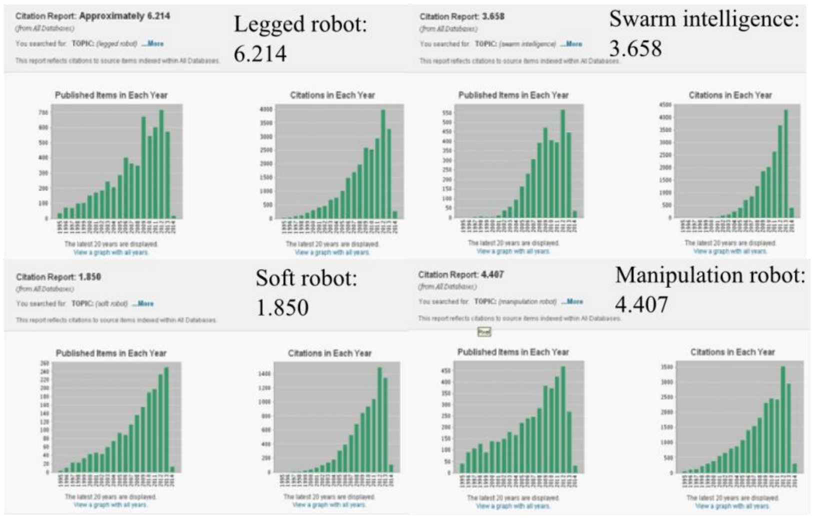
Marzo, 2015

Visione Artificiale:

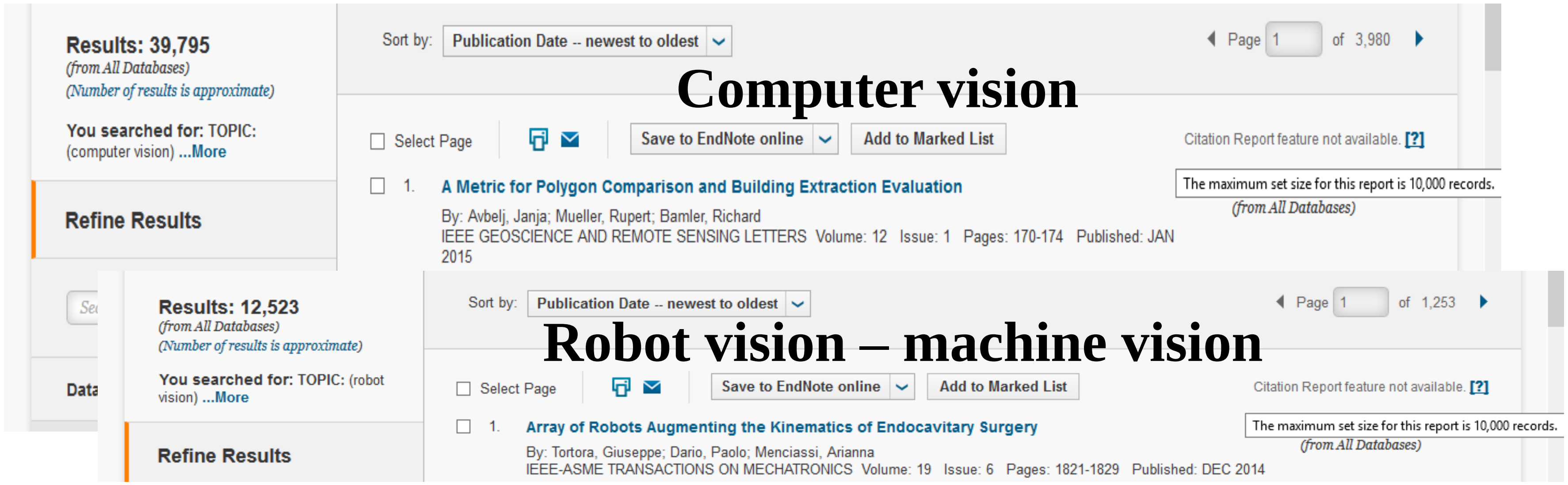
introduzione all’elaborazione di immagini digitali

ISTITUTO DI BIOROBOTICA

Scuola Superiore Sant’Anna



La visione è un campo molto studiato, con applicazioni che variano dalla medicina all’intrattenimento



Computer vision

Results: 39,795
(from All Databases)
(Number of results is approximate)

You searched for: TOPIC: (computer vision) ...More

Refine Results

Sort by: Publication Date -- newest to oldest

Page 1 of 3,980

Select Page | Save to EndNote online | Add to Marked List

Citation Report feature not available. [?]

The maximum set size for this report is 10,000 records.
(from All Databases)

1. **A Metric for Polygon Comparison and Building Extraction Evaluation**
By: Avbelj, Janja; Mueller, Rupert; Bamler, Richard
IEEE GEOSCIENCE AND REMOTE SENSING LETTERS Volume: 12 Issue: 1 Pages: 170-174 Published: JAN 2015

Robot vision – machine vision

Results: 12,523
(from All Databases)
(Number of results is approximate)

You searched for: TOPIC: (robot vision) ...More

Refine Results

Sort by: Publication Date -- newest to oldest

Page 1 of 1,253

Select Page | Save to EndNote online | Add to Marked List

Citation Report feature not available. [?]

The maximum set size for this report is 10,000 records.
(from All Databases)

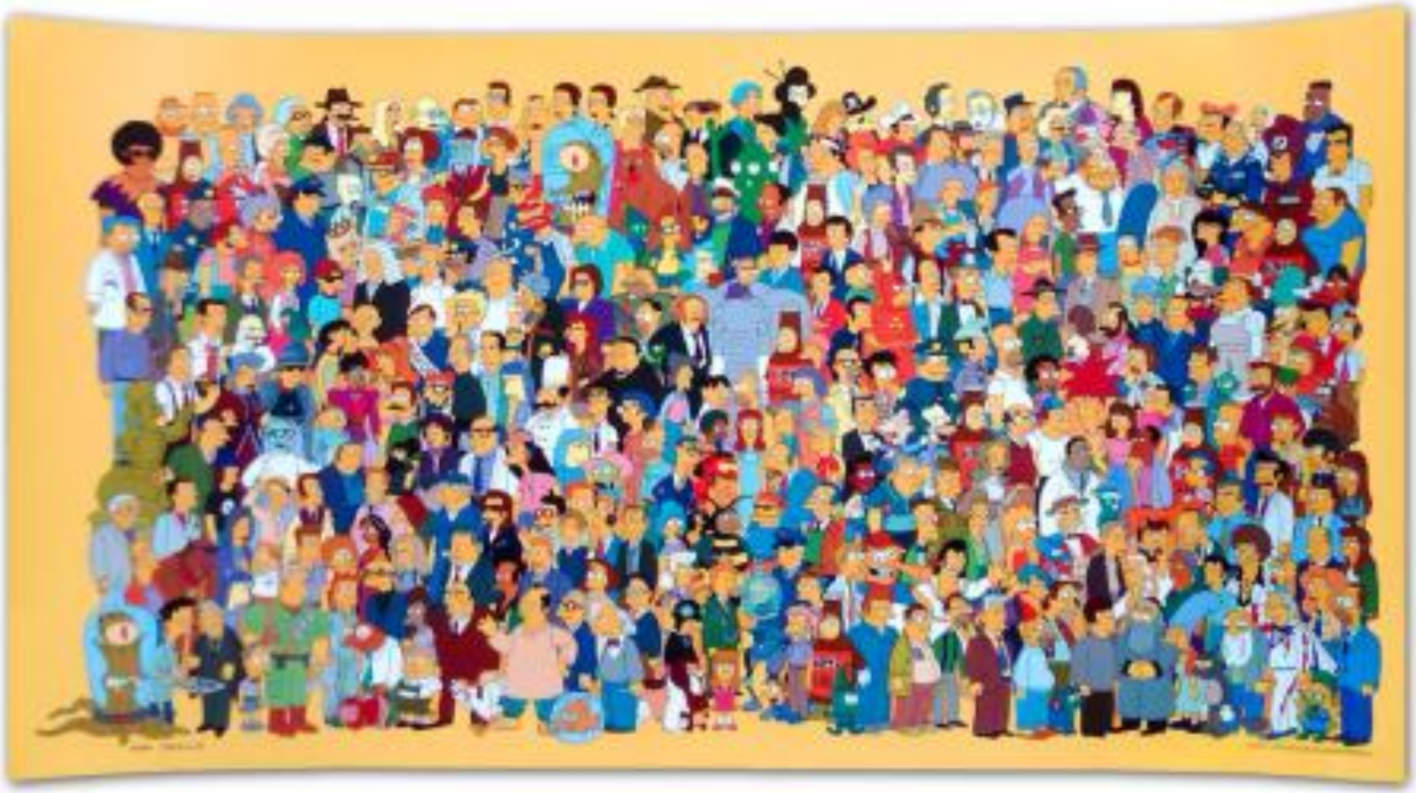
1. **Array of Robots Augmenting the Kinematics of Endocavitary Surgery**
By: Tortora, Giuseppe; Dario, Paolo; Menciassi, Arianna
IEEE-ASME TRANSACTIONS ON MECHATRONICS Volume: 19 Issue: 6 Pages: 1821-1829 Published: DEC 2014

Io, modestamente, può!

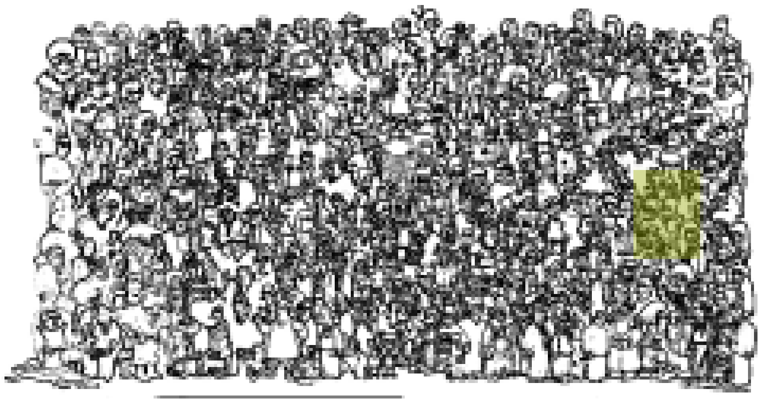
Ho visto cose che voi umani...

anche i ricercatori sognano pecore elettriche?

Where is Bart?

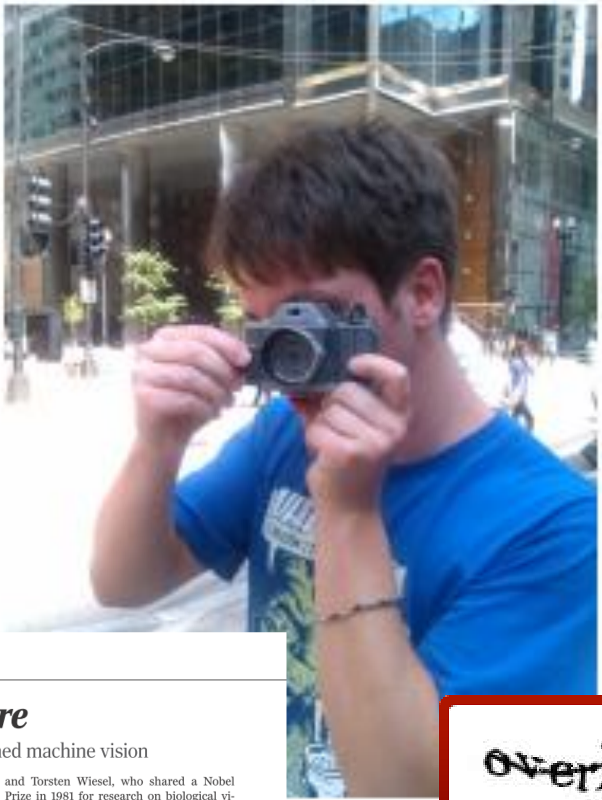


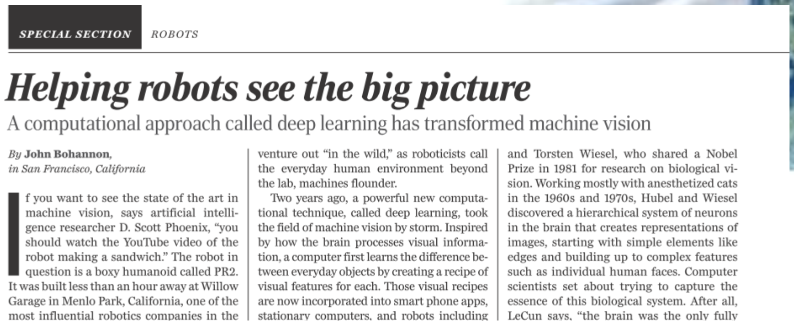
Può essere facile per un computer…



- so esattamente cosa cercare
- (so anche come cercarlo)

Digita un nome...





SPECIAL SECTION ROBOTS

Helping robots see the big picture

A computational approach called deep learning has transformed machine vision

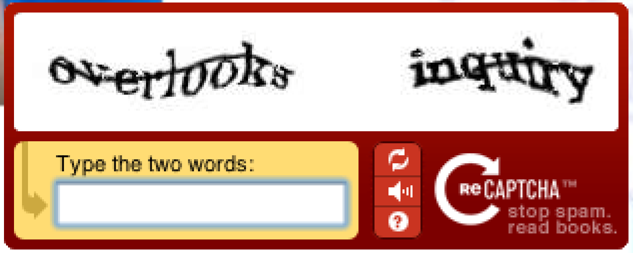
By **John Bohannon,**
in San Francisco, California

If you want to see the state of the art in machine vision, says artificial intelligence researcher D. Scott Phoenix, "you should watch the YouTube video of the robot making a sandwich." The robot in question is a boxy humanoid called PR2. It was built less than an hour away at Willow Garage in Menlo Park, California, one of the most influential robotics companies in the

venture out "in the wild," as roboticists call the everyday human environment beyond the lab, machines flounder.

Two years ago, a powerful new computational technique, called deep learning, took the field of machine vision by storm. Inspired by how the brain processes visual information, a computer first learns the difference between everyday objects by creating a recipe of visual features for each. Those visual recipes are now incorporated into smart phone apps, stationary computers, and robots including

and Torsten Wiesel, who shared a Nobel Prize in 1981 for research on biological vision. Working mostly with anesthetized cats in the 1960s and 1970s, Hubel and Wiesel discovered a hierarchical system of neurons in the brain that creates representations of images, starting with simple elements like edges and building up to complex features such as individual human faces. Computer scientists set about trying to capture the essence of this biological system. After all, LeCun says, "the brain was the only fully



Domanda

Un algoritmo per il riconoscimento di forme viene impiegato in una catena di montaggio di cerchioni per auto. L'algoritmo guida un braccio robotico riconoscendo la forma del cerchione e individuandone la posizione sul nastro trasportatore. Tutto funziona alla perfezione da Ottobre a Marzo, quando improvvisamente l'algoritmo non riconose più la forma. Cosa è successo?

SW & HW

Fatevi aiutare dall'hardware!

Spesso gli algoritmi di visione sfruttano conoscenze pregresse: forme, ambiente di lavoro, etc... Questo è ovviamente un limite per la generalità, ed è il problema dell'algoritmo dei cerchioni

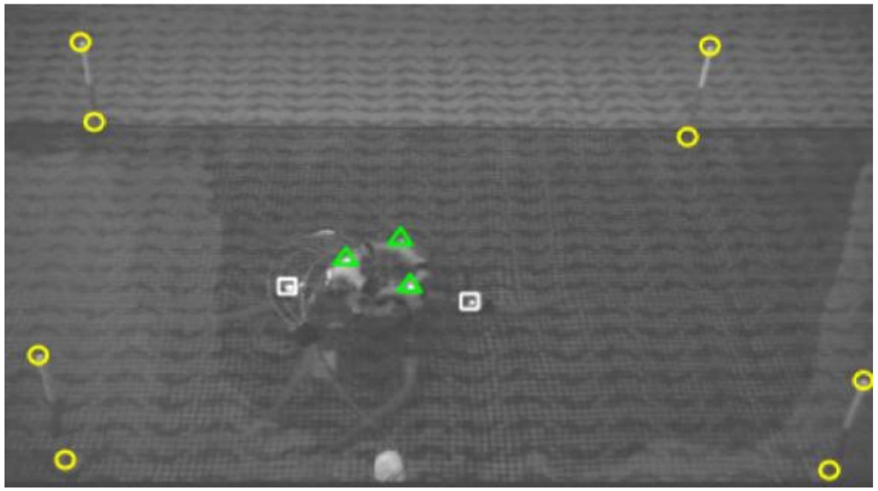
Sommario delle lezioni

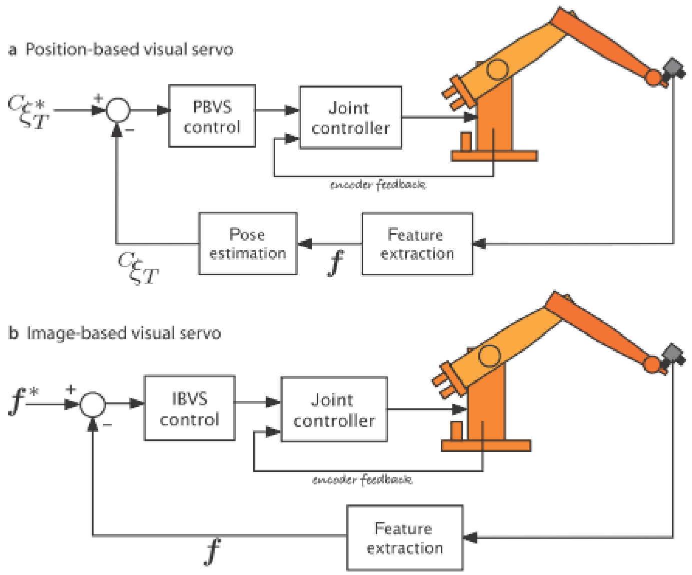
- Introduzione:
  - Immagini digitali; operatori; istogrammi;
- Elaborazione di immagini:
  - Denoising; segmentazione; sogliatura automatica (Otsu); individuazione di bordi: Hough; basata sul gradiente; rappresentazione di bordi; erosione
- Ricostruzione tridimensionale
  - Principi di base; Direct Linear Transformation; Flusso ottico; controllo di volo ispirato dalle api;
- Argomenti avanzati
  - Valutazione della sfocatura e auto-focus;
- Esempi finali: Justin, Circle detection per diagnosi

Visione per

Ricostruzione di posizione e posa

Controllo del robot





The 3D positions of the centre of mass of the robot over time is extracted to analyze the robot dynamics.

The overall example

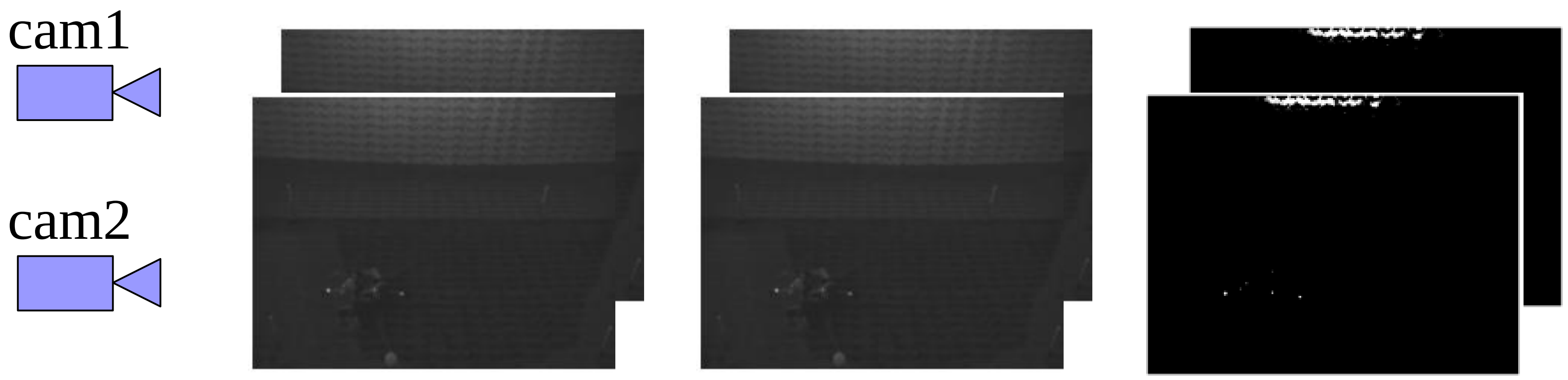
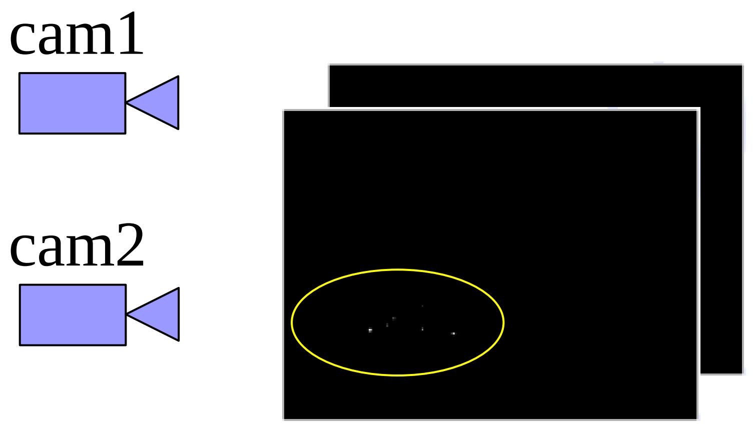


Image acquisition

Denoising

Segmentation

The overall example



$(x_1,y_1)_1$ $(x_1,y_1)_2$
$(x_2,y_2)_1$ $(x_2,y_2)_2$
… …
$(x_n,y_n)_1$ $(x_n,y_n)_2$

(x_1,y_1,z_1)
(x_2,y_2,z_2)
…
(x_n,y_n,z_n)

Shape recognition

Marker positions

3D reconstruction

Image function

- $f(x,y) \in \Re^2 x \Re$
- (x,y): spatial coordinates
- f(x,y): value of light intensity in a point

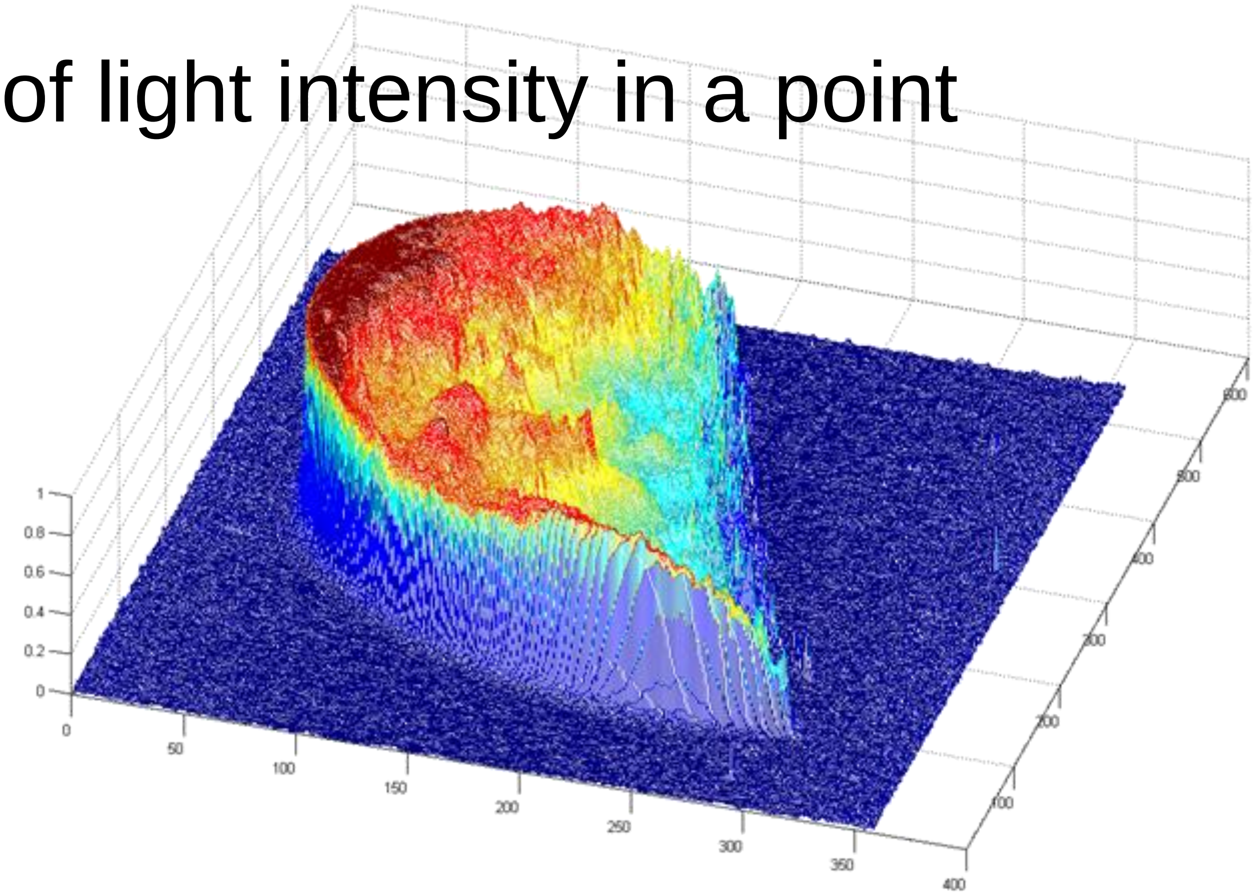


Image function and digital images

- $f(x,y) \in \Re^2 x \Re$
- (x,y): spatial coordinates
- f(x,y): value of light intensity in a point

- IMAGE SAMPLING: digitalizing spatial coordinates
- LIGHT INTENSITY (OR GREY LEVEL) QUANTIZATION: digitalizing in amplitude

- PIXEL: basic element of a digital image

Example of digital image

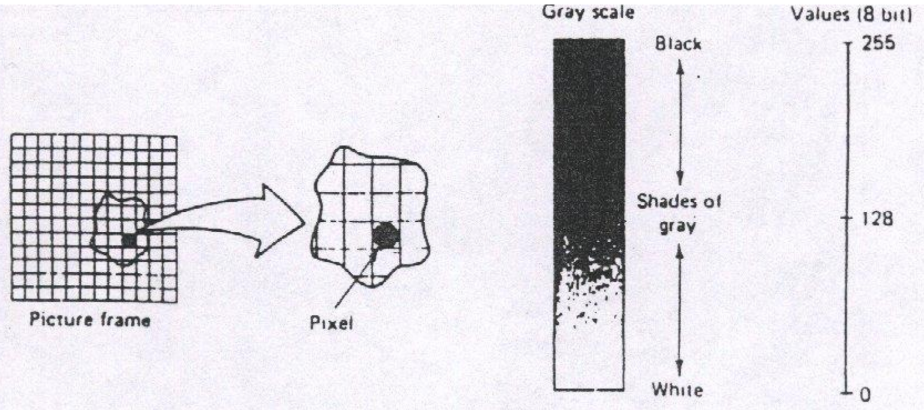
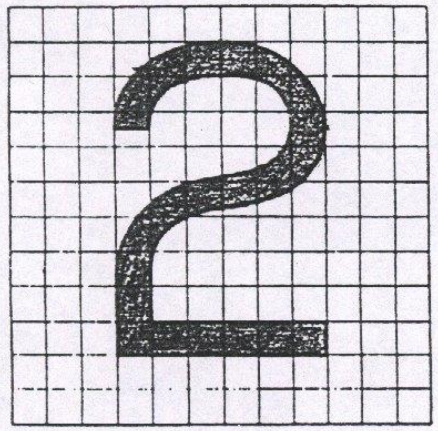
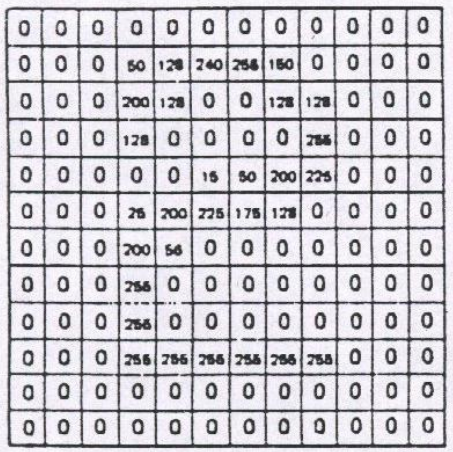


Figure 6-12 A picture frame is divided into picture elements, called pixels, for conversion to a gray-scale value.



(a)

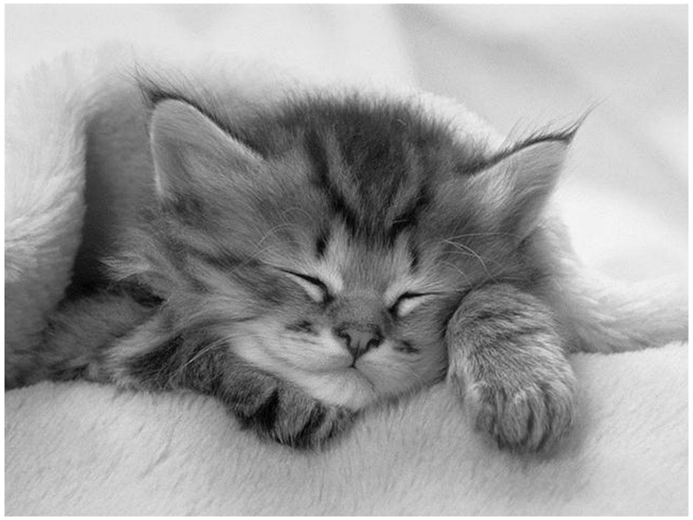
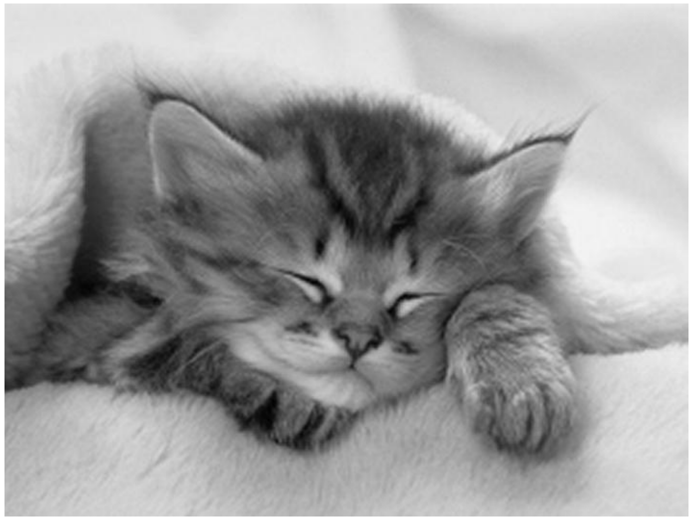
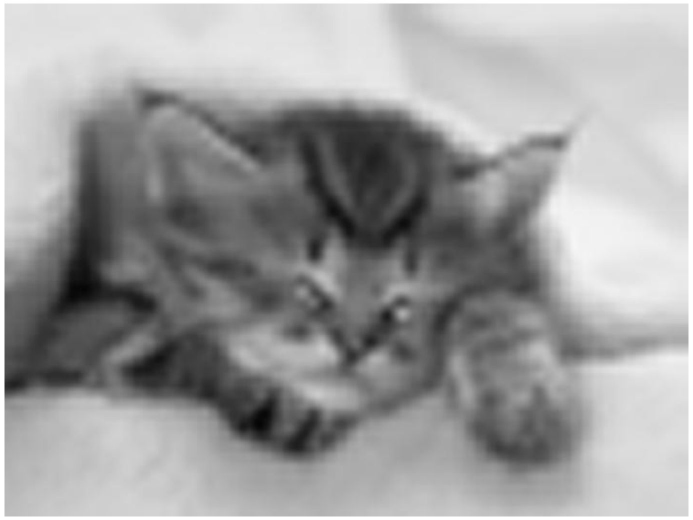
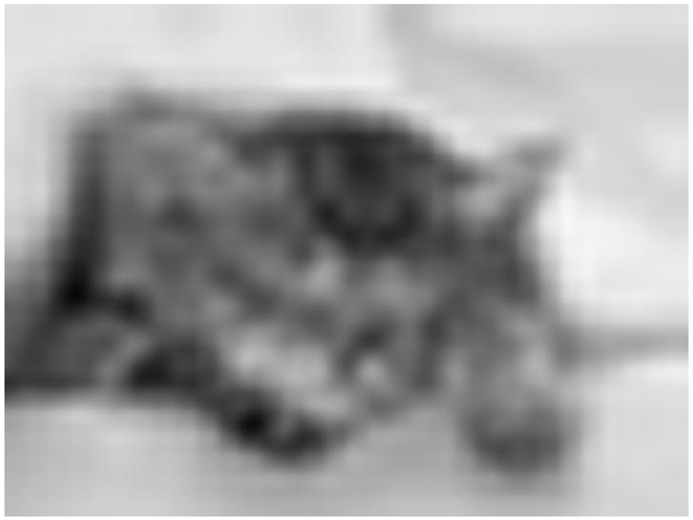


0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	50	128	240	256	160	0	0	0	0
0	0	0	200	128	0	0	128	128	0	0	0
0	0	0	128	0	0	0	0	256	0	0	0
0	0	0	0	0	15	50	200	225	0	0	0
0	0	0	25	200	225	175	128	0	0	0	0
0	0	0	200	56	0	0	0	0	0	0	0
0	0	0	256	0	0	0	0	0	0	0	0
0	0	0	256	0	0	0	0	0	0	0	0
0	0	0	256	256	256	256	256	256	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

(b)

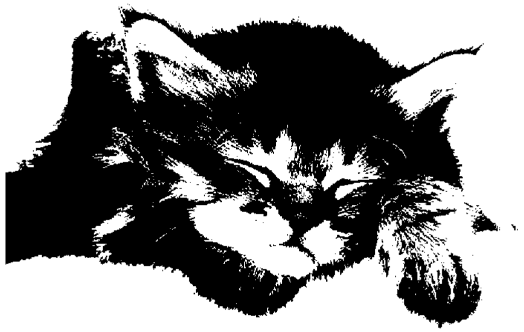
Figure 6-13 (a) a 12 × 12 pixel grid and (b) matrix for the number 2 (Example 6-1).

Space resolution

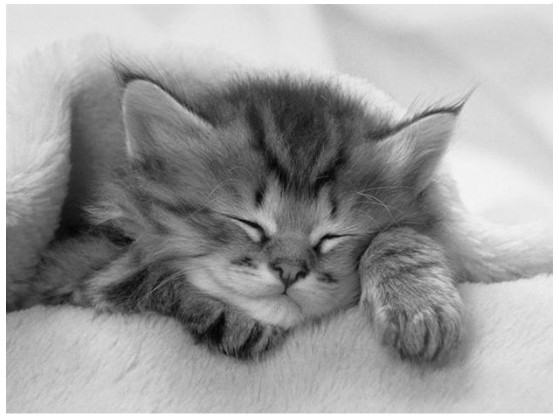





Color depth

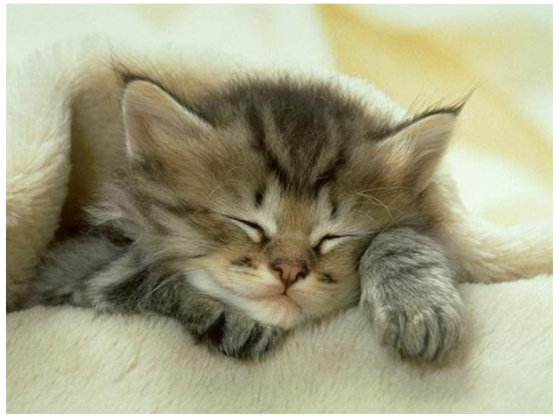
- Binary – 1 bit



- Grey levels – 8 bit



- True color – 24 bit



Main classes of image processing techniques

EARLY PROCESSING

- Processing of pixel values, at a pixel/local level

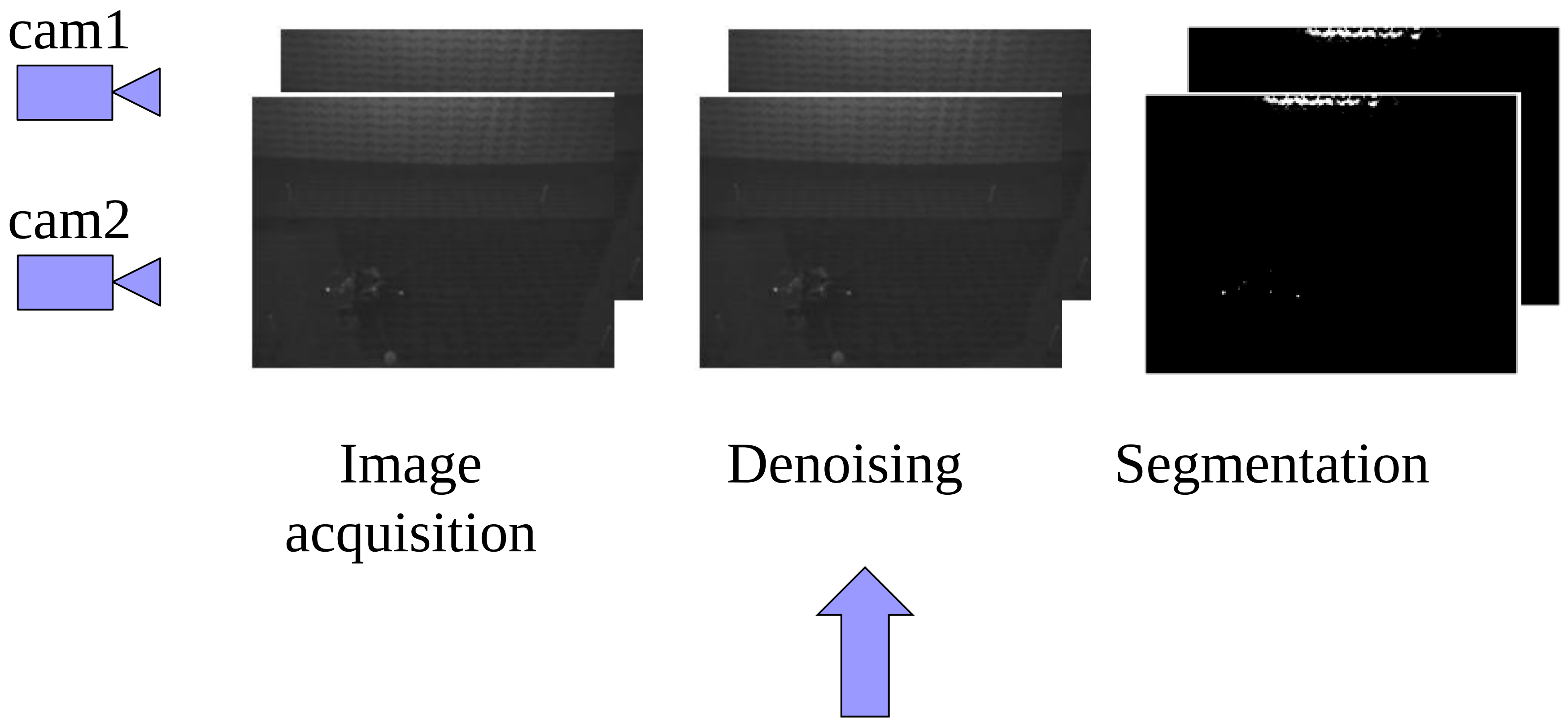
e.g.:

- FILTERING
- EDGE DETECTION

SEGMENTATION

- Detection of the parts that constitute a scene
- BOUNDARIES: elements of a segmented image based on **discontinuity**
- REGIONS: elements of a segmented image based on **uniformity**

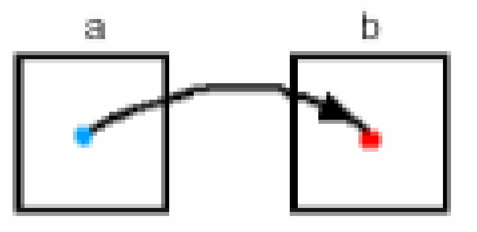
Where we are



Digital image operators

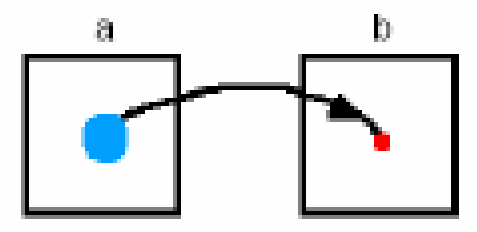
- **Pixel operators**

  the output value of pixel (i,j) depends on the input value of pixel (i,j), only



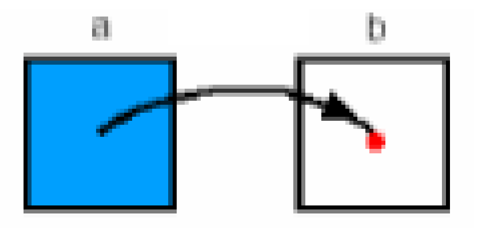
- **Local operators**

  the output value of pixel (i,j) depends on the input value of a neighborhood of pixel (i,j), only

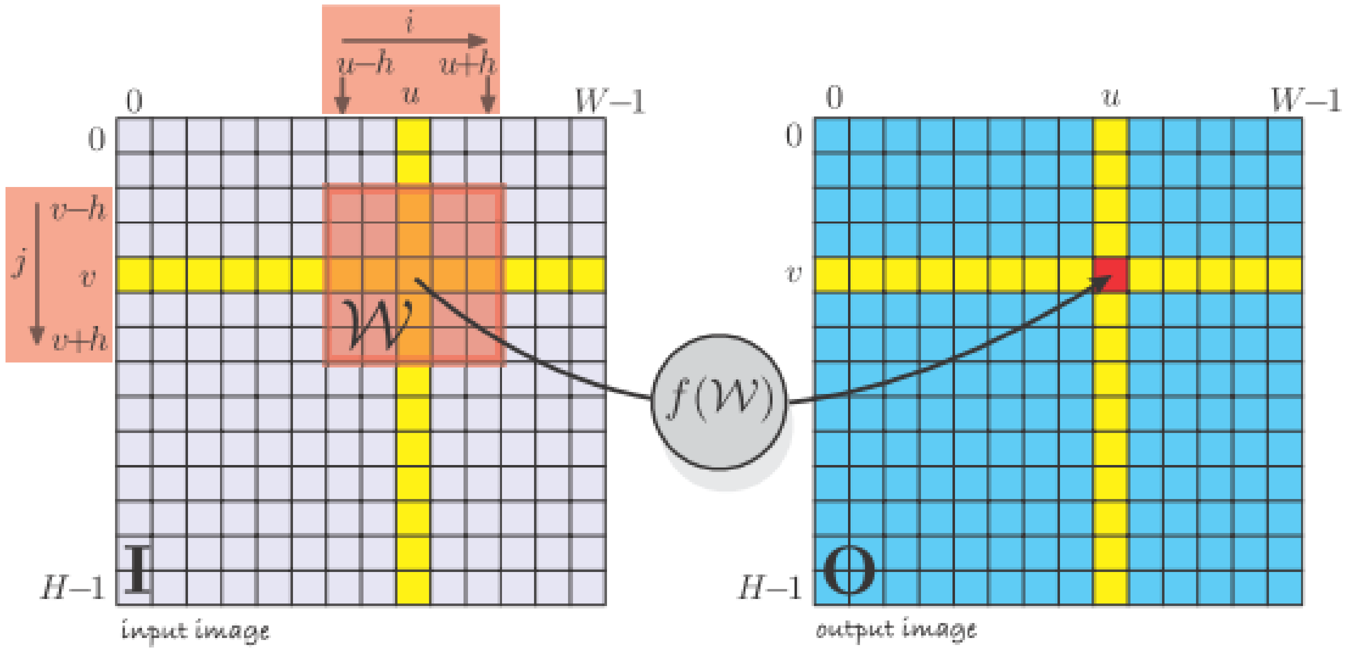


- **Global operators**

  the output value of pixel (i,j) depends on the input value of all the image pixels



Local operators



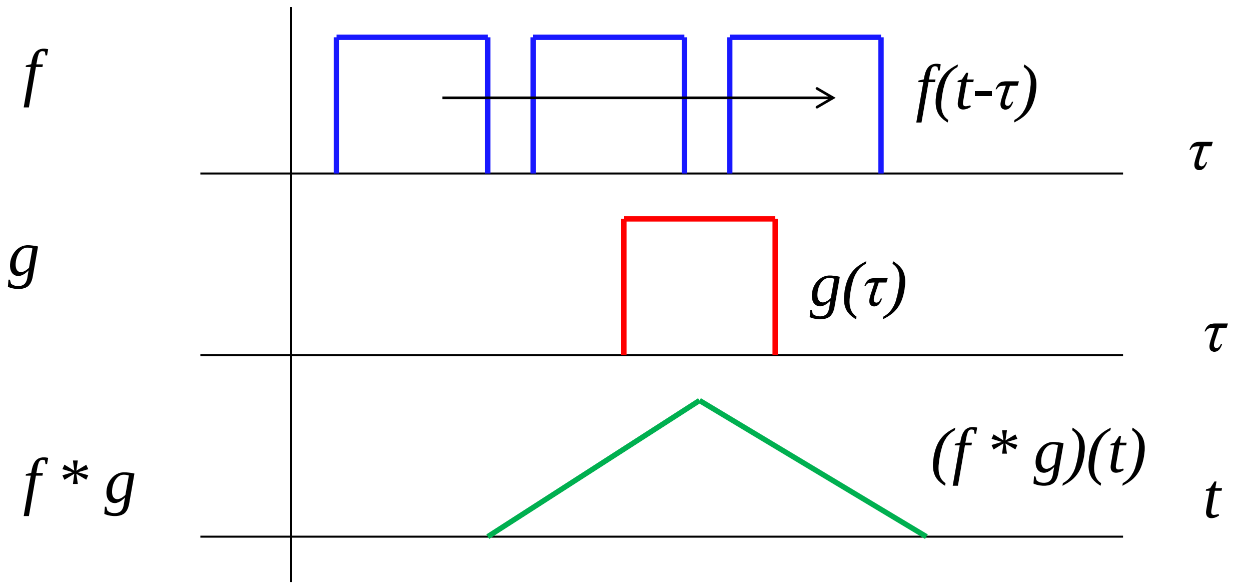
Filtering techniques:

- smoothing
    - mean, median, gaussian

Convoluzione

E’ una delle operazioni più utilizzate:

$$(f * g)(t) := \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

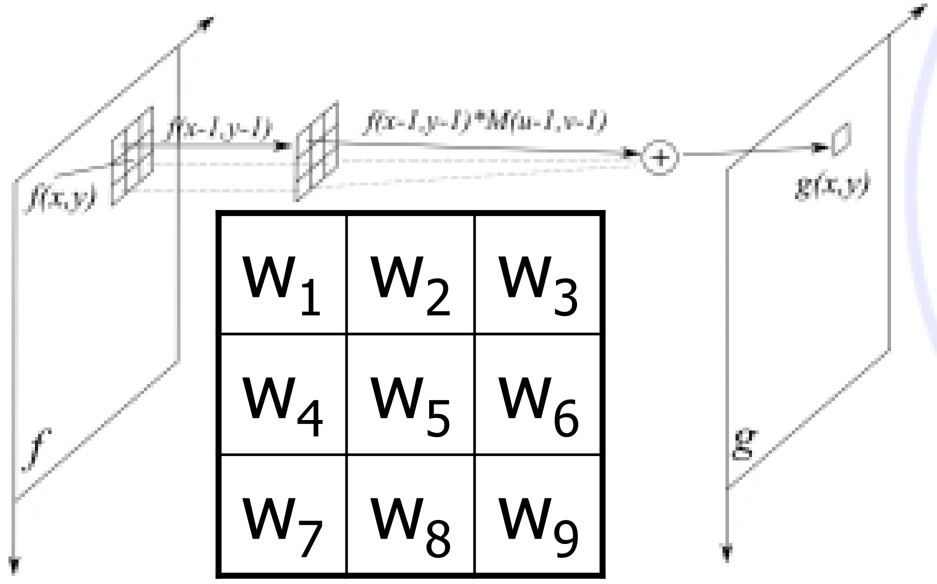


Bidimensionale

Convoluzione

E’ una delle operazioni più utilizzate:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} f(x+s, y+t) \cdot M(u+s, v+t)$$



w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Convolution mask

Smoothing

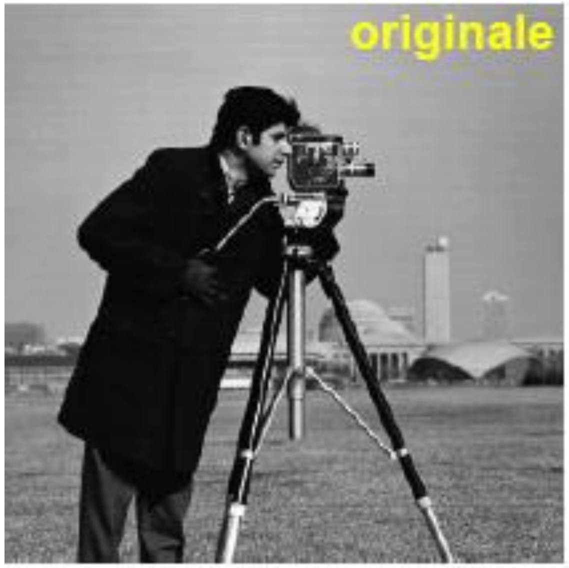
$$g(x,y) = 1/P \sum_{n,m \in S} f(n,m)$$

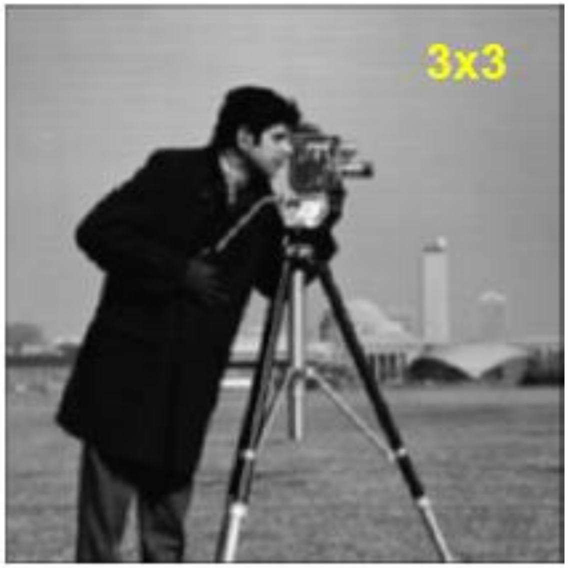
S: neighborhood of (x,y)

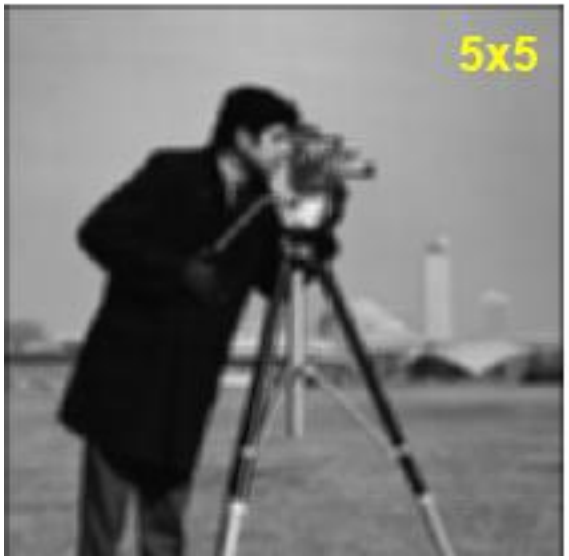
P: number of pixels in S

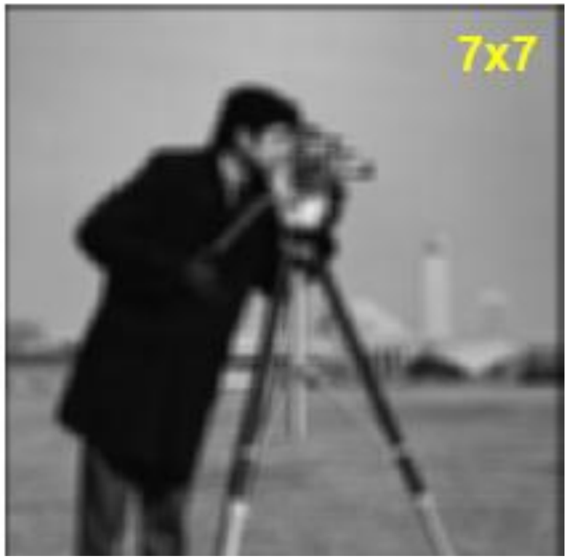
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Example: Blurring

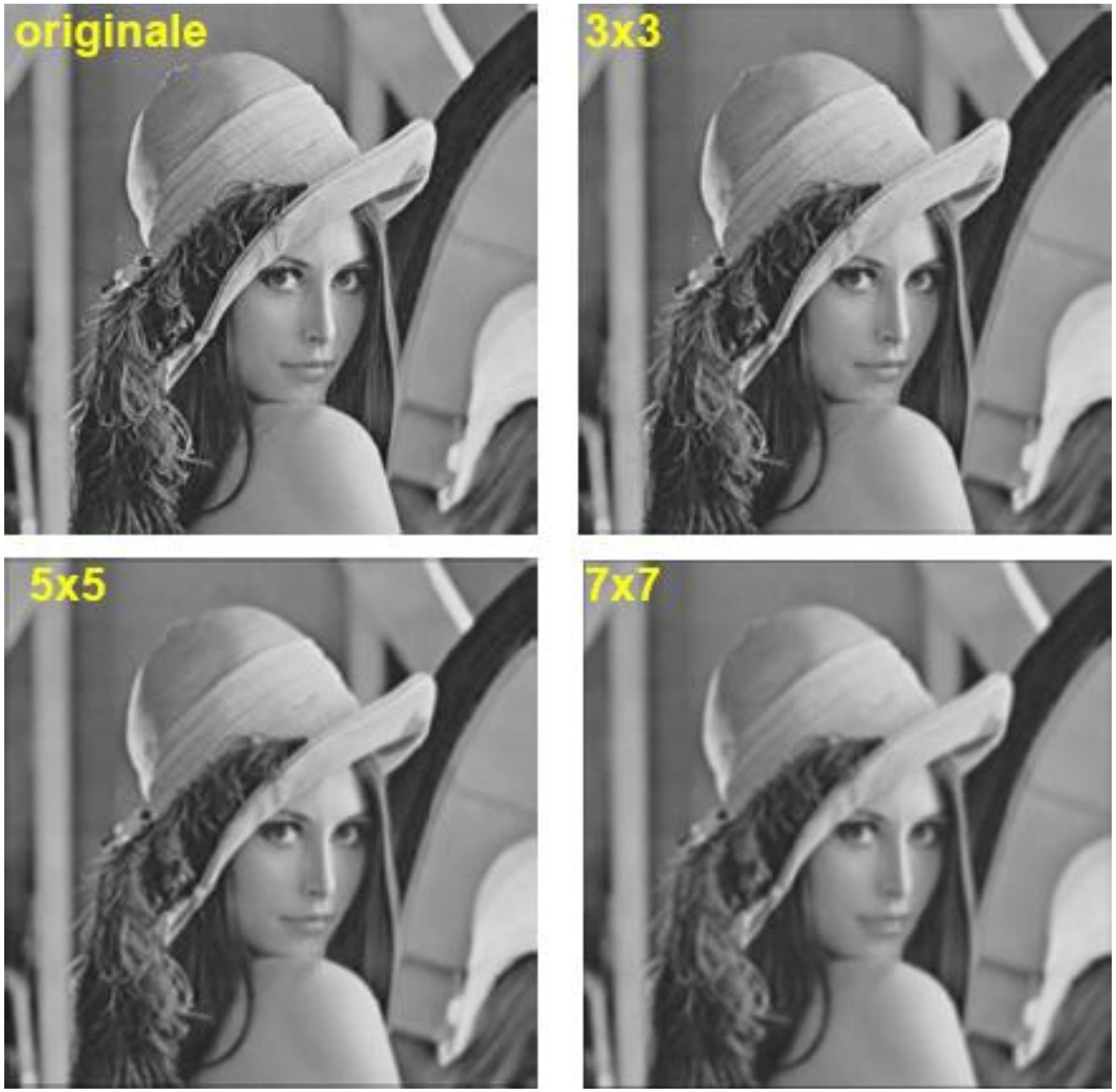






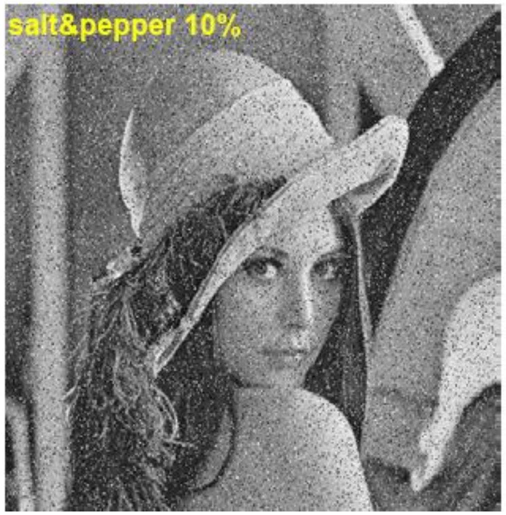


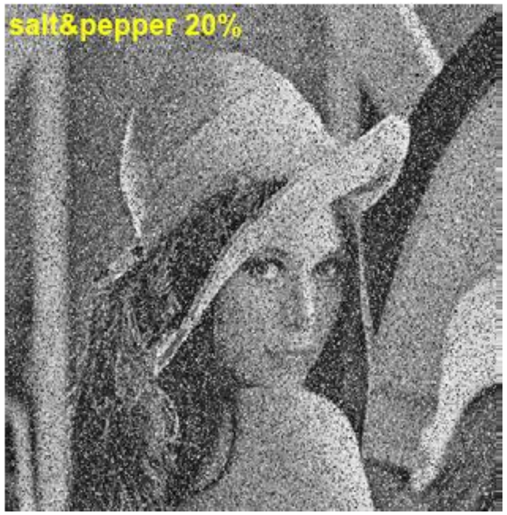
Example: Blurring



Example of noise cleaning

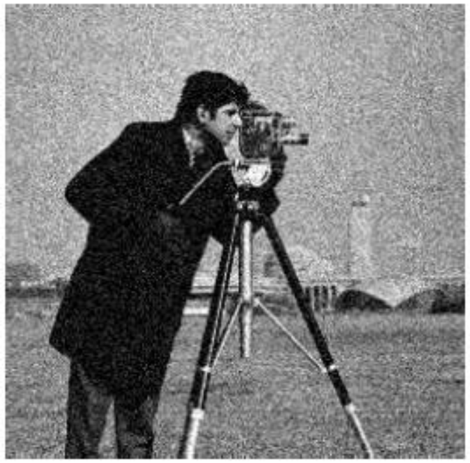
- **impulsive noise** (*salt & pepper*): characterized by the ratio of image modified by noise (in %)



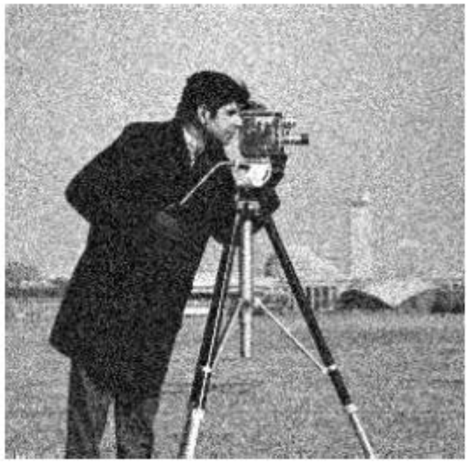


- **white gaussian noise**: characterized by the average and variance

media=0 varianza=0.01



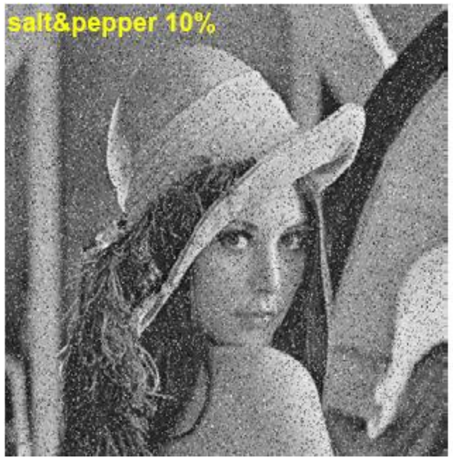
media=0 varianza=0.1

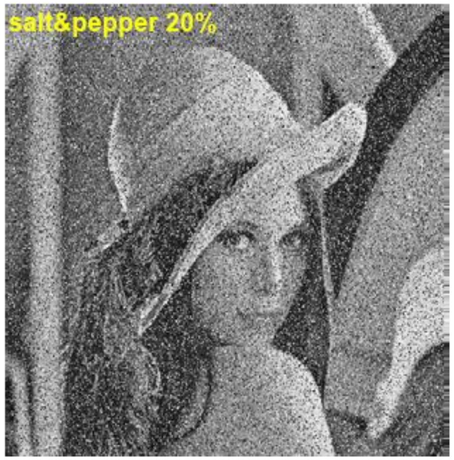


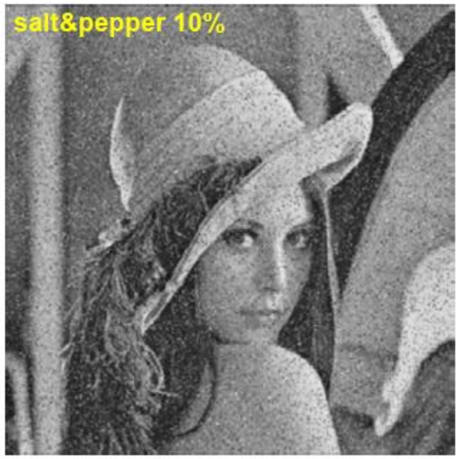
Impulsive noise and smoothing

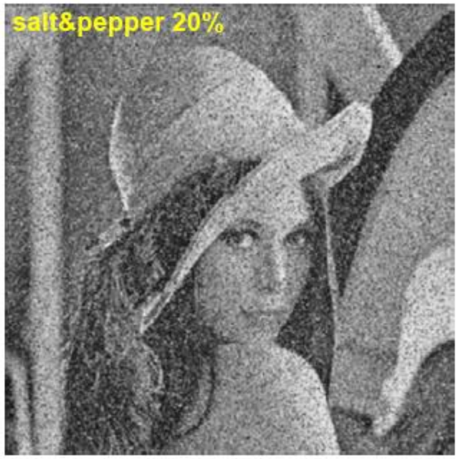
Original noisy image

Smoothing 3x3

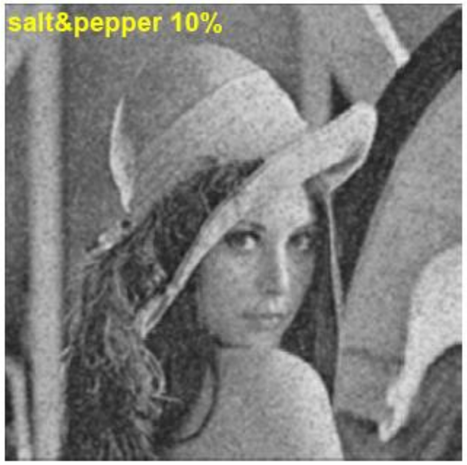


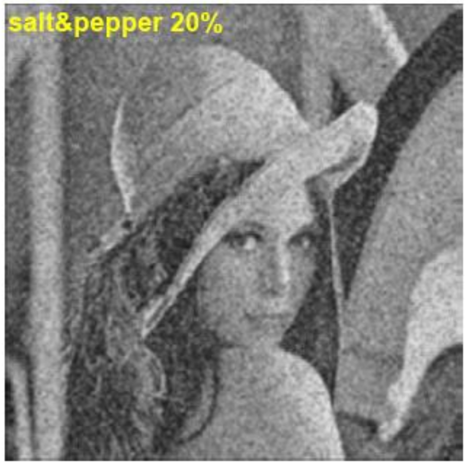




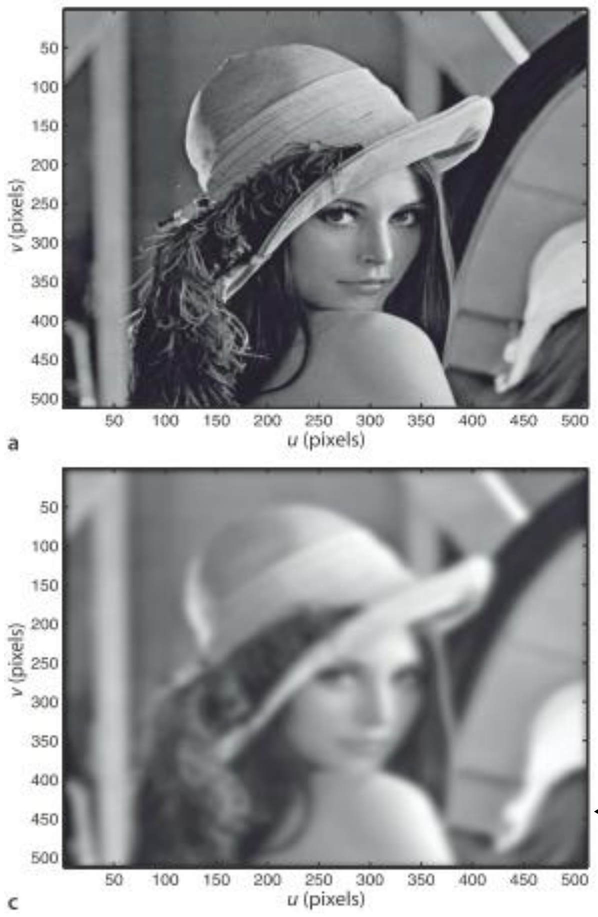


Smoothing 5x5





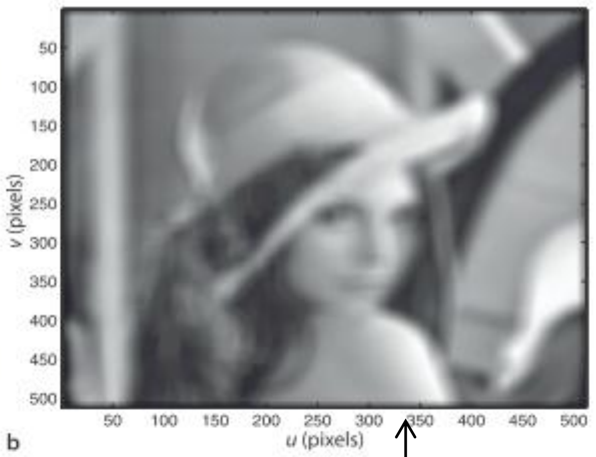
Gaussian filter



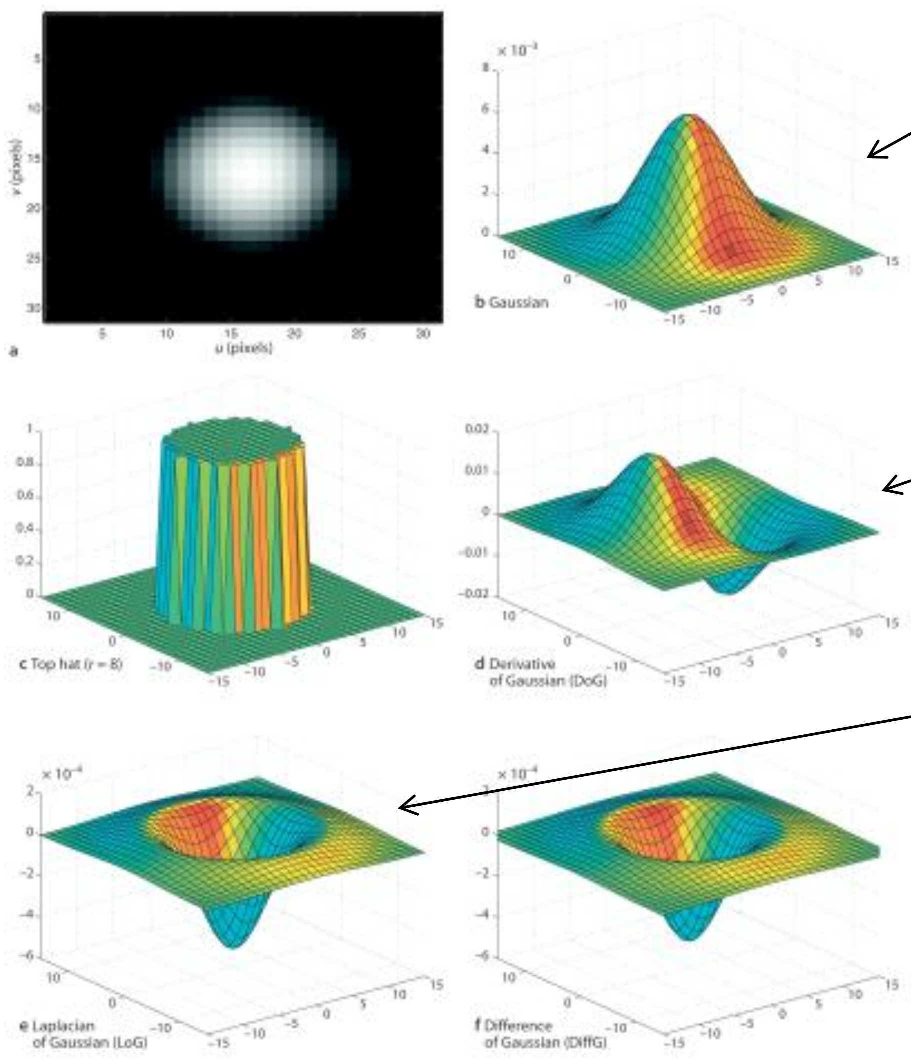
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

It induces ringing

$$G(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



Some common kernel



Gaussian

Derivative of Gaussian (DoG)

Laplacian of Gaussian (LoG)

Median filter

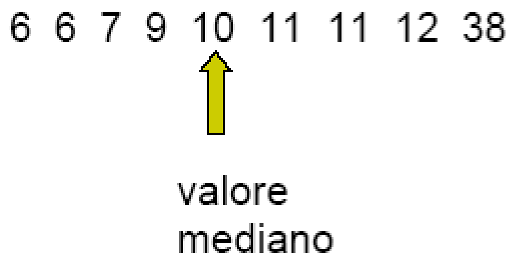
g(x,y) = median of (x,y)

Median M of a set of values:

the value by which half values of the set are lower than M and half values are higher

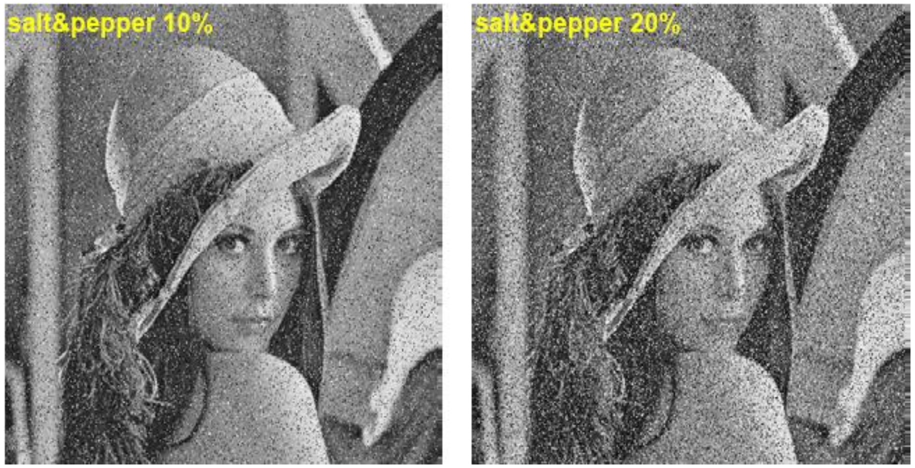
- Example:

7	10	12
6	38	11
9	11	6

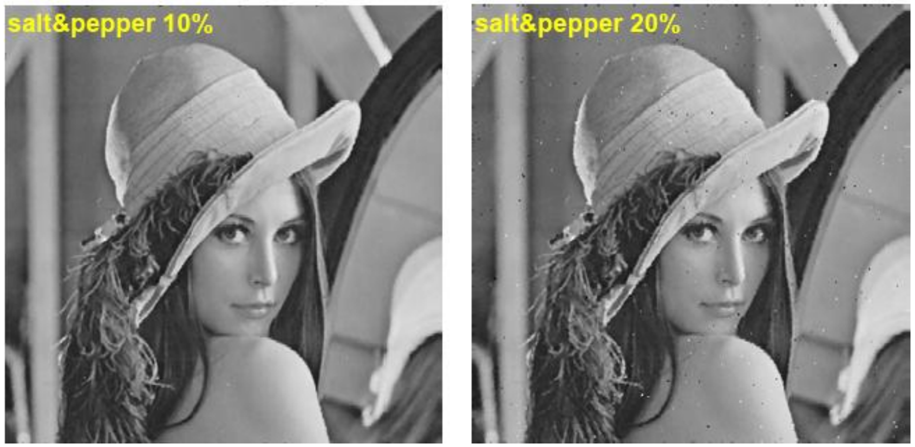


Example of application of a Median filter

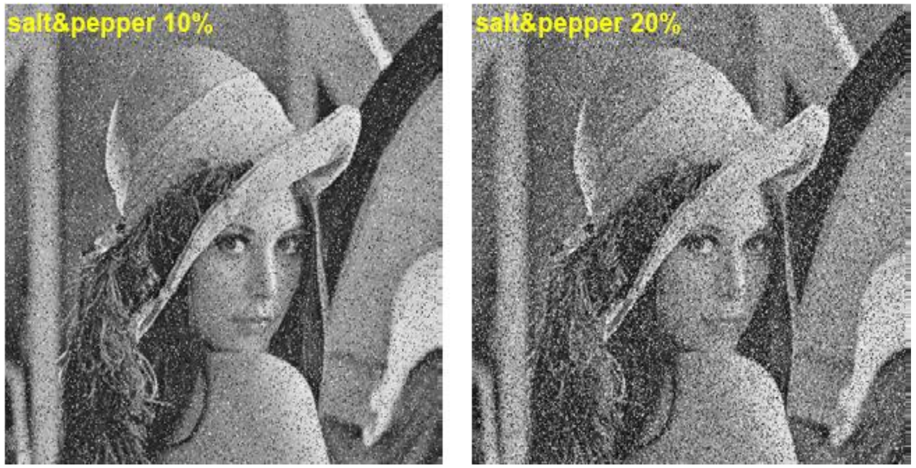
Original noisy image



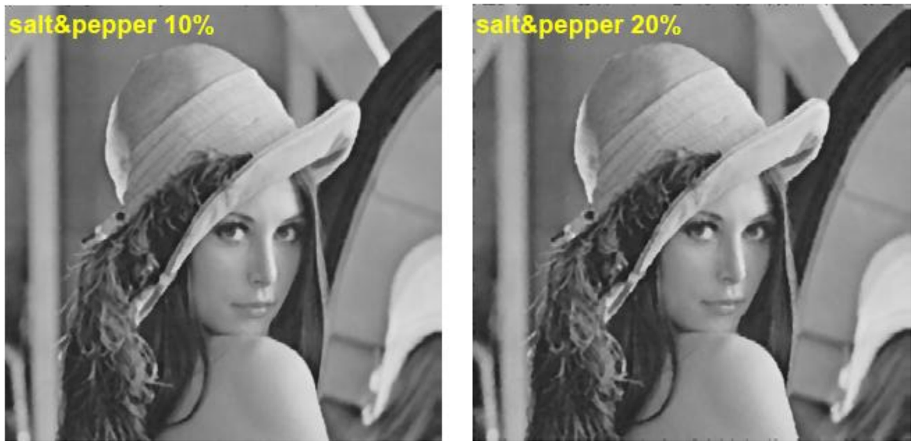
Filtro mediano 3x3

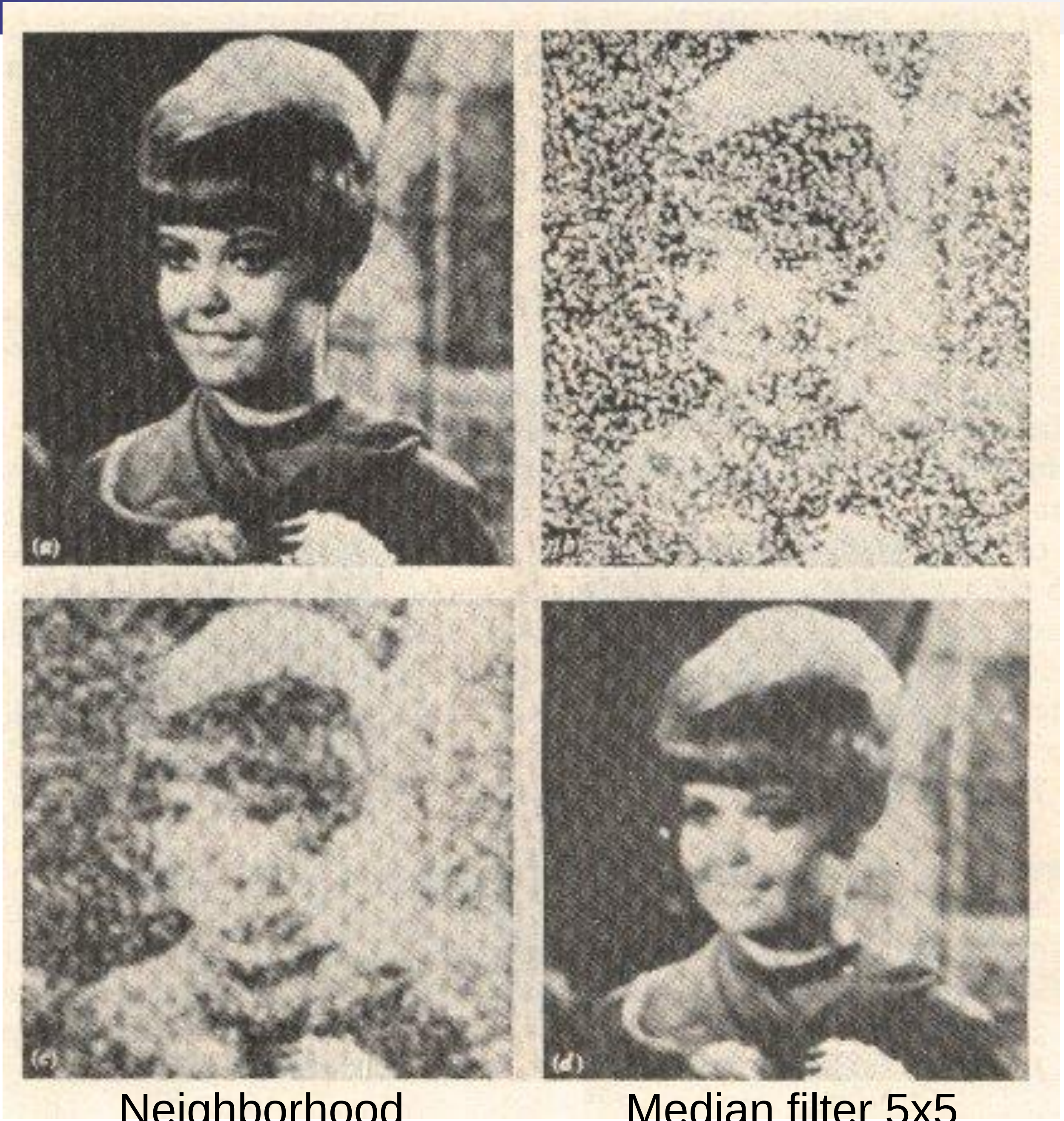


Original noisy image



Filtro mediano 5x5



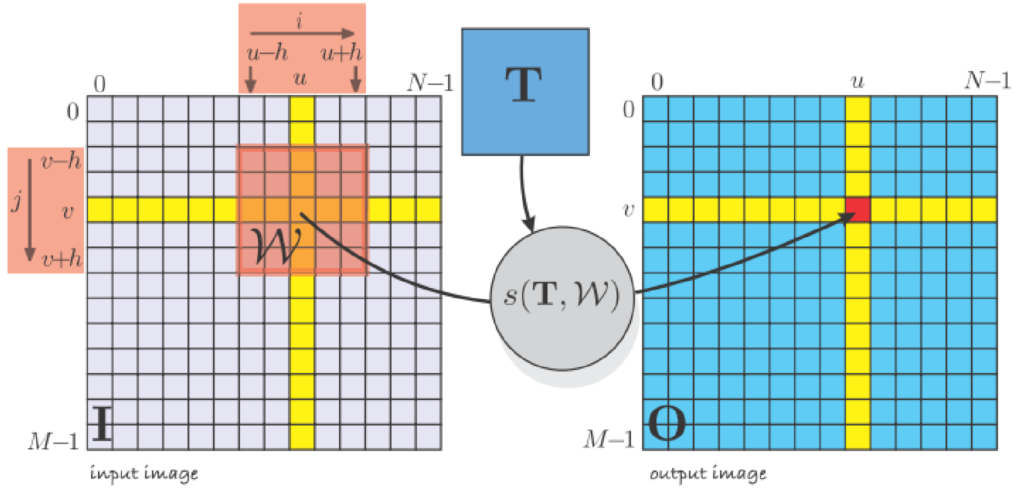


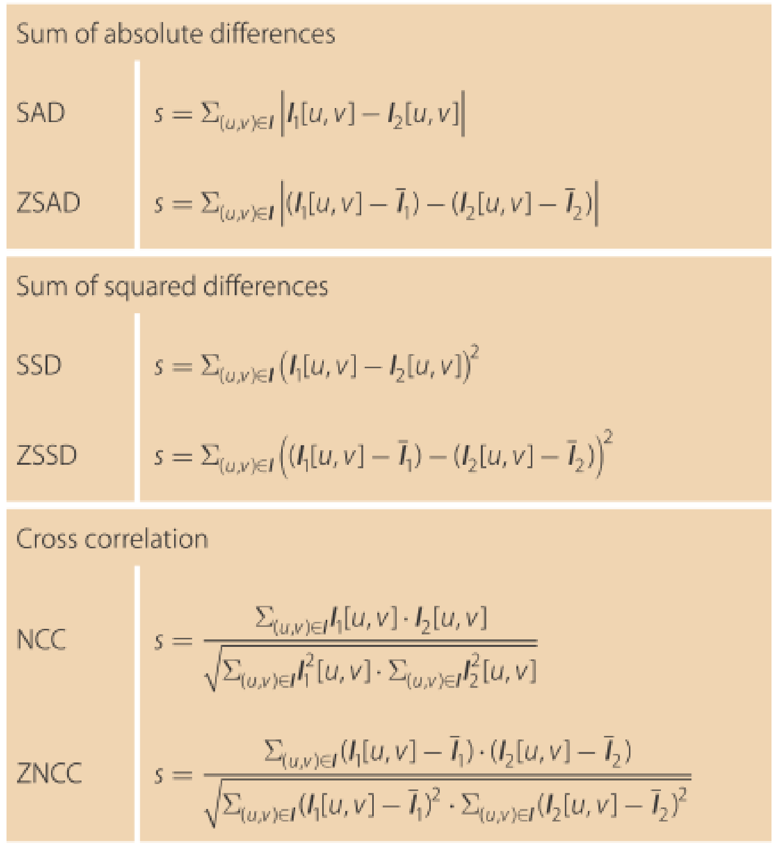
Neighborhood average 5x5

Median filter 5x5

Template matching

We will consider that the kernel is an image or a part of an image and such a kernel is referred to as a template. In template matching we wish to find which parts of the input image are most similar to the template.





Sum of absolute differences	
SAD	$s = \Sigma_{(u,v)\in I} \left\| I_1[u,v] - I_2[u,v] \right\|$
ZSAD	$s = \Sigma_{(u,v)\in I} \left\| (I_1[u,v] - \bar{I}_1) - (I_2[u,v] - \bar{I}_2) \right\|$
Sum of squared differences	
SSD	$s = \Sigma_{(u,v)\in I} \left(I_1[u,v] - I_2[u,v] \right)^2$
ZSSD	$s = \Sigma_{(u,v)\in I} \left((I_1[u,v] - \bar{I}_1) - (I_2[u,v] - \bar{I}_2) \right)^2$
Cross correlation	
NCC	$s = \dfrac{\Sigma_{(u,v)\in I} I_1[u,v] \cdot I_2[u,v]}{\sqrt{\Sigma_{(u,v)\in I} I_1^2[u,v] \cdot \Sigma_{(u,v)\in I} I_2^2[u,v]}}$
ZNCC	$s = \dfrac{\Sigma_{(u,v)\in I} (I_1[u,v] - \bar{I}_1) \cdot (I_2[u,v] - \bar{I}_2)}{\sqrt{\Sigma_{(u,v)\in I} (I_1[u,v] - \bar{I}_1)^2 \cdot \Sigma_{(u,v)\in I} (I_2[u,v] - \bar{I}_2)^2}}$

The Z-prefix indicates that the measure accounts for zero-offset or the difference in mean of the two images

Template matching: example

- Similarity measure for images, to compare an image function f(x) with a template t(x)

$$R_{ft}(y) = \Sigma_x f(x)t(x-y)$$

Template

1	1	1
1	1	1
1	1	1

(corner evaluated)

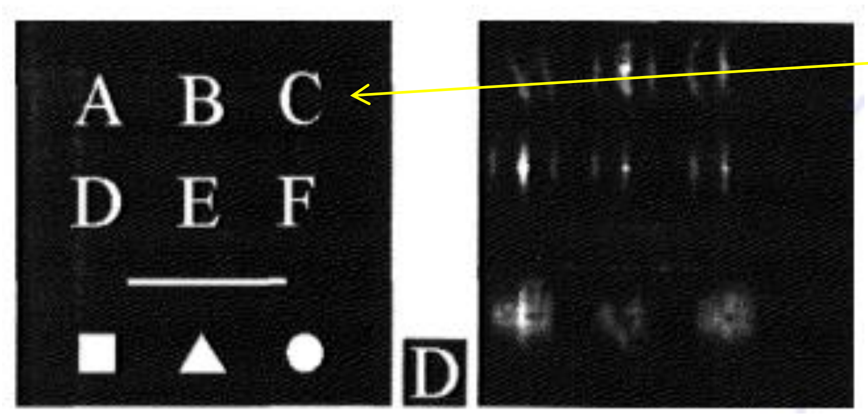
Image

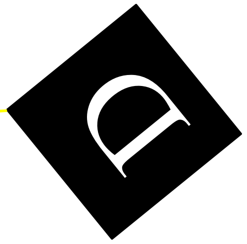
1	1	0	0	0
1	1	1	0	0
1	0	1	0	0
0	0	0	0	0
0	0	0	0	8

Correlation

7	4	2	x	x
5	3	2	x	x
2	1	9	x	x
x	x	x	x	x
x	x	x	x	x

Correlation matching



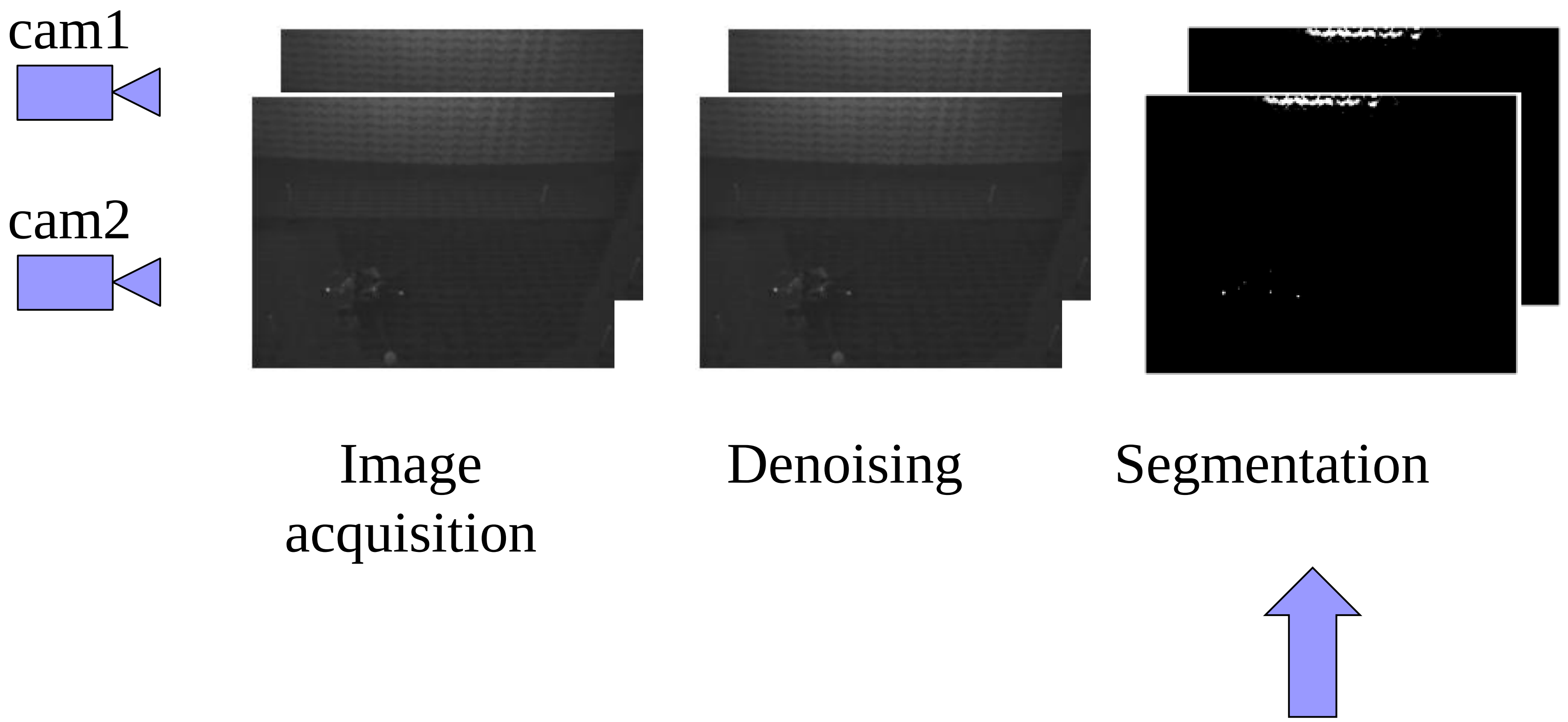


Invariant features!

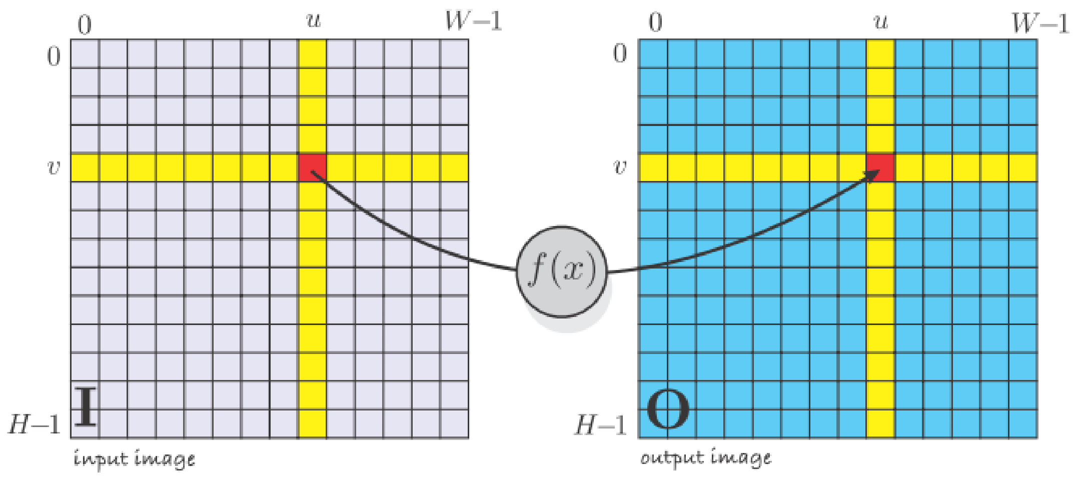
Scale-invariant feature transform

SIFT features are located at scale-space maxima/minima of a Difference of Gaussian function.

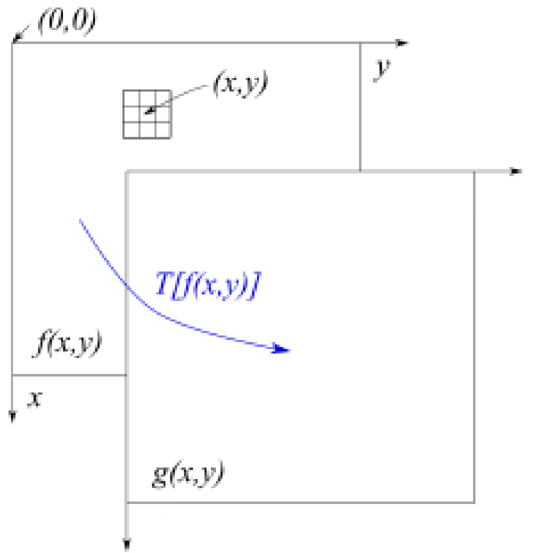
Where we are



Pixel operators (Monodic)



$$g(x, y) = T[f(x, y)]$$

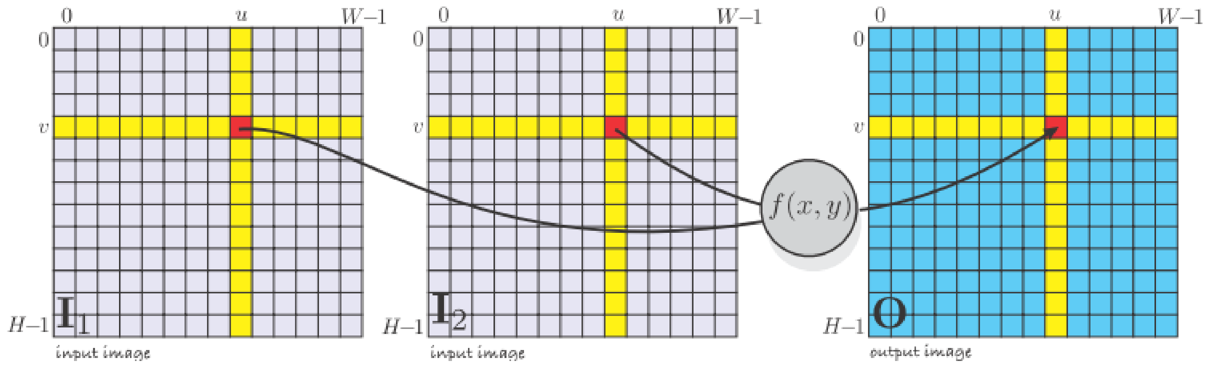


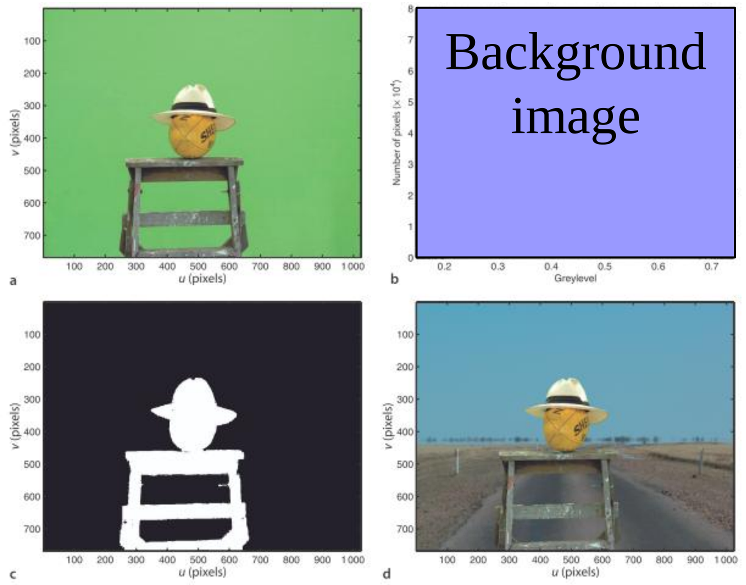
Defining *r = f(x, y)* as the value of the *(x, y)* pixel for the image *f(x,y)*, and *s = g(x, y)* as the value of the *(x, y)* pixel for the image *g(x,y)*. Thus is possible to write a transformation as:

$$s = T[r]$$

with *L* grey levels (values that the pixels can assume).

Diadic Operations



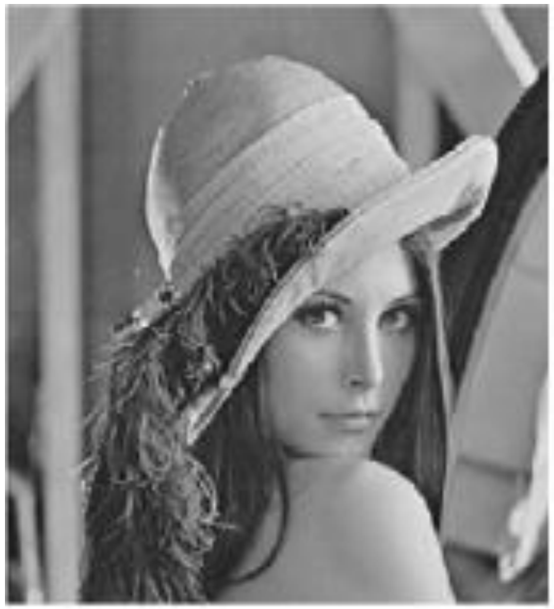
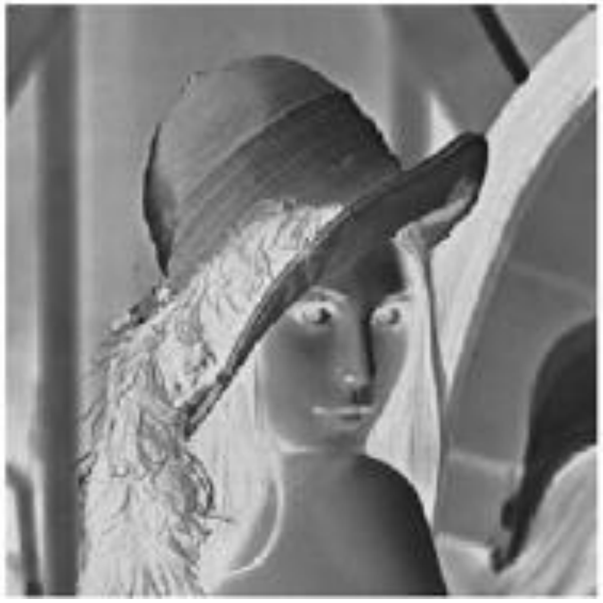


```
if (OBJ_img==green)
        copy(BG_img)
else
        copy(OBJ_img)
```

Example of pixel operator: grey level inversion

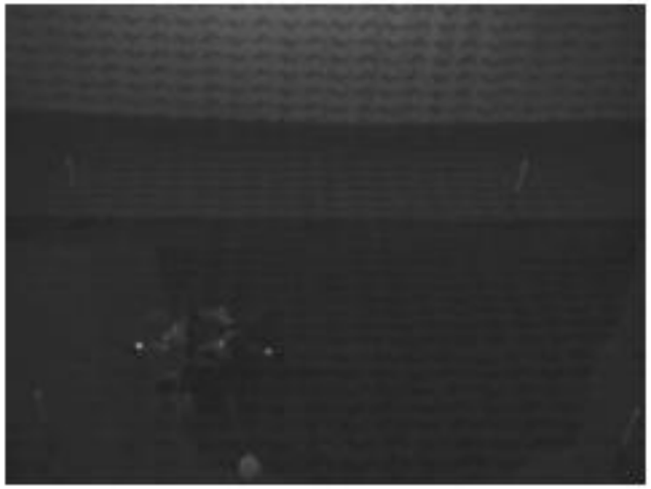
$$g(x,y) = 255 - f(x,y)$$

Provides the "negative" image

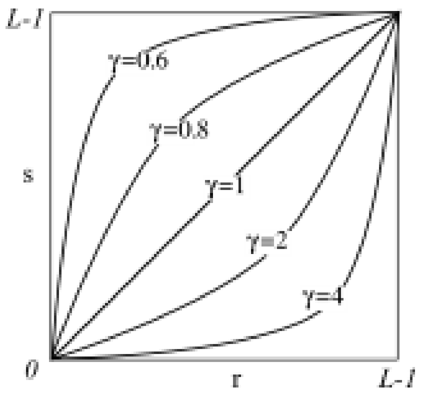
Example of pixel operator: contrast enhancement

Image enhancement is application driven.

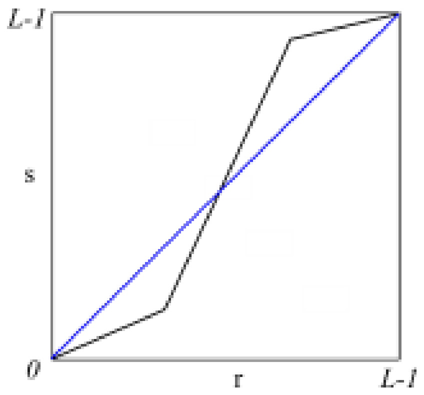


Common transformations

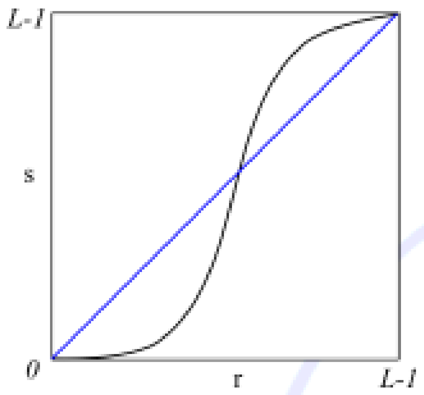
- Log transformation
- Sigmoid function transformation
- Piecewise linear transformation function



(a) Power law



(b) Piecewise

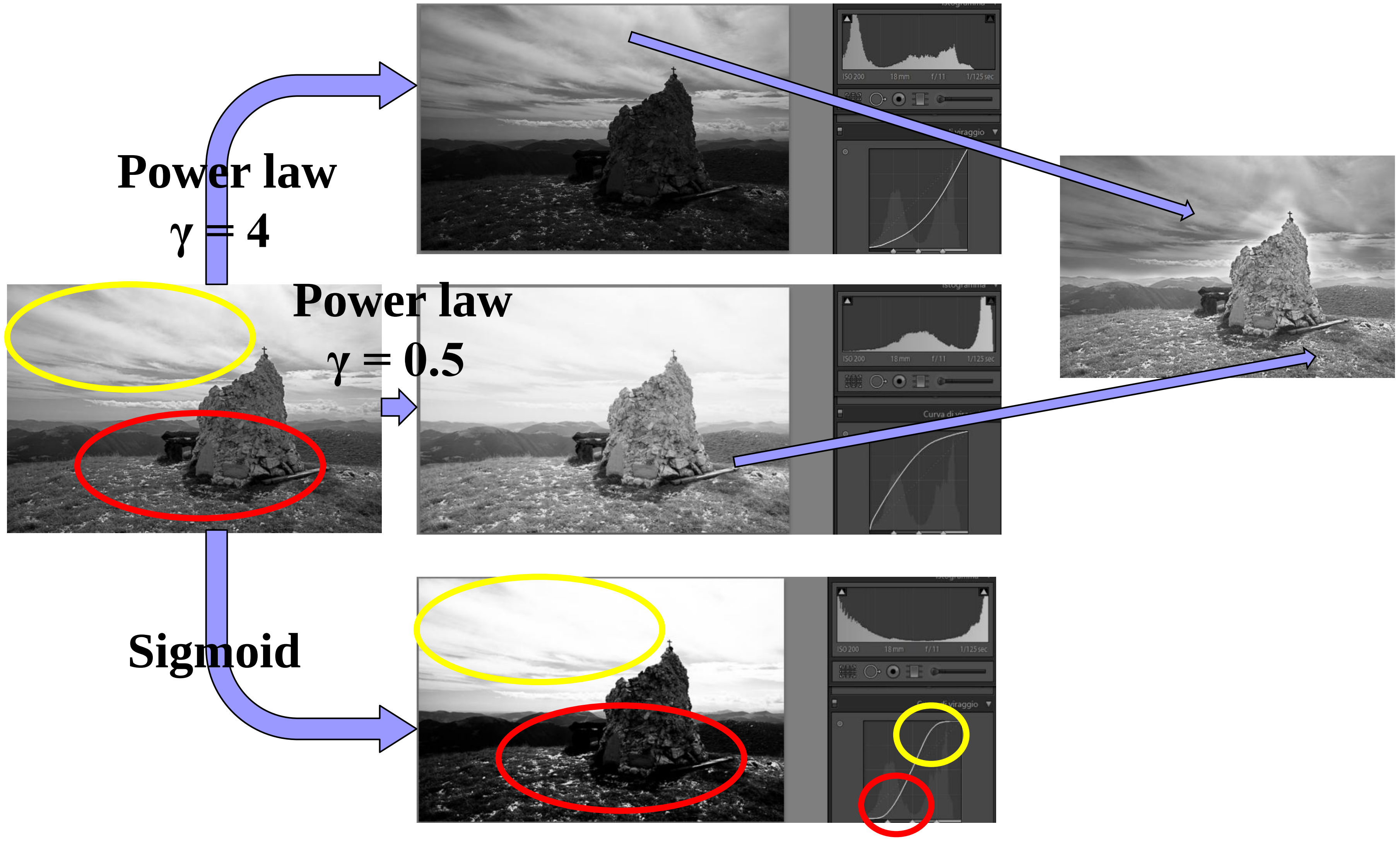


(c) Sigmoid

Figure: Transformations

Power law	lighten or darken
Piecewise	flexible
Sigmoid	enhance the contrast

Examples


Power law
γ = 4
Power law
γ = 0.5
Sigmoid
Istogramma
ISO 200
18 mm
f/11
1/125 sec
Curva di viraggio

$$s(r) = c \cdot r^{\gamma}$$

$$s(r) = \begin{cases} c_1 \cdot r & 0 \leq r < r_{min} \\ c_2 \cdot r & r_{min} \leq r < r_{max} \\ c_3 \cdot r & r_{max} \leq r < L - 1 \end{cases}$$

$$s(r) = \frac{c_1}{1 + e^{-r}}$$

- power law
- piecewise
- sigmoid

Each function requires parameters definition

So consider this code

```
\%Power-Law Transformation: apply the power-law transform for
\%gamma = 0.05,0.2,0.67,1.5,2,5 and comment your results

f = imread('pout.tif');
[m n]=size(f);
c = 1;
y=input('Gamma value:');
for i=1:m
for j=1:n
s(i,j)=c*(f(i,j)^y);
end
end
figure,imshow(f),title('Original Image');
figure,imshow(s),title('After Power-Law Transformation');
```

$$\begin{aligned} L &= 21 \\ c &= 1 \\ \gamma &= 2 \\ s(r) &= c \cdot r^{\gamma} \end{aligned}$$

$$\begin{aligned} s(1) &= 1 \\ s(2) &= 2^2 = 4 \\ s(3) &= 3^2 = 9 \\ s(4) &= 4^2 = 16 \\ s(5) &= 5^2 = 25 \\ \ldots &= \ldots \\ s(20) &= 20^2 = 400 \end{aligned}$$

???

We have only 21 levels, but:

$$s(5) = 5^2 = 25$$

We need to remap the output between $[0, L-1]$:

$$\frac{s'}{s} = \frac{20}{400}$$

$$\frac{20}{400} = \frac{L-1}{(L-1)^{\gamma}} = (L-1)^{1-\gamma}$$

$$s' = (L-1)^{1-\gamma} s = c \cdot s = c \cdot r^{\gamma}$$

Thus c is related to L and γ.

Image enhancement is often a preliminary step to another pixel operator:

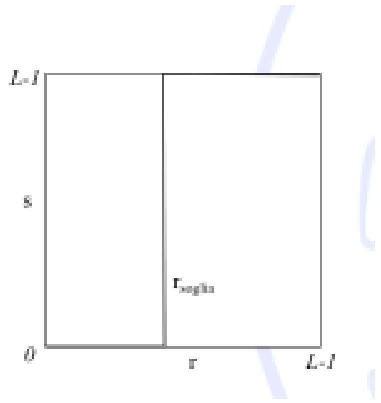
Example of pixel operator: thresholding

Transformation of the image in a binary image, through comparison with a threshold T:

$$g(x,y) = \begin{cases} 255 \; if \; f(x,y) > T \\ 0 \; if \; f(x,y) \leq T \end{cases}$$

- T can be constant or variable with respect to x, y, f(x,y) or other local properties
- T can be found empirically or with statistical techniques

$$s(r) = \begin{cases} 0 & 0 \le r < r_{threshold} \\ 255 & r_{threshold} \le r < L-1 \end{cases}$$



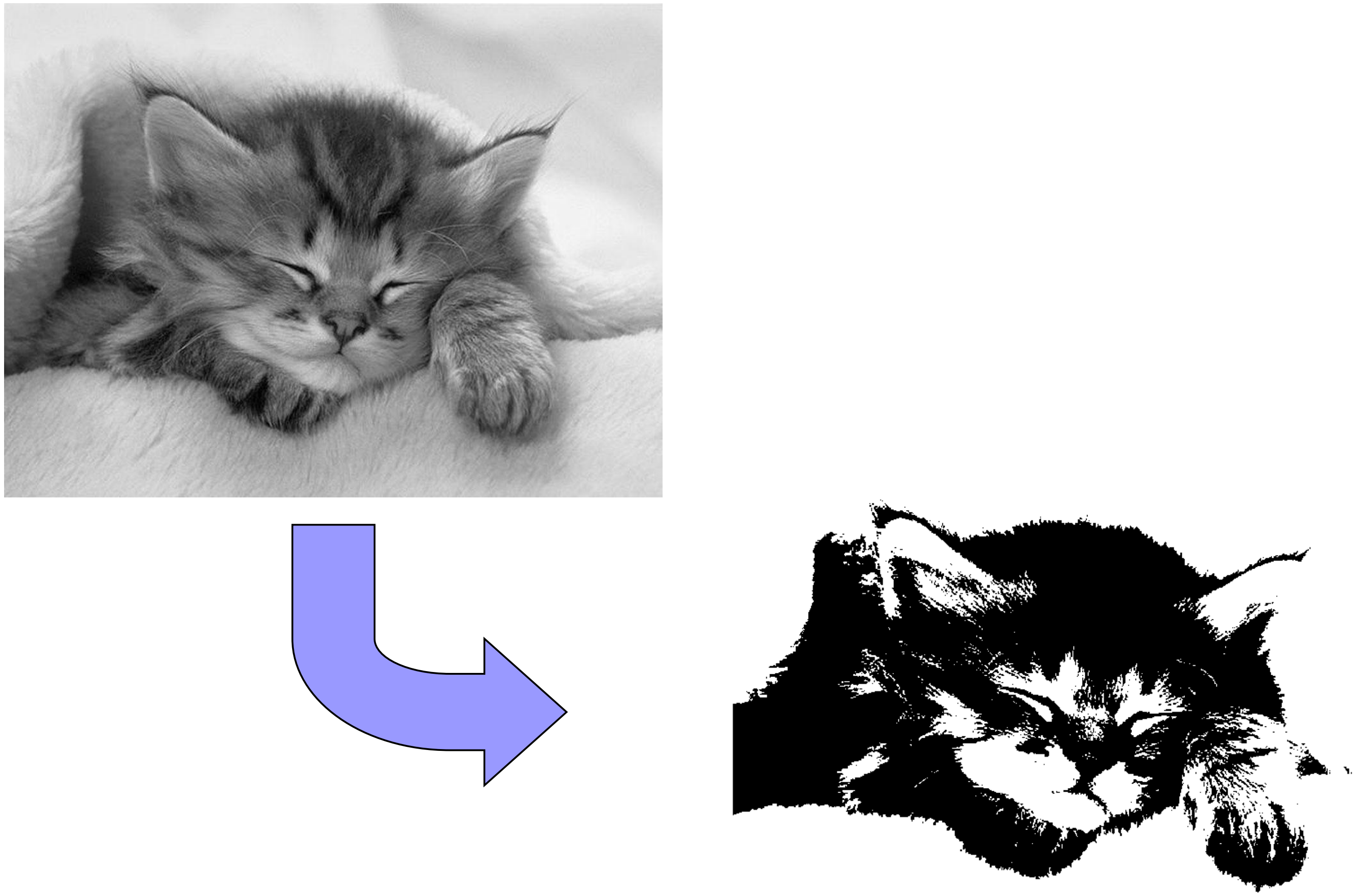
Threshold with and without contrast enhancement

Detect the key features

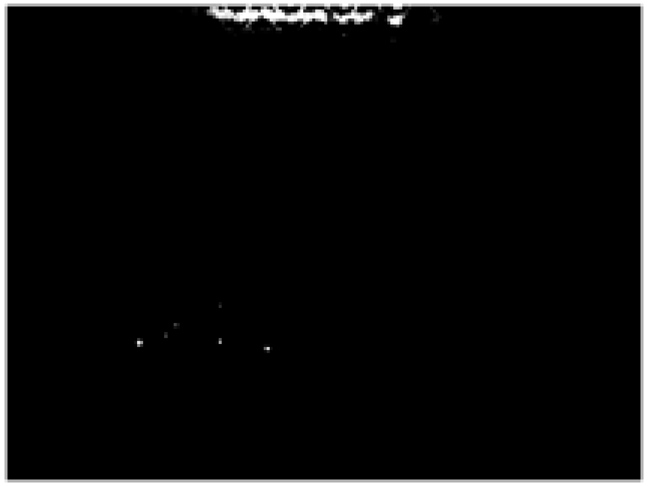
• contrast enhancement "increase the distance" among objects of interest from the background

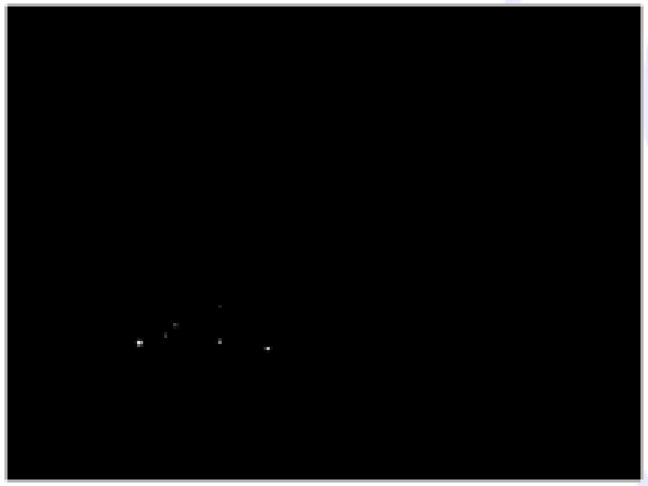
• threshold identifies the objects and the background

Examples



Original image


(a) Threshold

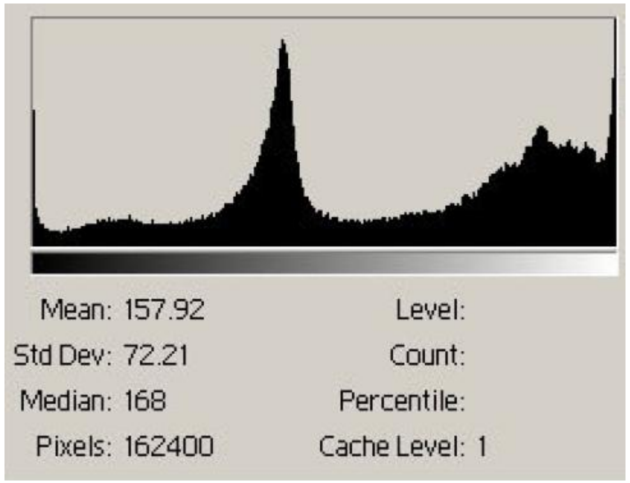

(b) Contrast–threshold

The problem is: how is possible to select the correct threshold?

Image histogram

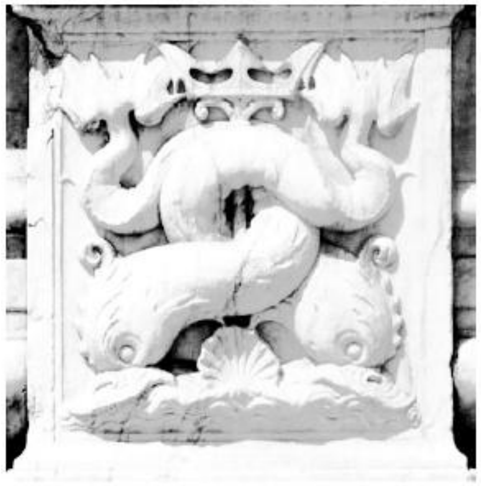
- Distribution of gray level
- For each grey level, the histogram gives the number of pixels with that grey level.
- For an image I[m,n]:

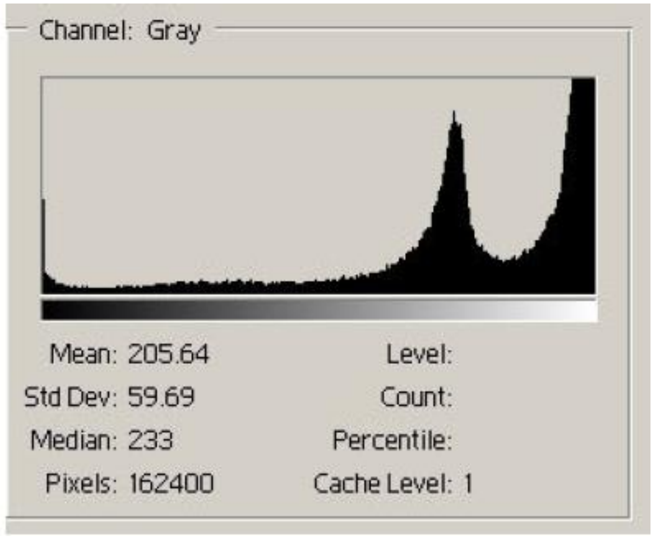
  H(k)= number of pixels with value k
- The sum of all H is exactly mxn



```
 Mean: 157.92           Level:
Std Dev: 72.21          Count:
 Median: 168       Percentile:
  Pixels: 162400  Cache Level: 1
```

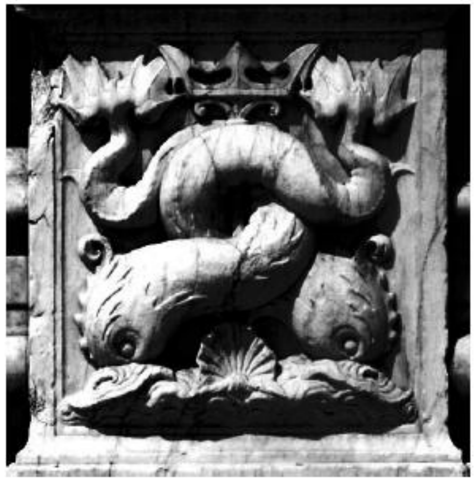
Examples of istogram

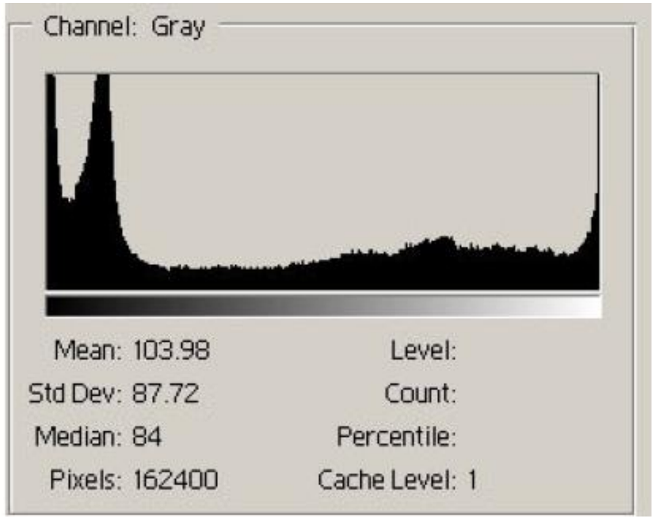




Channel: Gray

Mean: 205.64	Level:
Std Dev: 59.69	Count:
Median: 233	Percentile:
Pixels: 162400	Cache Level: 1



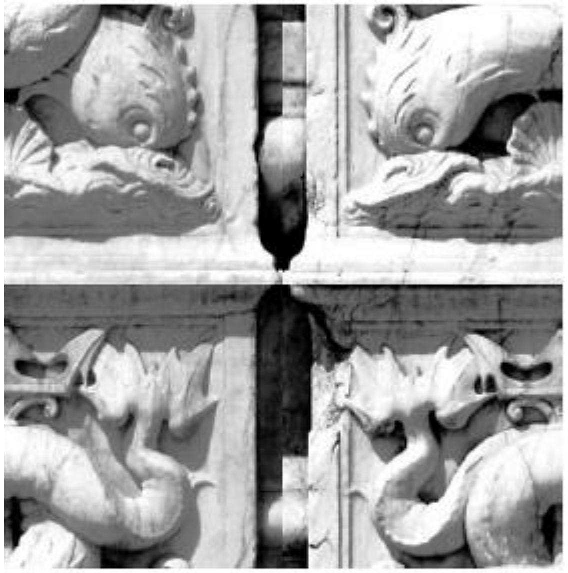


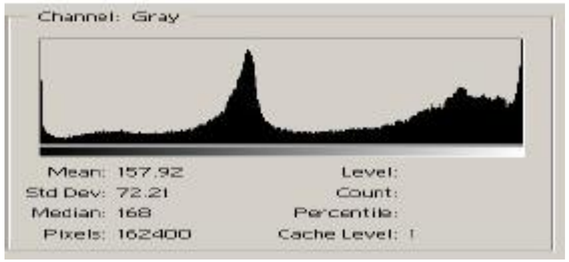
Channel: Gray

Mean: 103.98	Level:
Std Dev: 87.72	Count:
Median: 84	Percentile:
Pixels: 162400	Cache Level: 1

A limitation of the histogram

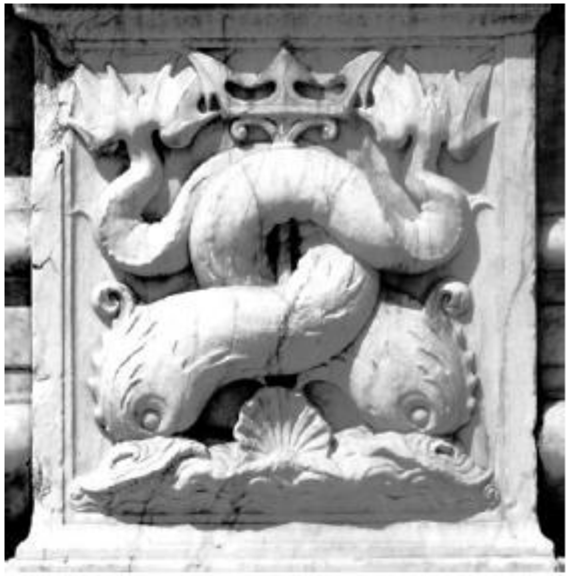
- Does not take into account the spatial distribution

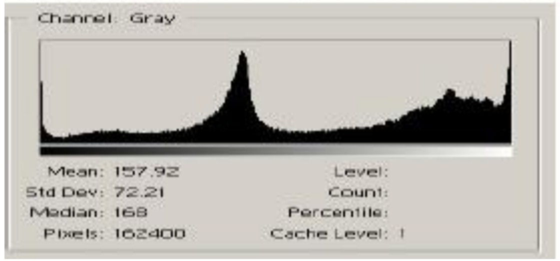




```
Channel: Gray

Mean: 157.92            Level:
Std Dev: 72.21          Count:
Median: 168             Percentile:
Pixels: 162400          Cache Level: 1
```





```
Channel: Gray

Mean: 157.92            Level:
Std Dev: 72.21          Count:
Median: 168             Percentile:
Pixels: 162400          Cache Level: 1
```

Istogram transformation (equalization)

Mapping of grey levels p in grey levels q with uniform distribution

$h(p)$ = number of pixels with value p ($0 \le p \le n$)

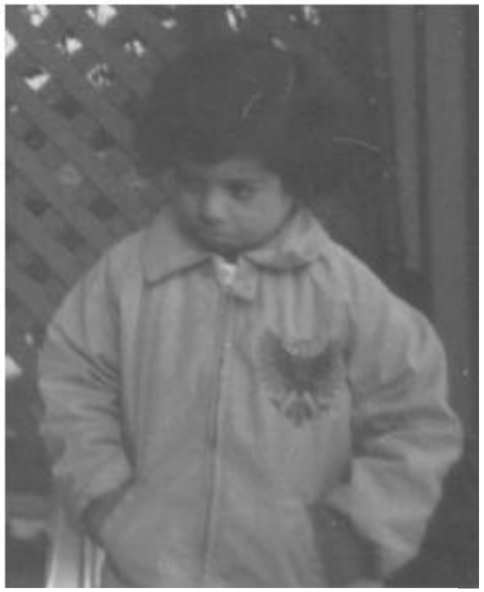
$g(q)\,dq = h(p)\,dp$

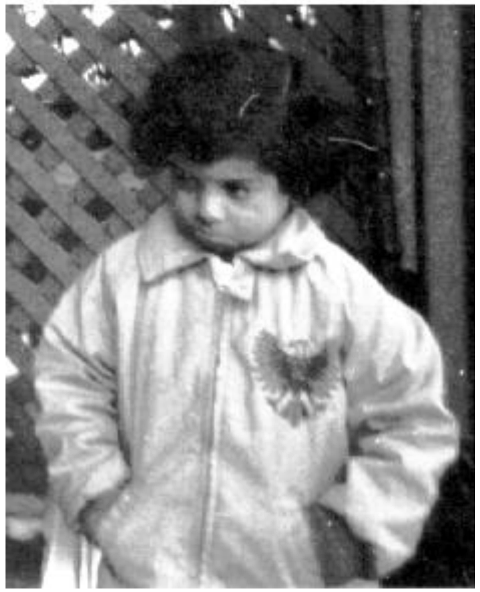
$g(q) = N^2/M$

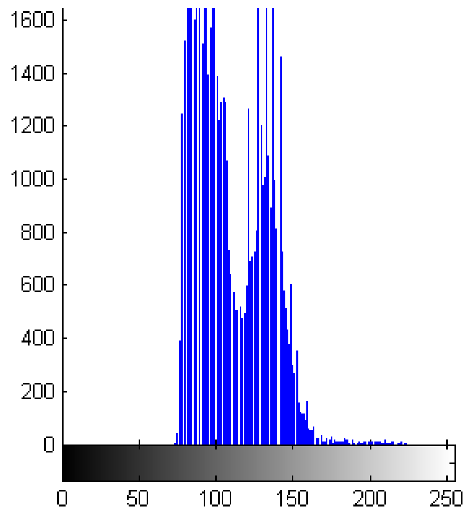
N^2: number of pixels, M: number of grey levels

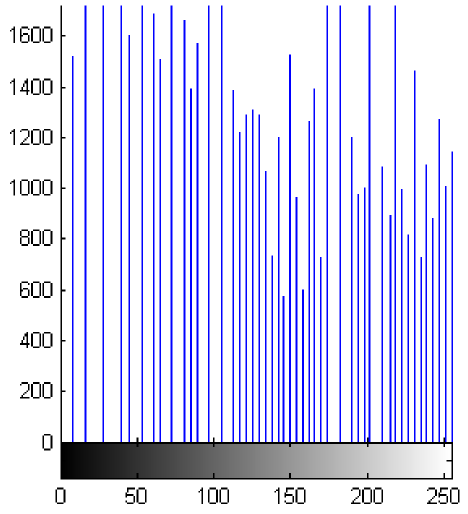
$$g(p) = M/N^2 \int_0^p h(s)\,ds$$

Example of histogram equalization

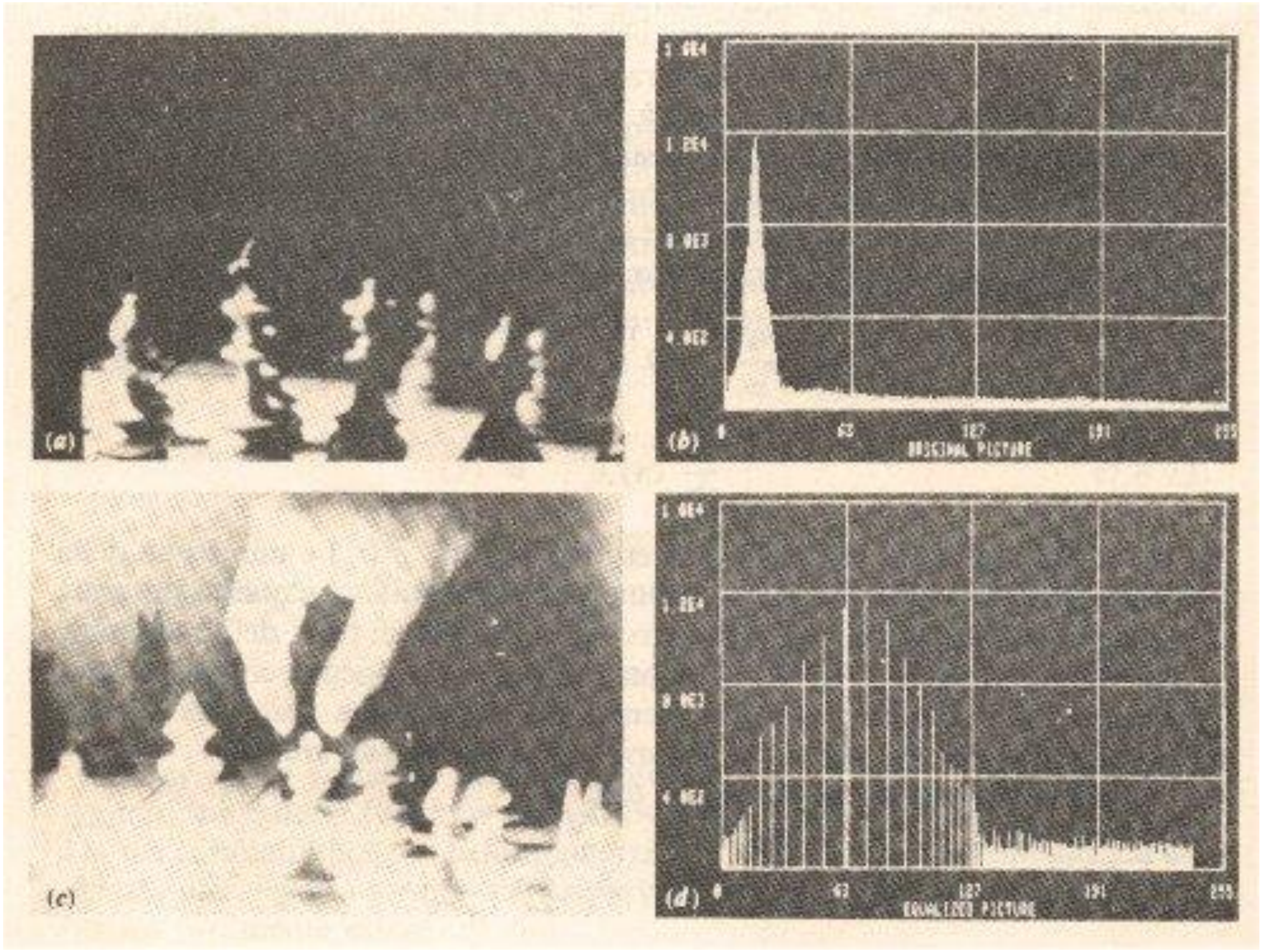








Example of histogram equalization



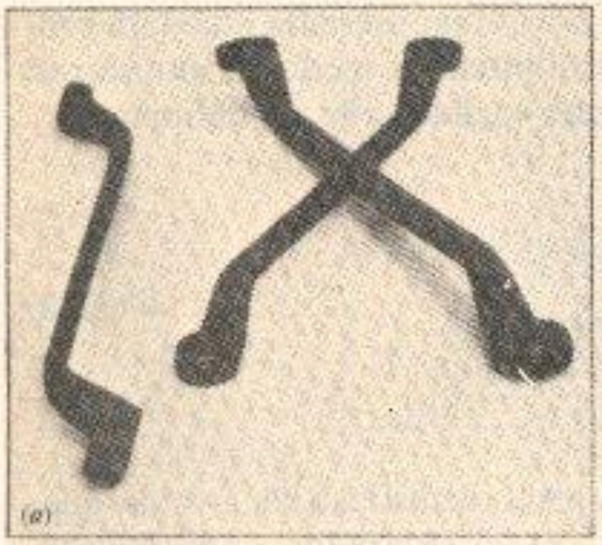
Thresholding

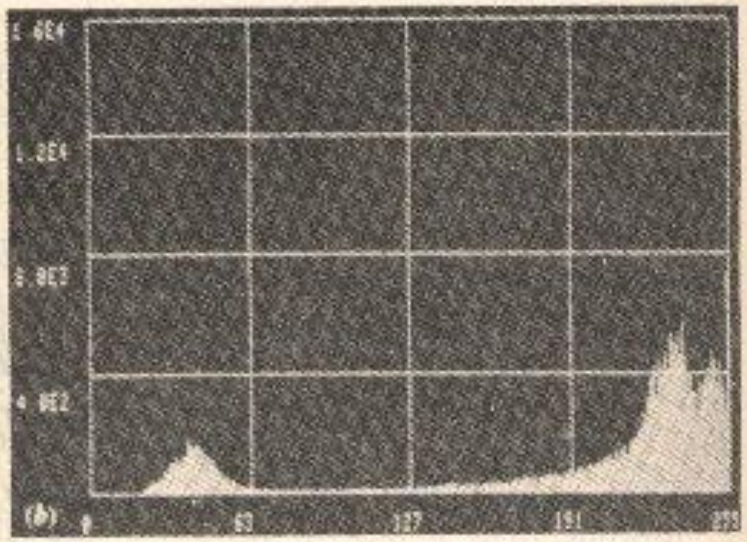
Transformation of the image in a binary image, through comparison with a threshold T:

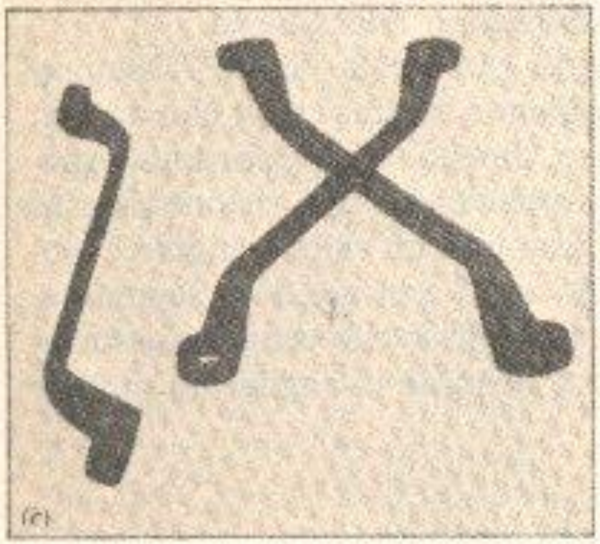
$$g(x,y) = \begin{cases} 1 \; if \; f(x,y) > T \\ 0 \; if \; f(x,y) \leq T \end{cases}$$

- T can be constant or variable with respect to x, y, f(x,y) or other local properties
- T can be found empirically or with statistical techniques

Threshold found from the image histogram

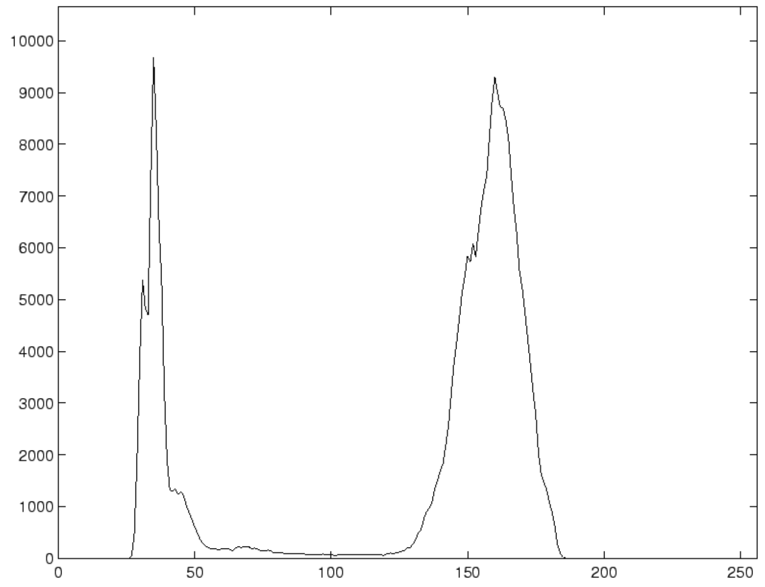






Background and object can be described as classes of the image histogram

Otsu thresholding method maximise the variance between classes.

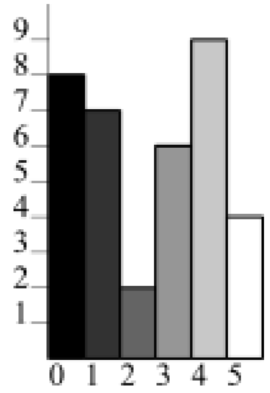


One implementation can be defined as follow:

1. Obtaining the image histogram
2. For each threshold value, t = 0, ..., L − 1 the following variables should be derived
3. Compute:

$$\begin{aligned}
{\mu_s}^t &= \frac{\Sigma_{i=0}^{t} \#(i) \cdot i}{\#s} \\
{\mu_o}^t &= \frac{\Sigma_{i=t+1}^{L-1} \#(i) \cdot i}{\#o} \\
W_s &= \frac{\#s}{\#s + \#o} \\
W_o &= \frac{\#o}{\#s + \#o} \\
\sigma_b^2 &= W_s W_o ({\mu_s}^t - {\mu_o}^t)^2
\end{aligned}$$

4. Maximum σ^2_b (t) defines the correct threshold t.

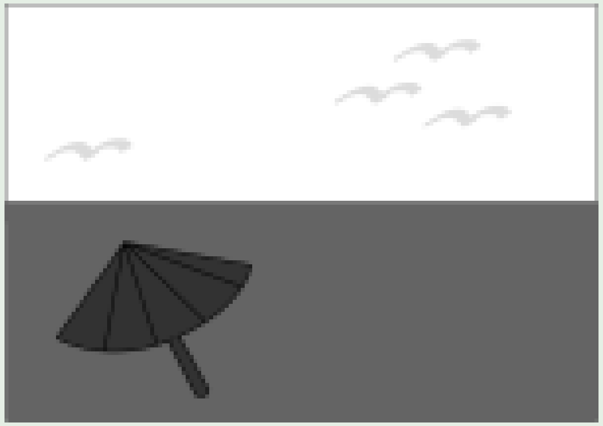


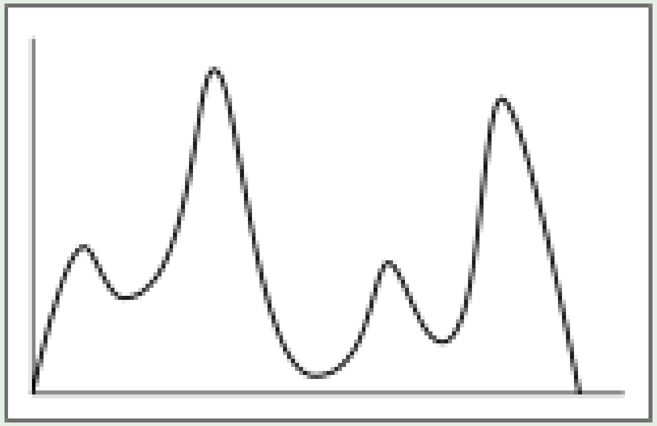
Per $t = 0$:

$$
\begin{aligned}
W_s &= \frac{8}{36} = 0.22 \\
\mu_s &= \frac{8 \cdot 0}{8} = 0 \\
W_o &= \frac{7+2+6+9+4}{36} = \frac{28}{36} = 0.78 \\
\mu_o &= \frac{7 \cdot 1 + 2 \cdot 2 + 6 \cdot 3 + 9 \cdot 4 + 4 \cdot 5}{28} = 3.04 \\
\sigma_b^2 &= 0.22 \cdot 0.78(3.04 - 0)^2 = 1.59
\end{aligned}
$$

Threshold	t=0	t=1	t=2	t=3	t=4
Variance	1.59	2.56	2.63	2.14	0.87

Example





How to retrieve (automatically) all the objects?

- It is possible to iterate Otsu on the histogram subset
- It is needed local threshold strategies

Background subtraction

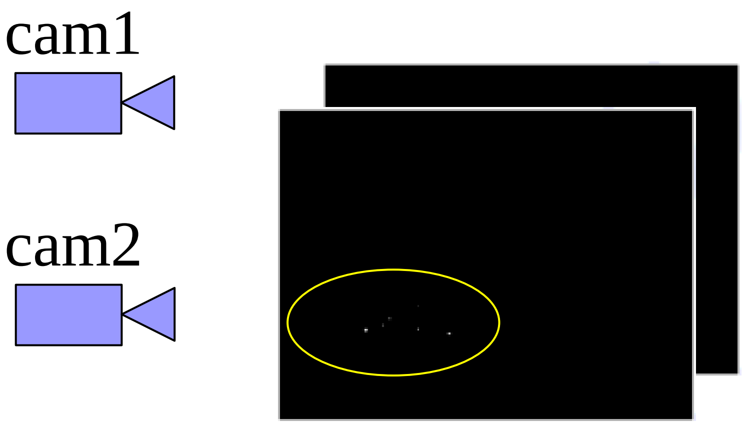
The technique of background subtraction removes the slight variations of the background grey levels, by approximating them by a function and by subtracting such function from the image function

$f_n(x,y) = f(x,y) - f_b(x,y)$

$f_b(x,y) = c$ (constant) or

$f_b(x,y)=m_1x+m_2y+c$ (linear)

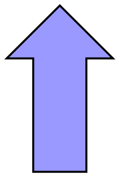
The overall example



$(x_1,y_1)_1$ $(x_1,y_1)_2$
$(x_2,y_2)_1$ $(x_2,y_2)_2$
… …
$(x_n,y_n)_1$ $(x_n,y_n)_2$

(x_1,y_1,z_1)
(x_2,y_2,z_2)
…
(x_n,y_n,z_n)

Shape recognition	Marker positions	3D reconstruction



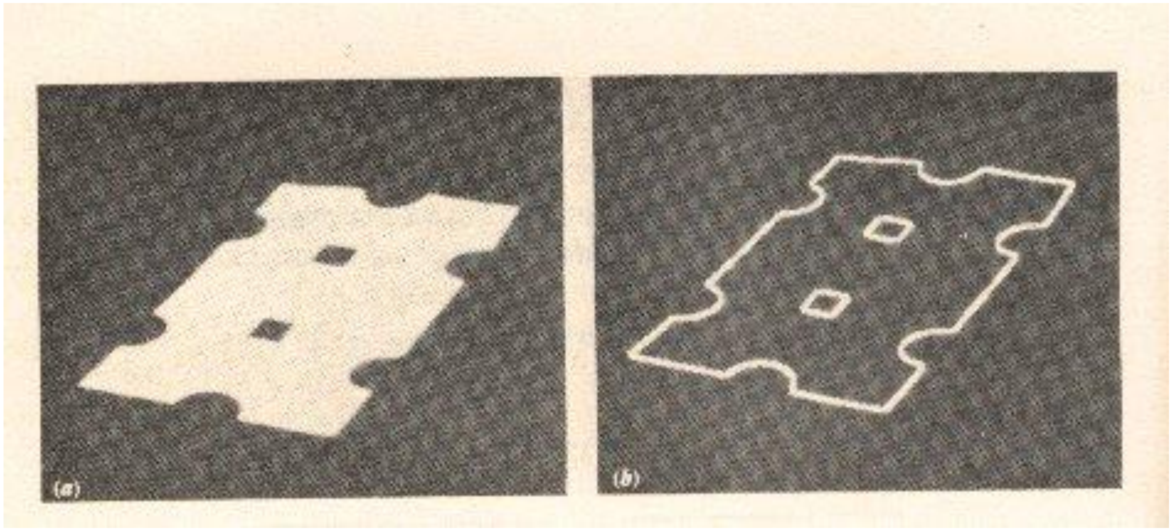
Once the objects were separated from the background, the shape can be identified:

- Hough transformation
- Gradient
- Erosion
- …

Early processing:
EDGE DETECTION techniques

EDGE: area where grey levels vary significantly

EDGE OPERATOR: mathematical operator that can detect an edge

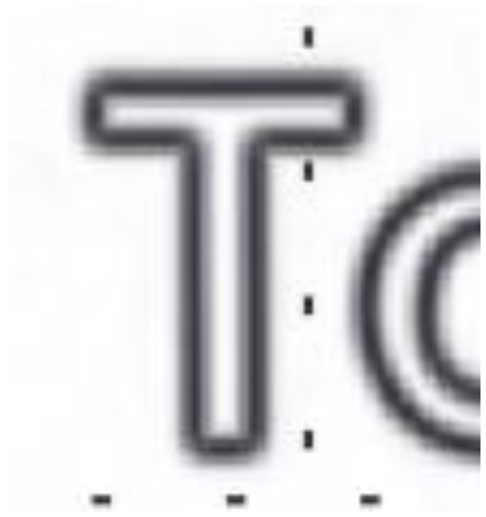


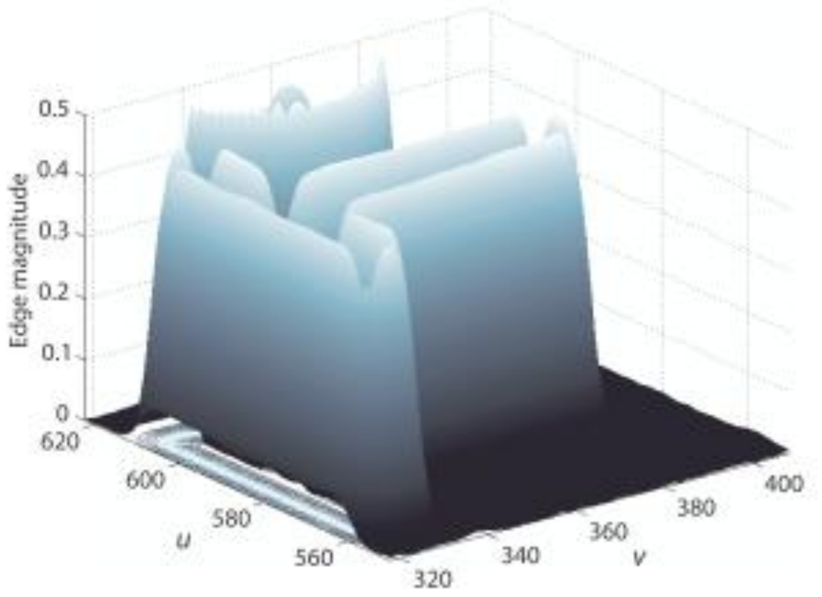
Gradient-based operators

GRADIENT: measure of discontinuity in an image point

GRADIENT DIRECTION ($\phi(x,y)$): direction of maximum variation of grey levels

GRADIENT INTENSITY ($s(x,y)$): intensity of the variation of grey levels





Gradient-based edge detection

$$G[f(x,y)] = \begin{vmatrix} \frac{\delta f}{\delta x} \\ \frac{\delta f}{\delta y} \end{vmatrix} = \begin{vmatrix} \Delta_1 \\ \Delta_2 \end{vmatrix}$$

$s(x,y) = (\Delta_1{}^2 + \Delta_2{}^2)^{1/2}$ GRADIENT INTENSITY

$\phi(x,y) = \tan^{-1}(\Delta_2 / \Delta_1)$ GRADIENT DIRECTION

Approximation of first derivative (difference)

$$\frac{\partial f}{\partial x} \cong f(x+1, y) - f(x, y)$$

$$\frac{\partial f}{\partial y} \cong f(x, y+1) - f(x, y)$$

0	0	0
0	-1	0
0	+1	0

0	0	0
0	-1	+1
0	0	0

Gradient-based edge detection and thresholding

- g(x,y)=G[f(x,y)]

- $$g(x,y)=\begin{cases} 1 \quad \text{if} \quad g[f(x,y)] > T \\ 0 \quad \text{if} \quad g[f(x,y)] <= T \end{cases}$$

T: threshold

Gradient-based edge detection and thresholding

DIFFERENCE OPERATOR:

Δ_1= f(x+a,y) - f(x,y)

Δ_2= f(x,y+a) - f(x,y)

$$\boxed{\begin{array}{lll} I_{i\text{-}a,j} & & \\ \vdots & & \\ I_{i,j} & \cdots & I_{i,j+a} \end{array}}$$

$$S_{i,j} = \sqrt{\Delta_1^2 + \Delta_2^2}$$

$$\Delta_1 = I_{i\text{-}a,j} - I_{i,j}$$

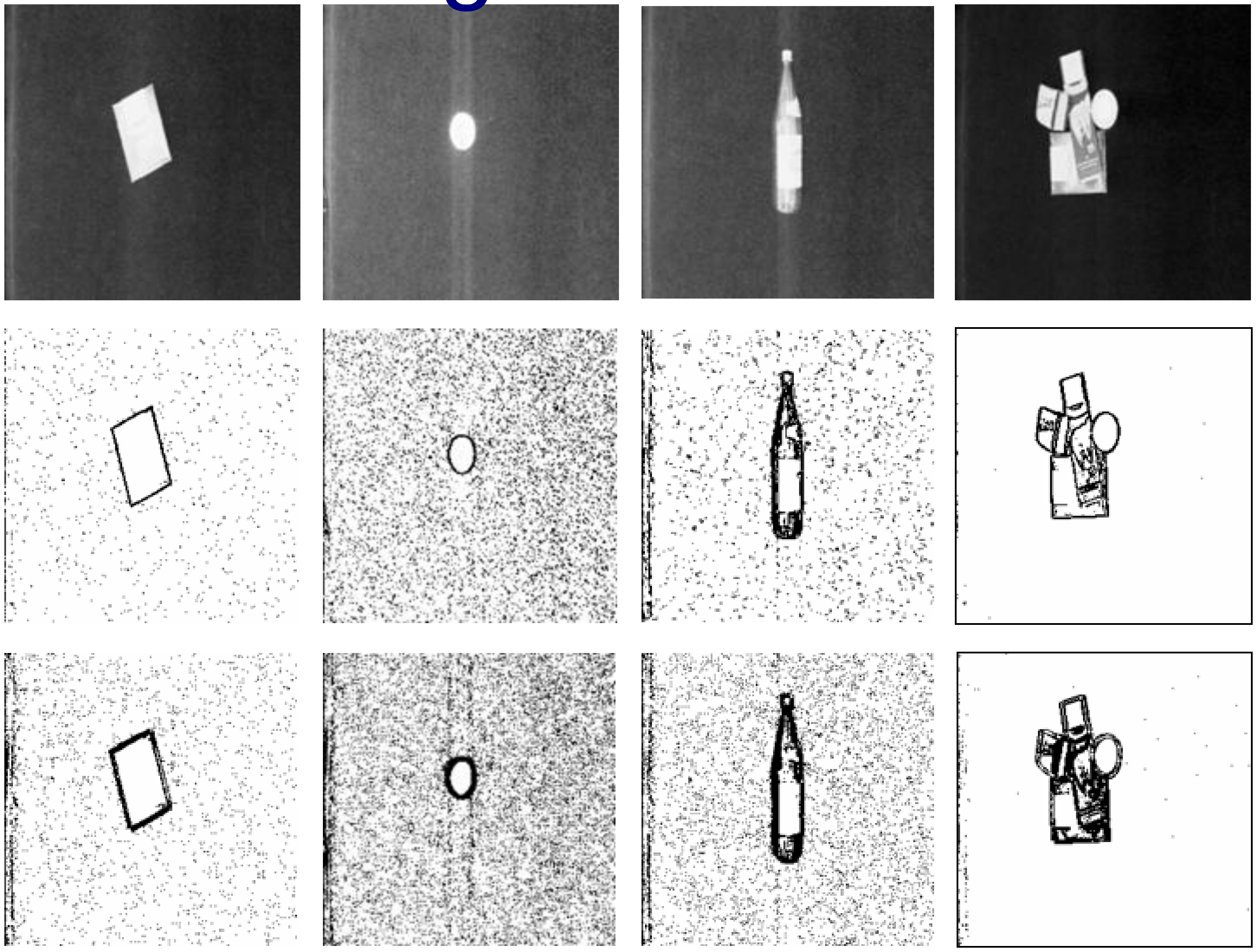
$$\Delta_2 = I_{i,j+a} - I_{i,j}$$

$$S_{i,j} = \Delta_1^2 + \Delta_2^2$$

$$E_{i,j} = \begin{cases} 0 & if\ S_{i,j} \le threshold \\ 1 & if\ S_{i,j} > threshold \end{cases}$$

$$E_{i,j} = \begin{cases} 0 & if\ S_{i,j} \le threshold^2 \\ 1 & if\ S_{i,j} > threshold^2 \end{cases}$$

Gradient-based edge detection and thresholding



A = 1

A = 2

Gradient-based edge detection

ROBERTS' CROSS OPERATOR:

Δ_1= f(x,y+a) - f(x+a,y) Δ_2= f(x,y) - f(x+a,y+a)

Δ_1

0	1
-1	0

Δ_2

1	
	-1

PREWITT'S OPERATOR:

Δ_1= f(x-1,y+1) + f(x,y+1) + f(x+1,y+1)
  - f(x-1,y-1) - f(x,y-1) - f(x+1,y-1)

Δ_2= f(x-1,y-1) + f(x-1,y) + f(x-1, y+1)
  - f(x+1,y-1) - f(x+1)y) - f(x+1,y+1)

Δ_1

-1	-1	-1
0	0	0
1	1	1

Δ_2

1	0	-1
1	0	-1
1	0	-1

Gradient-based edge detection

SOBEL'S OPERATOR:

Δ_1= f(x-1,y+1) + 2f(x,y+1) + f(x+1,y+1)
- f(x-1,y-1) - 2f(x,y-1) - f(x+1,y-1)

Δ_2= f(x-1,y-1) + 2f(x-1,y) + f(x-1, y+1)
- f(x+1,y-1) - 2f(x+1)y) - f(x+1,y+1)

Δ_1

-1	-2	-1
0	0	0
1	2	1

Δ_2

-1	0	1
-2	0	2
-1	0	1

Comparison among masks

Gradient

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

The magnitude is:

$$G = \sqrt{G_x^2 + G_y^2}$$

Sobel implementation (or others: Prewit, Robert, . . .)

$G_x=$

-1	0	1
-2	0	2
-1	0	1

$G_y=$

-1	-2	-1
0	0	0
1	2	1

Laplacian (by convolution of kernel)

Analytic

$$\nabla^2 = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Discreet (one possible)

0	-1	0
-1	4	-1
0	-1	0

Example

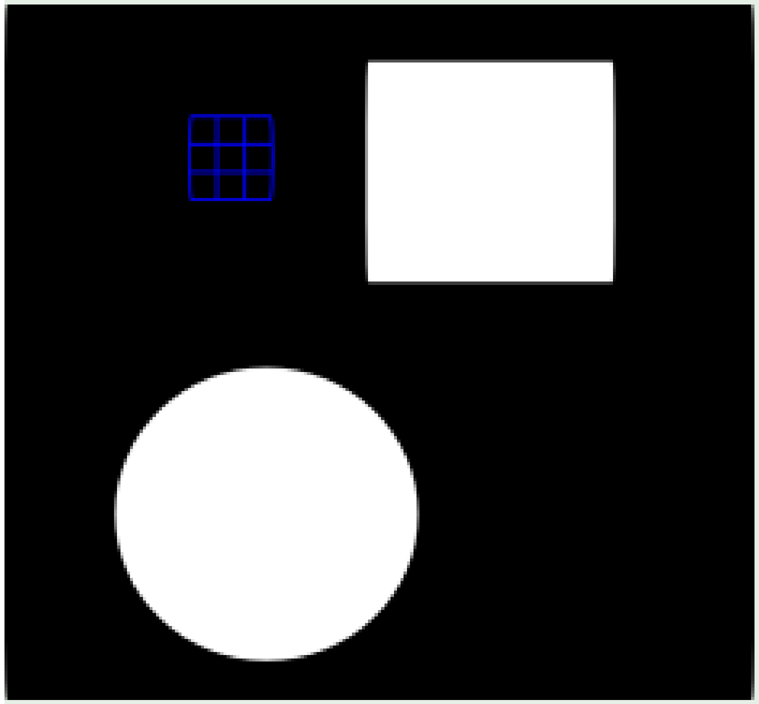


Image window:

$$f(x,y) = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

$$\nabla^2 = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

$$G_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

Products are:

$$\nabla^2 \otimes f(x,y) = 0$$

$$G_x \otimes f(x,y) = 0$$

Example

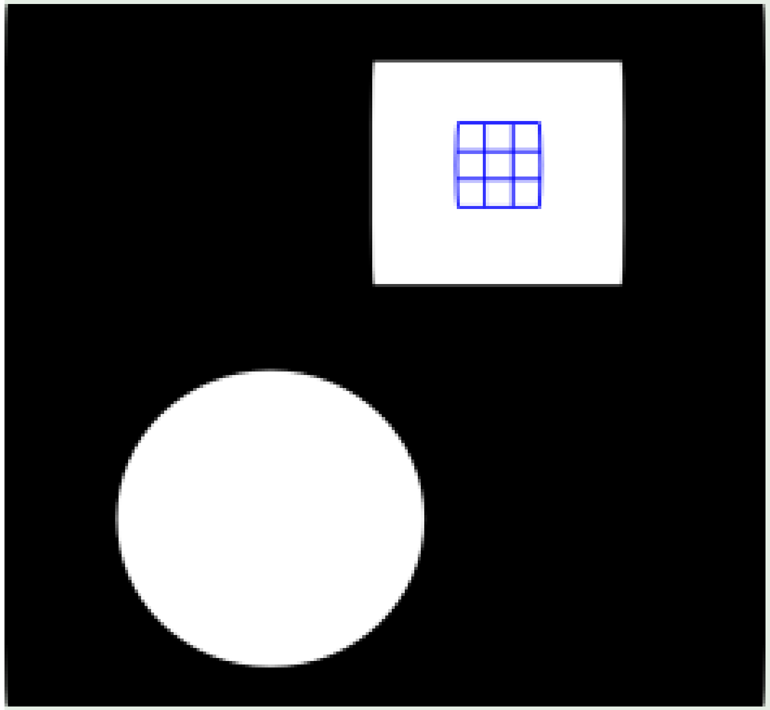


Image window:

$$f(x,y) = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

$$\nabla^2 = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

$$G_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

Products are:

$$\nabla^2 \otimes f(x,y) = 0$$
$$G_x \otimes f(x,y) = 0$$

Example

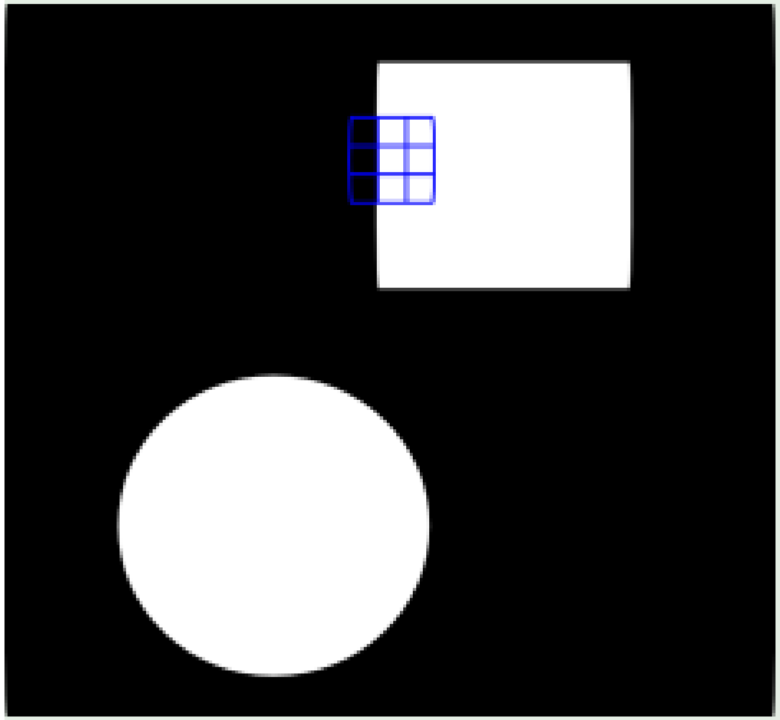


Image window:

$$f(x,y) = \begin{array}{|c|c|c|}\hline 0 & 1 & 1 \\ \hline 0 & 1 & 1 \\ \hline 0 & 1 & 1 \\ \hline\end{array}$$

$$\nabla^2 = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline\end{array}$$

$$G_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline\end{array}$$

Products are:

$$\nabla^2 \otimes f(x,y) = 1$$
$$G_x \otimes f(x,y) = 4$$

Example

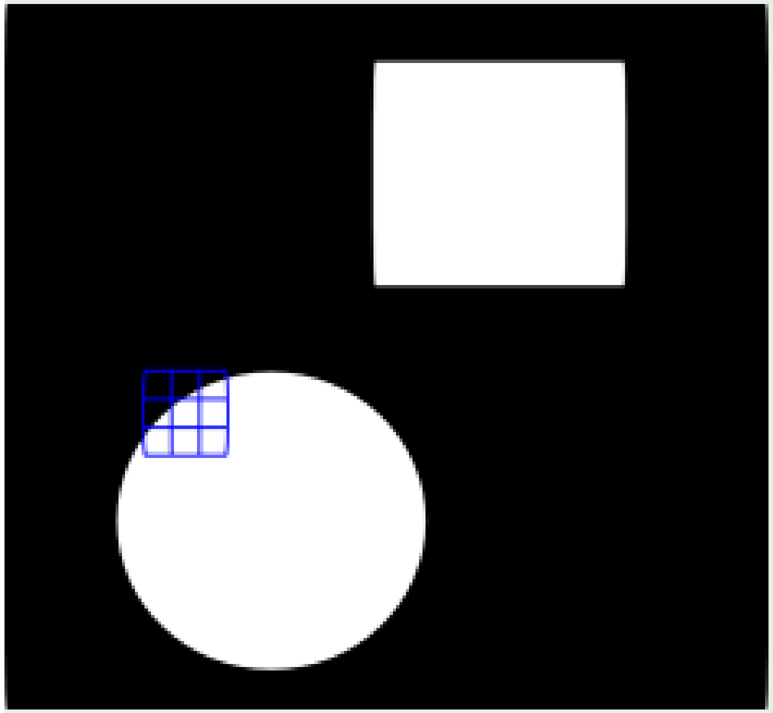


Image window:

$$f(x,y) = \begin{array}{|c|c|c|}\hline 0 & 0 & 1 \\ \hline 0 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

$$\nabla^2 = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline\end{array}$$

$$G_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline\end{array}$$

Products are:

$$\nabla^2 \otimes f(x,y) = 2$$

$$G_x \otimes f(x,y) = 3$$

Example

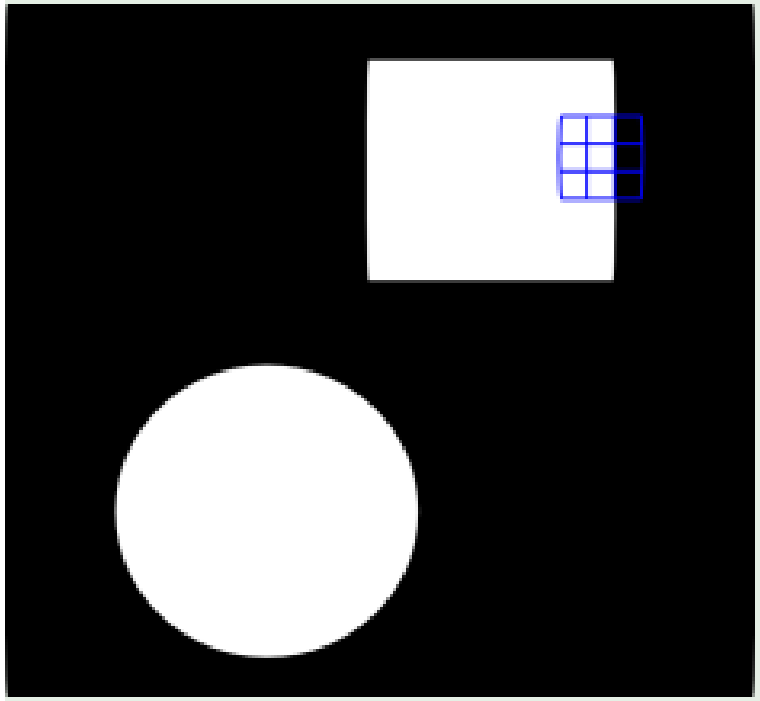


Image window:

$$f(x,y) = \begin{array}{|c|c|c|}\hline 1 & 1 & 0 \\ \hline 1 & 1 & 0 \\ \hline 1 & 1 & 0 \\ \hline\end{array}$$

$$\nabla^2 = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline\end{array}$$

$$G_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline\end{array}$$

Products are:

$$\nabla^2 \otimes f(x,y) = 1$$
$$G_x \otimes f(x,y) = -4$$

Hough’s method

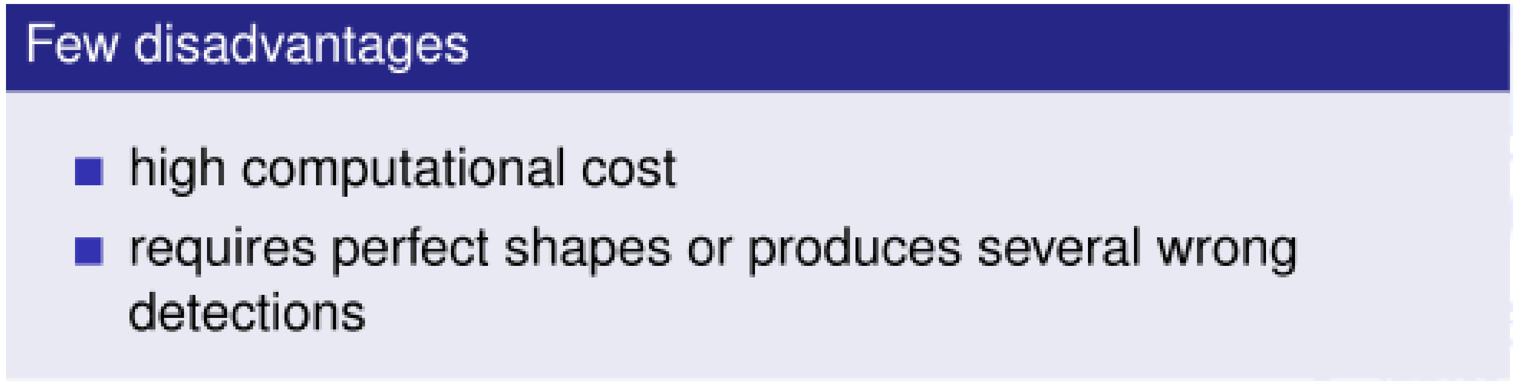
It can detect any boundary that can be described by a parametric curve.

It compares the edge image with all the curves obtained by varying the parameter values

Hough in a nutshell

It transforms the original image in an accumulation matrix in the parameters plane.

Every points that belong to the researched function (line, circle,. . .) increase the accumulation value.

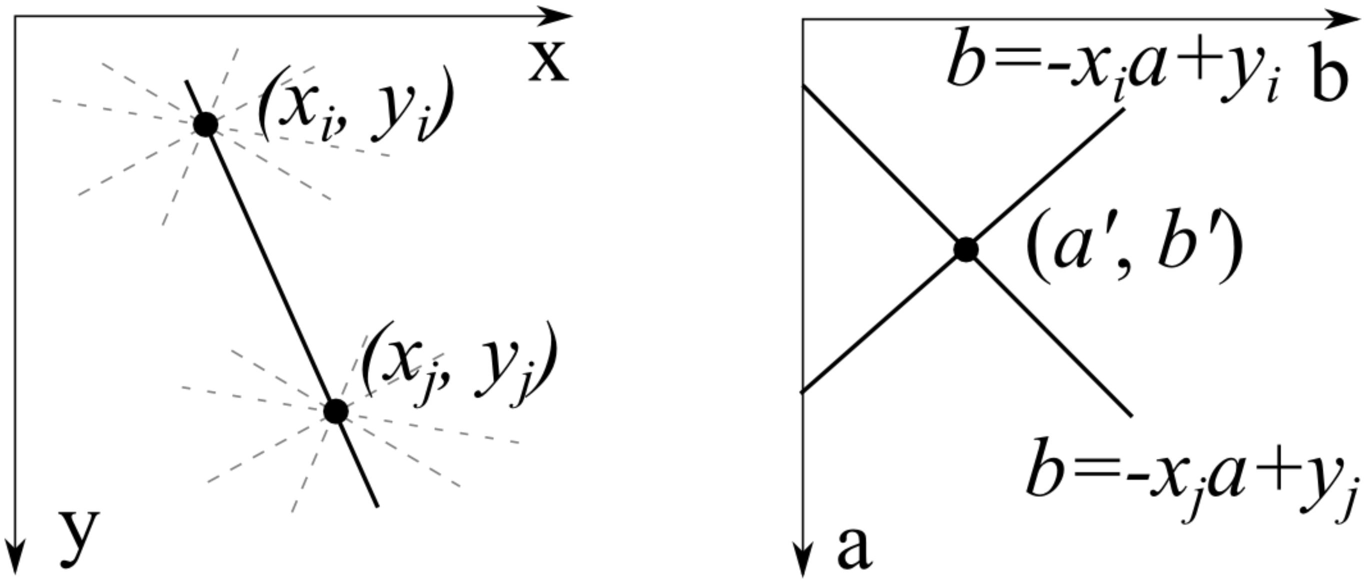


Few disadvantages

- high computational cost
- requires perfect shapes or produces several wrong detections

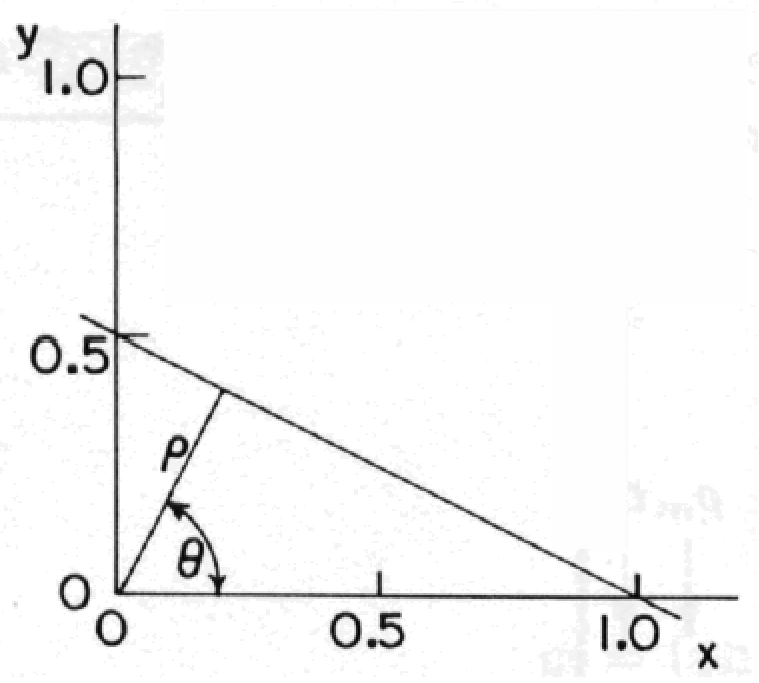
$$y = a \cdot x + b \rightarrow b = -x_i \cdot a + y_i$$

The linear function $b = -x_i \cdot a + y_i$ in the parameters plane represents all the linear functions that belong to the generic point (x_i , y_i).



Each point of the same function increase the accumulation (a', b').

Polar parametrization



- The line shown can be described by the function

$$y=ax+b$$

and identified by the couple of parameters:

$$(a,b)=(-0.5,0.5)$$

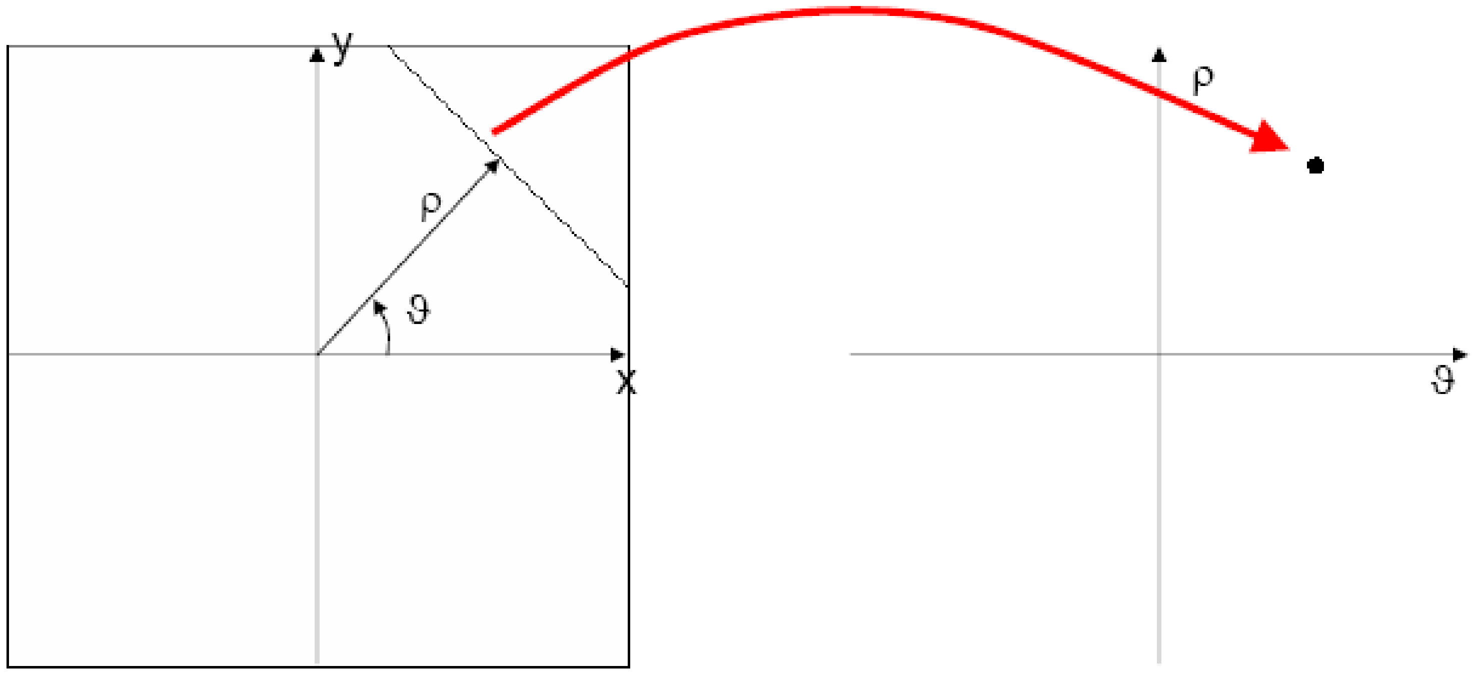
- or by the function

$$\rho = x \cos\vartheta + y \sin \vartheta$$

- and identified by the couple:

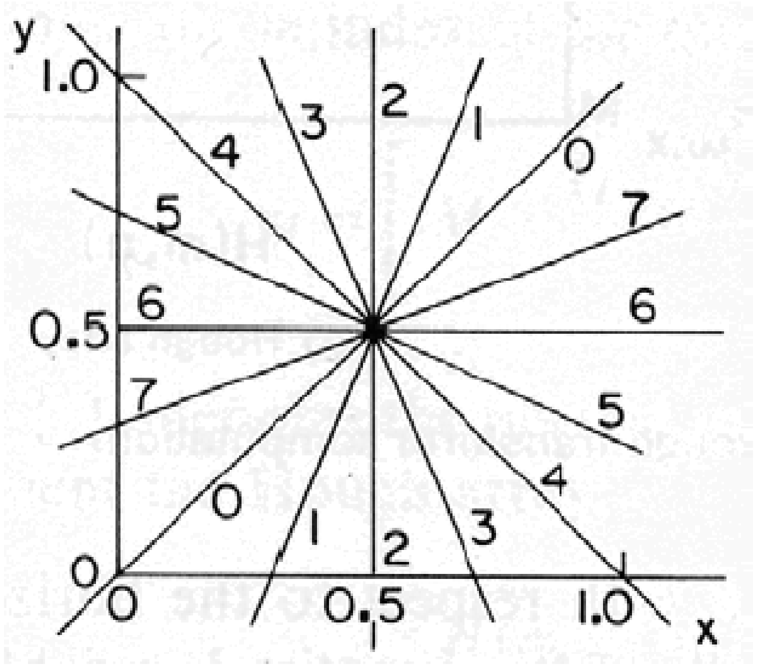
$$(\rho,\vartheta)=(0.447,1.107)$$

Transformation of the plane

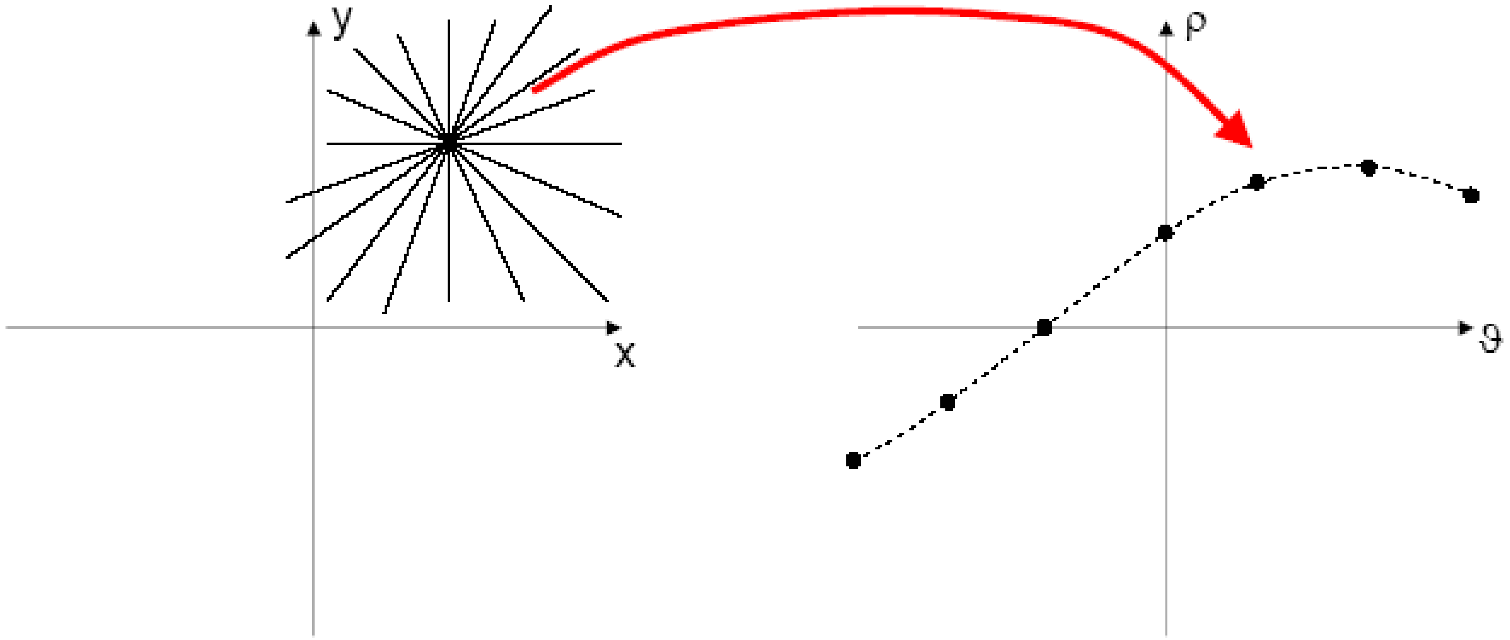


Transformation of a point

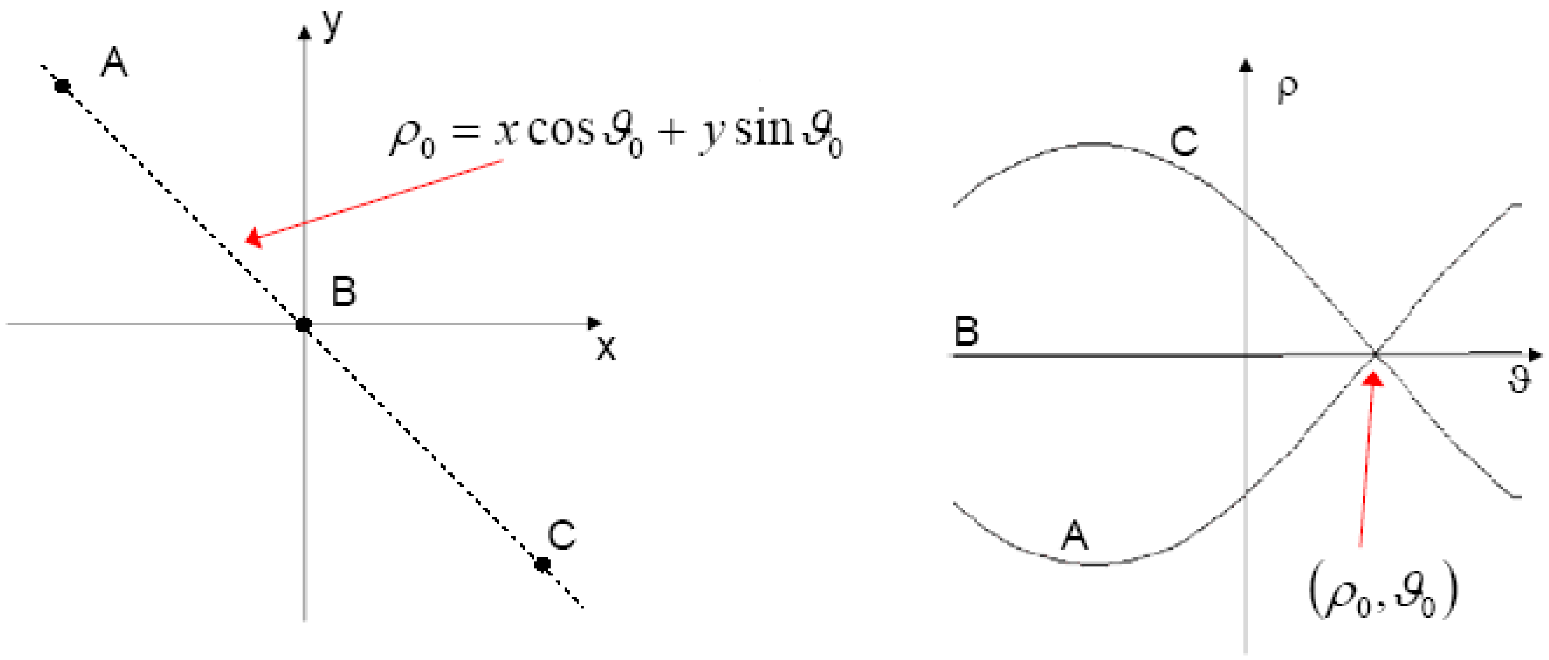
- In the image plane, one point is identified by the intersection of lines.
- Each point P corresponds, in the parameter plane, to the curve given by the image points of the lines passing through P



Transformation of a point

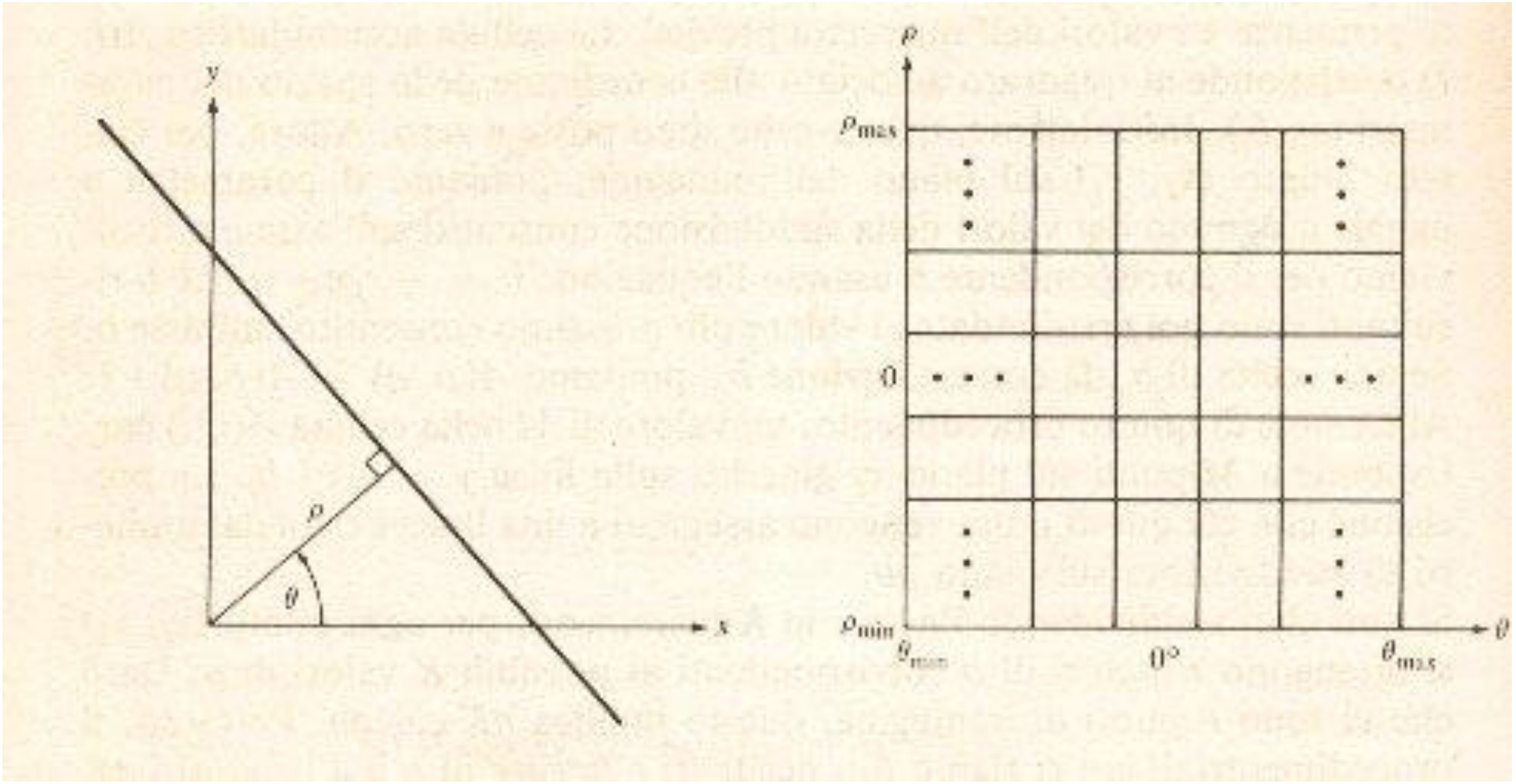


Detection of a lines on the transformed plane



Hough's method with the polar representation of the line

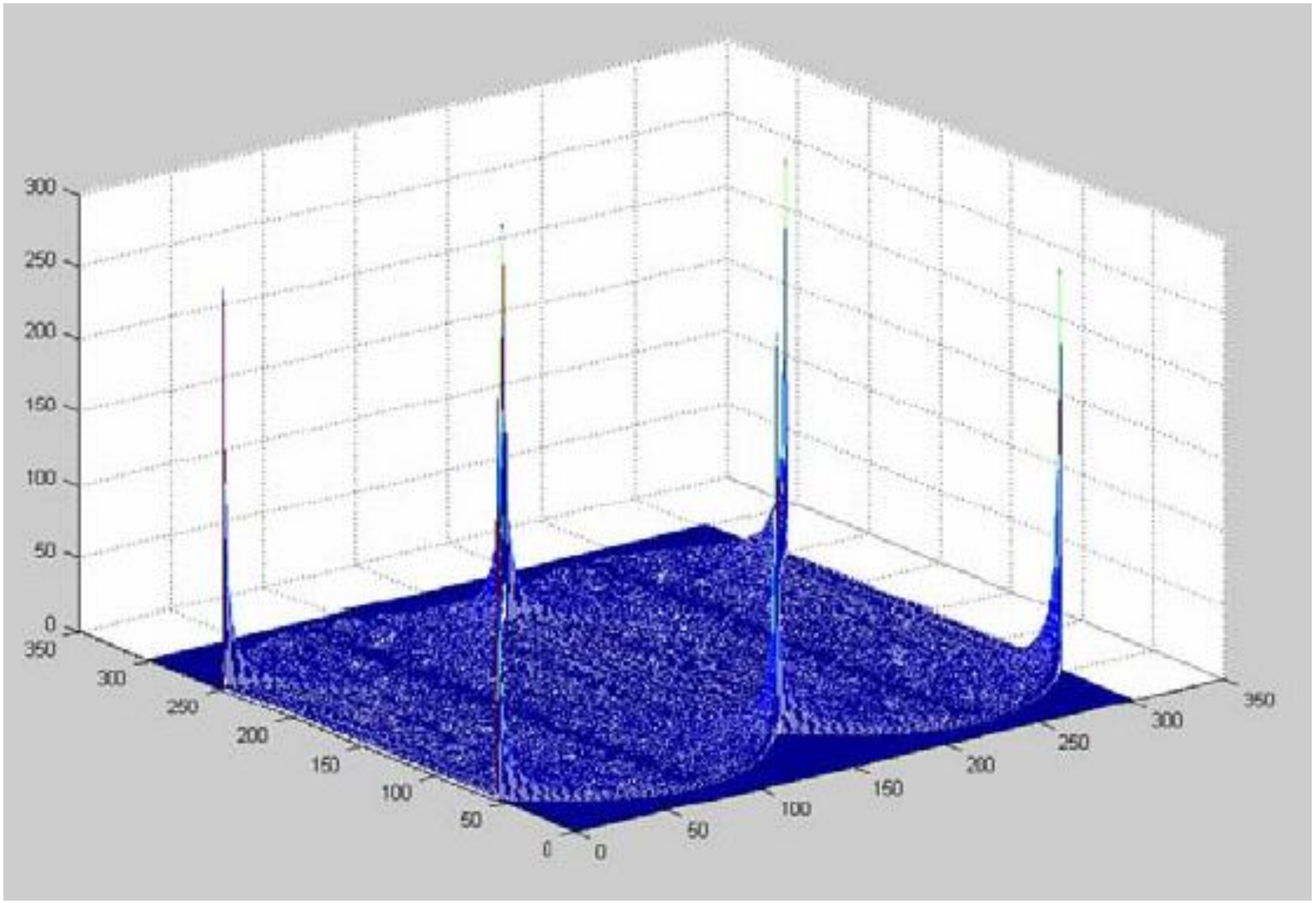
$\rho = x \cos \theta + y \sin \theta$



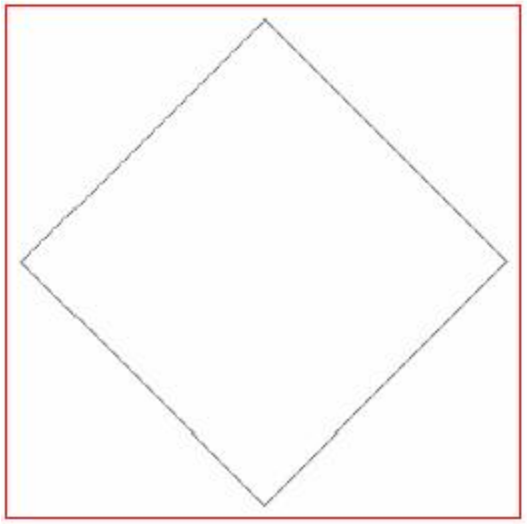
Hough’s algorithm

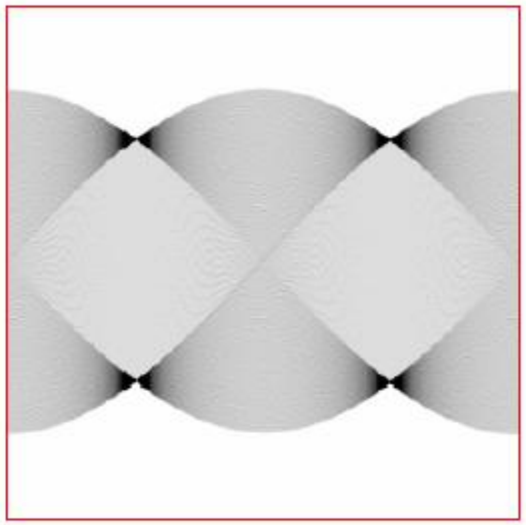
1. Quantize the parameter space between appropriate minimum and maximum values
2. Create an accumulation array with size equal to the number of parameters, initialized to 0
3. For each edge in the image, increment of the element of the accumulation array corresponding to the parameter values of the curves on which the edge lays
4. The local maxima in the accumulation array represent the parameter values of the curves that better approximate the boundary

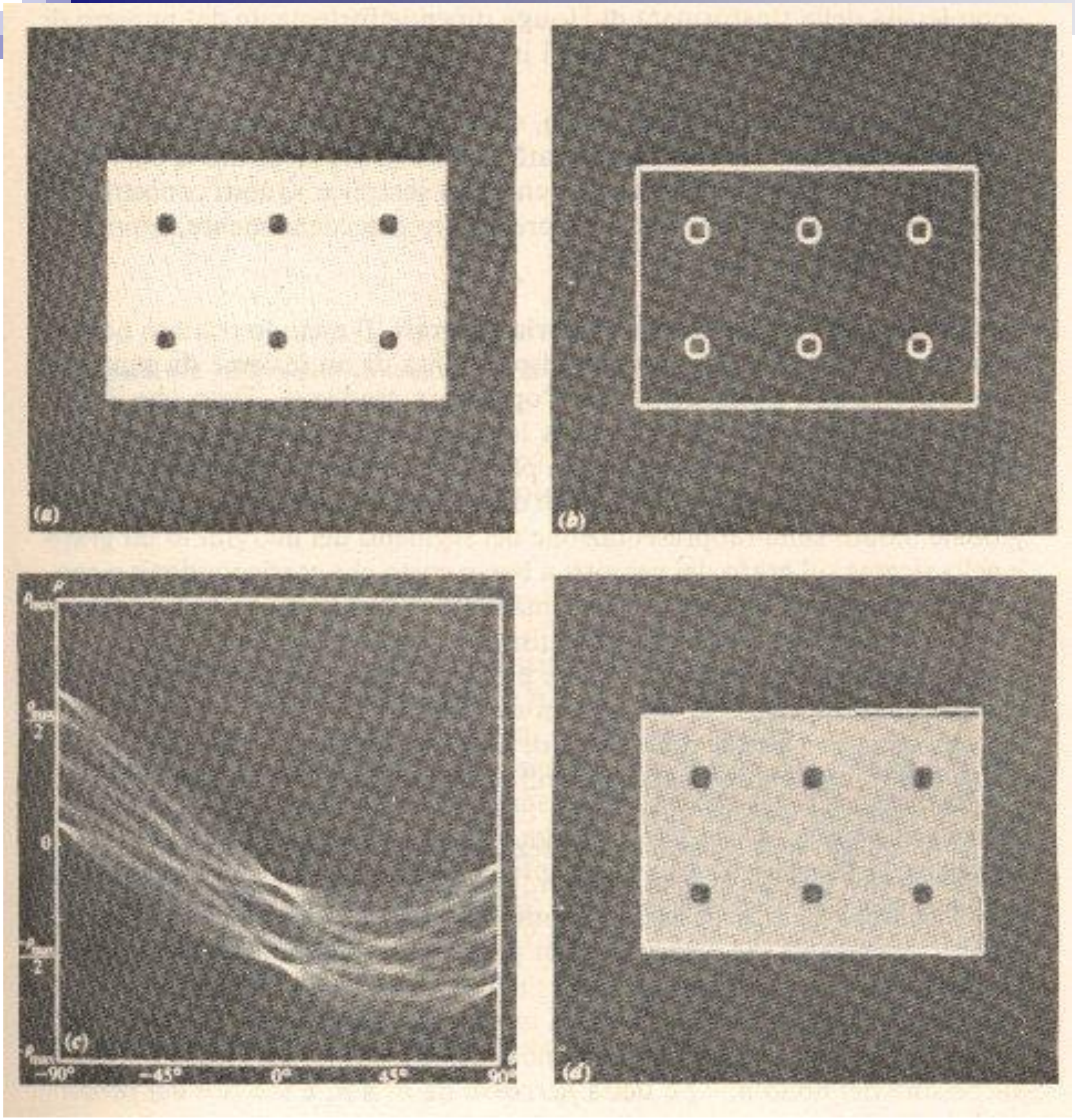
Example of accumulation matrix



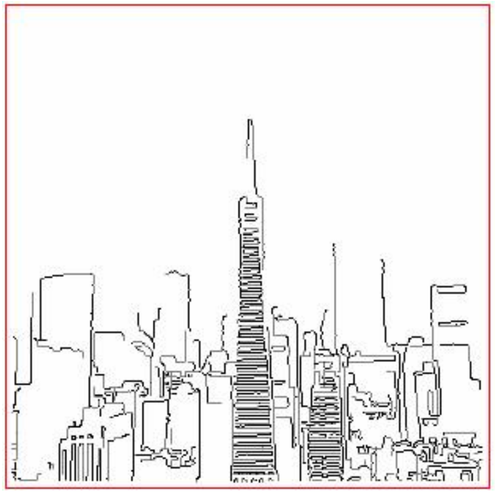
Example of the Hough's method to a rectangle

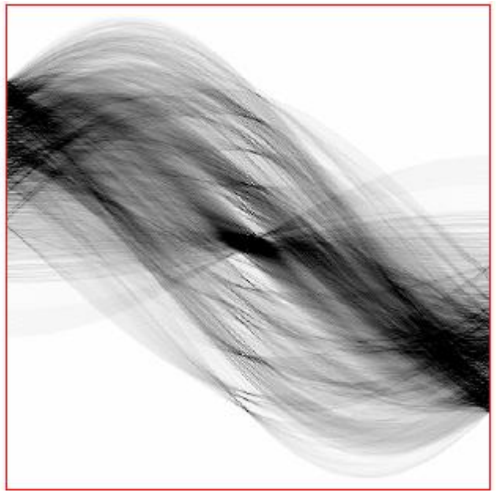


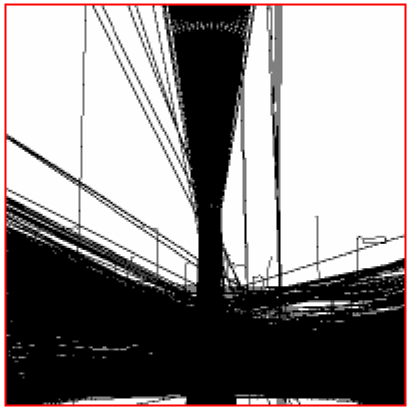



(a)
(b)
ρ
ρmax
ρmax/2
0
−ρmax/2
−ρmax
(c)
θ
−90°
−45°
0°
45°
90°
(d)

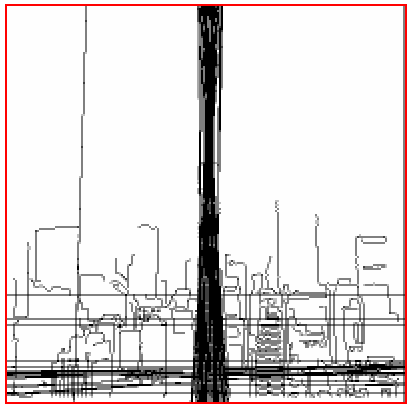
The problem of selecting the ‘right’ curves



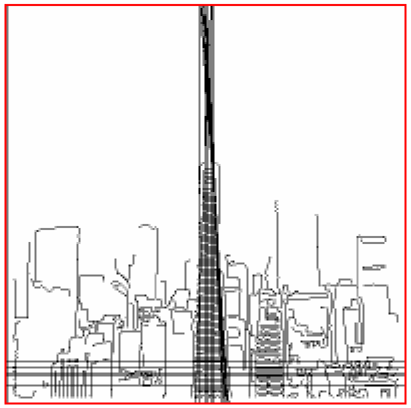




Soglia: 101

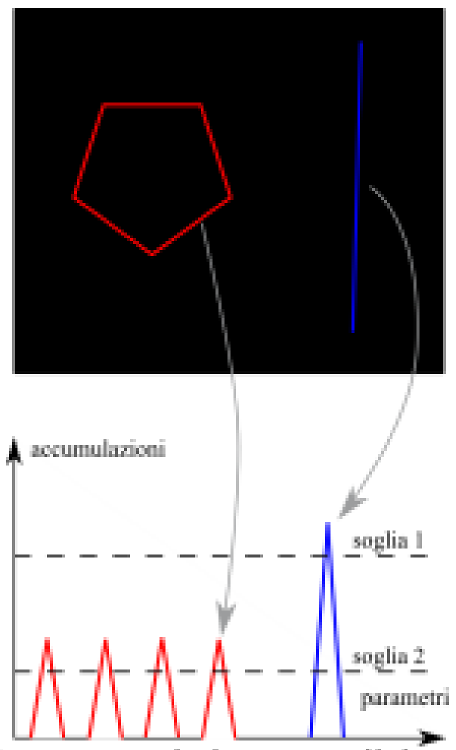


Soglia: 140



Soglia: 160

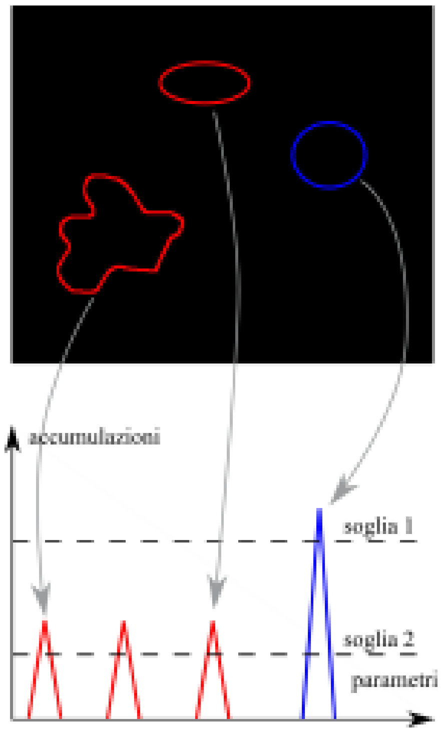
Square and lines



- depending on the length of the line, the accumulation has different values (lower or higher)
- a threshold should be selected to separate the key features from the other elements

In this case, it is possible to lower the threshold and each lines will be detected.

Ellipsis and circles



- same procedure can be used to detect circles
- in this case is difficult to identify the right threshold

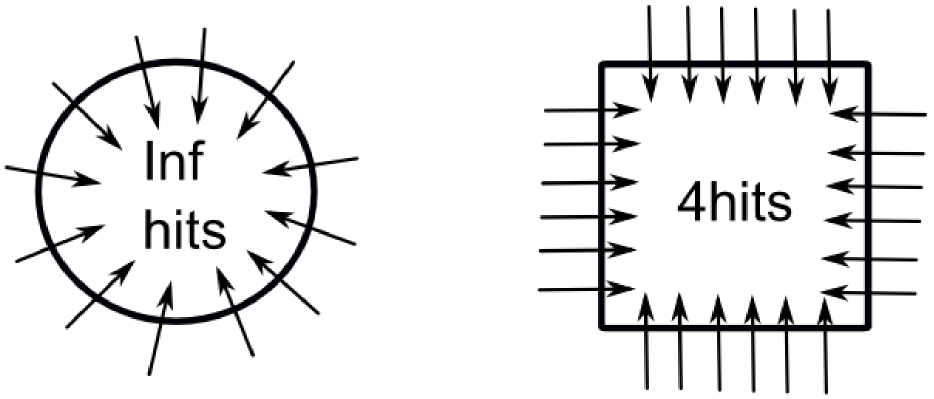
Gradient-based accumulation

Gradient can be used to detect edges.
It provides also other information:

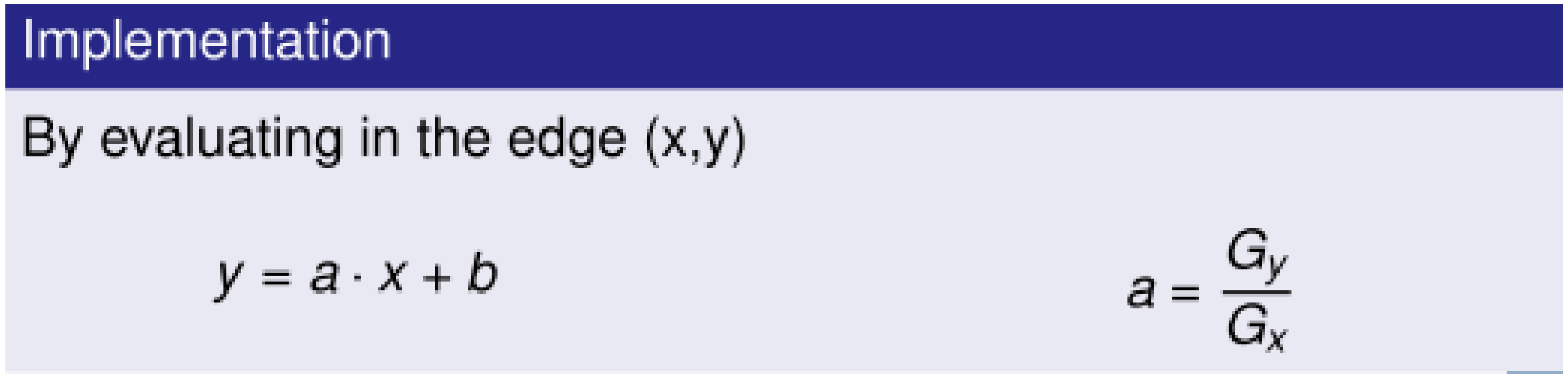
- ❑ magnitude
- ❑ direction

It is possible to state that the gradient can characterise a shape and the speed of the grey level variation

By applying again the accumulation:



We accumulate points in the gradient direction, by drawing lines from the edges.

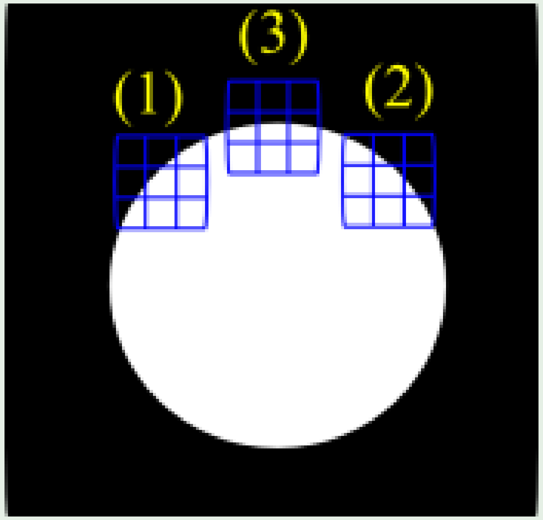


Implementation

By evaluating in the edge (x,y)

$$y = a \cdot x + b \qquad a = \frac{G_y}{G_x}$$

Example

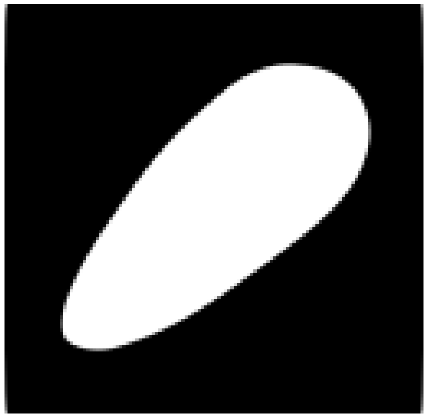


$$
\begin{aligned}
y &= a \cdot (x - x_0) + b \\
a &= \frac{G_y}{G_x} \\
b &= (x_0, y_0)
\end{aligned}
$$

1. $G_x = 3; G_y = 3; a = 1;$
2. $G_x = -3; G_y = 3; a = -1;$
3. $G_x = 0; G_y = 4; a = \infty;$

It is possible to detect

- Circles
- Ellipses
- Droplet shapes

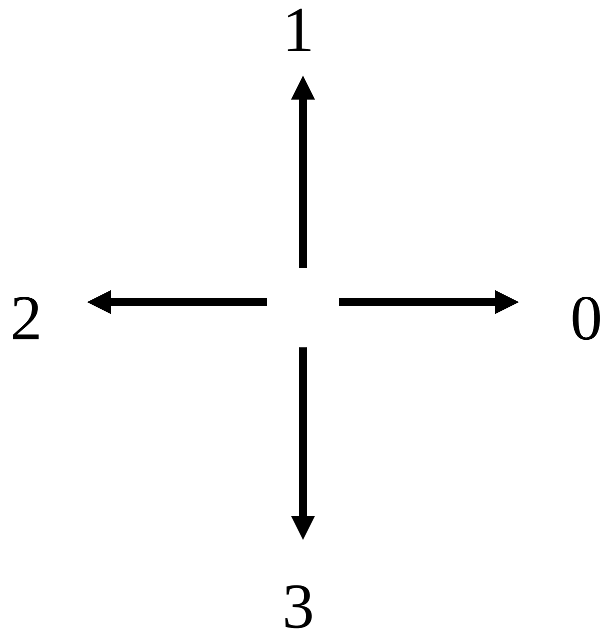


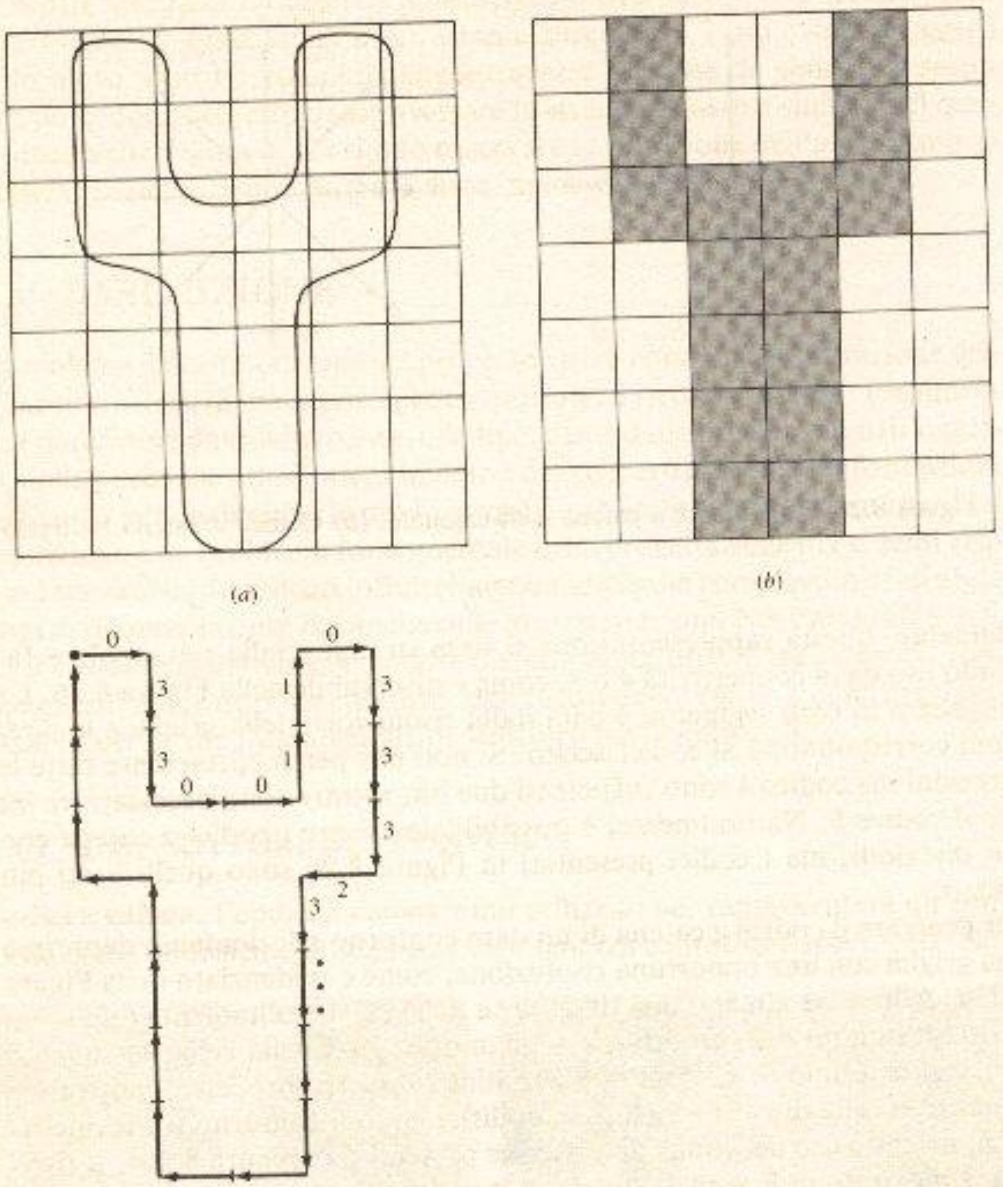
They are common deformation when the subject is fast and the image capturing is slow

Boundary representation

- Chain codes
- Characteristic shapes

Chain code and shape numbers





Chain code and shape numbers

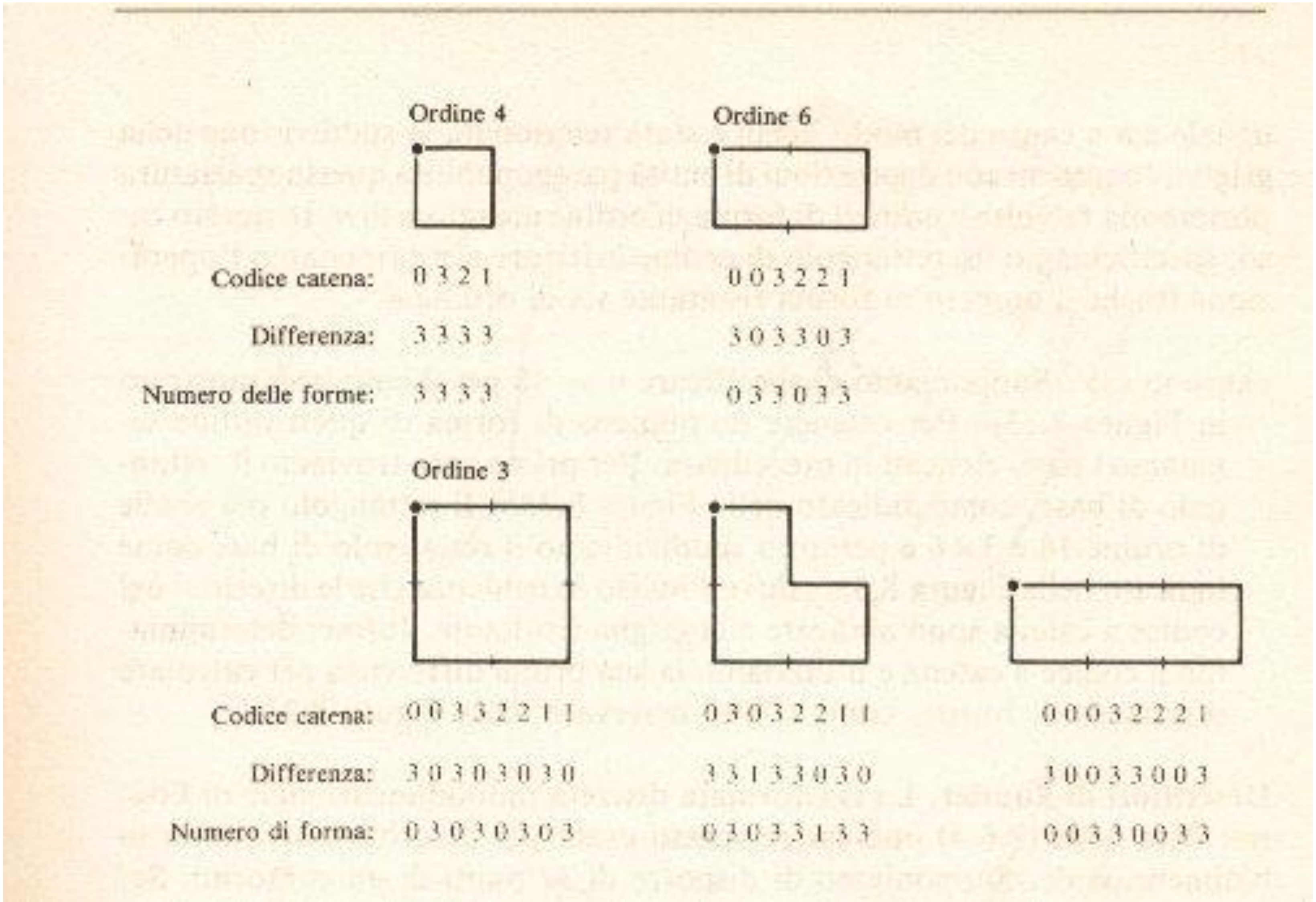
Normalization with respect to rotation:
use the difference of the chain code by counting counterclockwise the number of directions which separate two neighboring elements

Es: 10103322

3133030

33133030

Chain code and shape numbers

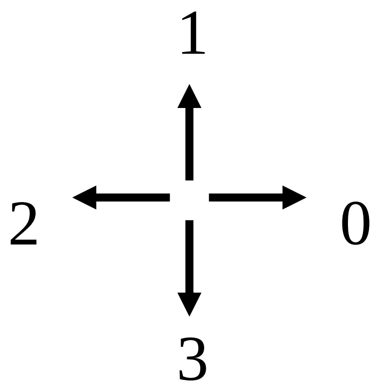


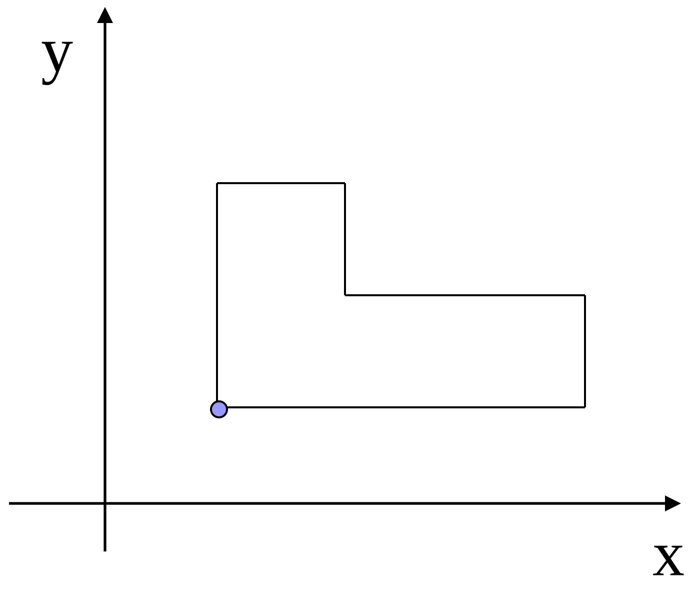
	Ordine 4	Ordine 6	
Codice catena:	0 3 2 1	0 0 3 2 2 1	
Differenza:	3 3 3 3	3 0 3 3 0 3	
Numero delle forme:	3 3 3 3	0 3 3 0 3 3	

	Ordine 3		
Codice catena:	0 0 3 3 2 2 1 1	0 3 0 3 2 2 1 1	0 0 0 3 2 2 2 1
Differenza:	3 0 3 0 3 0 3 0	3 3 1 3 3 0 3 0	3 0 0 3 3 0 0 3
Numero di forma:	0 3 0 3 0 3 0 3	0 3 0 3 3 1 3 3	0 0 3 3 0 0 3 3

Area computation with chain codes

```
1. Area=0;
2. y=0;
3. For each element of
   the chain code
     case direction:
       0: area+=y;
       1: y++;
       2: area-=y;
       3: y--;
```





Characteristic shapes

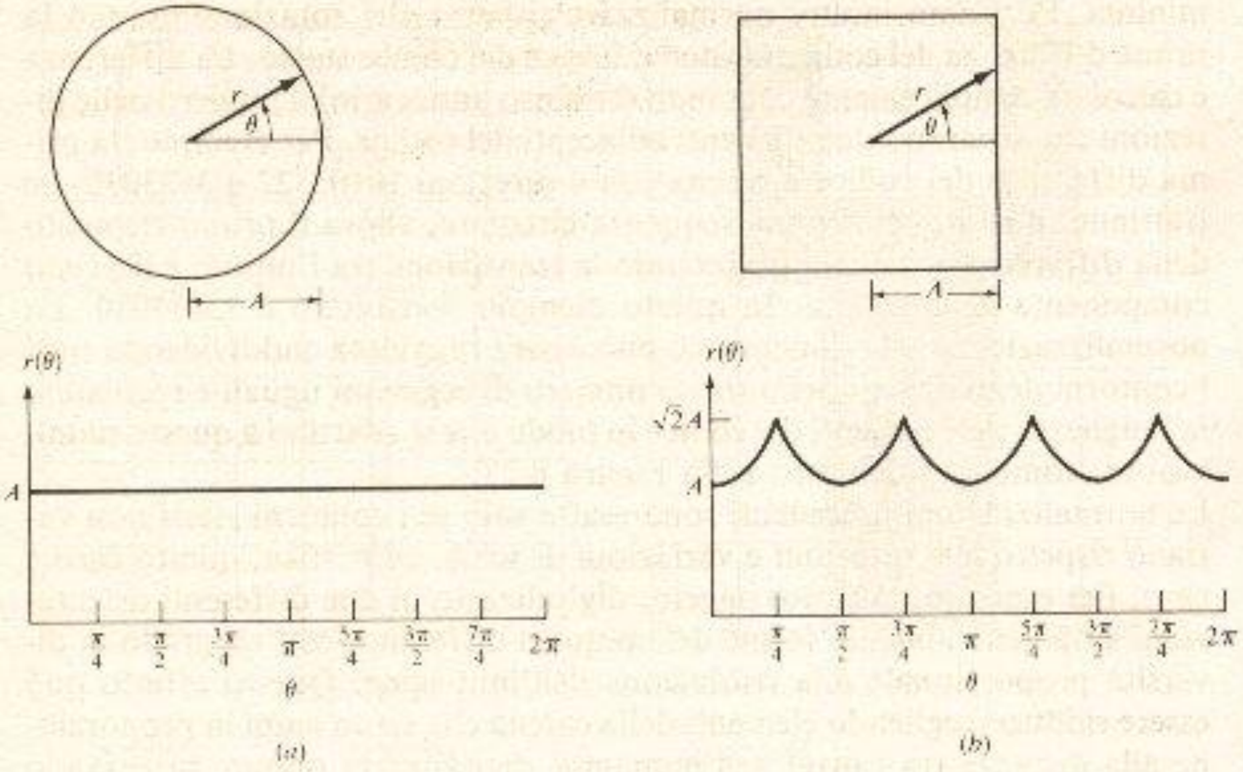


Figura 8.29 Due semplici forme di contorno e le rispettive forme caratteristiche con la distanza in funzione dell'angolo. In (*a*), $r(\theta)$ è costante, mentre in (*b*), $r(\theta) = A \sec \theta$.

Erosion

Erosion is a specific procedure of the more general Morphological Image Processing techniques.

It belongs to the concept of mathematical morphology and it is strictly related to the set theory.

Here the concept is roughly introduced to understand the basis of erosion.

Notation

Let consider A as a set in Z^2

Definition $a = (a_1, a_2)$ **belongs to** A

We write:

$a \in A$

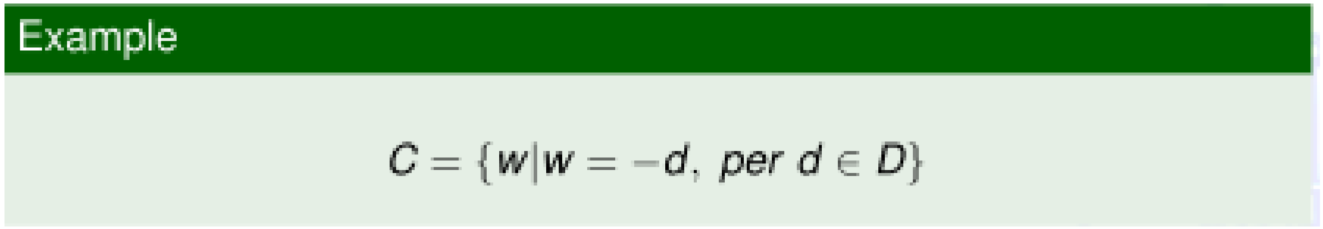
Definition $a = (a_1, a_2)$ **does not belong to** A

We write:

$a \notin A$

A set is represented by the parenthesis$\{\cdot\}$.

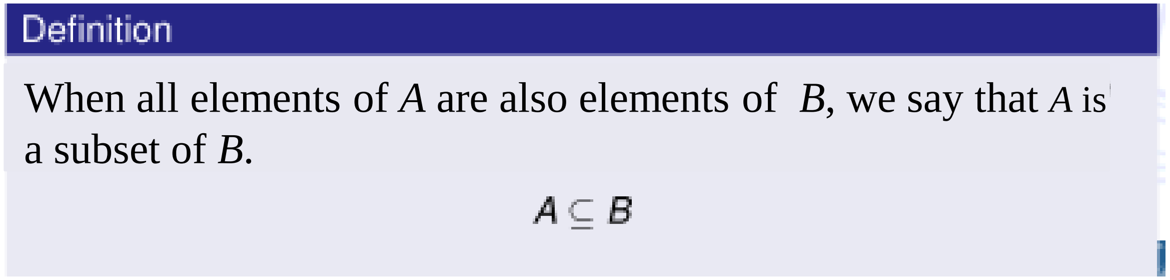
In our case, the elements of a set are the pixels belonging to a certain area or object of an image. When we write:



Example

$$C = \{w | w = -d, \text{ per } d \in D\}$$

This means that C is composed by all the elements w which are obtained by scalar product of the elements of D and the value -1.



Definition

When all elements of A are also elements of B, we say that A is a subset of B.

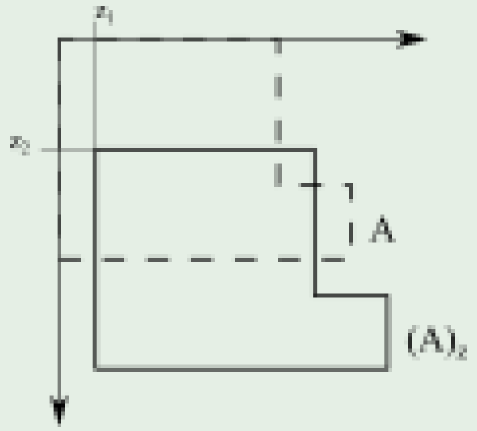
$$A \subseteq B$$

Definition

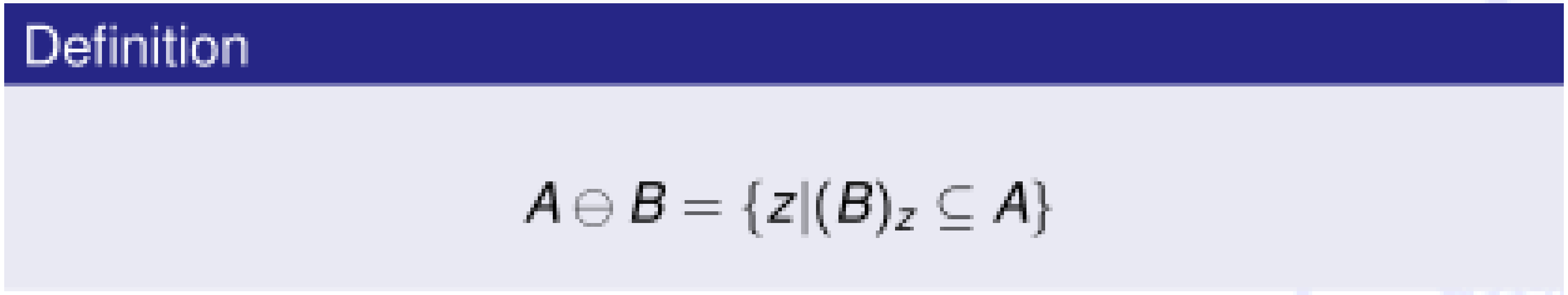
The translation of a set *A* by an element *z*, is represented as $(A)_z$ and is defined by:

$$(A)_z = \{c | c = a + z, \ per \ a \in A\}$$

Example



Now we can write the morphological operation which interest us, thus:


Definition

$$A \ominus B = \{z \mid (B)_z \subseteq A\}$$

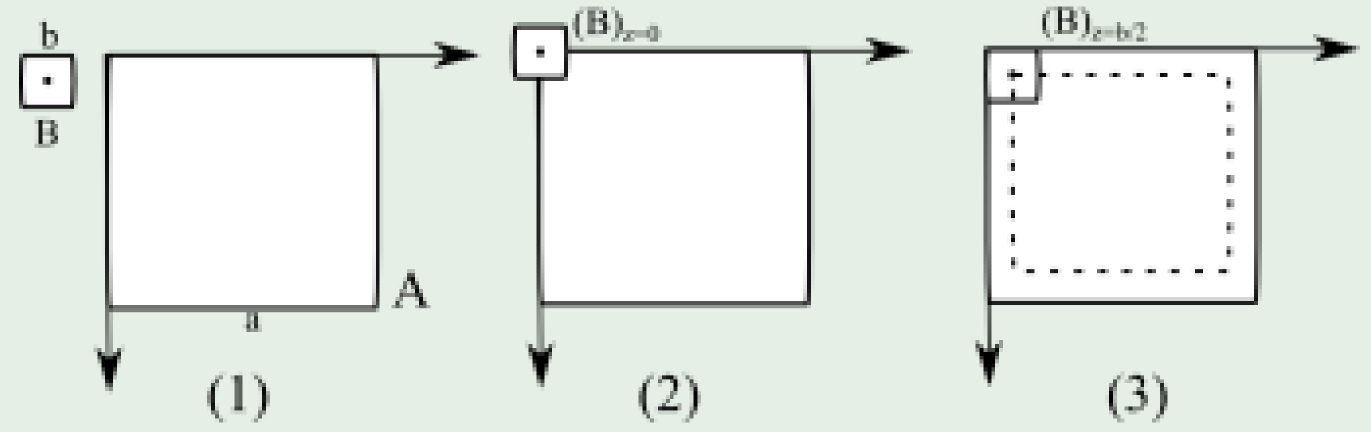
This definition represents an:

erosion

The verbose definition is: the erosion of *A* through *B* is the set of all the points *z* whom the translation of *B* by *z* is a subset of *A*.

It’s simple to see that graphically:

Example



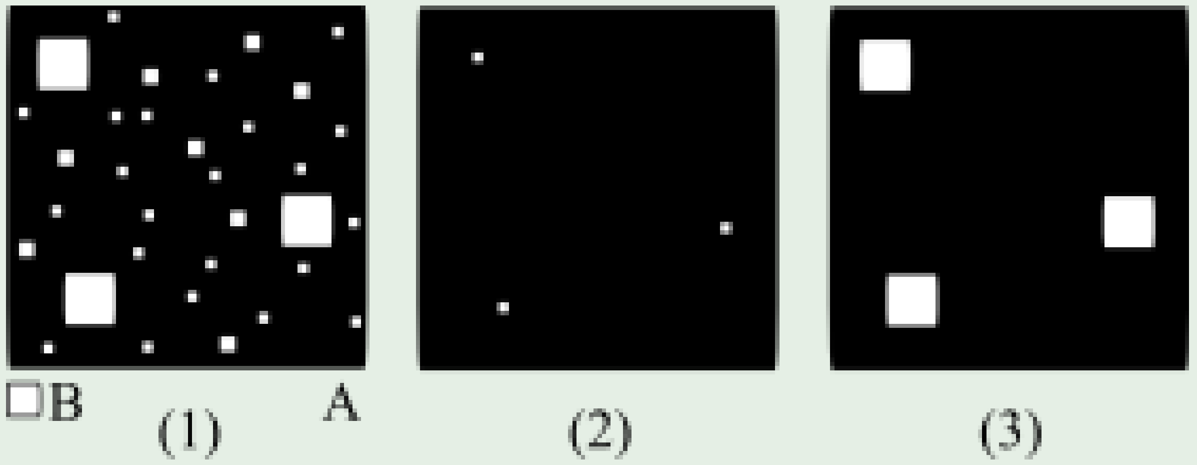
1. I due insiemi A e B.
2. $(B)_0 \subseteq A$? No   z=0
3. $(B)_0 \subseteq A$? Sì   z=b/2

In this case the erosed set will be;

$$A \ominus B = [\frac{b}{2} : a - \frac{b}{2}; \frac{b}{2} : a - \frac{b}{2}]$$

We can figure the erosion as a “shaped-cutting” of the most external part of the set.

Example



1) Original image
2) Erosion by the element B
3) Dilatation (the opposite procedure of the Erosion)

Pros:

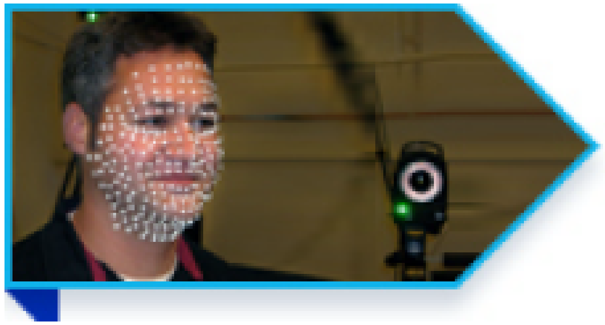
- It does not need edge detection
- It removes small noise artifacts (relaxing the denoising phase)

Cons:

- The object of interest should be bigger than the noise artifcats
- The erosion mask (shape and dimension) should be carefully evaluated

A che punto siamo:

- Abbiamo contrastato l'immagine
- Abbiamo binarizzato l'immagine
- Abbiamo individuato le coordinate di interesse
- Dobbiamo trasformare i punti 2d p=(x,y) nei punti 3d p=(x,y,z)

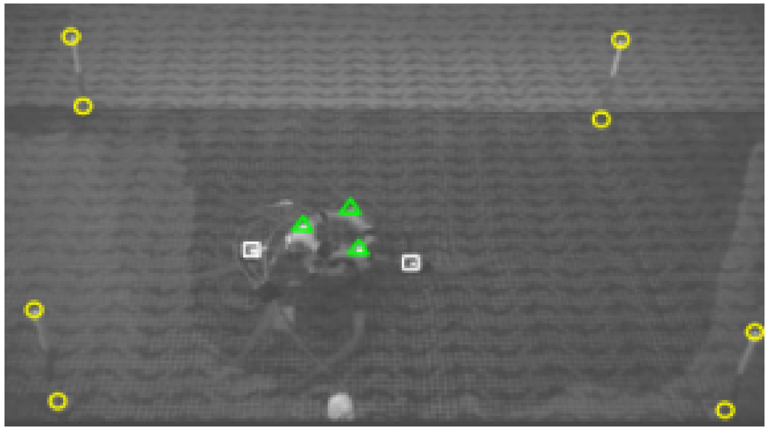


Ma prima...

Un altro po' di esempi...

Quale trasformazione è stata usata?

Per la pubblicazione, la lettura dell'immagine è lo scopo principale del processo di miglioramento.

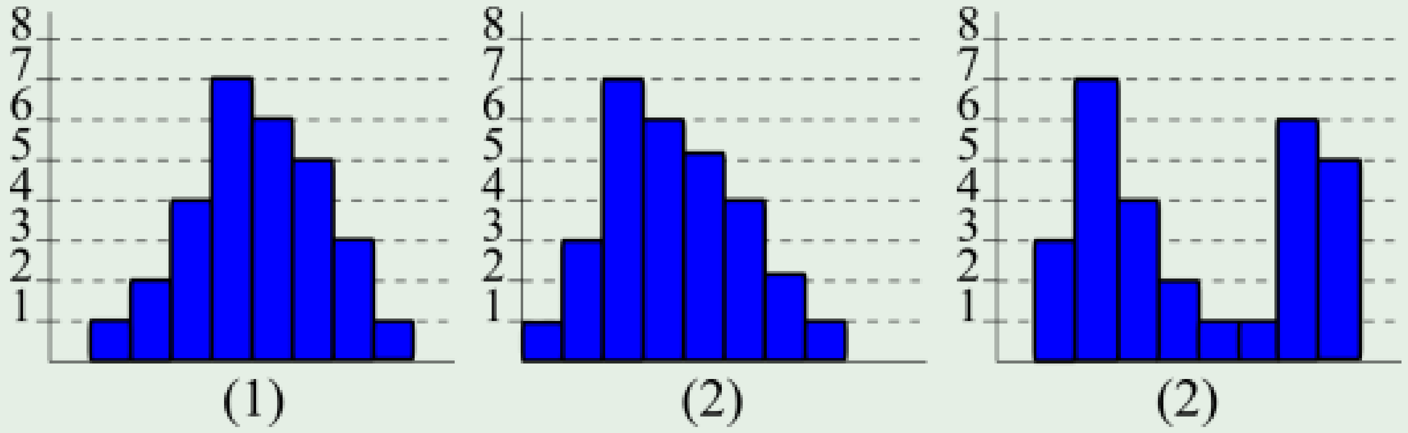


Quale trasformazione?

- trasformazioni di potenza
  - $\gamma > 1$
  - $\gamma = 1$
  - $\gamma < 1$
- trasformazioni sigmoidee
- trasformazioni lineari a tratti

A quale immagine si associano gli istogrammi

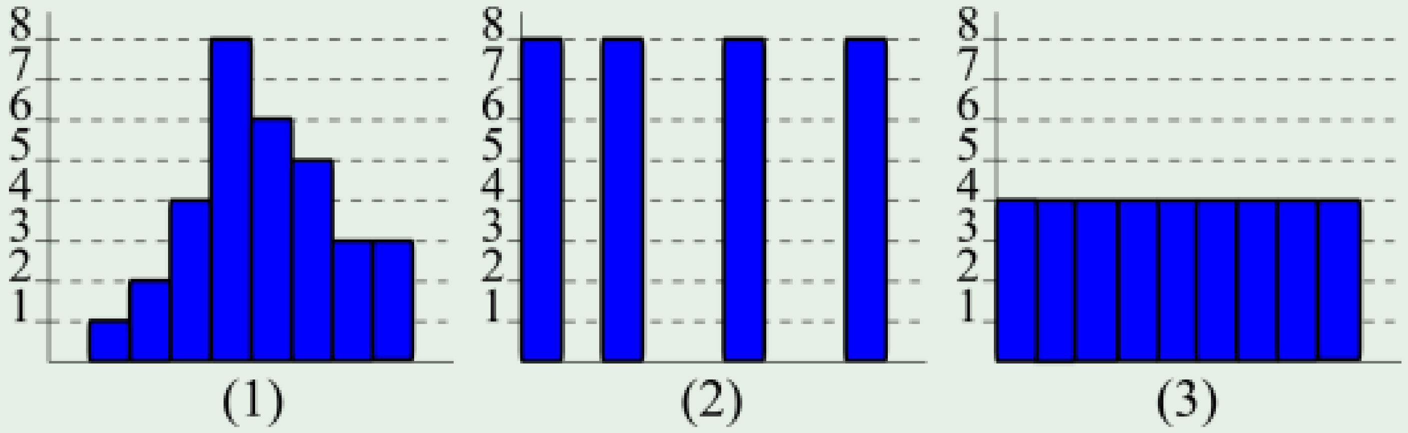
Example



1. Immagine originale
2. Immagine scurita
3. Immagine con contrasto aumentato

Quale istogramma è impossibile?

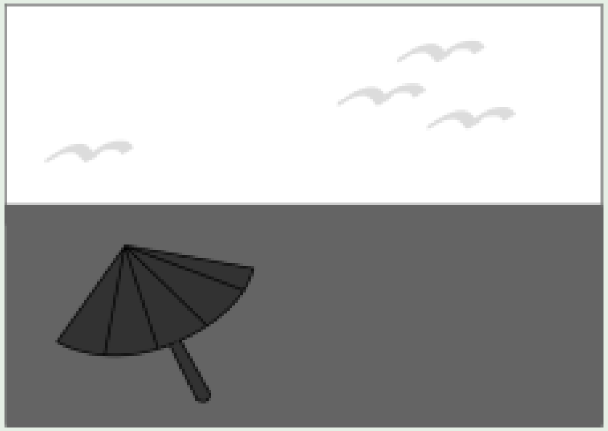
Example

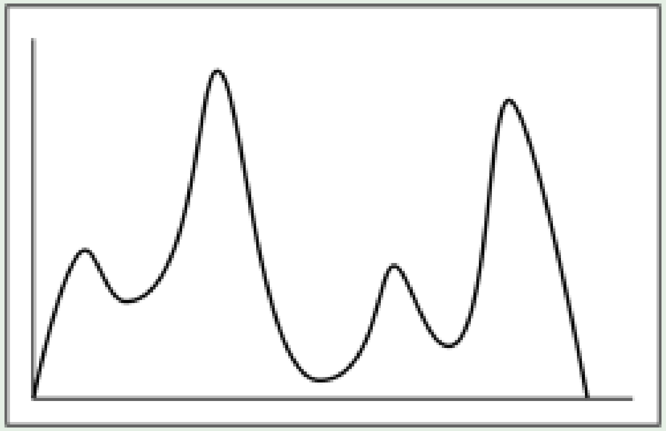


1. Istogramma originale
2. Istogramma numero 2
3. Istogramma numero 3

Più oggetti di interesse

Example



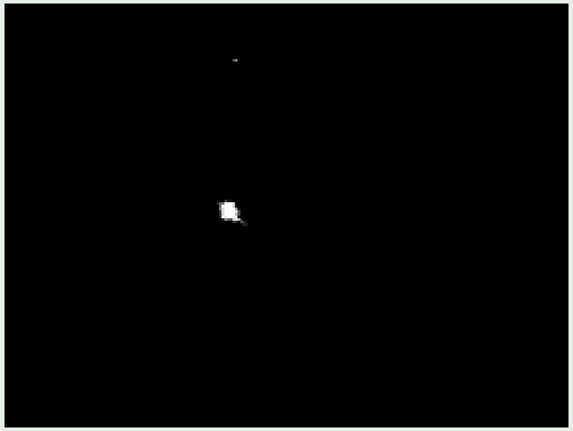


Come separo (automaticamente) tutti gli elementi?

- Reitero Otsu da 0 al livello soglia, e dal livello soglia al livello massimo!
- Opero la sogliatura localmente!

Marker con aloni

Example

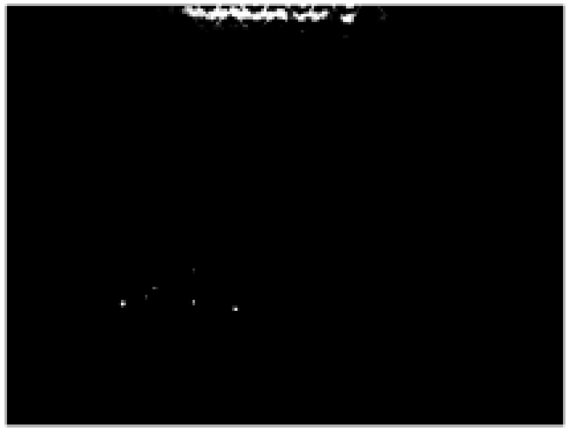


Quale procedura?

- Hough
- Gradiente
- Erosione

Regioni di interesse (ROI)

Abbiamo detto che immagini con grossi artefatti non sono adatte per essere l'erosione...



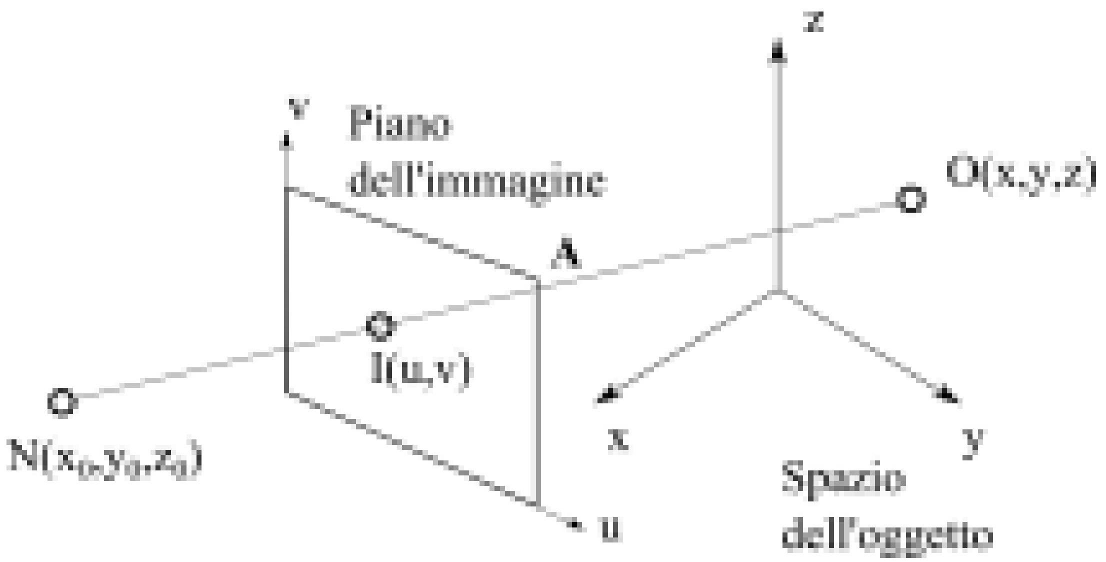
Come si può usare l'erosione su questa immagine?
Se sono sicuro che l'artefatto è fuori della mia regione di interesse (spazio dove si muove il robot, struttura anatomica in cui non ho una certa malattia,. . .) allora posso restringere il campo di applicazione

Direct Linear Transformation

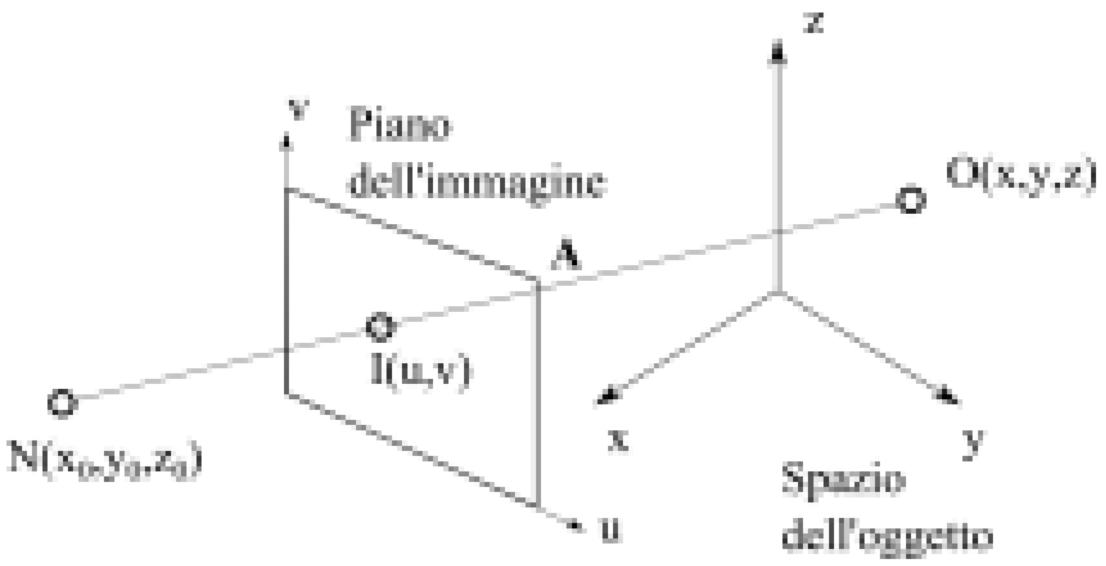
It is a simple transformation used for 3D reconstruction, which hypothesize the linearity between the object and its projection on the acquisition plane.

It requires more views of the scene and control points in the working space

Let’s consider the object *O* which is mapped directly on the projected image *I*.



O	object
N	projection centre
I	image pojected



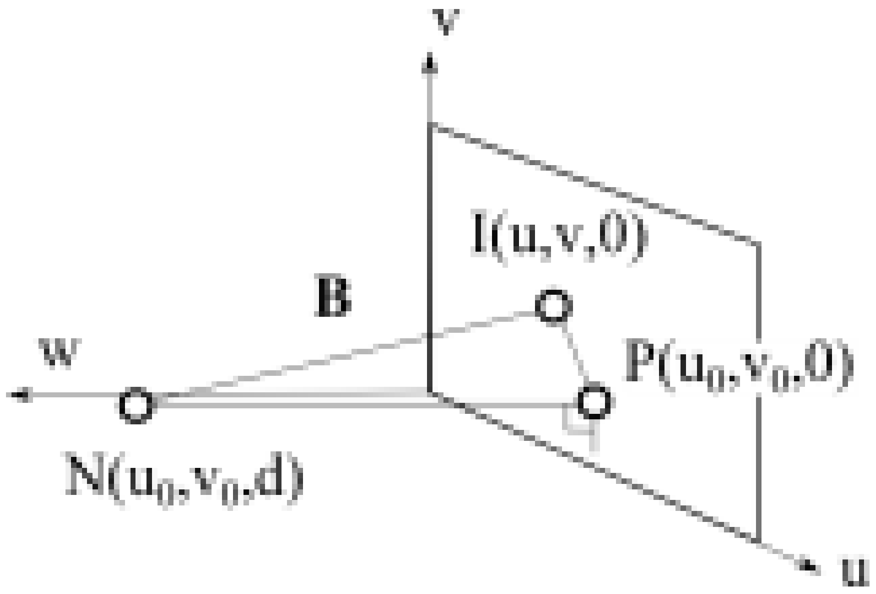
In the object space:

$$N = (x_0, y_0, z_0)$$

And the vector **A** from the object to the projection centre is:

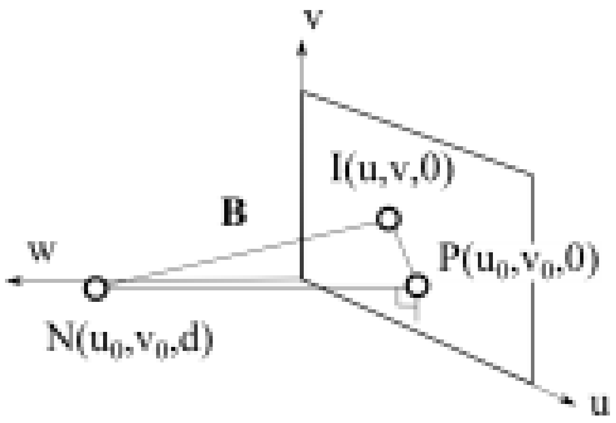
$$\mathbf{A} = (x - x_0, y - y_0, z - z_0)$$

It is also possible to write N with respect to the image plane reference frame.



P is called principal point and d principal distance

The line NP is parallel to the w axis and orthogonal to the *uv* plane



The vector **B** is thus:

$$\mathbf{B} = (u - u_0, v - v_0, -d)$$

And for the co-linearity hypothesis:

$$\mathbf{B} = c\,\mathbf{A}$$

Where c is a scalar.
It is worth noticing that **B** is expressed in the image reference frame, while **A** is expressed in the object reference frame.

We can express **A** with respect to the image reference frame:

$$\mathbf{A}^i = \mathbf{T}_{i/o}\mathbf{A}^o = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \mathbf{A}^o \qquad (1)$$

Where $\mathbf{A^i}$ is **A** expressed with respect to the image reference frame, while $\mathbf{A^o}$ is expressed with respect to the object space. $\mathbf{T_{i/o}}$ is the transformation from the object-to-image reference frame. We do not know $\mathbf{T_{i/o}}$ a priori.

We put in (1) the expression of **B** and **A^o**:

$$\begin{bmatrix} u - u_0 \\ v - v_0 \\ -d \end{bmatrix} = c \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix} \qquad (2)$$

The product is:

$$\begin{aligned} u - u_0 &= c[r_{11}(x - x_0) + r_{12}(y - y_0) + r_{13}(z - z_0)] \\ v - v_0 &= c[r_{21}(x - x_0) + r_{22}(y - y_0) + r_{23}(z - z_0)] \\ -d &= c[r_{31}(x - x_0) + r_{32}(y - y_0) + r_{33}(z - z_0)] \qquad (3) \end{aligned}$$

By solving (3) with respect to *c*:

$$c = \frac{-d}{r_{31}(x-x_0)+r_{32}(y-y_0)+r_{33}(z-z_0)} \tag{4}$$

and substituting (4) in (3):

$$\begin{aligned} u-u_0 &= -d\frac{r_{11}(x-x_0)+r_{12}(y-y_0)+r_{13}(z-z_0)}{r_{31}(x-x_0)+r_{32}(y-y_0)+r_{33}(z-z_0)} \\ v-v_0 &= -d\frac{r_{21}(x-x_0)+r_{22}(y-y_0)+r_{23}(z-z_0)}{r_{31}(x-x_0)+r_{32}(y-y_0)+r_{33}(z-z_0)} \end{aligned} \tag{5}$$

Standard DLT equations:

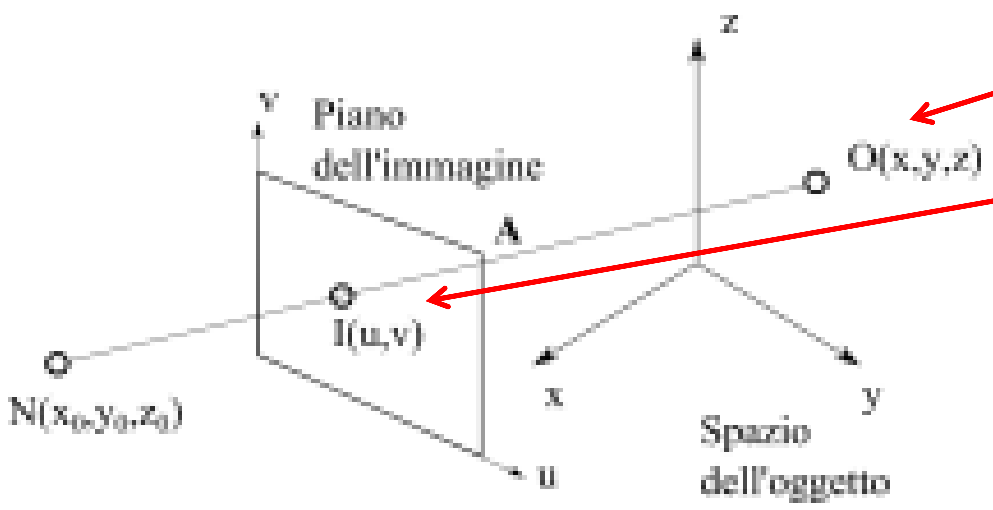
Solving (5) for the image coordinates:

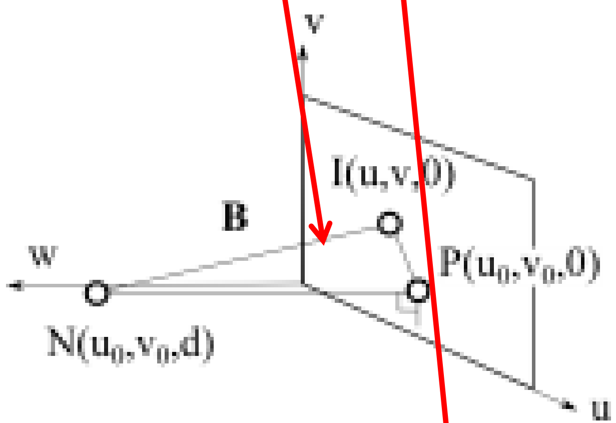
$$u = \frac{L_1 x + L_2 y + L_3 z + L_4}{L_9 x + L_{10} y + L_{11} z + 1}$$
$$v = \frac{L_5 x + L_6 y + L_7 z + L_8}{L_9 x + L_{10} y + L_{11} z + 1} \qquad (6)$$

Where L_1 … L_{11} are called DLT parameters.

$$u = \frac{L_1x + L_2y + L_3z + L_4}{L_9x + L_{10}y + L_{11}z + 1}$$

$$v = \frac{L_5x + L_6y + L_7z + L_8}{L_9x + L_{10}y + L_{11}z + 1}$$





Equazione DLT standard

Variabili

- u e v sono le coordinate nel piano-immagine del punto da ricostruire
- x,y e z sono le coordinate nello spazio oggetto del punto da ricostruire

Parametri

- $L_1 \cdots L_{11}$ sono parametri che dipendono dal sistema di acquisizione, e rimangono fissi una volta derivati
- la derivazione di questi parametri si chiama calibrazione

$$\mathbf{A}^i = \mathbf{T}_{i/o}\mathbf{A}^o = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \mathbf{A}^o$$

Calibration

For calibration we need to rearrange (6)*:

$$\frac{1}{R}u = \frac{1}{R}(L_1x + L_2y + L_3z + L_4 - L_9ux - L_{10}uy - L_{11}uz)$$
$$\frac{1}{R}v = \frac{1}{R}(L_1x + L_2y + L_3z + L_4 - L_9vx - L_{10}vy - L_{11}vz)$$

dove $R = L_9x + L_{10}y + L_{11}z + 1$. In vectorial form:

$$\frac{1}{R}\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{R}\begin{bmatrix} x & y & z & 1 & 0 & 0 & 0 & 0 & -ux & -uy & -uz \\ 0 & 0 & 0 & 0 & x & y & z & 1 & -vx & -vy & -vz \end{bmatrix}\begin{bmatrix} L_1 \\ \vdots \\ L_{11} \end{bmatrix}$$

Thus:

Y=**X L**

The aim of the calibration is to find the vector **L**, thus is required to know the matrixes **X** and **Y**.

*Note (example of rearrangement for *u*):

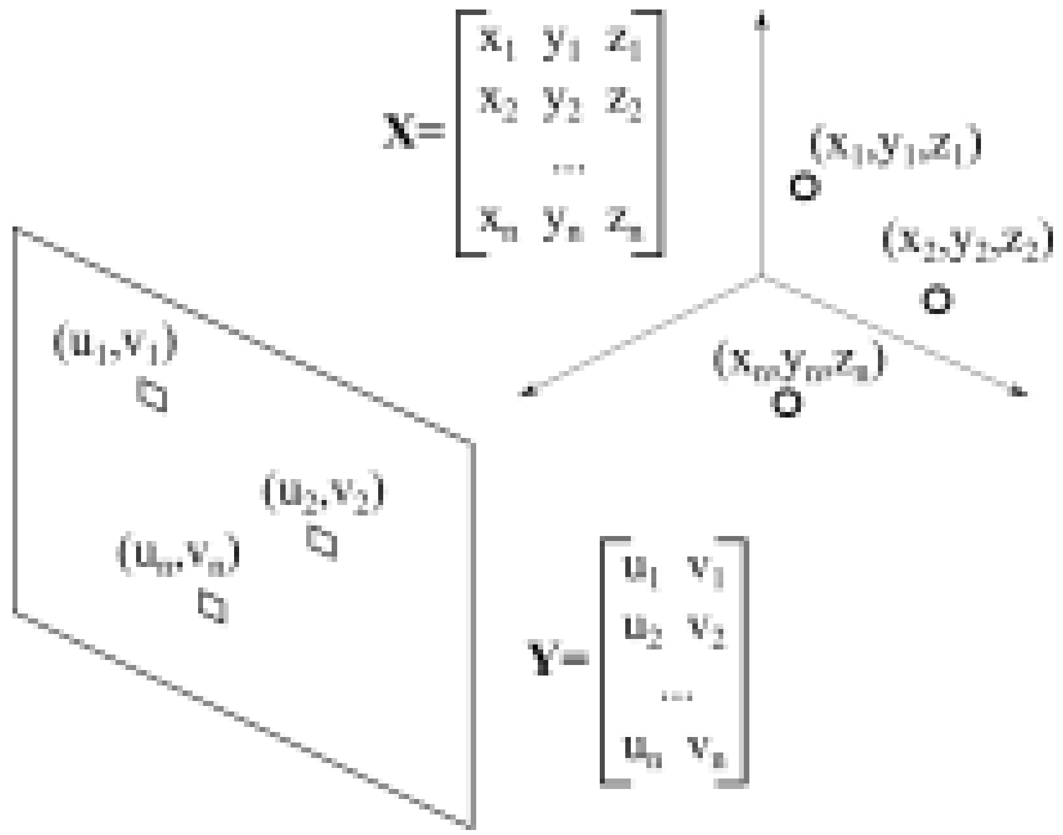
$$u = \frac{L_1 x + L_2 y + L_3 z + L_4}{L_9 x + L_{10} y + L_{11} z + 1} \qquad \text{with} \qquad R = L_9 x + L_{10} y + L_{11} z + 1$$

$$u = \frac{L_1 x + L_2 y + L_3 z + L_4}{R}; \qquad \frac{R}{R} u = \frac{L_1 x + L_2 y + L_3 z + L_4}{R};$$

$$\frac{L_9 xu + L_{10} yu + L_{11} zu + u}{R} = \frac{L_1 x + L_2 y + L_3 z + L_4}{R};$$

$$\frac{u}{R} = \frac{L_1 x + L_2 y + L_3 z + L_4 - L_9 xu - L_{10} yu - L_{11} zu}{R}.$$

Same goes for *v*.



$\mathbf{X}=\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ & \dots & \\ x_n & y_n & z_n \end{bmatrix}$
(x_1,y_1,z_1)
(x_2,y_2,z_2)
(x_n,y_n,z_n)
(u_1,v_1)
(u_2,v_2)
(u_n,v_n)
$\mathbf{Y}=\begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \\ \dots & \\ u_n & v_n \end{bmatrix}$

We can consider *n* points in the space reference frame with known 3d coordinates **X**, then we measure the 2d coordinates **Y** in the image reference frame of the same points. These known positions are called *control points*.

Keep in mind that:

▪R depends on L9 … L11, thus we need to hypothesize some initial values and then proceed in iterative way

▪ The system can be solved with the Least Square Method, however for this method it is required to have a number of control points greater than the parameters to be derived

Calibration equations for *n* control points:

$$\begin{bmatrix} \frac{x_1}{R_1} & \frac{y_1}{R_1} & \frac{z_1}{R_1} & \frac{1}{R_1} & 0 & 0 & 0 & 0 & \frac{-u_1 x_1}{R_1} & \frac{-u_1 y_1}{R_1} & \frac{-u_1 z_1}{R_1} \\ 0 & 0 & 0 & 0 & \frac{x_1}{R_1} & \frac{y_1}{R_1} & \frac{z_1}{R_1} & \frac{1}{R_1} & \frac{-v_1 x_1}{R_1} & \frac{-v_1 y_1}{R_1} & \frac{-v_1 z_1}{R_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{x_n}{R_n} & \frac{y_n}{R_n} & \frac{z_n}{R_n} & \frac{1}{R_n} & 0 & 0 & 0 & 0 & \frac{-u_n x_n}{R_n} & \frac{-u_n y_n}{R_n} & \frac{-u_n z_n}{R_n} \\ 0 & 0 & 0 & 0 & \frac{x_n}{R_n} & \frac{y_n}{R_n} & \frac{z_n}{R_n} & \frac{1}{R_n} & \frac{-v_n x_n}{R_n} & \frac{-v_n y_n}{R_n} & \frac{-v_n z_n}{R_n} \end{bmatrix} \begin{bmatrix} L_1 \\ \vdots \\ L_{11} \end{bmatrix} = \begin{bmatrix} \frac{u_1}{R_1} \\ \frac{v_1}{R_1} \\ \vdots \\ \frac{u_n}{R_n} \\ \frac{v_n}{R_n} \end{bmatrix}$$

$$\begin{bmatrix} \frac{x_1}{R_1} & \frac{y_1}{R_1} & \frac{z_1}{R_1} & \frac{1}{R_1} & 0 & 0 & 0 & 0 & \frac{-u_1 x_1}{R_1} & \frac{-u_1 y_1}{R_1} & \frac{-u_1 z_1}{R_1} \\ 0 & 0 & 0 & 0 & \frac{x_1}{R_1} & \frac{y_1}{R_1} & \frac{z_1}{R_1} & \frac{1}{R_1} & \frac{-v_1 x_1}{R_1} & \frac{-v_1 y_1}{R_1} & \frac{-v_1 z_1}{R_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{x_n}{R_n} & \frac{y_n}{R_n} & \frac{z_n}{R_n} & \frac{n}{R_n} & 0 & 0 & 0 & 0 & \frac{-u_n x_n}{R_n} & \frac{-u_n y_n}{R_n} & \frac{-u_n z_n}{R_n} \\ 0 & 0 & 0 & 0 & \frac{x_n}{R_n} & \frac{y_n}{R_n} & \frac{z_n}{R_n} & \frac{n}{R_n} & \frac{-v_n x_n}{R_n} & \frac{-v_n y_n}{R_n} & \frac{-v_n z_n}{R_n} \end{bmatrix} \begin{bmatrix} L_1 \\ \vdots \\ L_{11} \end{bmatrix} = \begin{bmatrix} \frac{u_1}{R_1} \\ \frac{v_1}{R_1} \\ \vdots \\ \frac{u_n}{R_n} \\ \frac{v_n}{R_n} \end{bmatrix}$$

We have 11 parameters, and since each control point provides 2 equations, they are required at least 6 control points.

L is derived as follows:

$$\begin{aligned} \mathbf{X} \cdot \mathbf{L} &= \mathbf{Y} \\ (\mathbf{X}^t \cdot \mathbf{X}) \cdot \mathbf{L} &= \mathbf{X}^t \cdot \mathbf{Y} \\ (\mathbf{X}^t \cdot \mathbf{X})^{-1} (\mathbf{X}^t \cdot \mathbf{X}) \cdot \mathbf{L} &= (\mathbf{X}^t \cdot \mathbf{X}) \cdot (\mathbf{X}^t \cdot \mathbf{Y}) \\ \mathbf{L} &= (\mathbf{X}^t \cdot \mathbf{X})^{-1} \cdot (\mathbf{X}^t \cdot \mathbf{Y}) \end{aligned}$$

Reconstruction

Once the calibration parameters L were derived, we should rearrange (6) to solve the equations with respect to the 3d coordinates:

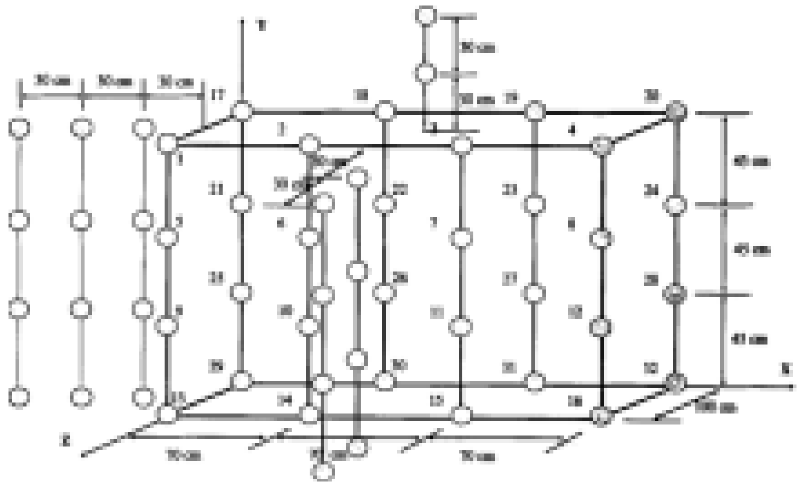
$$\begin{bmatrix} \frac{u^1 L_9^1 - L_1^1}{R^1} & \frac{u^1 L_{10}^1 - L_2^1}{R^1} & \frac{u^1 L_{11}^1 - L_3^1}{R_1} \\ \frac{v^1 L_9^1 - L_1^1}{R^1} & \frac{v^1 L_{10}^1 - L_2^1}{R^1} & \frac{v^1 L_{11}^1 - L_3^1}{R_1} \\ \frac{u^m L_9^m - L_1^m}{R^m} & \frac{u^m L_{10}^m - L_2^m}{R^m} & \frac{u^m L_{11}^m - L_3^m}{R_m} \\ \frac{v^m L_9^m - L_1^m}{R^m} & \frac{v^m L_{10}^m - L_2^m}{R^m} & \frac{v^m L_{11}^m - L_3^m}{R_m} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{L_4^1 - u^1}{R^1} \\ \frac{L_8^1 - v^1}{R^1} \\ \frac{L_4^m - v^m}{R^m} \\ \frac{L_8^m - u^m}{R^m} \end{bmatrix}$$

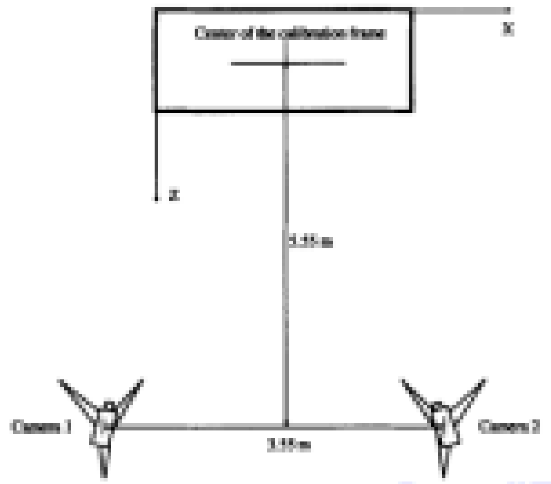
Notice that, in analogy with the calibration, also in the reconstruction we have 3 variables and only 2 equations each point: in this case it is necessary to have more views of the scene, thus the number of cameras required are *m* equal or greater than 2.

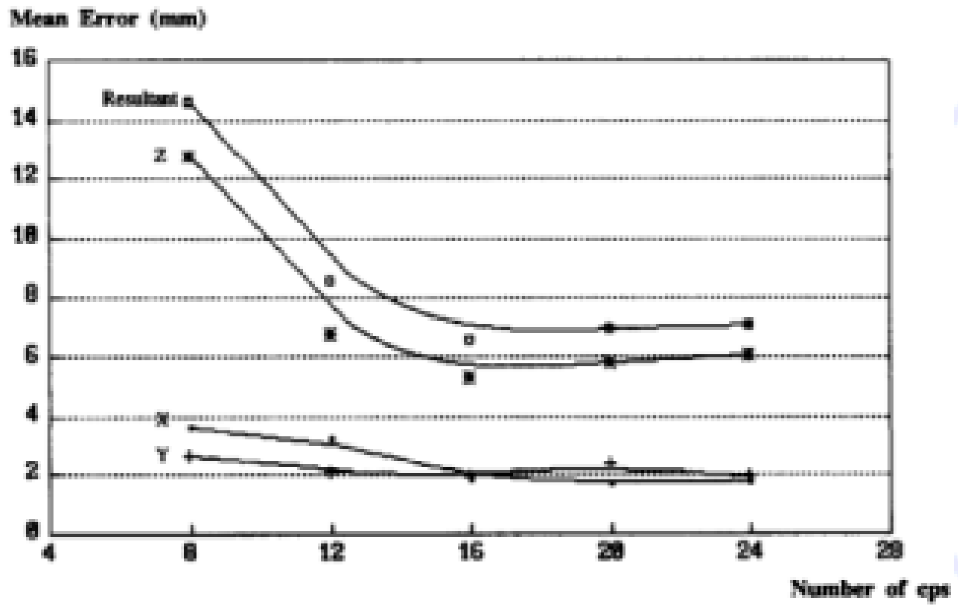
Final remarks

- Each camera should be calibrated, thus it should be performed at least 2 times
- Camera principal axis should not be collinear, thus the cameras should not face each other
- Control points should not belong to the same plane, thus they should define a volume
- The objects to be reconstructed should belong to the calibration space
- Increasing the number of control points increase the accuracy of the reconstrcution
- Increasing the number of cameras increase the quality of the reconstruction and also avoids occlusions

Accuracy considerations







Optic flow

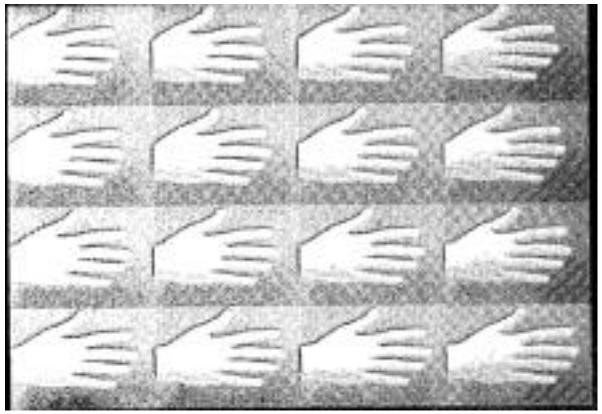
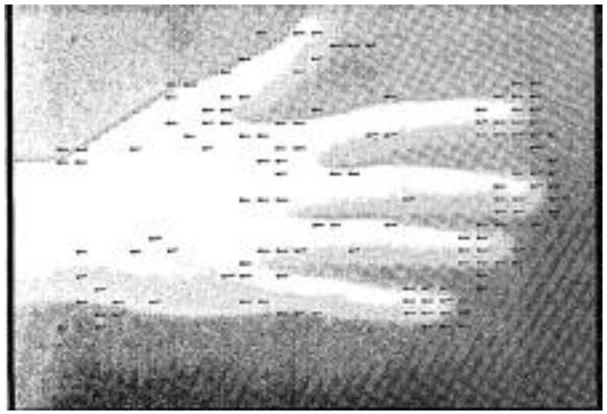
The motion is a significant part of our visual process, and it is used for several purposes:

- to recognize tridimensional shapes
- to control the body by the oculomotor control
- to organize perception
- to recognize object
- to predict actions
- …

Optic flow

A surface or object moving in the space projects in the image plane a bidimensional path of speeds, *dx/dt* and *dy/dt* that is often referred to as *bidimensional motor field*.

The aim of the *optic flow* is to approximate the variation over time of the intensity levels of the image.

We consider that the intensity *I* of a pixel *(x, y)* at the instant *t*, moves to a neighbor pixel in the instant *t+dt*, thus:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \qquad (7)$$

Expanding in Taylor serie:

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt$$

and taking as a reference (7) :

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = 0$$

This is the *gradient constraint equation* :

$$\frac{\partial I}{\partial x}\dot{x} + \frac{\partial I}{\partial y}\dot{y} + \frac{\partial I}{\partial t} = 0$$

That can be rewritten as:

$$\nabla I \cdot V^T + \frac{\partial I}{\partial t} = 0$$

with $V=(\dot{x}, \dot{y})$

Since the gradient constraint equation has two variables, it cannot be solved directly. This is called the *aperture problem*.

Lucas-Kanade hypothesis

To solve the aperture problem, they hypothesize:

- the motion of the intensity of pixel among two subsequent frames is small
- the motion in a small local neighbor of the pixel is constant

This is equivalent to say that the optical flow is constant for each pixel centered in *p*, thus:

$$\begin{aligned} I_x(q_1)\dot{x} + I_y(q_1)\dot{y} &= -I_t(q_1) \\ I_x(q_1)\dot{x} + I_y(q_1)\dot{y} &= -I_t(q_1) \\ &\vdots \\ I_x(q_1)\dot{x} + I_y(q_1)\dot{y} &= -I_t(q_1) \end{aligned}$$

Where $q_1, \ldots, q_n$ are pixel of the window centered in p and $I_{x,y,t}$ are the derivative with respect to x,y,t

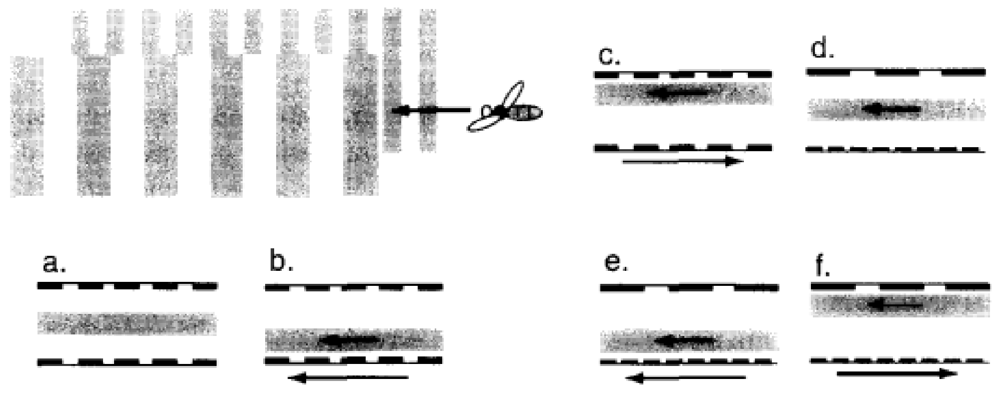
By writing the equations in vectorial form ***A·v* = *b*** where:

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix}, v = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}, b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}$$

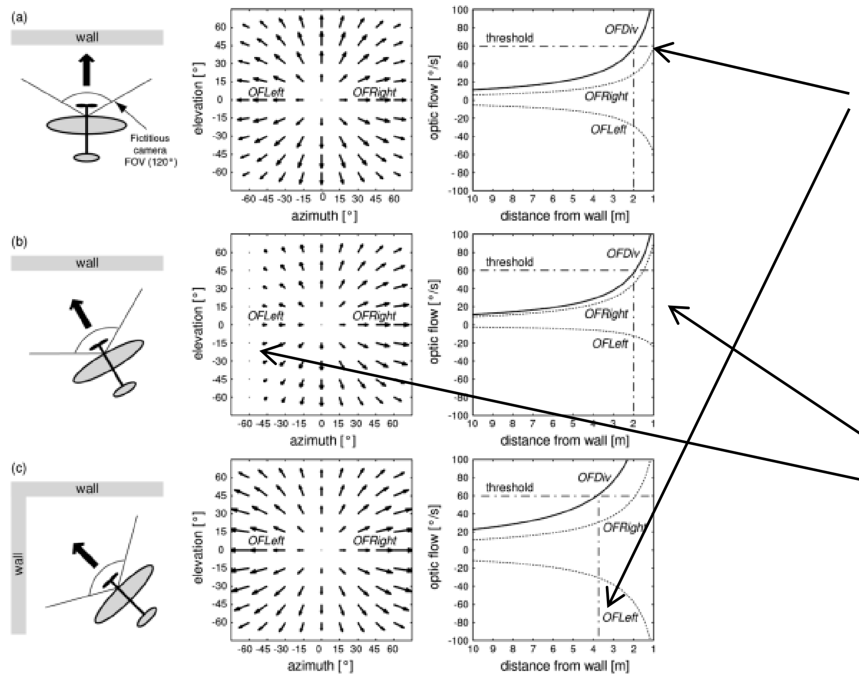
This system can be solved with the least square method:

$$v = (A^T A)^{-1} A^T b$$

Bee-inspired navigation control



The optic flow provides information for wall following and landing

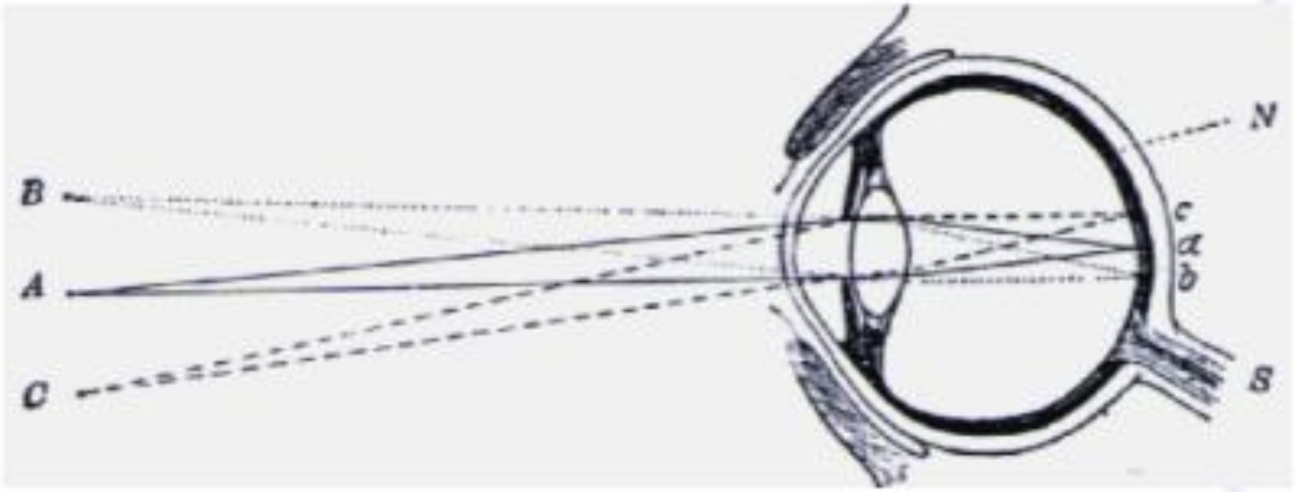


The difference is inversely proportional to the distance from an obstacle

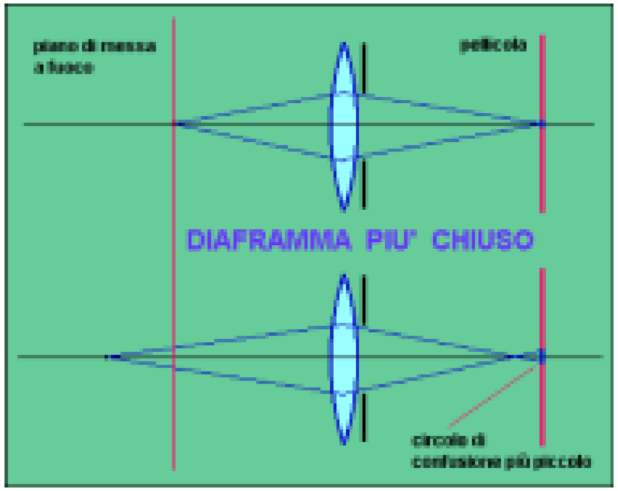
The absolute difference defines the turning behaviour

Focus and blur

The image is projected by the lens onto the focal plane



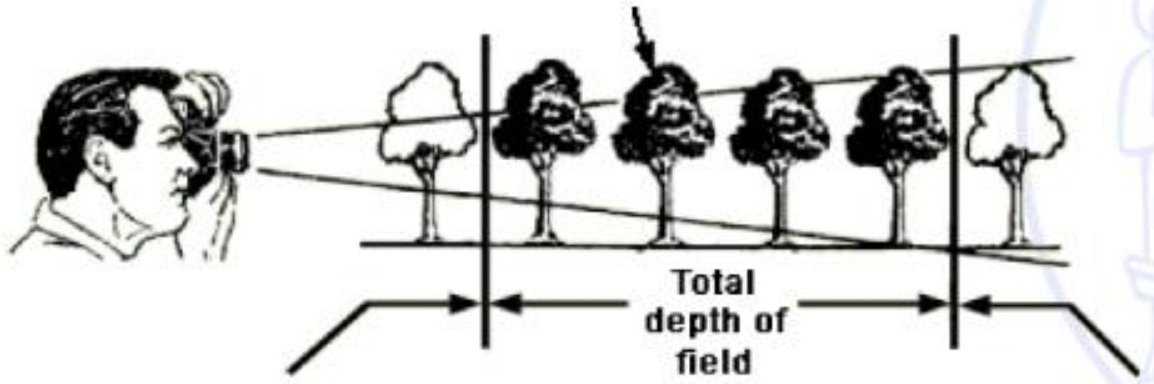
When the convergence of the ray is not exactly on the sensor plane, the edges of the image are fuzzy, and the image is called blurred



This mean that a point is no more projected as a point but as a small disk, called circle of confusion, which size defines the blur intensity.

Depth of field

Also the elements before and after the fixation plane are blurred: the portion of space where an image is considered in focus, is called *depth of field*



Depth of field

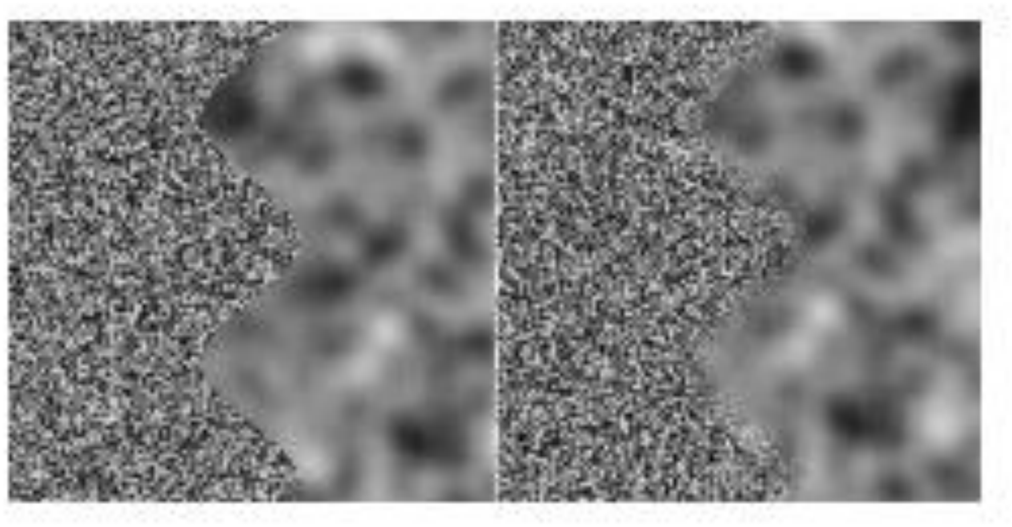
$$
\begin{aligned}
H &= \frac{f^2}{N \cdot c} + f \\
D_n &= \frac{s(H-f)}{H+s-2f} \\
D_f &= \frac{s(H-f)}{H-s}
\end{aligned}
$$

DoF increase:

1. when f-number increase, N
2. when the subject is distant, s
3. when the focal distance of the objective decrease, f

Perception of the distance

Studies on blurred images demonstrate that blur intensity influences the distance perception.



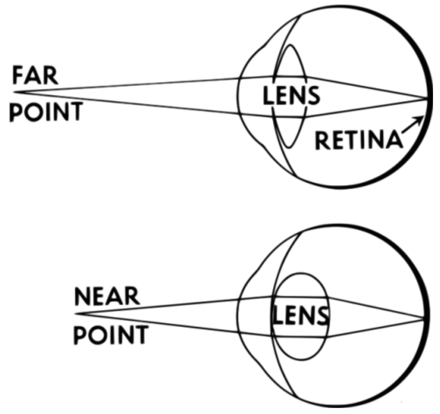
Focussed images are perceived as closer with respect to blurred images, but a precise evaluation of the distance is not provided by this clue.

Accommodation and retinal blur

Depth perception is mainly monocular over 30 meters, while it is mainly binocular under this threshold.

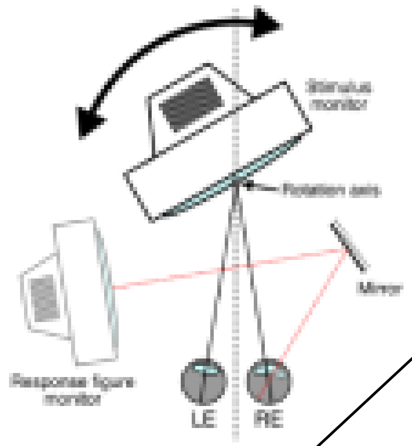
Several clues are used to perceive and evaluate the depth:

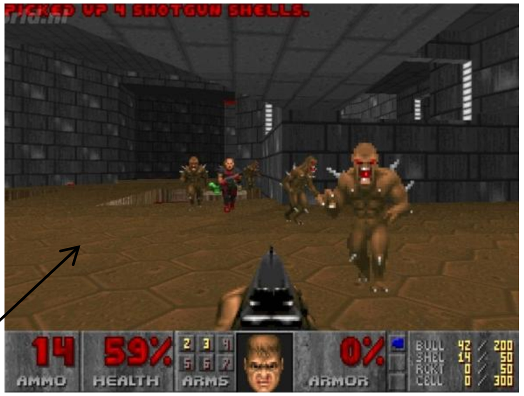
1. Dimension
2. Parallaxes
3. ...
4. Accommodation
5. Retinal blur

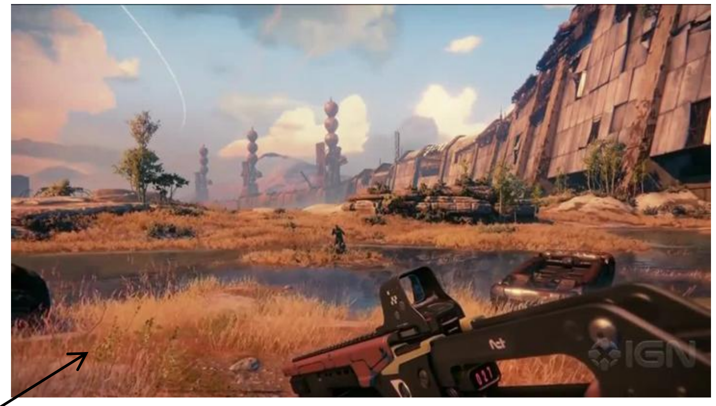


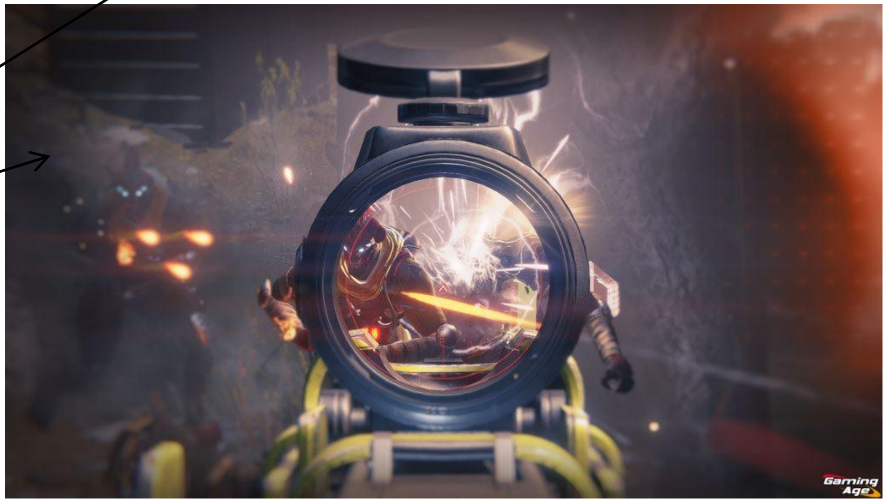
Some divergence among clues “deform” the depth evaluation, or produce an “unreality” perception.

Videogames often provide contrasting clues: the accommodation is provided for the actual distance of the screen, but images (blur prospective, etc...) provide information about other distances.









Totally sharp

Progressive blur

How is it possible to evaluate the blur?

The blur can be evaluated by:

- direct means: measuring the distance from the subject, and moving the focusing system
- indirect means: several images of the same scene are captured, then a metric defines which one is focussed (or blurred).

Marziliano *et al*, (2008), introduced a simple metric based on the edges: the concept is that a blurred edge is wider than a sharp one.

Their metric is independent from the blur source:

- movement
- optical
- aberration
- . . .

Both vertical or horizontal edges can be used. Practically, vertical edges obtained with a Sobel mask are used.

Alg

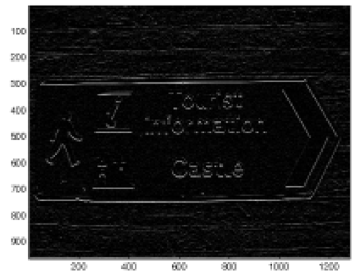
1. Find the edges
2. Evaluate the width of the local edge
3. Calculate the average of the edge's width

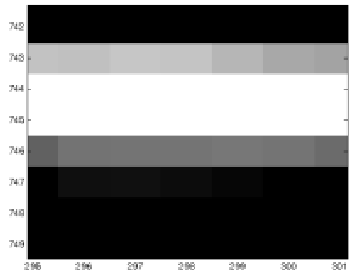
Example

Focussed image

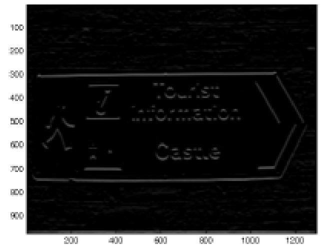
Light blurred image

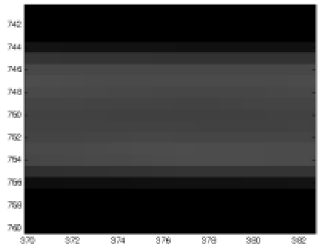
Example





Example

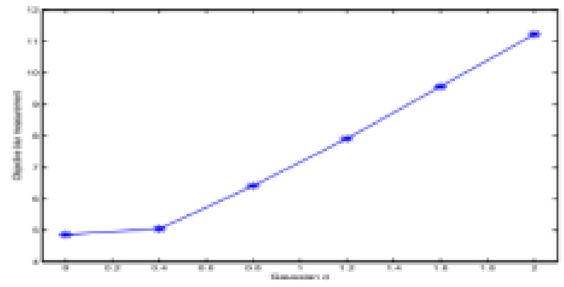




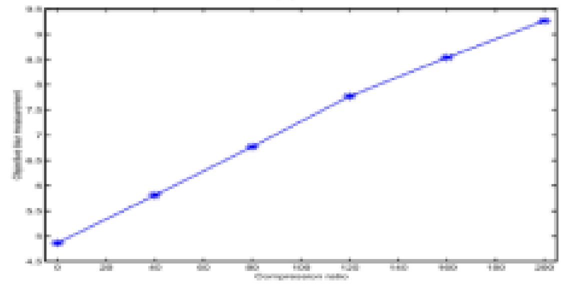
It should be noted that also the spikes of the images are cancelled

Blur measure

The first graph identifies the blur measured with the metric, and the second plot the effective blur.

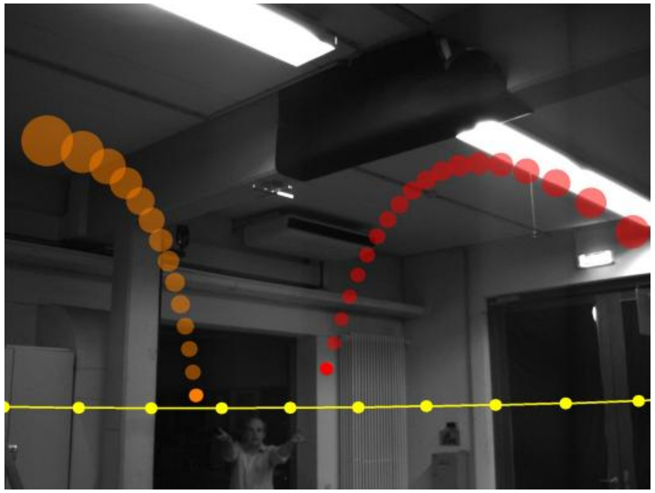


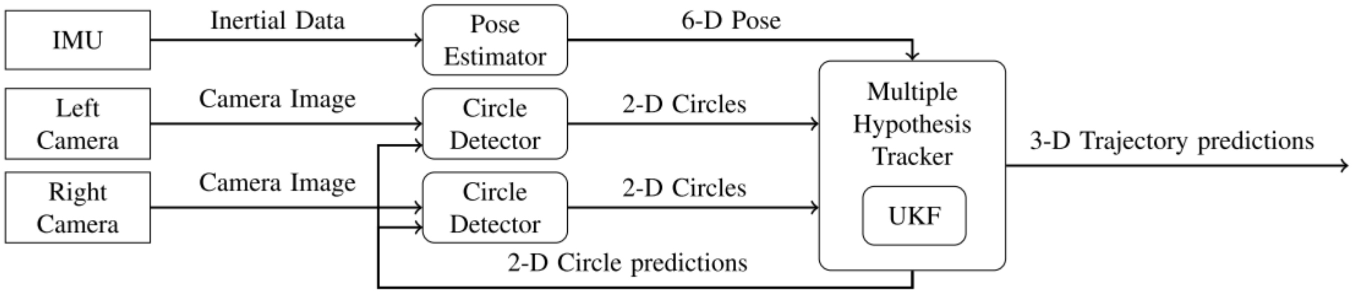
(a)

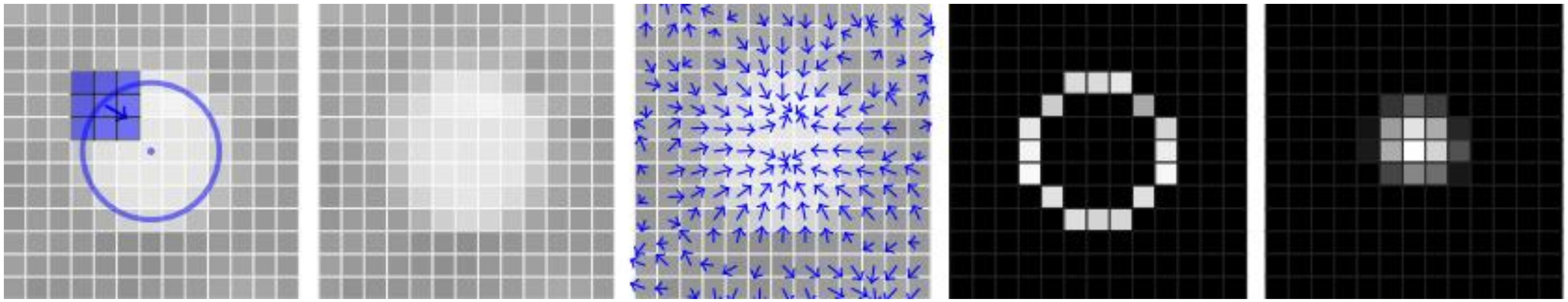


Example

Esempio: Justin





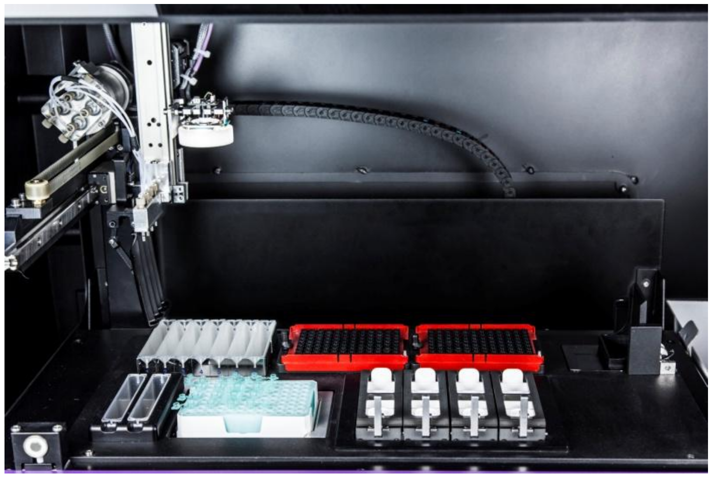


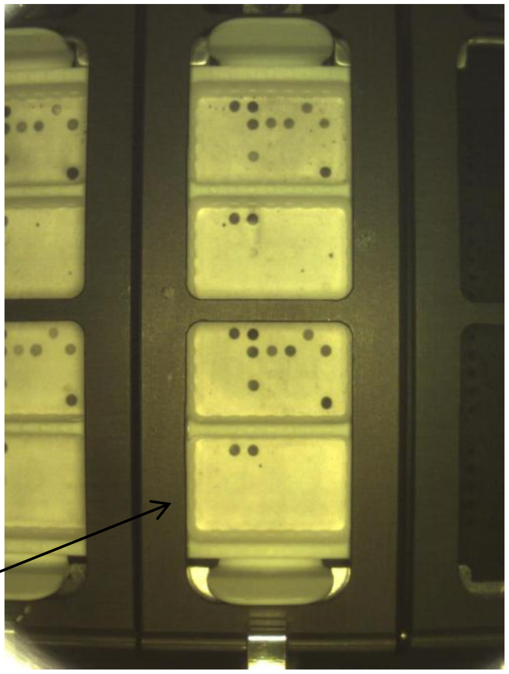
$$C = \frac{\sqrt{2}\left(\left(\begin{smallmatrix}+1 & 0 & -1\\ +2 & 0 & -2\\ +1 & 0 & -1\end{smallmatrix}\right) * I,\ \left(\begin{smallmatrix}+1 & +2 & +1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{smallmatrix}\right) * I\right)^T}{\sqrt{16\left(\begin{smallmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{smallmatrix}\right) * I^2 - \left(\left(\begin{smallmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{smallmatrix}\right) * I\right)^2 + \varepsilon^2}}$$

$$R(x,y,\alpha) = \left(\begin{pmatrix}\cos\alpha\\ \sin\alpha\end{pmatrix} \cdot C(x,y)\right)^2 = |C(x,y)|^2 \cdot \cos^2\delta$$

$$CR(x_c,y_c,r) = \frac{1}{2\pi}\int_{\alpha=0}^{2\pi} R(x_c + r\cos\alpha, y_c + r\sin\alpha, \alpha)\, d\alpha$$

Esempio: Macchina per diagnosi





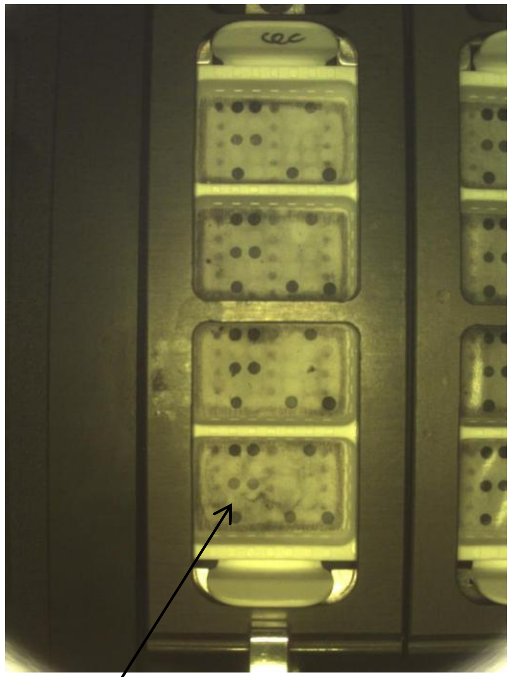


Immagine ottimale

Immagine rumorosa

Additional topic: cloud of points

Environment and objects can be described with other means.
Laser scanner: the laser derives the distance, with respect to the emitter, of a reflective obstacle.
By this mean representations of objects, or environment, can be obtained by clouds of points.

Several methods can be applied to extract edges from clouds of point: here 2 are illustrated that are used to obtain image registration.

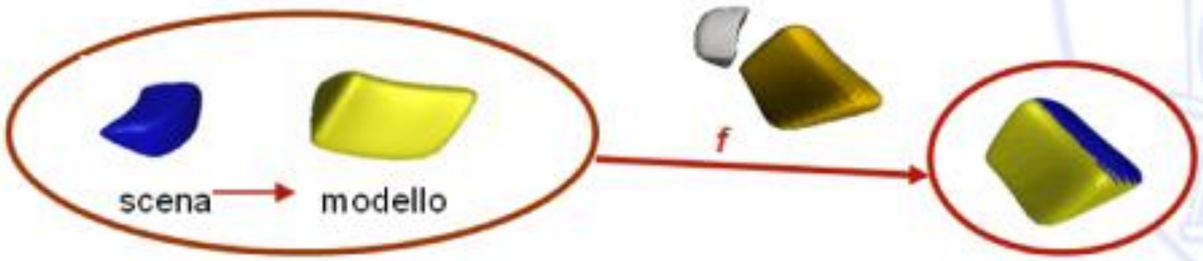
Image registration

Image registration aims to:

- overlaps two or more views of the same object

Optimisation

By reducing the size of the elements that should be aligned the procedure is optimised.



For a given image $I_s \in R^3$ called scene, made of $I_s = \{p_1, p_2, \ldots, p_s\}$, and the image $I_m \in R^3$ called model, made of $I_m = \{p_1, p_2, \ldots, p_m\}$, and such that their intersection is not null, the aim of the registration is to build a third image made of both scene and model, properly aligned.
It is similar to compose a puzzle

Since the clouds of points can be made of several million of points, it is required to extract the key feature:

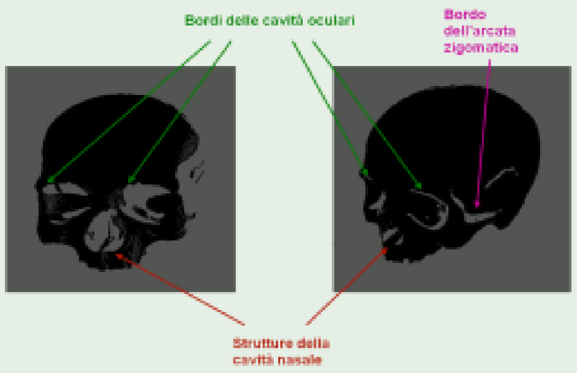
$$
\begin{aligned}
I'_s &= \{p_1, p_2, \ldots, p'_s\} \\
I'_m &= \{p_1, p_2, \ldots, p'_m\}
\end{aligned}
$$

with $s' < s$ and $m' < m$ such as $I'_s \cap I'_m \neq \emptyset$.

A forensic example

Example

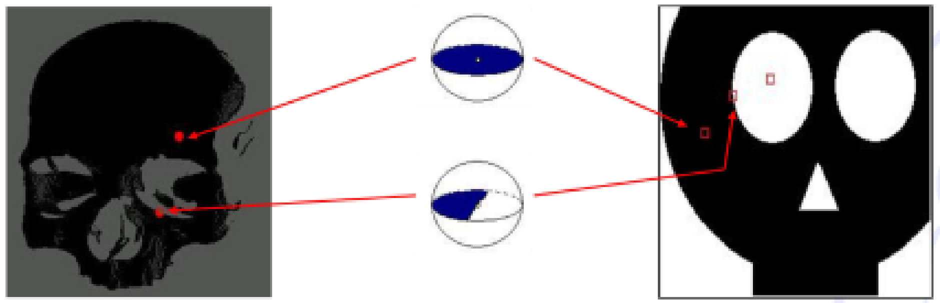
Anatomical key features:


Bordi delle cavità oculari
Bordo dell'arcata zigomatica
Strutture della cavità nasale

- Density bubbles
- Smoothing extraction

Density bubble

2D analogy



a point $q = (q_x, q_y, q_z)$ belongs to the bubble $\mathbf{B}$ of radius r centred in p, $q \in \mathbf{B}_p$, if:

$$\sqrt{(p_x - q_x)^2 + (p_x - q_x)^2 + (p_x - q_x)^2} \leq r$$

The number of points that belong to each bubble define the *bubble density*.

$$d_i = \sum_j q_j, \forall q_j \in \mathbf{B}_{p_i}$$

By mean of an heuristic selection a cut-off density is selected:

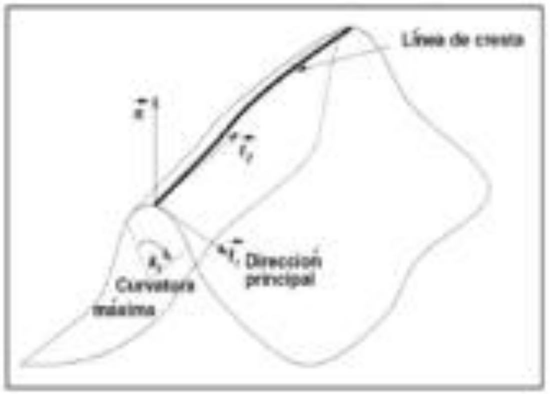
$$\alpha \cdot d^\gamma \leq d_t$$

with α e γ heuristic parameters.

Example

Interesting features are:

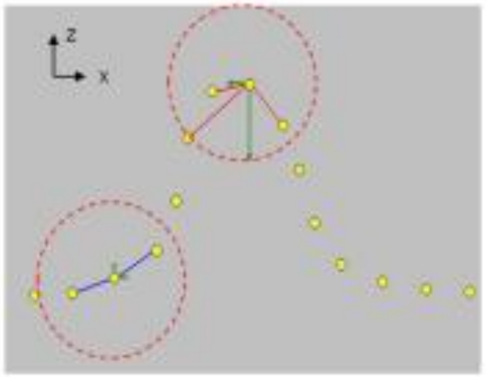
- Edges (already extracted with the bubbles)
- Valleys and ridges
- Corners

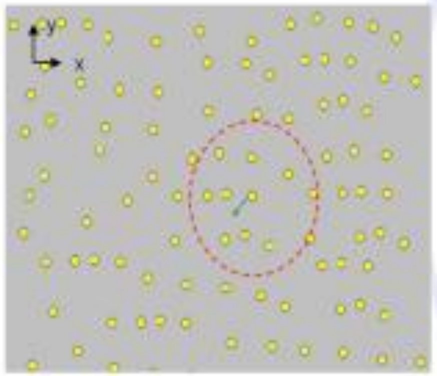


Attraction criterion

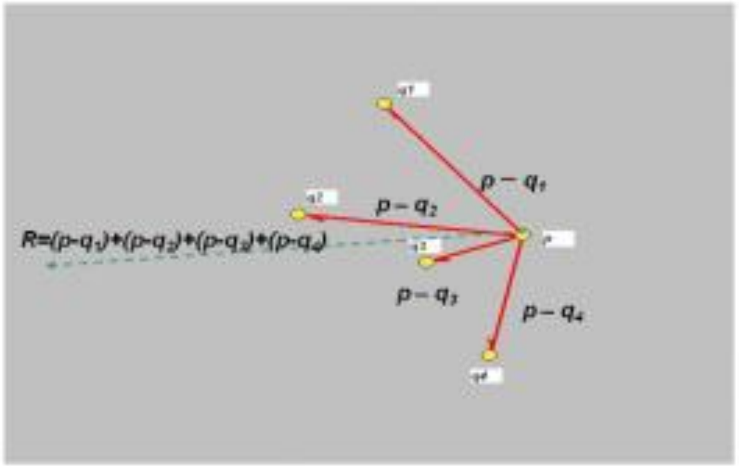
Based on attraction between close points and points displacement

The points $q_i \in \mathbf{B}$ attract p.





The resulting displacement is defined by:

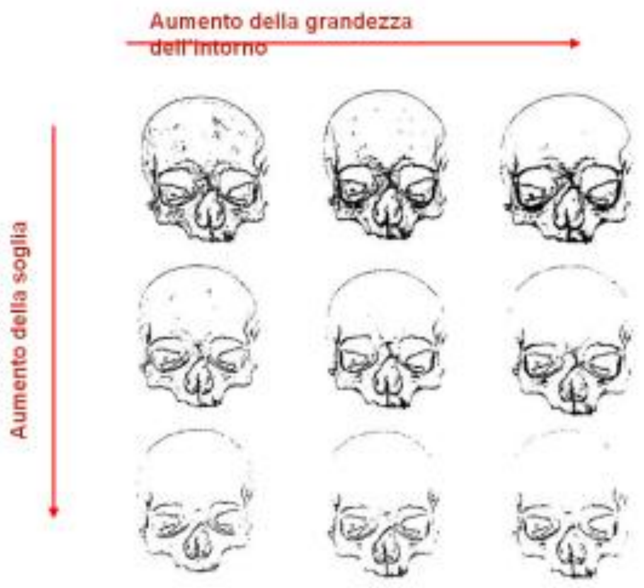


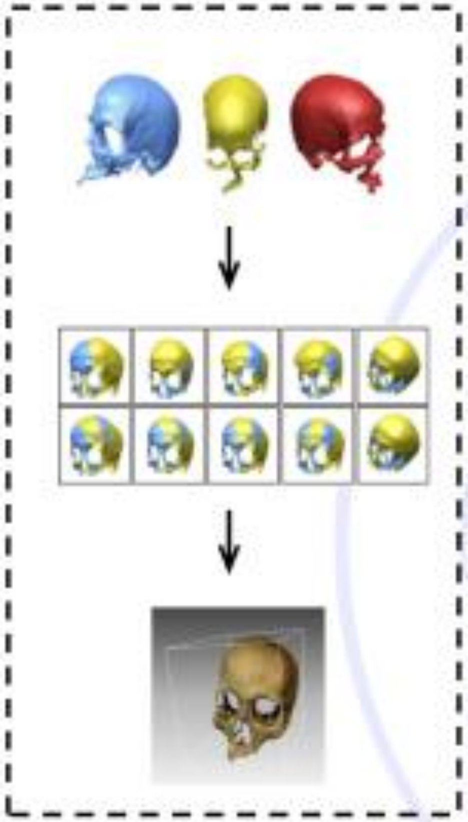
and the final position of the point is:

$$(p'_x, p'_y, p'_z) = \sum_{i \in \mathbf{B}} (\beta(p_x - q_{x_i}), \beta(p_y - q_{y_i}), \beta(p_z - q_{z_i}))$$

The separation criterion is based on the displacement magnitude:

$$s_i = \sqrt{(p'_x - p_x)^2 + (p'_y - p_y)^2 + (p'_z - p_z)^2}$$





References:

1. Digital Image Processing, Gonzalez-Wood, 3° edition, Prentice Hall, 2008
2. Robotics, Vision and Control, Corke, Springer, 2011

1) Nobuyuki Otsu, *A threshold selection method from gray-level histograms* in *IEEE Trans. Sys., Man., Cyber.*, vol. 9, 1979, pp. 62–66
2) Chen, L. *An investigation on the accuracy of three-dimensional space reconstruction using the direct linear transformation technique* in *Journal of Biomechanics*, **1994**, *27*, 493-500
3) Marziliano, P.; Dufaux, F.; Winkler, S. & Ebrahimi, T. A No-reference perceptual blur metric *IEEE International Conference on Image Processing*, **2002**, *3*, III/57-III/60
4) Srinivasan, M. V.; Zhang, S. W.; Lehrer, M. & Collett, T. S. Honeybee navigation en route to the goal: Visual flight control and odometry *Journal of Experimental Biology*, **1996**, *199*, 237-244
5) Ballerini, L.; Calisti, M.; Cordón, O.; Damas, S. & Santamaría, J. Automatic feature extraction from 3D range images of skulls **2008**, *5158 LNCS*, 58-69
6) Oliver Birbach, Udo Frese and Berthold Bauml; Realtime Perception for Catching a Flying Ball with a Mobile Humanoid; 2011 IEEE ICRA Shanghai International Conference Center May 9-13, 2011, Shanghai, China