Bioimmagini – Esercitazione1 - 19 Marzo 2012

Introduction to Matlab IMAGE PROCESSING Toolbox

Matlab:
high-level technical computing language permitting to operate without code compiling. Matrices are the fundamental data type in MATLAB. No a priori dimensioning is required for matrix arrays.

Image Processing Toolbox:
Collection of special-purpose MATLAB functions, which extend the MATLAB environment to solve particular classes of problems in image processing application area.

Image=matrix. Each matrix element corresponds to a *pixel* of the visualized image.

How to read a file containing an image in Matlab?

- Image in Matlab array format (*.mat)

  load filename

- Image in standard graphical image format (*.bmp, *.jpg, *.tiff,…)

  immagine=imread('filename');
  info=imfinfo('filename');

- Image in DICOM (Digital Imaging and Communications in Medicine) format (*.dcm)

  immagine=dicomread('filename');
  info=dicominfo('filename');

- Image in analyze7.5 (*.hdr, *.img) format

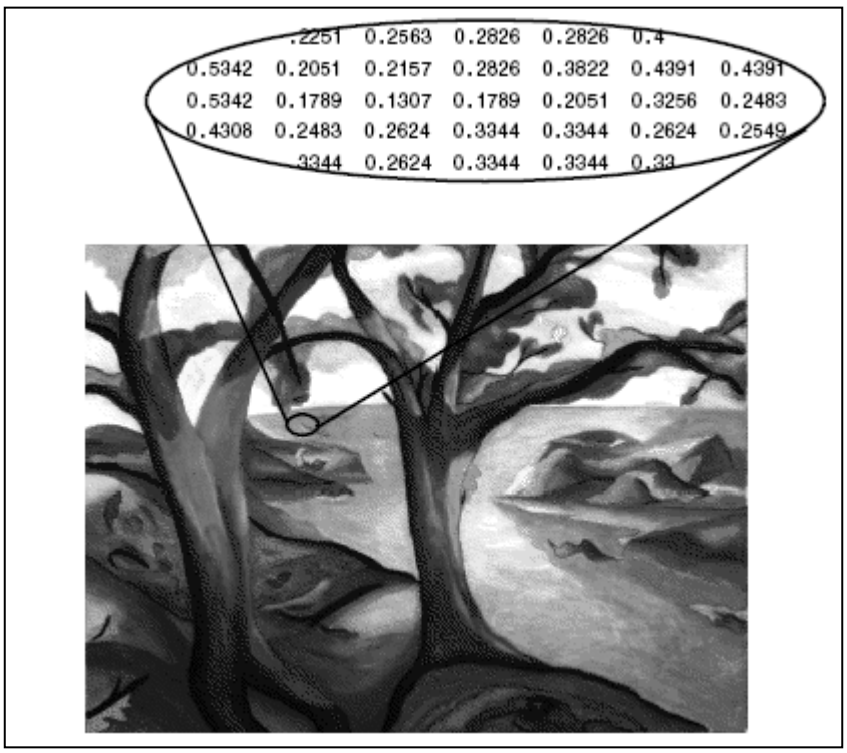
  immagine=analyze75read('filename');
  info=analyze75info('filename');

- Image in other formats: *fopen.m* e *fread.m* functions must be used.

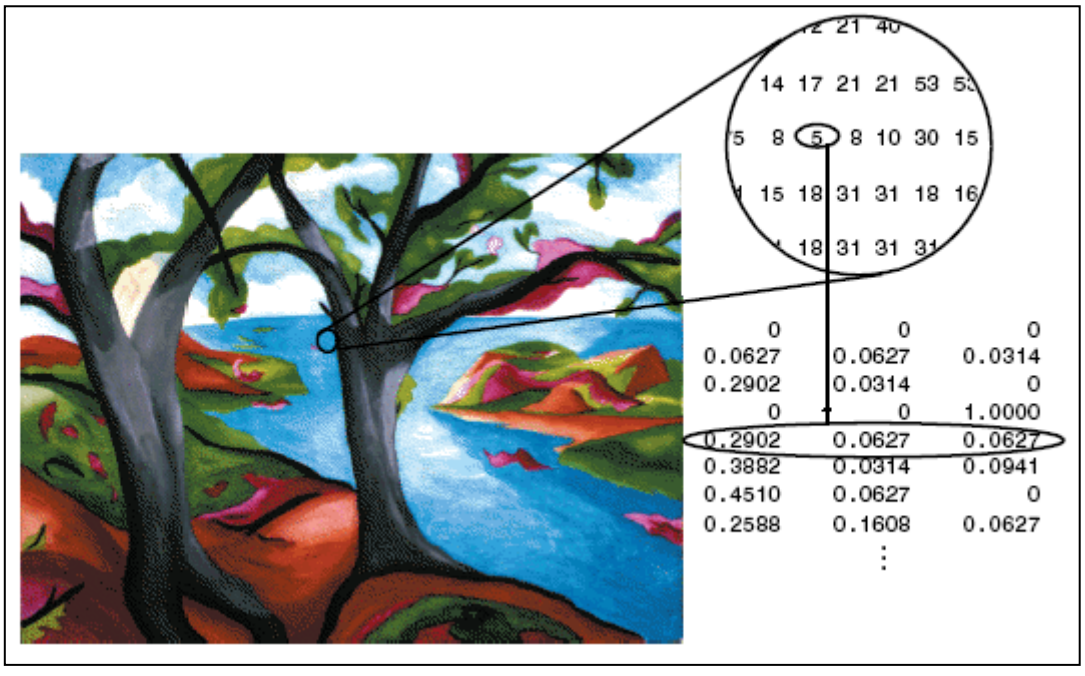
Which image and data types does Matlab support?

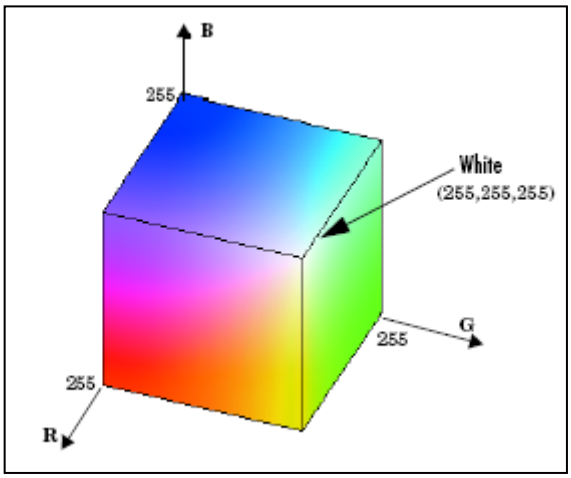
- Intensity-type images (gray levels): single matrix, where each element corresponds to a pixel grey level value.
  Data types:
  - double: real numbers in double precision, 64 bits, range [0 1];
  - uint8: unsigned integer numbers, 8 bits, range [0 255];
  - uint16: unsigned integer numbers, 16 bits, range [0 65535];

- single: real numbers in single precision, 32 bits, range [0 1];
- int8, int16, int32, uint32 (signed and unsigned integer numbers, 8, 16 and 32 bits).

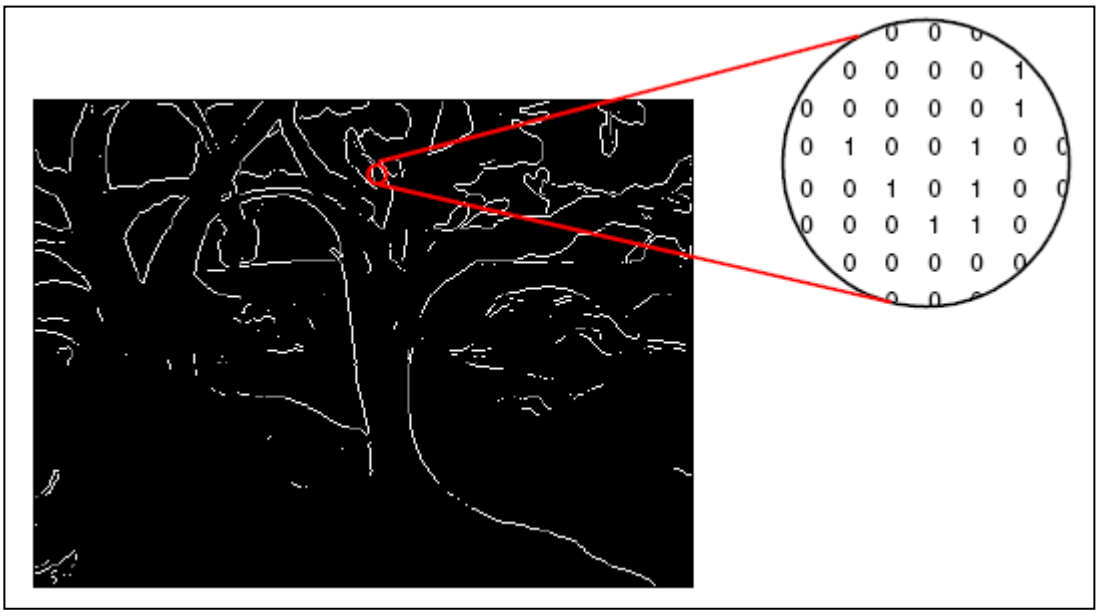


- Indexed-type images: two matrices, one image matrix and one colormap matrix.
  - Image matrix: each element value is the colormap row that indicates the corresponding pixel colour.
  - Colormap: each row of the map (double data type) defines R G B components of a single colour (real numbers from 0 to 1).

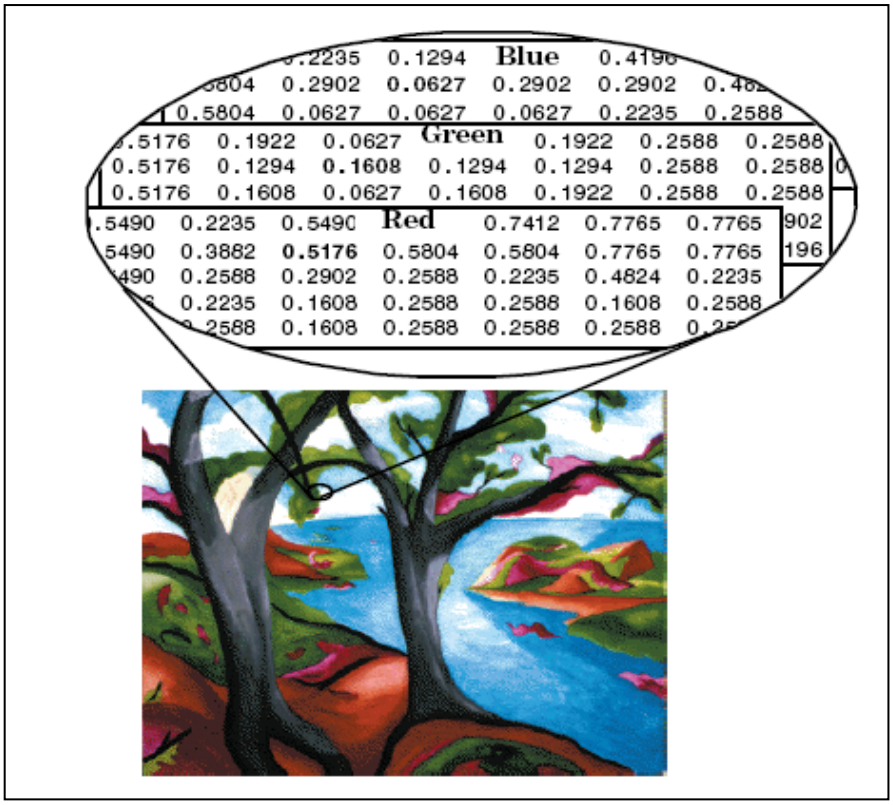




- Binary images: a single matrix where each element can assume only two discrete values: 0 and 1. Data type: logical.



- RGB images: a MxNx3 multidimensional array defining R G B components of each pixel (double, uint8 o uint16 data types).



How to visualize images in Matlab?

- Visualization of a matrix as an intensity-type image:
  imshow(im,[]) (default=*imshow(im,[intmin(class(im))intmax(class(im))]*)

  For intensity-type and indexed images it is possible to change colormap with *colormap* command (help graph3d).
  To visualize more than one image in a single figure, *imshow* together with *subplot* must be used: *subplot* divides a figure into mxn regions which can be visualized by making the selected region active. Example:
  subplot(121),imshow(im1,[]), subplot(122),imshow(im2,[])

- Visualization of a matrix content a along z axis in 3D space:
  mesh(im) or *surf(im)*

  Using *axis* command the axis visualization can be changed:
  axis on/off: shows axis or not
  axis xy: Cartesian reference system
  axis ij :matrix reference system
  axis image: the same scale is used to visualize the tree axis
  axis square: makes the box a square
  axis normal: removes the effects of axis square and axis image

- Visualization of equipotential surfaces:
  imcontour(im)

EXAMPLE 1: load and visualize the image in 'CT.dcm' file.

```
ct=dicomread('CT.dcm');
figure,imshow(ct,[])
```

Select a portion containing the most informative part of the image.

```
impixelinfo
ct_crop=ct(80:336,15:440);
figure,imshow(ct_crop,[])
```

N.B. In searching elements inside an image matrix, the first index refers to rows, while the second one refers to columns (e.g. A(1,3) is the voxel value in row 1 and column 3). In *impixelinfo* function, opposite conventions are used.
In order to extract an image portion it is possible to use also the *imcrop* function, which permits to use the pointer to directly select the region of interest on the figure.

```
close all, figure,imshow(ct,[]),
ct_crop=imcrop;
```

On the original ct image, set the value of pixels in the external portion (which is not of interest) to the minimum vale of the image.

```
minimo=min(ct(:))
cornice=ones(size(ct));
cornice(80:336,15:440)=0;
ct(cornice==1)=minimo;
```

NB: relational operators < <= > >= == ~=

EXAMPLE 2: load and visualize the image contained in 'fant_pet.mat' file.

```
load fant_pet
figure,imshow(fant_pet,[]),colormap(jet),colorbar
figure,imcontour(fant_pet)
```

In order to use the *mesh* function, the image must firstly converted into double format.

```
im=double(fant_pet);
figure,mesh(im)
```

N.B. Specific functions of Image Processing Toolbox exist to convert images from a class to another one: im2double, im2uint8, …

```
max(fant_pet(:))
im1=im2double(fant_pet);        max(im1(:))
im2=double(fant_pet);           max(im2(:))
im3=uint8(im1);                 max(im3(:))
im4=im2uint8(im1);              max(im4(:))
```

EXAMPLE 3: load and visualize the image contained in ‘SPECT.jpg’ file.

```
spect=imread('SPECT.jpg');
figure,imshow(spect,[])
```

Visualize separately the R component of the RGB image.

```
R=zeros(size(spect),'uint8');
R(:,:,1)=spect(:,:,1);
figure,imshow(R)
```

Convert the RGB image to intensity-type image.

```
im=rgb2gray(spect);
figure,imshow(im,[])
```

N.B. Matlab allows to convert image type but in some cases this operation leads to a loss of information and it is not possible to come back to the original content of the image.

```
RGB=ind2rgb(X,MAP);          [X,MAP]=rgb2ind(RGB);
                             I=rgb2gray(RGB);
[X,MAP]=gray2ind(I);         I=ind2gray(X,MAP);
BW=im2bw(RGB,threshold);
BW=im2bw(X,MAP,threshold);
```

EXAMPLE 4: load and visualize the image contained in ‘PET.jpg’ file.

```
pet=imread('PET.jpg');
figure,imshow(pet,[])
```

Try to convert the RGB image to a binary image by varying the threshold.

```
 pet=pet(5:220,20:270,:);
im1=im2bw(pet,0.2);figure,imshow(im1)
im2=im2bw(pet,0.5); figure,imshow(im2)
```

EXAMPLE 5: load the images contained in ‘volume_coeff_atten.mat ‘ file.

```
load volume_coeff_atten
size(vol)
```

Visualize a slice of the 3D array in the axial, coronal and sagittal planes.

```
figure,imshow(vol(:,:,40),[]),colormap(jet)
figure,imshow(rot90(squeeze(vol(75,:,:)),-1),[]);colormap(jet)
figure,imshow(rot90(squeeze(vol(:,75,:)),-1),[]);colormap(jet)
```

N.B. Matlab has functions able to process sequences of 2D images (3D array), that is the same 2D transformation is applied to all images contained into the sequence (e.g. *imfilter* for image filtering).

EXAMPLE 6: load the images contained in 'mri.mat ' file.

```
load mri
figure,imshow(D(:,:,25),map)
```

Visualize the whole image sequence by using *montage* and *immovie* functions.

```
figure,montage(D,map)
mov=immovie(D,map);
figure, movie(mov,3)
```

N.B. *montage* and *immovie* require concatenation of images (frames) along the fourth dimension (multiframe image arrays) Each frame must have the same dimension and, if indexed-type, must use the same colormap.

EXAMPLE 7: write a function able to load the dicom images contained in 'testa_collo' folder and to put them in a 3D array.

```
function volume=leggi_cartella_dicom;
dicomfiles=dir('testa_collo\*.dcm');
Nfiles=length(dicomfiles);
for i=1:Nfiles
Y = dicomread(['testa_collo\',dicomfiles(i).name]);
if i==1
volume=Y;
else
volume=cat(3, volume, Y);
end;
end;
```

Save the obtained image volume into the file 'testa_collo.mat'.

```
save testa_collo volume
```

Linear algebra matrix operations are valid, but pay attention to matrix size.
To perform algebraic operation in an element-by-element manner, use .* ./ .^ .