Quick primer on image processing in Matlab

Spatial domain processing

Credits: Giacomo Boracchi October 1st, 2018

Edits: Luca Magri, September 2023, for comments and suggestions write to luca.magri@polimi.it

```
close all
clear
```

Images are matrices

```
im = imread('E1_data/image_Lena512.png');
whos im
```

```
  Name        Size              Bytes  Class    Attributes

  im        512x512            262144  uint8
```

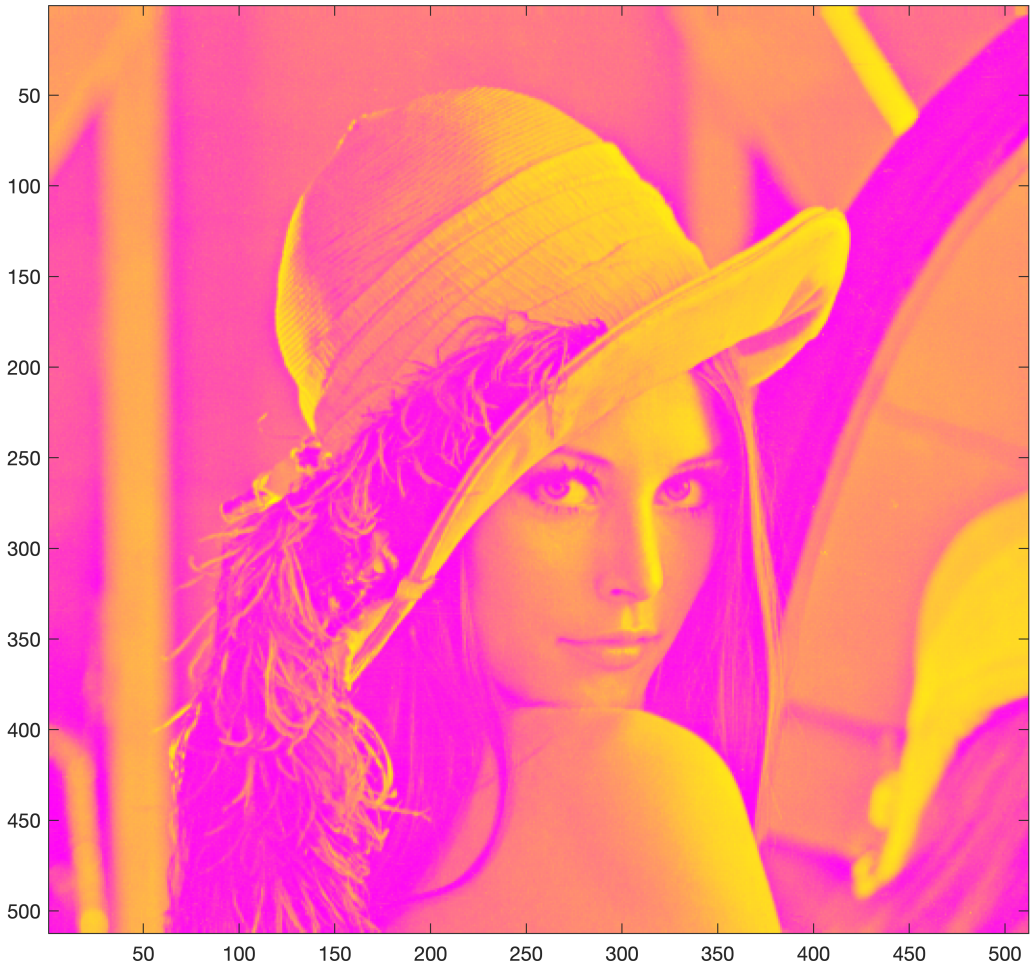
images can be displayed:

- imshow preserve aspect ratio and uses grayscale colormap for 2D images

```
imshow(im);
```

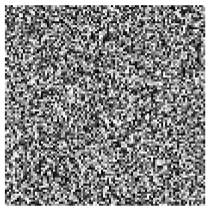
- imagesc does not preser aspect ratio and can use arbitrary colormaps

```
imagesc(im), colormap spring
```



Matrices are images

```
M = rand(100,100);
imshow(M);
```



... but datatype matters! Careful about data range: [0 1] for double and logical, [0 255] for uint8

```
imshow(double(im)); %An image of double in [0,255]
```

rescaling is necessary when casting to double images

```
imshow(double(im)/255); % An image of double in [0,1]
```

```
imshow(uint8(double(im)/255)); % An image of uint8 in [0,1]
```

plot the image histogram

compute histogram of image intensity values

```
h = hist(im(:), [0: 255]);
```

Plot the histogram

```
figure(2)
stairs([0: 255], h, 'b', 'LineWidth', 3)
title('intensity histogram')
axis tight
```

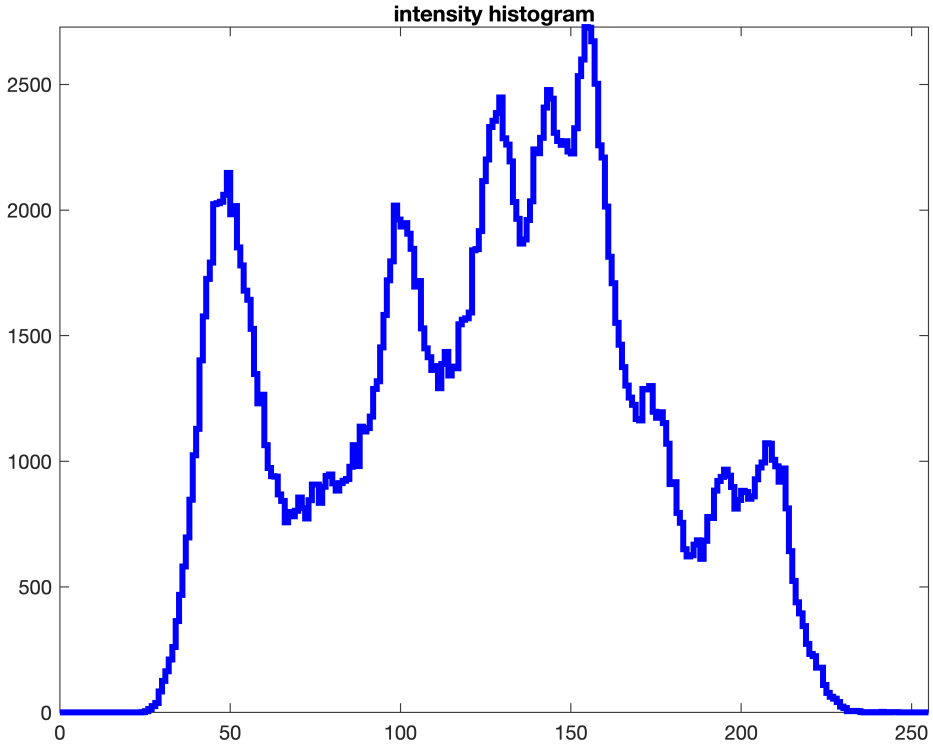
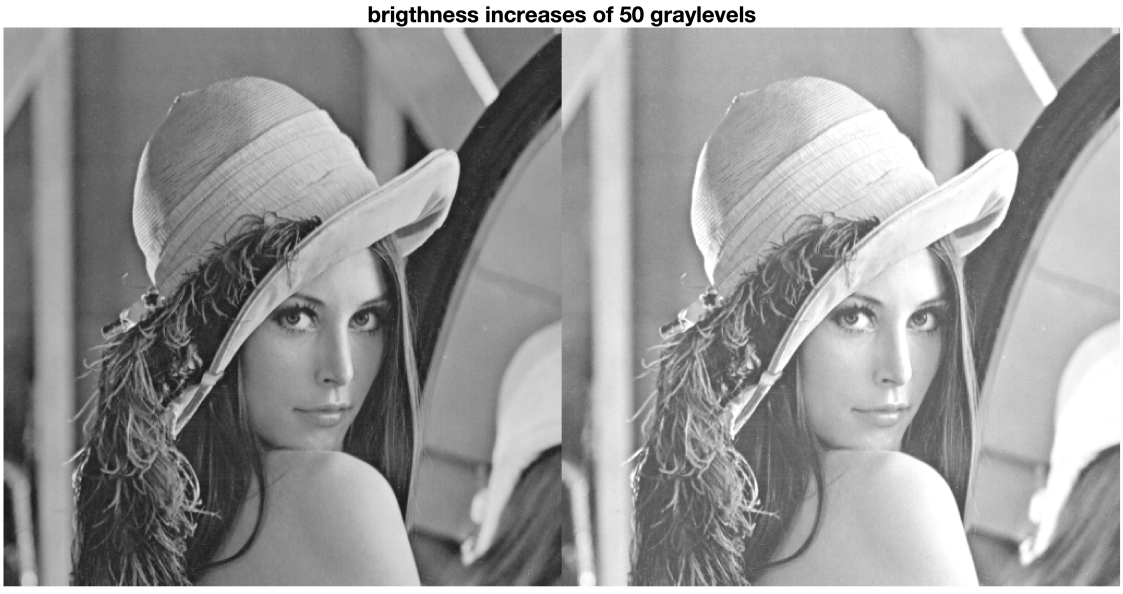


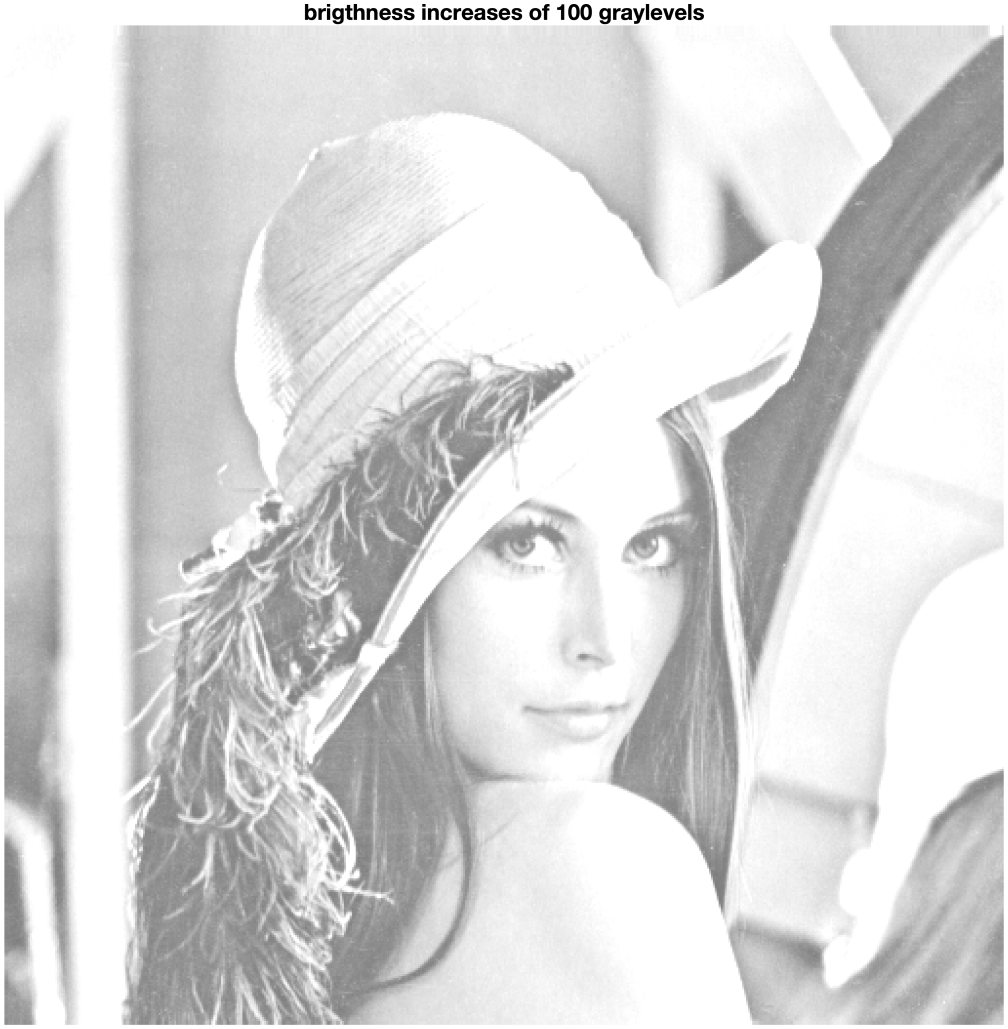
Image brigthness can be modified by adding an offset to each pixel

```
figure(1), imshow([im, im + 50]), title('brigthness increases of 50 graylevels');
```



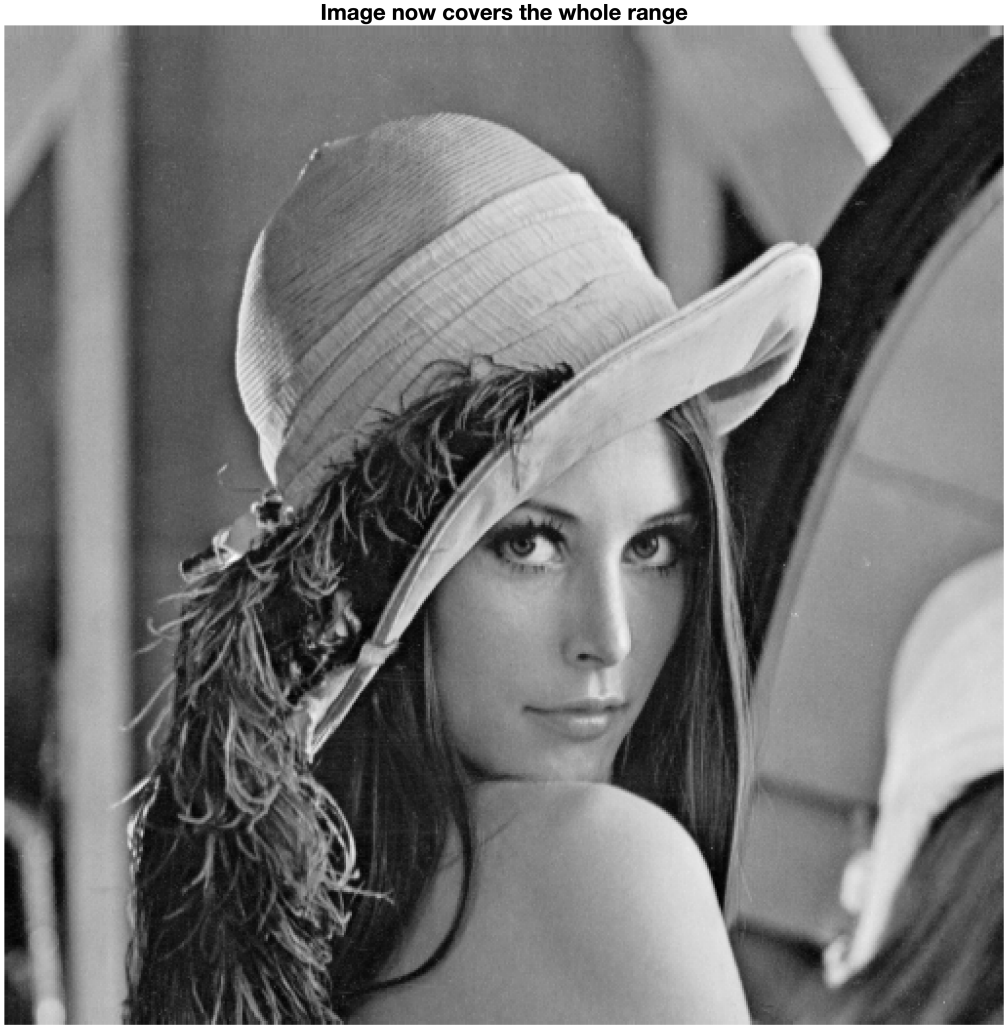

```
% few pixels are saturated
figure(1), imshow(im + 100), title('brigthness increases of 100 graylevels');
```

brigthness increases of 100 graylevels

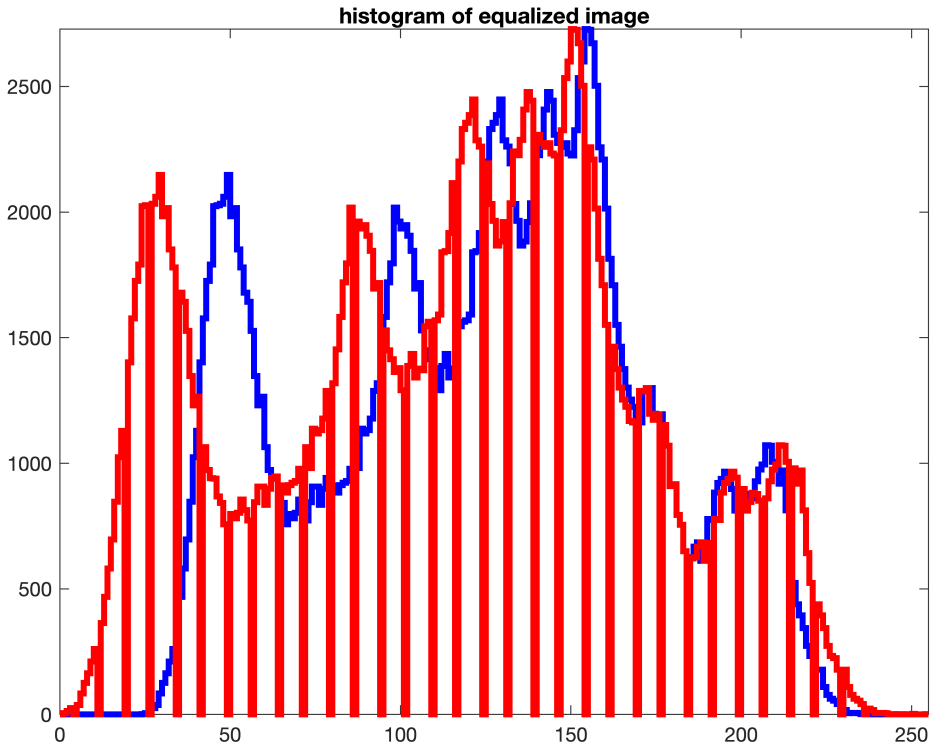


```
% contrast can be modified by streching the histogram, image has to be
% converted in double format
eq = double(im - min(im(:)))/double(max(im(:)) - min(im(:))) * 255;

figure(1), imshow(uint8(eq), []), title('Image now covers the whole range');
```

Image now covers the whole range



```
heq = hist(eq(:), [0: 255]);
figure(2),
stairs([0: 255], h, 'b-', 'LineWidth', 3), title('histogram of original image')
hold on;
stairs([0: 255], heq, 'r-', 'LineWidth', 3), title('histogram of equalized image')
axis tight
```

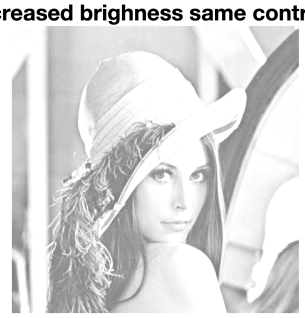


The equalized image (in red) occupy the whole range [0,255]

Image ranges (in the visualization) can be also controlled by the second argument to imshow

```
figure;
subplot(2,2,1)
imshow(im,[-100 156]);
title('increased brighness same contrast');

subplot(2,2,2)
imshow(im,[0 156]);
title('increased brighness and contrast');

subplot(2,2,3)
imshow(im,[ 100 256]);
title('decreased brighness, increased contrast');

subplot(2,2,4)
imshow(im,[ 100 356]);
title('decreased brighness, same contrast');
```

increased brighness same contrast



increased brighness and contrast

decreased brighness, increased contrast

decreased brighness, same contrast

Consider that brightness and contrast can be also changed by

- adding an offset

- scaling the image

Color is represented by 3 channels (RGB)

You get a 3d matrix

```
im=imread('E1_data/bluecube.jpg');
whos im
```

```
  Name         Size                 Bytes  Class    Attributes

  im        1417x1417x3           6023667  uint8
```

```
figure(45)
imshow(im);
```

this is a 3D matrix, because it has 3 dimensions (not because the 3rd dimension has only 3 elements).

```
test=im(:,:,1:2);
whos test
```

```
  Name         Size                 Bytes  Class    Attributes

  test      1417x1417x2           4015778  uint8
```

We can instead see each single channel as a grayscale image

```
imr=im(:,:,1);
whos imr % note: now it's 2D, so it will be displayed as grayscale
```

```
  Name         Size                 Bytes  Class    Attributes

  imr       1417x1417            2007889  uint8
```

```
imshow(imr), title('red channel')
```


```
img=im(:,:,2);
imshow(img), title('green channel')
```

green channel

```
imb=im(:,:,3);
imshow(imb), title('blue channel')
```

blue channel

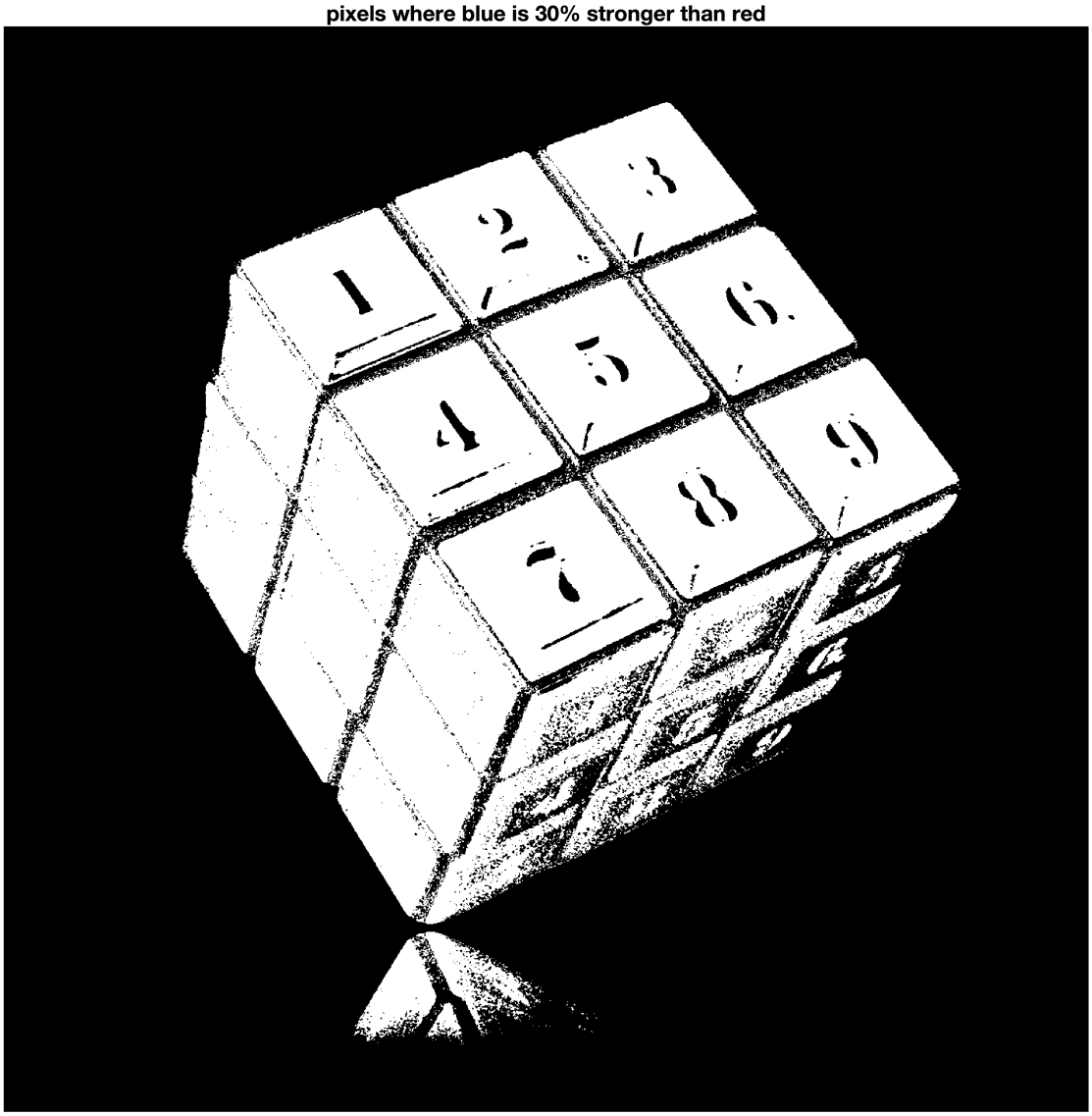
Logicals

We can plot the result of logical operations

```
l = imb > (1.3 * imr);
whos l
```

```
  Name         Size                Bytes  Class      Attributes

  l         1417x1417            2007889  logical
```

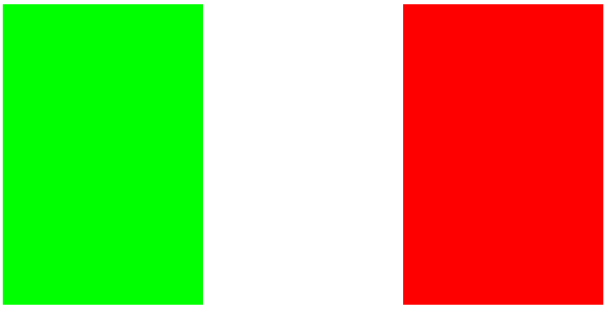
```
imshow(l); title('pixels where blue is 30% stronger than red');
```



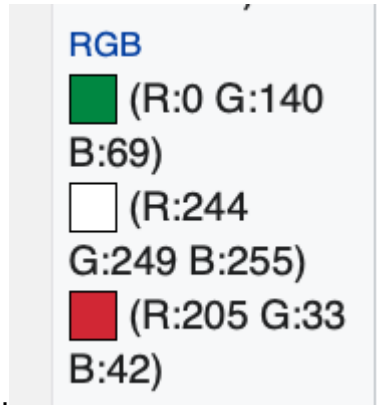
TODO: Display the italian flag

```
C = 300;
R = 150;

flag = zeros(R, C, 3);

% GREEN CHANNEL
g  = zeros(R, C);
g(:, 1 : C/3) = 255;
g(:, C/3 +1 : 2 * C /3) = 255;
g(:, 2 * C /3 + 1: C) = 0;

% BLUE CHANNEL
b  = zeros(R, C);
b(:, C/3 +1 : 2 * C/3) = 255;
```

```
% RED CHANNEL
r  = zeros(R, C);
r(:, 1 : C/3) = 0;
r(:, C/3 +1 : end) = 255;


flag(:,:,1) = r;
flag(:,:,2) = g;
flag(:,:,3) = b;

figure, imshow(flag)
```

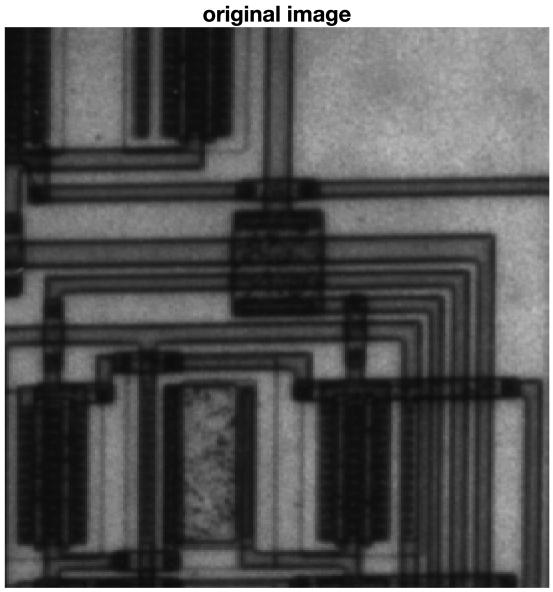


Note: these are not exactly the official colors of the italian flag. If you want a better result, you can check for the official palette.

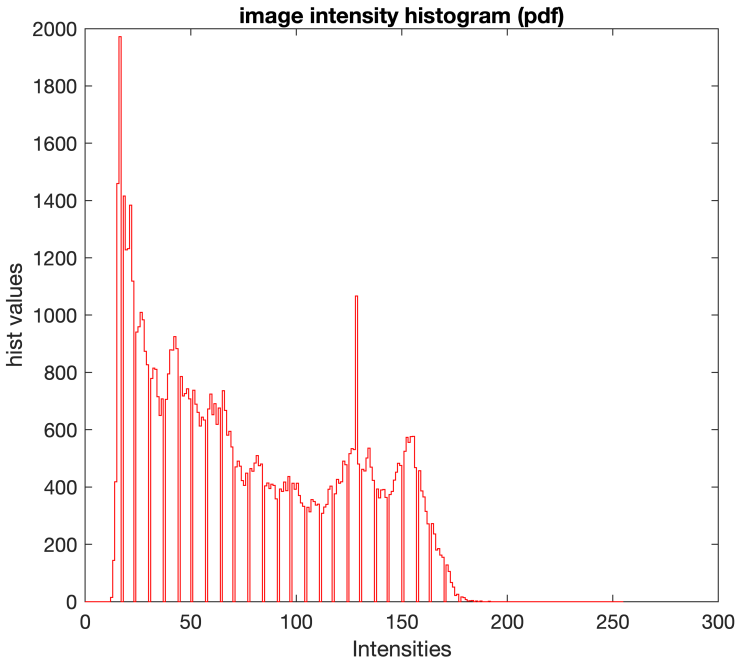


Histogram transformation

```
im = imread('circuit.tif');
figure, imshow(im), title('original image');
```




```
% compute the histogram of all the intensities
hist_im = hist(im(:), [0 : 255]);
figure(5),
stairs([0 : 255], hist_im, 'r'), title('image intensity histogram (pdf)')
xlabel('Intensities');
ylabel('hist values');
```

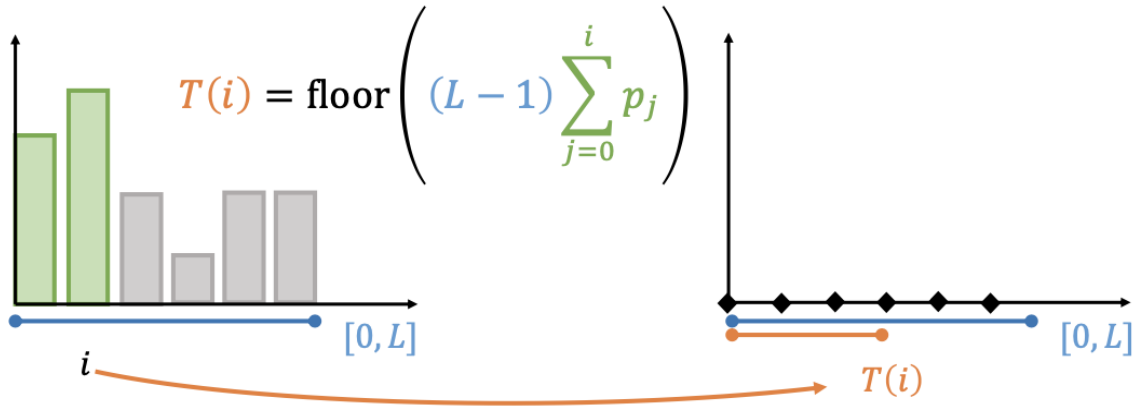


Histogram equalization

```
% the histogram can be computed also as
% imhist(im);
```

The histogram can be seen as the pdf of a RV corresponding to pixel realization histogram equalization consists in applying a monotonic transformation that brings this pdf towards a uniform distribution.

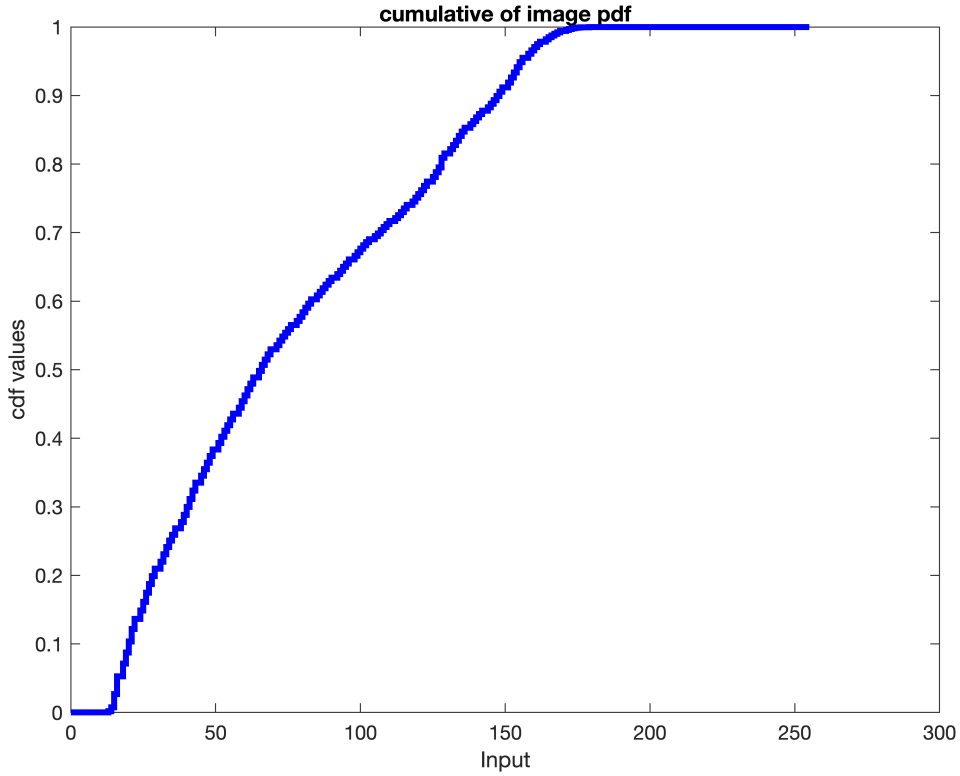
The trasformation is $T : i \mapsto \text{floor}((L-1)\text{CDF}(\text{hist}, i))$ where L are the available intensity levels and CDF is the cumulative density function of the histogram x is the intensity of a pixel: $\text{CDF}(\text{hist}, i) = \sum_{j=0}^{i} p_j$




```
cdf_im = cumsum(hist_im);
cdf_im = cdf_im/cdf_im(end)
```

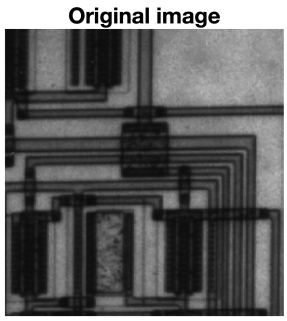
```
cdf_im = 1×256
         0         0         0         0         0         0         0         0 ···
```

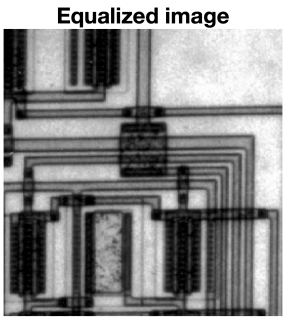
```
figure
stairs([0 : 255], cdf_im, 'b','LineWidth',3), title('cumulative of image pdf')
xlabel('Input');
ylabel('cdf values');
```

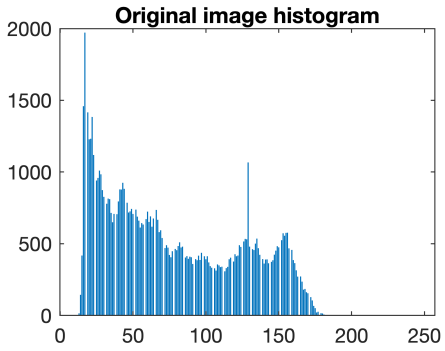


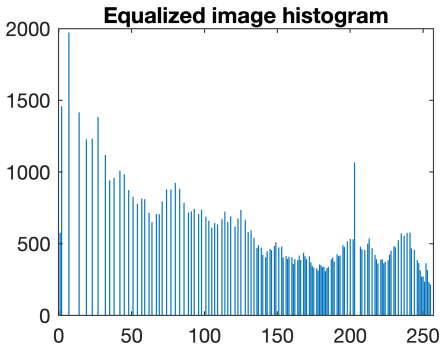
Mapping pixel intensities

```
map = floor(255 * cdf_im);
im_eq = map(im);

hist_im_eq = hist(im_eq(:), 0 : 255);


figure(8);
subplot(2,2,1);
imshow(im);
title('Original image');

subplot(2,2,2);
imshow(im_eq/255);
title('Equalized image');

subplot(2,2,3);
bar(hist_im);
title('Original image histogram');

subplot(2,2,4);
bar(hist_im_eq);
title('Equalized image histogram');
```



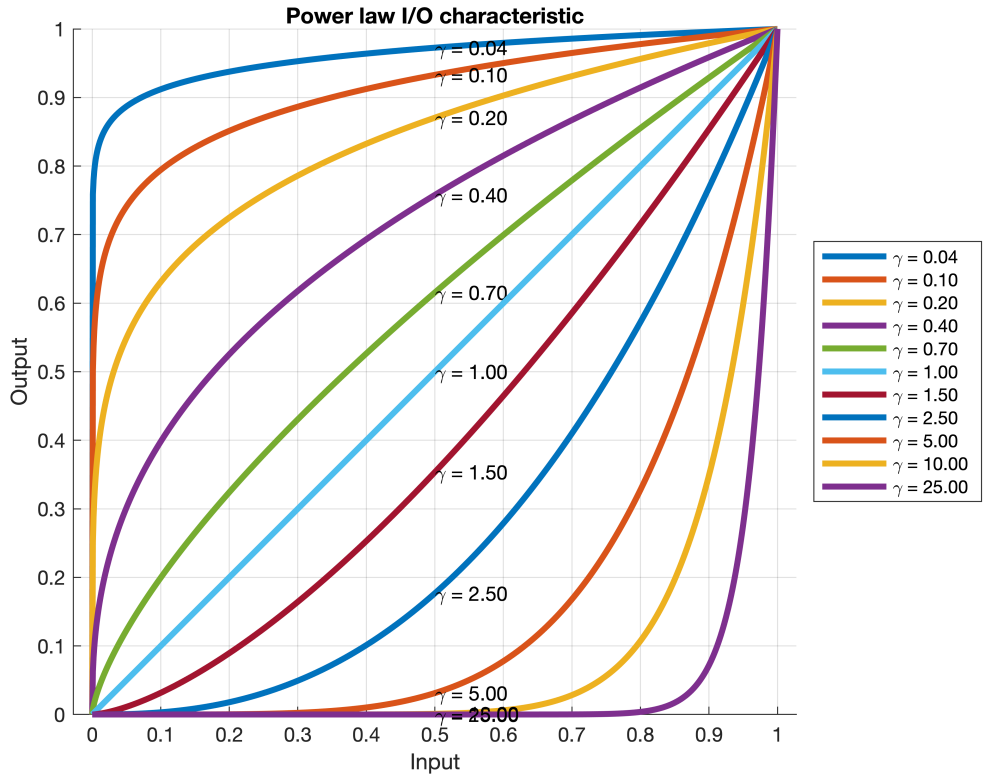






Gamma correction

```
im = imread('E1_data/image_Lena512.png');
figure;
cla
hold on;
x = 0:0.001:1;
for gamma = [.04 .1 .2 .4 .7 1 1.5 2.5 5 10 25]
    y = x.^gamma;
    plot(x,y,'DisplayName',sprintf('\\gamma = %.2f',gamma),'Linewidth',3);
    % display the text
    text(x(round(end/2)),y(round(end / 2)), sprintf('\\gamma = %.2f',gamma));
end
xlabel('Input');
ylabel('Output');
title('Power law I/O characteristic');
leg = legend('Location','eastoutside');
axis equal
grid on
hold off
```



play with gammaVal and set a different value for the gamma Correction.

check which part of the image are being mostly affected by this transformation

```
imd = im2double(im);
gammaVal = 0.97
```

```
gammaVal = 0.9700
```

```
figure
subplot(1,2,1)
imshow(imd, [])
title('original image')
subplot(1,2,2)
imshow(imd.^gammaVal, [])
title(sprintf('Gamma corrected image using \\gamma = %.2f', gammaVal));
```

original image

Gamma corrected image using γ = 0.97