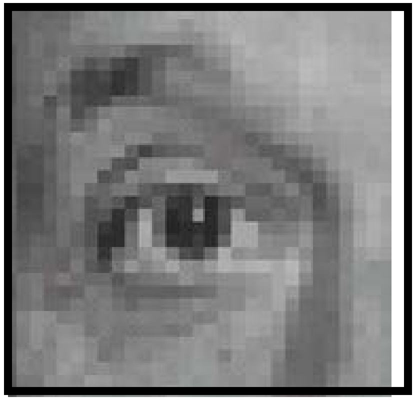
Source: D. Lowe

Practice with linear filters



Original

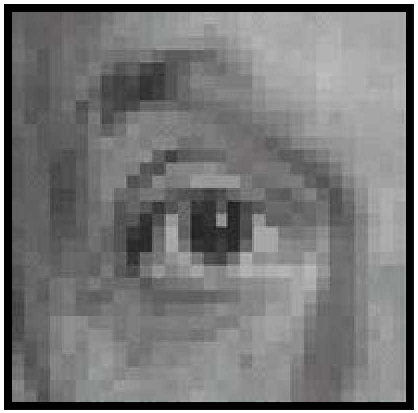
0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Source: D. Lowe

Practice with linear filters



Original

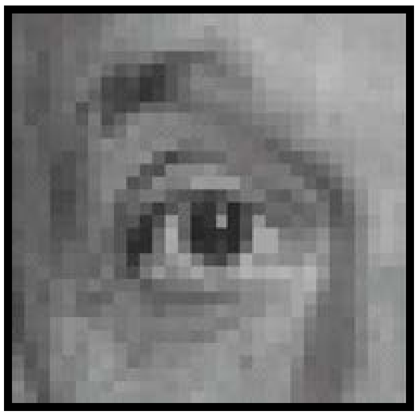
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

?

(Note that filter sums to 1)

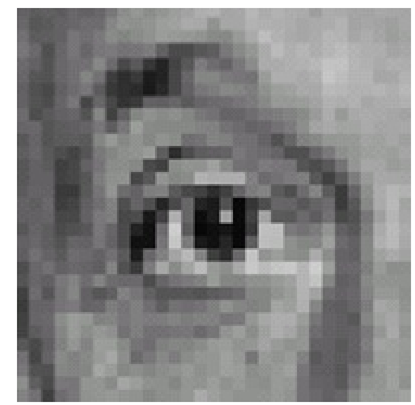
Source: D. Lowe

Practice with linear filters



Original

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

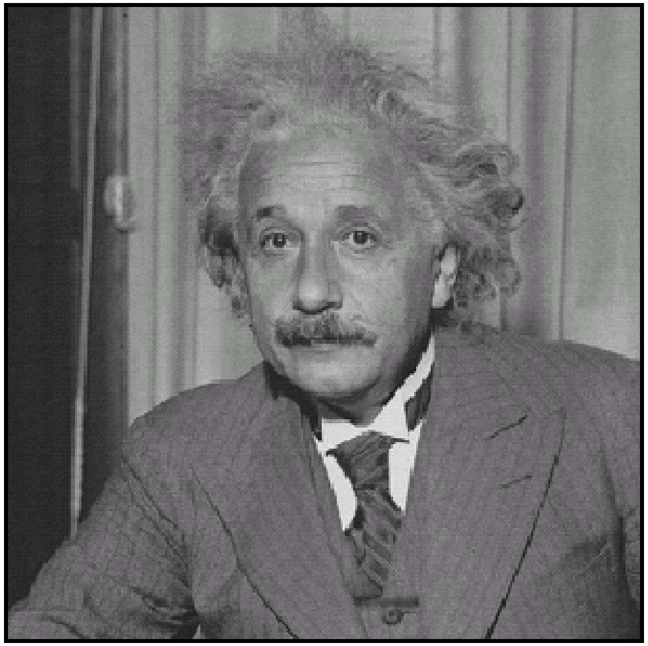


Sharpening filter

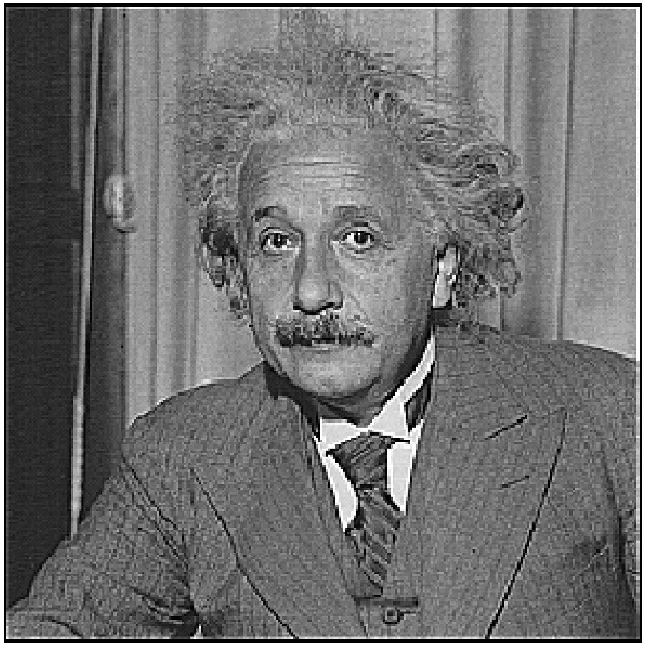
- Accentuates differences with local average

Source: D. Lowe

Sharpening



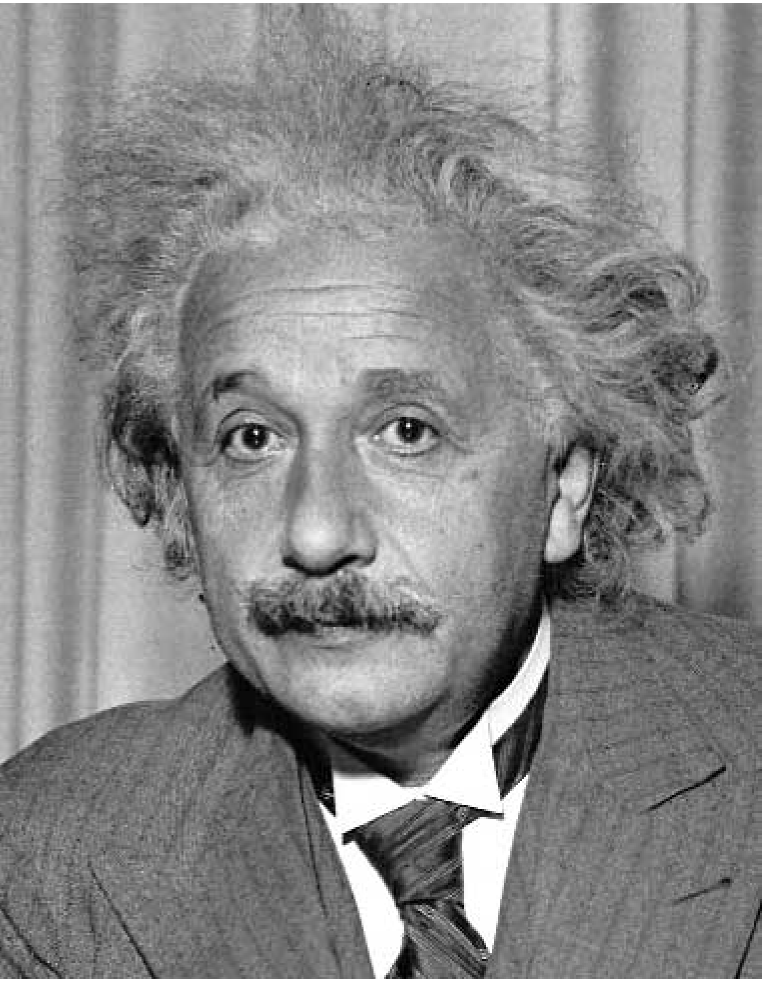
before



after

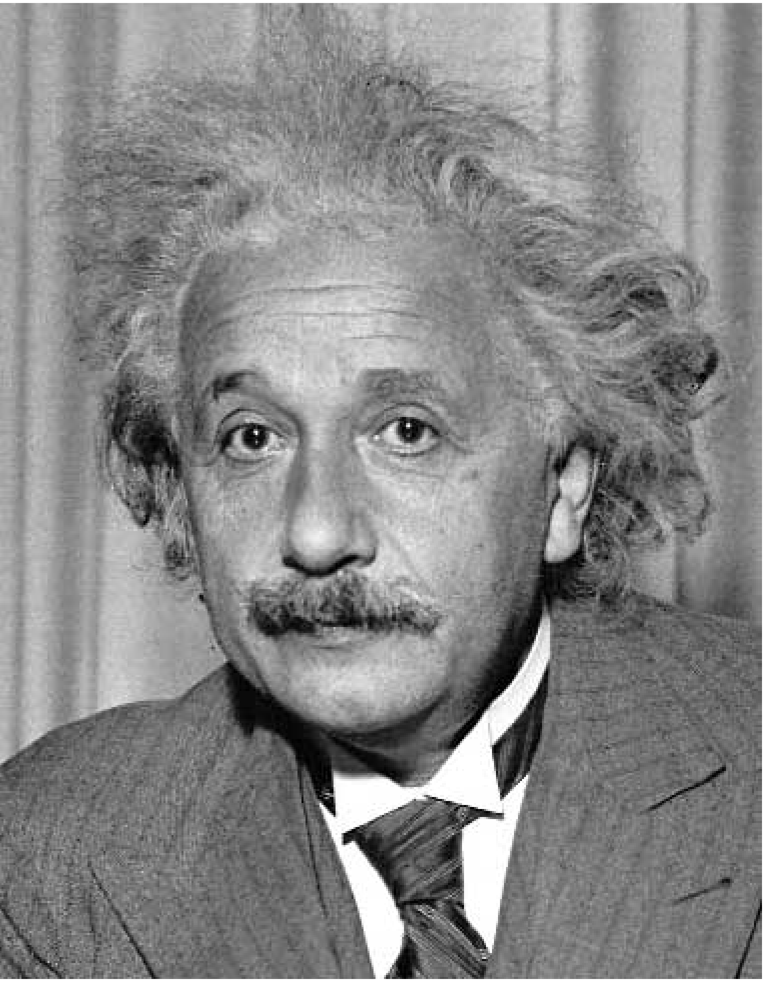
Source: D. Lowe

Other filters



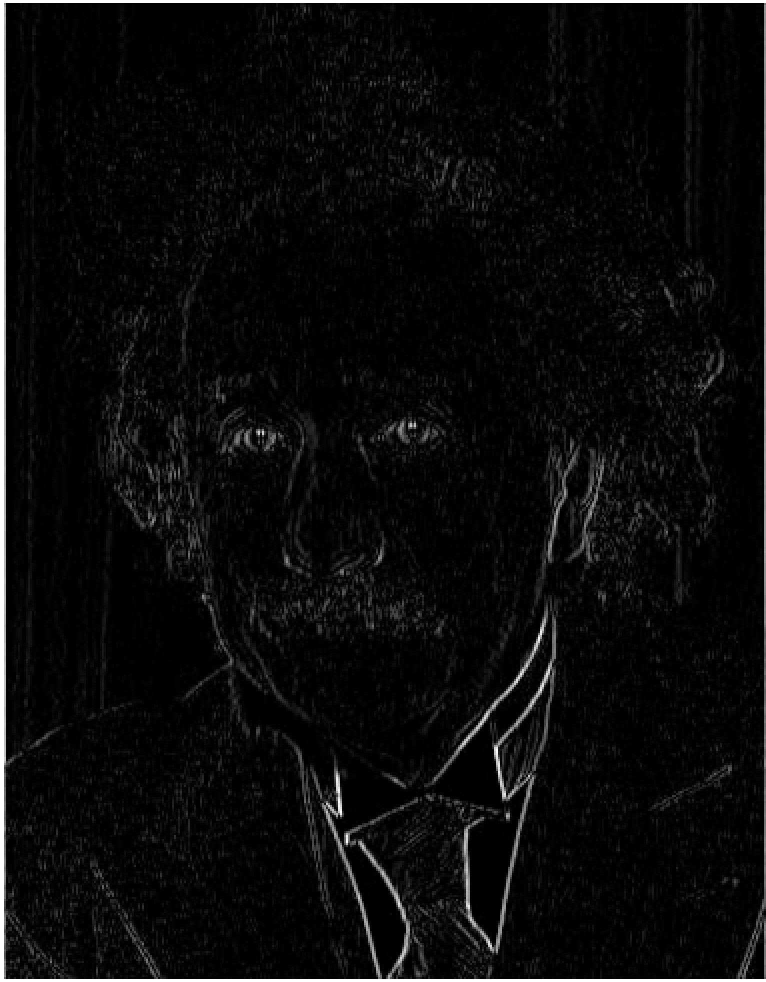
1	0	-1
2	0	-2
1	0	-1

Other filters



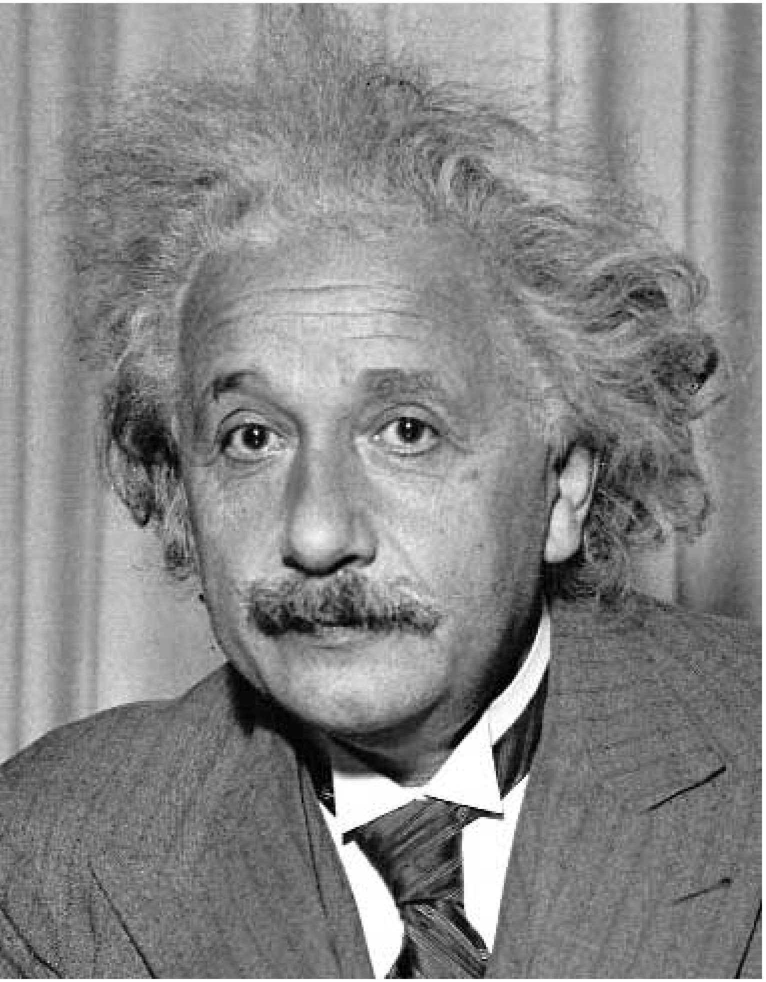
1	0	-1
2	0	-2
1	0	-1

Sobel



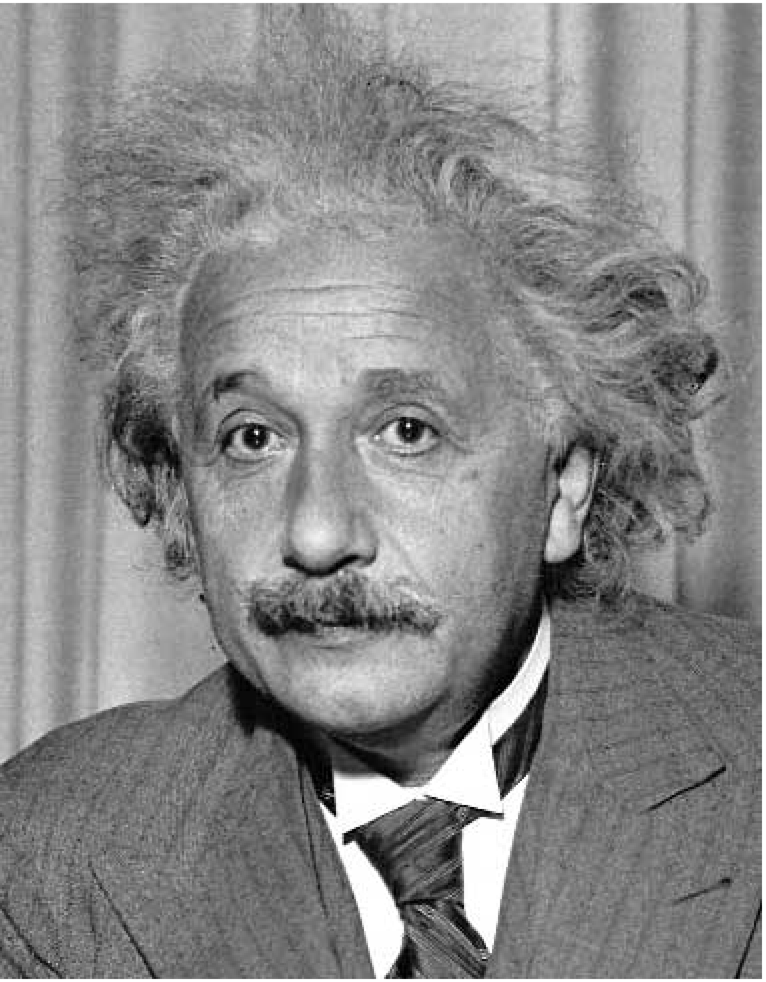
Vertical Edge
(absolute value)

Other filters



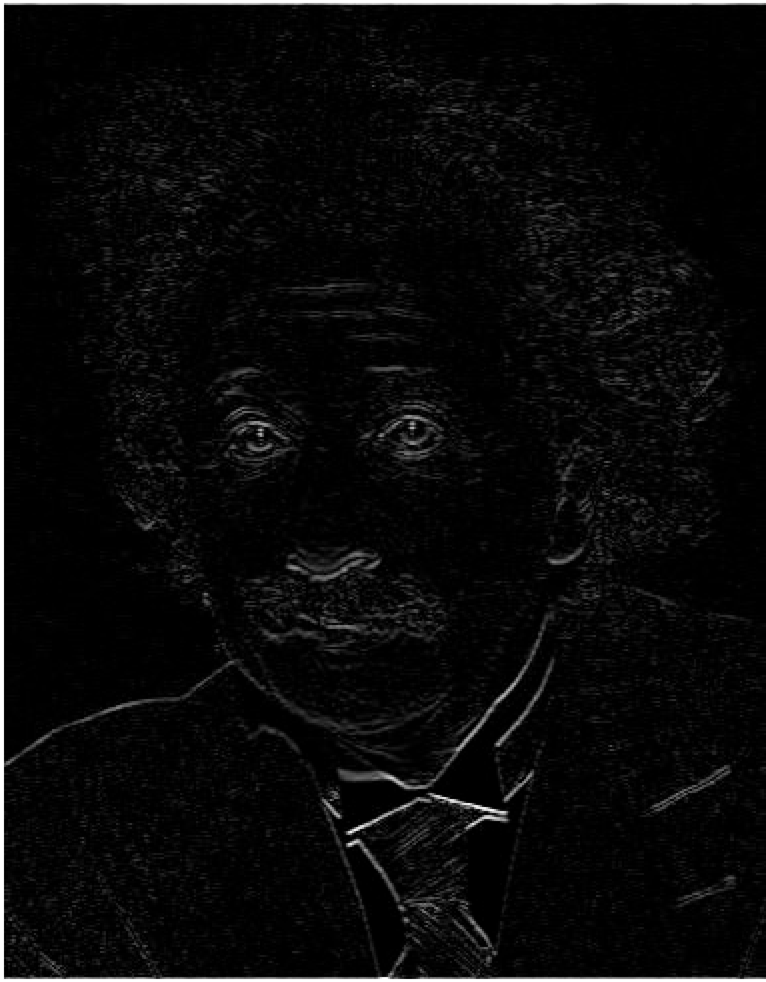
1	2	1
0	0	0
-1	-2	-1

Other filters



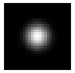
1	2	1
0	0	0
-1	-2	-1

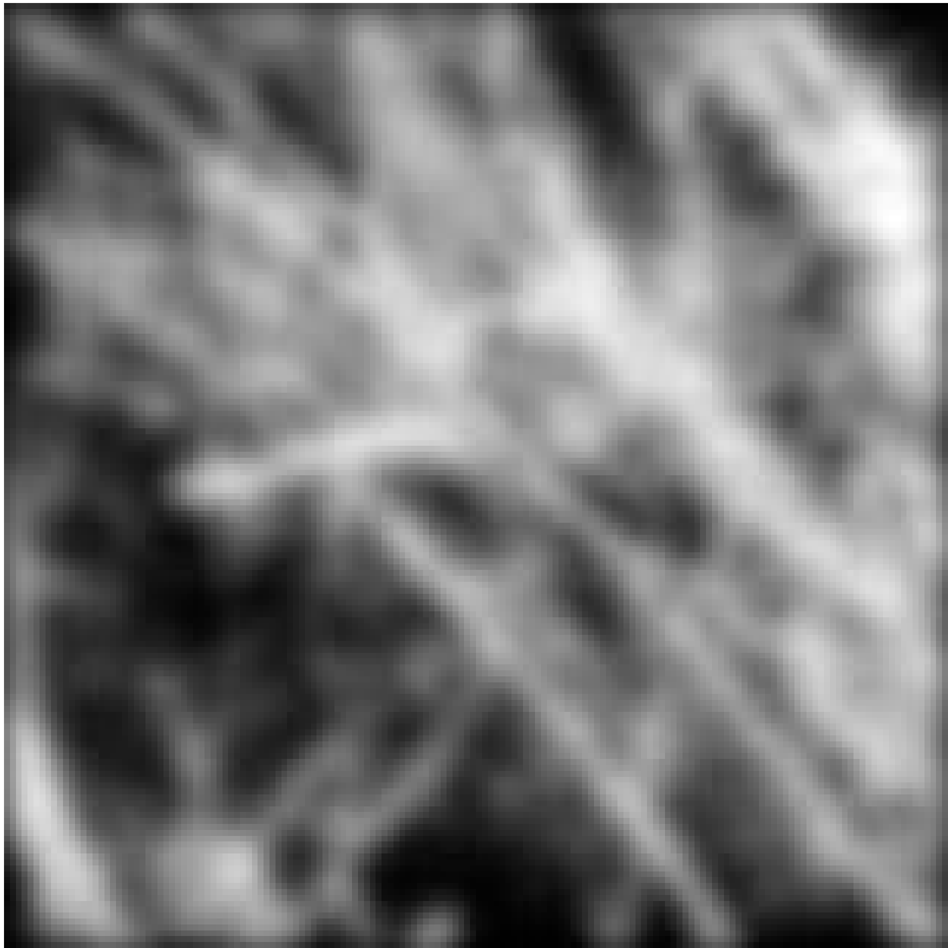
Sobel



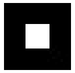
Horizontal Edge
(absolute value)

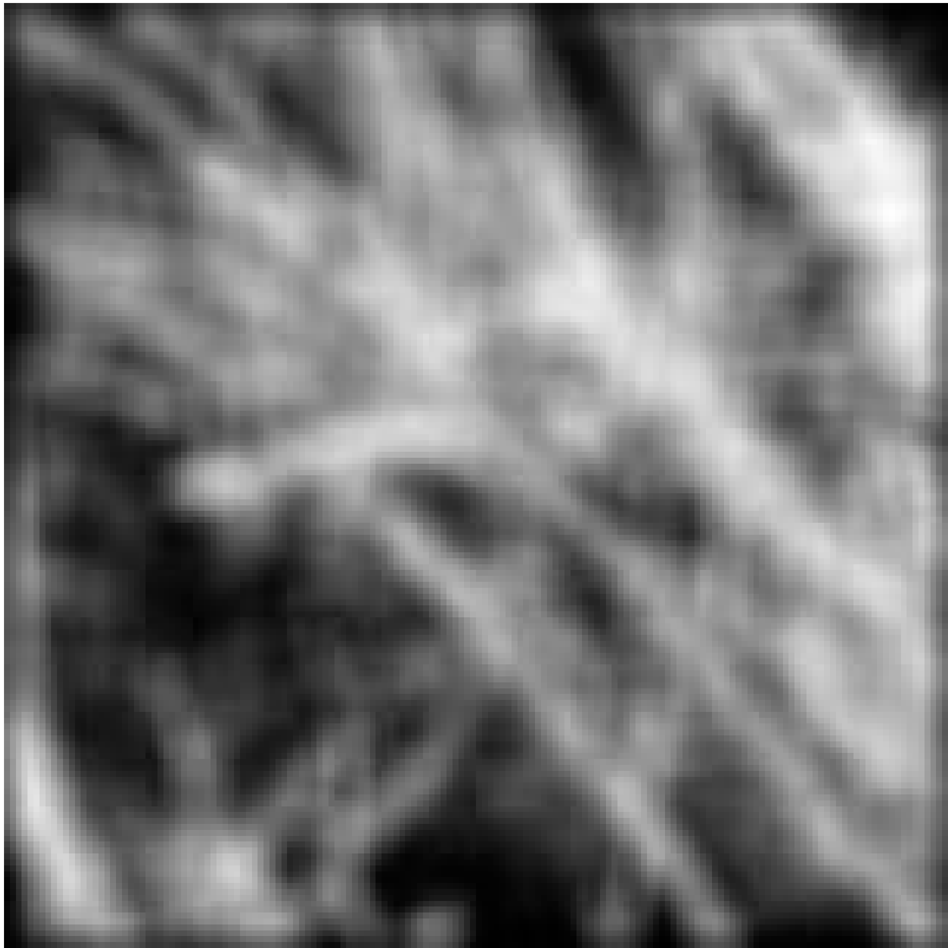
Smoothing with Gaussian filter





Smoothing with box filter





Key properties of linear filters

Source: S. Lazebnik

Key properties of linear filters

Linearity:

$$\mathtt{filter}(f_1 + f_2) = \mathtt{filter}(f_1) + \mathtt{filter}(f_2)$$

Source: S. Lazebnik

Key properties of linear filters

Linearity:

`filter(f`$_1$` + f`$_2$`) = filter(f`$_1$`) + filter(f`$_2$`)`

Shift invariance: same behavior regardless of pixel location

`filter(shift(f)) = shift(filter(f))`

Source: S. Lazebnik

Key properties of linear filters

Linearity:

`filter(`f_1` + `f_2`) = filter(`f_1`) + filter(`f_2`)`

Shift invariance: same behavior regardless of pixel location

`filter(shift(f)) = shift(filter(f))`

Any linear, shift-invariant operator can be represented as a convolution

Source: S. Lazebnik

More properties

Source: S. Lazebnik

More properties

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
  - But particular filtering implementations might break this equality

Source: S. Lazebnik

More properties

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
  - But particular filtering implementations might break this equality

- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: a * $(b_1 * b_2 * b_3)$

Source: S. Lazebnik

More properties

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
  - But particular filtering implementations might break this equality

- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: a * $(b_1 * b_2 * b_3)$

- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$

Source: S. Lazebnik

More properties

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
  - But particular filtering implementations might break this equality

- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: a * $(b_1 * b_2 * b_3)$

- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$

- Scalars factor out: $ka * b = a * kb = k\,(a * b)$

Source: S. Lazebnik

More properties

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
  - But particular filtering implementations might break this equality
- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: a * $(b_1 * b_2 * b_3)$
- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$
- Scalars factor out: $ka * b = a * kb = k (a * b)$
- Identity: unit impulse e = [0, 0, 1, 0, 0],
  $a * e = a$

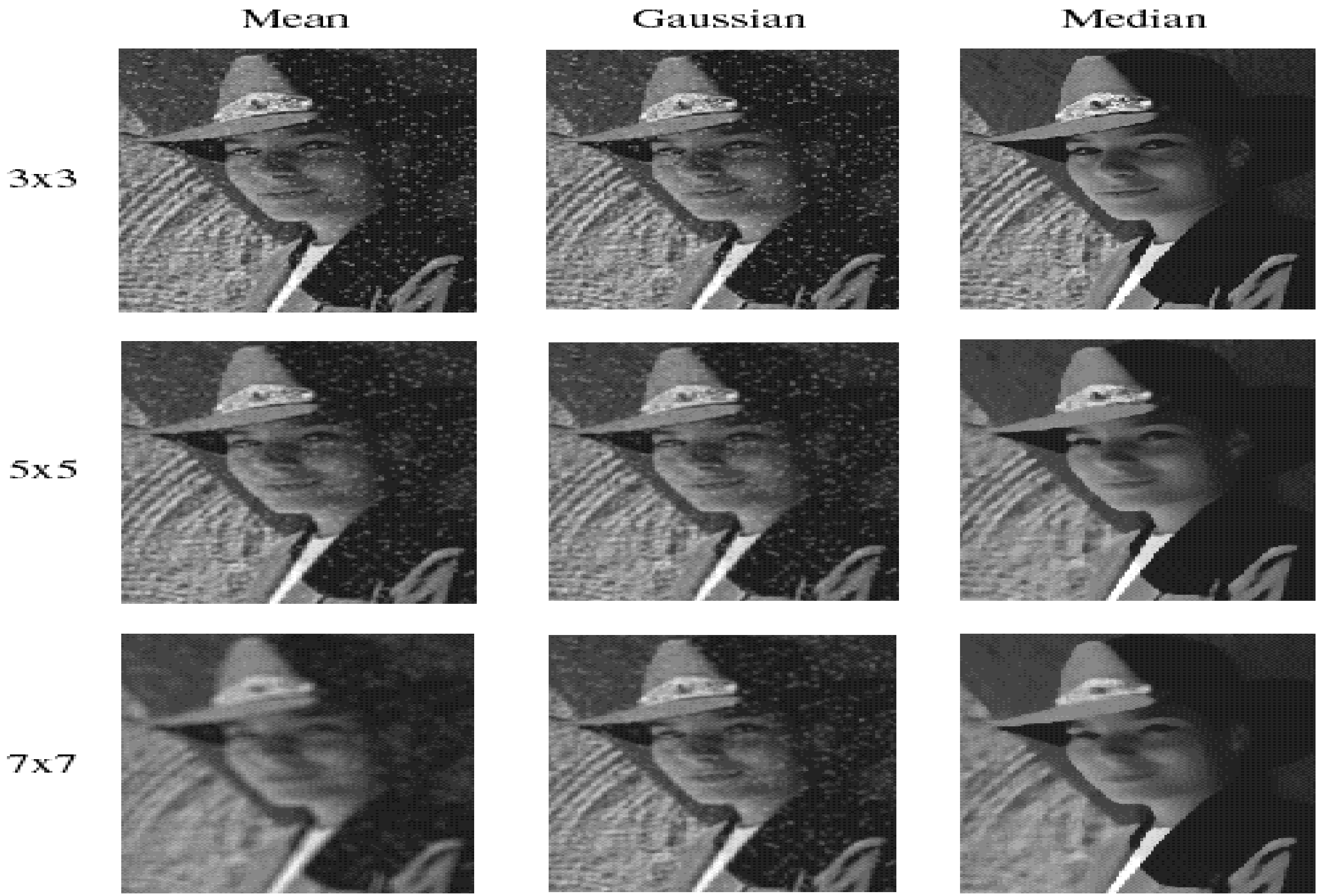
Source: S. Lazebnik

Median filters

- A **Median Filter** operates over a window by selecting the median intensity in the window.
- What advantage does a median filter have over a mean filter?
- Is a median filter a kind of convolution?

Slide by Steve Seitz

Comparison: salt and pepper noise



Slide by Steve Seitz

MATLAB Functions

MATLAB Functions

- conv: 1-D Convolution.

  C = conv(A, B) convolves vectors A and B.

MATLAB Functions

- conv: 1-D Convolution.

  C = conv(A, B) convolves vectors A and B.

- conv2: Two dimensional convolution.
  - C = conv2(A, B) performs the 2-D convolution of matrices A and B.

MATLAB Functions

MATLAB Functions

- filter2: Two-dimensional digital filter.

MATLAB Functions

- filter2:   Two-dimensional digital filter.
  - *Y = filter2(B,X)* filters the data in X with the 2-D filter in the matrix B.

MATLAB Functions

- filter2: Two-dimensional digital filter.
  - *Y = filter2(B,X)* filters the data in X with the 2-D filter in the matrix B.
  - The result, Y, is computed using 2-D correlation and is the same size as X.

MATLAB Functions

- filter2: Two-dimensional digital filter.
  - *Y = filter2(B,X)* filters the data in X with the 2-D filter in the matrix B.
  - The result, Y, is computed using 2-D correlation and is the same size as X.
  - filter2 uses CONV2 to do most of the work. 2-D correlation is related to 2-D convolution by a 180 degree rotation of the filter matrix.

MATLAB Functions

MATLAB Functions

- gradient: Approximate gradient.

MATLAB Functions

- gradient: Approximate gradient.
  - *[FX,FY] = gradient(F)* returns the numerical gradient of the matrix F. FX corresponds to dF/dx, FY corresponds to dF/dy.

MATLAB Functions

- gradient: Approximate gradient.
  - *[FX,FY] = gradient(F)* returns the numerical gradient of the matrix F. FX corresponds to dF/dx, FY corresponds to dF/dy.
- mean: Average or mean value.

MATLAB Functions

- gradient: Approximate gradient.
  - *[FX,FY] = gradient(F)* returns the numerical gradient of the matrix F. FX corresponds to dF/dx, FY corresponds to dF/dy.
- mean: Average or mean value.
  - For vectors, *mean(X)* is the mean value (average) of the elements in X.

MATLAB Functions

MATLAB Functions

- fspecial: Creates predefined 2-D filter

MATLAB Functions

- fspecial: Creates predefined 2-D filter
  - *H = fspecial(TYPE)* creates a two-dimensional filter H of the specified type. Possible values for TYPE are:

MATLAB Functions

- fspecial: Creates predefined 2-D filter
  - *H = fspecial(TYPE)* creates a two-dimensional filter H of the specified type. Possible values for TYPE are:
  - 'average'   averaging filter;

MATLAB Functions

- fspecial: Creates predefined 2-D filter
  - *H = fspecial(TYPE)* creates a two-dimensional filter H of the specified type. Possible values for TYPE are:
  - 'average' averaging filter;
  - 'gaussian' Gaussian lowpass filter

MATLAB Functions

- fspecial: Creates predefined 2-D filter
  - *H = fspecial(TYPE)* creates a two-dimensional filter H of the specified type. Possible values for TYPE are:
  - 'average'   averaging filter;
  - 'gaussian' Gaussian lowpass filter
  - 'laplacian' filter approximating the 2-D Laplacian operator

MATLAB Functions

- fspecial: Creates predefined 2-D filter
  - *H = fspecial(TYPE)* creates a two-dimensional filter H of the specified type. Possible values for TYPE are:
  - 'average' averaging filter;
  - 'gaussian' Gaussian lowpass filter
  - 'laplacian' filter approximating the 2-D Laplacian operator
  - 'log' Laplacian of Gaussian filter

MATLAB Functions

- fspecial: Creates predefined 2-D filter
  - *H = fspecial(TYPE)* creates a two-dimensional filter H of the specified type. Possible values for TYPE are:
  - 'average' averaging filter;
  - 'gaussian' Gaussian lowpass filter
  - 'laplacian' filter approximating the 2-D Laplacian operator
  - 'log' Laplacian of Gaussian filter
  - 'prewitt' Prewitt horizontal edge-emphasizing filter

MATLAB Functions

- fspecial: Creates predefined 2-D filter
  - *H = fspecial(TYPE)* creates a two-dimensional filter H of the specified type. Possible values for TYPE are:
  - 'average' averaging filter;
  - 'gaussian' Gaussian lowpass filter
  - 'laplacian' filter approximating the 2-D Laplacian operator
  - 'log' Laplacian of Gaussian filter
  - 'prewitt' Prewitt horizontal edge-emphasizing filter
  - 'sobel' Sobel horizontal edge-emphasizing filter

MATLAB Functions

- fspecial: Creates predefined 2-D filter
  - *H = fspecial(TYPE)* creates a two-dimensional filter H of the specified type. Possible values for TYPE are:
  - 'average' averaging filter;
  - 'gaussian' Gaussian lowpass filter
  - 'laplacian' filter approximating the 2-D Laplacian operator
  - 'log' Laplacian of Gaussian filter
  - 'prewitt' Prewitt horizontal edge-emphasizing filter
  - 'sobel' Sobel horizontal edge-emphasizing filter

  Example: H=fspecial('gaussian',7,1) creates a 7x7 Gaussian filter with variance 1.

Some practical matters

Practical matters

How big should the filter be?

- Values at edges should be near zero
- Rule of thumb for Gaussian: set filter half-width to about 3 σ

Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

Practical matters

- methods (MATLAB):
  - clip filter (black): imfilter(f, g, 0)
  - wrap around: imfilter(f, g, 'circular')
  - copy edge: imfilter(f, g, 'replicate')
  - reflect across edge: imfilter(f, g, 'symmetric')

Source: S. Marschner

Q?

Practical matters

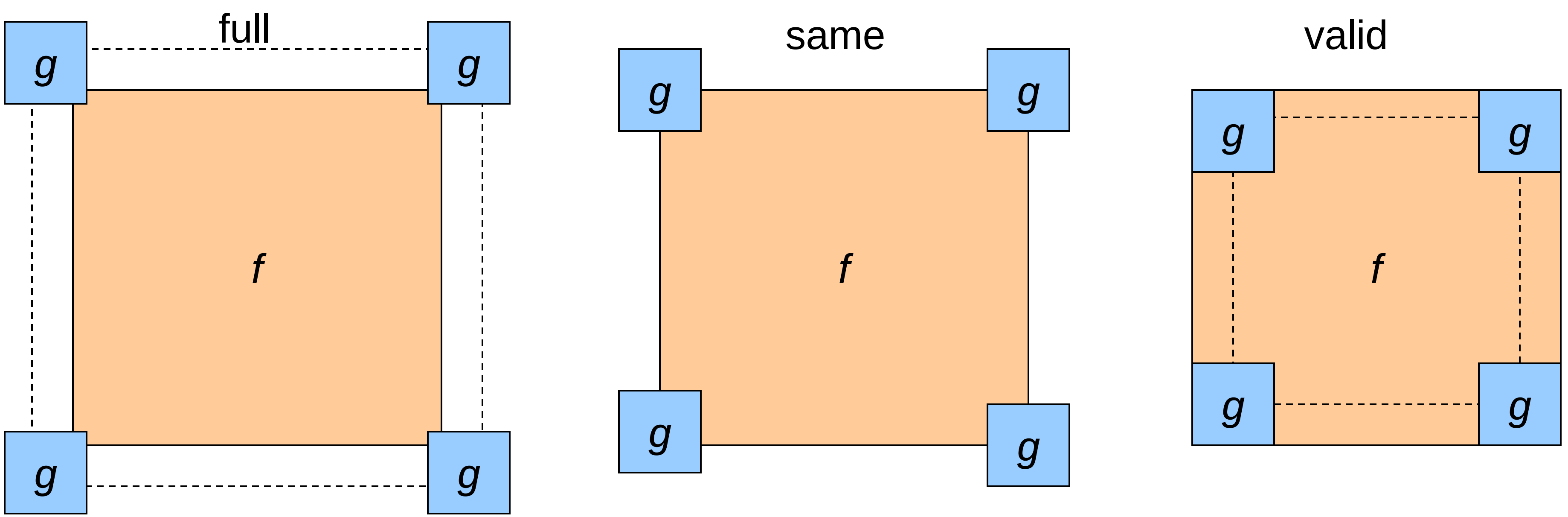
– methods (MATLAB):

- clip filter (black): imfilter(f, g, 0)
- wrap around: imfilter(f, g, ‘circular’)
- copy edge: imfilter(f, g, ‘replicate’)
- reflect across edge: imfilter(f, g, ‘symmetric’)

Source: S. Marschner

Practical matters

- What is the size of the output?
- MATLAB: filter2(g, f, *shape*)
  - *shape* = 'full': output size is sum of sizes of f and g
  - *shape* = 'same': output size is same as f
  - *shape* = 'valid': output size is difference of sizes of f and g



Source: S. Lazebnik

Reading Material

Reading Material

- Mubarak Shah, "Fundamentals of Computer Vision".
  - Chapter, 2

Reading Material

- Mubarak Shah, "Fundamentals of Computer Vision".
  - Chapter, 2
- Richard Szeliski, "Computer Vision: Algorithms and Applications".
  - Section 3.1 and 3.2