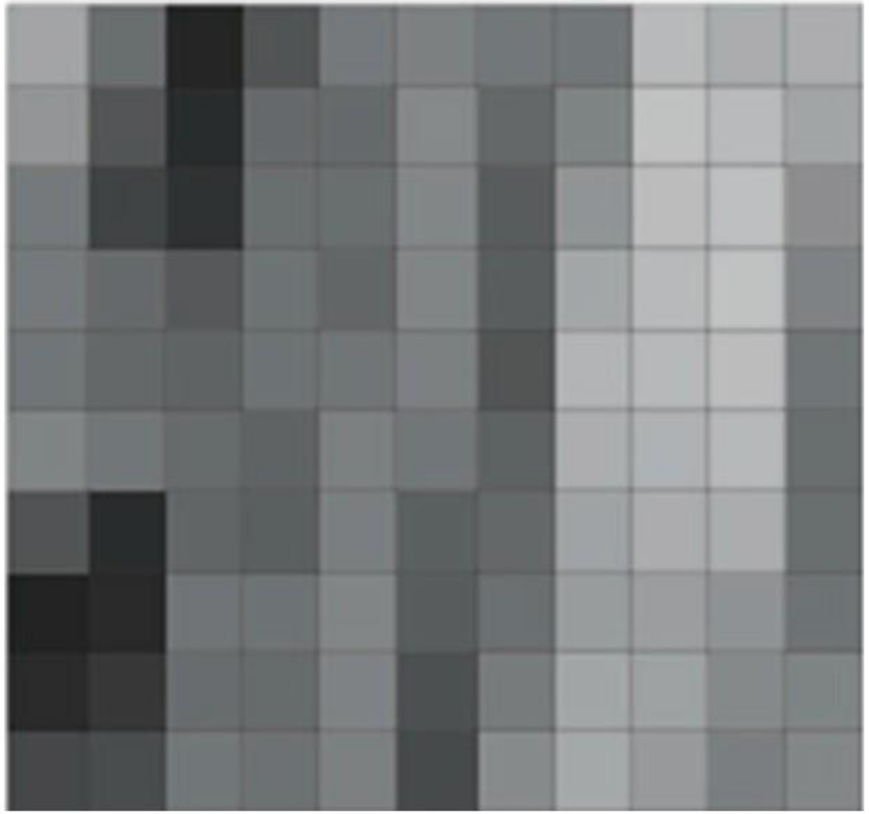
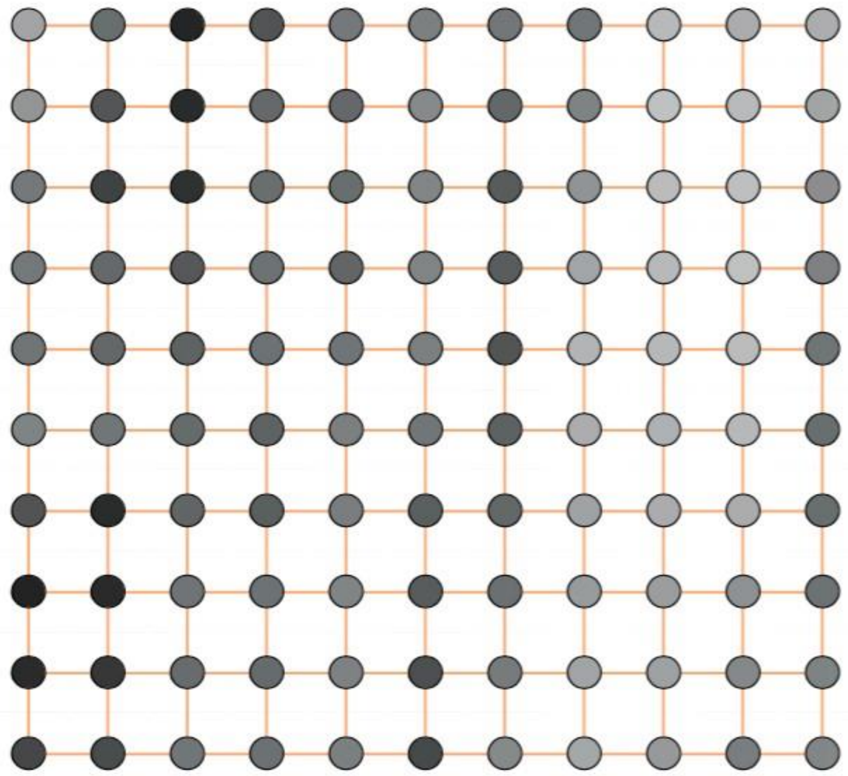
Computer Vision: Working with Images

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', l) #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

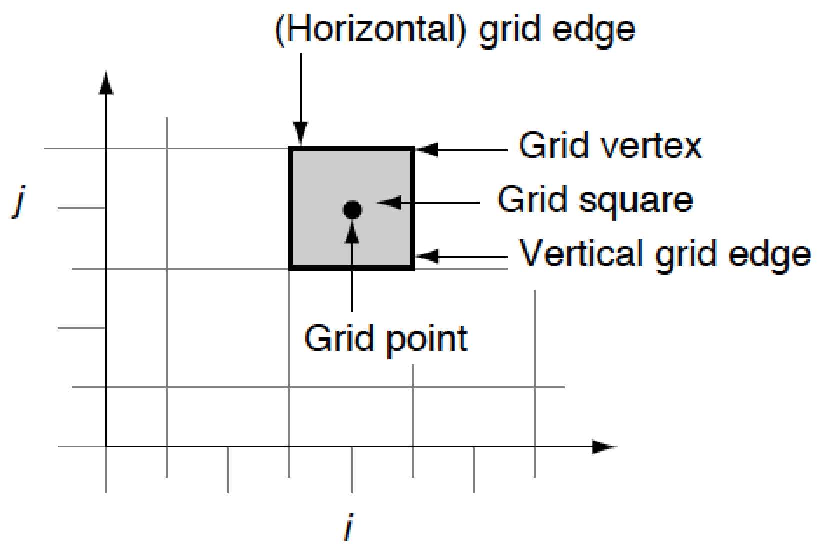
More powerful features than the

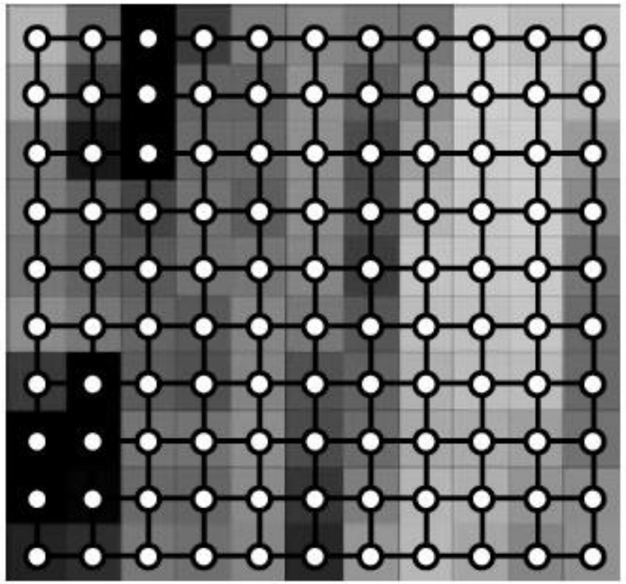
Definition

- 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

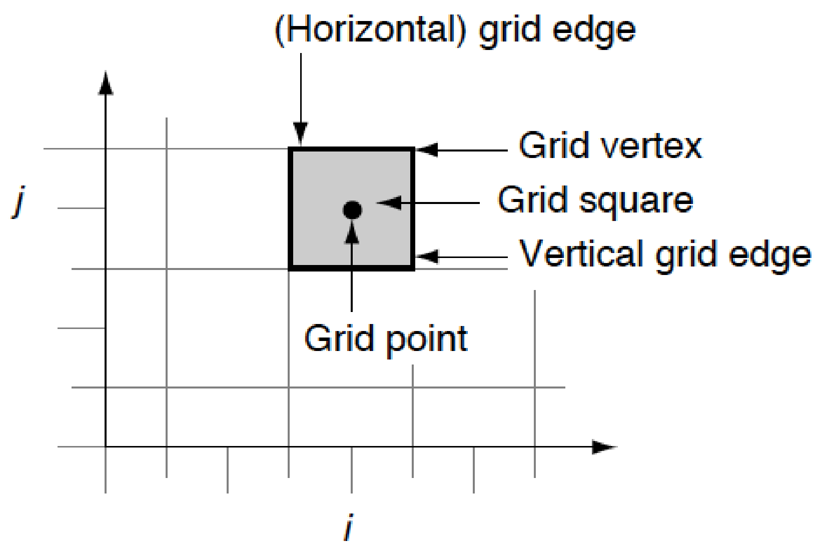
- A (2D) picture P is a function defined on a (finite) rectangular subset G of a regular planar orthogonal array. G is called (2D) grid, and an element of G is called pixel. P assigns a value of P(p) to each point

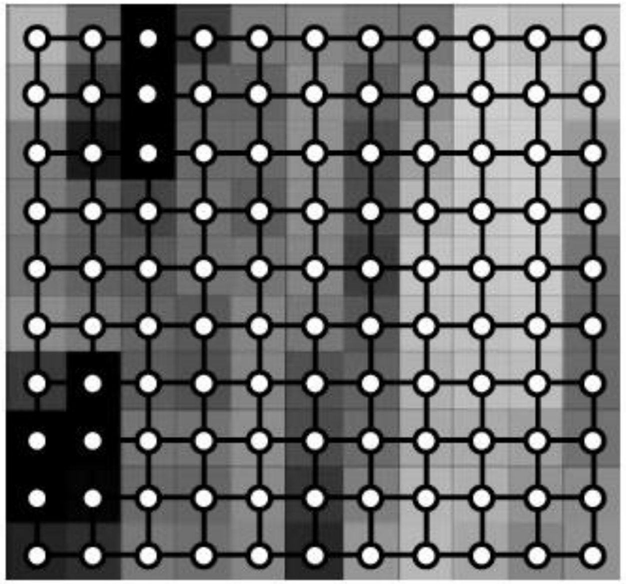




Definition

- 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, …)





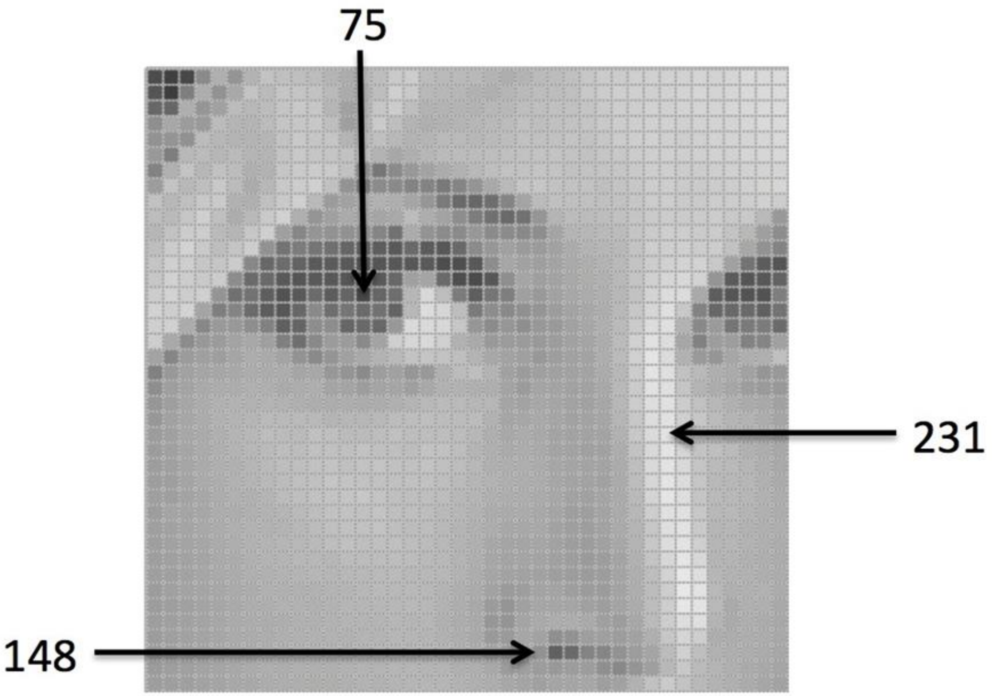
Resolution

- 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

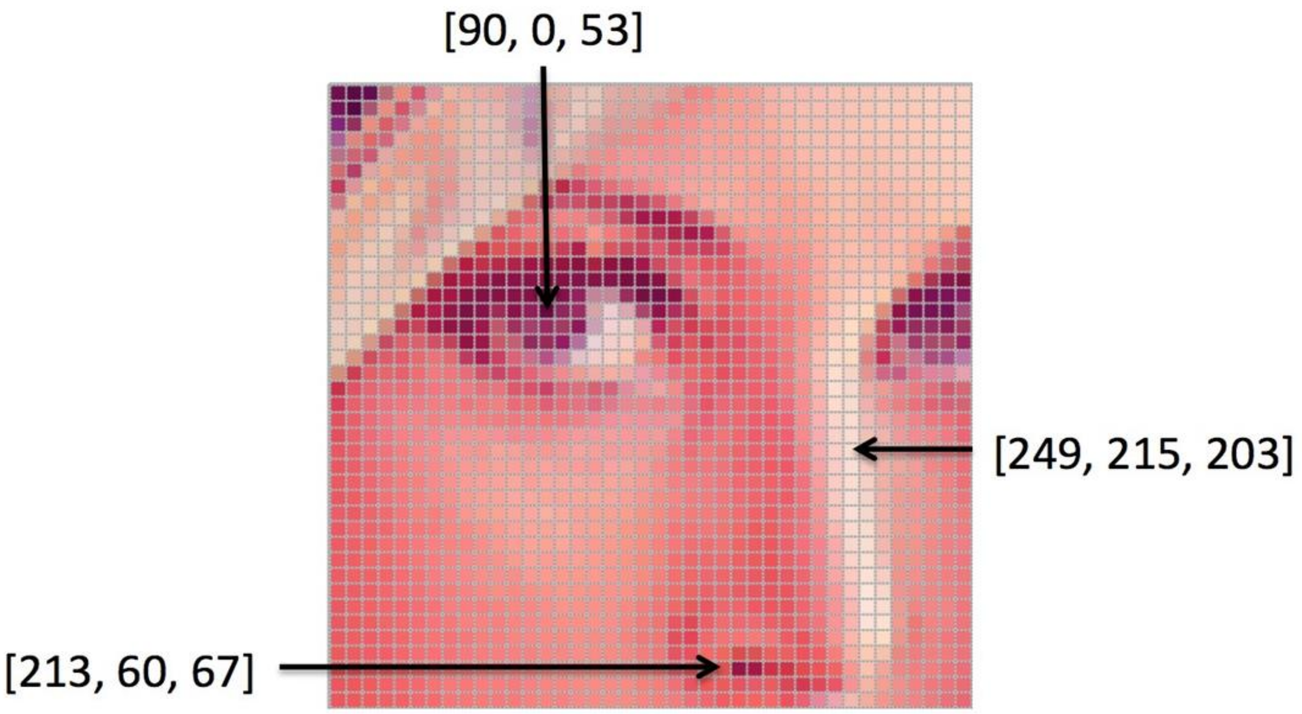
- An image contains discrete number of pixels
  - A simple example
  - Pixel value:
    - “grayscale”

    (or “intensity”): [0,255]
    - “color”

    – RGB: [R, G, B]

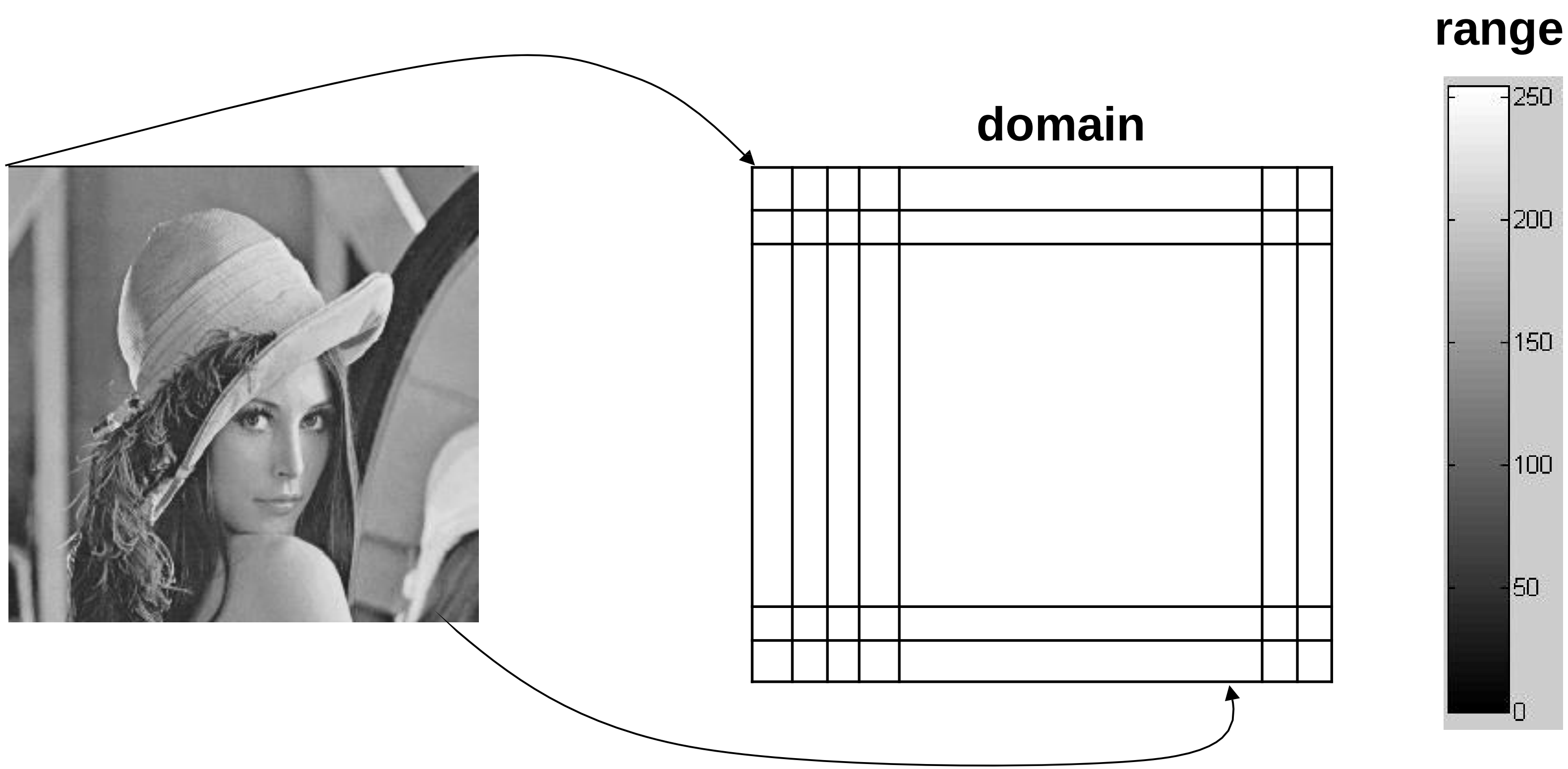
    – Lab: [L, a, b]

    – HSV: [H, S, V]



Source: F.F. Li

Filtering



Source: F.F. Li

Filtering : RGB Channels

Image Histogram

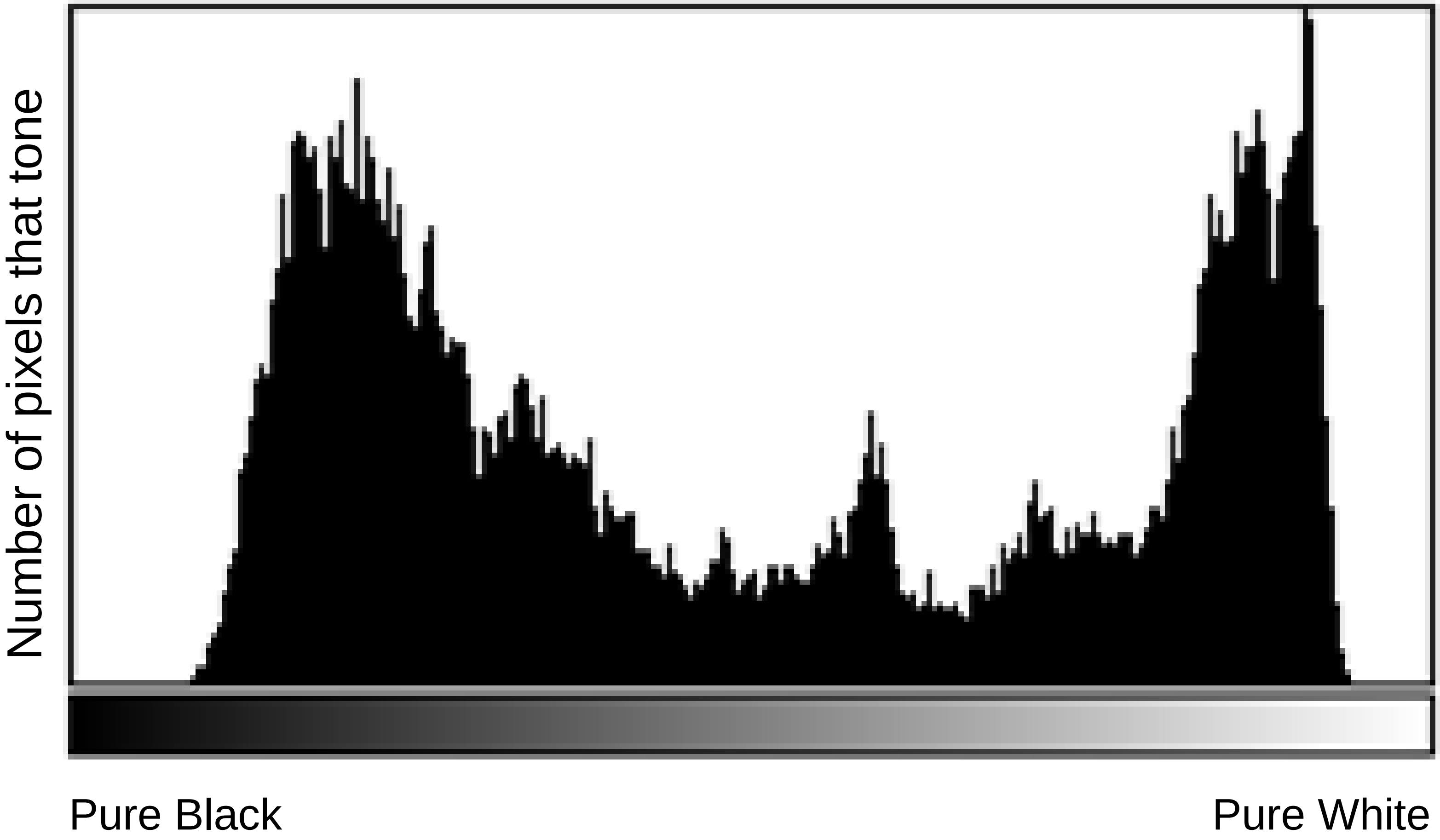
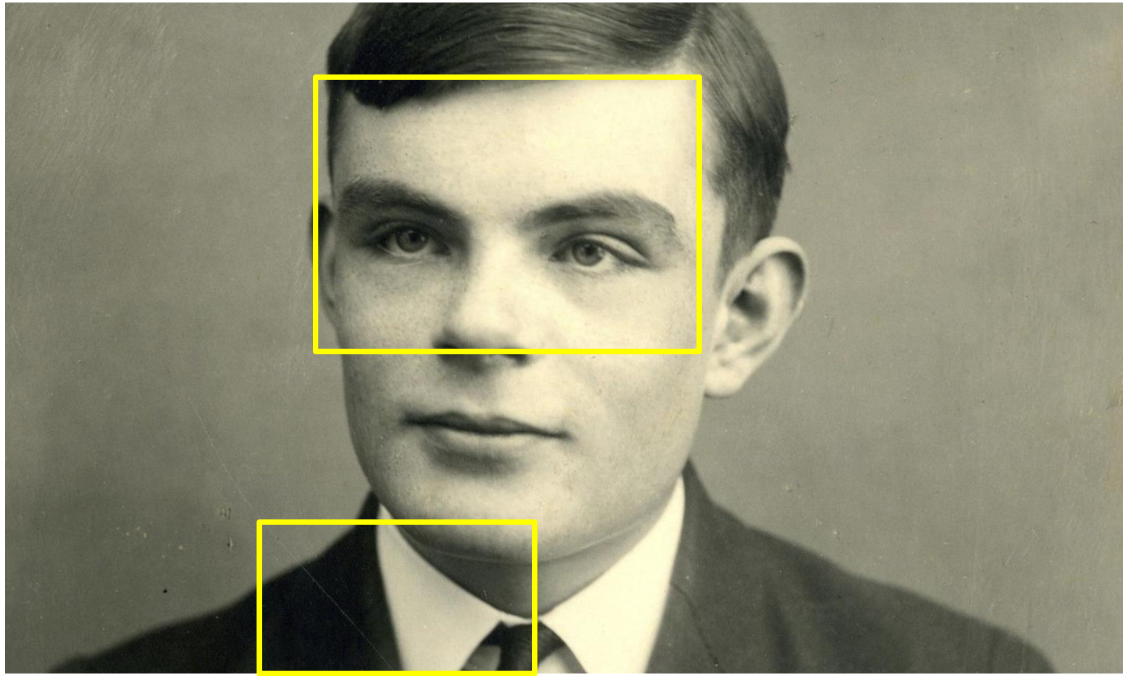
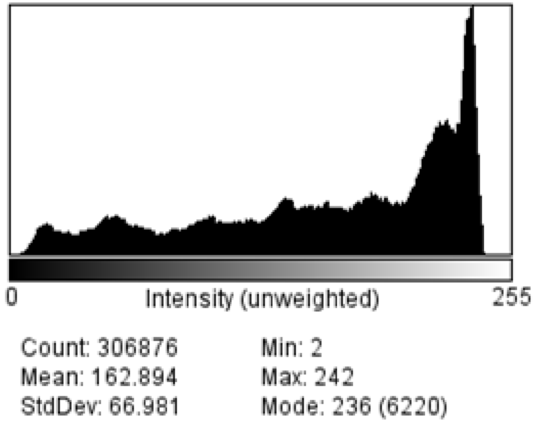
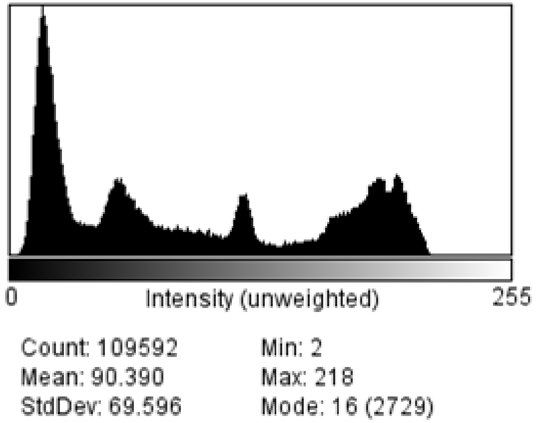


Image Histogram

Use ImageJ and/or FIJI







Convolution

$$I \bigotimes W = \sum_k \sum_l I(k,l) W(i+k, j+l)$$

I = Image

W = Kernel

I

i_1	i_2	i_3
i_4	i_5	i_6
i_7	i_8	i_9

$\bigotimes$

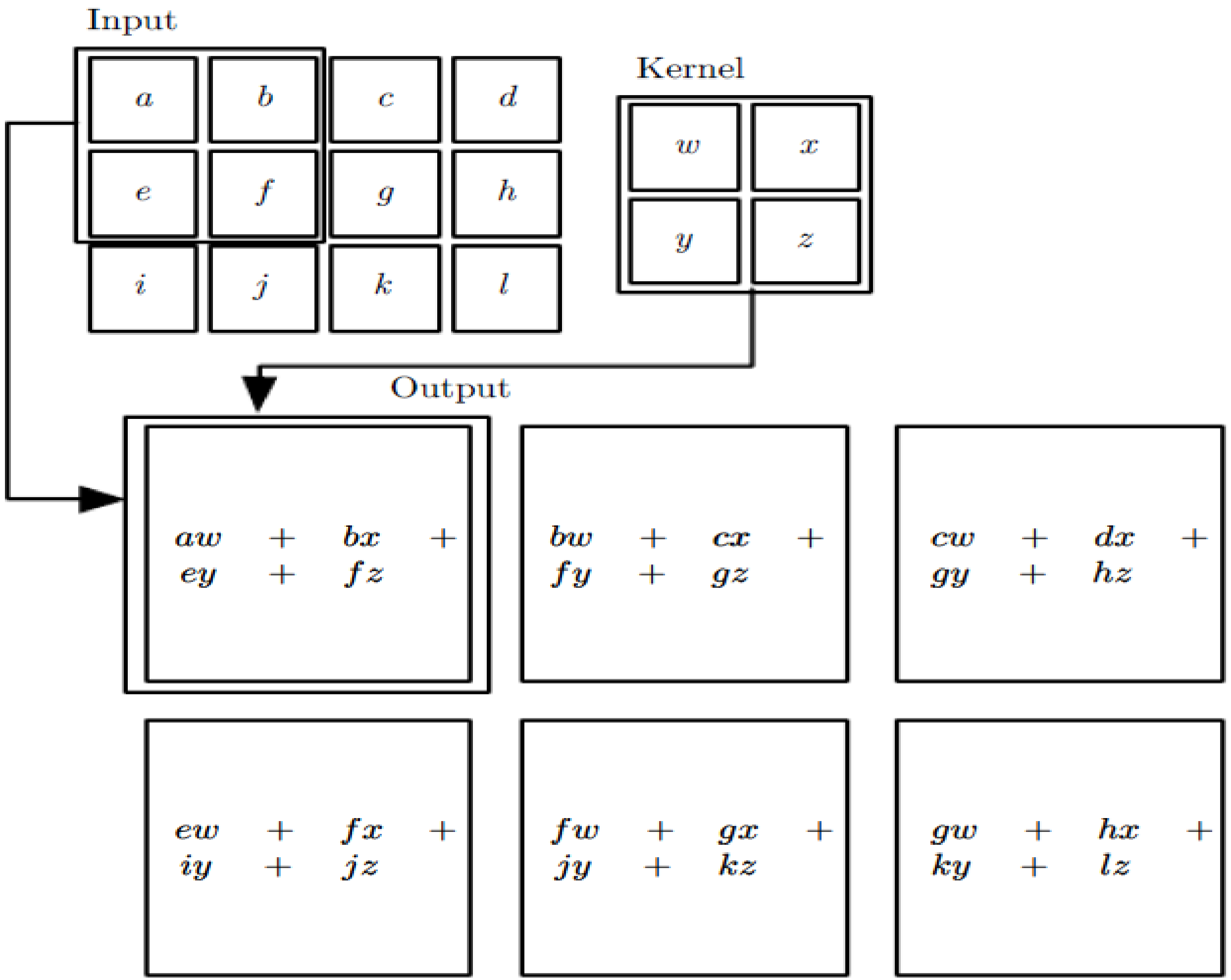
W

W_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$\longrightarrow$

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

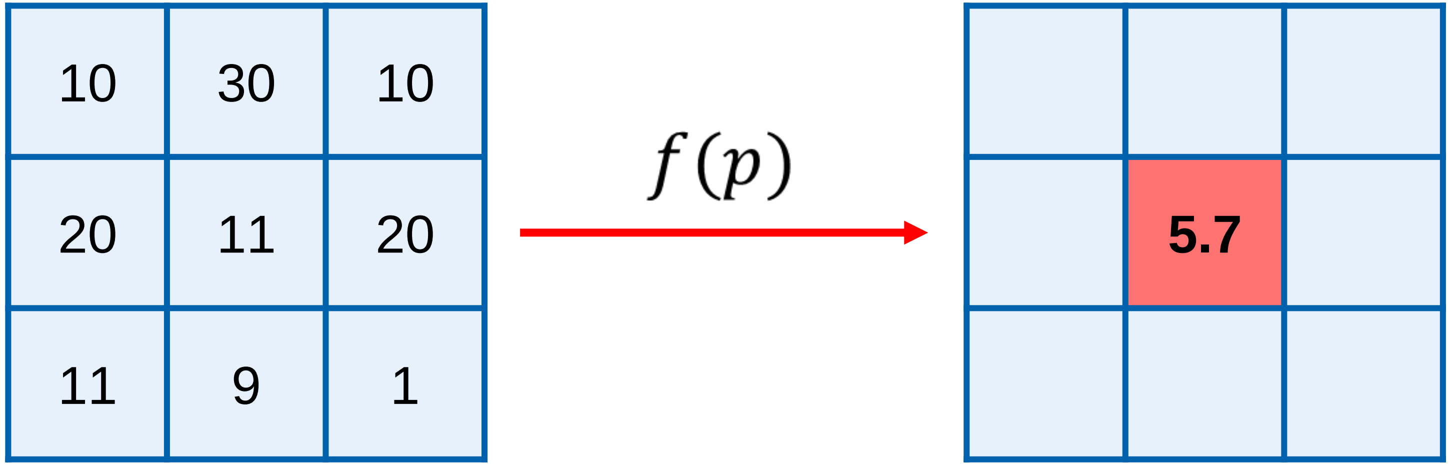
2D Convolution



Filtering

Filtering

- Modify pixels based on some function of the neighbourhood



Filtering

- The output is the linear combination of the neighbourhood pixels

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

Filtering

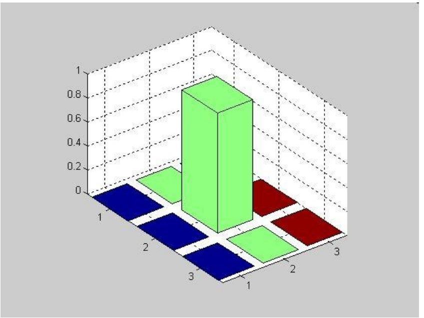
- The output is the linear combination of the neighbourhood pixels

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

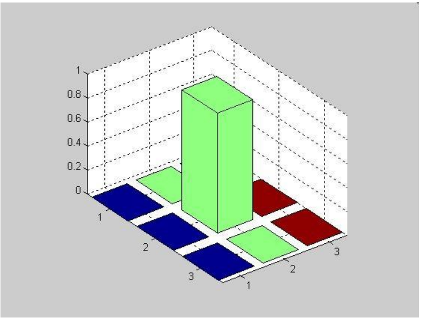
Image Input | Kernel Weight | Filter Output

Filtering examples

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



Filtering examples

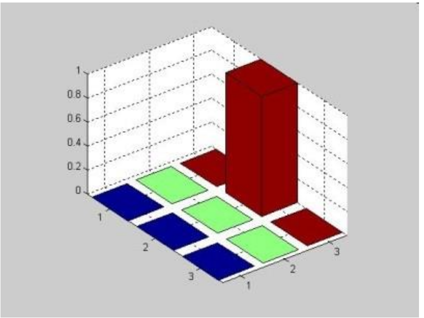


*

0	0	0
0	1	0
0	0	0

=

Filtering examples

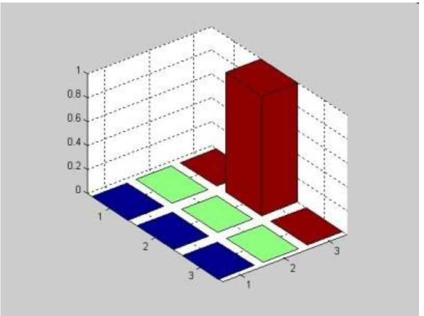


*

0	0	0
1	0	0
0	0	0

=

Filtering examples

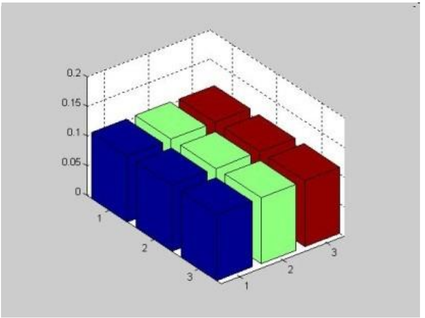


*

0	0	0
1	0	0
0	0	0

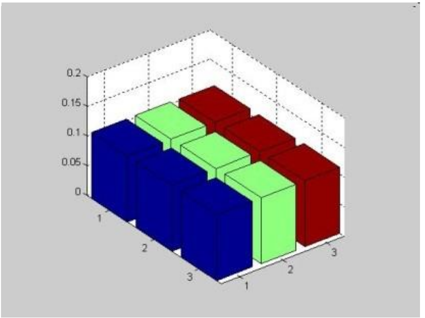
=

Filtering examples



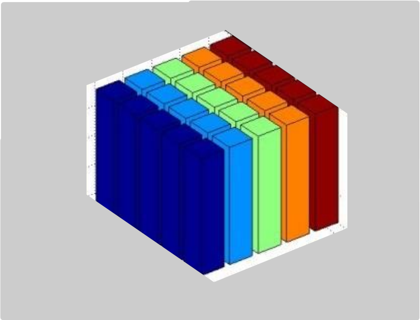
$$* \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} =$$

Filtering examples



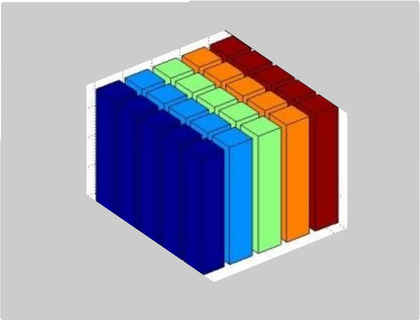
$$* \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} =$$

Filtering examples



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

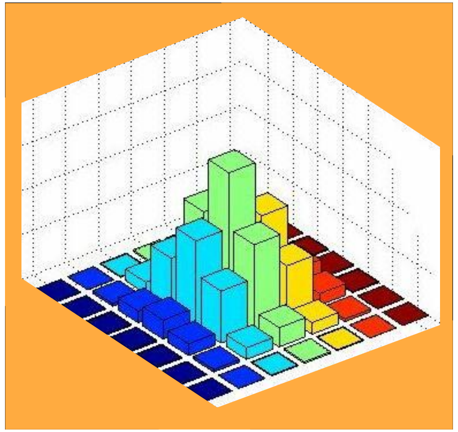
Filtering examples



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

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

Thank you!

Happy Learning :)