# CAP 5415 Computer Vision
# Fall 2011

Dr. Mubarak Shah
Univ. of Central Florida
www.cs.ucf.edu/~vision/courses/cap5415/fall2012
Office 247-F HEC

# Filtering

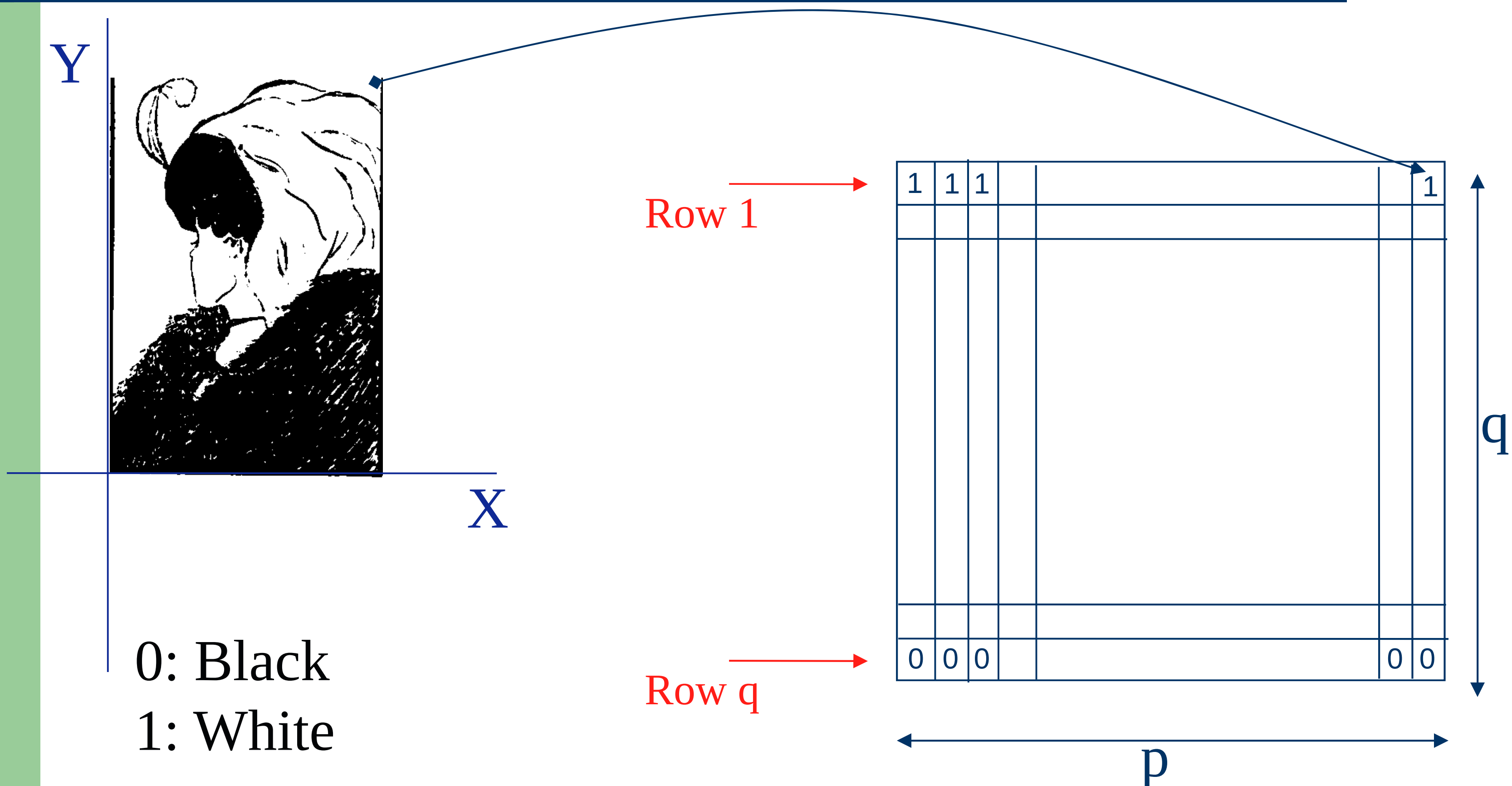
Lecture-2

# General

- Binary
- Gray Scale
- Color

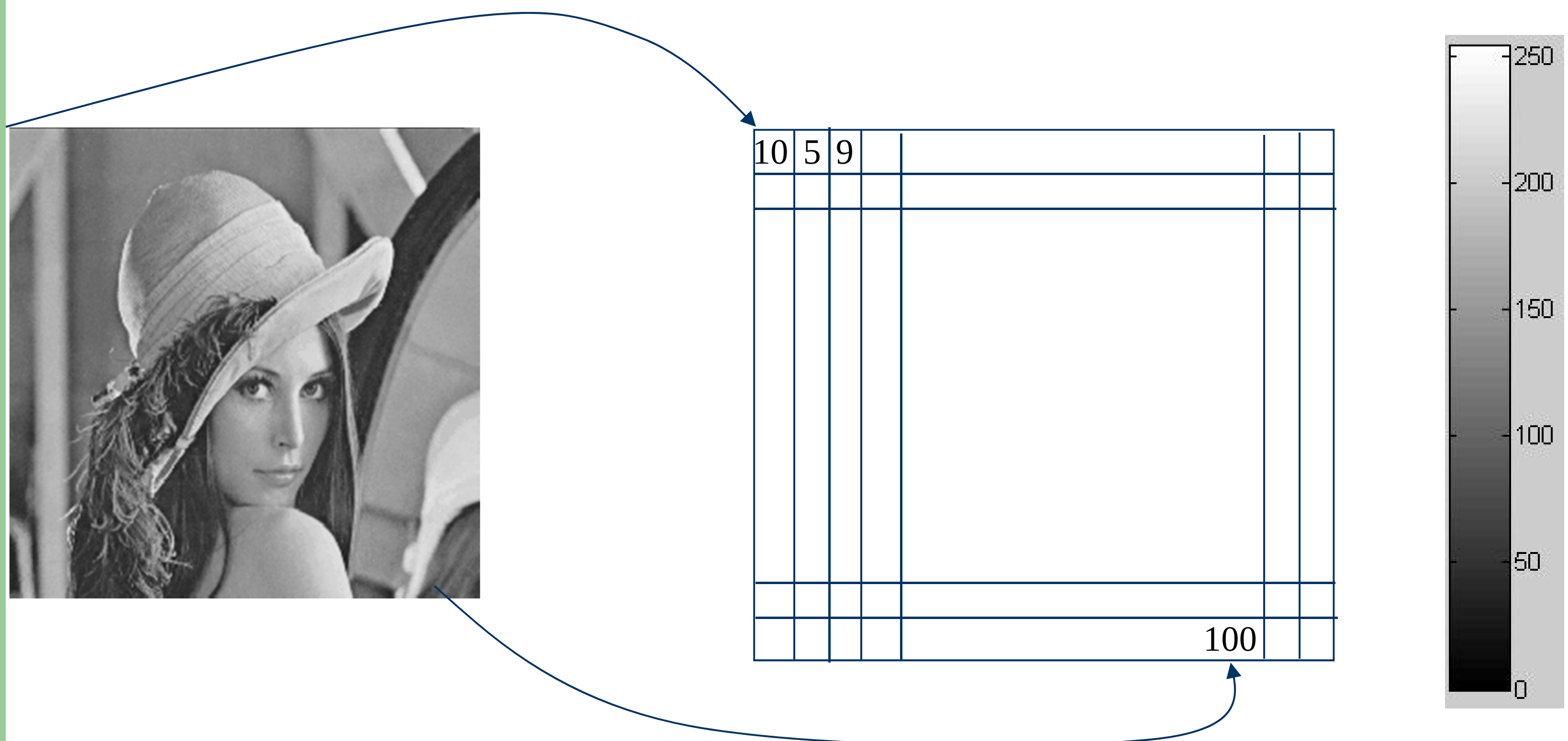


# Binary Images

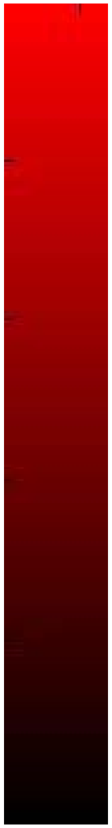



0: Black
1: White

# Gray Level Image

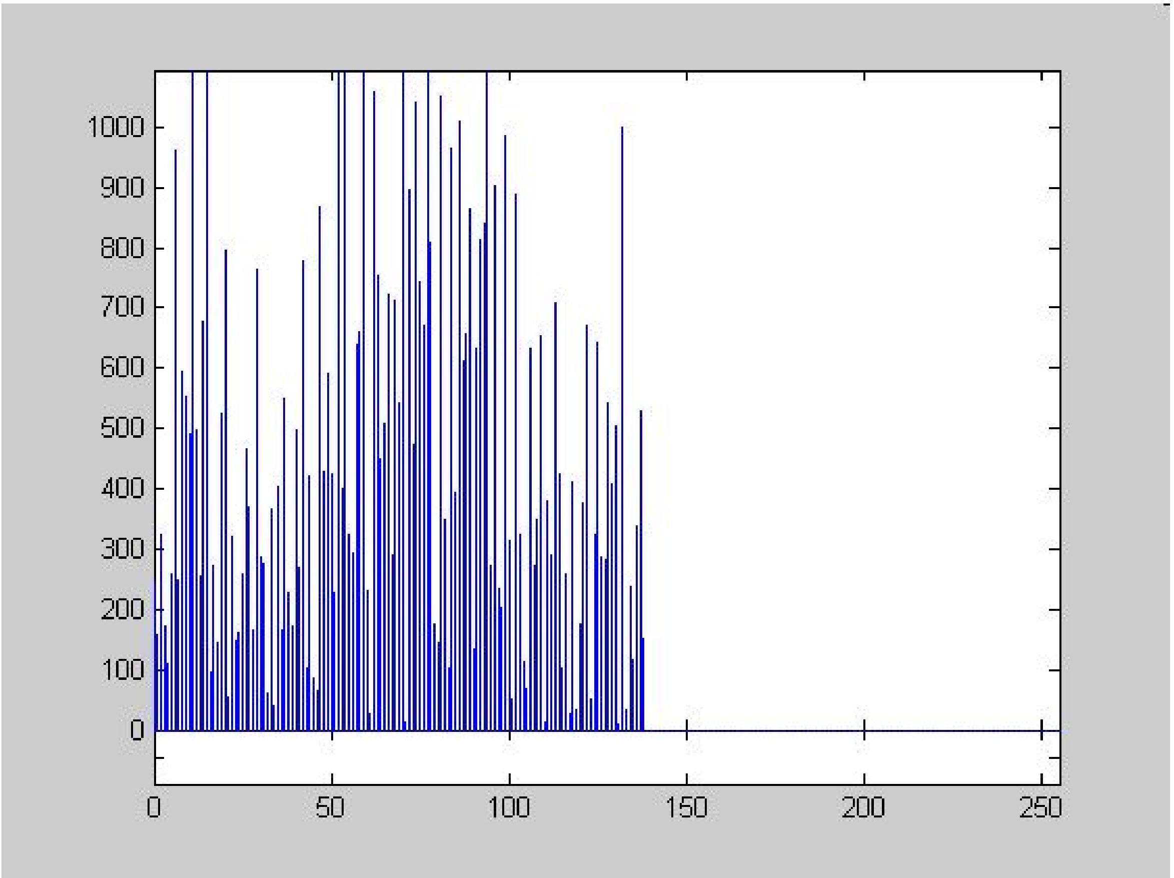


# Gray Scale Image



# Color Image
# Red, Green, Blue Channels


# Image Histogram



# Image Noise

- Light Variations
- Camera Electronics
- Surface Reflectance
- Lens

# Image Noise

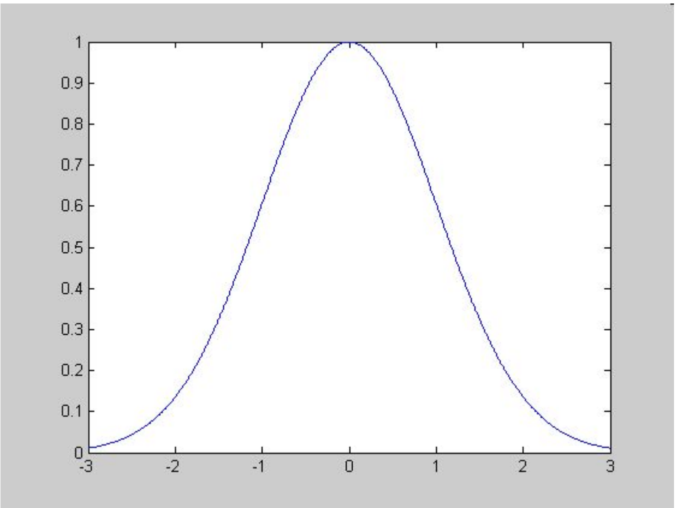
- $I(x,y)$ : the true pixel values
- $n(x,y)$ : the noise at pixel $(x,y)$

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

➔

# Gaussian Noise

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



# Image Derivatives & Averages

# 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

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

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

# Examples

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

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

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

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

# Discrete Derivative

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

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

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

# Discrete Derivative
# Finite Difference

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

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

$$
\begin{array}{lllllllll}
f(x) = & 10 & 15 & 10 & 10 & 25 & 20 & 20 & 20 \\
f'(x) = & 0 & 5 & -5 & 0 & 15 & -5 & 0 & 0 \\
f''(x) = & 0 & 5 & -10 & 5 & 15 & 20 & 5 & 0
\end{array}
$$

## Derivative Masks

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

# Derivatives in 2 Dimensions

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

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

# 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

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

# Correlation

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

$f$ = Image

$f$ = Kernel

$f$

| | | |
|---|---|---|
| $f_1$ | $f_2$ | $f_3$ |
| $f_4$ | $f_5$ | $f_6$ |
| $f_7$ | $f_8$ | $f_9$ |

$\otimes$

$h$

| | | |
|---|---|---|
| $h_1$ | $h_2$ | $h_3$ |
| $h_4$ | $h_5$ | $h_6$ |
| $h_7$ | $h_8$ | $h_9$ |

$$\begin{aligned} f * h = {} & f_1h_1 + f_2h_2 + f_3h_3 \\ & + f_4h_4 + f_5h_5 + f_6h_6 \\ & + f_7h_7 + f_8h_8 + f_9h_9 \end{aligned}$$

# Convolution

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

$f$ = Image

$h$ = Kernel

$h$

| | | |
|---|---|---|
| $h_1$ | $h_2$ | $h_3$ |
| $h_4$ | $h_5$ | $h_6$ |
| $h_7$ | $h_8$ | $h_9$ |

$X - flip$

| | | |
|---|---|---|
| $h_7$ | $h_8$ | $h_9$ |
| $h_4$ | $h_5$ | $h_6$ |
| $h_1$ | $h_2$ | $h_3$ |

$Y - flip$

| | | |
|---|---|---|
| $h_9$ | $h_8$ | $h_7$ |
| $h_6$ | $h_5$ | $h_4$ |
| $h_3$ | $h_2$ | $h_1$ |

$f$

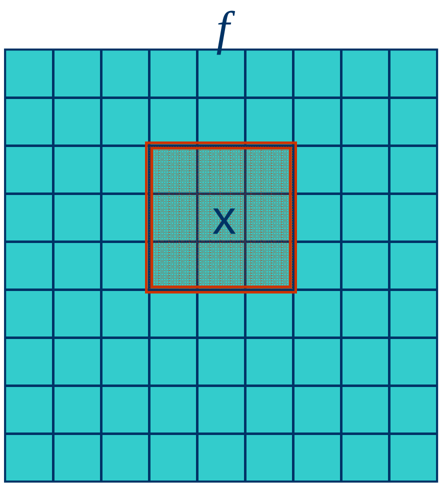
| | | |
|---|---|---|
| $f_1$ | $f_2$ | $f_3$ |
| $f_4$ | $f_5$ | $f_6$ |
| $f_7$ | $f_8$ | $f_9$ |

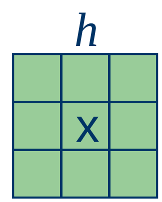
$\otimes$

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

# Convolution

$$
\begin{aligned}
f(x,y)*h = \quad & f(x+1,y+1)h(-1,-1) + f(x,y+1)h(0,-1) + f(x-1,y+1)h(1,-1) + \\
& f(x+1,y)h(-1,0) + f(x,y)h(0,0) + f(x-1,y)h(1,0) \\
& f(x+1,y-1)h(-1,1) + f(x,y-1)h(0,1) + f(x-1,y-1)h(1,1)
\end{aligned}
$$




$$f * h = \sum_{i=-1}^{1} \sum_{j=-1}^{1} f(x-i, y-i)h(i,j)$$

Coordinates

| **-1,0** | **0,1** | **1,1** |
|---|---|---|
| -1,0 | 0,0 | 1,0 |
| -1,-1 | 0,-1 | 1,-1 |

# Averages

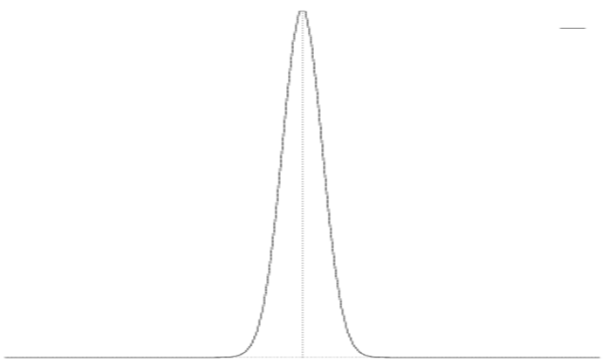
- Mean

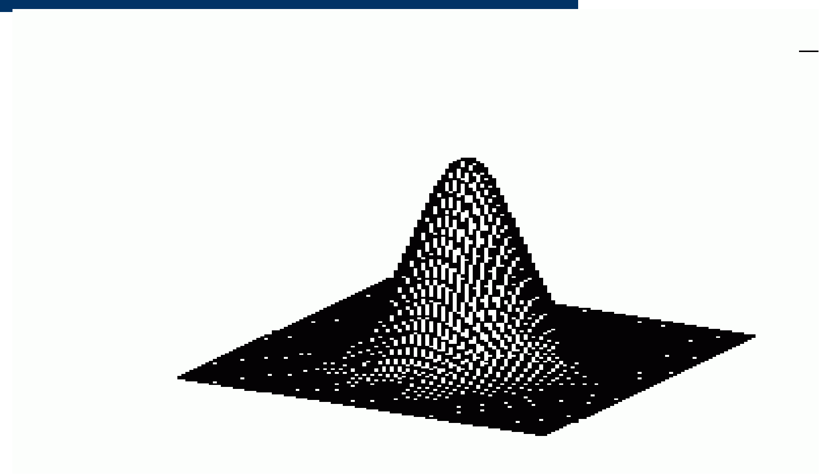
$$I = \frac{I_1 + I_2 + \ldots I_n}{n} = \frac{\sum_{i=1}^{n} I_i}{n}$$

- 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





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

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

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

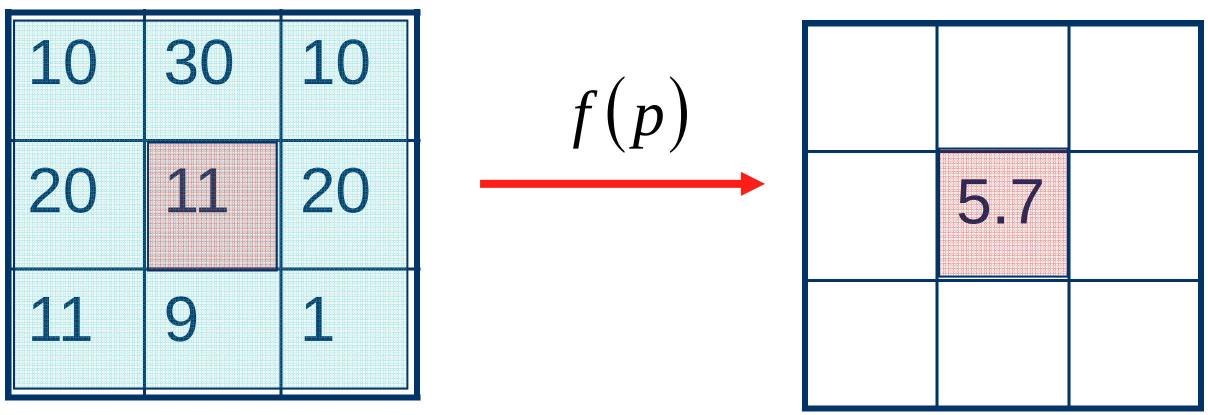
$$\sigma = 1$$

# Properties of Gaussian

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

| | | |
|---|---|---|
| 1 | 3 | 0 |
| 2 | 10 | 2 |
| 4 | 1 | 1 |

Image

$\otimes$

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0.1 | -1 |
| 1 | 0 | -1 |

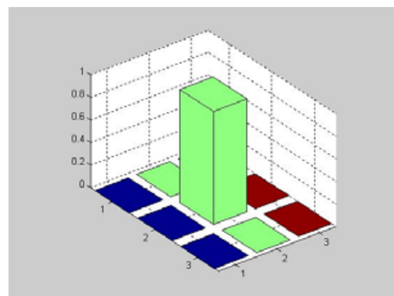
Kernel

$=$

| | | |
|---|---|---|
| | | |
| | 5 | |
| | | |

Filter Output

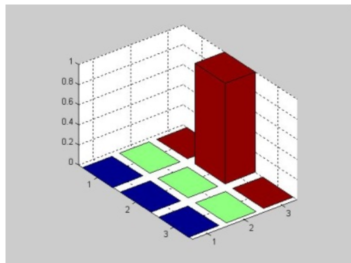
# Filtering Examples



$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} =$$

# Filtering Examples

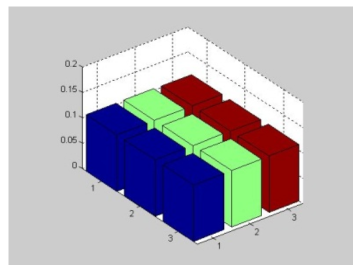
*



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

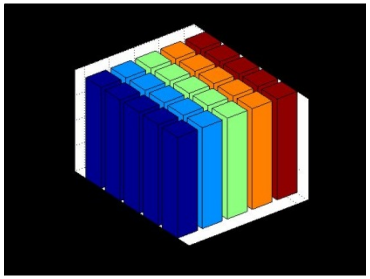
=

# Filtering Examples



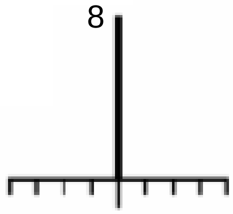

$$* \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

# Filtering Examples

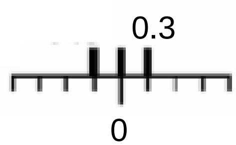


$$* \frac{1}{25} \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline \end{array} =$$

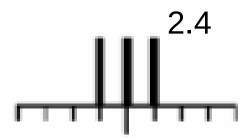
# Blurring Examples



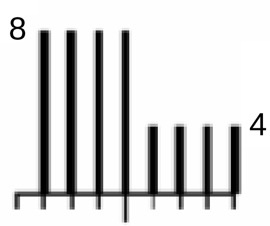

original



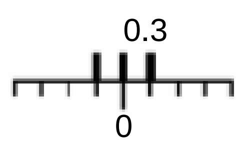

pixel offset



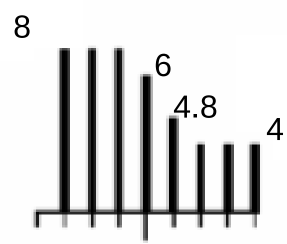

filtered




original



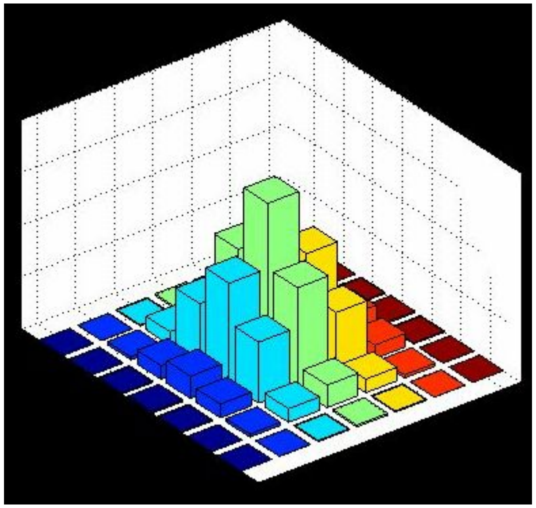

pixel offset




filtered

# Filtering Gaussian

$*$



$=$

# Gaussian vs. Smoothing

Gaussian Smoothing

Smoothing by Averaging

# Noise Filtering

Gaussian Noise

After Averaging

After Gaussian Smoothing

# 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

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

- 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

- special: Create predefined 2-D filters
  - *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.