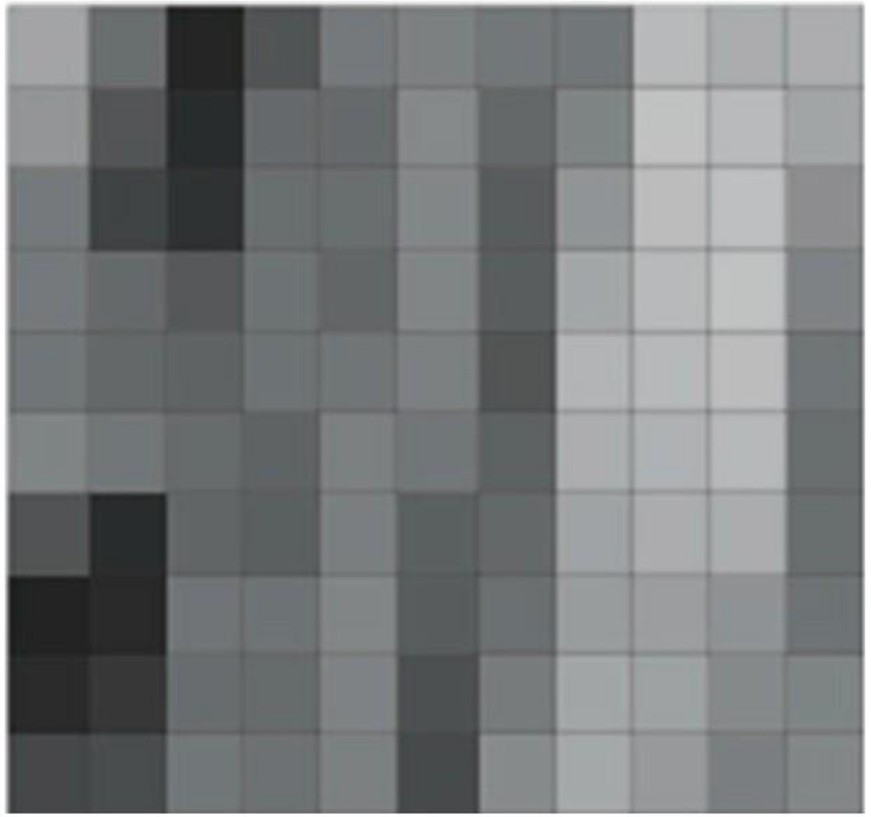
greatlearning
Learning for Life

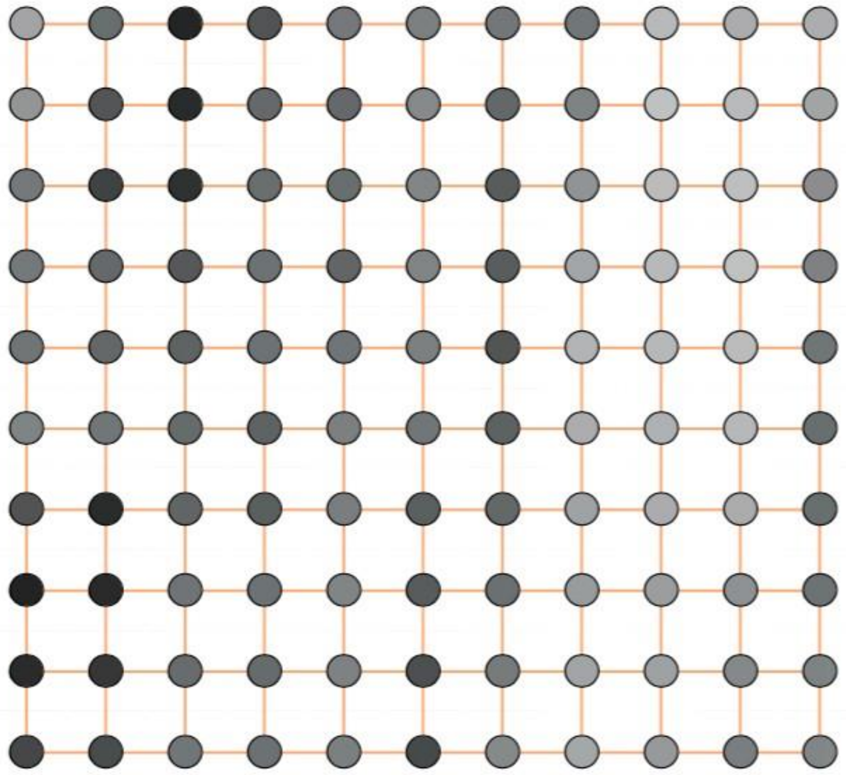
# Deep Learning (for Computer Vision)

Arjun Jain

# Computer Vision: Working with Images

## Picture Elements - PIXEL





PIXELS are ATOMIC ELEMENTS of an image.

In late 1960s, terminology ‘pixel’ was introduced by a group of scientists at JPL in California!

## Image Types: Scalar and Binary

- A scalar image has $2^a - 1$ integer values

$$u \, 2\{0,\ 1,\ \ldots,\ 2^a — 1\}$$

- a: level (bit)

- **Ex.** If 8 bit (a=8), image spans from 0 to 255
  - **0 black**
  - **255 white**
- **Ex.** If 1 bit (a=1), it is binary image, 0 and 1 only

## Image Type: RGB (red, green, blue)

Image has three channels (bands), each channel spans a-bit values

## Image format

- Some formats: TIF, PGM, PBM, GIF, JPEG, PNG,  RAW etc.
- Medical Images: DICOM, Analyze, NIFTI etc.

- **HEADER:** contains image information, image size, pixel size, …
- **DATA:** integer, double, float, unsigned integer, char,…

# Practice: Image Format/Read/Show

```
from scipy import misc
l = misc.lena()
misc.imsave( 'lena.png', 1) #uses the image module (PIL)

import matplotlib.pyplot as plt
plt.imshow(l)
plt.show()
```

**PIL:** Python Imaging Library

from PIL import Image

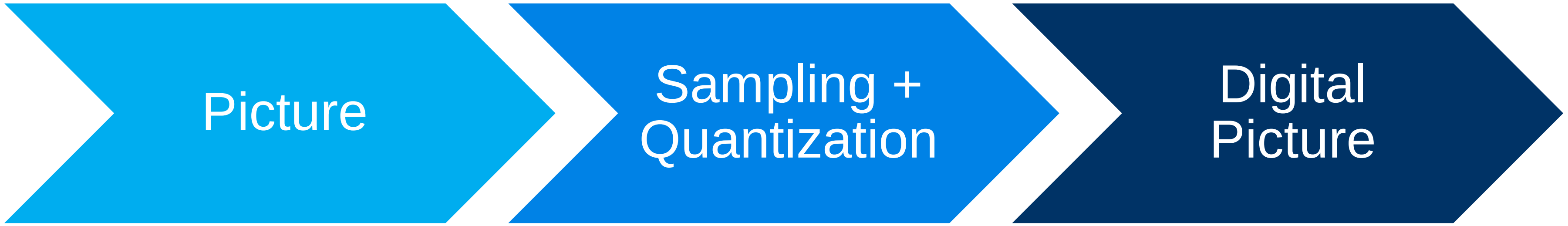
Img = Image.open(‘empire.jpg’)

Matplotlib is a good graphics library with much

More powerful features than the

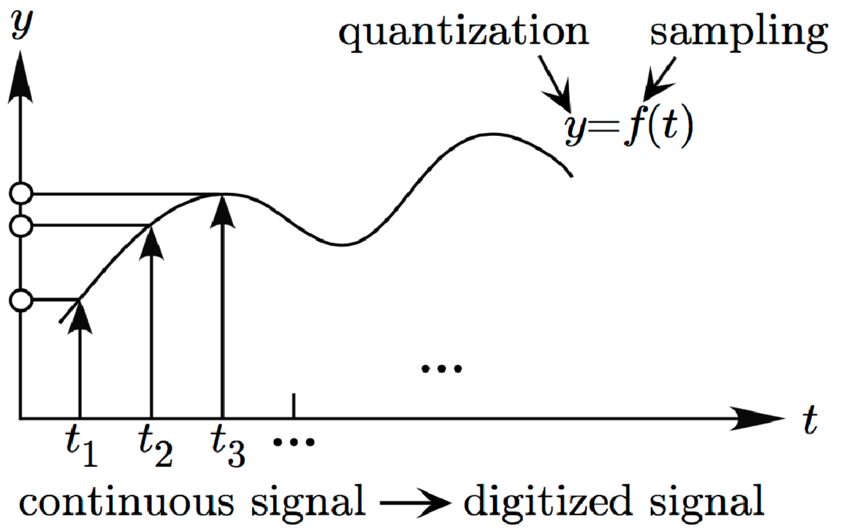
## Pictures

- Computers use discrete form of the pictures
- The process transforming ***continuous space*** into ***discrete space*** is called **digitization**

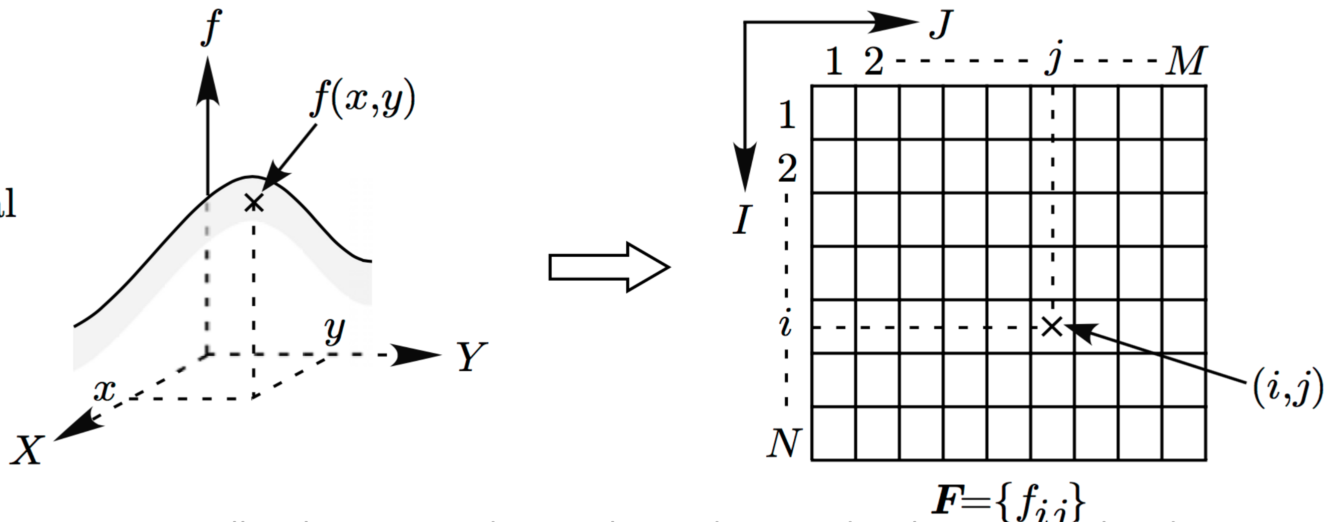


## Pictures

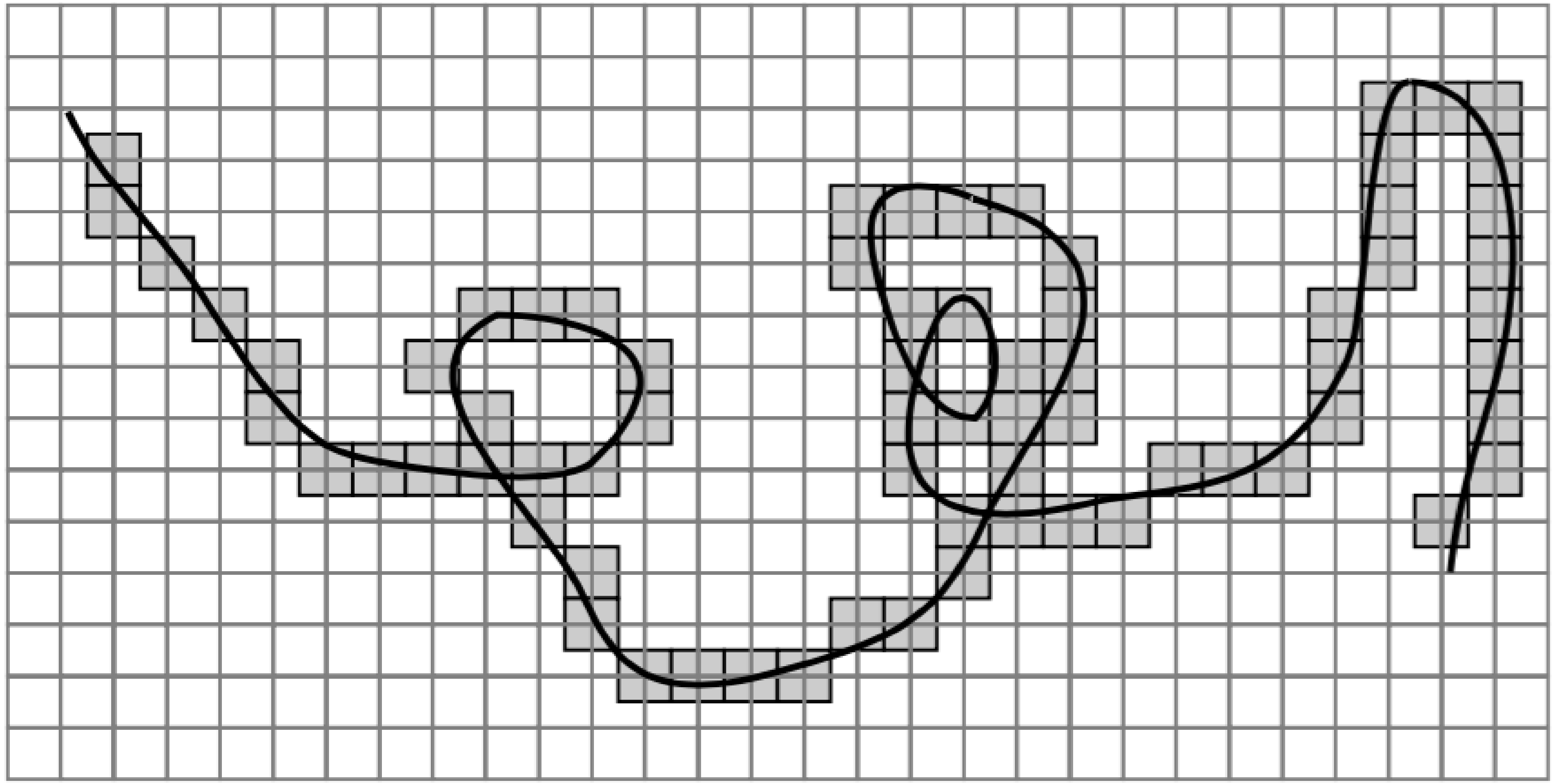
one-dimensional



two-dimensional



# Digitization of an arc

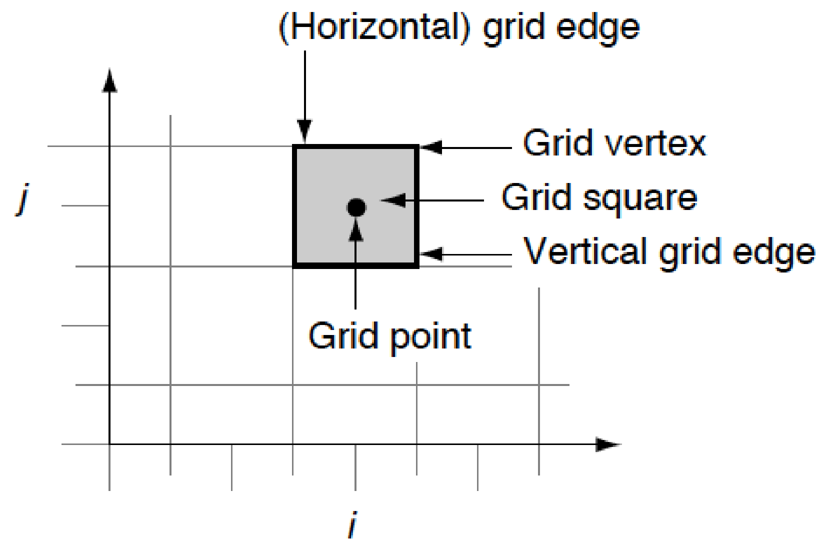


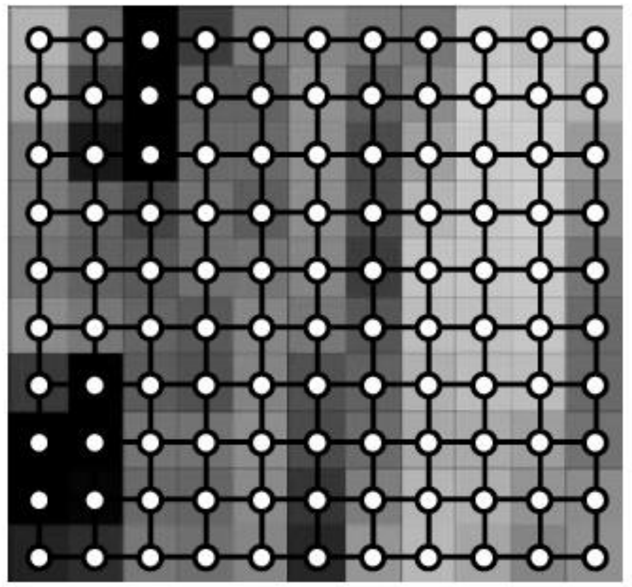
## Definition

- A (2D) picture P is a function defined on a  (finite) rectangular subset G of a regular planar orthogonal array. G is called (2D) grid,  and an element of G is called pixel. P assigns a  value of P(p) to each point

## Definition

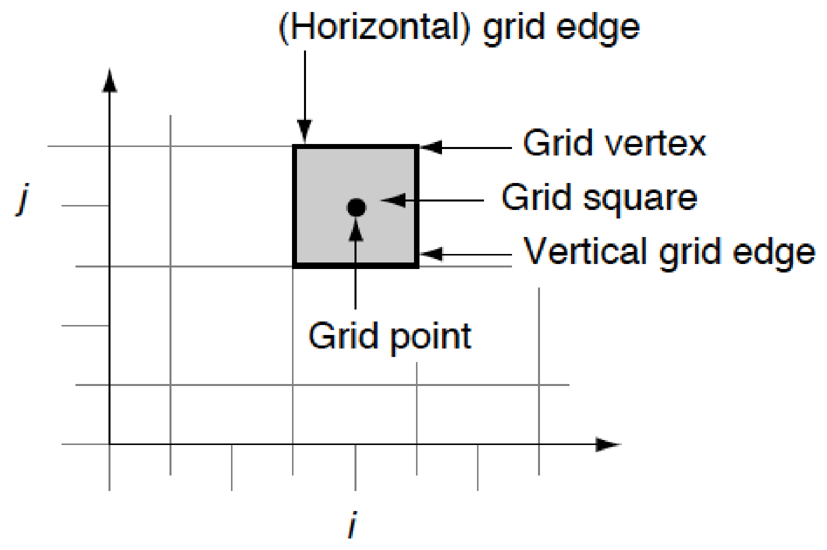
- A (2D) picture P is a function defined on a (finite) rectangular subset G of a regular planar orthogonal array. G is called (2D) grid, and an element of G is called pixel. P assigns a value of P(p) to each point

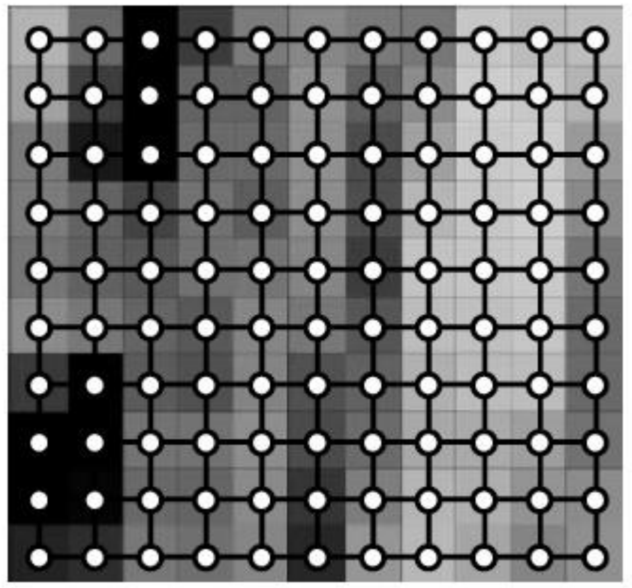




# Definition

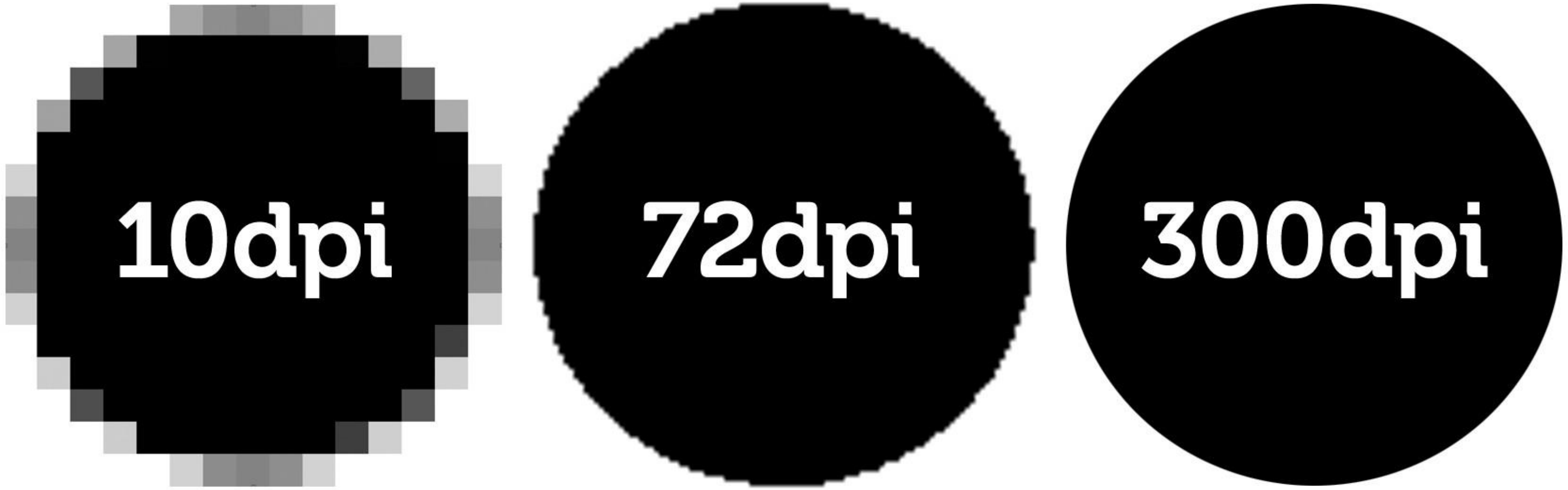
- Pictures are not only sampled, they are also quantized: they may have only a finite number of possible values (i.e., 0 to 255, 0-1, …)





## Resolution

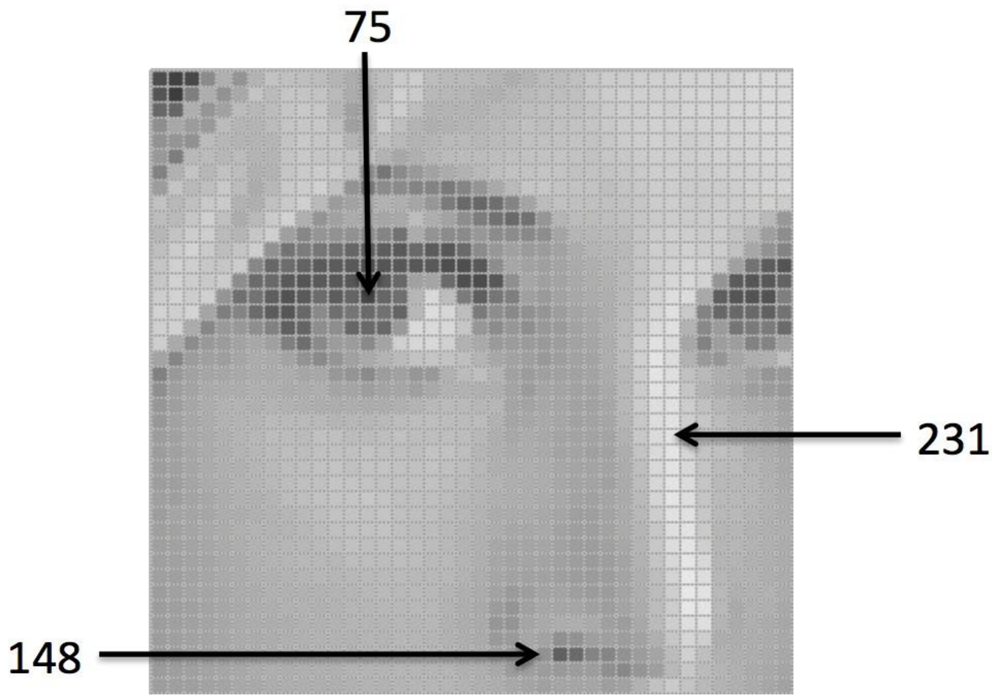
- Resolution is a display parameter, defined in dots per inch (DPI) or equivalent measures of spatial pixel density, and its standard value for recent screen technologies is 72 dpi. Recent printer resolutions are in 300 dpi and/or 600 dpi.



## Filtering

- An image contains discrete number of pixels
  - A simple example
  - Pixel value:
    - “grayscale”

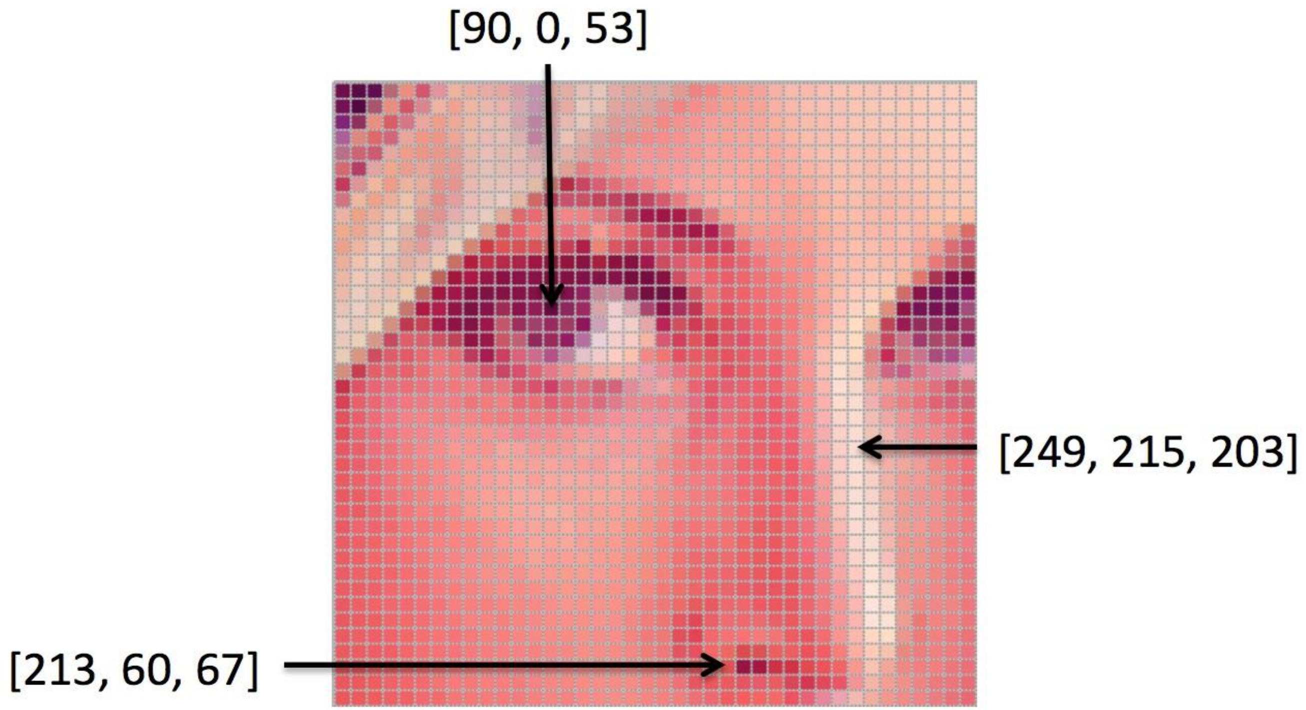
    (or “intensity”): [0,255]



## Filtering

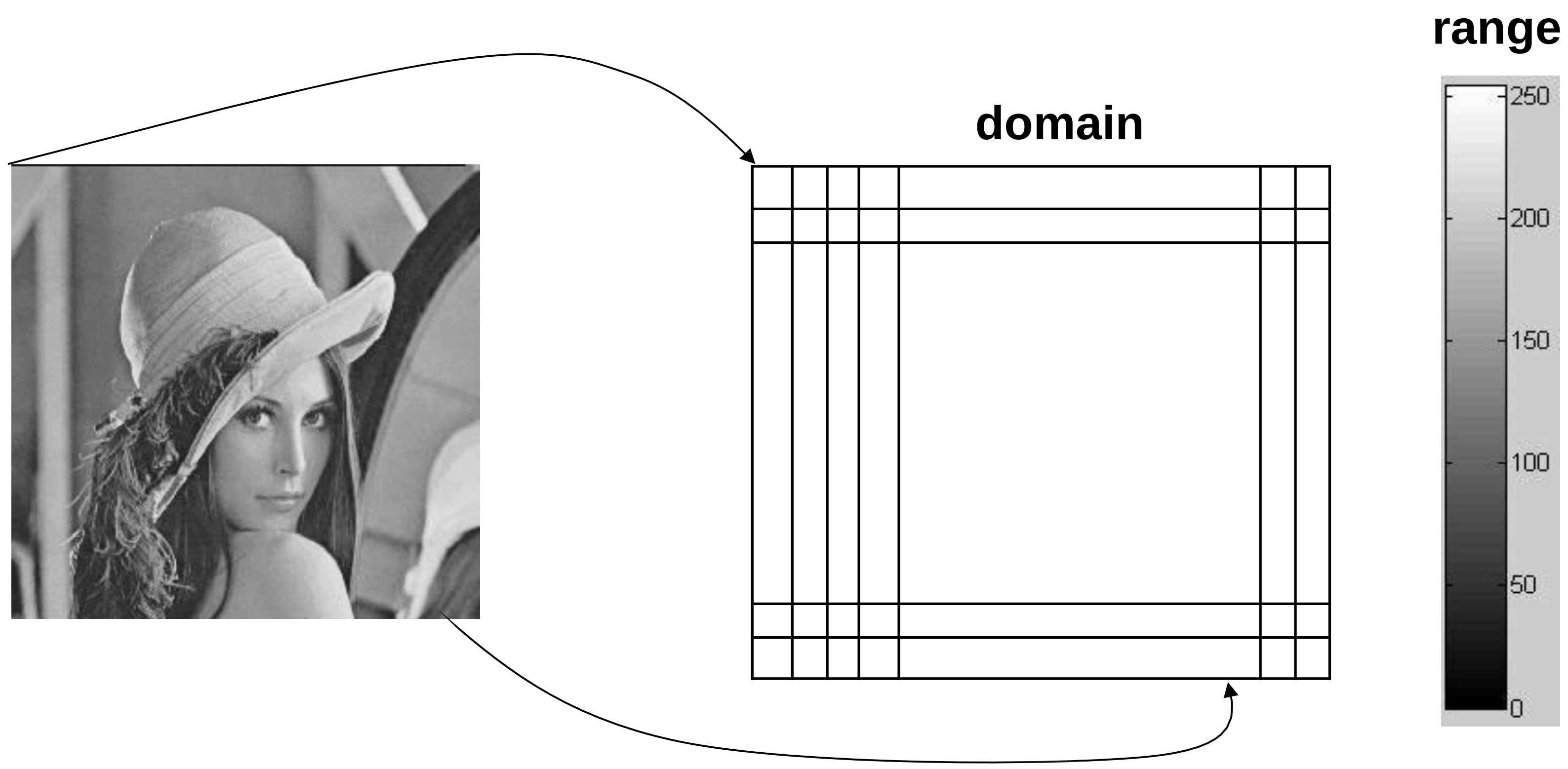
- An image contains discrete number of pixels
  - A simple example
  - Pixel value:
    - “grayscale”

    (or “intensity”): [0,255]
    - “color”
    - RGB: [R, G, B]
    - Lab: [L, a, b]
    - HSV: [H, S, V]



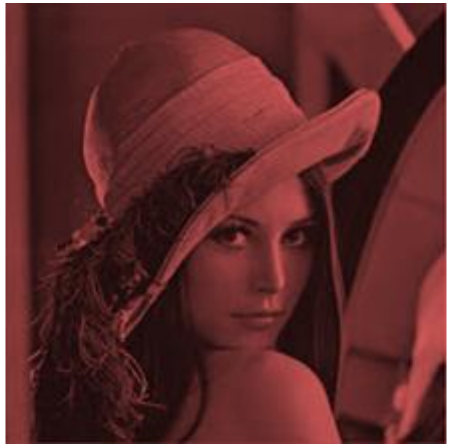
*Source: F.F. Li*

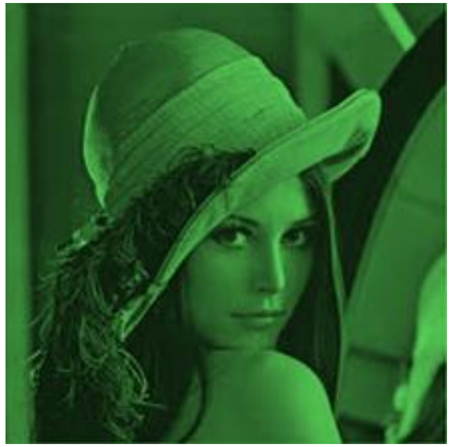
## Filtering

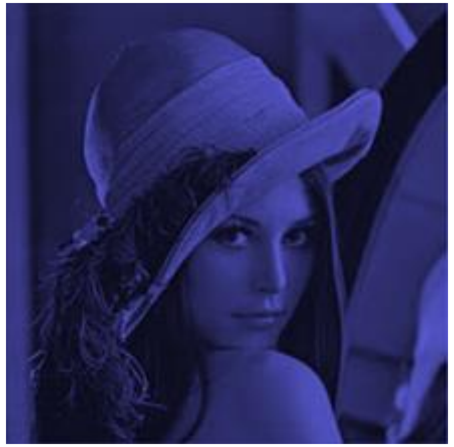


*Source: F.F. Li*

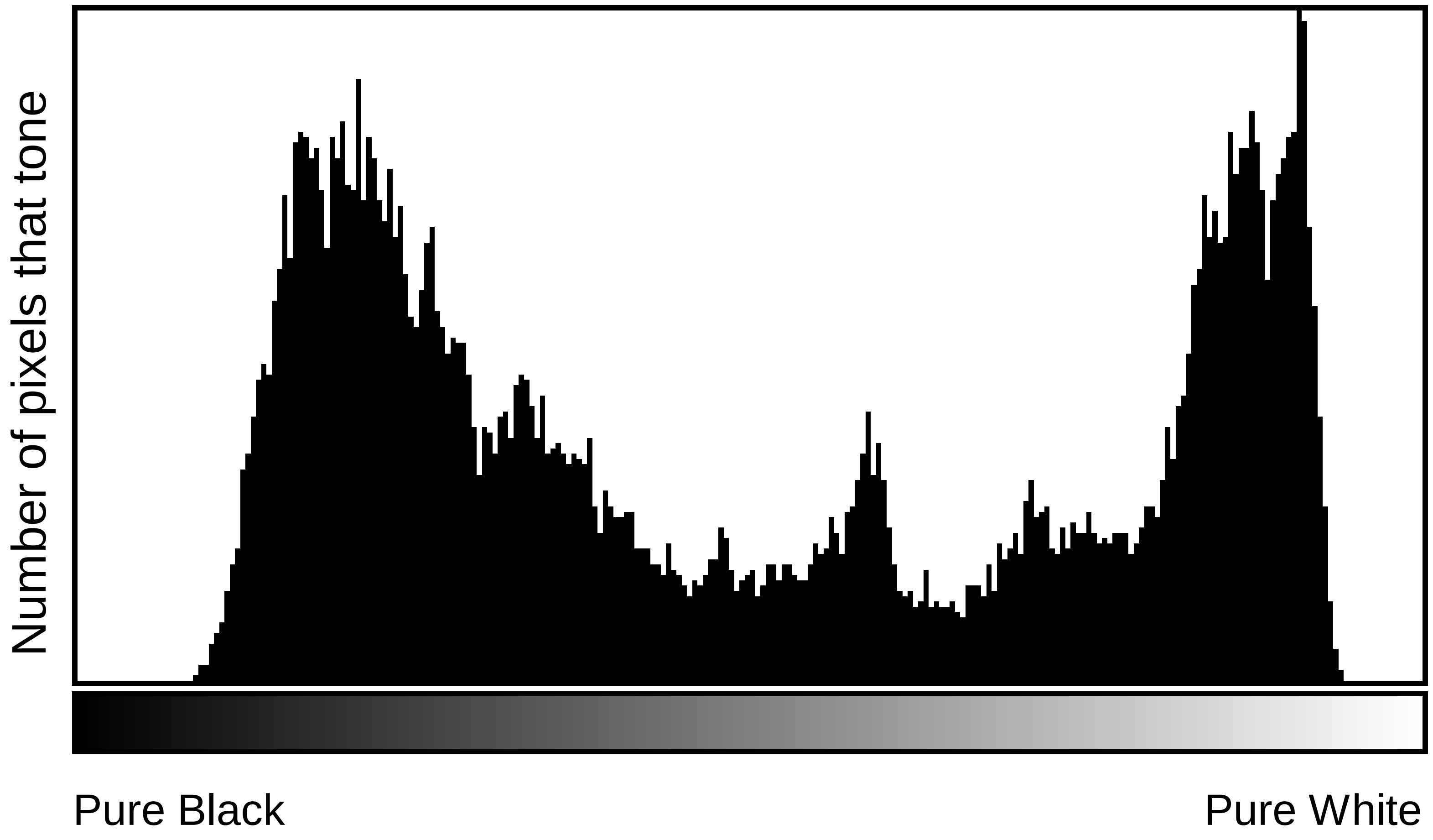
## Filtering : RGB Channels





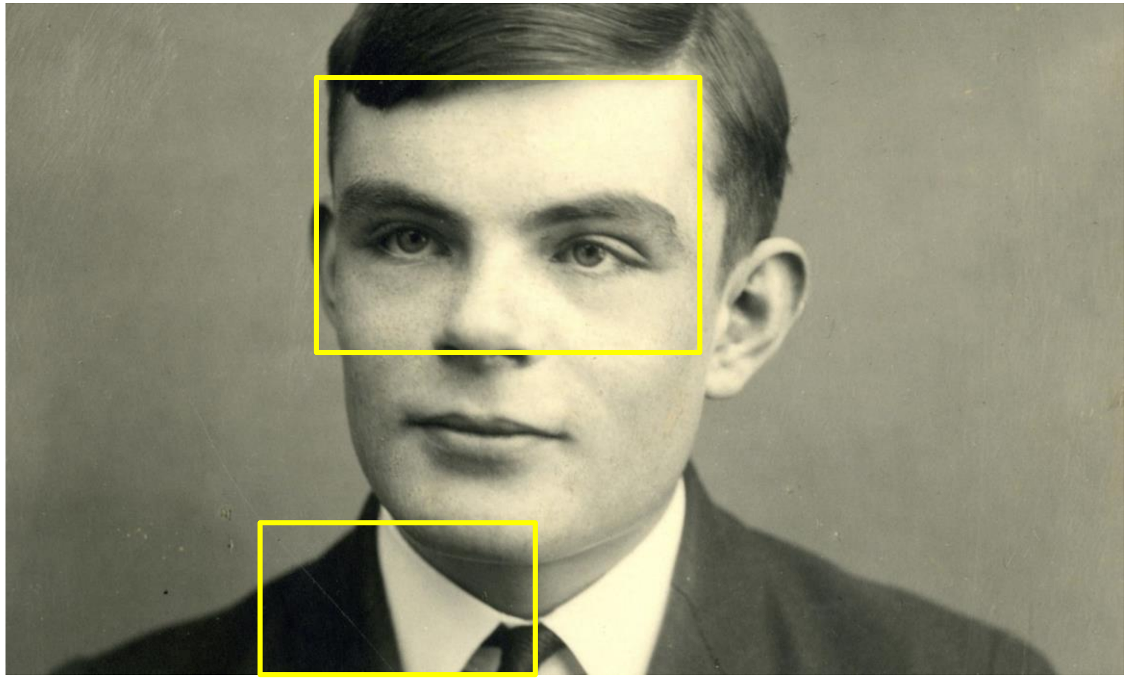


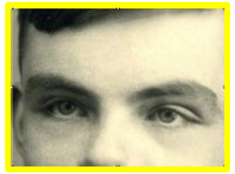
## Image Histogram

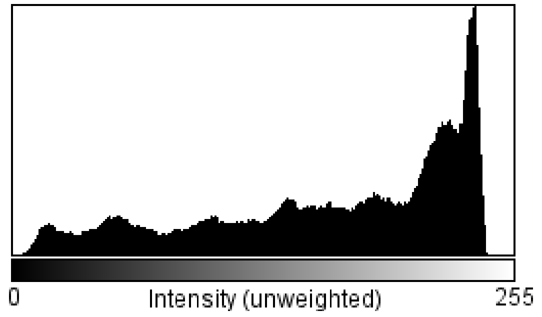


## Image Histogram

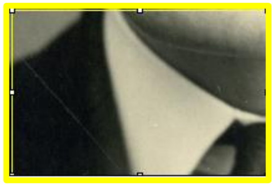
Use ImageJ and/or FIJI

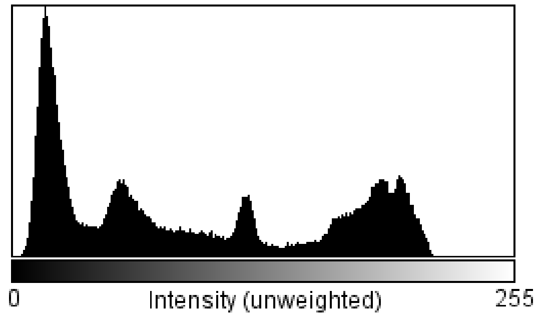






Count: 306876 Min: 2
Mean: 162.894 Max: 242
StdDev: 66.981 Mode: 236 (6220)





Count: 109592 Min: 2
Mean: 90.390 Max: 218
StdDev: 69.596 Mode: 16 (2729)

## Correlation

$$I \bigotimes W = \sum_{k} \sum_{l} I(k,l)W(i+k, j+l)$$

$I$ = Image

$W$ = Kernel

$$\overset{\boldsymbol{I}}{\begin{array}{|c|c|c|}\hline \mathrm{i}_1 & \mathrm{i}_2 & \mathrm{i}_3 \\ \hline \mathrm{i}_4 & \mathrm{i}_5 & \mathrm{i}_6 \\ \hline \mathrm{i}_7 & \mathrm{i}_8 & \mathrm{i}_9 \\ \hline \end{array}} \otimes \overset{\boldsymbol{W}}{\begin{array}{|c|c|c|}\hline \mathrm{W}_1 & \mathrm{w}_2 & \mathrm{w}_3 \\ \hline \mathrm{w}_4 & \mathrm{w}_5 & \mathrm{w}_6 \\ \hline \mathrm{w}_7 & \mathrm{w}_8 & \mathrm{w}_9 \\ \hline \end{array}} \longrightarrow$$

$$I * W = i_1w_1+i_2w_2+i_3w_3$$
$$+i_4w_4+i_5w_5+i_6w_6$$
$$+i_7w_7+i_8w_8+i_9w_9$$

## Convolution

$$I \bigotimes W = \sum_{k} \sum_{l} I(k,l)W(i-k,j-l)$$

$I$ = Image

$W$ = Kernel

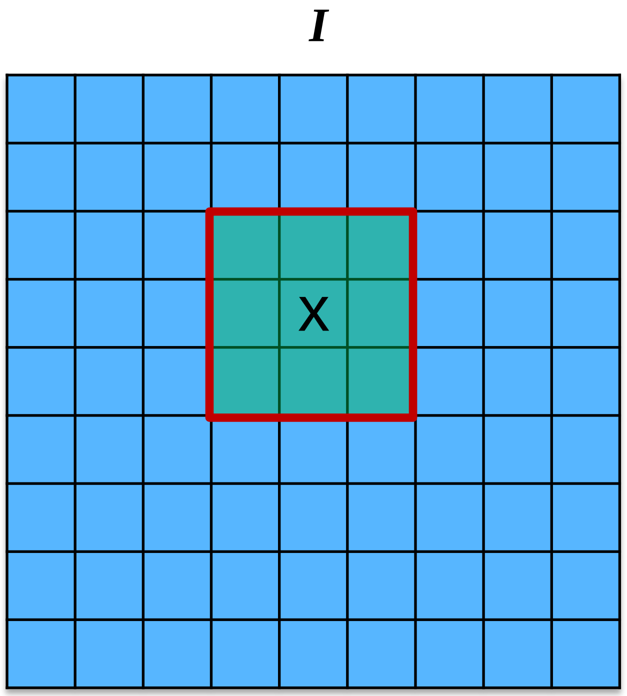
$$\overset{\boldsymbol{w}}{\begin{array}{|c|c|c|}\hline w_1 & w_2 & w_3 \\ \hline w_4 & w_5 & w_6 \\ \hline w_7 & w_8 & w_9 \\ \hline\end{array}} \xrightarrow{\text{X - flip}} \begin{array}{|c|c|c|}\hline w_7 & w_8 & w_9 \\ \hline w_4 & w_5 & w_6 \\ \hline w_1 & w_2 & w_3 \\ \hline\end{array} \xrightarrow{\text{Y - flip}} \begin{array}{|c|c|c|}\hline w_9 & w_8 & w_7 \\ \hline w_6 & w_5 & w_4 \\ \hline w_3 & w_2 & w_1 \\ \hline\end{array}$$

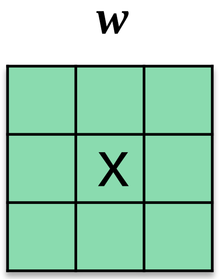
$$\overset{\boldsymbol{I}}{\begin{array}{|c|c|c|}\hline f_1 & f_2 & f_3 \\ \hline f_4 & f_5 & f_6 \\ \hline f_7 & f_8 & f_9 \\ \hline\end{array}} \bigotimes \begin{array}{|c|c|c|}\hline w_9 & w_8 & w_7 \\ \hline w_6 & w_5 & w_4 \\ \hline w_3 & w_2 & w_1 \\ \hline\end{array} \longrightarrow$$

$$I * W = i_1 w_9 + i_2 w_8 + i_3 w_7$$
$$+ i_4 w_6 + i_5 w_5 + i_6 w_4$$
$$+ i_7 w_3 + i_8 w_2 + i_9 w_1$$

## Convolution

$$I(x,y) * W = I(x+1,y+1)W(-1,-1) + I(x,y+1)W(0,-1) + I(x-1,y+1)W(1,-1) +$$
$$I(x+1,y)W(-1,0) + I(x,y)W(0,0) + I(x-1,y)W(1,0) +$$
$$I(x+1,y-1)W(-1,1) + I(x,y-1)W(0,1) + I(x-1,y-1)W(1,1)$$





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

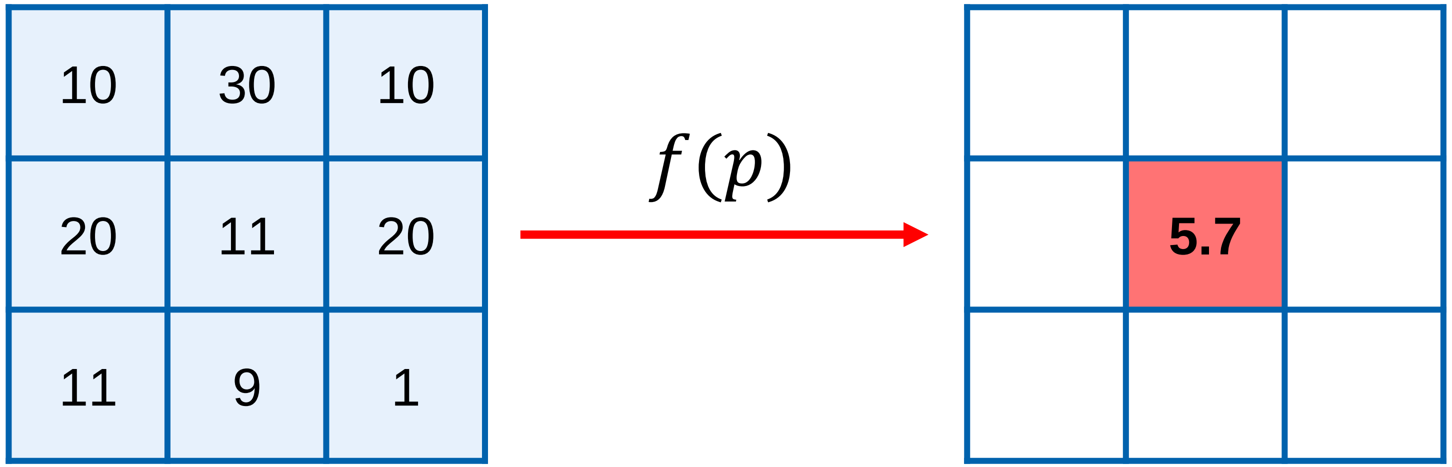
**Coordinates**

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

## Filtering

## Filtering

- Modify pixels based on some function of the neighbourhood



## Filtering

- The output is the linear combination of the neighbourhood pixels

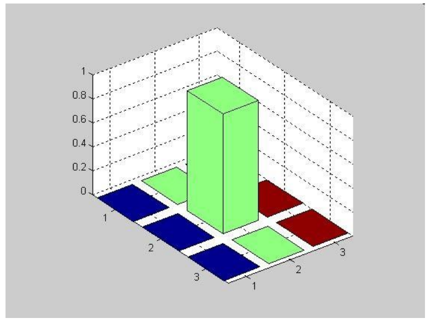
$$\underset{\text{Image}}{\begin{bmatrix} 1 & 3 & 0 \\ 2 & 10 & 2 \\ 4 & 1 & 1 \end{bmatrix}} \otimes \underset{\text{Kernel}}{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0.1 & -1 \\ 1 & 0 & -1 \end{bmatrix}} =$$

## Filtering

- The output is the linear combination of the neighbourhood pixels

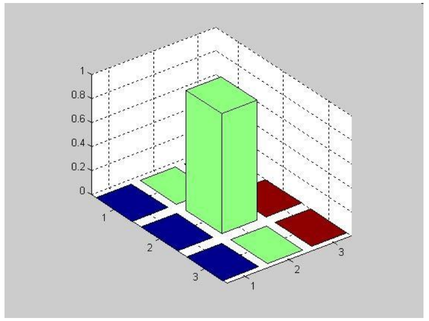
$$\underset{\text{Image Input}}{\begin{bmatrix}1&3&0\\2&10&2\\4&1&1\end{bmatrix}} \otimes \underset{\text{Kernel Weight}}{\begin{bmatrix}1&0&-1\\1&0.1&-1\\1&0&-1\end{bmatrix}} = \underset{\text{Filter Output}}{\begin{bmatrix}\phantom{0}&\phantom{0}&\phantom{0}\\ &5& \\ & & \end{bmatrix}}$$

## Filtering examples



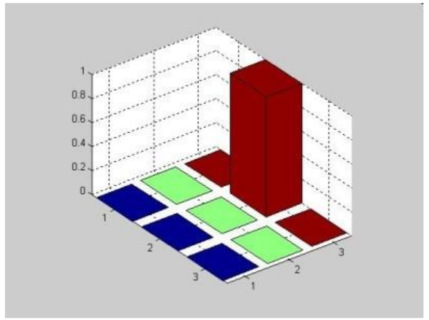
$$\begin{array}{c} * \end{array}\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} =$$

## Filtering examples



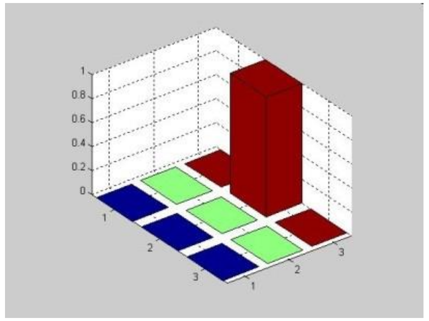
$$* \quad \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \quad =$$

## Filtering examples



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

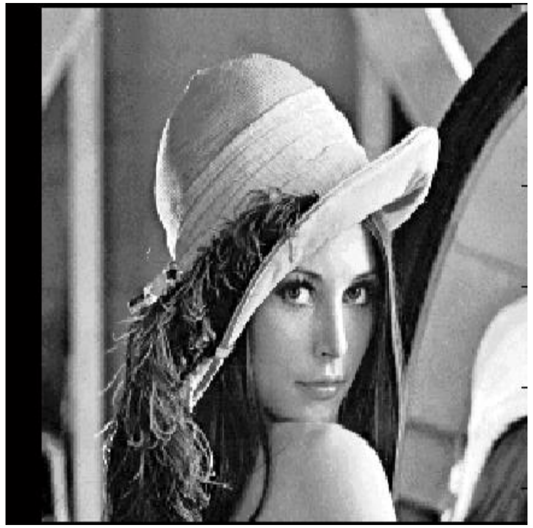
## Filtering examples



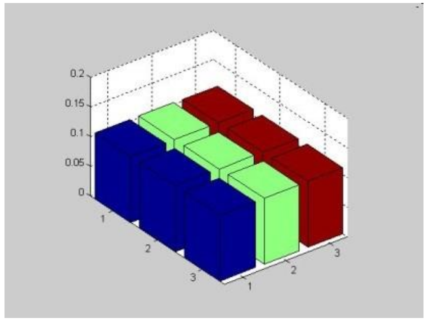
*

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

=

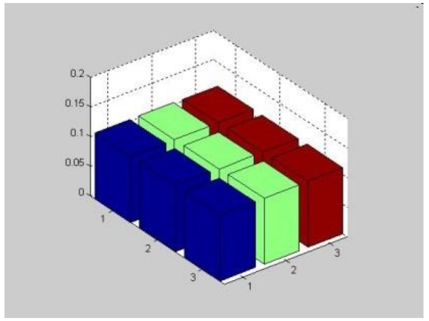


## Filtering examples

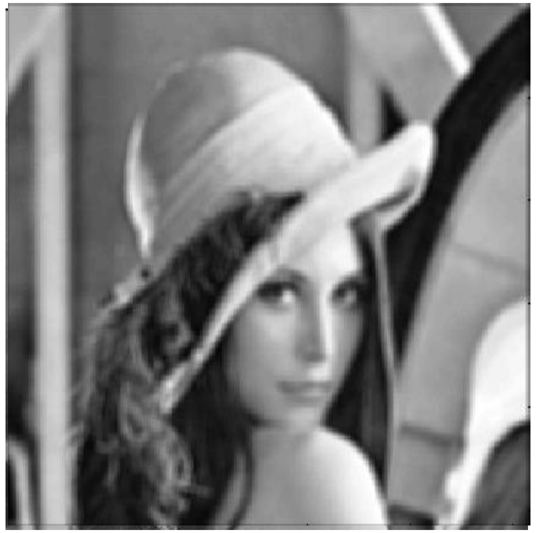


$$* \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

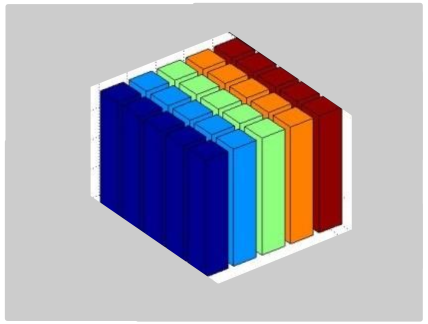
## Filtering examples



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

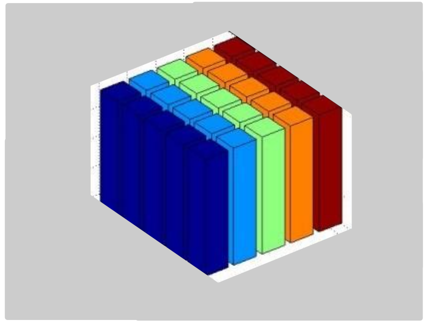


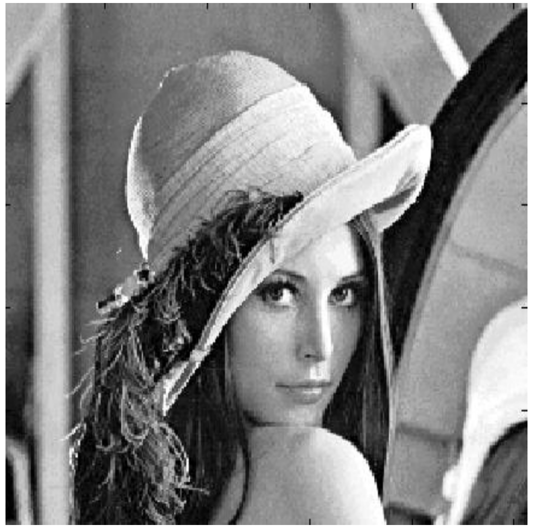
# Filtering examples



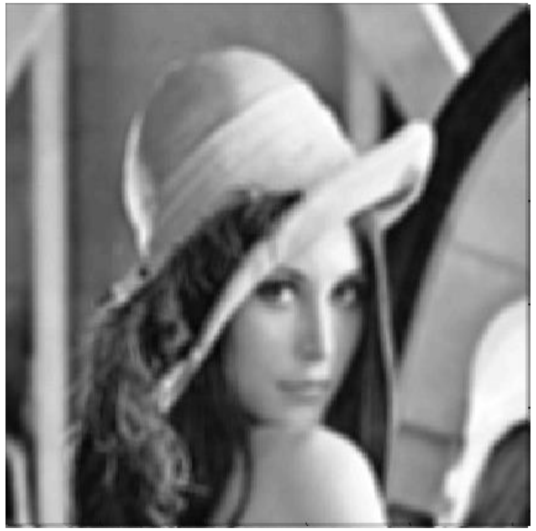
$$* \frac{1}{25} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} =$$

## Filtering examples



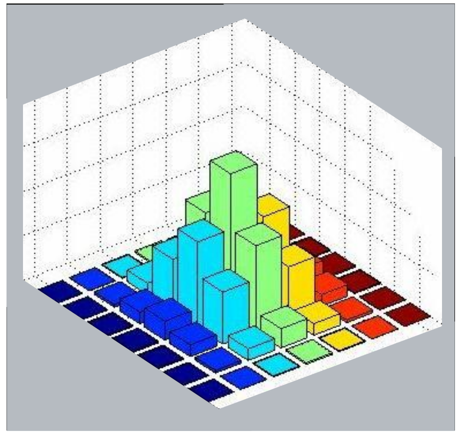


$$* \frac{1}{25} \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} =$$

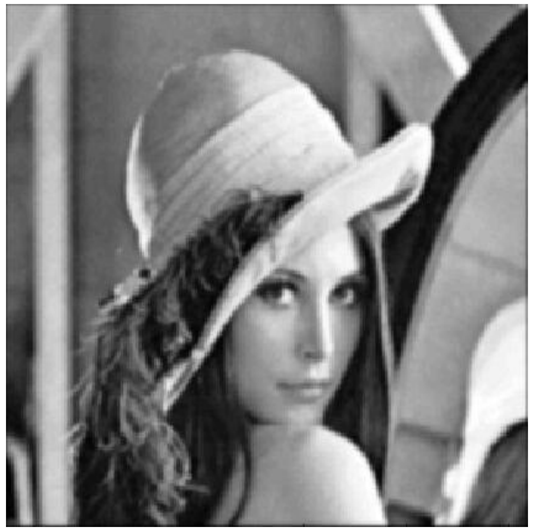


## Filtering examples - Gaussian

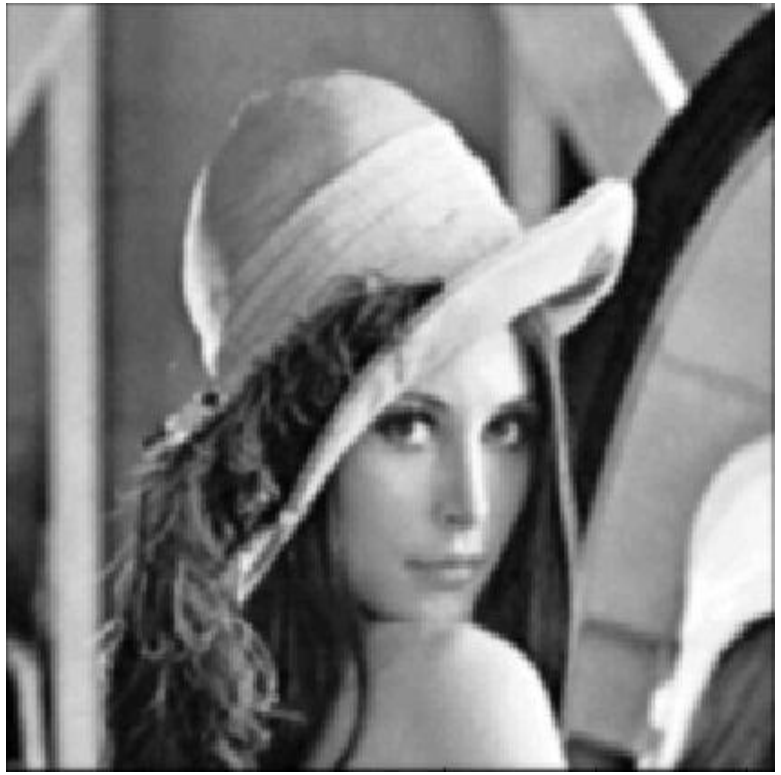
$*$

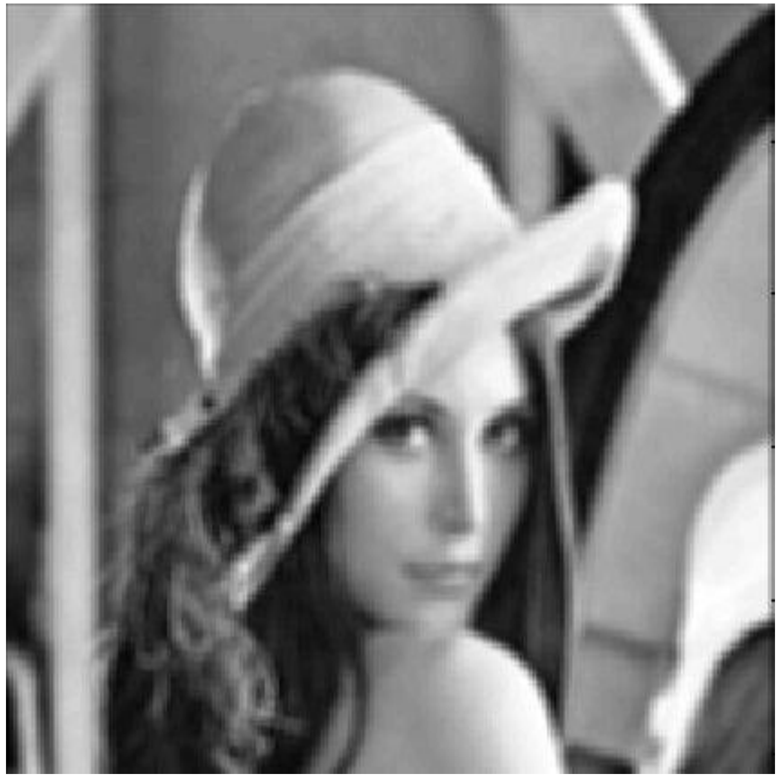


$=$

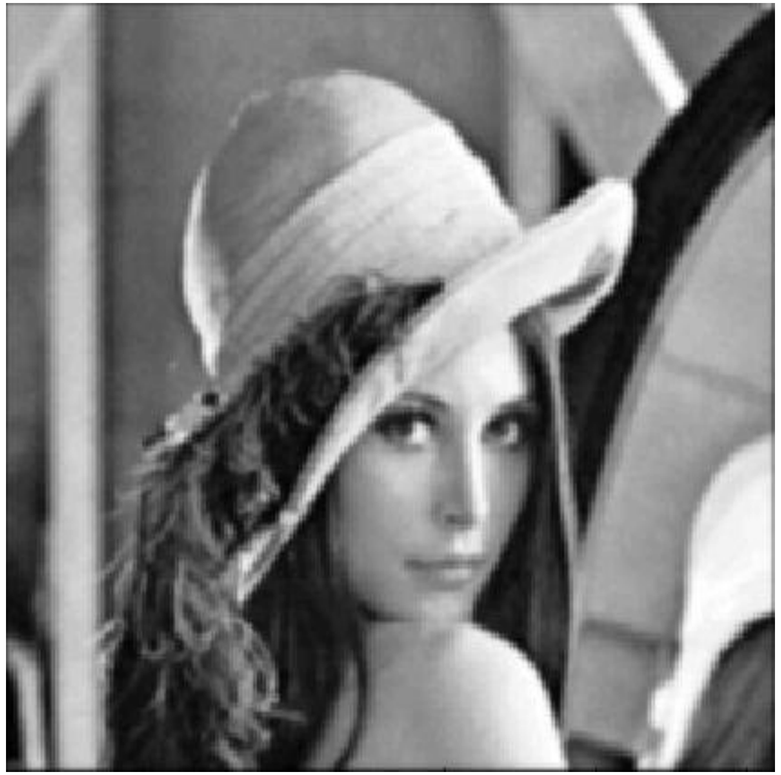


## Filtering example – Gaussian vs. Smoothing

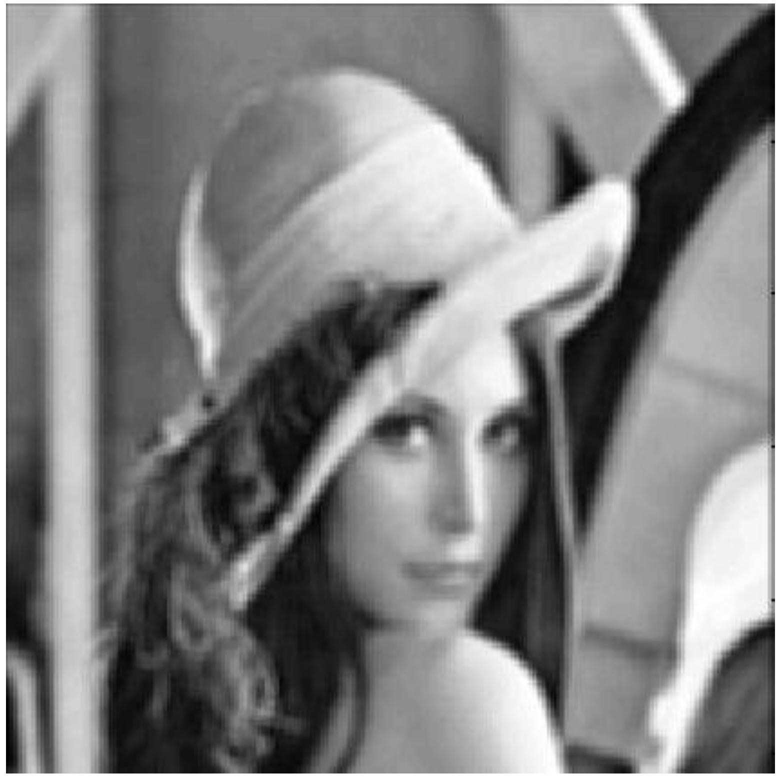

Gaussian Smoothing


Smoothing by Averaging

# Filtering example – Noise filtering

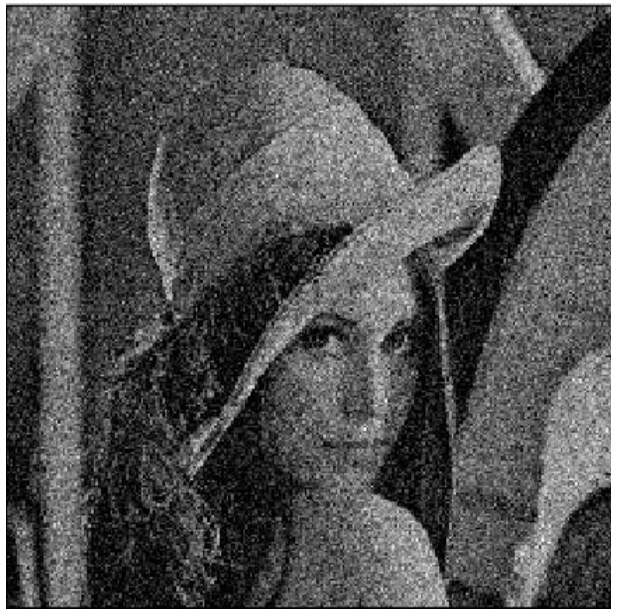


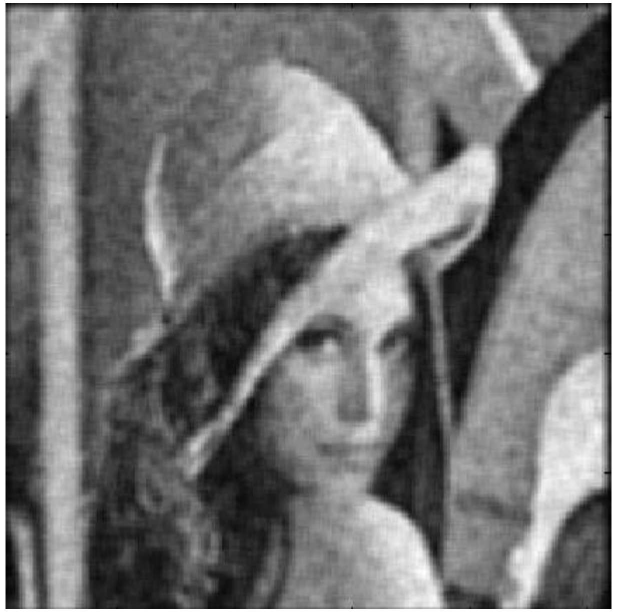
Gaussian Smoothing

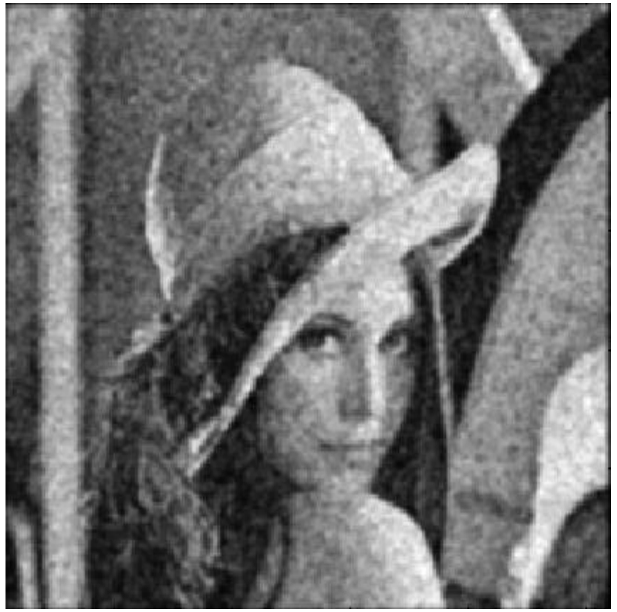


Smoothing by Averaging

## Filtering example – Noise filtering


Gaussian Noise


After averaging


After Gaussian Smoothing

greatlearning
Learning for Life

Thank you!