Image Processing

Javier González Jiménez

Reference Books:

- *Computer Vision: Algorithms and Applications*. Richard Szeliski. Springer. 2010.
http://szeliski.org/Book
- *Visión por Computador.* Javier Gonzalez Jim*enez. Thomson-Paraninfo, 1999.*

Content

1. Introduction
2. IP tools and concepts
    - Image color
    - Image Histogram
    - Look-up-tables
    - Distance between pixels
    - Convolution
3. Image Smoothing
4. Image Enhancement

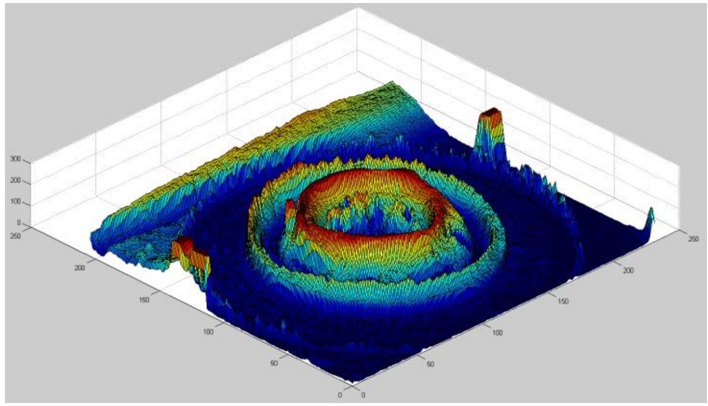
1. Introduction

Goal: Improve the quality of the image by attenuating noise, adjusting colors, modifing contrast and brightness, …

In CV this may be a necessary step to prepare the image for a better feature extraction or segmentation

An image can be seen as:

a two-dimensional matrix A[x,y]



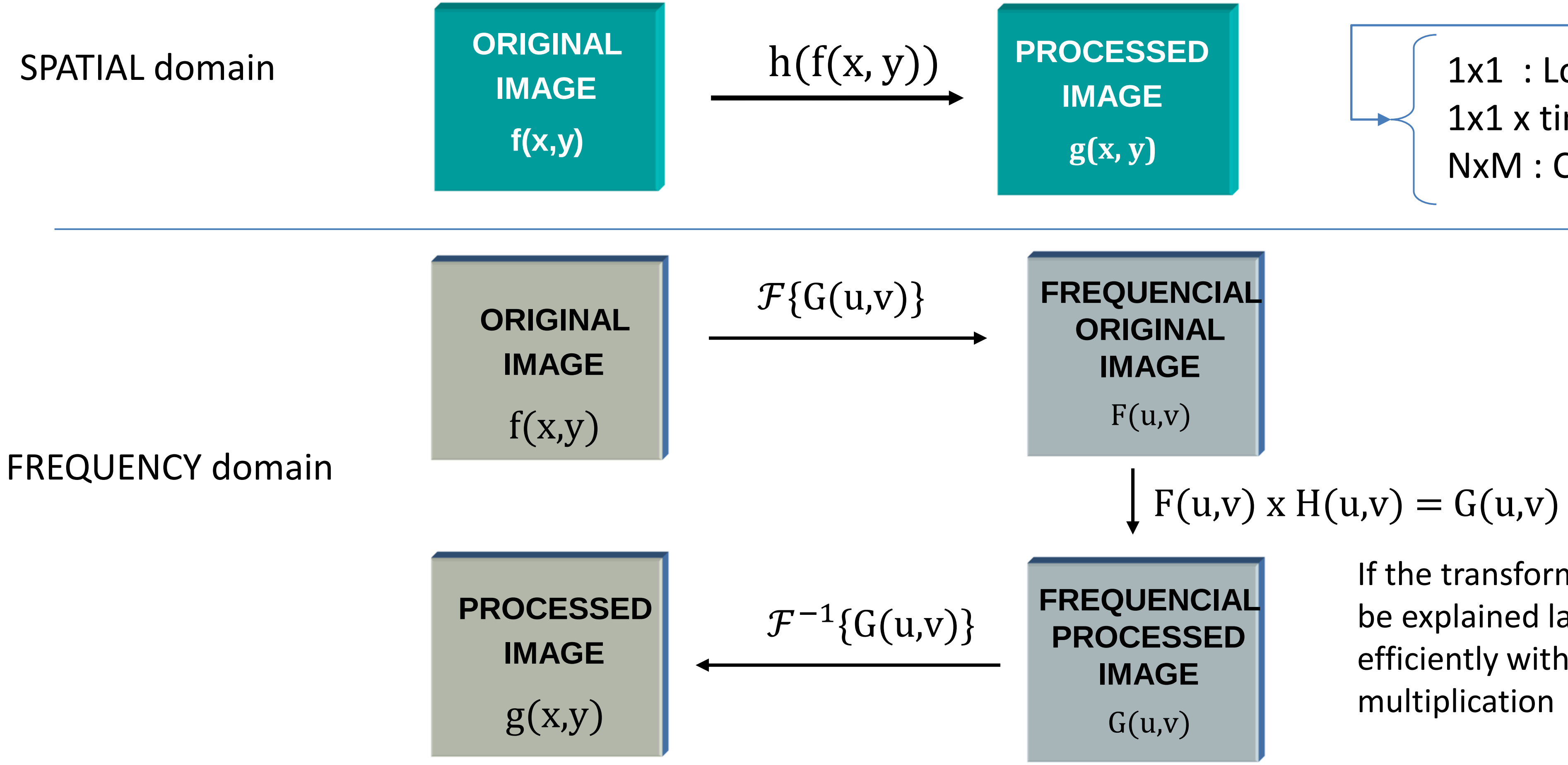
a two-dimensional function f(x,y) with (x,y) discrete

Both are the same thing!

1. Introduction

Smoothing and enhancement can be applied in the SPATIAL and in the FREQUENCY domain

h is a transformation fuction that takes values from a <u>neighborhood</u> of (x,y)



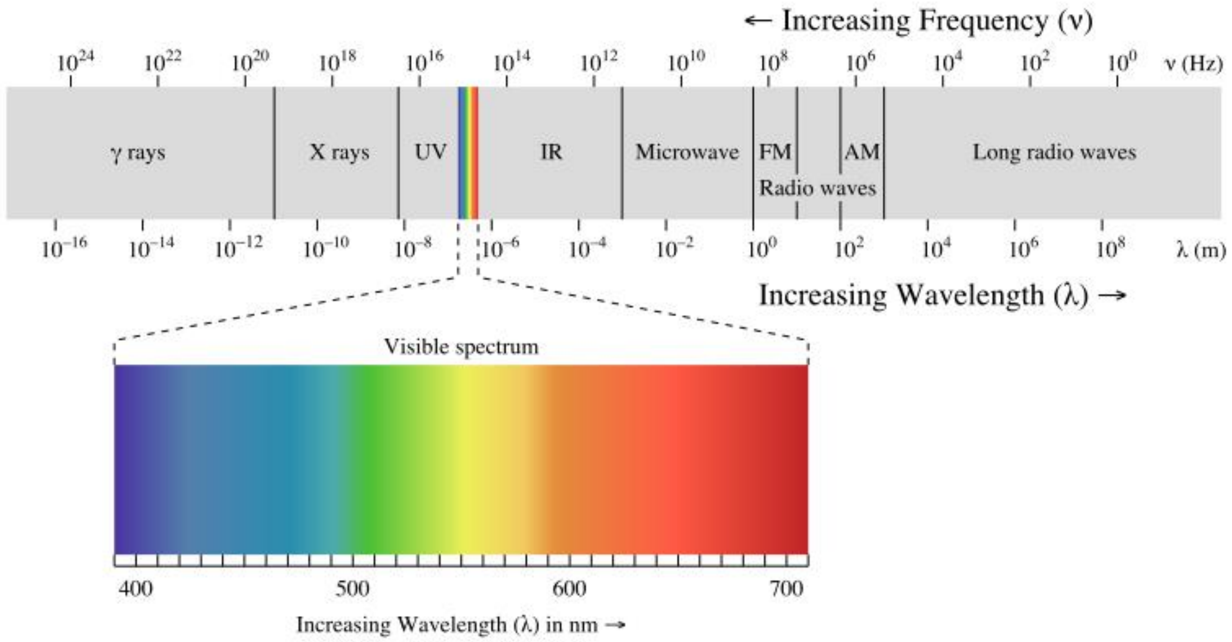
1x1 : Look-up-table
1x1 x time : Image average
NxM : Convolution

If the transformation H is a convolution (to be explained later), it is applied very efficiently with a point-by-point multiplication

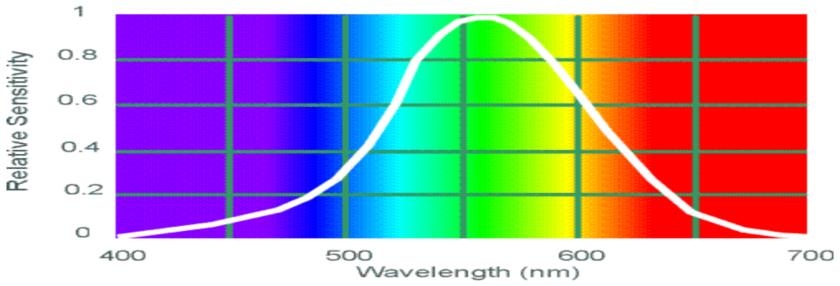
2. IP tools and concepts

Image color

Electromagnetic spectrum



All these wavelengths are produced by some light source (sun or artificial)



Human Luminance Sensitivity Function

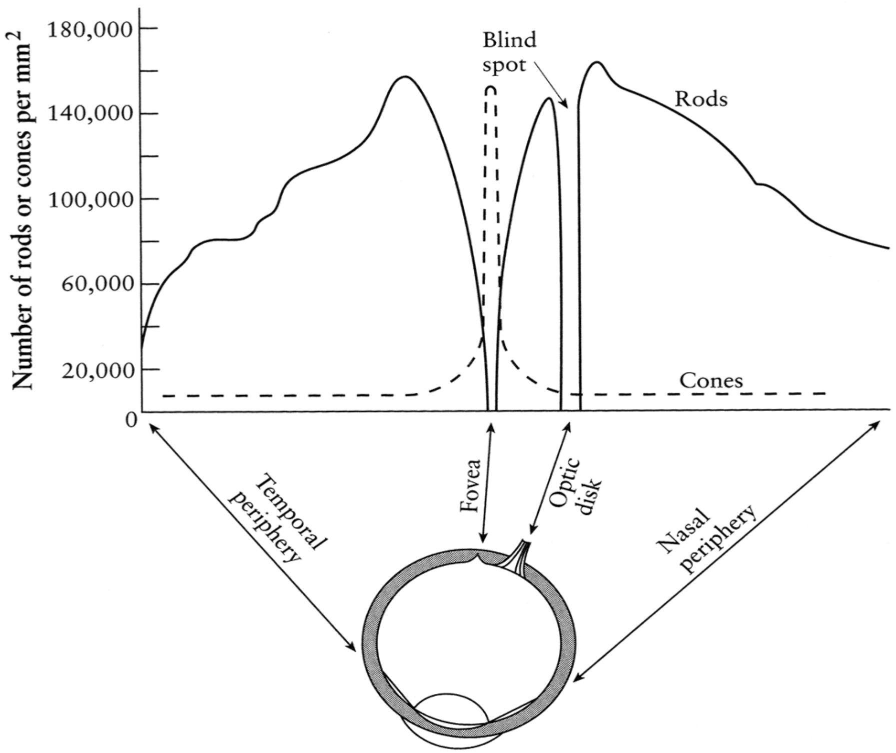
We see these wavelengths because the receptors (cones) in our eyes

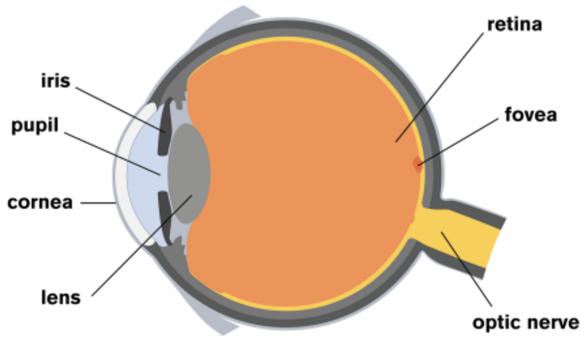
Image color

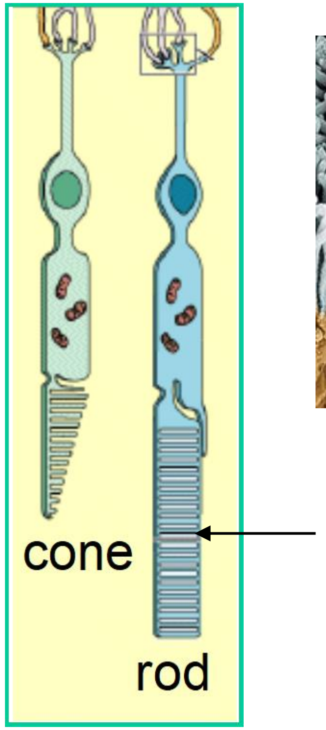
Human eye

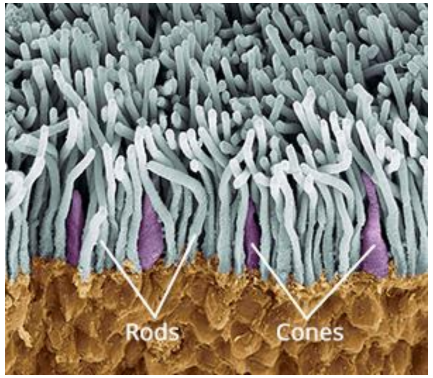
- Two types of photoreceptors in the human retina: Rods responsible for intensity, cones responsible for color
- Fovea - Small region (1 or 2°) at the center of the visual field containing the highest density of cones (and no rods).

Rods and cones are *non-uniformly* distributed on the retina (Less visual acuity in the periphery)





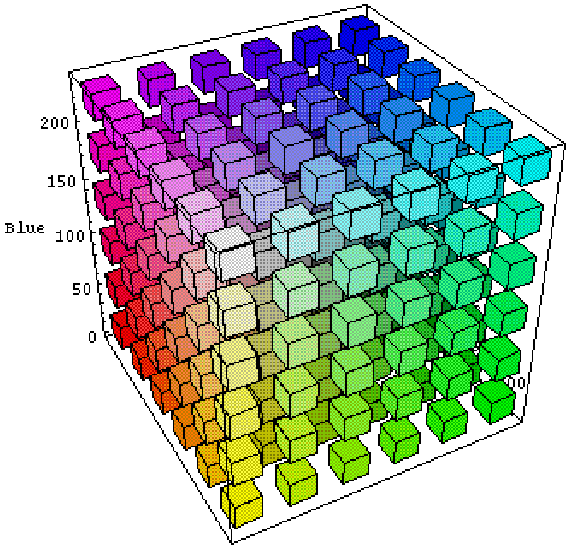


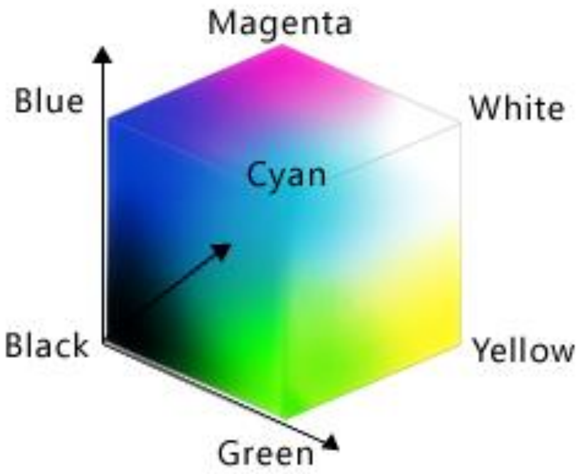


Slide by Steve Seitz

Image color space

RGB: Linear color space used by computers





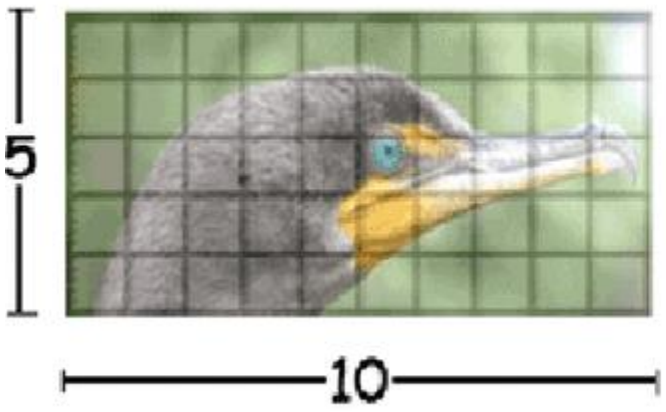
Any color can be computed as the linear combination of the three components RGB

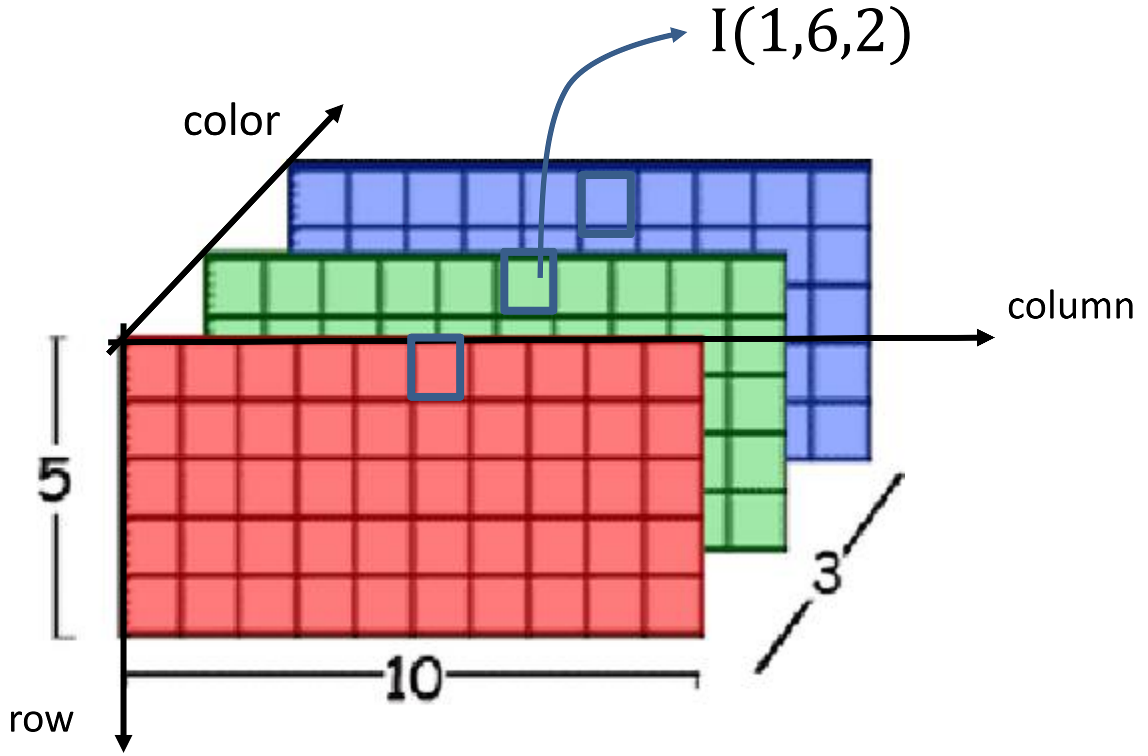
When a color has the same value of the three components RGB (diagonal of the RGB trihedron) we have grey-levels.

R=G=B → grey level (or gray scale)

Image color

A color image





Mathematically, a color image is a tri-dimensional array: I(*row,column,color*) with $color \in \{R, G, B\}$

Easiest way to convert from RGB color to grey-level (grayscale):

Average method: Grayscale = (R + G + B / 3) → 33% R, 33% G, 33% B

Image color: Chromaticity

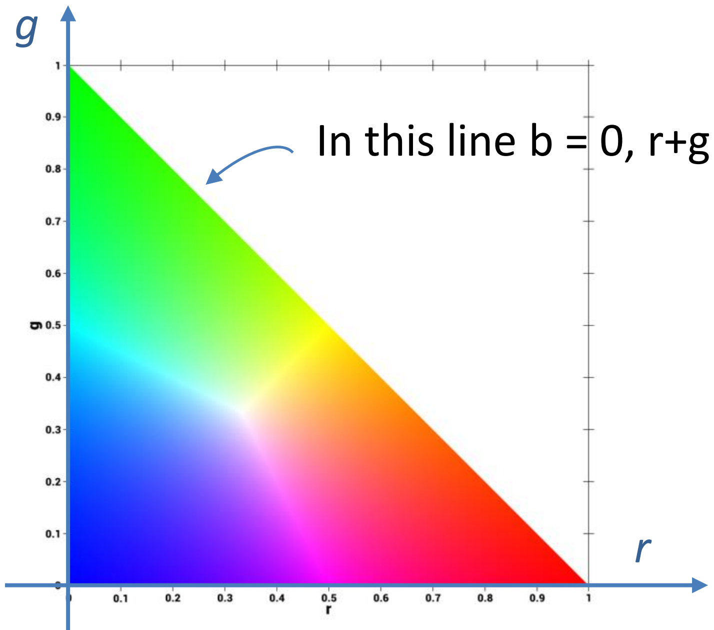
A color (R,G,B) can be converted to normalized colors (*r,g,b*) which gives us the proportion of red, green and blue:

$$r = \frac{R}{R+G+B} \quad g = \frac{G}{R+G+B} \quad b = \frac{B}{R+G+B}$$

$$r + g + b = 1$$

One constraint

Chromaticity chart is a 2D space, for example, <*r,g*>



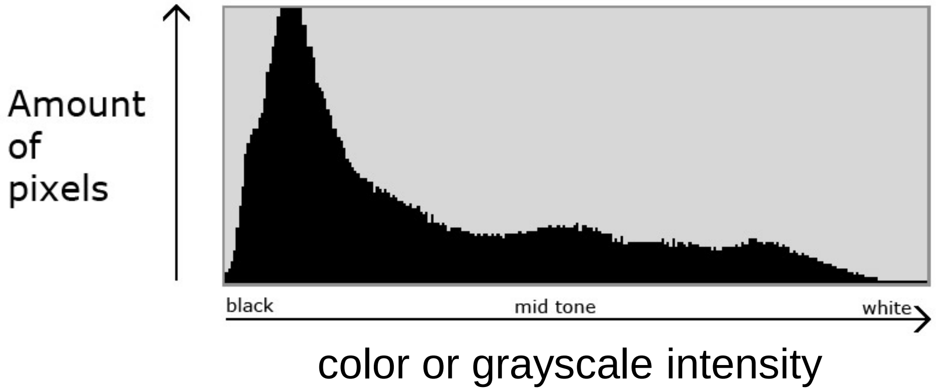
In the chromaticity chart, the missing component (b) is the value of the pixel.
For example: each 45 deg. line (e.g. r+g = 0.8 → b = 0.2.

This representation is useful to segment regions based on their color

2. IP tools and concepts

Image Histogram

- Is a representation of the frequency each color intensity appears in the image
- Built by counting the ocurrence of each color in the image
- A color image has 3 histograms (e.g. R, G, B)
- Provides statistical information of the intensity distribution (e.g. brightness and contrast)



Brightness and contrast (moments of a histogram)

Moment k of a function *f(x)*: $m_k = \int_{-\infty}^{\infty} x^k f(x)dx$

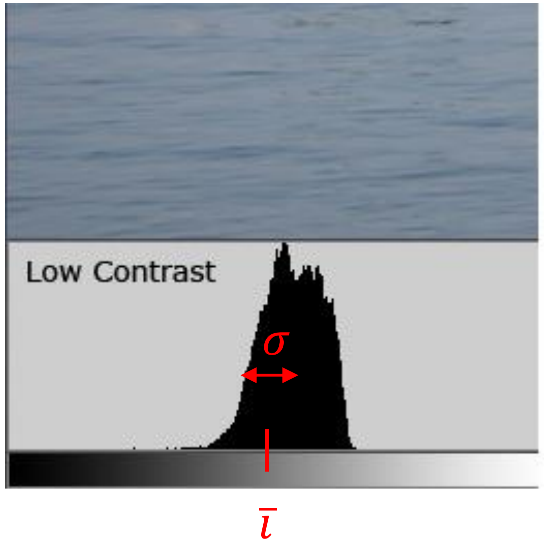
Central moment k of a function *f(x)*: $\mu_k = \int_{-\infty}^{\infty} (x - m_1)^k f(x)dx$

Brightness: **Average of the pixel intensities** = One order moment of the histogram ($h(i)$)

$$m_1 = \bar{\imath} = \sum_{i=0}^{255} i^1\, h(i)$$

Contrast: Standard deviation of the pixel intensities. Square root of the second central moment of the histogram

$$\sigma = \sqrt{\sum_{i=0}^{255} (i - \bar{\imath})^2\, h(i)}$$



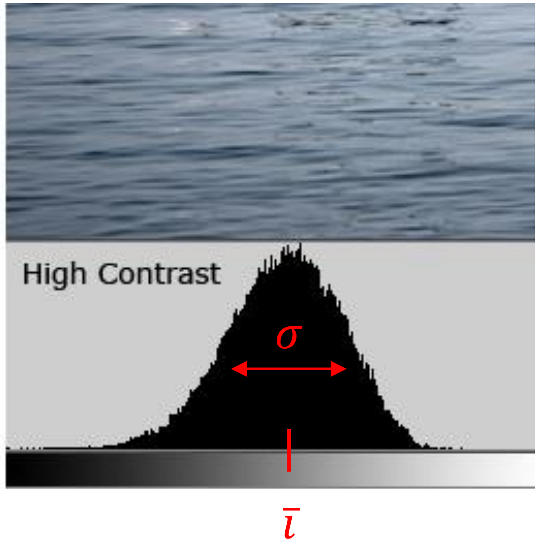
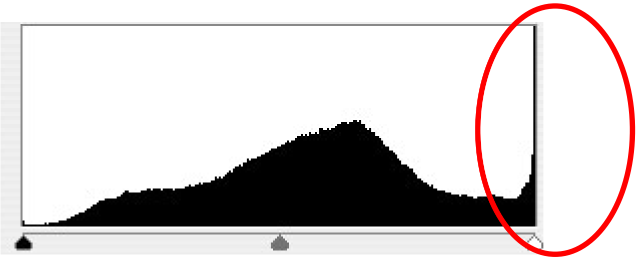
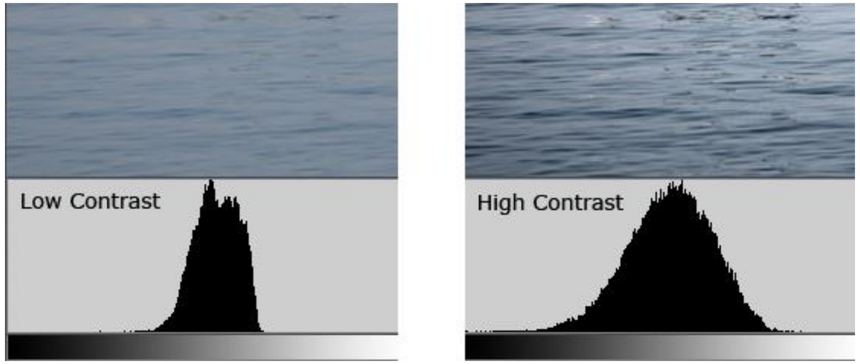


Image histogram is useful for:

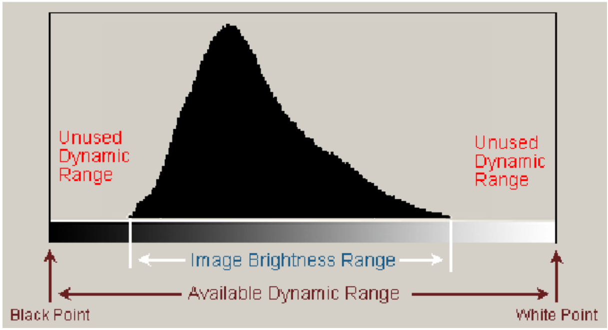
- Detecting Black or White saturation



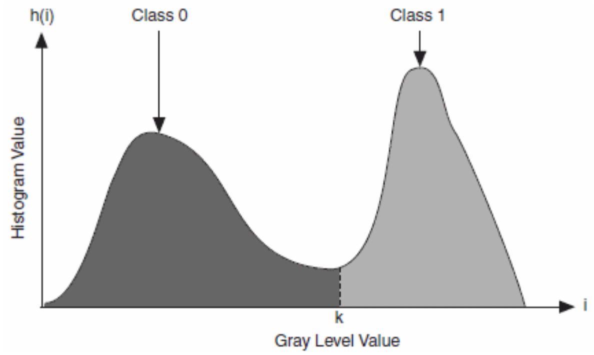
- Contrast and brightness



- Use of the whole intensity range (i.e. the 256 values)

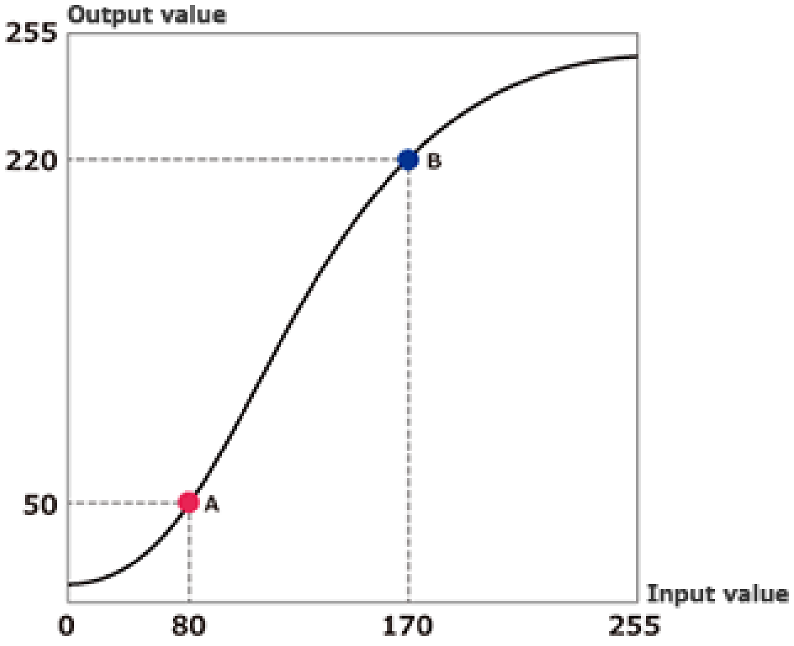


- Select a suitable threshold for image binarization



Look-up Tables (LUTs)

A LUT replaces an index intensity (input) by another intensity value stored in the table (vector)



Continuous LUT

256 values

Input value	Output value
0	5
1	6
2	8
3	10
79	47
80	50
81	53
169	218
170	220
171	222
252	248
253	250
254	252
255	254

A
B

All the pixels with intensities 80 in the image are replaced by the value 50

Discrete LUT (a vector)

Example: Processed image

- does not change the geometry of objects in the image
- only changes the histogram of the image

Look-up Tables (LUTs)

RGB LUT

LUT for R: Input value	LUT for R: Output value	LUT for G: Input value	LUT for G: Output value	LUT for B: Input value	LUT for B: Output value
0	5	0	10	0	0
1	6	1	12	1	0
2	8	2	13	2	1
3	10	3	15	3	1
79	47	79	72	79	25
80	50	80	75	80	25
81	53	81	77	81	26
252	242	252	251	252	234
253	245	253	252	253	236
254	248	254	253	254	238
255	251	255	255	255	240

256 values

3 sets

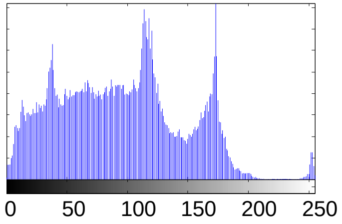
c

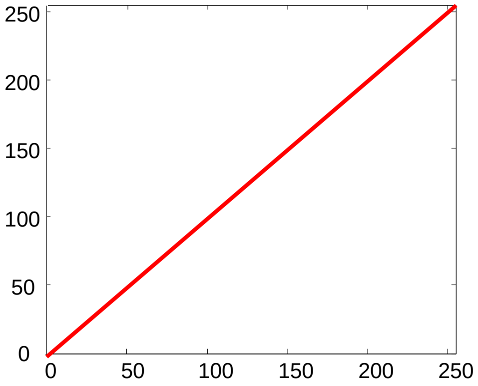
Applied to an image it is like replacing the original color palette by another one

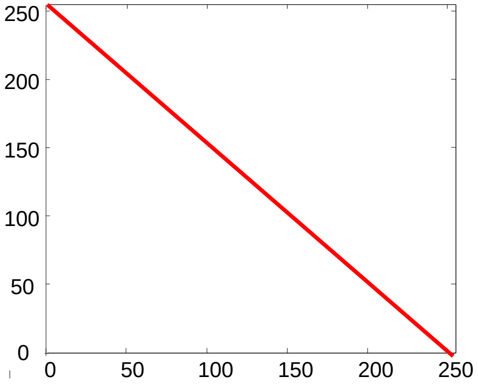
Look-up Tables (LUTs)

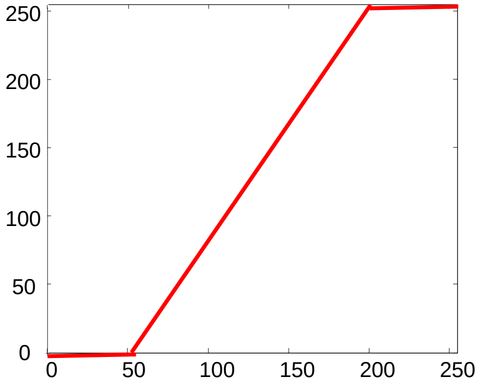
Examples

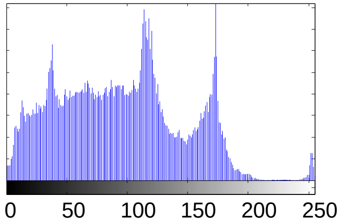
Original image

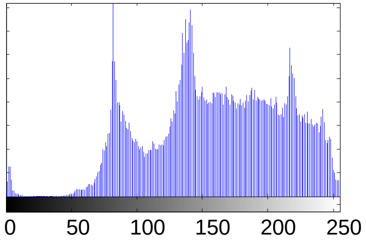


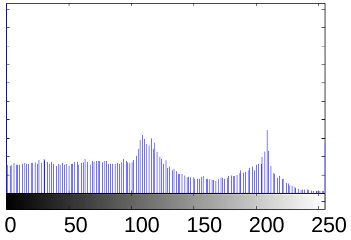






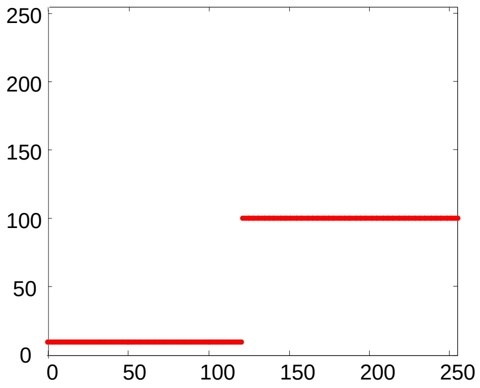


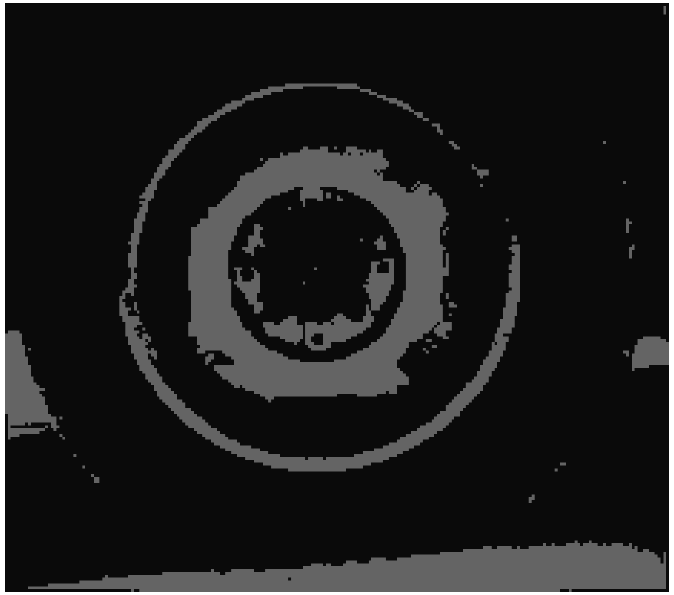


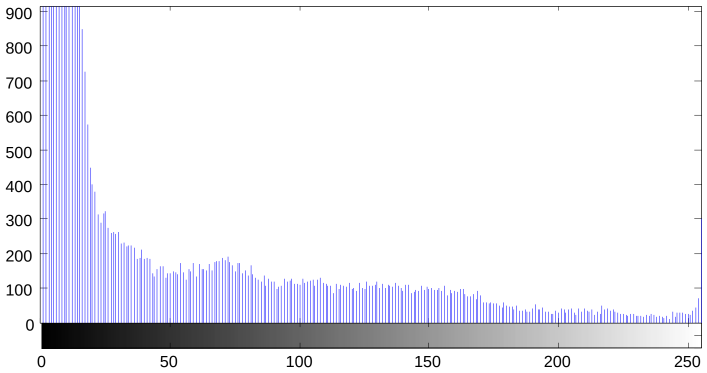


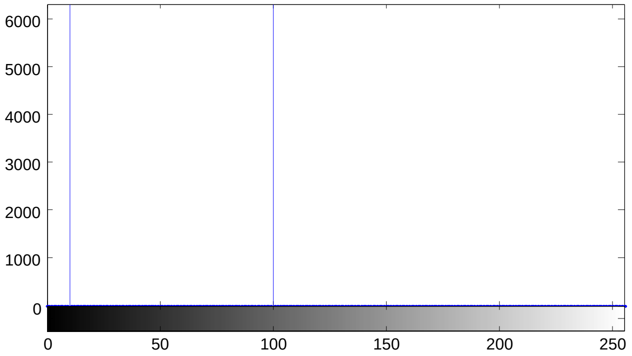
Look-up Tables (LUTs)

Implementation (example binarization)









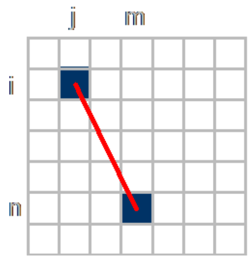
Distance between two pixels p_1,p_2 :

$D(p_1,p_2)$ is a distance function if:

1. $D(p_1,p_2) \geq 0$ [$D(p_1,p_2)=0$ si $p_1=p_2$]
2. $D(p_1,p_2) = D(p_2,p_1)$
3. $D(p_1,p_3) \leq D(p_1,p_2)+D(p_2,p_3)$

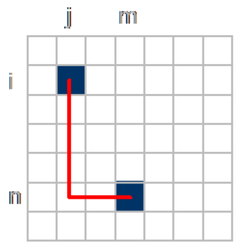
- *Euclidean distance (L^2-norm)* between p_1 y p_2 :

$$D_e(p_1,p_2) = \left[(x_1 - x_2)^2 + (y_1 - y_2)^2\right]^{1/2}$$



- *Manhattan distance (l_1-Norm)* between p_1 y p_2 :

$$D_4(p_1,p_2) = |x_1 - x_2| + |y_1 - y_2|$$

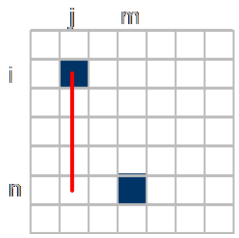


		2		
	2	1	2	
2	1	0	1	2
	2	1	2	
		2		

D_4

- *8-Distance (∞-Norm)* between p_1 y p_2 :

$$D_8(p_1,p_2) = \max(|x_1 - x_2|, |y_1 - y_2|)$$



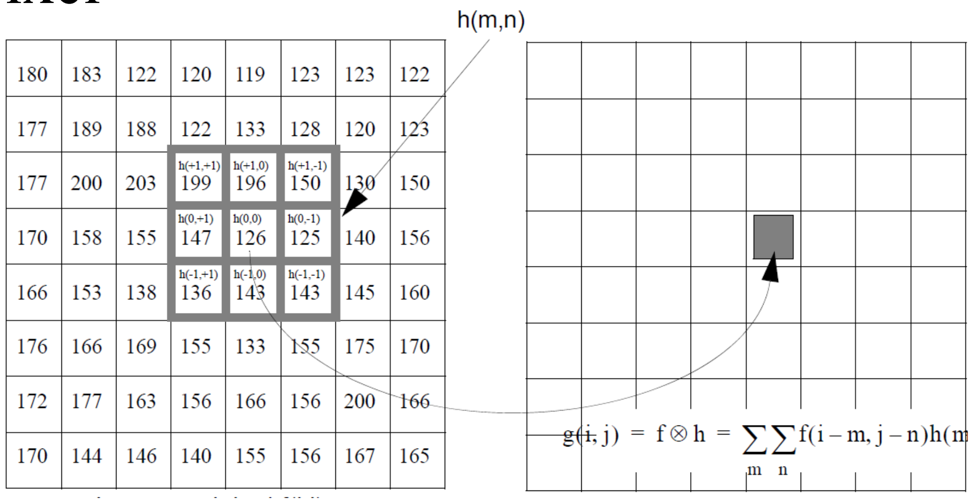
2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

D_8

Image Convolution

Weighted average of the neighborhood of each pixel

$$g(i,j) = \mathbf{f} \otimes \mathbf{h} = \sum_m \sum_n f(i-m, j-n)\,h(m,n)$$



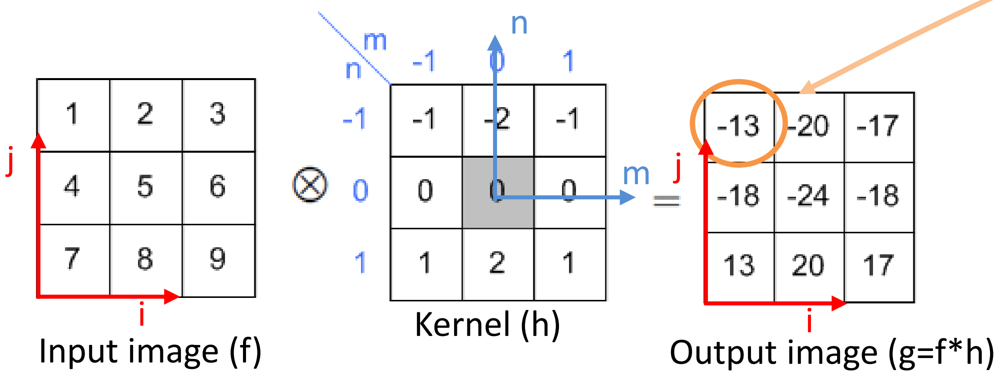
The value **g** at (i,j) is the weighted average of **f** at (i,j) with the values of **h**

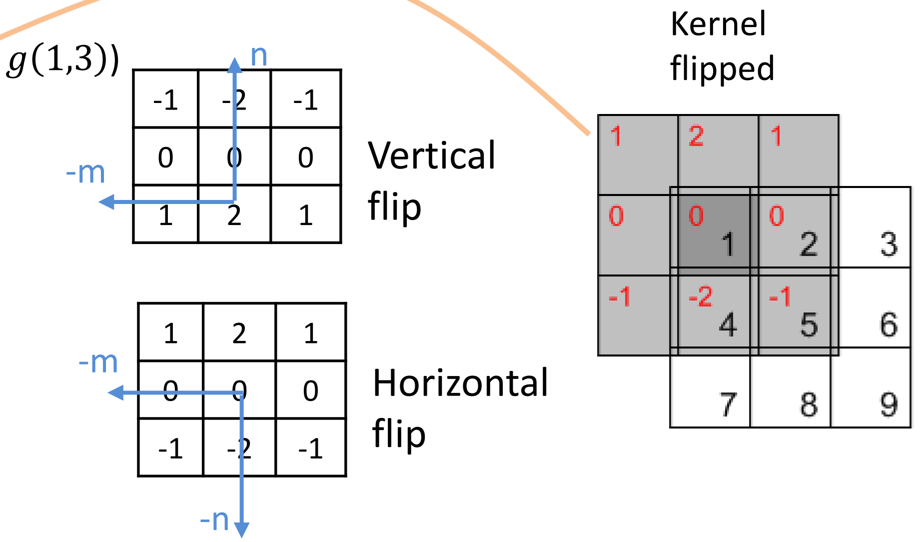
Equivalent to multiply the coordinates (n,m) by:

The kernel is flipped in both axes

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} n \\ m \end{bmatrix} = \begin{bmatrix} -n \\ -m \end{bmatrix}$$

Example:





$$\begin{aligned} g(1,3) = {} & f(0,2) + f(0,1) + f(0,0) \\ & + f(1,2) + f(1,1) + f(1,0) \\ & + f(2,2) + f(2,1) + f(0,0) \end{aligned}$$

Image Convolution

How it works (visually):

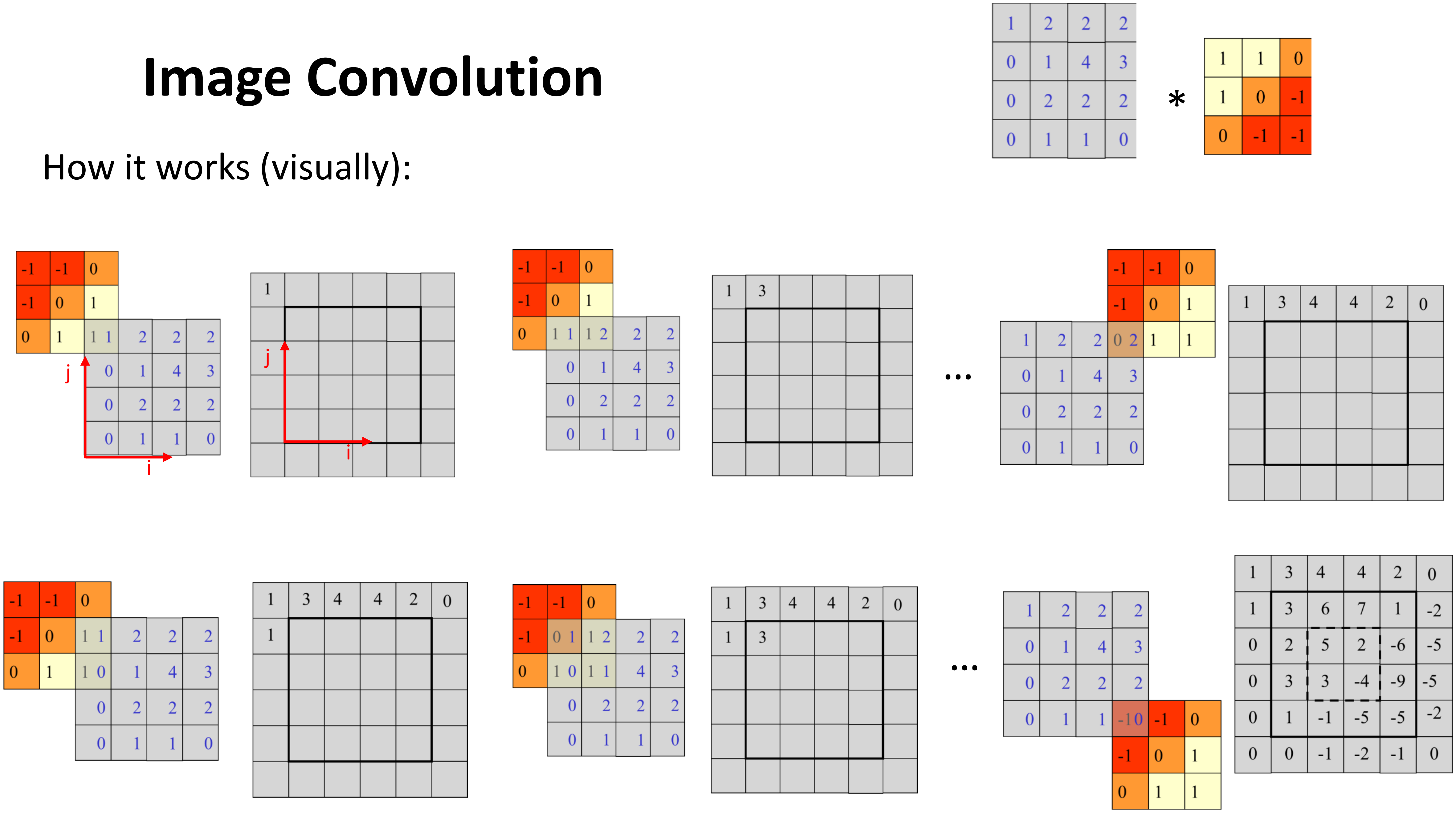


Image Convolution

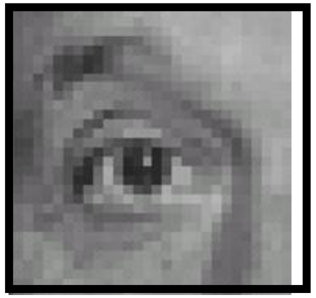
Original | Kernel | Output

Examples:

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

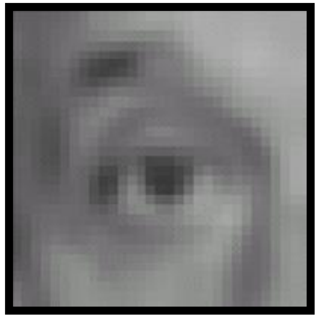
Same

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



Shifted 1 pixel to the left

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



Blurred: equal average of the 8-neighbors

Source: D. Lowe

Convolution properties

Conmutative: $f * g = g * f$

Associative: $f * (g * h) = (f * g) * h$

Distributive: $f * (g + h) = (f * g) + (f * h)$

Asociative wrt scalar product: $a(f * g) = (af) * g = f * (ag)$

Derivative: $\mathcal{D}(f * g) = \mathcal{D}f * g = f * \mathcal{D}g$

Theorem of convolution: $\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$

3. Smoothing

Objective: Eliminate/reduce noise in image (in pixel intensities).

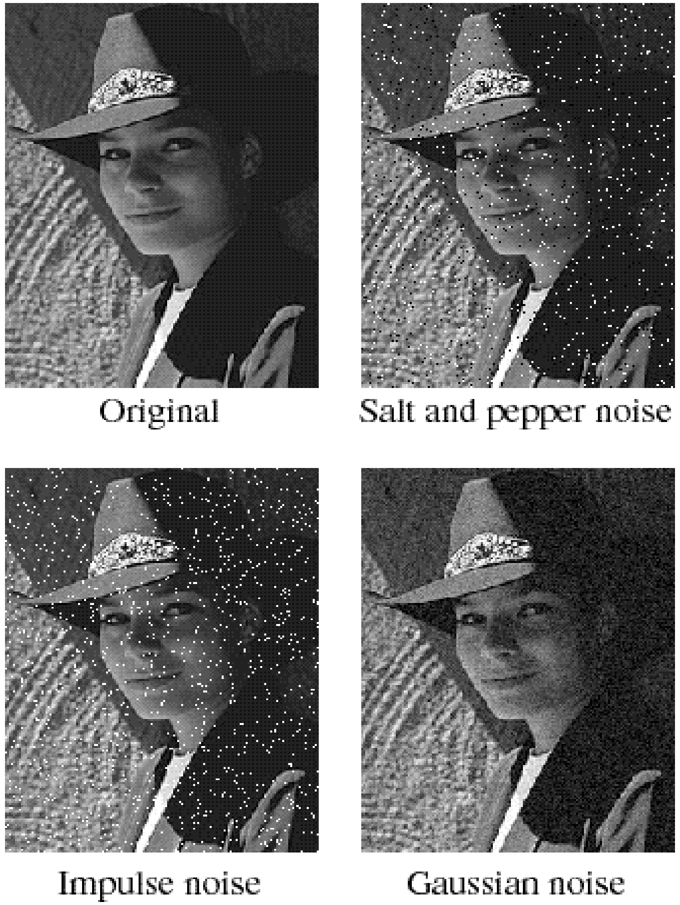
Noise appears because of the sensor response (more in CMOS technology), digitalization and transmition.

Approaches:

- Neighborhood averaging
- Gaussian filter
- Median filter
- Image average
- Filters in the frequency domain (not addressed in this course)

3. Smoothing

Objective: Eliminate/reduce noise in pixel intensities



Fuente: S. Seitz

Salt and pepper noise:

black and white pixels appears at random

Impulse noise:

black or white pixels appears at random

Gaussian noise:

- intensities are affected by an additive zero-mean **Gaussian error.**
- simulates thermal noise in sensors, which shows up in **poor illumination conditions**

3. Smoothing

Neighborhood averaging

The value of each pixel is substituted by the average value of the neighbors

$$g(x,y) = \frac{1}{p} \sum_{(m,n) \in s} f(m,n)$$

S : set of “p” pixels in the neighborhood (mxn) of (x,y)

Implemented with the convolution with a kernel (linear operation)

For example, for a 3x3 neighborhood

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Main drawback: edge blur

3. Smoothing

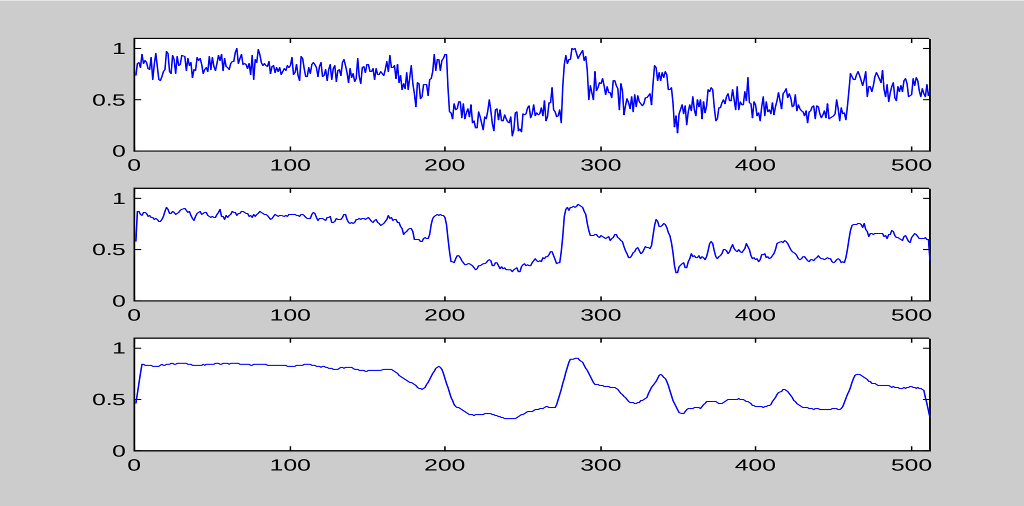
Gaussian noise reduction by Neighborhood averaging

row

(a) Original image

(b) Noisy image

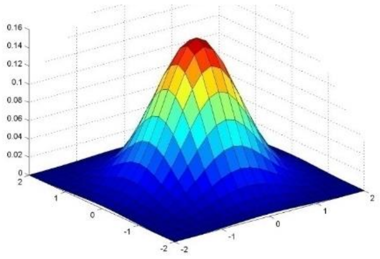
Intensity profile for the row marked in the original image

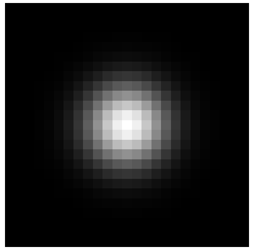


Gaussian filter

- is a neighborhood weighted averaging
- weights from a Gaussian bell

$$g_\sigma(x,y)=\frac{1}{2\sigma^2\pi}e^{-\frac{1}{2}\left(\frac{x^2+y^2}{\sigma^2}\right)}=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2}\cdot\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}\left(\frac{y}{\sigma}\right)^2}$$





Separability property (rows and columns separated)

- Implemented with the convolution of the image with a kernel.

  For example 5x5:

0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

Gaussian filter

Separability property (rows and columns separated)

Convolution of two 1D kernels

$$\underbrace{g_\sigma(x,y) = \frac{1}{2\sigma^2\pi} e^{-\frac{1}{2}\left(\frac{x^2+y^2}{\sigma^2}\right)}}_{g_{xy}} = \underbrace{\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{y}{\sigma}\right)^2}}_{g_y} \otimes \underbrace{\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2}}_{g_x}$$

Example:

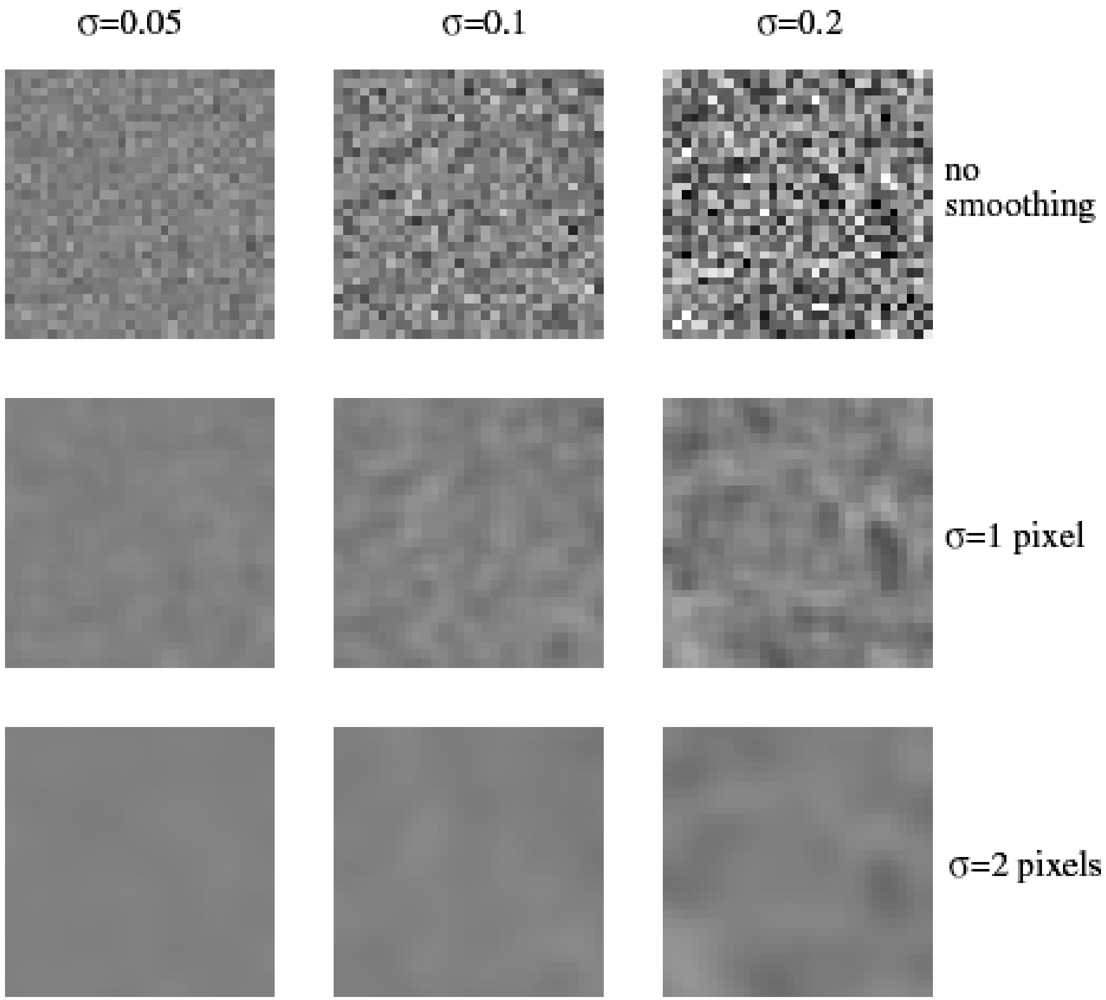
$$\underbrace{\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}}_{g_{xy}} = \underbrace{\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}}_{g_y} \otimes \underbrace{\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}}_{g_x}$$

Because of the **Associative property**:

$$\underbrace{f \otimes g}_{\text{2D convolution}} = f \otimes (g_x \otimes g_y) = \underbrace{(f \otimes g_x) \otimes g_y}_{\text{Two 1D convolutions}}$$

Gaussian filter

σ allows us to control the degree of smoothing



Bigger σ …

- more smoothing
- blurrier image

Size of the kernel (w):

- proportional to σ
- must be big enough to account for non-negligible values in the kernel

0	1	3	4	3	1	0
1	6	18	25	18	6	1
3	18	50	71	60	18	3
4	25	71	$\boxed{100}$	71	25	4
3	18	50	71	60	18	3
1	6	18	25	18	6	1
0	1	3	4	3	1	0

← w →

Another row/column would have very small values (negligible)

3. Smoothing

Smoothing with **Median filter** (not averaging)

- Replace the intensity value of a pixel by the median of its neighborhood

10	11	12
10	50	10
10	12	11

3x3 image neighborhood around the pixel analyzed (intensity 50)

Sorted list of the neighborhood

10	10	10	10	11	11	12	12	50

median

10	11	12
10	11	10
10	12	11

Resulting image

- It is **NOT a linear operation**
- **High computational cost** … but there are efficient implementations (e.g. pseudomedian, sliding median, …)

Median filtering

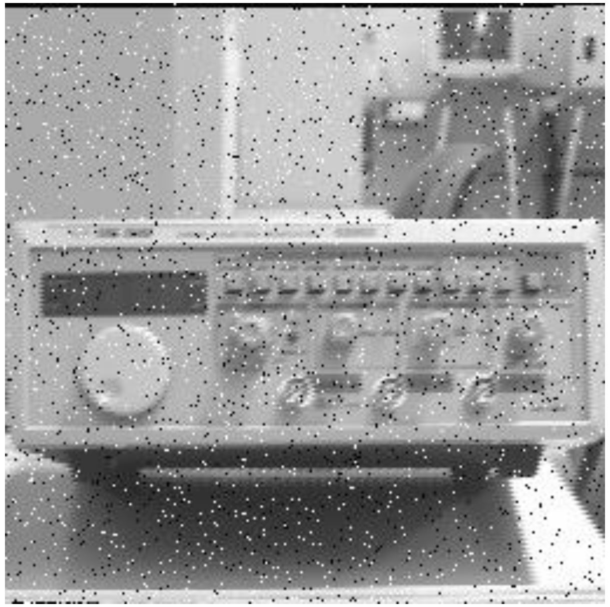


Image corrupted by
“salt & pepper” noise

Median

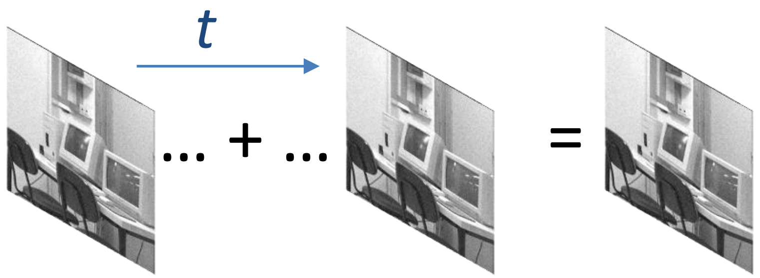
3x3 average

- Preserve bordes (**no image blur**)
- Very effective to remove **salt&pepper noise**

3. Smoothing

Image averaging

IDEA: To average several (M) images along a sequence of a static scene (assumption: the only difference is the noise, i.e. static scene)



$$g(x,y) = \frac{1}{M}\sum_{i=1}^{M} f_i(x,y) = \frac{1}{M}\sum_{i=1}^{M}\left[f_{noise\_free}(x,y) + \eta_i(x,y)\right] = f_{noise\_free}(x,y) + \frac{1}{M}\sum_{i=1}^{M}\eta_i(x,y)$$

Average image

Noise Image

If the noise $\eta_i(x,y)$ has zero-mean $E[\eta_i(x,y)] = 0$: The expected value of noise is zero

$$E[g(x,y)] = E\left[f_{noise\_free}(x,y) + \frac{1}{M}\sum_{i=1}^{M}\eta_i(x,y)\right]$$

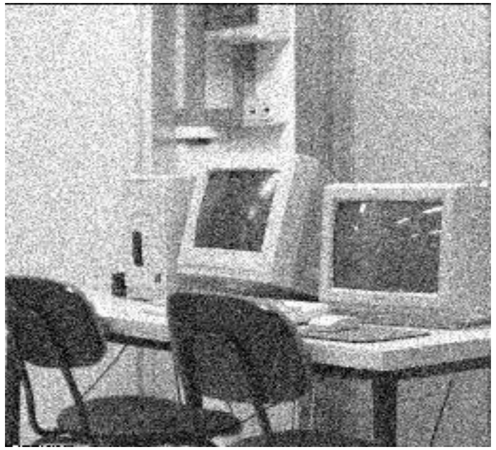
$$= f_{noise\_free}(x,y) + \frac{1}{M}\sum_{i=1}^{M} E[\eta_i(x,y)] = f_{noise\_free}(x,y)$$

The expected value is the noise-free image!

3. Smoothing

Image averaging

Gaussian noise



1 image

10 images

50 images

Salt&Pepper noise

3. Smoothing

Image averaging

Advantage:

- Preserves edges
- Very effective with Gaussian noise (not with salt&pepper)

Drawback:

Only applicable to sequence of images of a still scene

4. Image Enhancement

Objective: Improve contrast and brightness of the image

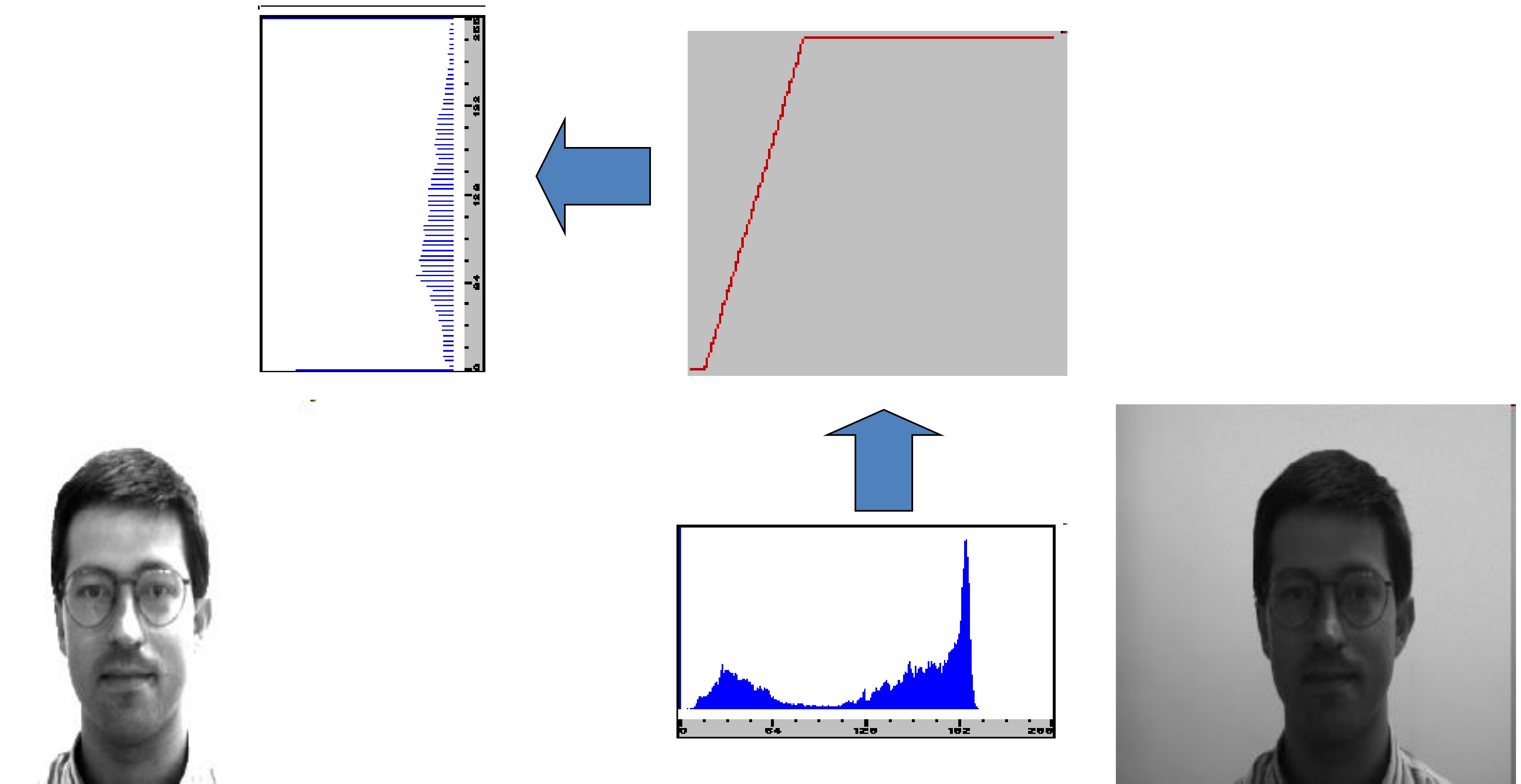
In Computer Vision this is required to prepare the image for subsequence operations (feature extraction, segmentation)

Typical operations:

- Lookup-table transformation
- Histogram equalization
- Histogram specification

Lookup-table transformation

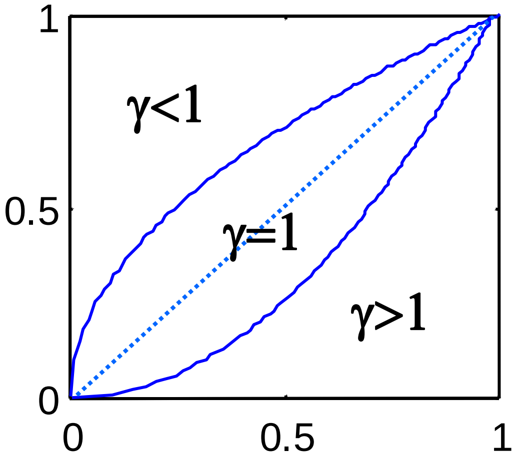
<u>IDEA:</u> To stretch and shift the histogram for a better leverage of the intensity range [typically, 0-255]



Lookup-table transformation

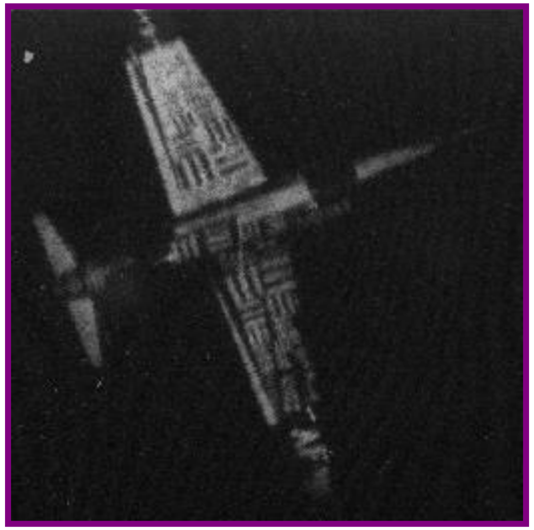
Non-linear transformation: $g(x,y) = f(x,y)^{\gamma}$

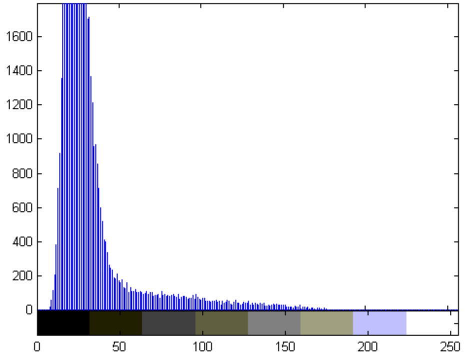
- $\gamma < 1$ to increase contrast of darker pixels (Stretching of low intensities)
- $\gamma > 1$ to increase contrast of brighter pixels (Stretching of high intensities)

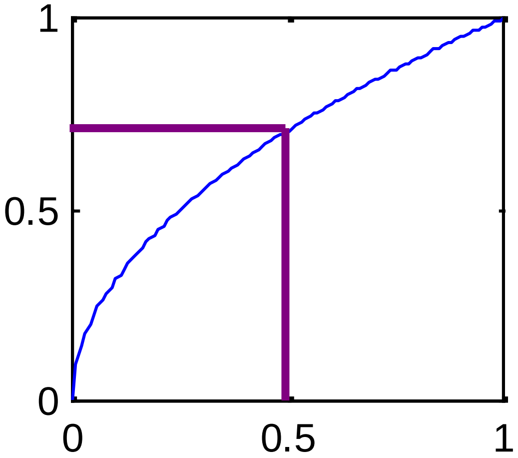


Lookup-tables transformation

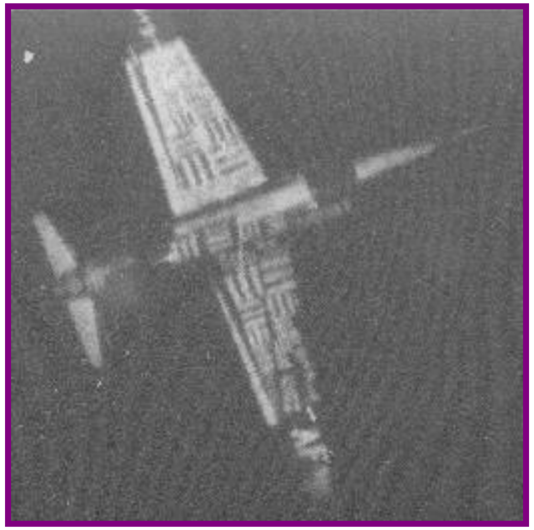
Non-linear transformation: $g(x,y) = f(x,y)^{\gamma}$



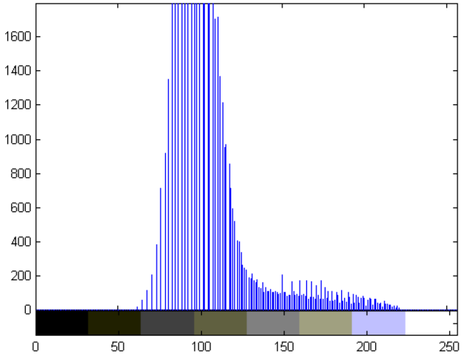




Stretching of low intensities



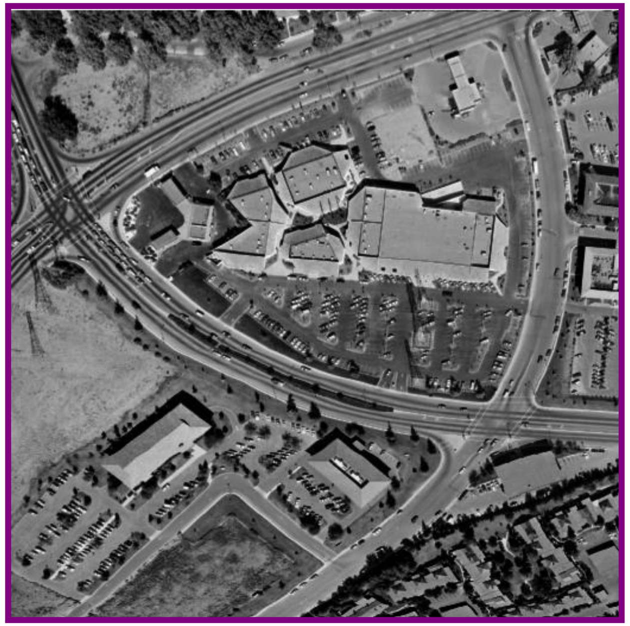
$\gamma = 0.5$ (root square)



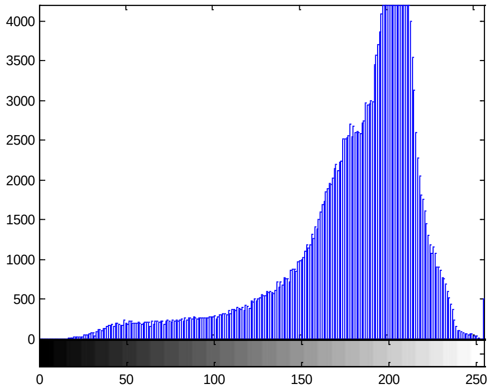
Courtesy of [Zhigang Zhu]

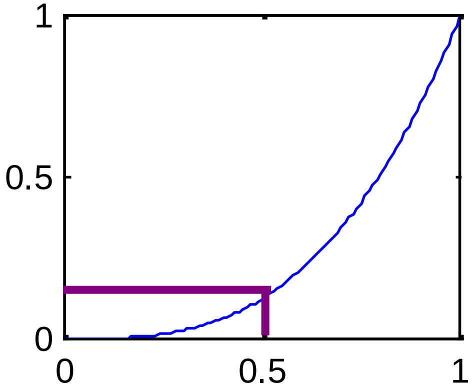
Lookup-table transformation

Non-linear transformation: $g(x,y) = f(x,y)^{\gamma}$

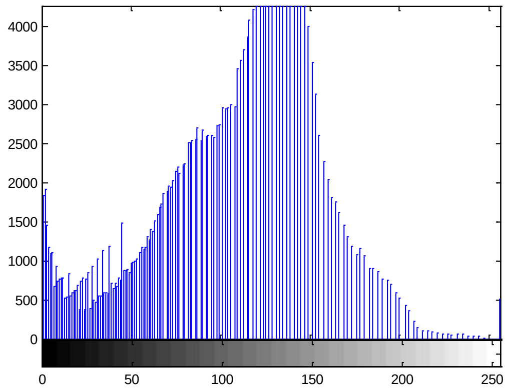


$\gamma = 3$





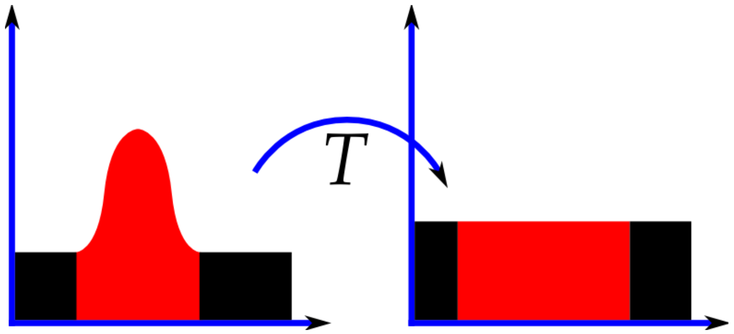
Shrinking of low intensities



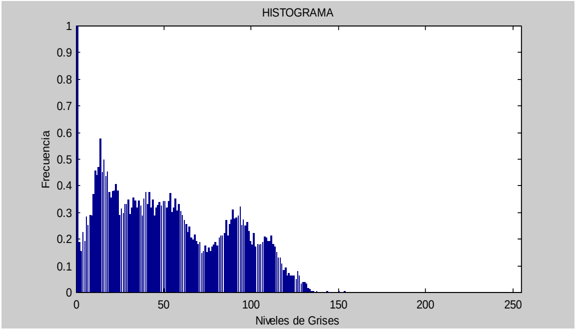
4. Image Enhancement

Histogram equalization

IDEA: Modify the intensities such that the new image achieves an uniform histogram



May be applied when low contrast and low brightness (dark) image

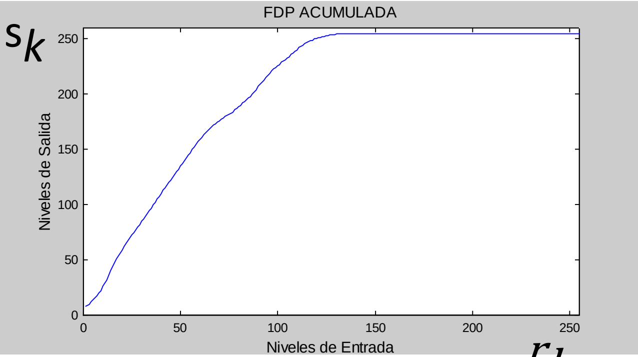


Histogram equalization

Original image

$\rightarrow$

s_k



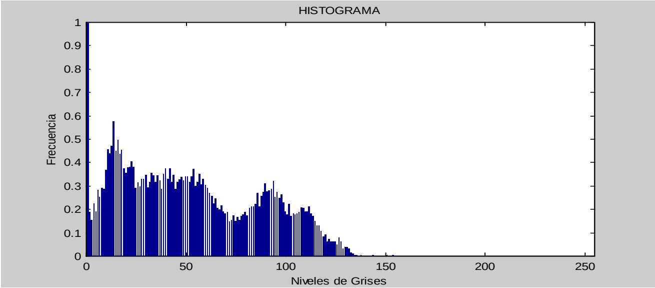
r_k

LUT

$\rightarrow$

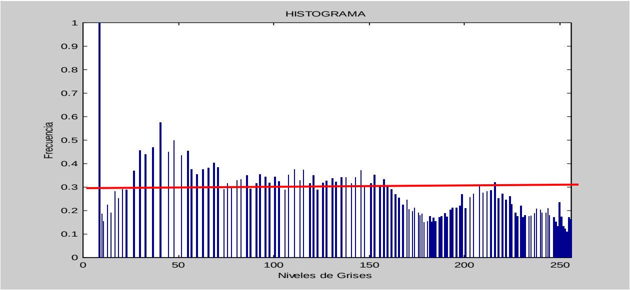
Equalized image

$h(r_k)$



Histogram of the original image r_k

$$s_k = T(r_k) = \sum_{i=0}^{k} h(r_k)$$

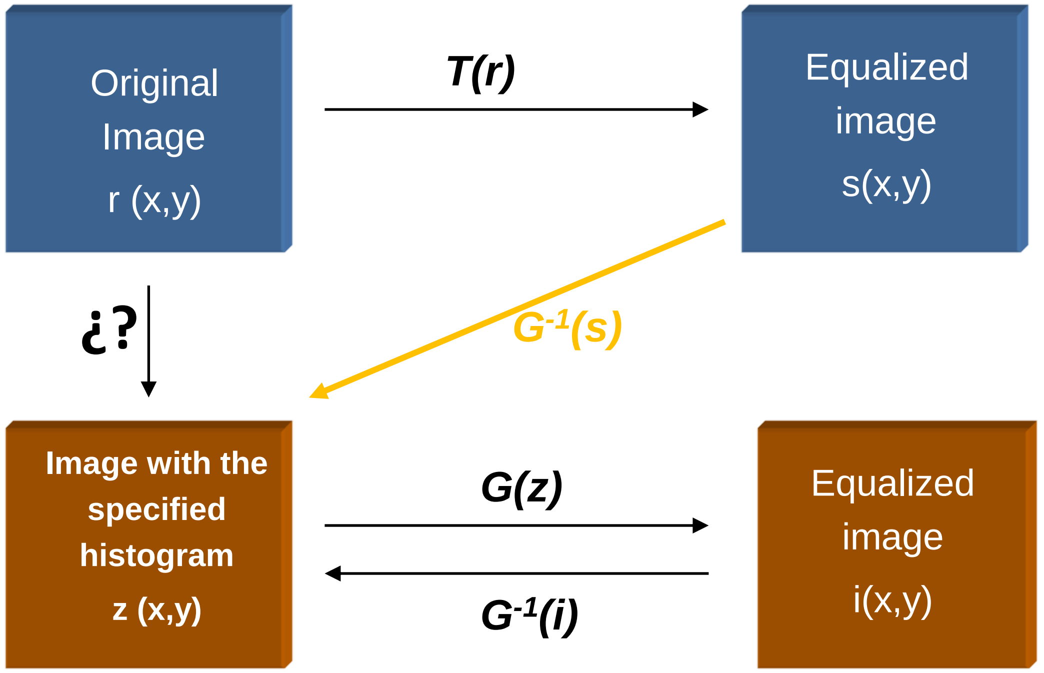


Histogram of the equalized image

In practice, not fully uniform!

Histogram specification (histogram matching)

- transformation of an image so that its histogram matches a specified histogram
- histogram equalization is a special case in which the specified histogram is uniformly distributed

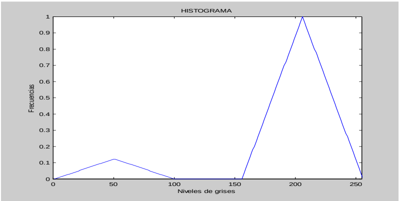


Histogram specification

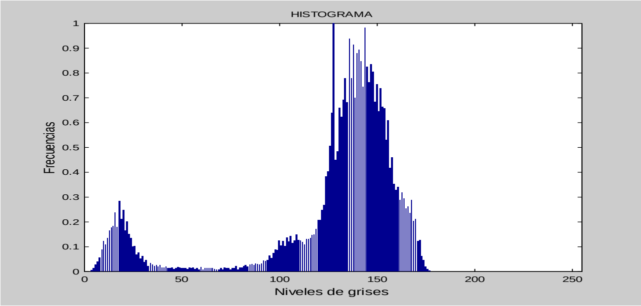
Example:

Original Image

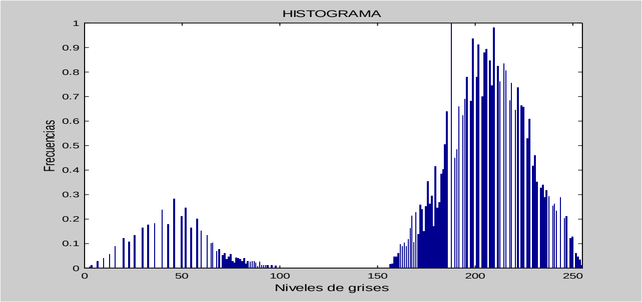
Specified histogram



Enhanced Image



Histogram of the original image



Histogram of the enhaced image

Summary

- IP operations needed to improve the quality of the image by attenuating noise, adjusting colors, modifing contrast and brigthness, …
- Necessary to **prepare the image** for a better edge detection or segmentation
- A **color image** is a Tri-dimensional array: I(x,y,c) with $c \in \{R, G, B\}$
- A **histogram** provides statistical information of the intensity distribution (e.g. brightness and contrast)
- **LUT transformations** very efficient and effective to change contrast and brightness
- **Convolution** is a linear operation between two images, very useful for IP
- **Gaussian Noise** in images is attenuated by convolving with a Gaussian kernel
- **Salt&Pepper Noise** is attenuated by a median (non-linear) filter