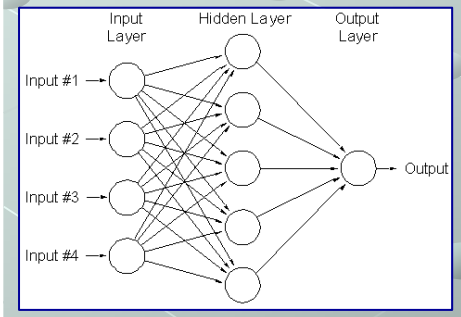
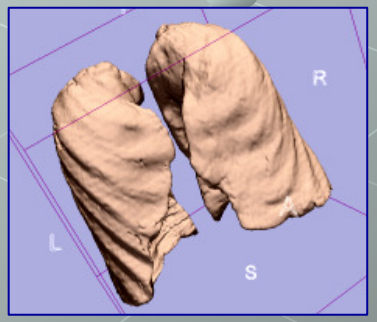
TECNICHE DI DIAGNOSTICA MEDICA

CdL in FISICA – aa 2010/2011





Giorgio De Nunzio

Dipartim. di Scienza dei Materiali – Univ. del Salento
e
Istituto Nazionale di Fisica Nucleare – Sez. di Lecce

`www.dsm.unisalento.it/WEBUtenti/giorgio.denunzio`
`giorgio.denunzio@unisalento.it`

Introduzione

Il **trattamento delle immagini** (Image Processing), coniugato a tecniche di **riconoscimento automatico** (Pattern Recognition, PR), è una disciplina che, negli ultimi lustri, ha assunto via via maggiore importanza nei campi più svariati (monitoraggio ambientale, diagnostica medica, controllo di alimenti, riconoscimento di persone e oggetti…).

In particolare, in ambito clinico, queste metodologie consentono di realizzare **sistemi software in grado di riconoscere in maniera automatica o semiautomatica la presenza di patologie** in immagini diagnostiche, e possono quindi essere di ausilio per il medico nella usuale pratica clinica o in caso di screening.

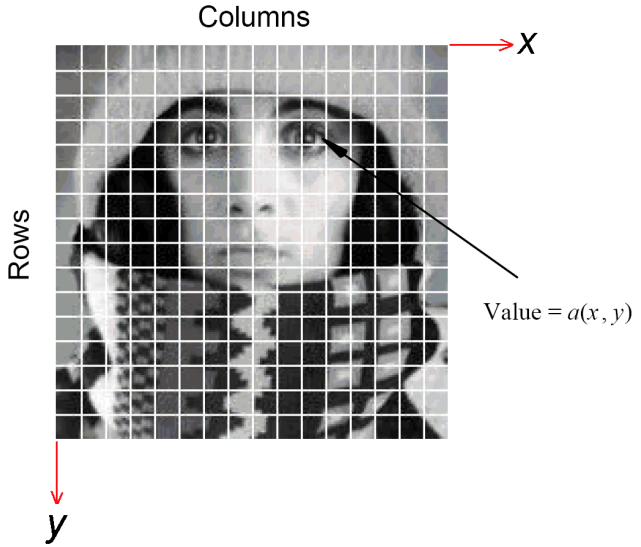
Si tratta, spesso, di sistemi basati su **modelli fisico-matematici di complessità non banale**, adattati a lavorare in ambito medico.

Dopo un'introduzione relativa al trattamento delle immagini in generale, e di quelle diagnostiche in particolare, le lezioni del Corso trattano delle **applicazioni delle tecniche di Image Processing e Pattern Recognition alla diagnostica medica**; esse descrivono in dettaglio il funzionamento di un sistema software di Computer Assisted Detection (CAD) operante su immagini: ne danno lo schema generico (pre-processing, ricerca di tessuti candidati ad essere patologici, calcolo delle feature discriminanti, classificazione), discutono dei vari tipi di *feature* utilizzabili, e accennano ai metodi di classificazione.

Immagini analogiche e digitali (1)

- Un'immagine definita nel "mondo reale" (**immagine analogica**) è idealmente paragonabile ad una **funzione di due variabili**, $a(x,y)$ in cui a è l'ampiezza (e.g. la luminosità) dell'immagine alla posizione di coordinate reali (x,y).
- L'ampiezza, in generale, avrà valori reali o interi: il secondo caso è il risultato di un processo di quantizzazione (campionamento) che converte un intervallo continuo di valori (per esempio, tra 0 e il 100% della luminosità) in un numero finito di livelli (in realtà, la possibilità che l'intervallo di valori sia continuo è un'astrazione e un'approssimazione).
- Un'**immagine digitale** a[m,n] definita in uno spazio bidimensionale discreto è **derivata** da un'immagine analogica $a(x,y)$ definita in uno spazio continuo attraverso un processo di **campionamento** (*digitalizzazione*).

Digitalizzazione di un'immagine continua.



Immagini analogiche e digitali (2)

- Nel **processo di digitalizzazione** l'immagine $a(x,y)$, definita nel continuo, è divisa in N righe e M colonne. L'intersezione di una riga ed una colonna è denominata *pixel*. Il valore assegnato alle coordinate intere (m,n) con $m = 0,1,2,\ldots,M-1$ (oppure $m = 1, 2, \ldots, M$) e $n = 0,1,2,\ldots,N-1$ (oppure è $n = 1,2,\ldots,N$) è $a(m,n)$. In molti casi, $a(x,y)$ — che possiamo considerare un segnale fisico rivelato da un sensore — è una funzione di molte variabili tra cui ad esempio il tempo (t).

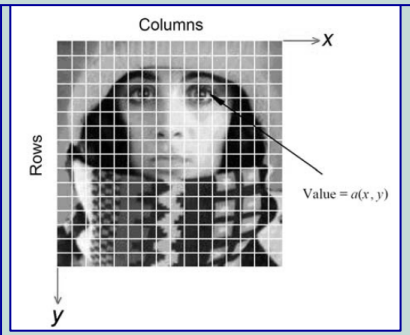
- Nell'esempio nella slide precedente l'immagine è stata divisa in N = 16 righe e M = 16 colonne. Il valore assegnato ad ogni pixel è la luminosità media nel pixel arrotondata all'intero più vicino. L'ampiezza del segnale 2D in una data posizione è dunque rappresentata con un valore intero, scelto tra L livelli di grigio diversi.

- I parametri: numero di colonne (larghezza), di righe (altezza) e di livelli di grigio (profondità) assumono spesso dei valori standard per le immagini digitali. **Larghezza ed altezza** spesso sono della forma 2^K, con $K \in \{8, 9, 10\}$; ciò è motivato dalla struttura dei circuiti digitali del computer, o dall'uso di certi algoritmi come la *Fast Fourier Transform* che richiedono preferenzialmente che le dimensioni dell'immagine su cui lavorano siano potenze di 2. Il **numero di livelli di grigio distinti** è di norma una potenza di 2, ovvero, $L = 2^B$ dove B è il numero di bit nella rappresentazione binaria dei livelli di luminosità. Valori comunemente incontrati per la profondità dell'immagine (a toni di grigio) sono: 2, 64, 256, 65536. Quando B>1 si parla di un'immagine a livelli di grigio; quando B=1 l'immagine è *binaria*. In un'immagine binaria ci sono solo due livelli di grigio ai quali ci si riferisce come "nero" e "bianco" o "0" e "1" (non necessariamente in quest'ordine).

Acquisizione ed elaborazione d'immagini (diagnostiche)

Campionamento spaziale dell'immagine. Il num. di bit di codifica è funzione della precisione di misura richiesta:

- 1 % ≥ 7 bit (2^7=128)
- 0.1 % ≥ 10 bit (2^{10}=1024)



INPUT (sensori)

↓

CONVERSIONE A/D

↓

ELABORAZIONE

↓ ↓

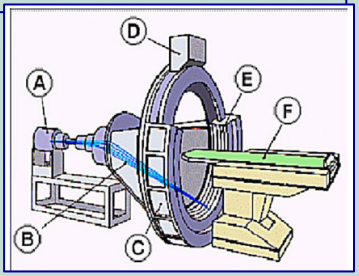
CONVERSIONE D/A **COMPRENSIONE / MISURA**

Visualizzazione, stampa

↓ ↓

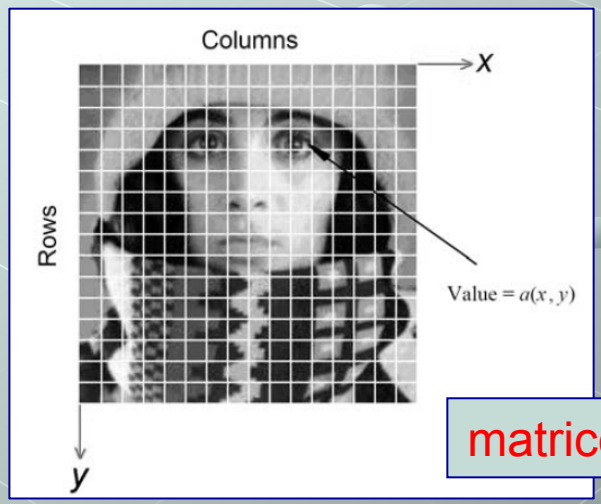
OUTPUT **OUTPUT**

Acquisizione di un segnale (assorbimento X, echi US…) e conversione in un segnale elettrico



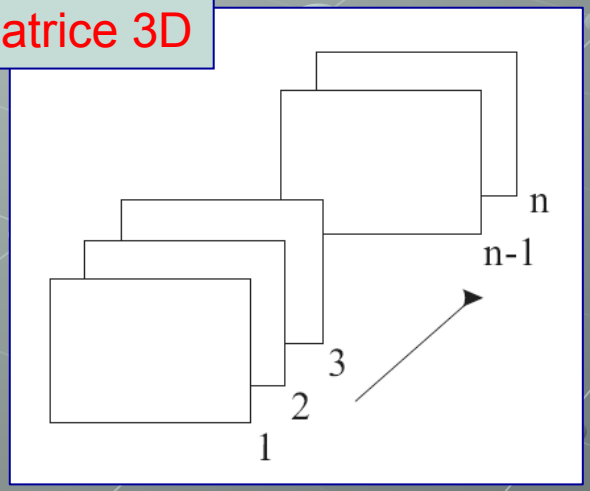
Immagini bi- e tri-dimensionali (1)

- L'immagine digitale bidimensionale (2D) è un segnale costituito da campioni quadrati (*pixel, picture element*) uniformemente organizzati su una griglia 2D rettangolare (una matrice: mosaico!). Un'immagine tridimensionale è basata su una griglia tridimensionale (matrice 3D) di punti (voxel) e rappresenta un volume.
- I pixel (voxel) non sono necessariamente rettangolari o cubici (anisotropia o anisometria)
- Il numero di colonne e di righe della matrice costituisce le dimensioni (impropriamente, la risoluzione) dell'immagine,
- Nel caso più comune in ambito radiologico (in cui al pixel è associata una misurazione di una grandezza fisica), si adoperano immagini monocromatiche, ovvero rappresentabili visivamente associando il valore dei pixel ad una scala di grigi.



matrice 2D

matrice 3D



Immagini bi- e tri-dimensionali (2)

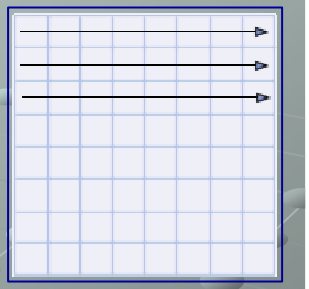
- Per le immagini monocromatiche, a ciascun *pixel* è associato un valore di intensità, uniformemente quantizzato all'interno di *range* finiti, e rappresentabile in formato binario attraverso un certo numero di bit per pixel, *bpp*.
- Per esigenze di rappresentazione e di gestione del dato, a seconda delle modalità di acquisizione vi potranno essere dei sottointervalli (es.12 bit/pixel) effettivamente occupati dal segnale utile, "ospitato" in parole binarie multiple del byte (es. 2byte=16bit). Il resto dei bit è usualmente nullo.
- Compressione lossy e lossless. Per ragioni di spazio sulla memoria di massa e di velocità di trasferimento attraverso reti telematiche, i file contenenti le immagini sono generalmente compressi con metodologie lossy (con perdita) o lossless (senza perdita). I metodi lossless, sebbene meno efficaci, sono naturalmente più adatti per le immagini diagnostiche mediche.

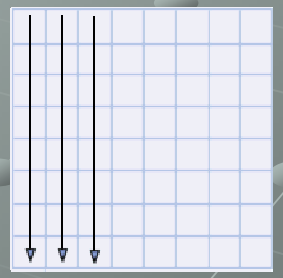
Compressione lossy e lossless (1)

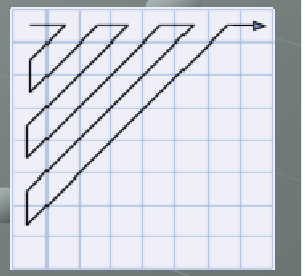
Gli algoritmi di compressione possono essere lossless o lossy. Mentre la compressione lossless consente di ricostruire l'informazione originale contenuta nel file, la compressione lossy modifica l'informazione in modo irreversibile a vantaggio di un maggior rapporto di compressione.

Compressione lossless tipo Run Length Encoding: è usata per numerosi formati di immagini (BMP, PCX, TIFF). E' basato sulla ripetizione di elementi consecutivi. Il principio di base consiste nel codificare un primo elemento che dà il numero di ripetizioni di un valore, poi completarlo con il valore da ripetere. Per esempio la stringa "AAAAAHHHHHHHHHHHHHH" compressa dà "5A14H". Il guadagno di compressione è di (19-5)/19 ossia circa 73,7%. Invece per la stringa "REELLEMENT", nella quale la ripetizione dei caratteri è ridotta, il risultato della compressione dà "1R2E2L1E1M1E1N1T"; la compressione si avvera molto costosa, con una perdita che vale (10-16)/10 ossia 60%! La compressione RLE è definita da regole che permettono di comprimere quando possibile e di lasciare la stringa originale quando la compressione produce uno spreco.

La compressione RLE ha senso solo per i dati con un numero di elementi consecutivi ripetitivi, soprattutto le immagini che hanno delle ampie parti uniformi. Questo metodo ha il vantaggio di essere piuttosto semplice da utilizzare. Esistono delle varianti in cui l'immagine è decodificata per tasselli di punti, per linee, oppure anche a zigzag.





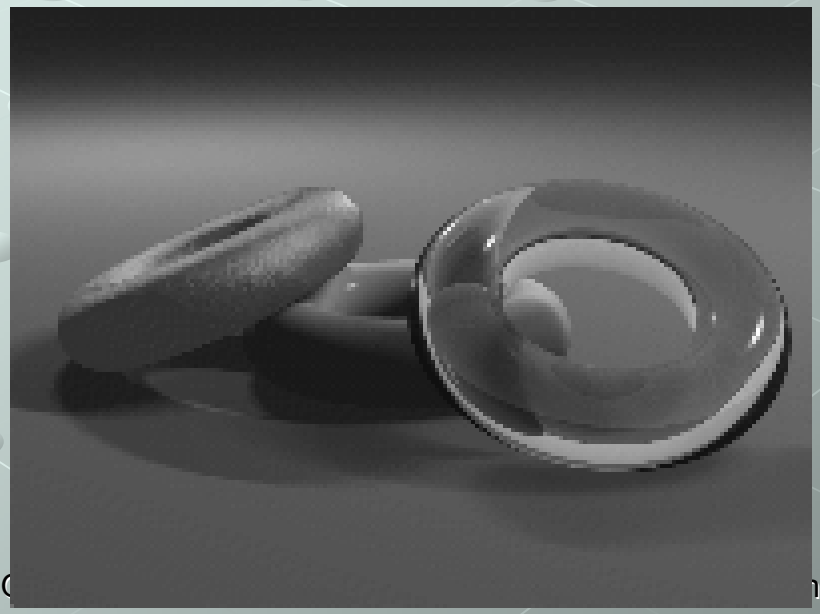


Compressione lossy e lossless. (2)

Compressione lossy

Per comprimere dati come il suono o le immagini, dove una perdita di qualità potrebbe non essere notata viene usata la compressione lossy . Gli algoritmi di compressione lossy sacrificano parte dei dettagli in favore di un maggiore rapporto di compressione. L'immagine ricostruita decomprimendo il file inganna l'occhio, ma contiene notevoli differenze. Solitamente tali differenze non risultano percettibili, in quanto la parte di informazione persa è comunque quella che l'utente non avrebbe notato.

Si è stabilito che l'occhio non è in grado di distinguere due immagini in bianco e nero che abbiano, la prima profondità 6 (64 grigi) e la seconda profondità 8 (256 grigi).



Compressione lossy e lossless. (3)

Compressione lossy

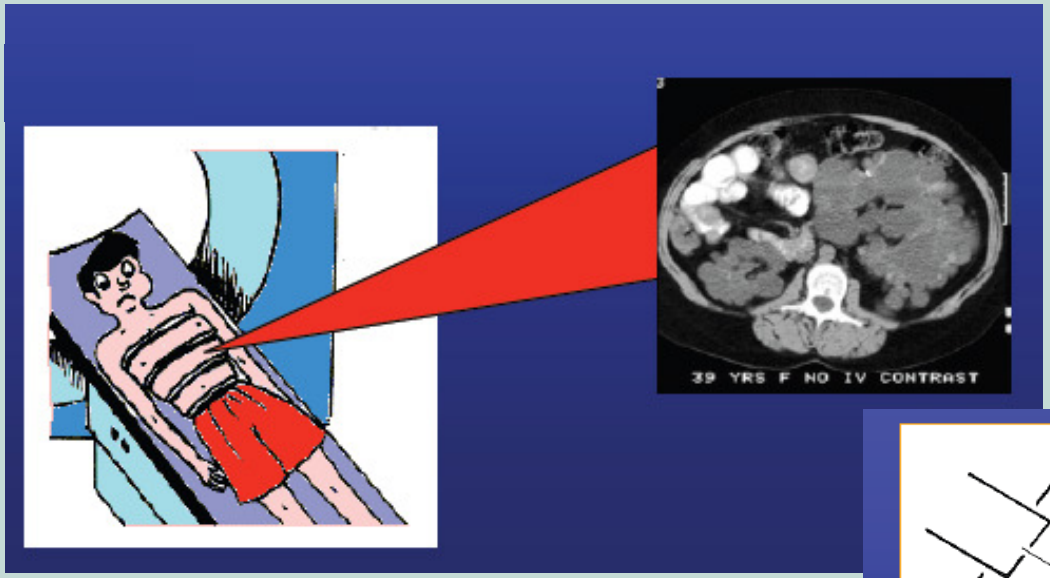
Un altro tipo di differenza non riscontrabile riguarda la luminosità e il colore di un pixel, è stato dimostrato che l'occhio umano è molto più sensibile a variazioni di luminosità che di tinta, per cui è possibile eliminare alcune sfumature da un'immagine senza che l'occhio lo rilevi.

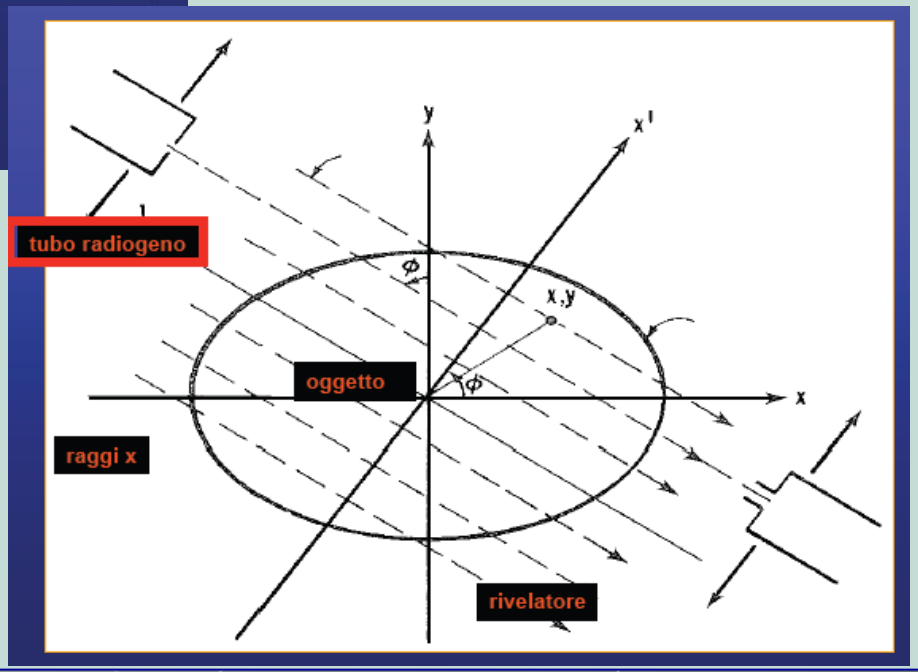
Altre caratteristiche dell'immagine possono essere eliminate, per cui la diminuzione dello spazio occupato deriva dall'eliminazione di dettagli ininfluenti, e dalla successiva compressione.

JPEG il primo vero standard che ha introdotto una codifica "lossy" delle immagini.

Per le immagini di diagnostica medica, è comunque preferibile una codifica lossless!

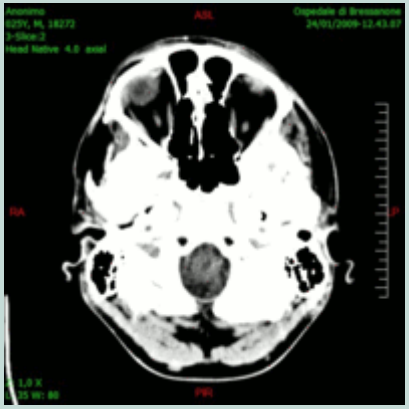
Il caso della tomografia computerizzata (CT) (1)

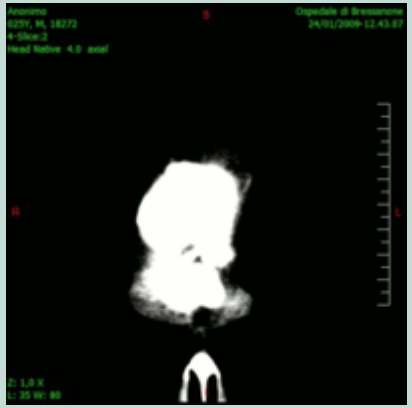




Il caso della tomografia computerizzata (CT) (2)

Il caso della tomografia computerizzata (CT):





- **Ciascun pixel rappresenta l'assorbimento dovuto ad un piccolo volume (voxel) del corpo umano**
- **Questo valore viene misurato secondo la scala denominata HU (Hounsfield Units), in base alla relazione:**

$$\mu(HU) = \frac{\mu - \mu_{H2O}}{\mu_{H2O}} \times 1000$$

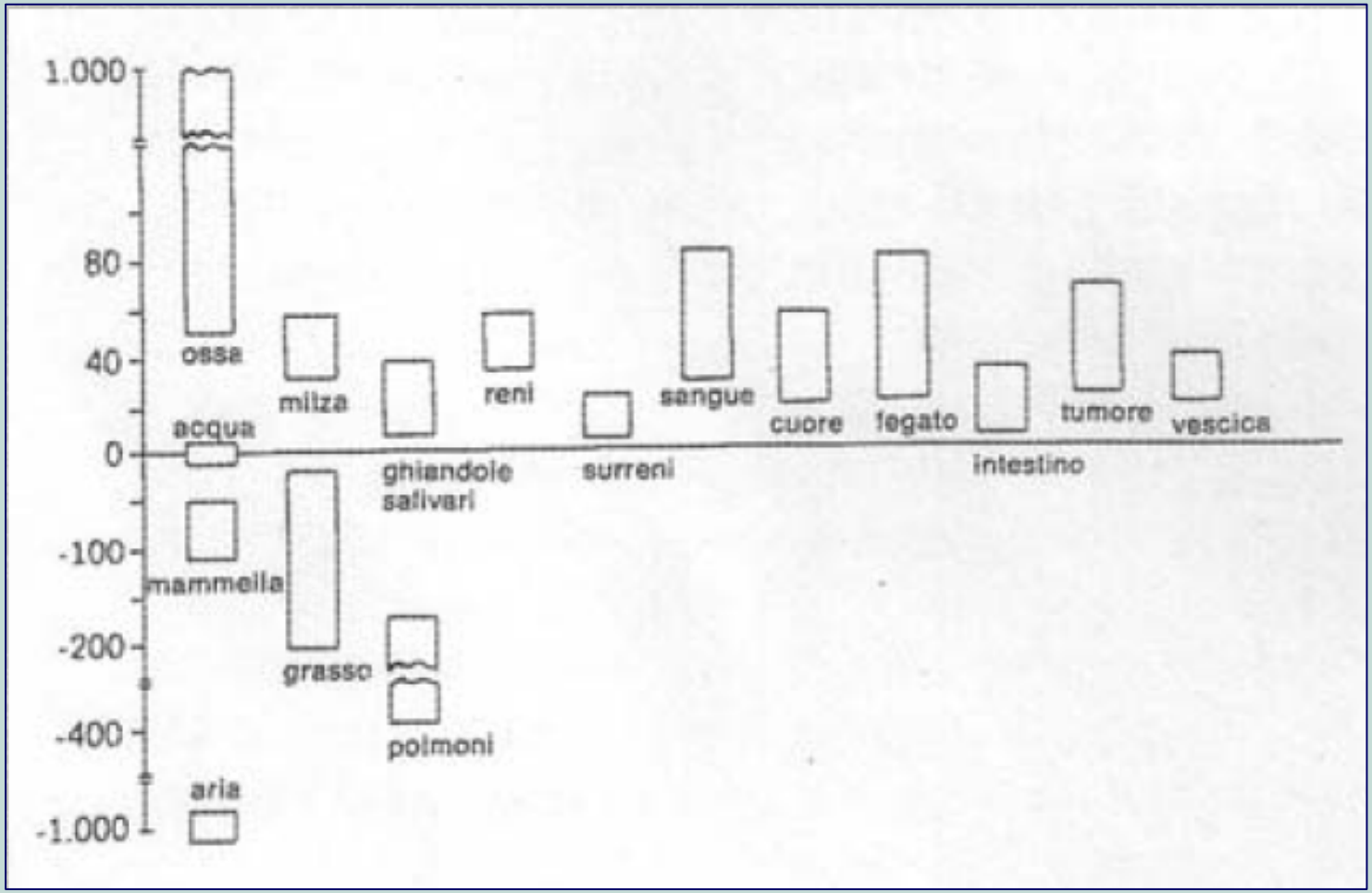
Quindi:

μ_{vuoto} = -1000 HU

μ_{H2O} = 0 HU

Il caso della tomografia computerizzata (CT) (3)

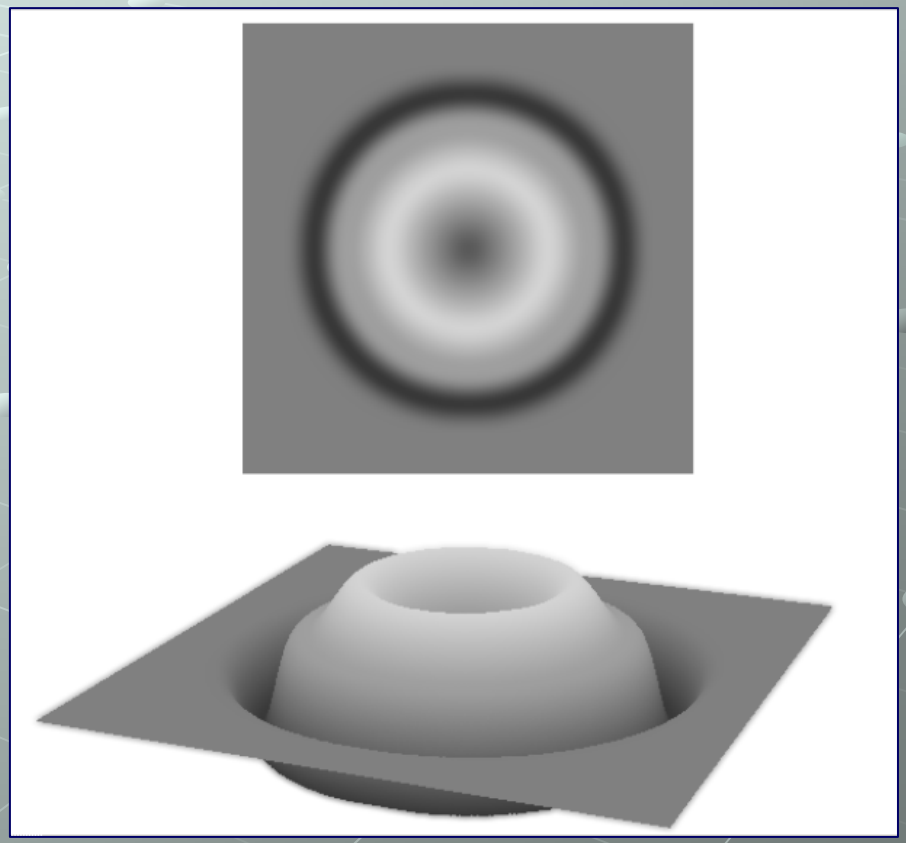
La scala di Hounsfield:



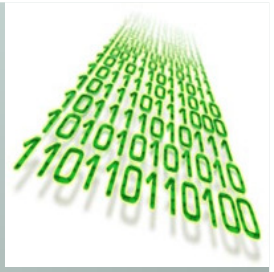
Immagini monocromatiche come superfici

I livelli di grigio a(x,y) di un'immagine possono essere considerati altezze, ossia coordinate $z = a(x,y)$, degli elementi di una superficie:

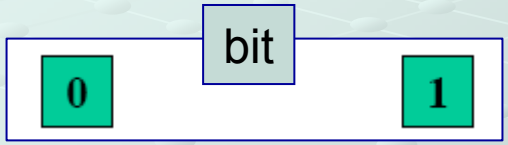
Conseguenza: uso di algoritmi di geometria differenziale per individuare dettagli nelle immagini



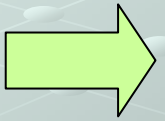
Bit e byte



- In un sistema digitale le informazioni sono ***rappresentate, mediante*** **LIVELLI DISCRETI** di una grandezza fisica.



cifra che può assumere solo due valori, 0 / 1



8 dispositivi elementari a 2 stati

0	1	1	0	1	1	0	0

byte

1 dispositivo a 256 stati

l'**unità pratica principale** in informatica

multipli del byte:

2^{10} byte = 1024 byte = Kilobyte = KB
2^{20} byte = 1024 Kbyte = Megabyte = MB
2^{30} byte = 1024 Mbyte = Gigabyte = GB
2^{40} byte = 1024 Gbyte = Terabyte = TB

Attenzione: 1 MB ≠ 1000000 byte!

Tipi di dati e valori rappresentabili

- Byte/caratteri: *char - signed char - unsigned char*
  - 8 bit, 2^8 = 256, 0..255 oppure -128..127
- interi *short – int – long, signed* o *unsigned*
  - **short**: 16 bit , 2^{16} = 65536, 0.. 65535 oppure -32768..32767 (ma dipende dalla macchina!)
  - **int**: 32 bit, 2^{32}
  - **long**: 32/64 bit
- reali *float - double - long double*
  - **float** (4 byte) rappresenta reali con segno da $1.2*10^{-38}$ a $3.4*10^{38}$ con 6 cifre significative
  - **double** (8 byte) rappresenta reali con segno da $2.2*10^{-308}$ a $1.8*10^{308}$ con 15 cifre significative
  - **long double** (10 byte o 16 byte o…)

Immagini e tipi di dato

- IMMAGINI A COLORI: a colori reali o a pseudocolori.
  - Ogni pixel è rappresentato da una terna di valori che misurano le componenti R, G e B rispettivamente. Di norma ciascuna componente è memorizzata in un byte, per cui l'occupazione di memoria è di **tre** byte per pixel (quindi 24 bit per pixel).
    Esistono anche le Immagini "indicizzate"…
- IMMAGINI MONOCROMATICHE: a toni di grigio
  - I pixel sono rappresentati da uno o due byte, a seconda della sensibilità (precisione di misura) dell'apparecchiatura diagnostica. Per esempio le immagini CT sono normalmente 2 byte / pixel (ricordare i numeri di Hounsfield!).
  - Normalmente (caso **8 bit**): **0 = nero, 255 = bianco**; valori intermedi rappresentano gradazioni intermedie di grigio.
  - Spesso il range numerico disponibile per ogni pixel non è completamente sfruttato: allora solo una parte frazionaria dei byte allocati è utilizzata:
    - 10 bit: 2^{10} = 1024 toni di grigio
    - 12 bit: 2^{12} = 4096 toni di grigio
    - 16 bit: 2^{16} = 65536 toni di grigio
- IMMAGINI BIANCO/NERO: usate per le **maschere**: un pixel a 0 indica mascheratura del pixel di un'altra immagine, a 1 (o 255, o comunque diverso da 0) indica conservazione di quel pixel. Normalmente sono immagini a 1 byte/pixel (potrebbero essere a 1 bit/pixel).

Quanta memoria occupa un'immagine?

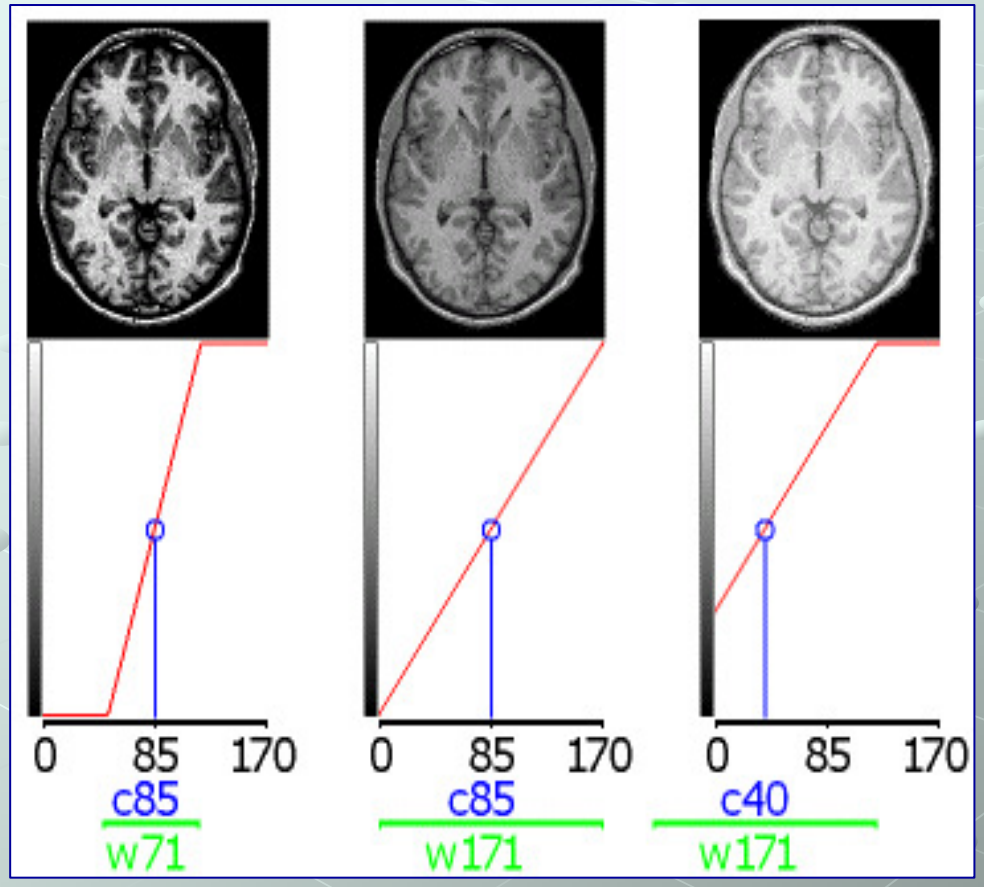
- Caso di un'immagine monocromatica a 16 bit/pixel (ossia 2 byte/pixel).

Se la risoluzione dell'immagine è 512 * 512 (ossia se l'immagine è rappresentata da una matrice di 512 colonne e di 512 righe) allora l'occupazione di memoria è:

$$M = 512 * 512 * 2 = 524288 \text{ byte}$$

L'occupazione di spazio su memoria di massa (disco) potrà essere uguale, circa uguale (un header!) oppure inferiore se l'immagine è compressa.

Finestra dei grigi: centro e larghezza



Quando si visualizza con un particolare dispositivo (e.g. un monitor di pc) un'immagine di diagnostica medica in scala di grigi, è usuale adoperare i termini di '**centro della finestra' (C) e 'ampiezza della finestra' (W)**, dove il termine 'finestra' si riferisce al *range* di valori di grigio dell'immagine che verranno mappati nell'intervallo di valori riproducibile dal dispositivo di visualizzazione.

Si tratta di un modo di indicare la **luminosità e il contrasto** dell'immagine.

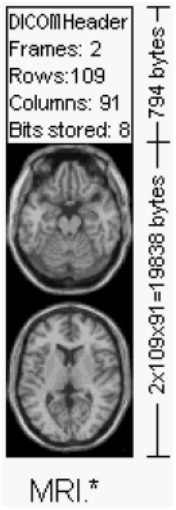
I sistemi di generazione di immagini diagnostiche di tipo **radiografico** sono tarati in maniera da produrre intensità di grigio calibrate (**approssimativamente**!!) in funzione della **densità** del materiale esaminato (osso, muscolo, grasso…), in modo che si possa usare una certa coppia di valori specifica C:W per visualizzare con ricchezza di dettagli determinati tipi di tessuto. Per esempio in una radiografia la coppia C:W pari a 400:2000 è in generale adatta a visualizzare le ossa, mentre la coppia 50:350 è adatta ai tessuti molli.

Formati dei file di immagine diagnostica (1)

- Standard DICOM ("*Digital Imaging and Communications in Medicine*") [ftp://medical.nema.org/medical/dicom/] creato dal *National Electrical Manufacturers Association* (NEMA) allo scopo di favorire la distribuzione e la visualizzazione di immagini di diagnostica medica.
- La "**Parte 10**" dello standard DICOM descrive un formato di file per la distribuzione delle immagini (estensione dello standard NEMA). File di immagini compatibili con la Parte 10 dello standard vengono denominati "immagini DICOM" o "file DICOM".
- Un file DICOM contiene un'intestazione o *header* (che immagazzina le informazioni relative al nome e al sesso del paziente, al tipo di scansione, alle dimensioni dell'immagine e così via; queste informazioni sono indicate globalmente con il termine *metadati*) e i dati dei pixel dell'immagine. La presenza di entrambe le componenti (*metadati* e dati grafici) in un singolo file differenzia il formato DICOM da un altro formato piuttosto diffuso: Analyze; quest'ultimo conserva *metadati* e dati grafici dei pixel in due file diversi aventi **uguale nome** ma **estensione diversa**, rispettivamente **.hdr** e **.img**. La versione più "moderna" del formato Analyze, è il formato **NIFTI** (unico file).
- Altra differenza tra DICOM e Analyze è la possibilità, in DICOM, di inserire immagini compresse. I file possono essere compressi utilizzando sia compressioni *lossy* (con perdita di informazione) che *lossless* (senza perdita di informazione).
- DICOM è il formato più comune di immagazzinamento e trasmissione di immagini diagnostiche mediche.
- DICOMDIR: immagine DICOM solitamente 3D suddivisa su più file (uno per ciascuna slice), referenziati da un file indice (chiamato, appunto, `dicomdir`).

Formati dei file di immagine diagnostica (2)

- Alcuni *metadati* presenti nell'*header DICOM* sono direttamente legati alle caratteristiche dell'immagine (o delle immagini) memorizzate nel file: le dimensioni (larghezza e altezza in *pixel*), il numero di bit allocati per ciascun pixel e quanti di essi sono effettivamente utilizzati; altri dati sono invece relativi a parametri quali lo spessore di una fetta nel caso di una TAC, oppure al tipo di analisi diagnostica; vi sono poi vari dati sul paziente, come il nome, l'età, il sesso…
- La dimensione dell'*header* dipende da quante informazioni esso conserva, considerando che la maggior parte dei dati inseribili in un file DICOM sono facoltativi.




```
First 128 bytes: unused by DICOM format
Followed by the characters 'D','I','C','M'
This preamble is followed by extra information e.g.:

0002,0000,File Meta Elements Group Len: 132
0002,0001,File Meta Info Version: 256
0002,0010,Transfer Syntax UID: 1.2.840.10008.1.2.1.
0008,0000,Identifying Group Length: 152
0008,0060,Modality: MR
0008,0070,Manufacturer: MRIcro
0018,0000,Acquisition Group Length: 28
0018,0050,Slice Thickness: 2.00
0018,1020,Software Version: 46\64\37
0028,0000,Image Presentation Group Length: 148
0028,0002,Samples Per Pixel: 1
0028,0004,Photometric Interpretation: MONOCHROME2.
0028,0008,Number of Frames: 2
0028,0010,Rows: 109
0028,0011,Columns: 91
0028,0030,Pixel Spacing: 2.00\2.00
0028,0100,Bits Allocated: 8
0028,0101,Bits Stored: 8
0028,0102,High Bit: 7
0028,0103,Pixel Representation: 0
0028,1052,Rescale Intercept: 0.00
0028,1053,Rescale Slope: 0.00392157
7FE0,0000,Pixel Data Group Length: 19850
7FE0,0010,Pixel Data: 19838
```

dimensioni: larghezza 91 colonne,
altezza:109 righe,
2 fotogrammi,

In breve (109 x 91 *voxels)* x 2

risoluzione: 8 bit (un byte) per voxel (tutti effettivamente utilizzati),

La memoria occupata dall'immagine è 109 x 91 x 2 = 19838 byte.

I dati dell'immagine seguono le informazioni dell'*header*.

Operazioni sulle immagini: teoria ed esercitazioni pratiche

- Il trattamento di un'immagine può essere classificato in tre diverse categorie:
  - *Image Processing* (elaborazione d'immagine) quando il risultato della manipolazione è un'altra immagine;
  - *Image Analysis* quando il risultato sono delle misure effettuate sull'immagine;
  - *Image Understanding* se il risultato è una descrizione di alto livello.
- Nelle prossime slide si parlerà soprattutto delle basi, e cioè di ***processing***, giungendo poi ad eseguire alcune esercitazioni di ***analysis*** che daranno risultati quantitativi, con lo scopo ideale di coadiuvare il medico nel suo lavoro di esame delle immagini diagnostiche, e di refertazione.

LAVORI IN CORSO!

Nel prosieguo, questo simbolo indicherà quando sarà il momento di mettersi alla tastiera, in ambiente Matlab, e "smanettare" con le immagini.

Image Processing: operazioni puntuali/locali/globali

Le operazioni sulle immagini digitali possono essere classificate in tre categorie:

Operazione	Caratterizzazione
puntuale	il valore di uscita (risultato di un'operazione) in una specifica posizione (coppia di coordinate) è dipendente solo dal valore di ingresso per **quella stessa posizione** (esempio, operazione di soglia o *thresholding*).
locale	il valore di uscita in una specifica posizione è dipendente dai valori di ingresso valutati **nell'intorno** di quella posizione (esempio: media "locale"dei valori di grigio).
globale	i valori in uscita (in ciascun punto dell'immagine risultato) dipendono dai valori dei pixel **nell'intera immagine** (es.: trasformata di Fourier, istogramma, media dei grigi)

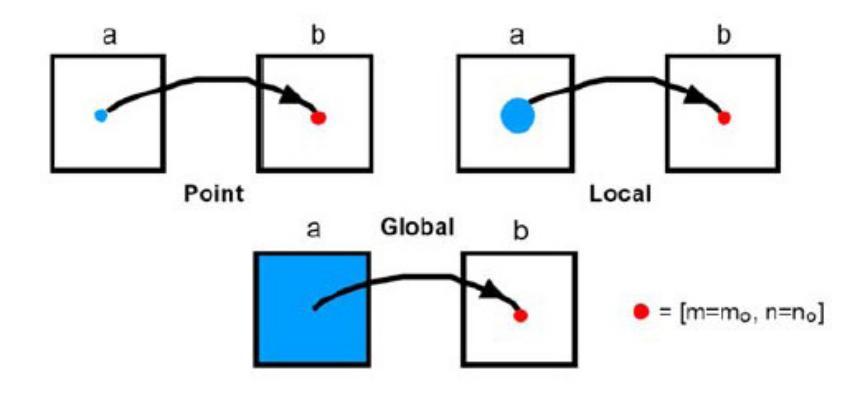


Image Processing: operazioni di base

- Alcune operazioni di processamento di immagini sono:
  - Variazioni di contrasto/luminosità/colore (operano sull'istogramma dei colori o dei grigi)
  - Operazioni che **attenuano o esaltano** i dettagli
  - Operazioni che **attenuano o esaltano** i contorni (bordi)
  - Operazioni che **attenuano o esaltano** strutture
  - Operazioni di soglia

Altre operazioni di image processing

Operazioni morfologiche (dilatazione, erosione, chiusura, apertura, scheletrizzazione, ricostruzione)

Operazioni logiche (and, or, xor, not)

Operazioni nel dominio delle frequenze (uso della trasformata di Fourier)

Ricerca di pattern (segmenti di rette, cerchi…) con la trasformata di Hough

…e tante, tante altre…

Istogramma di intensità (o dei grigi) (1)

- Diagramma che mostra la distribuzione dei valori di grigio presenti in un'immagine **monocromatica**

- Si costruisce dividendo l'asse che rappresenta i livelli di grigio (valori, e.g., tra 0 e 1, se **normalizzati**, o tra 0 e 255, se un pixel è rappresentato da un byte di 8 bit) in un certo numero di intervalli I_i, tutti di ugual larghezza.

- Esaminando i valori di grigio di ogni pixel nell'immagine, si conta, per ciascun intervallo I_i, il numero di pixel h_i il cui valore di grigio cade nell'intervallo stesso. Si grafica il risultato come diagramma a barre.

- Nell'istogramma d'intensità l'altezza di ogni colonna rappresenta il numero di pixel che hanno valore compreso nell'intervallo corrispondente.

- Il processo di costruzione dell'istogramma corrisponde dunque al riempimento delle "scatole" (*bin* in inglese) definite in ascissa.

Istogramma di intensità (o dei grigi) (2)

Costruzione di un istogramma a partire da un'immagine elementare 4x4 della quale sono mostrati i valori di grigio dei singoli pixel.

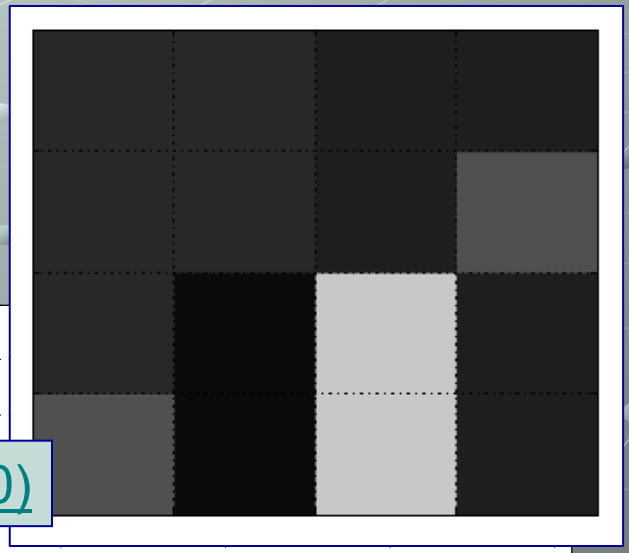
	1	2	3	4
1	40	40	30	30
2	40	40	30	80
3	40	10	200	30
4	80	10	200	30

Nella command window di Matlab o in un file..

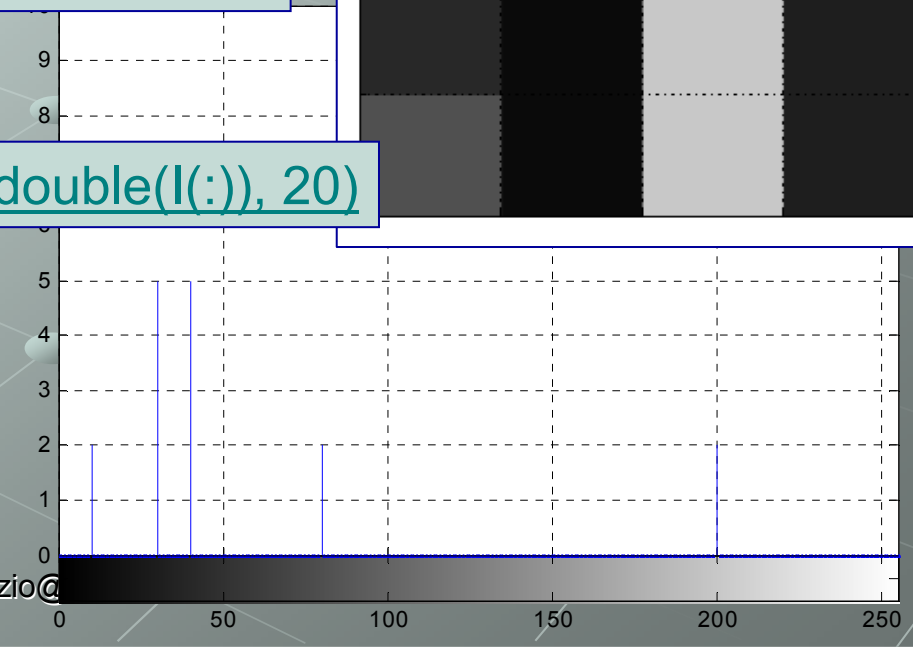
```
I = uint8([40, 40, 30, 30;
           40, 40, 30, 80;
           40, 10, 200, 30;
           80, 10, 200, 30])
figure, image(I), grid on
colormap(gray(256))
a = gca;
set(a,'YTick',[1.5:3.5])
set(a,'XTick',[1.5:3.5])
set(a,'YTickLabel',[])
set(a,'XTickLabel',[])

figure, imhist(I), grid on
set(gca, 'YLim',[0,10])
```

imshow(I, [ ])



hist(double(I(:)), 20)



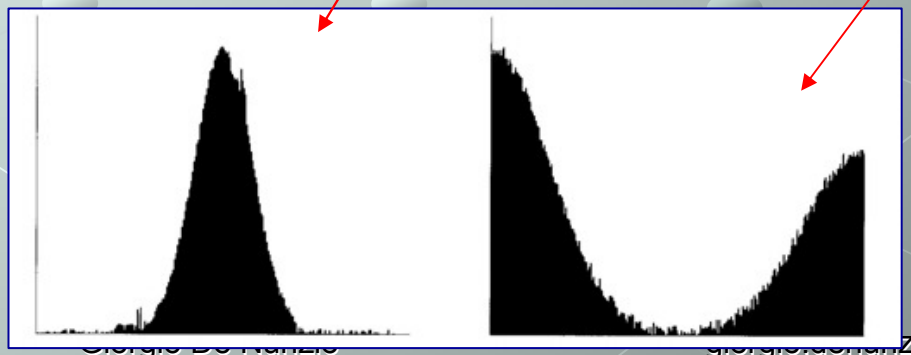
Istogramma di intensità (o dei grigi) (3)

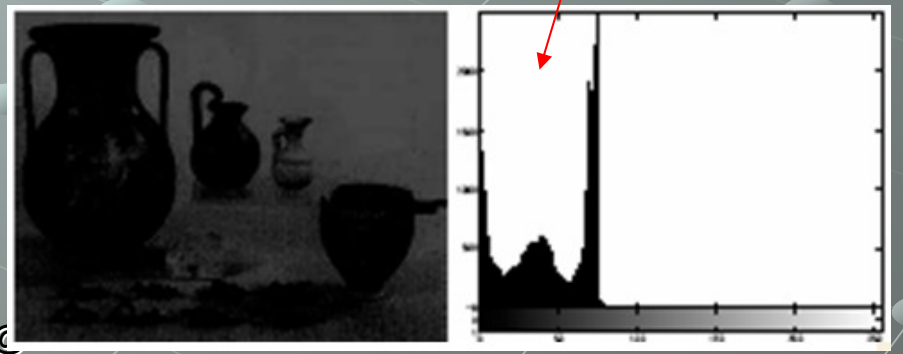
- L'istogramma fornisce informazioni sul profilo di intensità dell'immagine.

- Da esso si deducono grandezze quali ad esempio il contrasto e la luminosità. Immagini scure hanno istogrammi con distribuzioni di pixel verso la parte sinistra; nelle immagini chiare, al contrario, i pixel si ammassano nella parte destra dell'asse.

- In un'immagine ideale ci attendiamo una distribuzione di pixel tendenzialmente più uniforme.

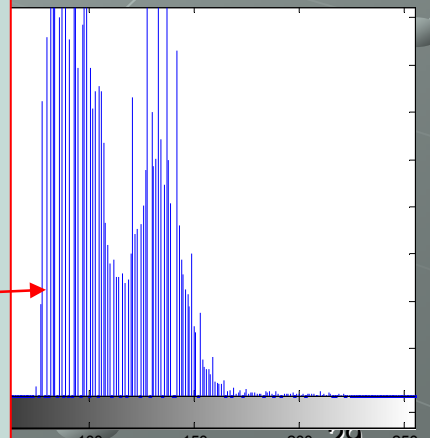
- In un'immagine poco contrastata le "scatole" più piene sono in una regione ristretta dell'asse, mentre in un'immagine molto contrastata ci sono zone dell'asse ben popolate e distanti tra loro





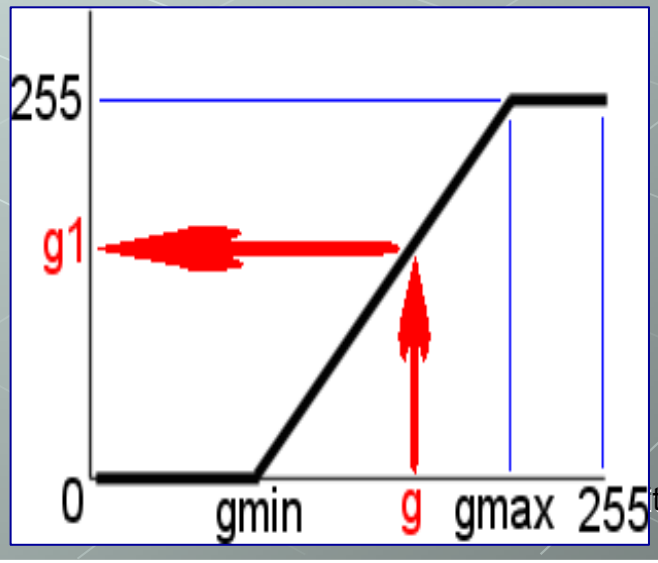
Istogramma di intensità (o dei grigi) (4)

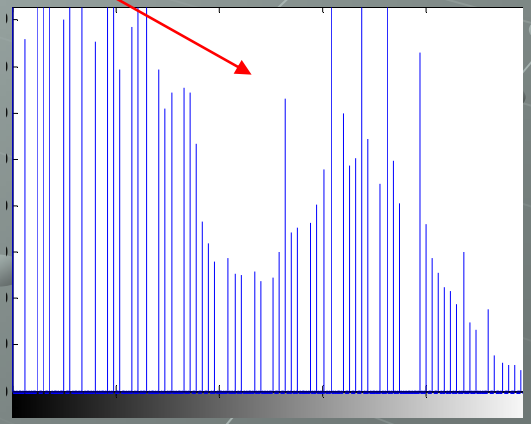
1. Andare nella `command window` di Matlab
2. Aprire un'immagine (inclusa nell'Image Processing Toolbox) sulla quale lavoreremo:
   `I = imread('pout.tif');`
   Il "`;`" ha lo scopo di evitare che MATLAB mostri a video il contenuto della matrice `I` ora definita (provare a non inserirlo). Formati leggibili: **`help imread`**
3. Per mostrare l'immagine ora letta vi sono diversi comandi: **`imshow`** e **`imtool`** dell'IPT, **`image`** e **`imagesc`** di Matlab; useremo **`imshow`**
   `imshow(I)`
4. Risultato: apertura di una finestra grafica contenente l'immagine
5. Guardare nel Workspace: l'immagine è conservata come una matrice <291x240 uint8>
6. Provare ad usare il comando **`imfinfo('pout.tif')`**. Il risultato dell'operazione può anche essere assegnato ad una variabile, che assumerà la connotazione di "struttura" e quindi avrà dei campi che potranno essere esplorati: **`info = imfinfo('pout.tif');`**
   `info.Height`
7. L'immagine è poco contrastata! Esaminiamo l'istogramma di intensità
   `figure, imhist(I)`
8. L'istogramma appare "compresso": i toni di grigio non sfruttano pienamente il *range* disponibile: ciò determina lo scarso contrasto.



Istogramma di intensità (o dei grigi) (5)

1. Dare i comandi:
```
I2 = imadjust(I, stretchlim(I), []);
figure, imshow(I2)
figure, imhist(I2)
```
2. Si otterrà un'immagine decisamente più contrastata. Questo risultato è stato ottenuto "stirando" l'istogramma in modo che copra l'intero *range* di grigi disponibile.
3. Scrivere ora su disco l'immagine modificata (cambiando magari il formato dell'immagine in "png") con:
```
imwrite (I2, 'pout2.png');
```
4. Abbiamo realizzato lo **stretching dell'istogramma**; l'immagine ora sfrutta tutto il range di grigi ("dinamica") disponibile.





Istogramma di intensità (o dei grigi) (6)

Esercizio 1: ripetere le operazioni di aumento del contrasto per l'immagine **`enhance-me.gif`**, distribuita con il programma di analisi d'immagine ImageJ.

Esercizio 2: scrivere una funzione MATLAB che realizzi esplicitamente (cioè non usando gli strumenti dell'Image Toolbox) lo *stretching* dell'istogramma di un'immagine in toni di grigio. Alla funzione devono essere passati la variabile contenente l'immagine da elaborare e i valori minimo e massimo dei grigi da considerare utili per la trasformazione (gmin e gmax). La funzione restituisca l'immagine trasformata dopo lo *stretching* dell'istogramma.

```
I1 = hs(I, gmin, gmax);
```

Lo scopo è dunque rimappare i valori di grigio dei pixel sull'intera dinamica disponibile (tipicamente 0..255): ogni pixel (valore di grigio g) deve assumere un valore di grigio g1 direttamente proporzionale a g-gmin e inversamente proporzionale al *range* dinamico (escursione dei grigi: gmax-gmin).

I pixel i cui valori di grigio sono scarsamente rappresentati: g<gmin o g>gmax, collasseranno a 0 o a 255 rispettivamente.

Suggerimenti per la soluzione:

Attenzione a come si accoppiano le operazioni! Si rischia che vengano fatte tutte in uint8, e che ci sia overflow non esplicitamente dichiarato, con la conseguenza di errori di calcolo! Infatti il calcolo:(g-gmin)*255 va in overflow! Meglio scrivere (g-gmin)/(gmax-gmin)*255, così viene forzato il calcolo in double, oppure dichiarare g1 come double (o int32).

Ricordare di definire I1 (l'immagine trasformata) all'inizio: I1 = zeros(size(I), 'uint8');

Istogramma di intensità (o dei grigi) (7)

Esercizio 2: (continua)

Come ulteriore suggerimento, ricordiamo che l'operazione di trasformare il range: [gmin, gmax] in [0, 255] è simile a rimappare i numeri casuali compresi tra 0 e 1, in un insieme qualsivoglia, per esempio tra 7 e 12.

```
if g<gmin, g1=0, elseif g>gmax, g1=255,
else g1=255*(g-gmin)/(gmax-gmin), end
```

Se si ha difficoltà nella scrittura del codice, ricordare che questa è un'operazione puntuale, per cui servirà un doppio loop:

```
for r = 1:righe          % righe e' size(I,1)
    for c = 1:colonne
        g = I(r,c);
        g1 = …..;
        I1(r,c) = g1;
    end
end
```

Ulteriore sviluppo: trovare gmin e gmax in base ai percentili!
prctile

Applicazione di una soglia (sogliatura/thresholding) (1)

Nell'analisi di un'immagine è essenziale distinguere tra i pixel appartenenti agli oggetti di interesse e "il resto" dell'immagine (lo sfondo). Le tecniche usate per distinguere gli oggetti di interesse sono denominate tecniche di segmentazione.

- Non esiste alcuna tecnica di segmentazione valida per qualunque immagine
- Nessuna tecnica di segmentazione è perfetta

Tecnica comune : *thresholding* (operazione di soglia) e creazione di una maschera:

Supponiamo di lavorare su immagini a toni di grigio
Si sceglie un parametro θ, chiamato soglia di luminosità, che è applicato all'immagine $a(m,n)$ come segue, supponendo di essere interessati ad oggetti luminosi (chiari) su sfondo scuro:

- Se $a(m,n) > \theta$ allora $b(m,n) = 1$ (oggetto)
- Altrimenti $b(m,n) = 0$ (sfondo)

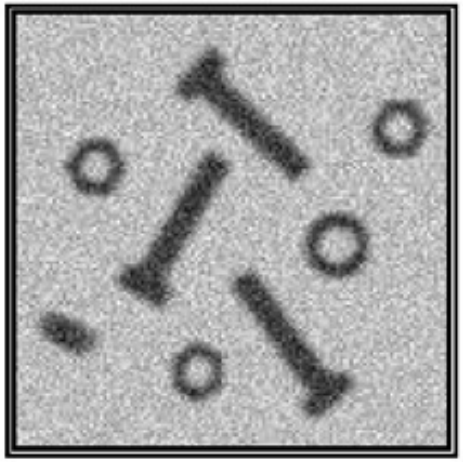
Nel caso opposto (oggetti scuri su fondo chiaro) occorre invertire il verso della disuguaglianza ("<" invece di ">").
Il concetto dell'applicazione di una soglia si generalizza considerando non solo soglie di luminosità, ma anche soglie su altre grandezze; in un'immagine a colori il test di soglia potrebbe essere: se la componente rossa del colore del *pixel* è maggiore di θ, allora il pixel appartiene ad un oggetto di interesse

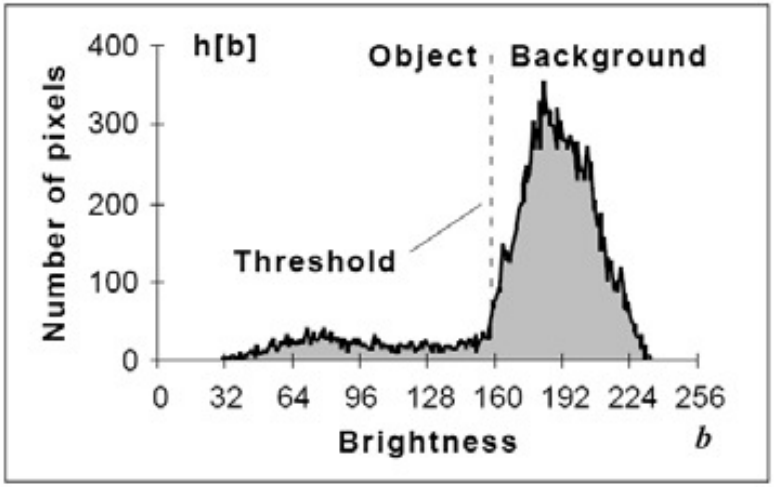
Applicazione di una soglia (sogliatura/thresholding) (2)

Come va scelta la soglia affinché il procedimento consenta una distinzione (sia pure grossolana e non definitiva) tra oggetti d'interesse e sfondo? Varie possibilità, tra cui:

• **soglia fissa** – scelta a priori in base ai dati dell'immagine. Es.: in immagini molto contrastate, dove gli oggetti sono molto scuri e lo sfondo è omogeneo e luminoso, una soglia pari a 128 (scala da 0 a 255) potrebbe essere sufficientemente accurata, nel senso che il numero di *pixels* erroneamente classificati è minimizzato.

• **soglia dedotta dall'istogramma** - Spesso la soglia è scelta in base all'istogramma di luminosità della regione o immagine che si desidera segmentare. Un'immagine e il suo istogramma di luminosità sono mostrati in figura.





Applicazione di una soglia (sogliatura/thresholding) (3)

1. Porre come current directory quella dov'é contenuta l'immagine **`blobs.gif`**.
2. Andare nella medesima directory partendo da Windows.
3. Aprire l'immagine da Windows cliccando due volte su di essa. Aprire l'immagine in Matlab con `I = imread('blobs.gif'); imshow(I);` Confrontare le due immagini (in Windows e in Matlab). Sono diverse!
4. Dare il comando `imfinfo('blobs.gif')`

Nelle **immagini indicizzate** (indexed) il valore del pixel non rappresenta direttamente il colore (o il grigio) bensì il posto occupato dal colore in una tabella di colori (LUT, Look Up Table, o mappa di colori) associata all'immagine… Utile quando i colori sono pochi, per comprimere l'immagine.

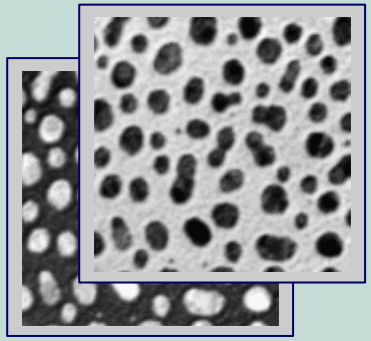
5. Rileggere l'immagine correttamente:

```
[I,map] = imread('blobs.gif');
figure, imshow(I, map)
```

6. Ed ora trasformarla in immagine NON indicizzata, per liberarsi della mappa di colori

```
I = ind2gray(I,map);
figure, imshow(I)
```

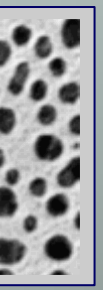
7. Confrontare le immagini, conservare solo l'ultima, esplorarla con `pixval` (oppure `impixelinfo`), decidere quale valore di soglia potrebbe opportunamente discriminare tra fondo e oggetti.



Threshold $\theta = 130$

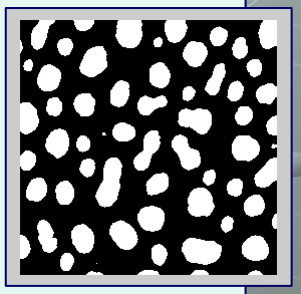
Giorgio De Nunzio giorgio.denunzio@unisalento.it

Applicazione di una soglia (sogliatura/thresholding) (4)



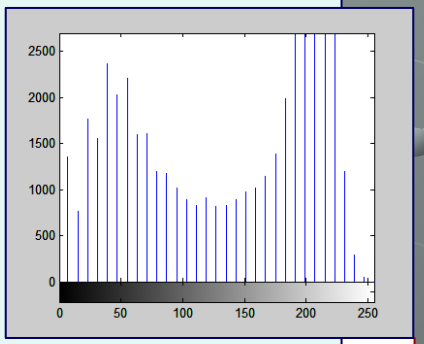
8. In MATLAB, applicare globalmente una soglia è immediato se l'immagine è a toni di grigio e contiene le intensità (non è indexed).

   Il comando **`B = A > th`** crea un'immagine binaria **B** (di classe '*logical*') in cui ciascun *pixel* ***b*[*m*,*n*]** è posto ad 1 ("vero", ossia condizione soddisfatta) se il corrispondente *a*[*m*,*n*] è maggiore della soglia th, è posto a 0 altrimenti. Il comando **`B = A < th`** si comporta in maniera inversa.



```
IM = I < 130;   % creazione della maschera
figure,imshow(IM)
```

9. Guardiamo ora l'istogramma dell'immagine originale:

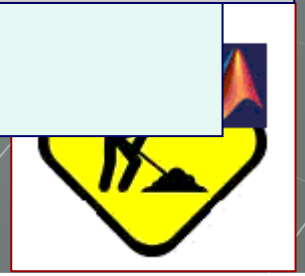



```
imhist(I)
```

   e verifichiamo se la scelta della soglia è stata coerente con la distribuzione dei grigi dell'immagine

10. Applichiamo la maschera all'immagine!

```
I2 = I .* uint8(IM) + uint8(255 * (1 - IM));
```



Applicazione di una soglia (sogliatura/thresholding) (5)

L'operazione di soglia non deve essere necessariamente applicata globalmente all'intera immagine con lo stesso valore θ (soglia uniforme), ma può essere piuttosto basata su una procedura sviluppata regione per regione.

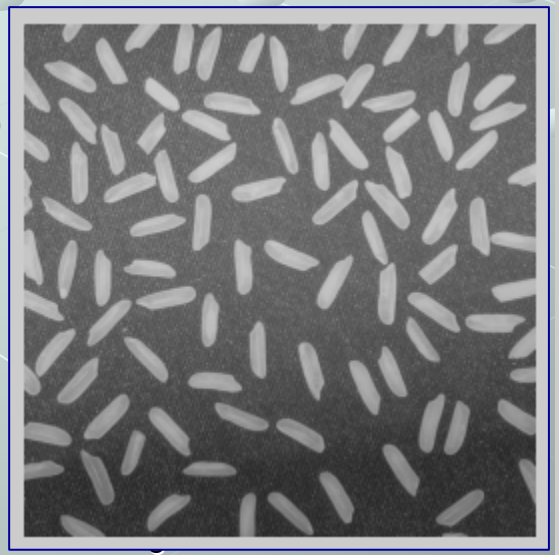
Chow-chow e Kaneko hanno sviluppato una variante in cui l'immagine è divisa in regioni disgiunte: in ogni regione si calcola la rispettiva soglia, e i valori di soglia risultanti sono interpolati in modo da formare una superficie di soglia per l'immagine intera. Le regioni dovrebbero essere di estensione "ragionevole" in modo da contenere ciascuna un numero sufficiente di pixel per valutare l'istogramma e quindi la soglia. L'utilità di questa procedura dipende naturalmente dal problema particolare da risolvere.

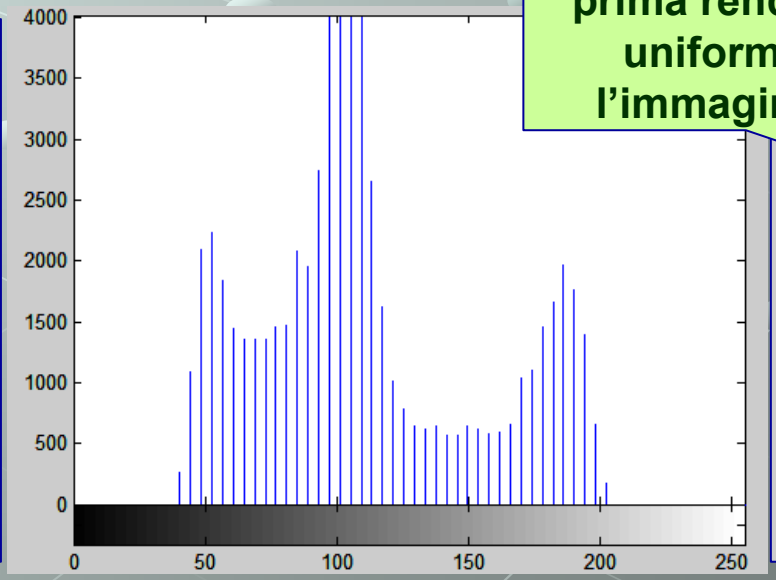
Applicazione di una soglia (sogliatura/thresholding) (6)

Vediamo cosa succede con un'immagine in cui la luminosità non sia uniforme:

```
I = imread('rice.png');
imhist(I,64)      % istogramma
J = I > 128;
figure, imshow(I), figure, imshow(J)
```

SOGLIA GLOBALE: NON APPROPRIATA! Oppure occorrerebbe prima rendere uniforme l'immagine!





Applicazione di una soglia (sogliatura/thresholding) (7)

Esiste una funzione preposta al compito di applicare una soglia: **`im2bw`**.

Applicabile anche ad immagini di tipo *indexed* (indicizzate); in quest'ultimo caso l'immagine è dapprima convertita in toni di grigio e poi è applicata la soglia.

La funzione **`im2bw`** richiede come secondo parametro il valore della soglia espresso come numero tra 0 e 1, per cui la conoscenza del range permesso per l'immagine non è importante: per esempio, 0.5 significa che la soglia è scelta a metà dei livelli permessi (e.g. 128 se l'immagine è a 8 bit).

Il codice diventa:

```
I = imread('rice.png');
imhist(I,64)
J = im2bw(I,0.5);
figure, imshow(I), figure, imshow(J)
```

Applicazione di una soglia (sogliatura/thresholding) (8)

MATLAB fornisce anche una funzione che ricava dall'immagine la soglia ottimale per la sua trasformazione in immagine binaria adoperando il metodo di Otsu che minimizza la varianza intraclasse dei *pixel* a 1 e dei *pixel* a 0.

```
I = imread('rice.png');
imhist(I,64)
level = graythresh(I) % individua la soglia e la scrive
J = im2bw(I,level);
figure, imshow(I), figure, imshow(J)
```

Il valore di soglia individuato è:

```
level =

   0.5137
```

Il metodo di Otsu non sempre è il più adatto!

Convoluzione (1)

Operazione locale, fondamentale per il trattamento d'immagini monocromatiche
Consente di eseguire operazioni di filtraggio, quali l'evidenziazione di particolari tipi di dettagli, la de-evidenziazione di altri, *smoothing*, e così via

La convoluzione di due segnali (eventualmente multidimensionali) è indicata comunemente in uno dei due modi seguenti:

$$c = a \otimes b = a * b$$

Definizione:

$$A \otimes B = \int_{-\infty}^{+\infty} A(u)B(x-u)du$$

Spazio bidimensionale continuo:

$$c(x,y) = a(x,y) \otimes b(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} a(\chi,\zeta)\, b(x-\chi, y-\zeta)\, d\chi\, d\zeta$$

Spazio bidimensionale discreto:

$$c[m,n] = a[m,n] \otimes b[m,n] = \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} a[j,k]\, b[m-j, n-k]$$

Convoluzione (2)

Quando i segnali in questione sono immagini (quindi di estensione finita), le definizioni precedenti vanno naturalmente adattate.
Uno dei segnali è l'immagine che si desidera elaborare, l'altro (*convolution kernel*) ha dimensioni in generale inferiori (ovvero può considerarsi funzione nulla al di fuori di una finestra, il supporto, in cui assume valori diversi da zero) ed è utilizzato come filtro.

La definizione diventa allora la seguente, in cui il *kernel*, indicato con il simbolo $h[j,k]$, è nullo fuori da una finestra rettangolare $\{j = 0,1,\ldots,J\text{-}1;\ k = 0,1,\ldots,K\text{-}1\}$:

$$c[m,n] = a[m,n] \otimes h[m,n] = \sum_{j=0}^{J-1} \sum_{k=0}^{K-1} h[j,k]\, a[m-j, n-k]$$

Convoluzione (3)

Operativamente, una "*sliding window*" (finestra mobile) – finestra (o maschera, o *kernel*) di convoluzione – è via via centrata su ogni pixel dell'immagine in input, e genera in corrispondenza nuovi pixel di output. La maschera è INVERTITA (rispetto all'origine, il suo centro) altrimenti si parla di correlazione.

I valori di intensità della maschera di convoluzione fungono da pesi in una somma pesata dei pixel dell'immagine originale, eseguita in un intorno del pixel sul quale la maschera è centrata (l'operazione è **locale**).

Ciascun nuovo pixel è calcolato moltiplicando ogni valore d'intensità dei pixel di partenza con il corrispondente peso della maschera di convoluzione e sommando, successivamente, tutti questi prodotti.

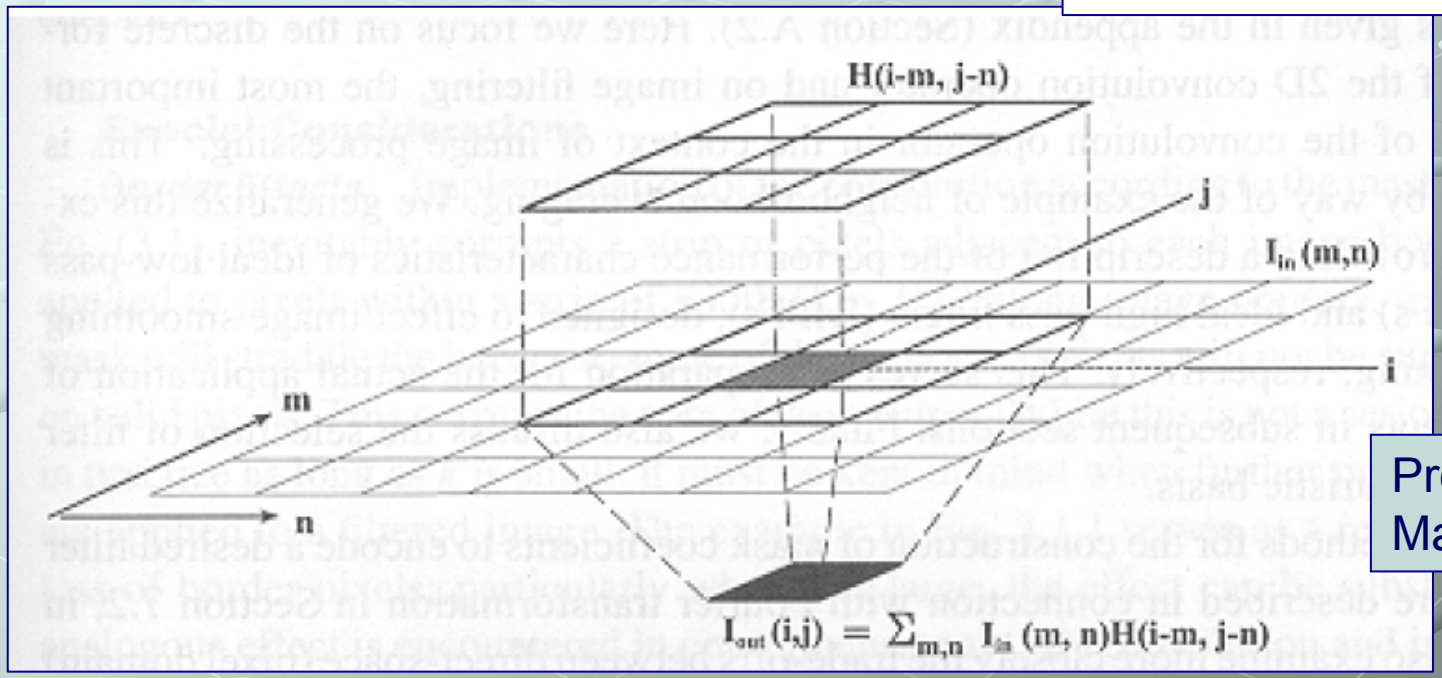
Si chiama convoluzione, pertanto, l'insieme delle tre operazioni:
(1) spostamento, lungo tutta l'immagine, della maschera di convoluzione (invertita),
(2) moltiplicazione dei valori del livello di grigio dei pixel che si sovrappongono (il pixel dell'immagine moltiplicato per il pixel corrispondente della maschera),
(3) somma dei valori trovati e relativa generazione pixel per pixel della nuova immagine.

Convoluzione (4)

Illustrazione della procedura, con una **finestra-immagine di input *I* (*x*, *y*)** e una **finestra-maschera di convoluzione *H*(*x*, *y*)** (matrice M×N, avente per elementi quelli della maschera di convoluzione).

L'elemento (*i*, *j*) di $I \otimes H$ di *I* con *H* è definito così:

$$(I \otimes H)_{i,j} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(m,n)\, H(i-m, j-n)$$



Problema dei bordi…
Maschere separabili…

APPLICAZIONI: Variando la natura del *kernel*, la convoluzione consente di effettuare elaborazioni diversissime sull'immagine, ad esempio lo *smoothing* o lo *sharpening* (rispettivamente addolcimento e evidenziazione dei dettagli), o la ricerca dei bordi degli oggetti rappresentati.

Convoluzione (5)

Smoothing tramite filtro lineare

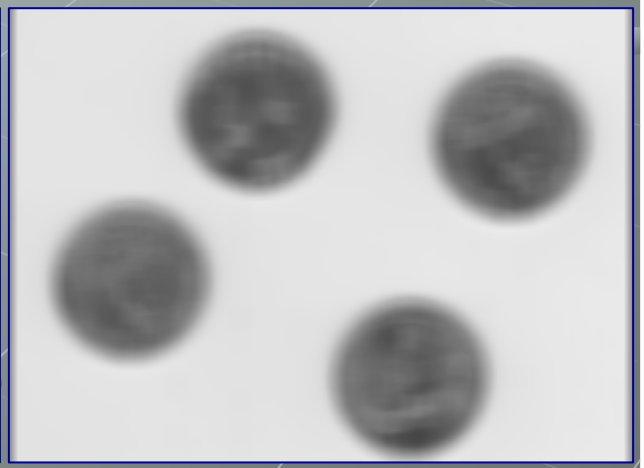
```
I = imread('eight.tif'); % Legge l'immagine originale

h = ones(5,5) / 25 % Kernel uniforme normalizzato 5x5
I2 = imfilter(I,h, 'conv'); % Convoluzione

k = ones(11,11) / 121 % Kernel uniforme normal. 11x11
I3 = imfilter(I,k,'conv'); % Convoluzione

imshow(I), figure, imshow(I2), figure, imshow(I3)
```

$$h = \frac{1}{25}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$



Convoluzione (6)

Smoothing tramite filtro lineare, con rumore

```
I = imread('eight.tif'); % Legge l'immagine originale

% (usare pixval, poi settare dei pixel come rumore)
I(202,82)=0; I(202,84)=0;  % coordinate invertite! Mettere i ";"!

h = ones(3,3) / 9 % Kernel uniforme normalizzato 3x3
I2 = imfilter(I,h, 'conv'); % Convoluzione
imshow(I), figure, imshow(I2)

%
h = fspecial('average', [3,3]);  % cosa e'?
h = fspecial('gaussian',[3,3],0.5);  % provare con filtro gaussiano
```

RIDUZIONE DEL RUMORE!
(ma anche dei dettagli…)

Smoothing non lineare

Provare il seguente metodo alternativo (non lineare, dunque non è applicazione della convoluzione) per smussare l'immagine:

FILTRO MEDIANO:

```
mediano3x3 = medfilt2(I, [3,3]);
imshow(mediano3x3);
```

Median filtering is a nonlinear operation often used in image processing to reduce "salt and pepper" noise. A median filter is more effective than convolution when the goal is to simultaneously reduce noise and preserve edges.

Smoothing non lineare

Provare il seguente metodo alternativo (non lineare, dunque non è applicazione della convoluzione) per smussare l'immagine:

FILTRO MID-POINT

```
minimo3x3 = ordfilt2(I, 1, ones(3,3));  % oppure true(3)
massimo3x3 = ordfilt2(I, 9, ones(3,3));
midpoint3x3 = 0.5 .*(minimo3x3 + massimo3x3); imshow(midpoint3x3);
```

(provare anche dimensioni diverse del kernel: 5x5 o 7x7)

In the midpoint method, the color value of each pixel is replaced with the average of maximum and minimum (i.e. the midpoint) of color values of the pixels in a surrounding region. A larger region (filter size) yields a stronger effect.

ORDFILT2 2-D order-statistic filtering.

B=ORDFILT2(A,ORDER,DOMAIN) replaces each element in A by the ORDER-th element in the sorted set of neighbors specified by the nonzero elements in DOMAIN.

Example

Use a maximum filter on snowflakes.png with a [5 5] neighborhood. This is equivalent to imdilate(image,strel('square',5)).

```
A = imread('snowflakes.png');
B = ordfilt2(A,25,true(5));
figure, imshow(A), figure, imshow(B)
```

Inserimento di rumore

imnoise

IMNOISE Add noise to image.
J = IMNOISE(I,TYPE,...) Add noise of a given TYPE to the intensity image
I. TYPE is a string that can have one of these values:

'gaussian' Gaussian white noise with constant mean and variance
'localvar' Zero-mean Gaussian white noise with an intensity-dependent variance
'poisson' Poisson noise
'salt & pepper' "On and Off" pixels
'speckle' Multiplicative noise

Alternativa per l'aggiunta di rumore bianco gaussiano:
noisy = x + n con
n = d*randn(size(x))
dove d è la deviazione standard del rumore.

Note

The mean and variance parameters for 'gaussian', 'localvar', and 'speckle' noise types are always specified as if for a double image in the range [0, 1]. If the input image is of class uint8 or uint16, the imnoise function converts the image to double, adds noise according to the specified type and parameters, and then converts the noisy image back to the same class as the input.

Esercizio

Denoising. Aggiungete del rumore (di varia specie) ad un'immagine x. Effettuate il denoising dell'immagine con vari filtri (e al variare della dimensione della finestra).
Valutate l'efficacia del filtraggio sia visivamente, sia calcolando l'errore quadratico medio tra x e l'immagine "ripulita", che rappresenta una misura quantitativa per stabilire quanto l'immagine elaborata sia simile all'originale.
L'MSE (Mean Squared Error) tra due immagini si definisce come:

$$\mathrm{MSE} = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |x(m,n) - y(m,n)|^2$$

ATTENZIONE AI LIMITI!

pdist….!

Costruite una tavola comparativa tra i tipi di rumore e i vari filtri per attenuarlo.

	uniform	**gauss**	**median**	**mid-point**
Salt & pepper				
gaussian				
speckle				

MSE

Convoluzione (7)

Edge detection

Operatore di Sobel

Coppia di maschere di convoluzione, di dimensione 3×3. Una si ottiene dall'altra per semplice rotazione di 90°:

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Convoluzione (8)

Edge detection

```
I = double(imread('rice.png')); % evita problemi...
h = [-1 0 1 ; -2 0 2 ; -1 0 1] % filtro di Sobel bordi vertic.
k = [1 2 1 ; 0 0 0 ; -1 -2 -1] % filtro di Sobel bordi orizz.
I2 = imfilter(I,h); I3 = imfilter(I,k); % applico i filtri
th = 100; % soglia per bordi significativi
I4 = abs(I2) > th; I5 = abs(I3) > th;
I6 = bitor(I4,I5);
imshow(I,[]);
figure;imshow(I4,[]); figure;imshow(I5,[]);figure;imshow(I6,[]);
```

Operazione logica “or”

FARE PRIMA LA SOMMA E POI APPLICARE LA SOGLIA? abs(I2)+abs(I3)>100 (o 200? Toglie rumore). UTILE PER BORDI OBLIQUI? Operazioni invertite: d’altra parte bitor è una specie di somma!

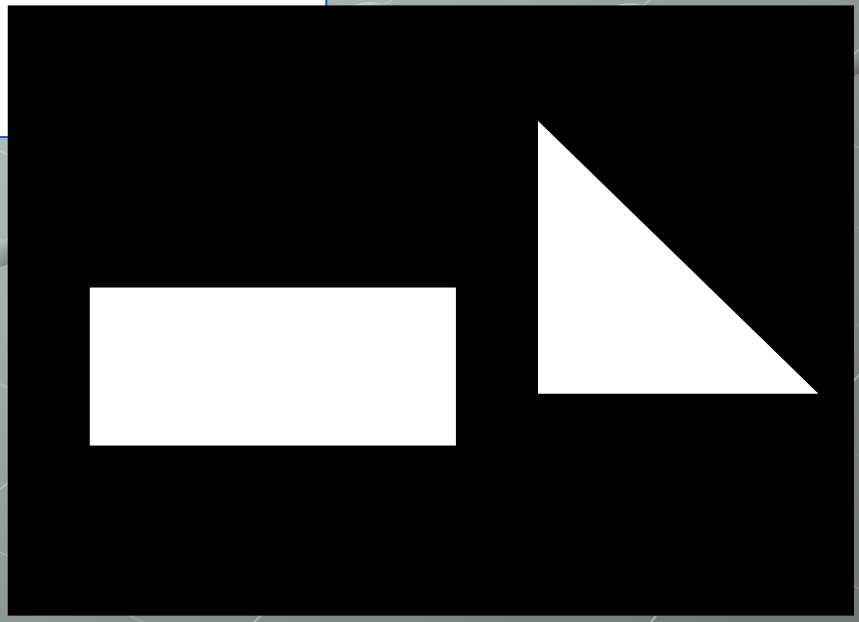
Convoluzione (9)

Edge detection

Esercizio: creare un'immagine che contenga un rettangolo e un triangolo rettangolo isoscele (bordi 45°, come da figura) bianchi su sfondo nero, poi applicare Sobel.

Per il triangolo, usare loop oppure "tril":

B=tril(ones(10,10))



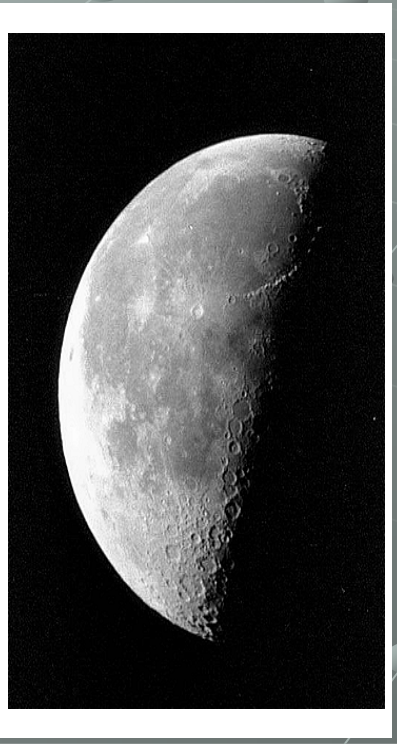
Convoluzione (10)

Unsharp masking

La funzione fspecial produce diversi filtri predefiniti. Dopo avere creato un filtro con fspecial, è possibile applicarlo direttamente con imfilter. L'esempio che segue illustra l'applicazione di un filtro '*unsharp masking*' ad un'immagine monocromatica della Luna. L'effetto è rendere bordi e dettagli fini più netti ed evidenti, sebbene la contropartita sia l'introduzione di un certo rumore di fondo.

```
I = imread('moon.tif');
h = fspecial('unsharp');
I2 = imfilter(I,h);
imshow(I), figure, imshow(I2)
```

```
>> h
h =
   -0.1667   -0.6667   -0.1667
   -0.6667    4.3333   -0.6667
   -0.1667   -0.6667   -0.1667
```



Operazioni morfologiche 1

Morfologia matematica

- branca della matematica rivolta all'elaborazione di immagini
- basata sull'elaborazione delle forme
- Usata per rimuovere particolari irrilevanti e mantenere le informazioni importanti sulla forma
- Lavora su immagini b/n (binarie) e a toni di grigio
- Derivata dall'insiemistica
- La struttura dell'immagine viene "sondata" con un insieme di forma definibile dall'utente (***elemento strutturante***) solitamente codificato da un piccola immagine raster (3×3 o 5×5).

Da: Morfologia Matematica - Alessandra Fascioli - Visione Artificiale

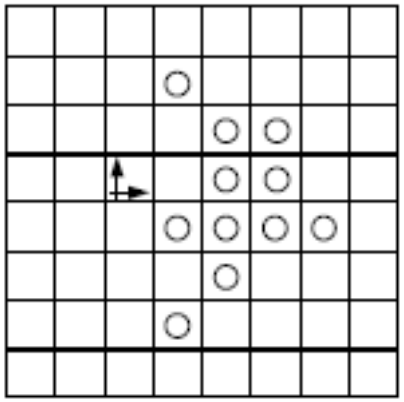
Operazioni morfologiche 2

Immagini binarie (BW)

L'immagine A è definita da:

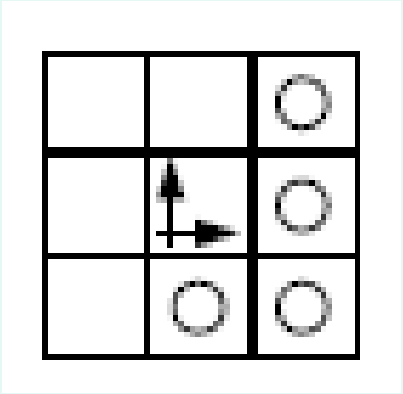
- un insieme di elementi (i pixel p_i)
- l'origine O del sistema di riferimento

$$A = \{ p_i \mid p_i = (x_i, y_i) \}$$



Operazioni morfologiche 3

- Elementi strutturanti (structuring elements) sono immagini – tipicamente di dimensioni ridotte – che parametrizzano le operazioni



Operazioni morfologiche 4

Definizioni

- *Intersezione* di due immagini:

$$A \cap B = \{ p \mid p \in A \text{ AND } p \in B \}$$

- *Unione* di due immagini:

$$A \cup B = \{ p \mid p \in A \text{ OR } p \in B \}$$

- Immagine *traslazione* rispetto a p :

$$A_p = \{ a+p \mid a \in A \}$$

p è un vettore
(o un punto rispetto
alla sua origine)

Operazioni morfologiche 5

Gli operatori fondamentali sono:

- Espansione o dilatazione (dilation)
- Erosione (erosion)

La dilatazione aggiunge pixel ai contorni di un oggetto, mentre l'erosione li rimuove. Il numero di pixel aggiunti o rimossi dipendono dalla forma e dalla dimensione dell'elemento strutturante.

Operatori più complessi (apertura, chiusura, hit-or-miss) sono costruiti come combinazione dei precedenti

Operazioni morfologiche 6

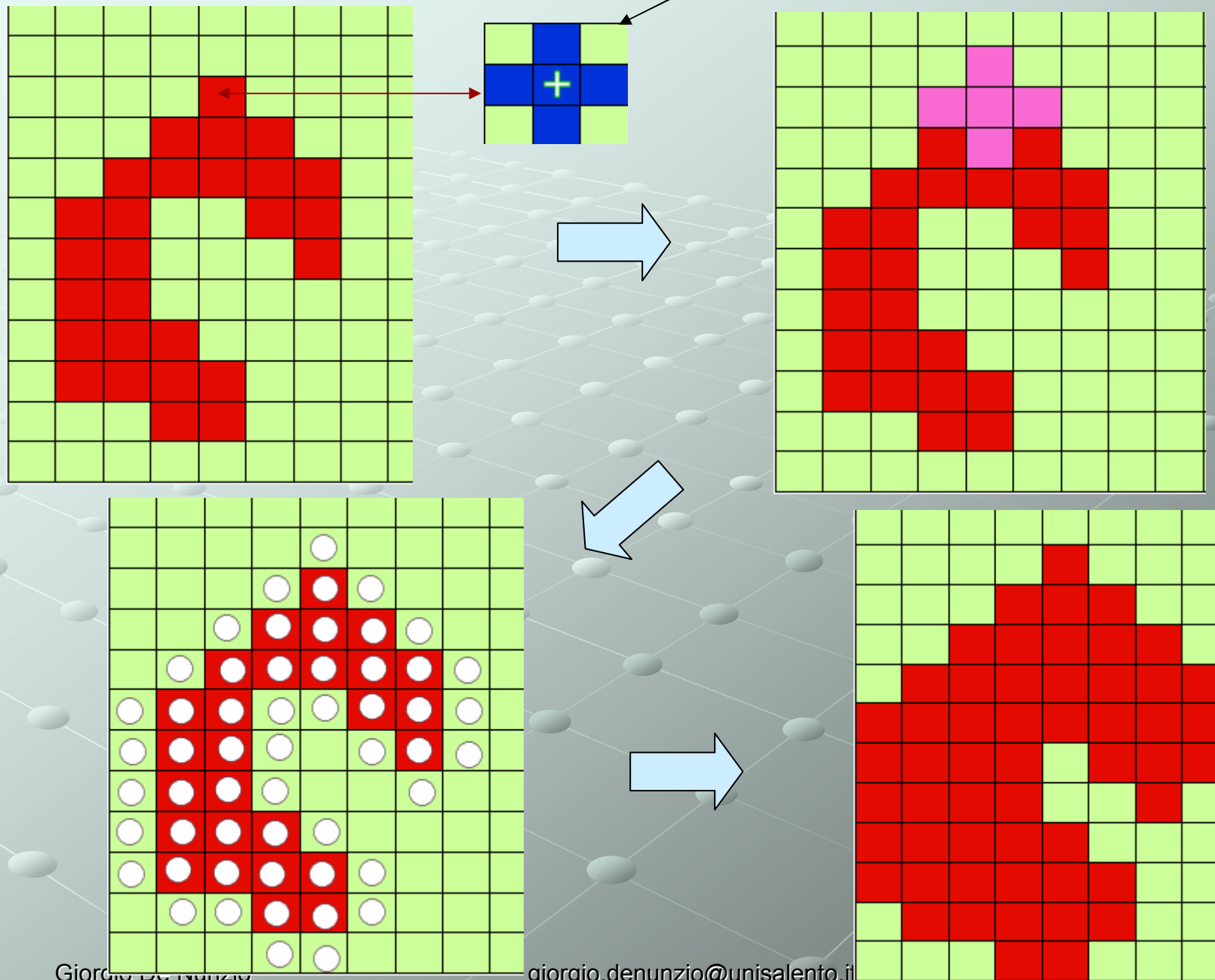
ESPANSIONE (DILATION)

- Se A_{b_1}, A_{b_2},... A_{b_n} sono traslazioni di A con i pixel dell'elemento strutturante $B=\{b_1,...b_n\}$, allora l'unione delle traslazioni è chiamata *espansione* di A con B e si scrive:

$$A \oplus B = \bigcup_{b_i \in B} A_{b_i}$$

- È caratterizzata da proprietà associative e commutative

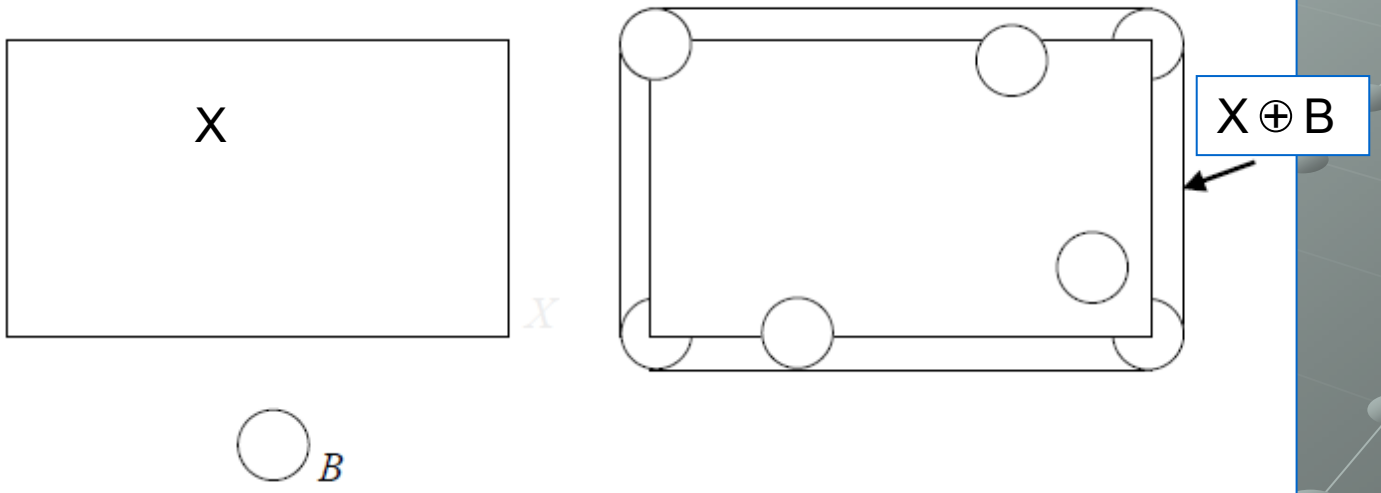
5 possibili traslazioni, da unire



Operazioni morfologiche 7

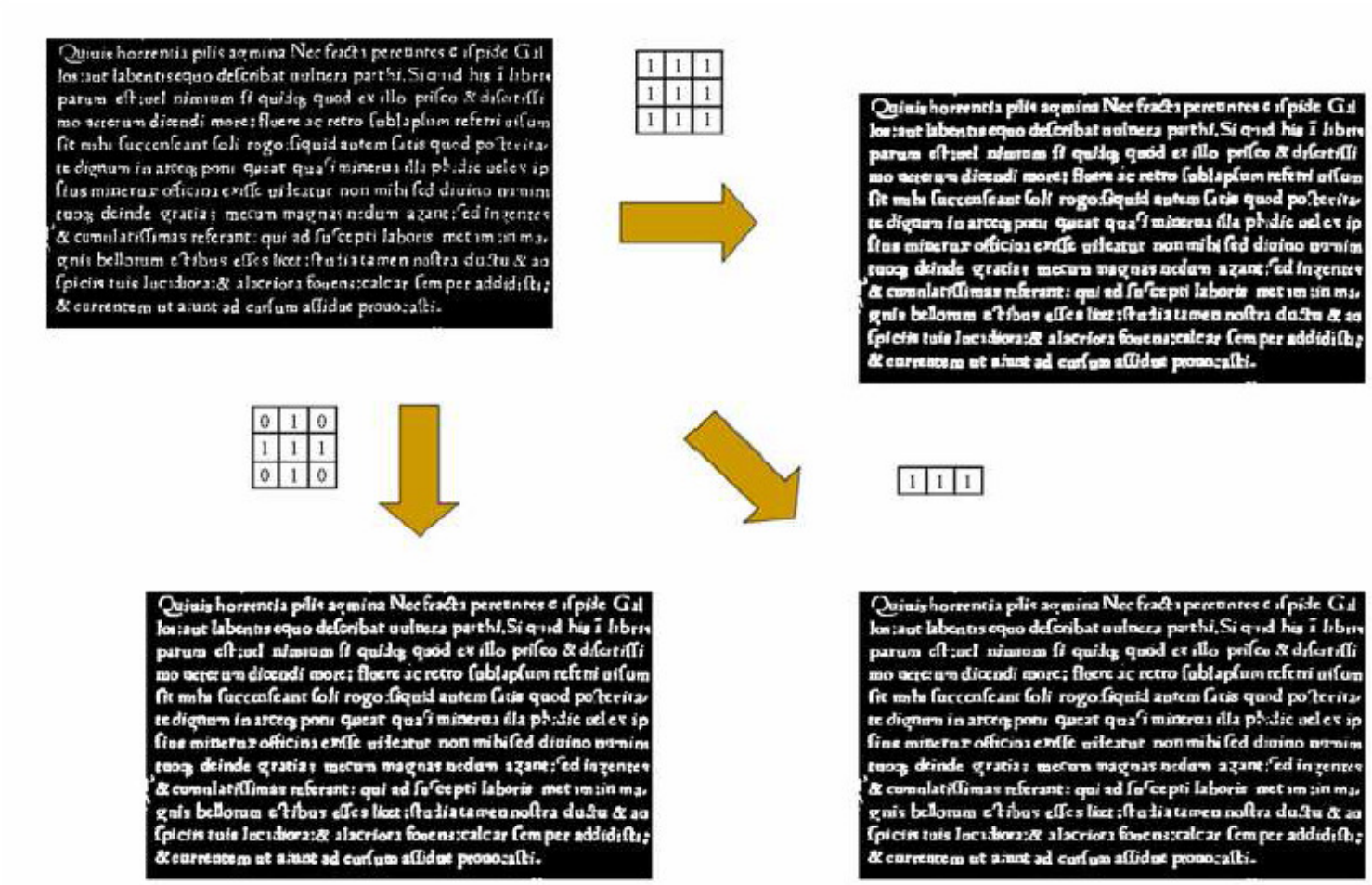
Espansione

- Esempio:



Operazioni morfologiche 8

Applicazioni: Riempimento



Operazioni morfologiche 9

Erosione

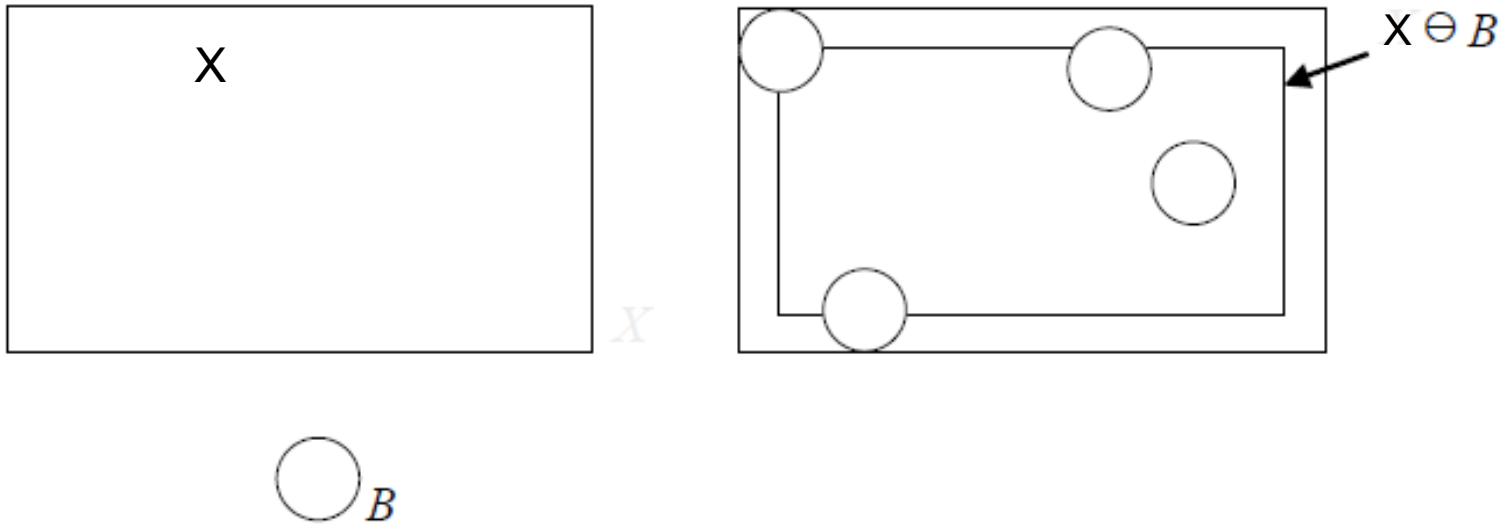
- L'*erosione* di A con B è l'insieme dei pixel p per i quali la traslazione di B rispetto ad essi genera un'immagine contenuta in A e si scrive:

$$A \ominus B = \{ p | B_p \subseteq A \}$$

Operazioni morfologiche 10

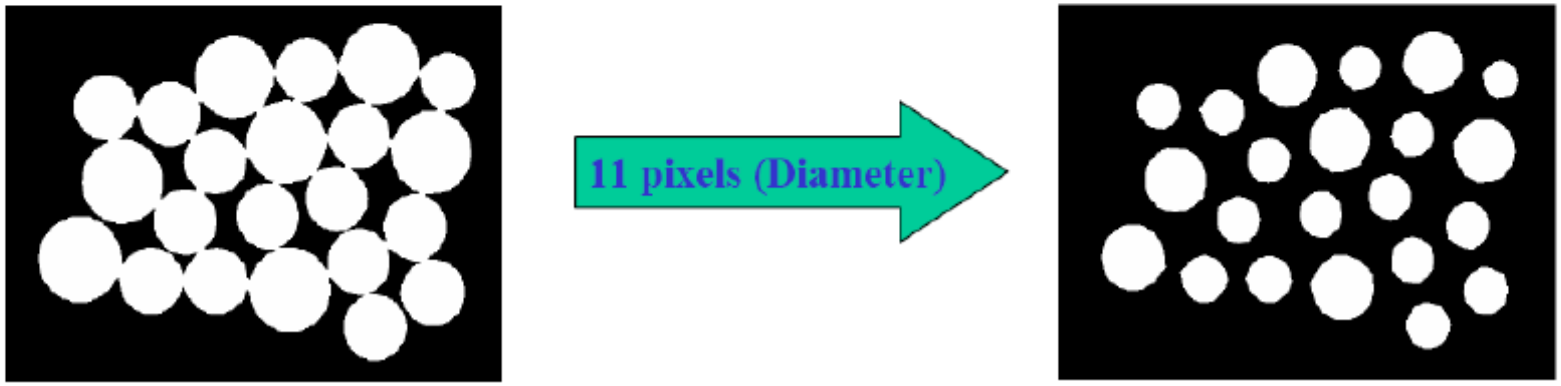
Erosione

- Esempio:



Operazioni morfologiche 11

Esempio di Erosione



Operazioni morfologiche 12

Proprietà

- Definendo la *riflessione* di un'immagine rispetto all'origine

$$B' = \{-p \mid p \in B\}$$

valgono le seguenti relazioni:

$$\overline{A \oplus B} = \overline{A} \ominus B' \qquad \overline{A \ominus B} = \overline{A} \oplus B'$$

diversamente dalle regole di De Morgan

Operazioni morfologiche 13

Apertura

- L'*apertura* di A con B è la successione di una erosione e una espansione con B e si scrive:

$$A \circ B = (A \ominus B) \oplus B$$

L'apertura separa oggetti debolmente uniti e rimuove regioni piccole

Operazioni morfologiche 14

Chiusura

- La *chiusura* di A con B è la successione di una espansione e una erosione con B e si scrive:

$$A \bullet B = (A \oplus B) \ominus B$$

La chiusura riempie i buchi e le piccole concavità e rafforza l'unione di regioni connesse debolmente

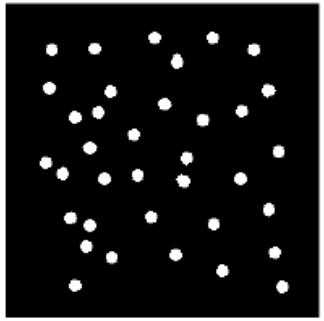
Operazioni morfologiche 15

Apertura e chiusura

- Sono operazioni la cui applicazione iterativa non provoca modifiche
  - Paragonabili a filtraggi passa-banda in cui la banda è data dall'elemento strutturante

Operazioni morfologiche 16

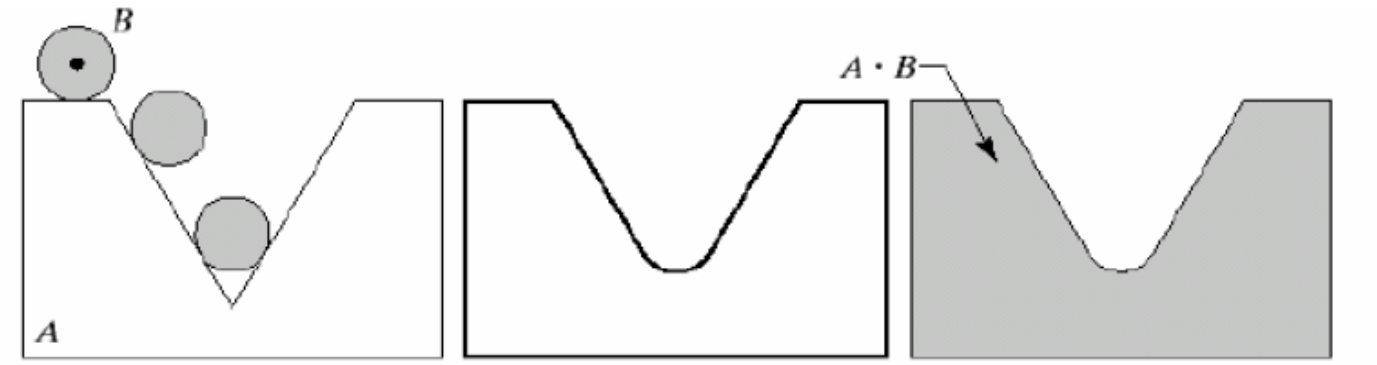
Opening

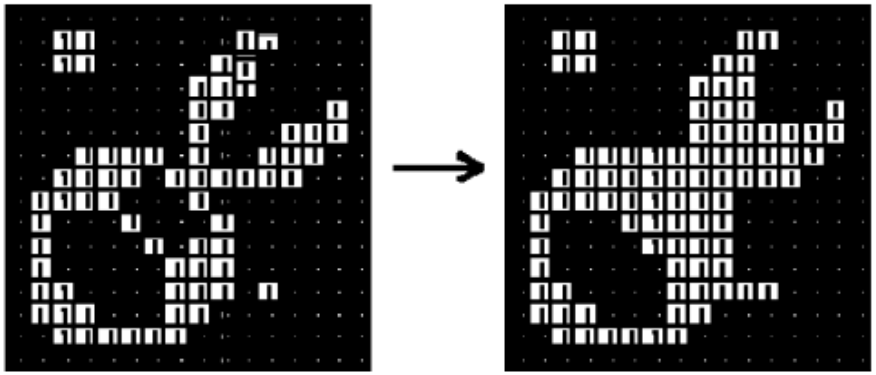


Un esempio di problema che richiede l'applicazione dell'apertura è l'eliminazione delle linee dall'immagine in figura. In questo caso viene utilizzato un elemento strutturale a forma sferica di raggio pari a quello dei cerchi da preservare che è maggiore dello spessore delle linee.

Operazioni morfologiche 17

Closing





Operazioni morfologiche 18

Morfologia a livelli di grigio

- Le definizioni viste possono essere estese al caso delle immagini a toni di grigio
- Nel seguito si vedranno solo i risultati, senza entrare nel dettaglio delle definizioni

Operazioni morfologiche 19

Morfologia in MATLAB

Gli operatori di base sono disponibili tramite i comandi MATLAB:

```
C=imerode(A,B)
C=imdilate(A,B)
C=imopen(A,B)
C=imclose(A,B)
```

O a livelli di grigio!

dove **A** e **C** sono immagini binarie e **B** è una matrice di 0 e 1 che specifica l'elemento strutturale.

L'elemento strutturale può essere anche generato utilizzando la funzione **strel,** che grazie ad una serie di parametri permette la creazione di diverse forme di varia grandezza.

Guardare cosa è s, generato da strel

```
x = imread('circbw.tif'); figure(1); imshow(x);
s = strel('rectangle',[40 30]);
z = imerode(x,s);
y = imdilate(z,s); figure(2); imshow(y);
```

Operazioni morfologiche 20

Esercizio: costruire un rettangolo, e poi applicare erosione seguita da dilatazione, per vedere quanto I due rettangoli (pre e post elaborazione) sono congruenti
Si nota che open e close hanno effetti diversi!

Esercizi: da MORFOLOGICHE_ex4.pdf, immagini in “immagini4.zip”

Operazioni morfologiche 21

Vedere MATLAB e Image Processing Toolbox, esempio 2

Etichettare oggetti 1

Esercizio: scrivere un programma, basato su alcune function, che:

1) Crea un'immagine vuota (nera) uint8, di dimensioni passate come parametro
2) Richiama alcune volte le funzioni che disegnano, rispettivamente, un cerchio o un quadrato, di diametro e lato passati come parametro, e posizione (centro) passata anch'essa come parametro. Le figure sono disegnate accendendo pixel a valore 255. Usare rand per le posizioni e i raggi (dispari!altrimenti non si puo' definire un centro coincidente con un pixel!), senza preoccuparsi se ci sono sovrapposizioni (dara' luogo a false classificazioni)
3) Individua gli oggetti disegnati, tramite bwlabel, e ne calcola dei parametri tramite regionprops (area, diametro equivalente)
4) Distingue i quadrati dai rettangoli in base al confronto tra l'area effettiva e quella del cerchio di diametro pari al diametro effettivo, emttendo delle crocette sui quadrati

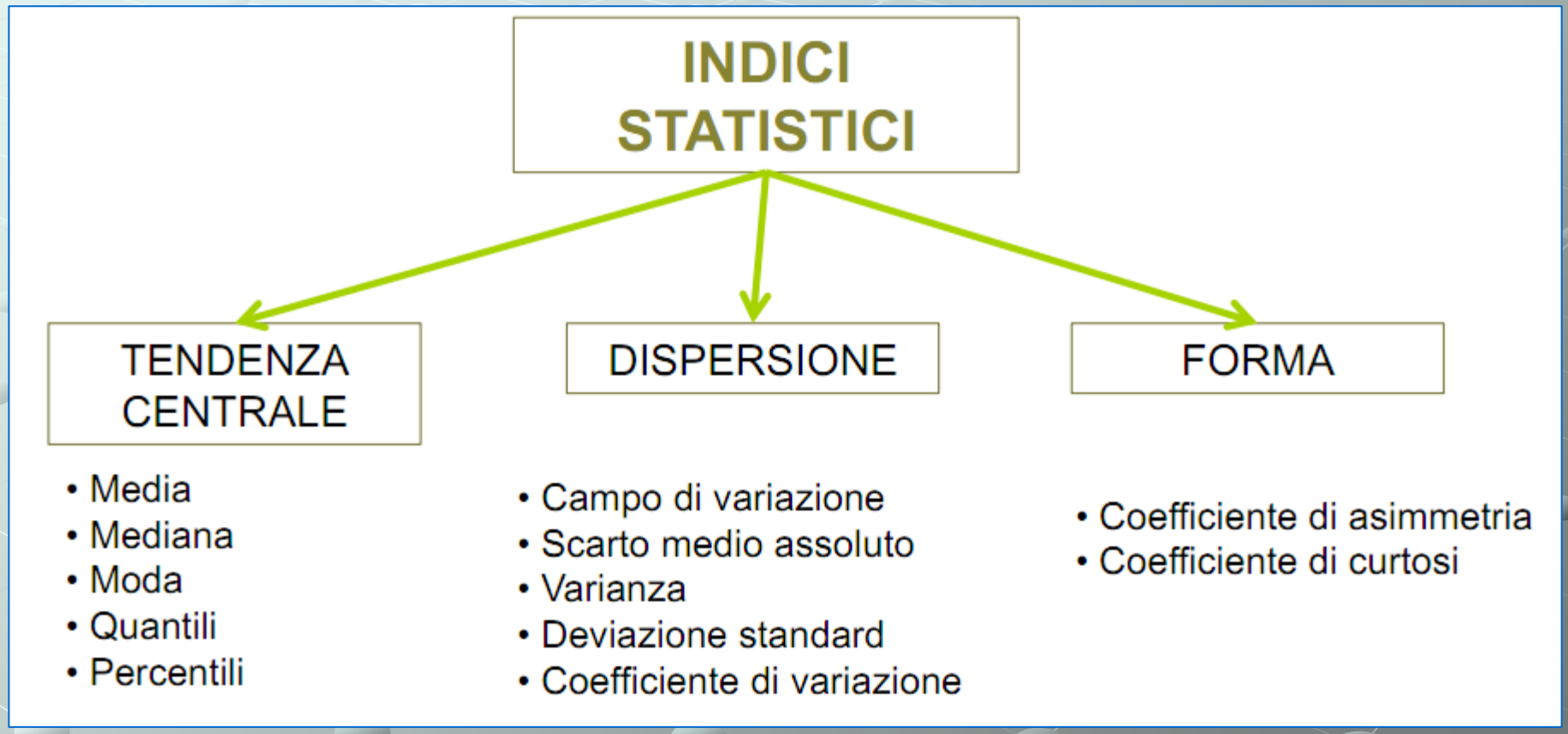
```
bwlabel
regionprops
```

CAD

- Funzionamento di un cad
- TP, FP, etc – sensib e specif
- Curva ROC

Feature tessiturali di 1° ordine

Indici calcolati dalla distribuzione di grigi



http://www.fe.infn.it/didattica/ing/ingciv/mom/momenti.pdf
http://www.mind.disco.unimib.it/public/site_files/file/Materiale%20Didattico/Lezione1.pdf

Momenti di una distribuzione di probabilità (1)

- In statistica, il **momento semplice** di ordine k di una variabile casuale discreta è definito come la media della k-esima potenza dei valori della variabile:

$$\mu_k = \sum_{i=1}^{n} x_i^k p_i$$

dove p_i è la funzione di probabilità della variabile casuale.

Notare che μ_1 è la media (valore atteso) della variabile casuale.

Il **momento centrale** di ordine *k* è definito come la media della *k*-esima potenza dello scarto dalla media $\mu = \mu_1$

$$m_k = \sum_{i=1}^{n} (x_i - \mu)^k p_i$$

Momenti di una distribuzione di probabilità (2)

Caratteristiche dei momenti semplici e centrali:

- μ_o e m_o sono uguali a 1
- m_1 è uguale a 0
- μ_1 è la media aritmetica, indicata usualmente con μ
- $m_2 = \mu_2 - \mu_1^2$ è la varianza, indicata tradizionalmente con σ^2
- In generale, la relazione tra il momento centrale (m_k) e i momenti semplici (μ_l) è data da:

$$m_k = \sum_{r=0}^{k} C(k;r)\mu_{k-r}(-\mu)^r$$

In rosso, i momenti più rilevanti in statistica!

per cui:

- $m_3 = \mu_3 - 3\mu_2\mu + 2\mu^3$ è la asimmetria, o skewness
- $m_4 = \mu_4 - 4\mu_3\mu + 6\mu_2\mu^2 - 3\mu^4$ è la curtosi (kurtosis)

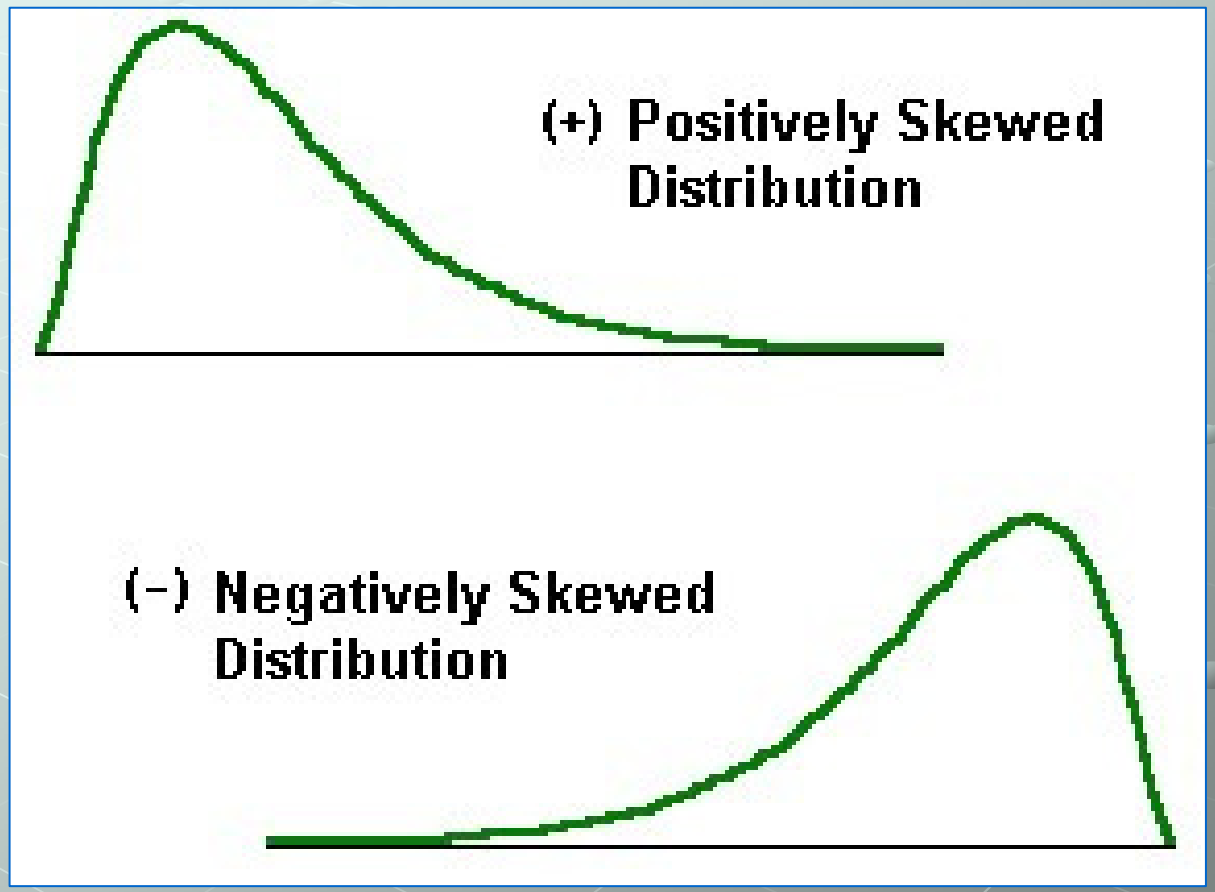
In analisi di immagini si considera l'immagine I_{mn} come un campione di una variabile casuale tratto da una distribuzione di probabilità (immaginando di "linearizzare" la matrice I, nel senso che dà MATLAB a questa operazione, cioè ogni I_{mn} è uno dei valori x_i).

Momenti di una distribuzione di probabilità (3)

Indice di asimmetria, o skewness

- le distribuzioni simmetriche sono costituite da due parti approssimativamente speculari rispetto al valore centrale del campo di variazione delle osservazioni,
- in caso di asimmetria si osserva un maggiore addensamento delle frequenze in una delle due parti.
- Prendendo la media come indice di centralità della distribuzione, in caso di asimmetria positiva gli scarti di segno positivo tendono ad essere di entità numerica superiore a quelli di segno negativo, mentre il contrario avviene in caso di asimmetria negativa.
- Non potendo prendere come indice di asimmetria la media degli scarti dalla media, poiché tale valore risulta sempre nullo, è conveniente utilizzare la media degli scarti al cubo, ovvero il terzo momento centrale, il quale risulta positivo in caso di asimmetria positiva (ovvero quando prevalgono numericamente gli scarti di segno positivo), negativo in caso di asimmetria negativa (ovvero quando prevalgono numericamente gli scarti di segno negativo), oppure approssimativamente nullo in caso di simmetria (ovvero quando gli scarti negativi equivalgono quelli positivi).
- Usualmente il terzo momento centrale viene standardizzato, dividendolo per il cubo della deviazione standard

Momenti di una distribuzione di probabilità (4)



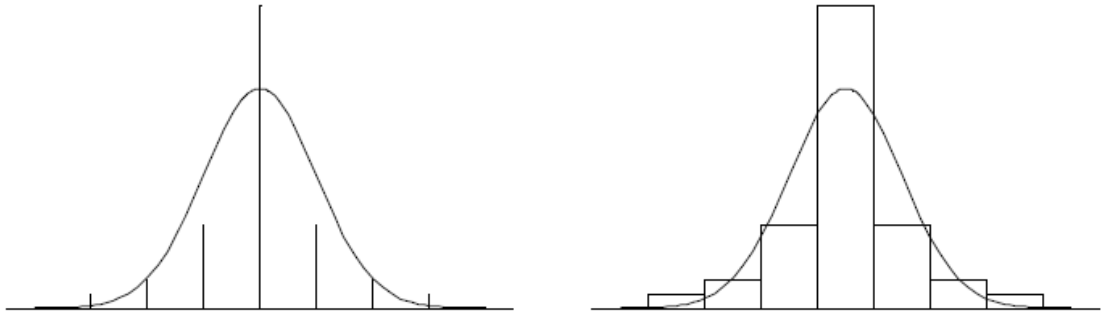
Momenti di una distribuzione di probabilità (5)

Indice di curtosi.

- La curtosi è un particolare aspetto delle distribuzioni di frequenza che riguarda la morfologia delle code.
- Una distribuzione campanulare con frequenze molto elevate in corrispondenza del valore modale e code molto lunghe, ovvero con frequenze che decrescono lentamente, è detta leptocurtica
- Una distribuzione campanulare con frequenze piuttosto basse in corrispondenza del valore modale e code molto corte, ovvero con frequenze che decrescono rapidamente è detta platicurtica
- L'elemento separatore tra queste due tipologie di distribuzioni è considerata la curva normale.
- Le distribuzioni di tipo uniforme e a forma di U costituiscono casi limite delle distribuzioni leptocurtiche.
- Per quantificare la curtosi di una distribuzione di frequenza si considera di solito il quarto momento centrale
- Usualmente il quarto momento centrale viene standardizzato, dividendolo per la quarta potenza della deviazione standard

Momenti di una distribuzione di probabilità (4)

Figura 5.1. Esempi di distribuzioni campanulari di tipo leptocurtico in rapporto alla curva normale.



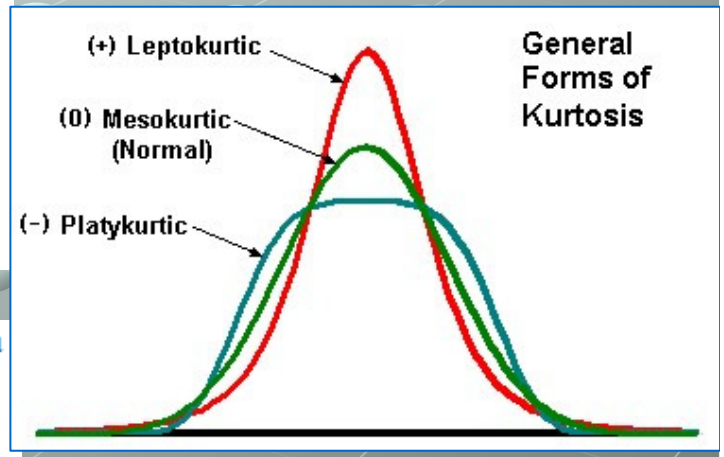
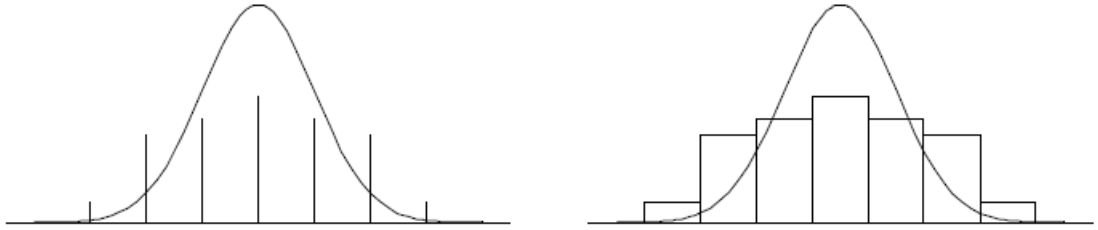


Figura 5.2. Esempi di distribuzioni campanulari di tipo platicurtico in rapporto alla curva normale.

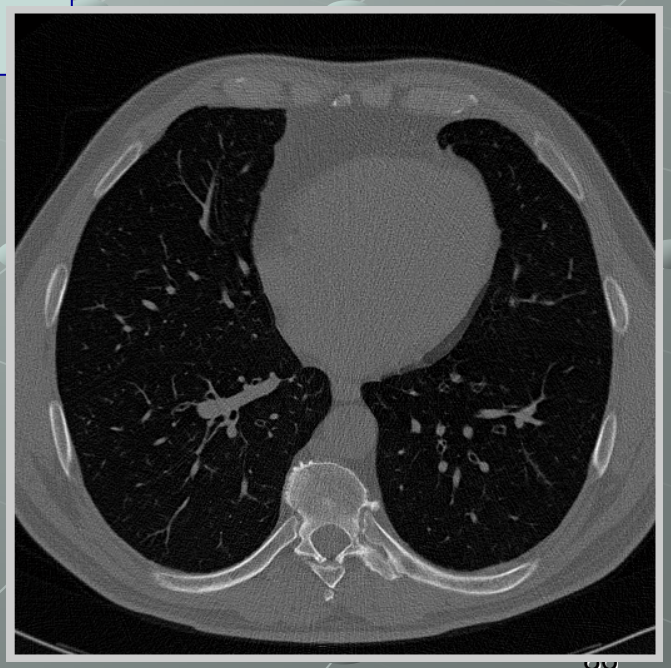


Lavorare con file DICOM (1)

```
cd segmenta2D
I = dicomread('16147870');
imshow(I,'DisplayRange',[])
                        % oppure
imshow(I,[])
```

Siccome l'immagine è a 16 bit, dobbiamo usare la funzione di autorange della scala dei grigi per vedere tutte le sfumature mappate nel range [0,255] visualizzato dallo schermo.

Provare pixval o impixelinfo!



Lavorare con file DICOM (2)

Per leggere i metadati dall'immagine DICOM, usiamo la seguente sintassi per riempire la variabile **info** (una struttura):

```
info = dicominfo('16147870')

info =
                          Filename: '16147870'
                      FileModDate: '26-Jan-2004 08:11:46'
                         FileSize: 526972
                           Format: 'DICOM'
                    FormatVersion: 3
                            Width: 512
                           Height: 512
                         BitDepth: 12
                        ColorType: 'grayscale'
<<omissis>
                  TransferSyntaxUID: '1.2.840.10008.1.2.1'
<<omissis>
```

Lavorare con file DICOM (3)

```
                        StudyDate: '20040112'
                       SeriesDate: '20040112'
                  AcquisitionDate: '20040112'
                      ContentDate: '20040112'
                        StudyTime: '124558.420000'
                       SeriesTime: '130141.135000'
                  AcquisitionTime: '125813.830496'
                      ContentTime: '125813.830496'
                  AccessionNumber: '97014'
                         Modality: 'CT'
<omissis>
                      PatientName: [1x1 struct]
                        PatientID: '691410'
                 PatientBirthDate: '19530106'
                       PatientSex: 'M'
                       PatientAge: '051Y'
                 BodyPartExamined: 'CHEST'
                   SliceThickness: 1.2500
                              KVP: 140
```

Lavorare con file DICOM (4)

```
<omissis>
                    SamplesPerPixel: 1
          PhotometricInterpretation: 'MONOCHROME2'
                              Rows: 512
                           Columns: 512
                      PixelSpacing: [2x1 double]
                     BitsAllocated: 16
                        BitsStored: 12
                           HighBit: 11
                 PixelRepresentation: 0
                        WindowCenter: [2x1 double]
                         WindowWidth: [2x1 double]
                    RescaleIntercept: -1024
                        RescaleSlope: 1
<omissis>
```

Lavorare con file DICOM (5)

- I singoli metadati sono recuperabili con la usuale sintassi delle strutture:

```
info.Width
```

- Inoltre la variabile contenente i metadati si può usare per leggere l'immagine, anziché dal filename:

```
I = dicomread(info);
```

- E' possibile anche scrivere un'immagine DICOM su disco (`dicomwrite`) e anonimizzarla (`dicomanon`): vedere l'help e gli esempi nella documentazione dell'Image Toolbox.