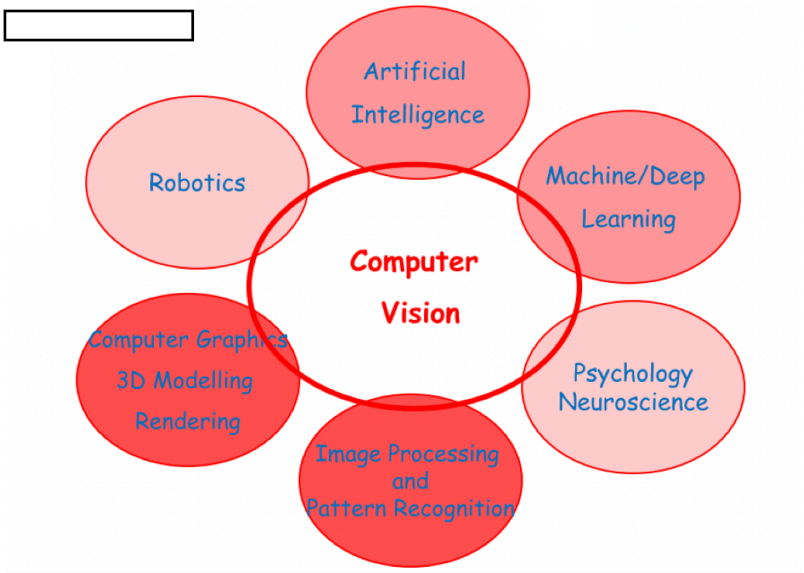
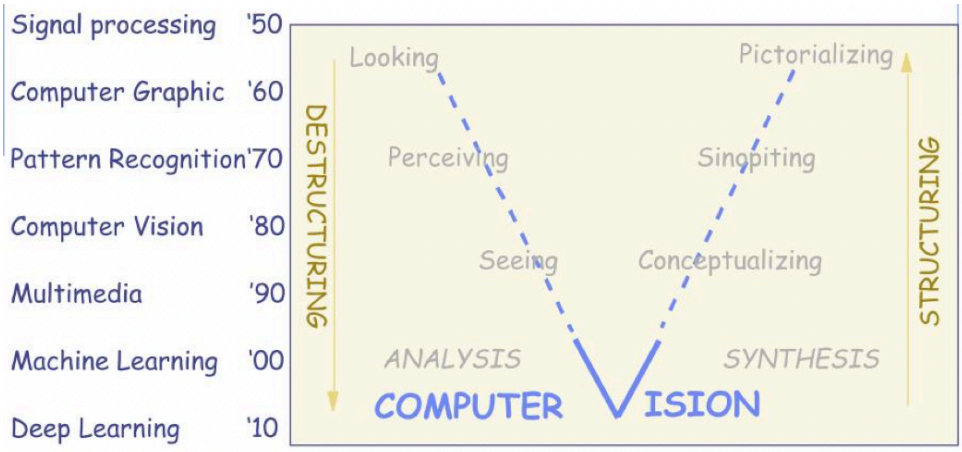
Computer Vision

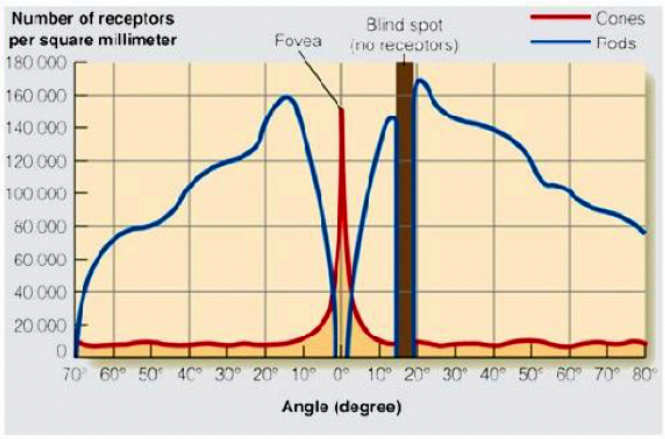
The task is to study the method by which a system can understand the environment where it's inserted.



Two inverse problems: analysis and synthesis.
Computer Vision produces models for rendering, surface design, animation and user-interfaces thanks to the manipulation and detection of images.



Density distribution of rod and cone receptors



Loss of resolution with eccentricity in retina

Human perception (problems)

Simultaneous contrast: the little squares are the same color but it doesn't seem.

Optical illusions: the eyes receive a false information because of the illusions.

Spatial resolution

The spatial resolution is related to the dimension of the details that can be detected. Quality of the information: the ability to detect details.
The **resolution cell** or **elementary cell** is the smallest area with an associated value in a digital image.
The pixel corresponds to the elementary cell.

Color depth

Number of bits used to describe the information related to each pixel.
Binary image: each pixel can have only two values (0,1 or object, background)
Gray image: larger ranges [0, 63] or [0, 256] or [0, 1023] (6,8,10 pixels)
Human beings can deal with 8 bits for pixel

Color images

Usually memorize **3 values** for each pixel (red, green, blue)
Each pixel usually use **1 byte** (8 bits) so we can have 256x256x256 different color (~16 millions)
The green channel is the most considered.

Computer Vision

COLOR MODELS

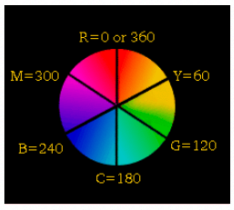
RGB: used for monitors, the sum of the component leads to the final color. (White is the result of the three components at the maximum level)
CMY(K): cyan, magenta, yellow, black. Used for printers for costs and paper issues.
(Black is the result of the components at maximum levels)

The colors of a monitor are not the same of printable colors. There are particular areas of the spectrum that cannot be represented by RGB or CMYK, even the two models represent different portion of colors.

Other models (hue=tinta):
YIQ – luminance, inphase, quadrature – old TVs color
HIS – hue, saturation, intensity
HSV – hue, saturation, value
HSB – hue, saturation, brightness

- 0°: 255, 0, 0
- 60°: 255, 255, 0
- 120°: 0, 255, 0
- 180°: 0, 255, 255
- 240°: 0, 0, 255
- 300°: 255, 0, 255

HSV



Color images
A possible choice to limit the memory is use a reduced number of colors for each pixel (the bits per pixels are 8, 4, 1).
So also a color LUT (look up table) is memorized.

Original image

256 colors

16 colors

8 colors

Every **BMP image** has this File structure: **Header** + **LUT** + **Pixel values**

PGM (portable gray map) images
File structure:
An ASCII Header (human readable):
«P2» (magic number)
width height

Maximum pixel value (usually 255)
An arbitrary number of comment lines may be present (starting with '#')
Image data: 1 human readable number for each pixel

NB: Il **magic number**, in genere, è una sequenza di bit, normalmente posta prima della sequenza di dati, che serve per definire il formato in cui i dati sono memorizzati.

PGM (portable gray map) images
File structure:
An ASCII Header (human readable):
«P5» (magic number)
width height

Maximum pixel value (usually 255)
An arbitrary number of comment lines may be present (starting with '#')
Image data: 1 byte for each pixel

PPM (portable pixel map) images
File structure:
An ASCII Header (human readable):
«P3» (magic number)
width height

Maximum pixel value (usually 255)
An arbitrary number of comment lines may be present (starting with '#')
Image data: 3 human readable numbers each pixel

PPM (portable pixel map) images

File structure:
An ASCII Header (human readable):
«P6» (magic number)
width height

Maximum pixel value (usually 255)
An arbitrary number of comment lines may be present (starting with '#')
Image data: 3 bytes each pixel (RGB values)

GIF images

File structure:
«GIF89a» (magic number)
A Header (width, height, number of colors): Color LUT

Compressed image data (memorized using less numbers than normal)

PPM image: 290 Kb
GIF image: 53 Kb
–> Lossless compression: if the number of colors is less than 256 it is possible to reconstruct the original image data.

JPG images

The image is subdivided in blocks of 16x16 pixels
An analysis in the frequency domain is done and high frequency components are eliminated (humans do not well recognize them)
For visualization the result is good

PPM image: 290 Kb
JPG image: 25 Kb
–> Lossy compression: it is not possible to reconstruct the original image data
The compression level is a parameter of the transformation process.

ARGB images

Sometimes pixel values are memorize as integer values of 32 bits
In this case it is used a fourth channel (**alpha channel**). It is used to memorize the degree of visibility of the pixel: 0 value corresponds to a transparent pixel, 255 to a opaque pixel
Alpha channel can be used in Java images, in PNG images and in BMP images.

EDGE DETECTION

Goal: Identify sudden changes (discontinuities) in an image.

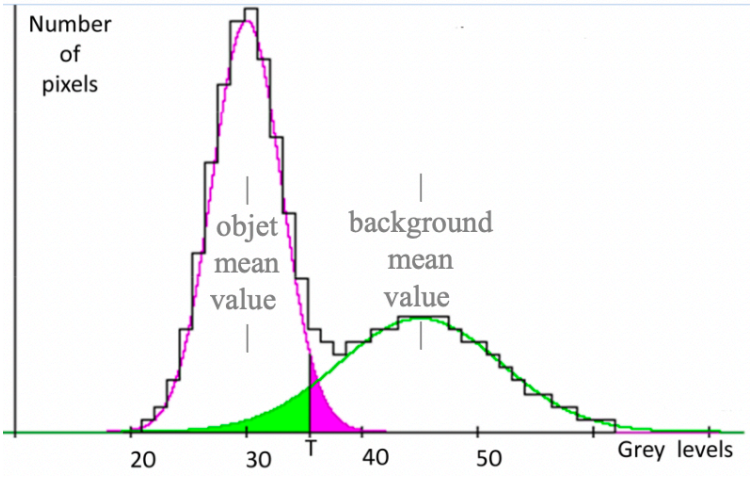
Image segmentation consists into the decomposition of the image in segments (i.e. components) that have homogeneous values for some aspects. This process is based on a given criteria of **homogeneity** (chromatic, morphologic, motion, depth, etc.).
Three approach of segmentation:
Clustering image data and growing regions: if the pixel are similar to the near one we consider the pixel as part of the segment
Border following: we select the region of transition and we look for the shape that describe the contours between two region
Search of borders: we start looking from region that...

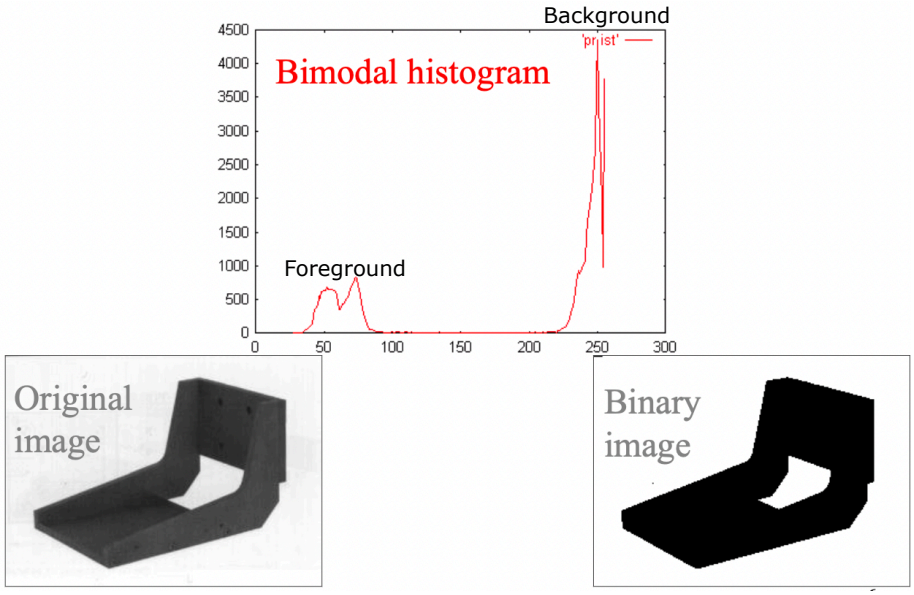
Binary images: The segmentation process leads to detect an individual object –foreground– in contrast to the background so it is a **binarization** process. Often they can contain various grey levels due to noise...

Bimodal Distribution

The easiest solution for the binarization is a **threshold** applied to the grey levels:
O(i, j) = 255 if I(i, j) < Th
O(i, j) = 0 otherwise

Problem: the pixels in the green and purple area that are mistaken from the other group





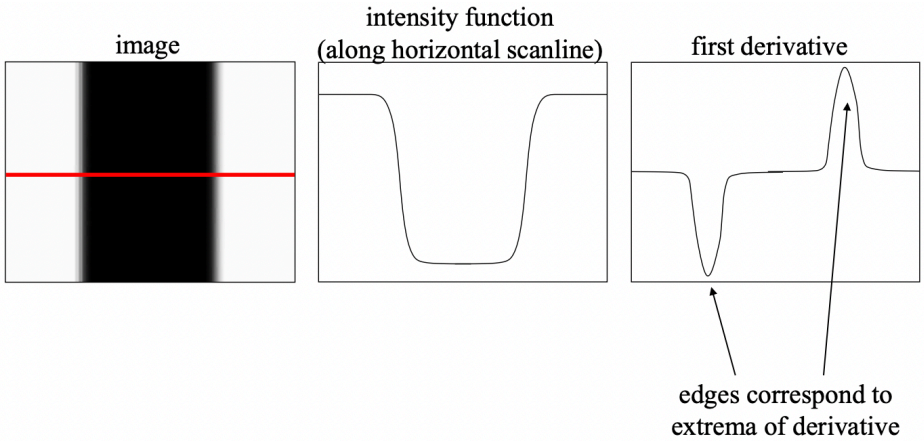
13/10/2022

Computer Vision

Searching for borders: different approaches

Consider the image a function and use a **derivative** approach in order to search for borders: the border search can be based on the **discontinuity** of an image feature like the grey level.
First approach: Looking for the first order partial derivatives.
Second approaches: Looking for second derivative among operators such as **Laplacian** (this operator is isotropic –> zero crossing)

An **edge** is a place of rapid change in the image intensity function.
E.g.: High level on the left and right and a low level in the center.



The first derivative is given by:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

The second derivative is given by:

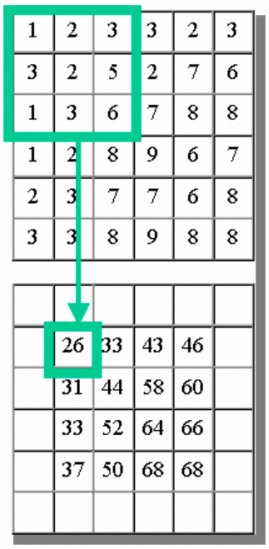
$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

We evaluate the intensity variation as difference between two digital point from left and right. Very often we don't divide by 2h (also h=1 very often).
It can be used the gradient in 2D.

The evaluation of the derivative may be considered as a particular case of **Convolution**. Convolution is a linear operator, that is applied when the image I(x, y) is continue. To the digital image I(i, j) a filter is applied represented by the mask:

$$O(x_0,y_0) = \iint f(x_0-x, y_0-y)\, I(x,y)\, dx\, dy$$

$$O(x,y) = \sum\sum f(x-i, y-j)\, I(i,j)$$

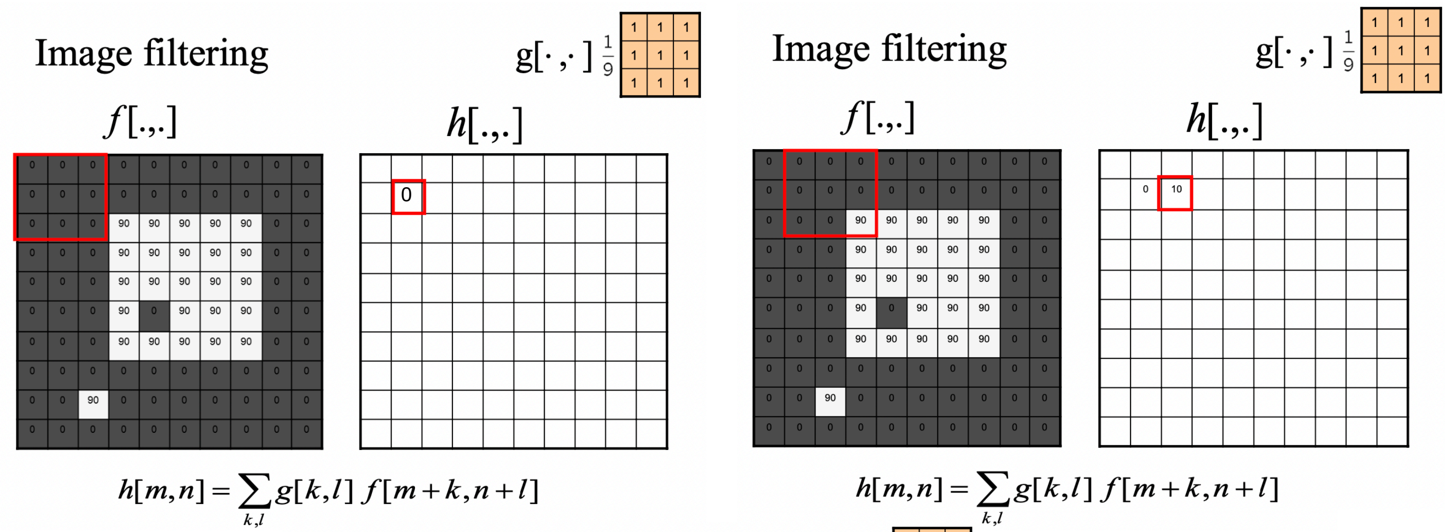


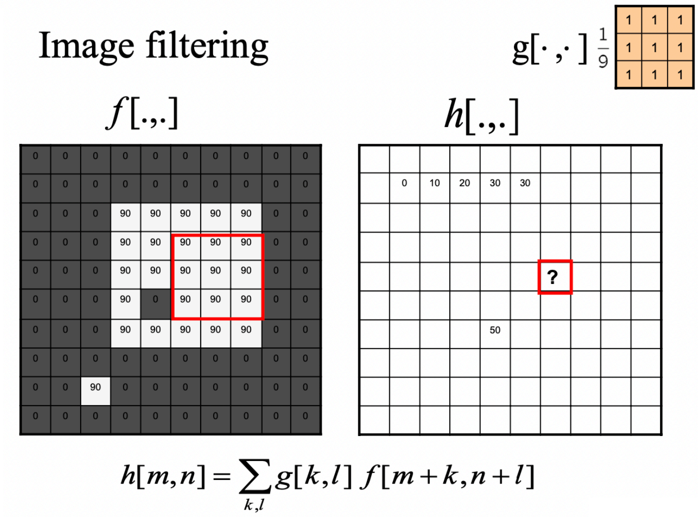
13/10/2022

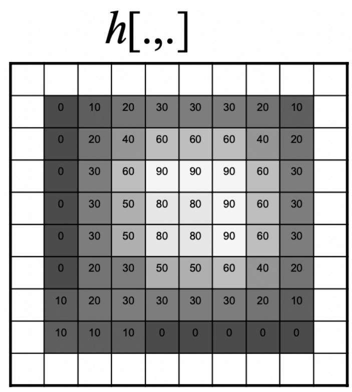
Box filter:

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

For each value of convolution we maintain the value.



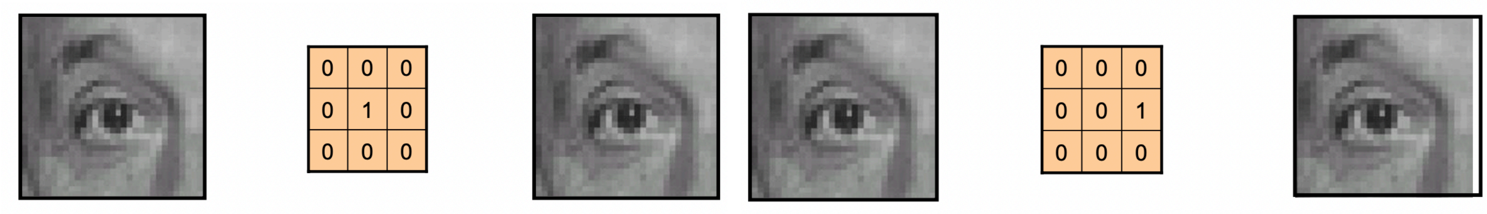




In the end it replaces each pixel with an **average** of its neighborhood

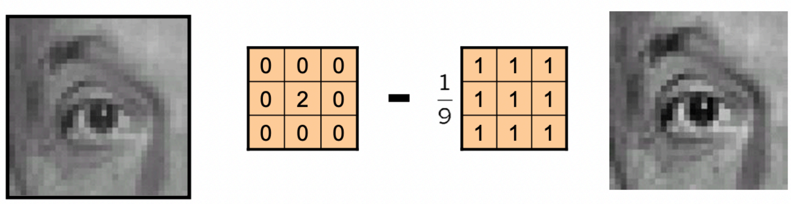
-> **Smoothing** effect.

Other cases:



Original — Filtered (no change) — Original — Shifted left By 1 pixel

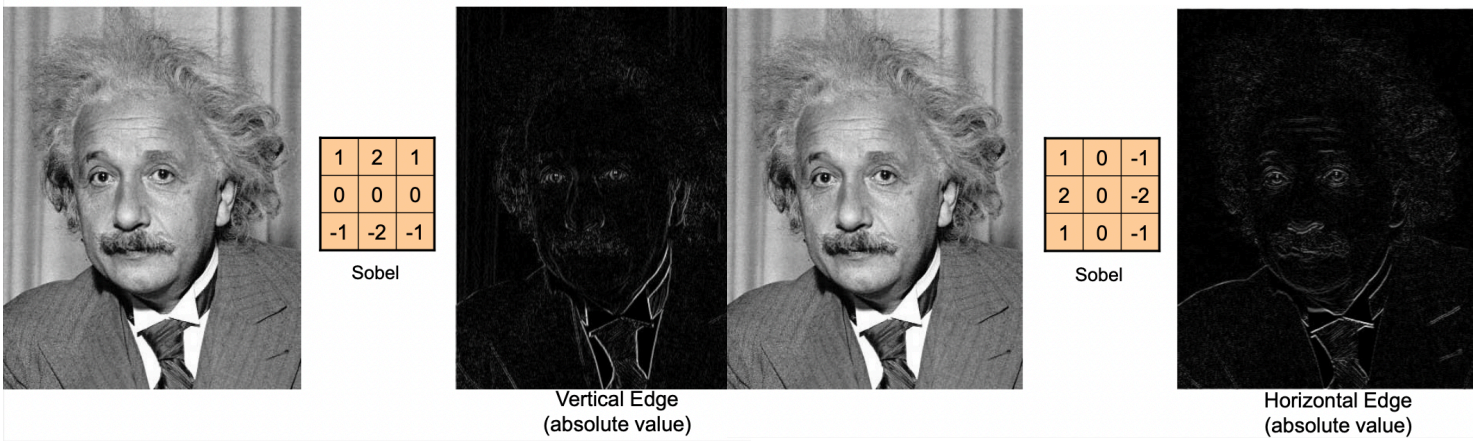
Sometimes some value can be greater than 255, so pay attention ->



Original

Sharpening filter
- Accentuates differences with local average

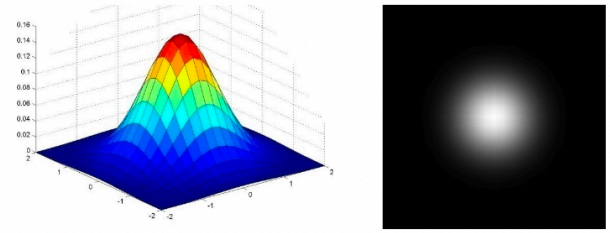
Sobel: it's useful to obtain the edges in two steps.



NB: The image can be considered as a signal, spatial filtering determines frequency effects –> Integrator = low-pass filter; Derivator = high-pass filter

Properties:
Commutative: a*b=b*a
Associative: a * (b * c) = (a * b) * c
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

Gaussian filter (low-pass filter)



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

5 x 5, σ = 1

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

It weights the contributions of the neighbors pixels by **nearness**, using the Gaussian formula.
Remove "high-frequency" components from the image (low-pass filter) –> **Smoothing**
Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width σ√2.

The 2D Gaussian can be expressed by a product of two functions, one function of x and the other function of y:

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

What about near the edge?

Methods:

- not consider borders pixels, removing them
- wrap around, the image is consider as a periodic image, each pixel of the left correspond to the pixel of the right and viceversa
- copy edge, the external pixels are copied in the same part of the image
- reflect across edge, the reflection of edges the pixels as a mirror

When we perform a **Convolution (G= H * F)** we start from:

$$G[i,j] = \sum_{u=-k}^{k}\sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

This is called cross-correlation, denoted $G = H \otimes F$

Cross-Correlation filtering replaces each pixel with a linear combination of its neighbors.

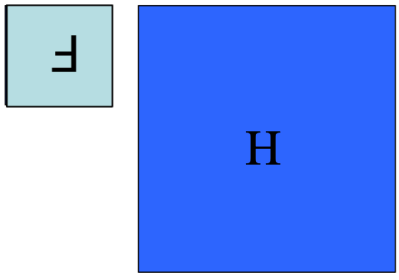
But before doing it we **flip the filter** in order to obtain the desired result:

- Flip the filter in both dimensions (bottom to top, right to left)
- Then apply cross-correlation

$$G[i,j] = \sum_{u=-k}^{k}\sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

$$G = H \star F$$

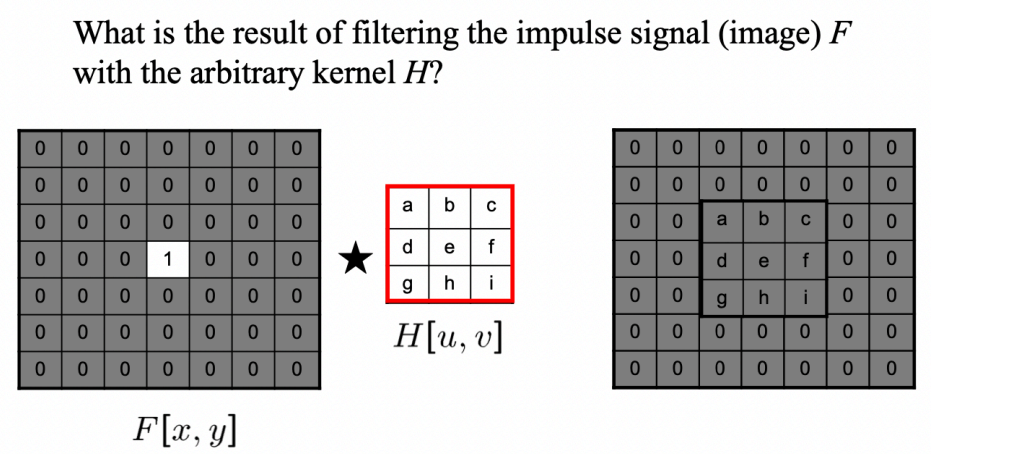
Notation for convolution operator

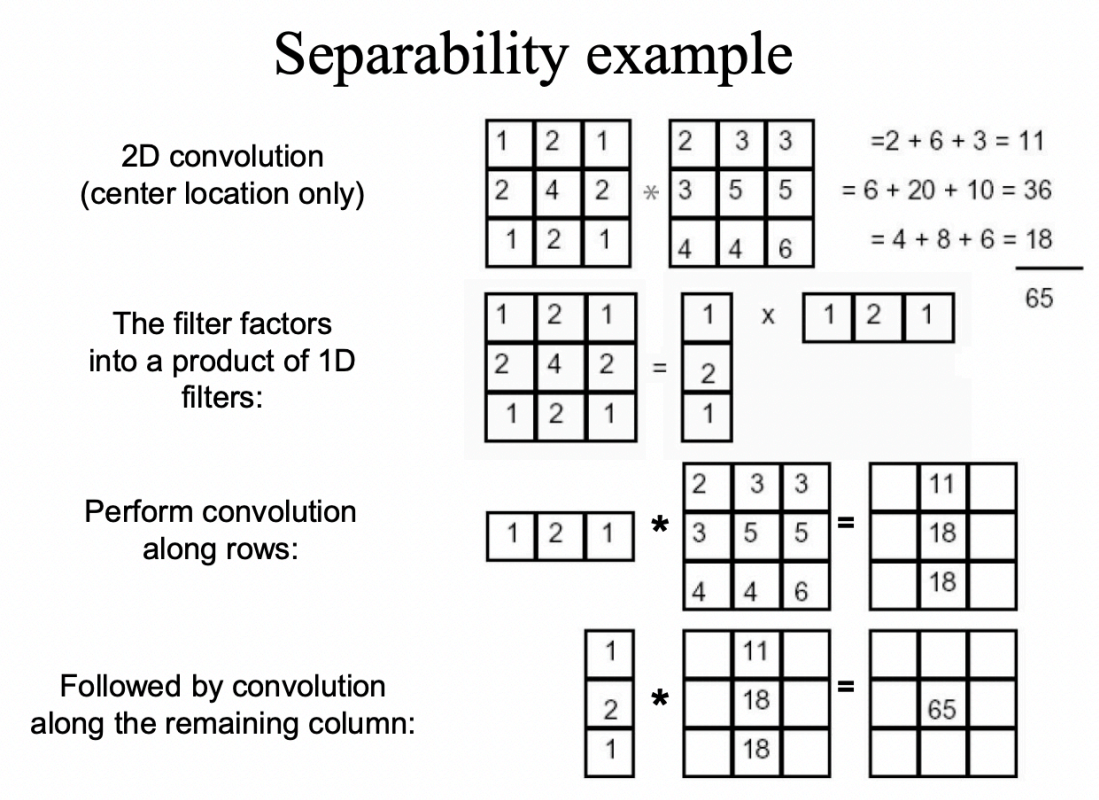


Convolution VS Cross-correlation

The **convolution** operation is very similar to the cross-correlation operation but has a slight difference. In convolution operation, the kernel is first flipped by an angle of 180 degrees and is then applied to the image.

A **correlation** operation yields a copy of the image but rotated by an angle of 180 degrees. Therefore, if we pre-rotate the filter and perform the same sliding sum of products operation, we should be able to obtain the desired result.





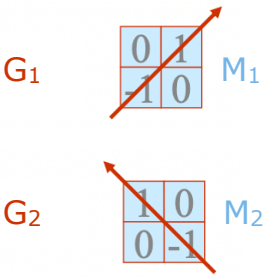
18/10/2022

Computer Vision

Gradient approximations

We use very simple **kernels** (2x2 or 3x3) or **masks M**, that are convoluted to the images **I** in order to obtain the values of **G**, that represent the variations of intensity (edges) [l'immagine 'trasformata' sarà la rappresentazione delle componenti del gradiente].

Roberts Operator

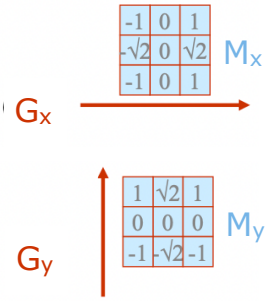


Most simple operator: difference between two pixels
Two templates are applied M1 and M2, obtaining the two orthogonal gradient components: $\mathbf{G_1=M_1*I}$, $\mathbf{G_2=M_2*I}$.
The gradient module and phase are:

$$\mathbf{G_m = sqrt(G_1^2 + G_2^2)}$$

$$\mathbf{G_f = arctg(G_2/G_1)+\pi/4}$$

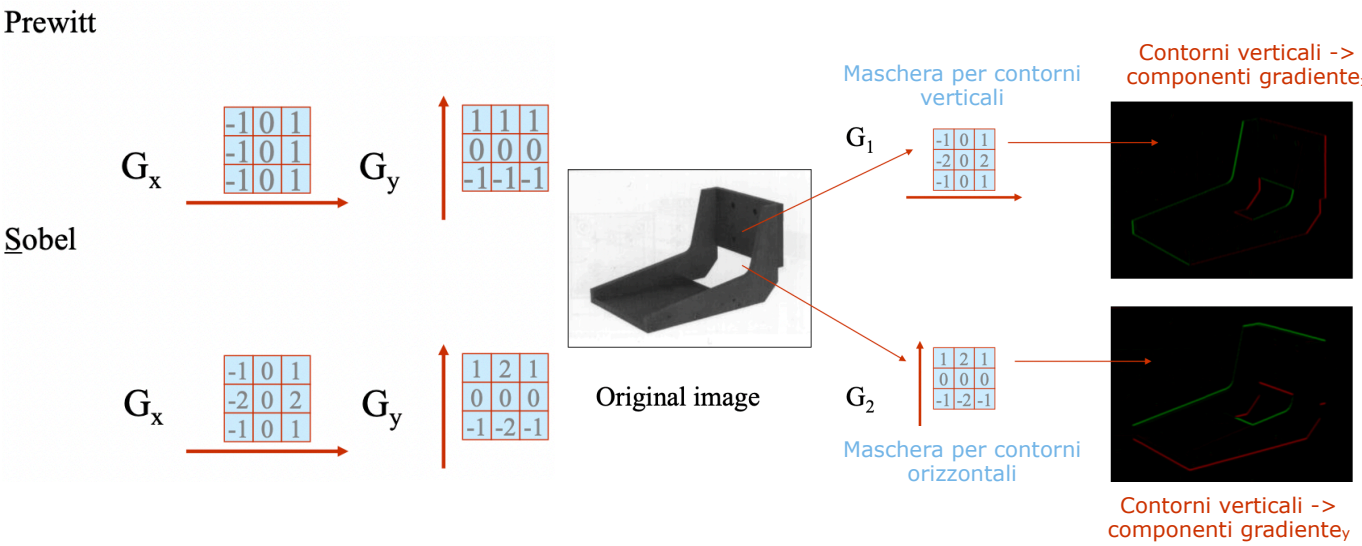
Isotropic Operator

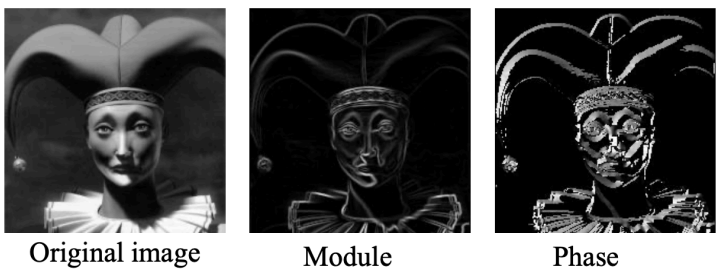


We have to do two convolutions
Two templates are applied M1 and M2, obtaining the two orthogonal gradient components: $\mathbf{G_x=M_x*I}$, $\mathbf{G_y=M_y*I}$.
The gradient module and phase are:

$$\mathbf{G_m = sqrt(G_1^2 + G_2^2)}$$

$$\mathbf{G_f = arctg(G_2/G_1)}$$





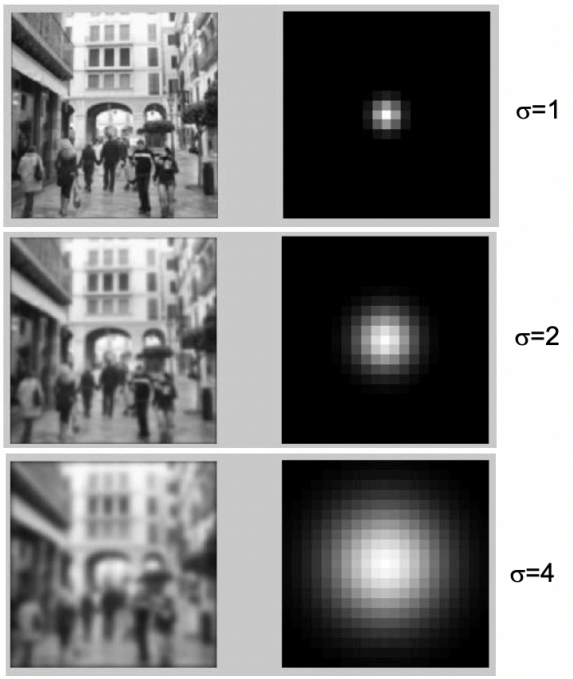
The **phase** shows more the variation of the colors than the **module**.

18/10/2022

Lateral inhibition is the ability of a receptor and/or neuron to reduce the activity of nearby cells. This mechanism makes it possible to better identify the origin of the stimulus. In the visual system it means improving the contrast of an image. Our entire visual system exists to see margins and contours. We see the world around us as a set of lines.
We evaluate colors and brightness based on comparisons and not on an absolute scale.

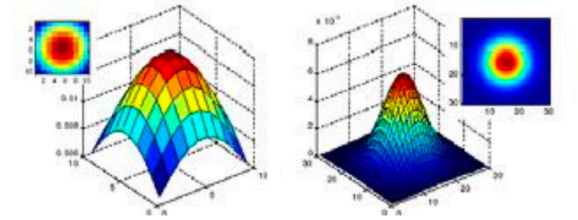
Gaussian filter

Variance of σ determines the extent of smoothing. The steepness (ripidità) of the gaussian curve gets bigger the less the σ is. It is better to use big kernels.



Small σ are better in terms of velocity.

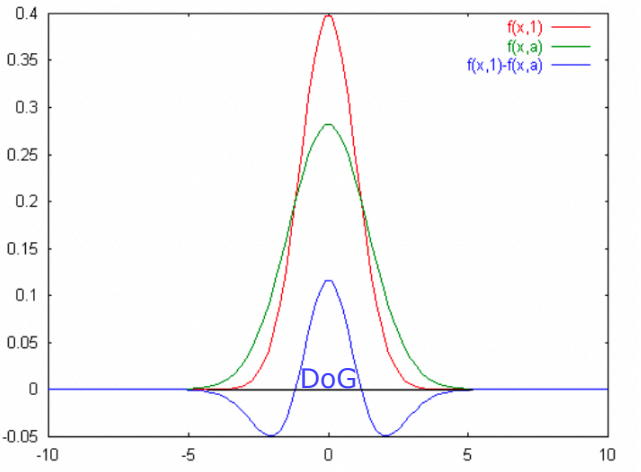
$$\frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



σ = 5 with 10 x 10 kernel σ = 5 with 30 x 30 kernel

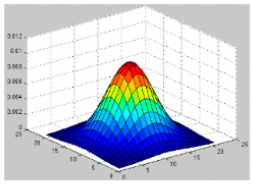
DoG filter

The implementation of the **lateral inhibition** can be done by a filter obtained by the difference of two Gaussian of equal area, having different σ.
The produced contour are closed.
We use different sigmas σ in order to obtain the closed contours as difference between the different smoothings from different gaussians.



Laplacian filter

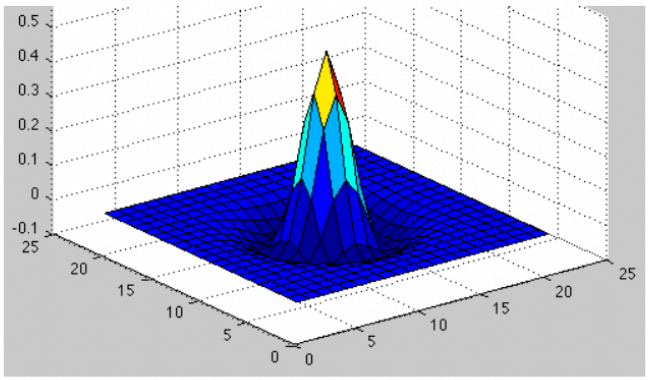
$$h(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



$$\nabla^2 g \otimes h = \left(\frac{\partial^2 g(x,y)}{\partial x^2} + \frac{\partial^2 g(x,y)}{\partial y^2} \right) \otimes h(x,y)$$

$$\nabla^2 g \otimes h = g \otimes \nabla^2 h$$

$$\nabla^2 h(x,y) = \left(\frac{x^2+y^2}{\sigma^4} - \frac{2}{\sigma^2} \right) \otimes h(x,y)$$



Canny edge detector (CED) **

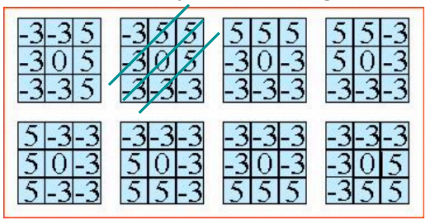
a) Filter image with derivative of Gaussian
b) Find magnitude and orientation of gradient
c) **Non-maximum suppression**:
   - Thin multi-pixel wide "ridges" down to single pixel width
d) Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

Template matching

An alternative method for edge detection.
Each operator tries different masks to choose the one with the best result.
In the second Kirsh's configuration the mask detects diagonal (45°) variation because the -3 are placed in front of the +5 on the opposite corners, because -3 and +5 have a strong value difference that leads to detect intensity variations.
NB: The inside sum has to be zero.

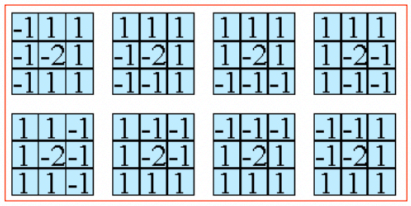
Kirsh's operator

8 possible configurations



Compass operator

8 possible configurations



18/10/2022

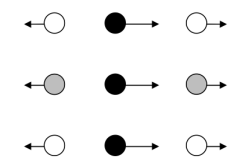
Riduzione del rumore con filtro Gaussiano (Fase a CANNY) **

Prima di iniziare l'elaborazione, l'immagine rumorosa viene sottoposta a convoluzione con un filtro gaussiano. Il risultato è un'immagine con una leggera "sfocatura" gaussiana, in cui nessun singolo pixel è affetto da disturbi di livello significativo.

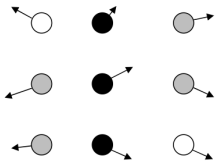
Ricerca del modulo e direzione del Gradiente (Fase b CANNY)

Al termine dell'elaborazione precedente va eseguito la ricerca dei massimi locali del gradiente. Il semplice confronto con una soglia non porta a risultati soddisfacenti. Un contorno di un'immagine può puntare verso una direzione qualsiasi.

Per ogni punto bisogna quindi individuare la direzione del gradiente e confrontare il modulo del gradiente nel punto in esame con i valori dei vicini e giacenti sulla direzione del gradiente.



caso semplice
gradiente orizzontale



caso generale

Soppressione dei non-massimi (Fase c CANNY)

Solo i punti corrispondenti a dei massimi locali sono considerati come appartenenti ad un contorno, e si cercheranno lungo la direzione del gradiente. Un massimo locale si ha nei punti in cui la derivata del gradiente si annulla.

Individuazione dei contorni mediante sogliatura (Fase d CANNY)

L'estrazione dei contorni si esegue con un procedimento chiamato sogliatura con isteresi, perché c'è bisogno di una sogliatura per decidere quali pixel rappresentano effettivamente un bordo e quali no. Vengono definite due soglie, una bassa ed una alta, che vengono confrontate con il gradiente in ciascun punto. Se il valore del gradiente è:

- inferiore alla soglia bassa, il punto è scartato;
- superiore alla soglia alta, il punto è accettato come parte di un contorno;

La presenza di due soglie (da cui il riferimento all'isteresi) è giustificato dal fatto che è praticamente impossibile trovare un unico valore del gradiente di luminosità per discriminare se un punto appartiene o no ad un contorno. Al termine di questo step si ottiene un'immagine binaria dove ciascun pixel è marcato come appartenente o no ad un contorno.

24/10/2022

Computer Vision

Common types of noise

The noise can come from electronic components.

Salt and pepper noise: random occurrences of black and white pixels
Impulse noise: random occurrences of white pixels
Uniform noise: constant probability density in a given range ±k. It seems like visible pixels.
Gaussian noise: variations in intensity drawn from a Gaussian normal distribution

First solution:
Replacing each pixel with an average of all the values in its neighborhood.
We can add weights to our average, for example a Gaussian set of weights:
[1,4,6,4,1]/16

Uniform noise

Each pixel as an additive noise, independent, with a uniform distribution.
This noise can be simulated adding in each pixel **n=2k(*rnd* – 0,5)** being k the noise max intensity and *rnd* a random number with $0 \leq rnd \leq 1$.

Salt and pepper noise (or impulse)

The image has dark pixels and bright pixels randomly distributed.
The noise can be simulated in this way:
if rnd ≥ Th_1 I(i,j) = 255
if rnd ≤ Th_2 I(i,j) = 0 with Ths = thresholds

Solutions:

Average (mean) filter

Example - given the neighborhood:

	3	6	8
	3	4	2
	5	8	3

the central pixel will take the new value:

(3+6+8+3+4+2+5+8+3)/9 = 4.67

Partially reduces noise but also some details.
Not good on uniform or salt and pepper noise.

Median filter

Operates over a window by selecting the median intensity in the window.
This is not a linear operation (not standard convolution).
Using a mean filter, it is not used an original value of the image.
The median, instead, is already a value of the image.
Good on salt and pepper noise and impulse noise.
The median filter assigns to a pixel the median value of neighborhood. It's good on impulsive noises because it excludes the extreme values.

Example - given the neighborhood:

	3	6	8
	3	4	2
	5	8	3

the correspondent values are:

2 3 3 3 4 5 6 8 8

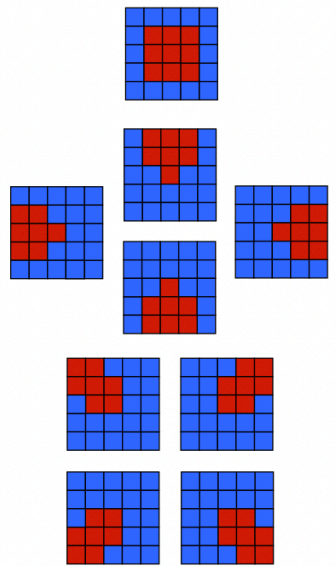
median value: 4;
over three values: 4;
over five values: 4,2;
over seven values: 4,57
over nine values: 4,66

In the **rank filters family** the pixel is assigned the average value over a predefined range of the neighbors histogram.

The bigger the dimension of the filter window, the more smoothed the image.

The Nagao-Matsuyama Filter

This filter selects for the centre pixel the average for the orientation with the least variation.



It calculates the **variance** for each of nine sub-groups (red) of the original image (blue).
All the sub-groups includes the central point.
It chooses the sub-group with the lowest variance and assign its mean to the centre pixel.

This filter improves borders and reduces noises.

24/10/2022

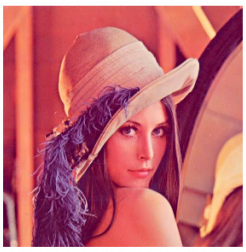
HISTOGRAM AND LUT OPERATIONS (LOCAL OPERATIONS)

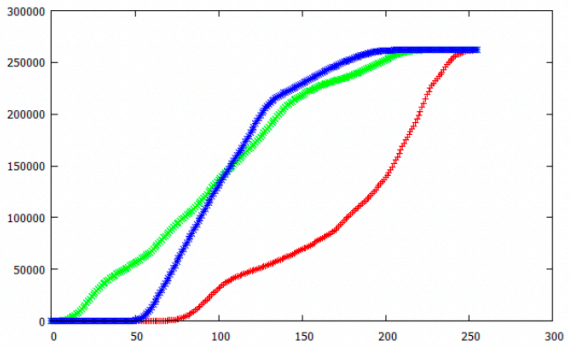
Histogram

An **histogram** is a representation of the number of pixels of the image divided in each value of the scale (gray-scale or color values).
For color images, we usually evaluate a different histogram for each channel.

Cumulative function

We have the number of pixel of the image with value less or equal to a value.
We have a monotonic function.
So C[255] is the total number of the image.





25/10/2022

Computer Vision

Local operation

The value of the pixels of the new image depends only on the value of the corresponding pixel of the original image.

newImage(i,j) = f(originalImage(i,j))

–> newImage(i,j) = LUT[originalImage(i,j)]

It is easily implemented by a **Look Up Table** (an array of integer).

Negative image

LUT(n)=255-n

The new image has a reflected histogram.

0XFF000000

Limited range image

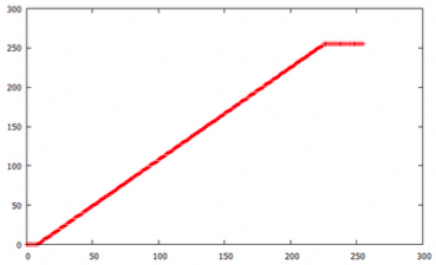
Only a small range is used in the image, very clear and very dark pixels are absent.

We can **extend the dynamics** performing a LUT.

$$LUT(n) = 255 \times \frac{n - \min}{\max - \min}$$

The pixel distribution may be extended by stretching with the following LUT:

(min e max, are the min and max values)

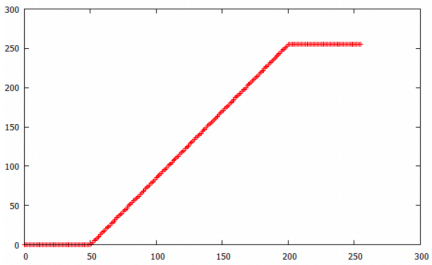


Highlight the contrast

Considering a small range values (a=50 and b=200)

We cannot go back to the original image.

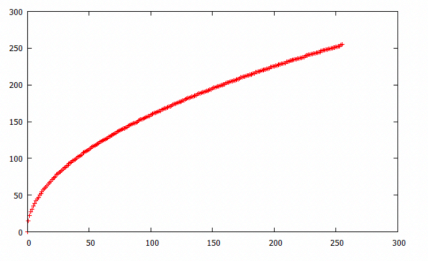
$$LUT(n) = \begin{cases} 0 & se \quad n < a \\ 255 \times \dfrac{n-a}{b-a} & se \quad a \le n \le b \\ 255 & se \quad n > b \end{cases}$$



Bright image

The histogram is compressed towards high values.

$$LUT(n) = \sqrt{255 \times n} = 255\sqrt{n/255}$$



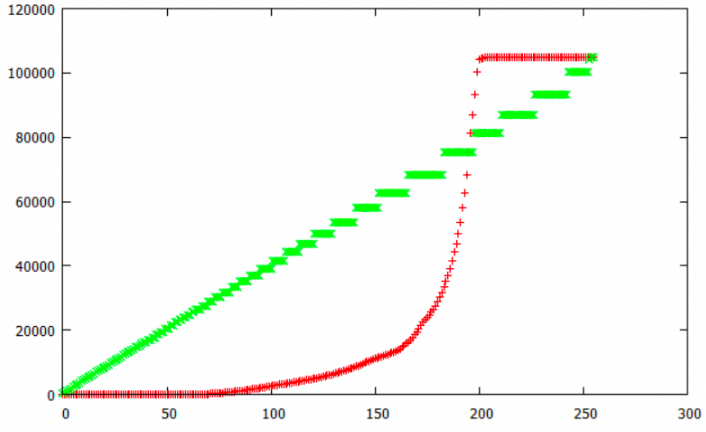
Dark image

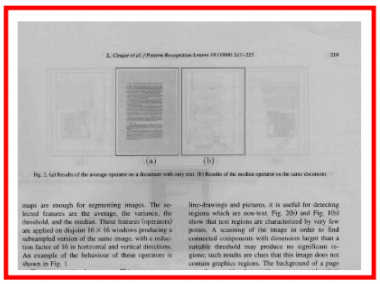
The histogram is compressed towards low values.

Image equalization

This technique consists in making the grey level distribution as close as possible to a uniform distribution, in an adaptive way.

$$LUT(n) = 255 \times \frac{\sum_{i=0}^{n} H(i)}{\sum_{i=0}^{255} H(i)}$$





Cumulative function: original equalized

Java memorizes a pixel value by an «int» (4 bytes: 3 for each RGB channel and one to represent the transparency):

```
int pixel = 0xFFRRGGBB
```

V is the int value of each pixel:

R = (V >> 16) & 255; // R = (V/0x10000) & 255

G = (V >> 8) & 255;

B = (V) & 255;

Of course: 0 ≤ R,G,B ≥ 255

In a similar way we get the pixel value from the channels:

V = (R<<16) | (G<<8) | (B) | 0xFF000000

A gray scale image is an image where all channels are equal

A simple way to get the gray value (or brightness) from color images is

G = (R+G+B)/3

But human eyes have different sensibilities for each color, a common choice is:

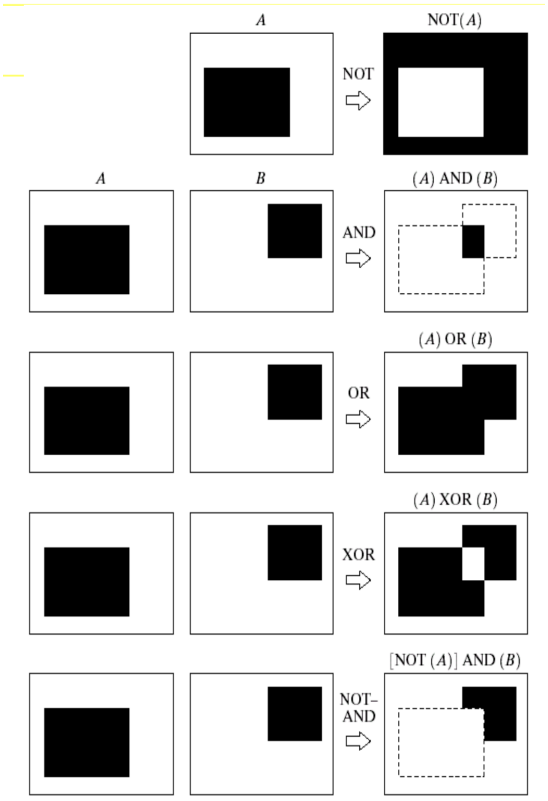
G = **0.299***R + **0.587***G + **0.114***B

The pixel values for gray image: **V = 0xFF000000 | (G<<16) | (G<<8) | G**

Computer Vision

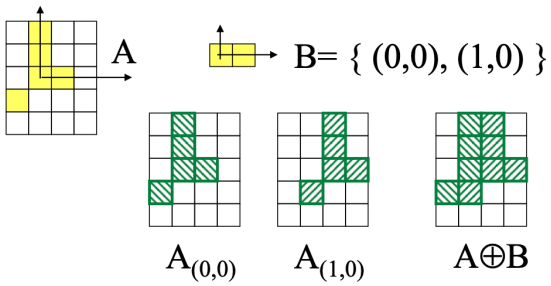
MATHEMATICAL MORPHOLOGY

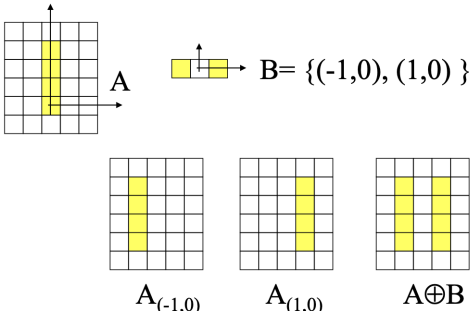
The term **morphology** is used to indicate the study of the geometrical structure or texture of an image.

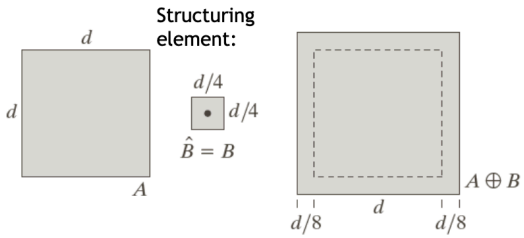


Dilation (Minkowski sum)

A⊕B where B is called the **structuring element**.



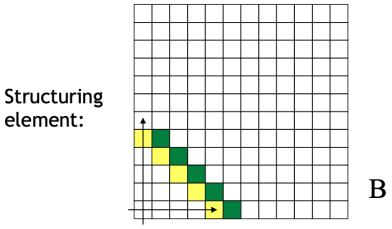


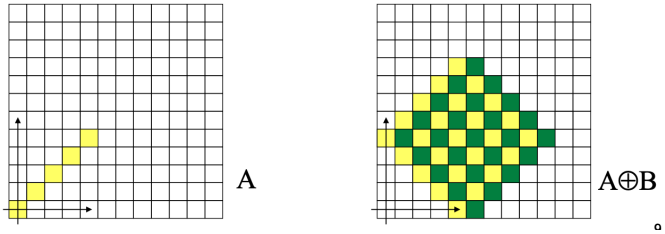


It's used to **enlarge the borders** as the structuring element is applied making its center match with the border pixels of image A.

In pratica: allineo l'origine dello structuring element B all'origine di A, se scorrendo B lungo A ho 'match' tra il centro di B e uno dei quadratini di A allora <u>disegno tutto B</u> su A. [Riporto tutto se il centro matcha]

Dilation

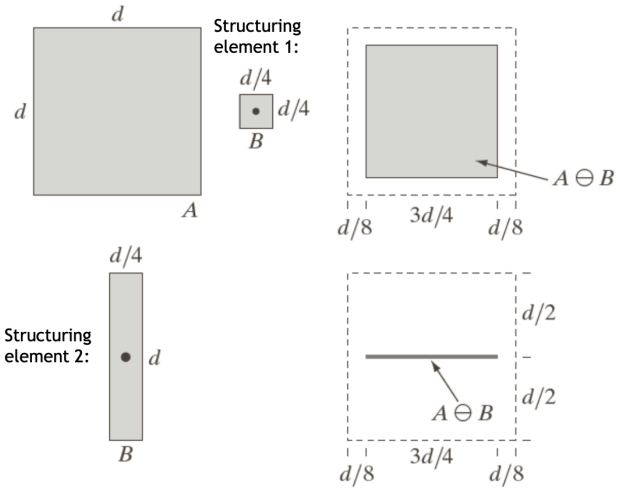


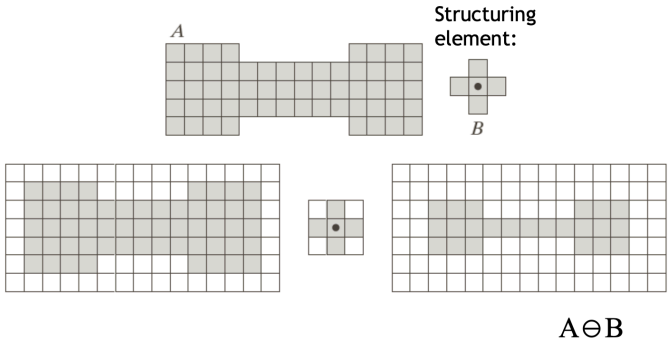


15/11/2022

Erosion (Minkowski difference)

A⊖B where B is called the **structuring element**.



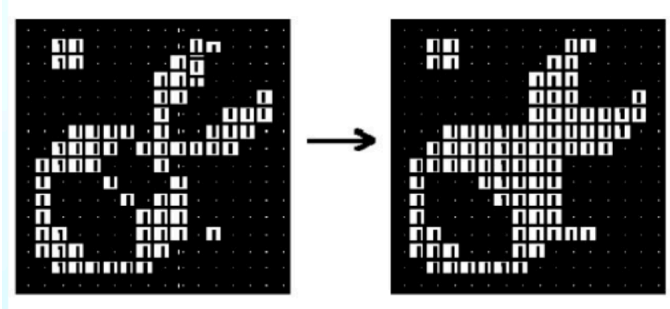


In pratica: allineo l'origine dello structuring element B all'origine di A, se scorrendo B lungo A ho un 'match' completo tra i quadratini, quindi se B è contenuto interamente in A, disegno solo il quadratino centrale di B in A altrimenti non riporto/disegno nulla.

Closing operator

A•K = (A⊕K)⊖K where K is called the **structuring element**.

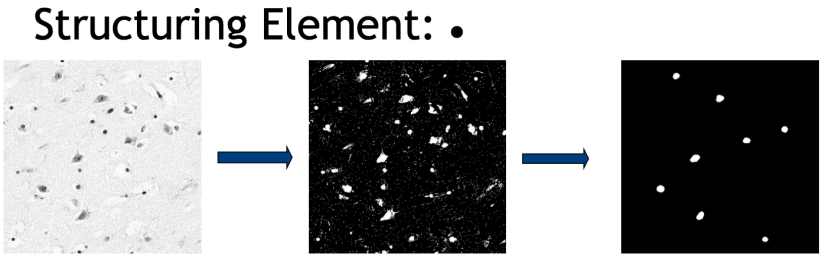
It's a **dilation + erosion** so it's used to close the eventual internal holes.



Opening operator

A∘K = (A⊖K)⊕K where K is called the **structuring element**.

It's an **erosion + dilation** so it's used to remove details from the image that are not similar to the shape of the chosen structuring element (and to smooth edges). It's smart to use a structuring element similar to the details I want to enhance.



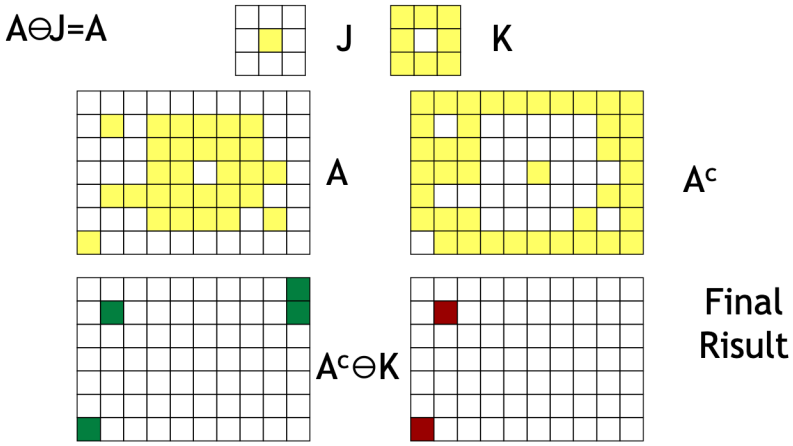
15/11/2022

Hit-and-Miss transform

$(A \ominus J) \cap (A^c \ominus K)$ where J and K are the **structuring elements**.

It's used to search for particular patterns made of pixels from foreground/ background.

Search of isolated points(8-connection)

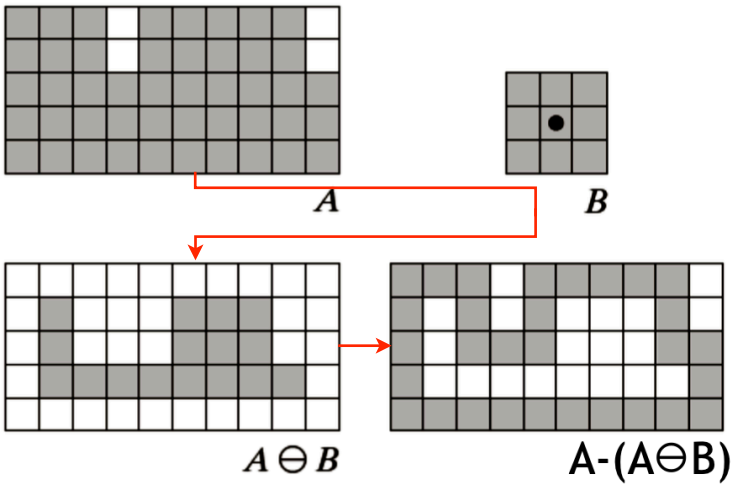


J and K can be seen as a single template with three values:

- Foreground points
- Background points
- Do not care points

It's like I have a double control of what I'm looking for.

Application of Erosion in order to obtain the contour:



21/11/2022

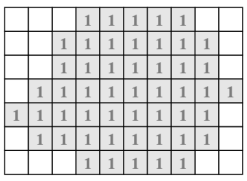
Computer Vision

Distance transform

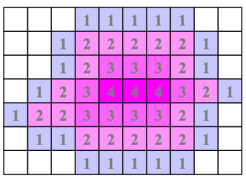
DT implementation using dilation and addition operators.
With R= evolving image at the end of the DT.
The result of the transform is a graylevel image that looks similar to the input image, except that the graylevel intensities of points inside foreground regions are changed to show the distance to the closest edges from each point.

$R = \varnothing$
while($A \neq \varnothing$) do
- $R = R + A$
- $A = A \ominus K$

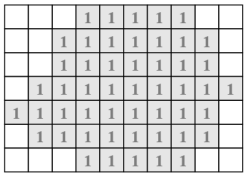
done



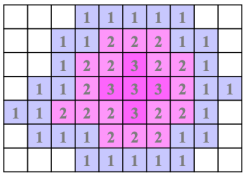
Structuring element:



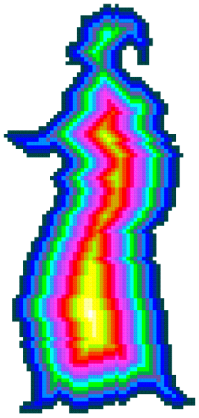
The binary image



Structuring element:



becomes



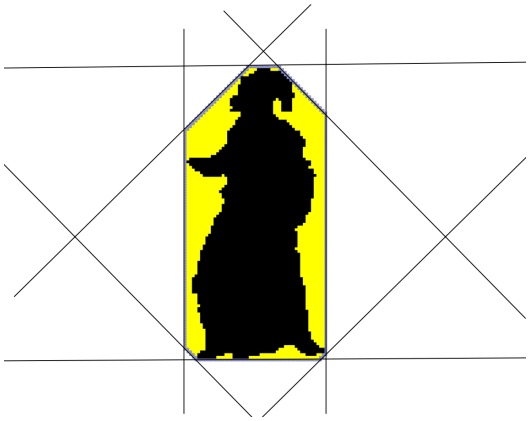
The Distance Transform (**DT**) is obtained by labeling all the pixels inside a binary object with their **distance to the background**.
Any pattern can be interpreted as the union of all its maximal **digital disks**.
A **disk** is a shape made of same intensity value pixels.
The set of the centers of the maximal disks with their labels, constitutes the **MAT**. This transform is complete in the sense that it is possible to revert it, so obtaining the original object back.

Weighted DT

In this case all neighbors are not considered at the same distance (e.g. 8-connectivity).

8-Convex Hull

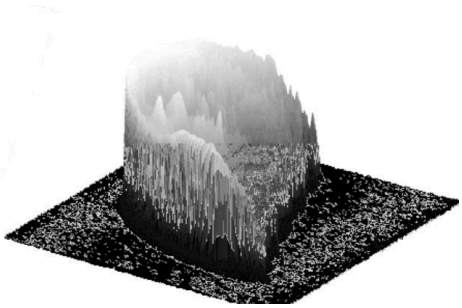
A set A is said to be **convex** iff the straight line segment joining any two points in A lies entirely within A. The convex hull is the minimum n-sided convex polygon that completely circumscribes an object, gives another possible description of a binary object.



22/11/2022

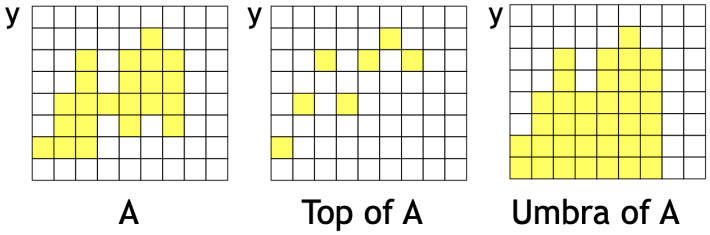
Computer Vision

MATHEMATICAL MORPHOLOGY: GRAYSCALE IMAGES AND 3D



A 2D grayscale image is treated as a 3D solid in space – a **landscape** – whose height above the surface at a point is proportional to the brightness of the corresponding pixel.
The **support** is the foreground, the **complement of support** is the background.

Umbra



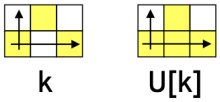
Top of A (example for n=2): T[A](x) = max { y | (x, y) ∈ A }
Umbra of f (f:F –> E): U[f] = { (x, y) ∈ F x E| y ≤ f(x) }

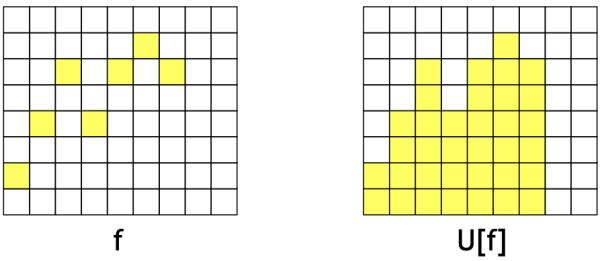
Gray scale Dilation

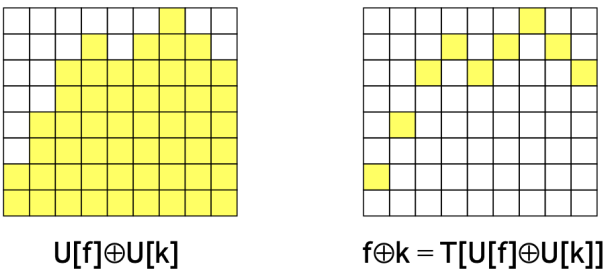
The dilation of an image f and structural element k can be defined as:

f⊕k = T{U[f]⊕U[k]}

It tends to brighten the image, reducing dark regions.





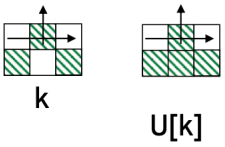


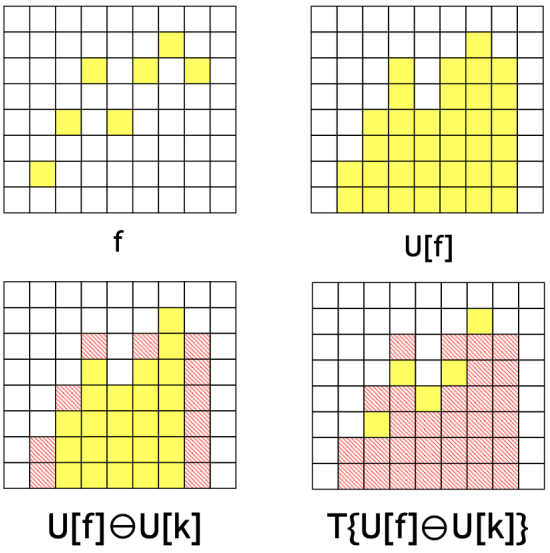
Gray scale Erosion

The erosion of image f and structural element k can be defined as:

$$f \ominus k = T\{U[f] \ominus U[k]\}$$

It tends to darken the image, reducing bright regions.

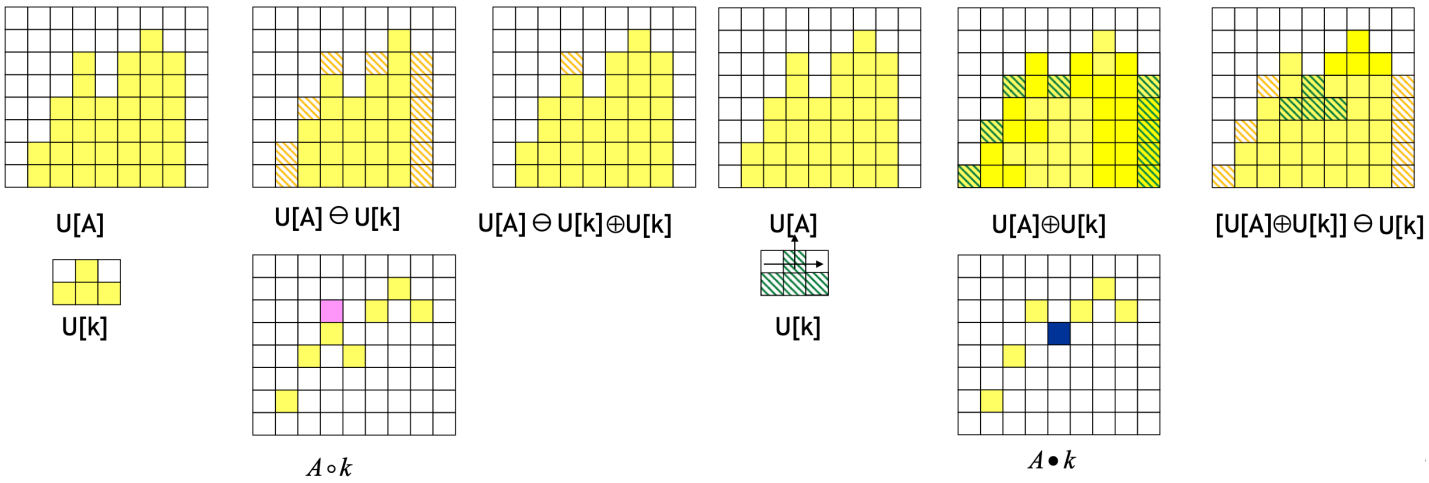


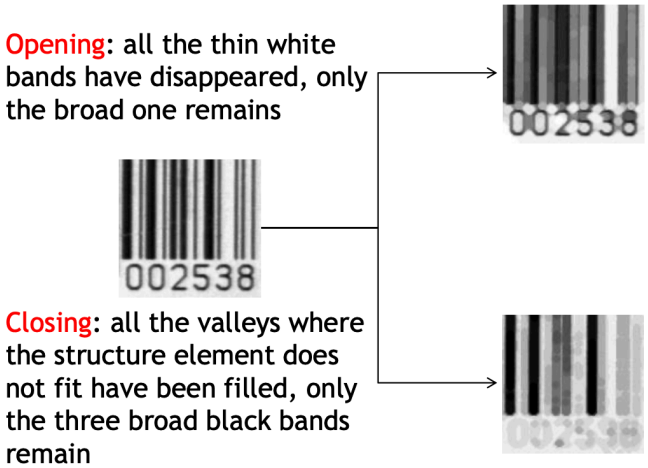


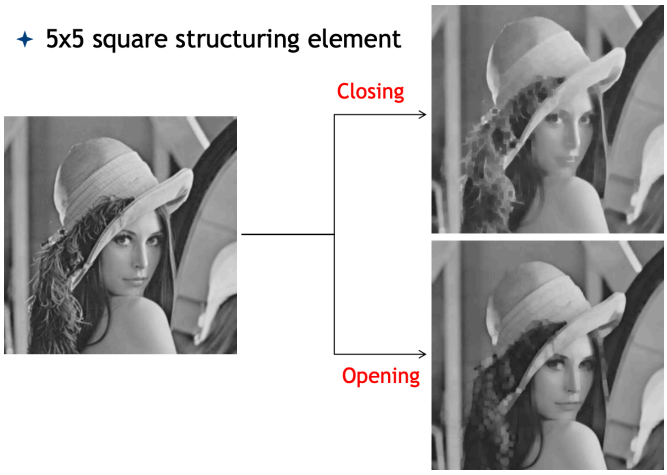
Gray scale Opening and Closing

Opening and closing of an image f(x,y) by a structuring element b(x,y) have the same form as their binary counterpart:

$$f \circ b = (f \ominus b) \oplus b \qquad f \bullet b = (f \oplus b) \ominus b$$

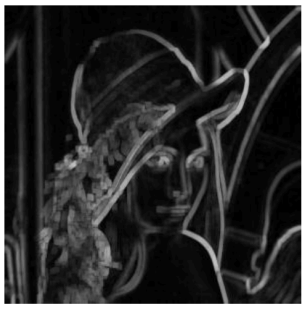






Morphological gradient

The **edges** are enhanced and the contribution of the homogeneous areas are suppressed, thus producing a "derivative-like" (gradient) effect.
It's **dilation minus erosion**.



Top-hat and Bottom-hat Transformations

The **top-hat** transformation of a grayscale image f is defined as
f minus its opening: $\mathbf{T_{hat}(f)=f-(f\circ b)}$
The bottom-hat transformation of a grayscale image f is defined as
its closing minus f: $\mathbf{B_{hat}\ (f)=(f\bullet b)-f}$

NB: One of the principal applications of these transformations is in removing objects from an image.

Granulometry

Granulometry deals with determining the size distribution of particles in an image.
A morphological approach to determine size distribution, to construct a histogram, is based on opening operations of particular size that have the most effect on regions of the input image that contain particles of similar size.

Computer Vision

PATTERN RECOGNITION: VISUAL SEARCH AND HOUGH TRANSFORM

Scene

Template

A simple idea:

$$E(y,x)=\sum_{i,j}\left(I(y+i,x+j)-T(i,j)\right)^2$$

$$E(y,x)=\sum_{i,j}\left|I(y+i,x+j)-T(i,j)\right|$$

I move the target through the image and I compare target and image minimizing an error function creating a **correlation map**.

Generally we can divide the different objects recognition techniques in:

Appearance-based methods, in which are used templates of the objects to perform recognition evaluating:

- Changes in lighting or color;
- Changes in viewing direction;
- Changes in size or shape.

Feature-based methods, a search is used to find feasible matches between object features and image features:

- Surface patches;
- Corners;
- Linear edges.

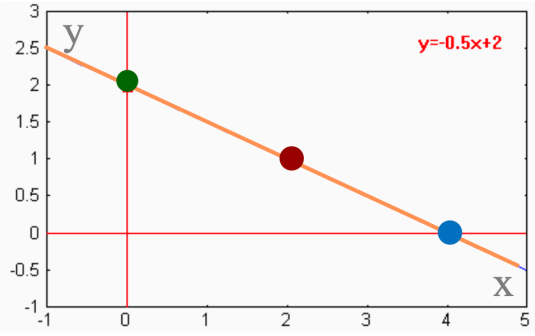
5/12/2022

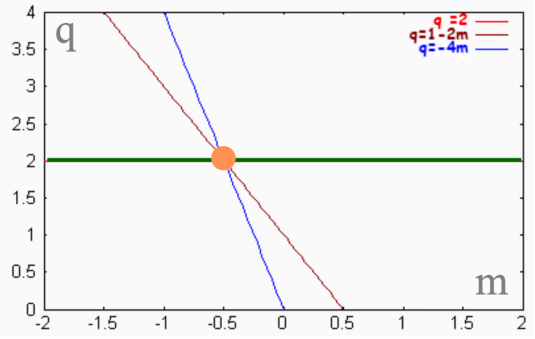
Hough Transformation

Each contour point identified in an image can support the existence of the set of straight lines crossing its location. If a straight line is present in the image, and N of its points are detected, N sets of lines receive a contribution but only the common single straight line receives N contributions.

Si consideri un punto **a'=(x' , y')** sul piano bidimensionale dell'immagine. Le rette passanti per tale punto sono **y'=m'•x+q'** per ogni **m'** e **q'**. Tale equazione descrive anche una curva nello spazio parametrico m-q. Allo stesso modo, considerando un secondo punto **a"=(x" , y")**, si ottiene che tutte le rette passanti per esso sono **y"=m"•x+q"**. Ancora una volta, fissando x e y, tale equazione descrive una curva nello spazio m-q.

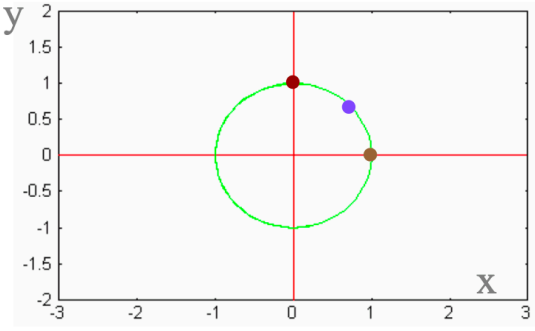
L'intersezione di tali curve nello spazio m-q (detto **spazio parametrico PS**) identifica gli m e q specifici della curva che nello spazio x-y collega i punti **a'** e **a"**.

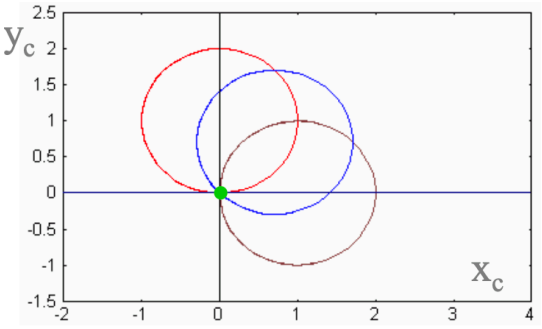




Può essere applicata anche con i cerchi modificando l'equazione da retta a circonferenza e usando come PS x_c–y_c ipotizzando un raggio noto.

$$(y-y_c)^2+(x-x_c)^2=r^2$$





Poiché per tre o più punti sarà sempre più difficile ottenere curve perfettamente passanti per essi, nello spazio parametrico otterremo più intersezioni tra curve e bisognerà scegliere quella che più si avvicina. Per fare ciò ciascun bordo/punto dello spazio immagine **darà un voto** al set di parametri che meglio lo 'fitta' e alla fine verrà scelto il set che ha ottenuto più voti dai diversi bordi/punti.

Implementation of the HT

Si definisce una **mapping rule** (un insieme di regole e proprietà tipiche del target) ad esempio per un poligono è costante la distanza dal centro. E si cercano quindi curve e punti che soddisfino queste mapping rules.

Generalized Hough Transformation (GHT)

L'algoritmo, invece di utilizzare l'equazione parametrica della curva, sfrutta le informazioni contenute in una tabella detta Reference-Table. In essa, per ogni punto p(x,y) del contorno appartenente al modello S inserito dall'utente, si memorizza il modulo e orientamento del contorno.

In general:
Both the classical Hough transform and its more modern variants proceed by converting the input image into a new representation called the Hough image which lives in a domain called the Hough space. Each point in the Hough space corresponds to a hypothesis about the object of interest being present in the original image at a particular location and configuration.
Any Hough transform based method essentially works by splitting the input image into a set of voting elements. Each such element votes for the hypotheses that might have generated this element.
Of course, voting elements do not provide evidence for the exact localization and thus their votes are distributed over many different hypothesis in the Hough space. Large values of the vote are given to hypotheses that might have generated the voting element with high probability. The votes from different voting elements pixels are added together into a Hough image. The objects of interest are then detected as peaks in the Hough image, with the height of the peak providing the confidence of the detection.

Vanishing lines

They are a set of straight lines that has an intersection in a single point, called **vanishing point**. The vanishing point is important because it's an invariant information specific for an image/situation that is useful to determine the position of objects inside a 3D environment. These lines are also useful to determine the horizons line that leads to an easier reconstruction of the 3D scene.