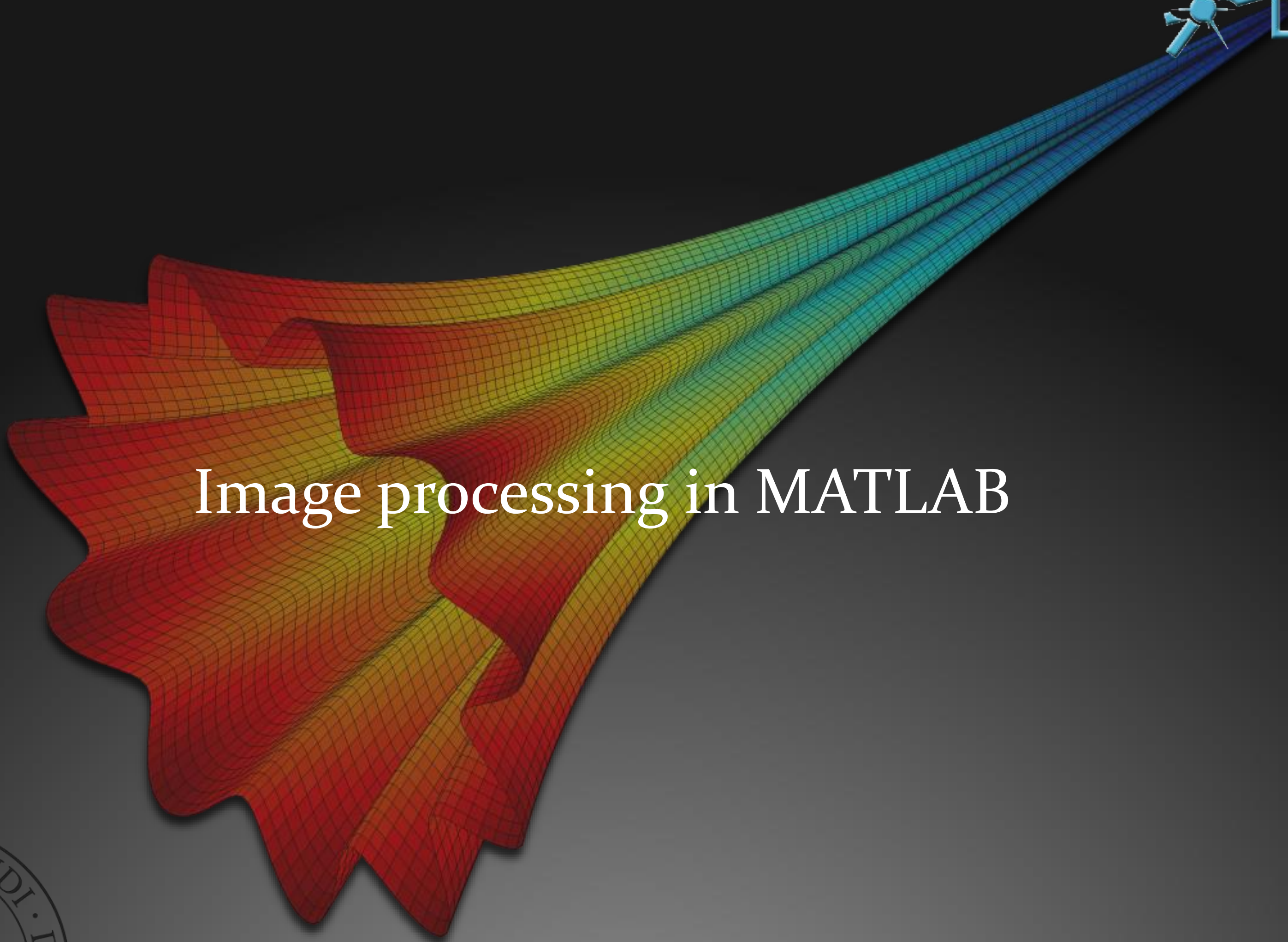


Image processing in MATLAB

Linguaggio Programmazione Matlab-Simulink (2018/2019)

Images in MATLAB

- MATLAB can import/export several image formats
  - BMP (Microsoft Windows Bitmap)
  - GIF (Graphics Interchange Files)
  - HDF (Hierarchical Data Format)
  - JPEG (Joint Photographic Experts Group)
  - PCX (Paintbrush)
  - PNG (Portable Network Graphics)
  - TIFF (Tagged Image File Format)
  - XWD (X Window Dump)
  - MATLAB can also load raw-data or other types of image data
  - DICOM data (dicomread…)

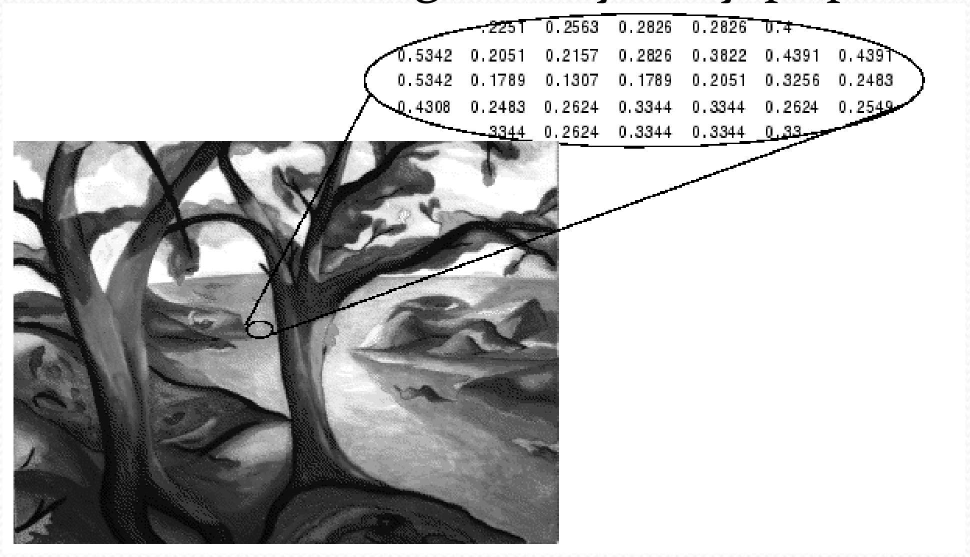
- Data types in MATLAB
  - Double (64-bit double-precision floating point)
  - Single (32-bit single-precision floating point)
  - Int32 (32-bit signed integer)
  - Int16 (16-bit signed integer)
  - Int8 (8-bit signed integer)
  - Uint32 (32-bit unsigned integer)
  - Uint16 (16-bit unsigned integer)
  - Uint8 (8-bit unsigned integer)

Images and Colors

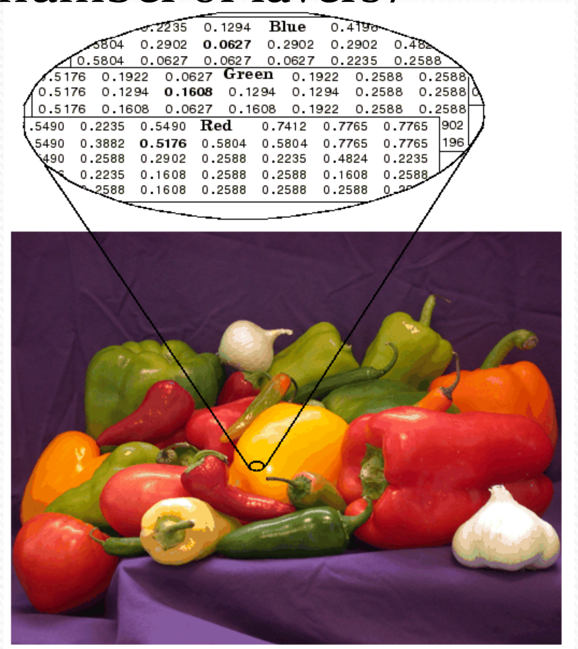
- Images are represented as grids, or matrices, of picture elements (called "***pixels***")
- In MATLAB an image is represented as an *m x n* matrix in which each element corresponds to a pixel
- All the operators in MATLAB defined on matrices can be used on images: +, -, *, /, ^, sqrt, sin, cos etc
- Each element that represents a particular pixel stores the color for that pixel

Images in MATLAB

- Binary images : {0,1}
- Intensity images : [0,1] or uint8, double etc.
- RGB images : m-by-n-by-3 (one for each color channel)
- Indexed images : m-by-n matrix, p-by-3 color map
- Multidimensional images m-by-n-by-p (p is the number of lavers)



```
        .2251   0.2563  0.2826  0.2826  0.4
0.5342  0.2051  0.2157  0.2826  0.3822  0.4391  0.4391
0.5342  0.1789  0.1307  0.1789  0.2051  0.3256  0.2483
0.4308  0.2483  0.2624  0.3344  0.3344  0.2624  0.2549
        .3344   0.2624  0.3344  0.3344  0.33
```

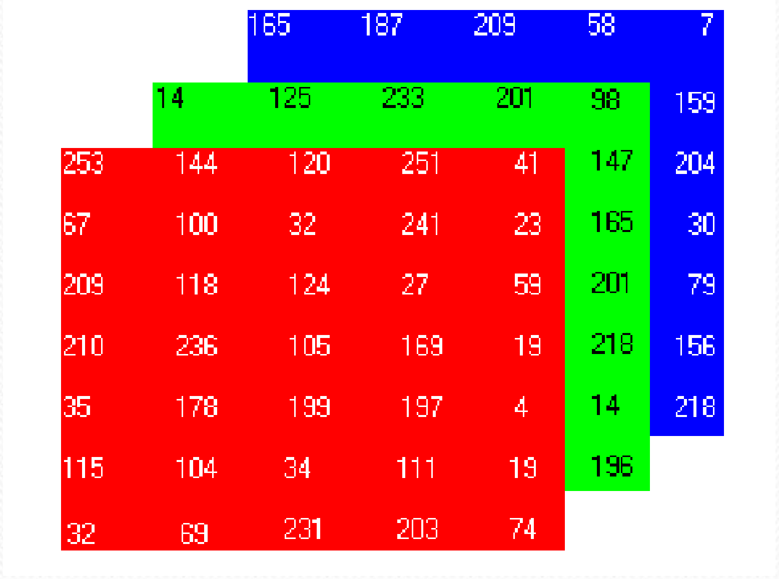


```
          .2235   0.1294   Blue     0.4196
   .5804  0.2902  0.0627   0.2902   0.2902   0.482
   0.5804 0.0627  0.0627   0.0627   0.2235   0.2588
 .5176   0.1922   0.0627   Green    0.1922   0.2588   0.2588
 0.5176  0.1294   0.1608   0.1294   0.1294   0.2588   0.2588
 0.5176  0.1608   0.0627   0.1608   0.1922   0.2588   0.2588
.5490   0.2235   0.5490   Red      0.7412   0.7765   0.7765   902
5490    0.3882   0.5176   0.5804   0.5804   0.7765   0.7765   196
490     0.2588   0.2902   0.2588   0.2235   0.4824   0.2235
        0.2235   0.1608   0.2588   0.2588   0.1608   0.2588
        .2588    0.1608   0.2588   0.2588   0.2588   0.2
```

Representing Color

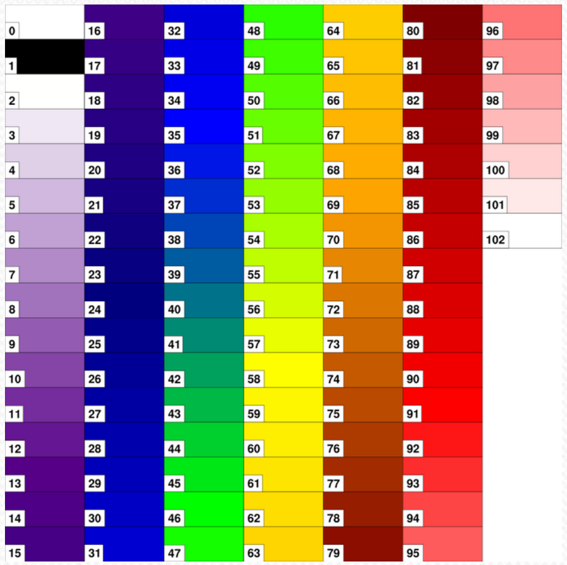
- There are two basic ways that the color can be represented for a pixel:
  - ***true color***, or ***RGB***, in which the three color components are stored (red, green, and blue, in that order)
  - → the matrix is

m x n x 3

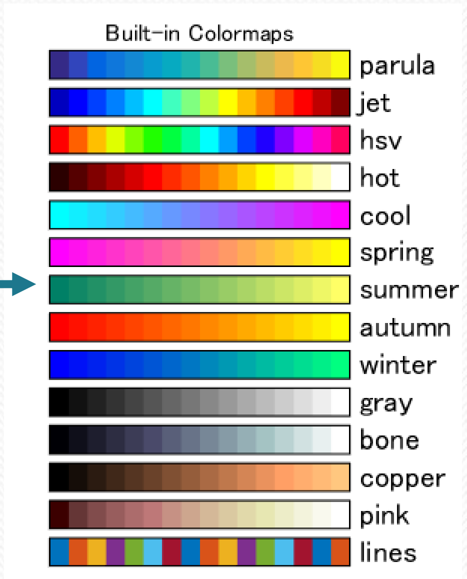


Representing Color

- index into a ***colormap***: the value stored in each element of the *m x n* matrix is an integer which refers to a row in another matrix which is called a colormap
  - The colormap stores the red, green and blue components in three separate columns so its size is *p x 3* where p is the number of colors



Matlab pre-defined colormaps



Colormaps

- every element in the *m x n* image matrix *mat* is an integer in the range from 1 to p which refers to a row in another matrix which is called a colormap
- The colormap stores the red, green and blue components of different colors in three separate columns so its size is *p x 3* where p is the number of colors
- If the type of the elements is **double**, the numbers range from 0 (which represents the absence of that color) to 1 (which is the brightest possible hue)
- the **image** function displays the image matrix using the current colormap:

```
    image(mat)
```

- the **colormap** function can be used two ways: if no argument is passed, it returns the current colormap; if a p x 3 matrix is passed, it sets that matrix to be the current colormap

Reading an image

>>A = imread(filename) %reads the image from the file specified by filename, inferring the format of the file from its contents.
>>A = imread(filename,fmt) %additionally specifies the format of the file with the standard file extension indicated by fmt. If imread cannot find a file with the name specified by filename, it looks for a file named filename.fmt.
>>[A,map] = imread(___) %reads the indexed image in filename into A and reads its associated colormap into map. Colormap values in the image file are automatically rescaled into the range [0,1].

```
>>[X,map,alpha] = imread('peppers.png');
>>whos alpha
  Name       Size            Bytes  Class     Attributes

  alpha      0x0                 0  double

%No alpha channel is present, so alpha is empty.
```

Using imread

- The function reads color images into a 3D matrix

  >> myimage1 = imread('xyz.JPG');

- Functions now in MATLAB (were in Image Processing Toolbox):
  - **imshow** displays an image
  - **rgb2gray** converts from RGB to gray scale
  - **im2double** converts an image matrix to **double**

Showing an image (1)

```
>>image(C)
```

%displays the data in array C as an image. Each element of C specifies the color for 1 pixel of the image. The resulting image is an m-by-n grid of pixels where m is the number of rows and n is the number of columns in C.

```
>>image(x,y,C)
```

%x,y specifies the image location. Use x and y to specify the locations of the corners corresponding to C(1,1) and C(m,n). To specify both corners, set x and y as two-element vectors.

```
>>imagesc(C) %similar with image
```

Observation: these functions uses the current colormap

```
>> x = [5 8];
>> y = [3 6];
>> C = [0 2 4 6; 8 10 12 14; 16 18 20 22];
>> image(x,y,C);
>> image(C);
>> imagesc (x,y,C)
```

Showing an image (2)

```
>>imshow(I)
%displays the grayscale image I in a figure. imshow optimizes figure, axes, and
image object properties for image display.
>>imshow(RGB) %displays the truecolor image RGB in a figure.
>>imshow(I,[low high]) %displays the grayscale image I, specifying the display
range as a two-element vector, [low high]
>>imshow(I,[]) %displays the grayscale image I, scaling the display based on
the range of pixel values in I. imshow uses [min(I(:)) max(I(:))] as the display
range.
>>imshow(X,map) %displays the indexed image X with the colormap map.
```

Observation: this function uses the colormap specified by ‘map’. Otherwise a m-by-n image will be shown as a grayscale image, while a m-by-n-by-3 will be shown in RGB.

Built-in colormaps

- There are several built-in colormaps, e.g. **parula** (which is the default), **jet**, **autumn**, **pink**, etc.
- All of these all have 64 colors so the size of the colormap is 64 x 3
- For example, the first four colors from **jet** are shades of blue:

```
>> colormap(jet)
>> colormap
ans =
      0      0   0.5625
      0      0   0.6250
      0      0   0.6875
      0      0   0.7500
           etc.
```

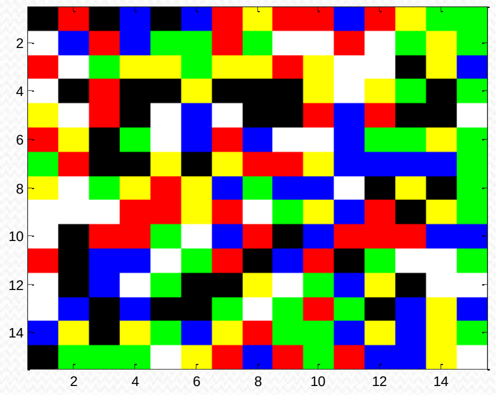
```
>> colormap default
>> size(colormap)

ans =

    64     3
>> mat = randi([1 64],15,15);
>> imagesc(mat)
>> colormap(jet);
>> size(jet);
>> imagesc(mat);
```

Example user-defined colormap

Create a colormap with 6 colors, and use a random matrix to display them

- [1 0 0] is red
- [0 1 0] is green
- [0 0 1] is blue
- [1 1 0] is yellow
- [0 0 0] is black
- [1 1 1] is white

```
>> mycolors = [1 0 0; 0 1 0; 0 0 1;...
 1 1 0; 0 0 0; 1 1 1];
>> colormap(mycolors)
>> mat = randi([1 6],15,15);
>> imagesc(mat)
>> colorbar
```



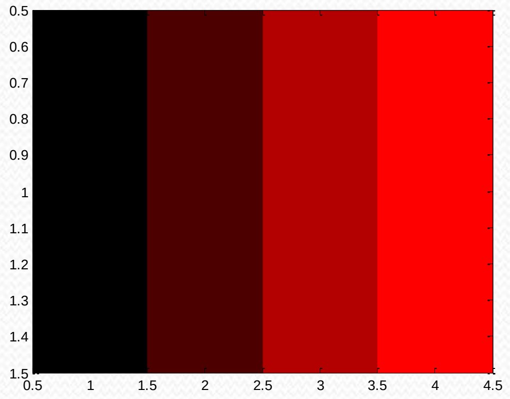
Example indexed image

```
>> [X,map] = imread('trees.tif');
>> size(X)
ans =
   258   350
>> size(map)
ans =
   256     3
>> image(X)
>> colormap summer
>> image(X)
>> imshow(X,map)
```

colormap shades

- The numbers in the colormap do not have to be integers; real numbers represent different shades
- The following shows four shades of red, from no red (black) to the brightest possible red:

```
>> colors = [0 0 0; 0.3 0 0; 0.7 0 0; 1 0 0];
>> colormap(colors)
>> vec = 1:4;
>> image(vec)
```



True Color Matrices

- True color matrices are three-dimensional matrices with size *m x n x 3*
  - The first two indices are the coordinates of the pixel
  - The third index is the color component; (:,:,1) is the red, (:,:,2) is the green, and (:,:,3) is the blue component
- If every element in the matrix is of the type **uint8**, which is an unsigned integer type using 8 bits; the range of values is 0 to 255
- Therefore the minimum value, 0, represents the darkest hue available so all 0' s results in a black pixel
- The maximum value, 255, represents the brightest hue
- For example, in the following the pixel in location 1,1 will be red:

```
>> mat(1,1,1) = 255;
>> mat(1,1,2) = 0;
>> mat(1,1,3) = 0;
```

- The image function shows the image:

```
>> mat = uint8(mat);
>> image(mat)
```

True Color Matrices

- Display Image of 3-D Array of True Colors

```
>>C = zeros(3,3,3);
C(:,:,1) = [.1 .2 .3; .4 .5 .6; .7 .8 .9]
C =
C(:,:,1) =

   0.1000   0.2000   0.3000
   0.4000   0.5000   0.6000
   0.7000   0.8000   0.9000
C(:,:,2) =
    0     0     0
    0     0     0
    0     0     0
C(:,:,3) =

    0     0     0
    0     0     0
    0     0     0
```

- Display an image of the data in C: >>image(C)

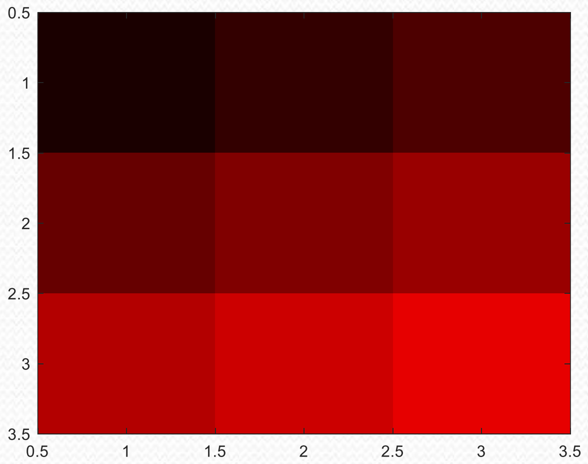


Image import and export example

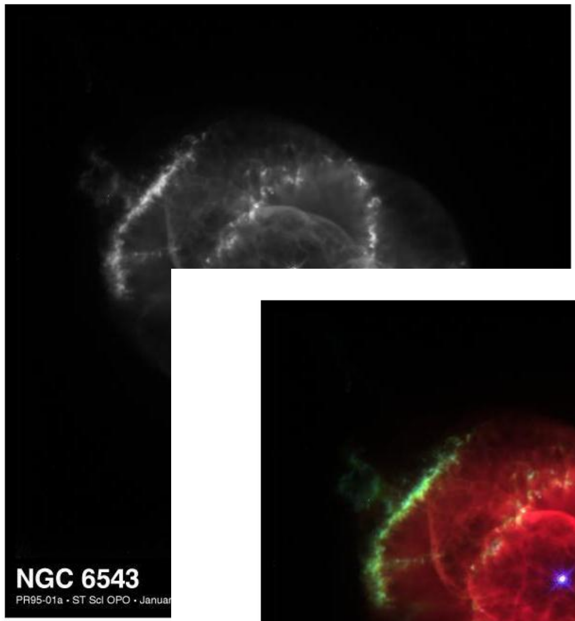
- Read and write images in Matlab

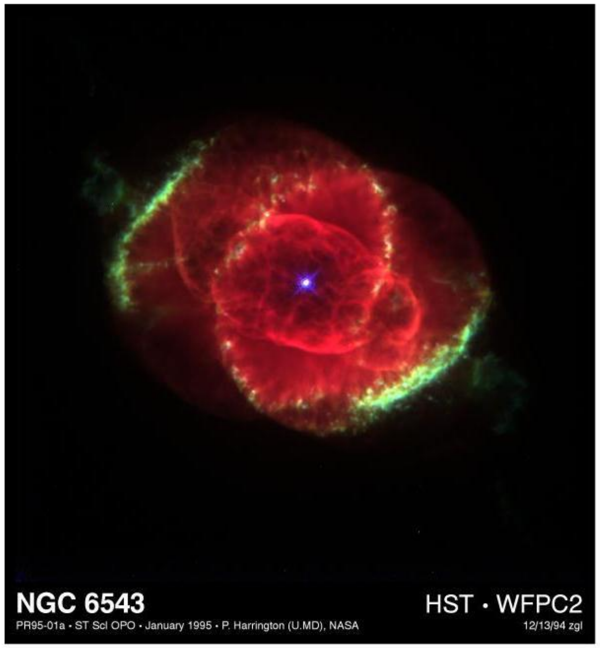
```
>> I=imread('ngc6543a.jpg');
>> imshow(I)
>> size(I)
ans =    479   600     3              (RGB image)
>> Igrey=rgb2gray(I);
>> imshow(Igrey)
>> imwrite(lgrey, 'cell_gray.tif', 'tiff')
```

- Alternatives to imshow

```
>>imagesc(I)
>>imtool(I)
>>image(I)
```

- Esercizio: trasformare un Imagine colormap in RGB

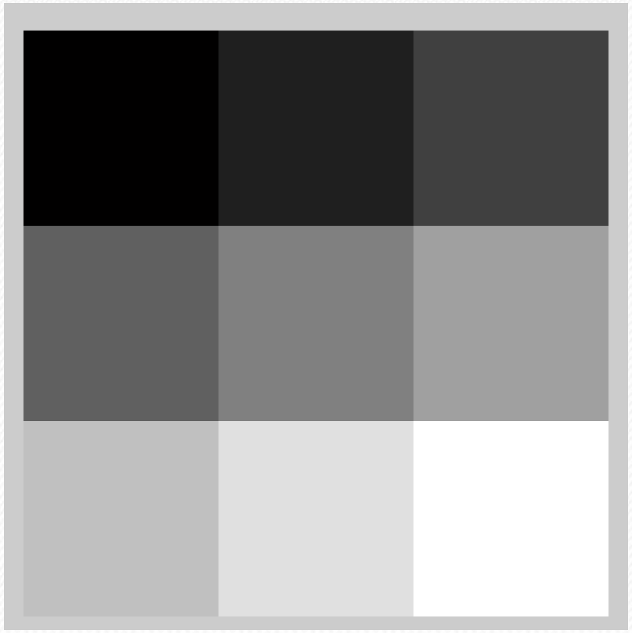



NGC 6543
PR95-01a · ST ScI OPO · January 1995 · P. Harrington (U.MD), NASA
HST · WFPC2
12/13/94 zgl

Images and Matrices

- Building matrices (or images)

```
>> A = [ 1 2 3; 4 5 6; 7 8 9 ];
 A =  1    2    3
      4    5    6
      7    8    9
>> B = zeros(3,3)
 B =  0    0    0
      0    0    0
      0    0    0
>> C = ones(3,3)
C =   1    1    1
      1    1    1
      1    1    1
```



```
>>imshow(A)      (imshow(A,[]) to get automatic pixel range)
```

Images and Matrices

- Accesing image elements (row, column)

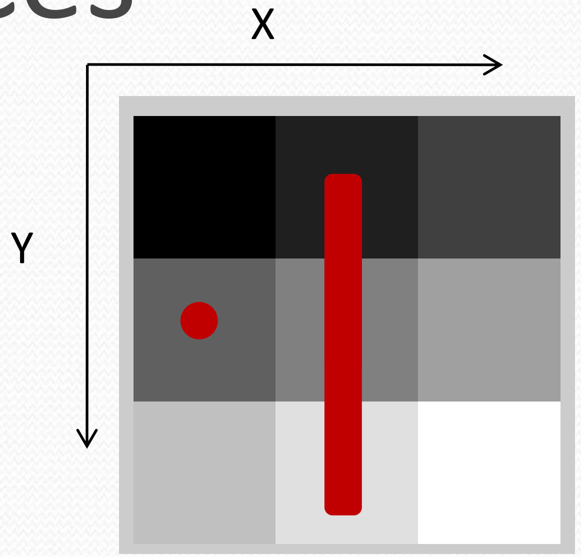
```
>> A(2,1)
ans = 4
```

- : can be used to extract a whole column or row

```
>> A(:,2)
ans =
2
5
8
```

- or a part of a column or row

```
>> A(1:2,2)
ans =
2
5
```

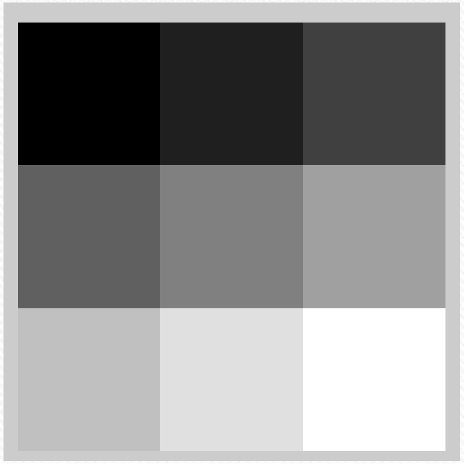


```
A =
     1     2     3
     4     5     6
     7     8     9
```

Image Arithmetic

- Arithmetic operations such as addition, subtraction, multiplication and division can be applied to images in MATLAB
  - +, -, *, / performs matrix operations

```
 >> A+A
ans =     2    4    6
          8   10   12
         14   16   18
 >> A*A
ans =    30   36   42
         66   81   96
        102  126  150
```



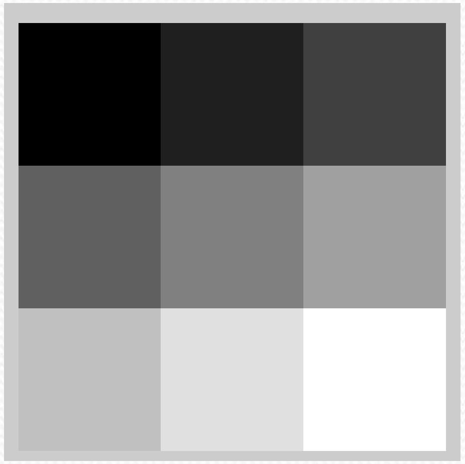
- To perform an elementwise operation use . (.*, ./, .*, .^ etc)

```
>> A.*A
ans =     1    4    9
         16   25   36
         49   64   81
```

Logical Conditions

- equal (==) , less than and greater than (< and >), not equal (~=) and not (~)
- find('condition') - Returns indexes of A's elements that satisfies the condition.

```
>> [row col]=find(A==7)
row =   3
col =    1
>> [row col]=find(A>7)
row =   3
          3
col =    2
          3
>> Indx=find(A<5)
Indx =  1
        2
        4
        7
```



```
A =
     1     2     3
     4     5     6
     7     8     9
```

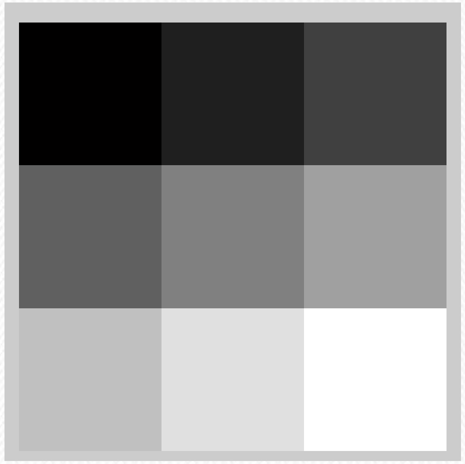
Flow control

- Flow control in MATLAB

  - if, else and elseif statements

  (row=1,2,3   col=1,2,3)

```
if row==col
        A(row, col)=1;
elseif abs(row-col)==1
        A(row, col)=2;
else
        A(row, col)=0;
end
```

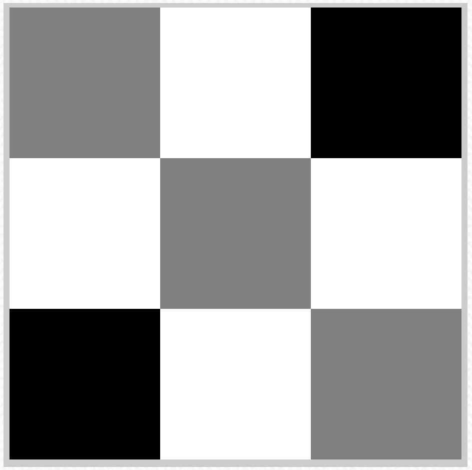


```
A =
    1    2    3
    4    5    6
    7    8    9
```

Flow control

- for loops

```
for row=1:3
        for col=1:3
                if row==col
                        A(row, col)=1;
                elseif abs(row-col)==1
                        A(row, col)=2;
                else
                        A(row, col)=0;
                end
        end
end
```



```
A =

     1     2     0
     2     1     2
     0     2     1
```

Image histogram

- The histogram plots the number of pixels in the image (vertical axis) with a particular brightness value (horizontal axis).

- Algorithms in the digital editor allow the user to visually adjust the brightness value of each pixel and to dynamically display the results as adjustments are made. Improvements in picture brightness and contrast can thus be obtained.

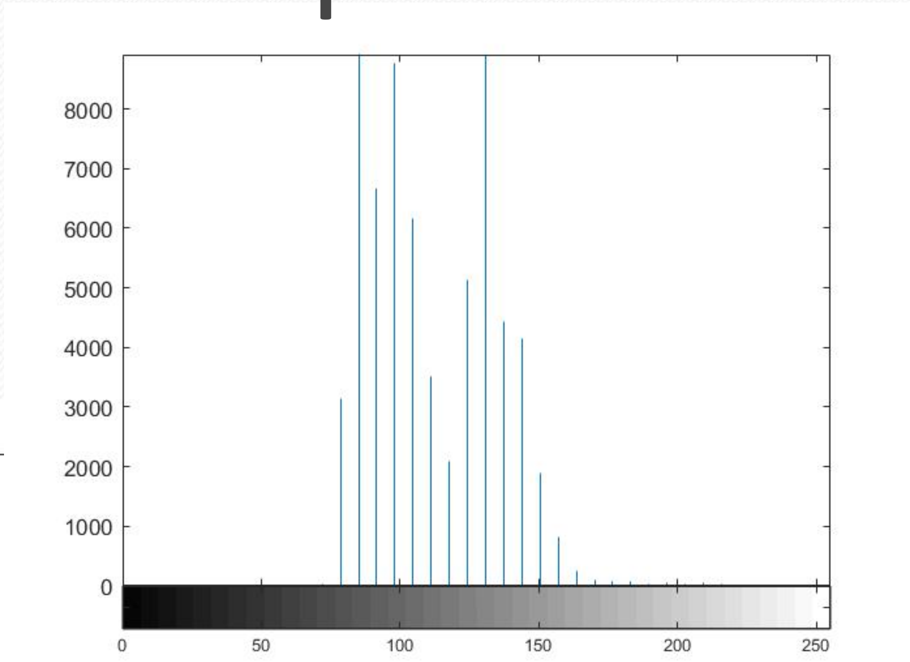
- In the field of computer vision, image histograms can be useful tools for thresholding. Because the information contained in the graph is a representation of pixel distribution as a function of tonal variation, image histograms can be analyzed for peaks and/or valleys. This threshold value can then be used for edge detection, image segmentation, and co-occurrence matrices. (Wikipedia)

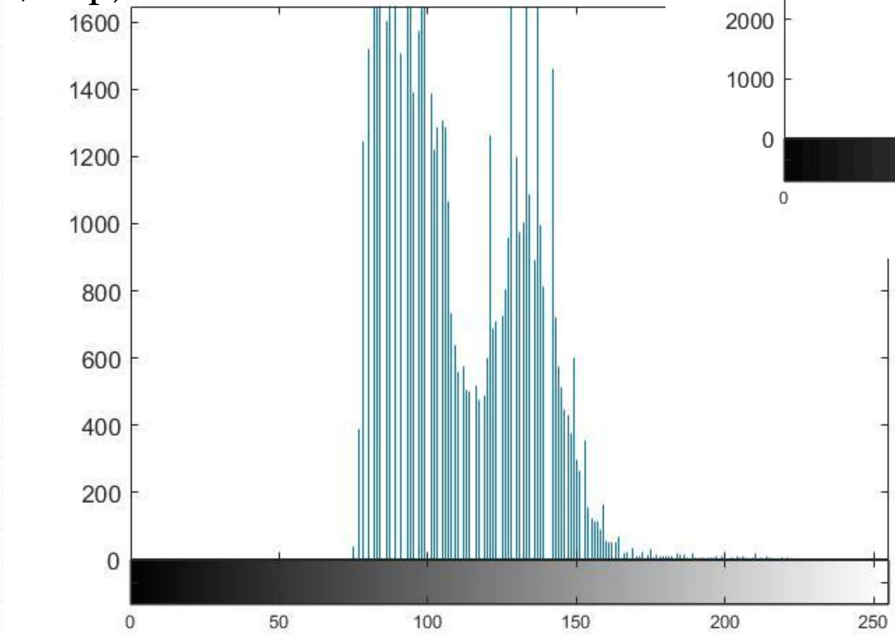
Image histogram

- imhist(I) calculates the histogram for the intensity image I and displays a plot of the histogram. The number of bins in the histogram is determined by the image type.
- imhist(I,n) calculates the histogram, where n specifies the number of bins used in the histogram.
- imhist(X,map) calculates the histogram for the indexed image X with colormap map. The histogram has one bin for each entry in the colormap.
- Syntax:
  - [counts,binLocations] = imhist(I)
  - [counts,binLocations] = imhist(I,n)
  - [counts,binLocations] = imhist(X,map)

Image histogram example

```
>>I = imread('pout.tif');
>>imhist(I)
>> imhist(I,40)

>>[X,map] = imread('trees.tif');
>>imshow(X,map)
>>imhist(X,map)
```





Exercises

- Given I = imread('pout.tif’);

Use the ‘histeq’ function to increase contrast. Represent the image before and after histogram equalization.

- Use histogram to divide an image in ‘Backround’ and ‘Foreground’. To do this, choose an image from internet and read the image directly from Matlab command line. Try the Otsu method
- E.g

```
>>I=imread(‘https://upload.wikimedia.org/wikipedia/commons/4/4b/Image_processing_pre_otsus_algorithm.jpg’);
>>imshow(I)
```