



Politechnika Łódzka
Instytut Elektroniki

Image Processing and Computer Graphics

Laboratory #1:

Introduction to Image Processing and wxPython

M. Kociński, J. Blumenfeld

Medical Electronics Division
Institute of Electronics

Remember to import all needed modules at the begening of your session

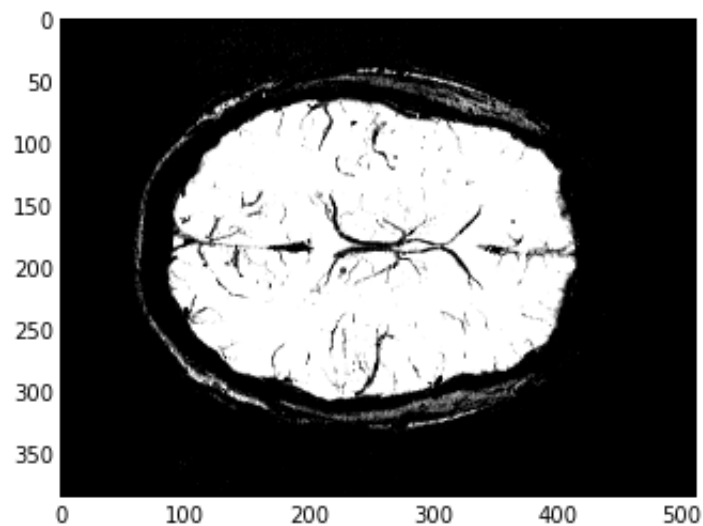
```
In [79]: import numpy as np
import matplotlib.pyplot as plt
import scipy.misc as misc
import Image #(Python Imaging Library - PIL)
```

Load and save image from/to disc file

With the use function imread (matplotlib.pyplot package) load "brain" image and store it into "img" variable. Use the path on your local drive (e.g. "D:\Biomed_2012\brain.bmp"). Display image with imshow() command.

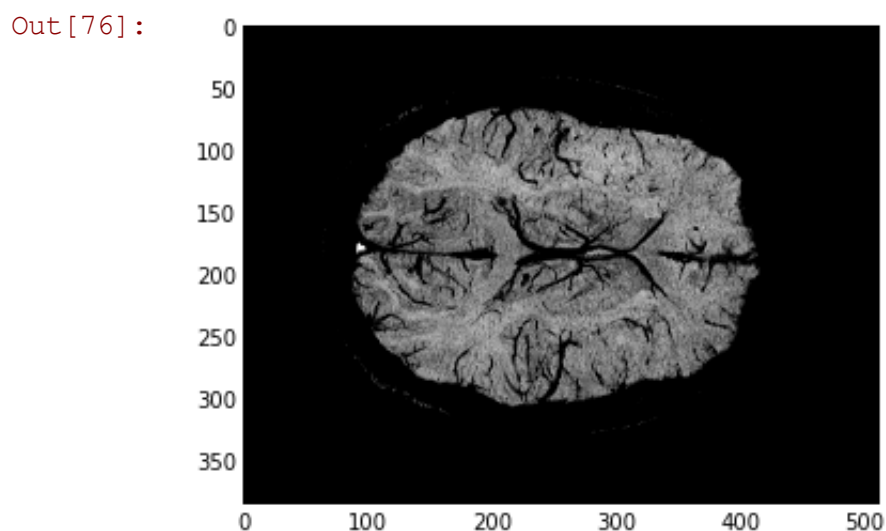
```
In [75]: img = plt.imread('./figures_for_student/brain.bmp')
plt.imshow(img, cmap=plt.cm.gray, vmin=10, vmax=100)
```

```
Out[75]: <matplotlib.image.AxesImage at 0x74e6230>
```



Explain the meaning of "vmin" and "vmax" parameters. Observe the image for several different values. Try to choose these values to obtain the image that contain only brain tissue (as showed below):

```
In [76]: from IPython.display import Image
Image(filename="Figures/brain_windowed.png")
```



Save image to the disc file. Use various file format: bmp, jpg, png, pdf. You can find helpful functions in the modules ([scipy.misc](#), [matplotlib.pyplot](#)):

- `scipy.misc.imsave()`
- `matplotlib.pyplot.imsave()`
- `Image.save()` (PIL package will be introduced soon)

To convert between [Python Imaging Library \(PIL\)](#) and `numpy.ndarray` objects use functions:

- **from Numpy to Image object:**
 - `Image_object = Image.fromarray(some_arguments)`
 - `Image_object = misc.toimage(some_arguments)`
- **from Image to Numpy.ndarray object:**
 - `numpy_array = misc.fromimage(some_arguments)`
 - `numpy_array = numpy.array(some_arguments)`

```
In [77]: plt.imsave('test.png',img, cmap=plt.cm.gray)
misc.imsave('test_1.png', img)
```

Conversion between PIL and Numpy objects

```
In [80]: im = Image.fromarray(img)
print im.mode, im.size, im.format
```

L (512, 384) None

```
In [81]: im2 = misc.toimage(img)
print im2.mode, im2.size, im2.format
```

L (512, 384) None

```
In [82]: img2 = misc.fromimage(im2)
print img2.shape, img2.dtype
```

(384, 512) uint8

```
In [83]: img3 = np.array(im2)
print img3.shape, img3.dtype
```

(384, 512) uint8

Save (Load) to (from) raw file

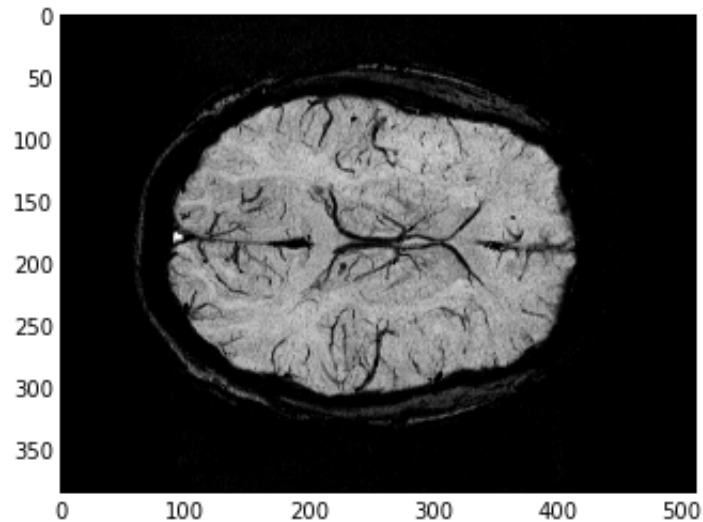
```
In [84]: # Save
name = 'test_2_' + str(img.shape[0]) + '_' + str(img.shape[1]) + '_' + str(img.dtype) + '.raw'
img.tofile(name,format=str(img.dtype))
```

```
In [85]: # Load and display
img4 = np.fromfile(name, dtype='uint8')
print 'shape of the loaded image is: %d' % (img4.shape)
img4.shape=(384,512)
```

```
print "after reshape: %d x %d " % (img4.shape)
imshow(img4, cmap=plt.cm.gray)
```

shape of the loaded image is: 196608
after reshape: 384 x 512

Out[85]: <matplotlib.image.AxesImage at 0x7dd2a10>



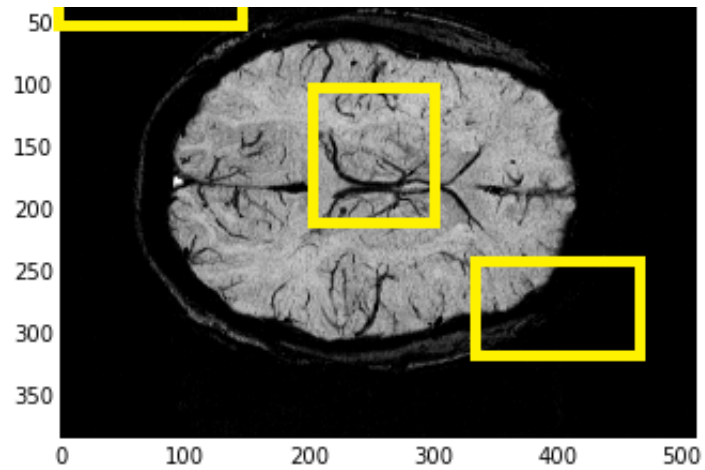
Basic statistical parameters of the image

Write the function that retruns a brightness and contrast of the input image according to the formula given in the [lecture](#). What type of mathematical operation is responsible for change of brightness? And what for the contrast? Calculate these parameters for the whole imge ("brain.bmp") and for some its parts (Retion Of Interests - ROI), next change them with the use of appropriate mathematical operation. How would you define brightness and contrast of the image?

Explain the benefits of sending ndarray as an function argument.

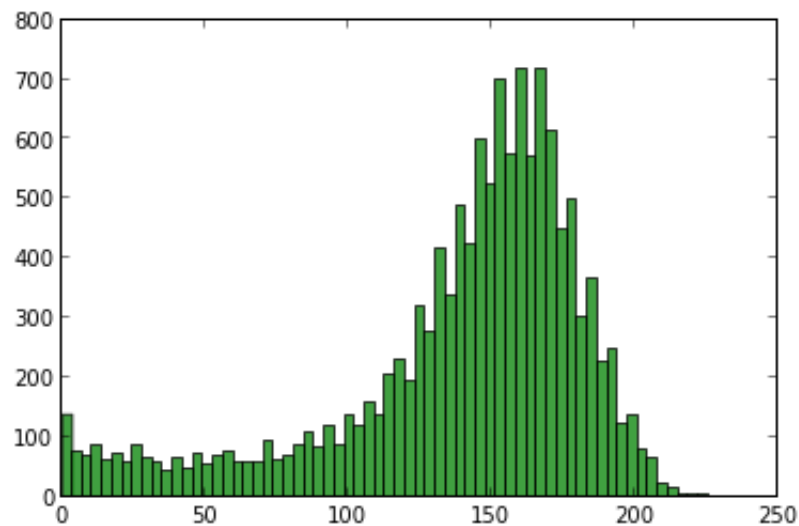
```
In [87]: from IPython.display import Image
Image(filename="Figures/brain_rois_brightness_contrast.png")
```

Out[87]: 



Based on [example](#) observe the histogram for each image. Change numbers of bins.

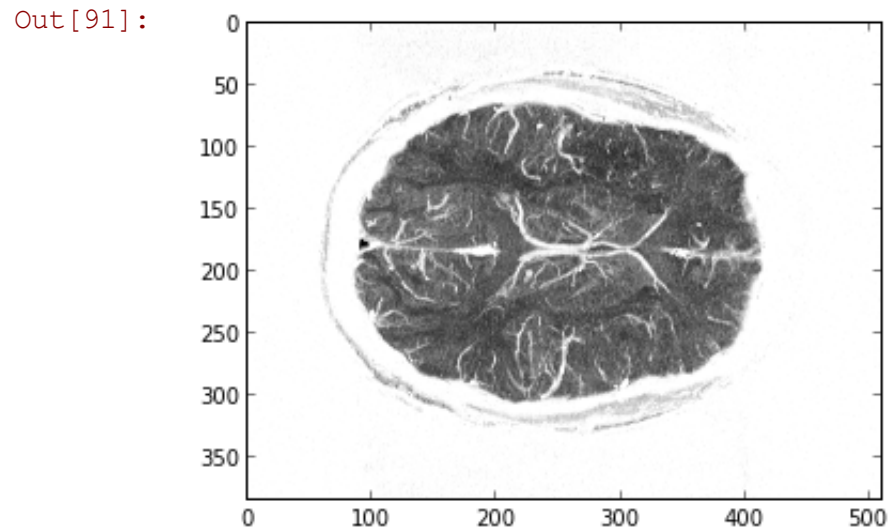
```
In [89]: n, bins, patches = plt.hist(img[100:230,200:300].flatten(0), 64, facecolor='green', alpha=0.75)
```



Grayscale inversion

Based on the [lecture](#) implement a function that invert grayscale levels. You should obtain the image as belowe.

```
In [91]: from IPython.display import Image  
Image(filename="Figures/inverted.png")
```



Nonlinear grayscale transformation

Based on the [lecture](#) implement a functiona of nonlinear grayscale transformations. Compare with oryiginal image. Using nonlinear transformation you should change type of your image to e.g. float. Remember about NORMALISATION (!)

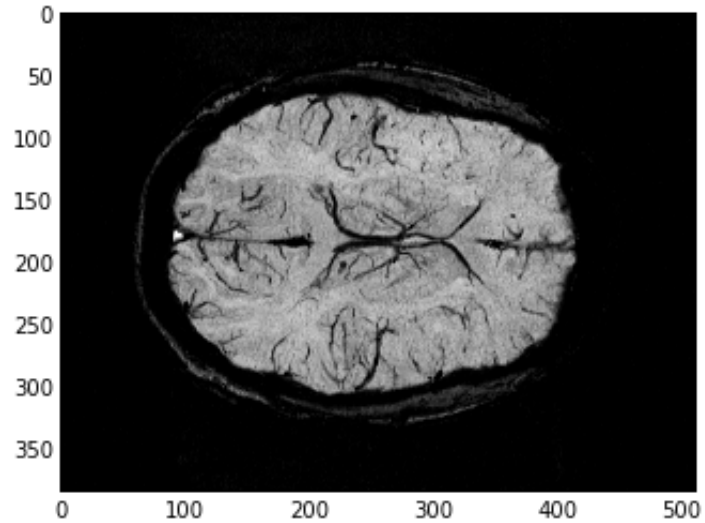
```
img2 = np.array(img, dtype=np.float)
```

Observe the histogram for each image.

Oryginal image

```
In [1]: from IPython.display import Image
```

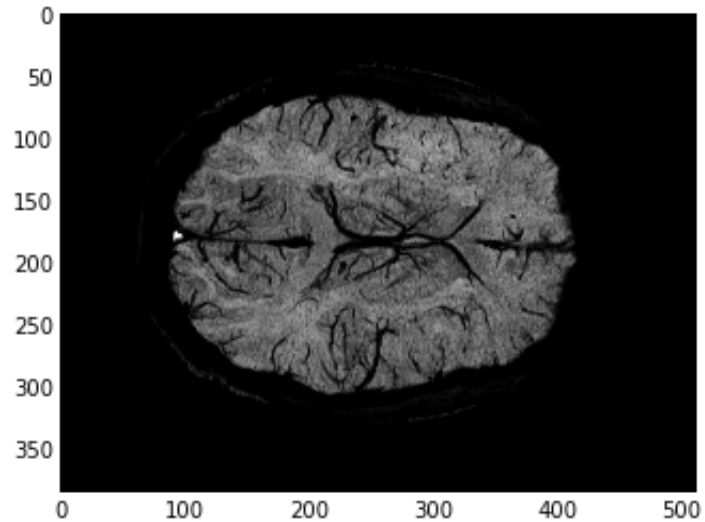
Out[1]:



Brightness image to power of 2

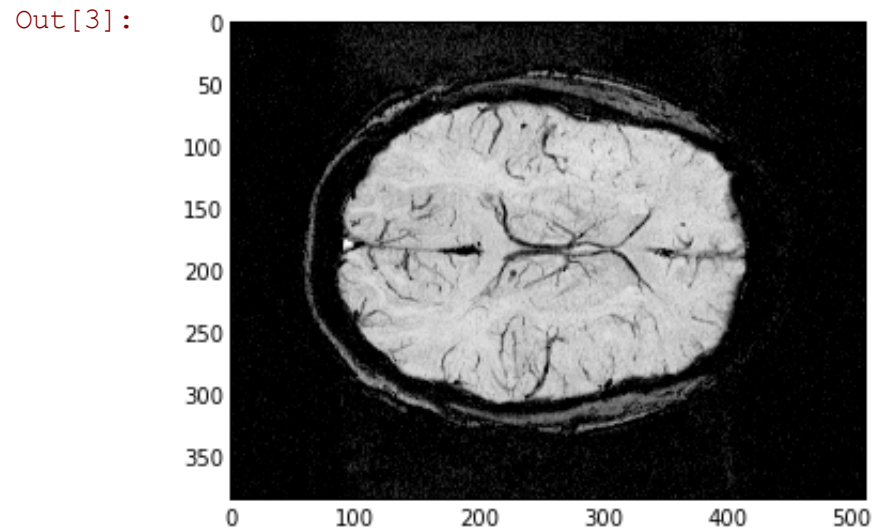
```
In [2]: from IPython.display import Image  
Image(filename="Figures/nonlinear_x2.png")
```

Out[2]:



Square root of the brightness

```
In [3]: from IPython.display import Image
        Image(filename="Figures/nonlinear_sqrt.png")
```



3D image(s)

Load 3D image from raw file (qinp01_3000_036_3_256.raw). Read about [raw](#) data format. Display:

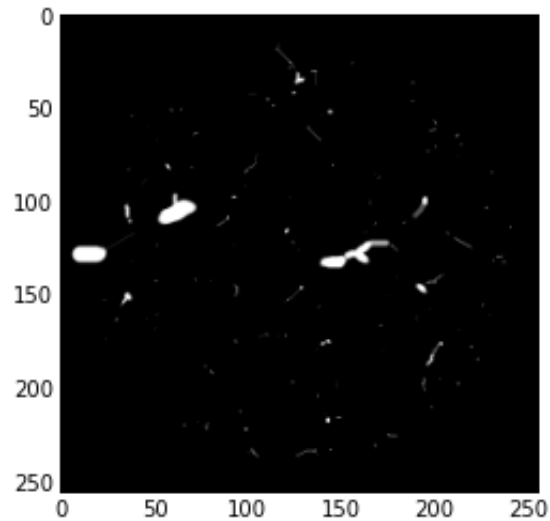
- 125th slide - (use slicing)
- MIP (Maximum Intensity Projection) - use max() function
- MIP with depth information - use argmax() function

```
In [97]: img3d = np.fromfile('./figures_for_student/qinp01_3000_036_3_256.raw', dtype=np.uint8)
        img3d.shape = (256,256,256)
```

```
In [98]: from IPython.display import Image
```

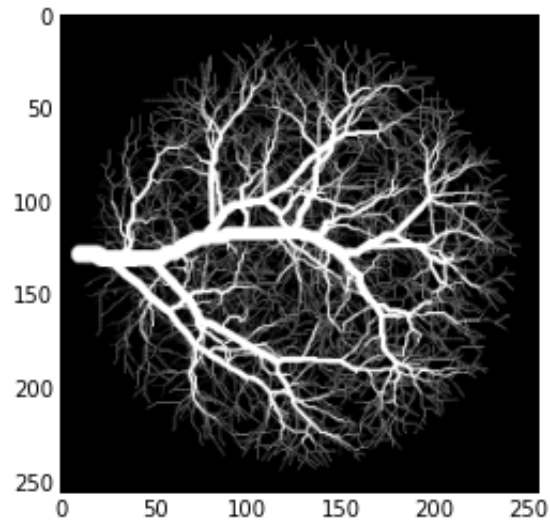
```
In [98]: from IPython.display import Image  
Image(filename="Figures/3d_slice.png")
```

Out[98]:



```
In [99]: from IPython.display import Image  
Image(filename="Figures/3d_mip.png")
```

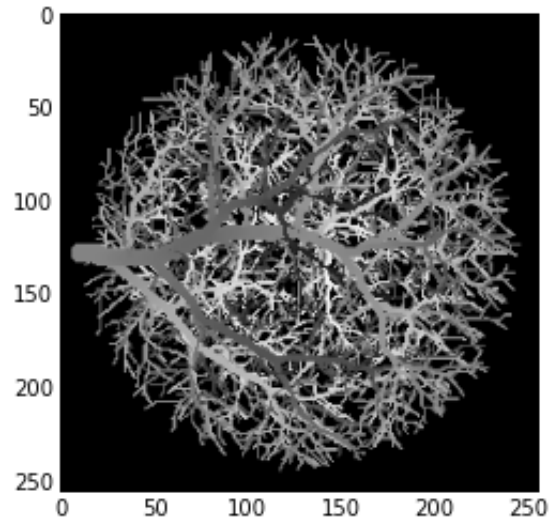
Out[99]:



```
In [100]: from IPython.display import Image
```

```
Image(filename="Figures/3d_argmip.png")
```

Out[100]:



Software for visualization consecutive slides from 3D image

Change software that you have used to thresholding in such a way that you could manipulate displayed slice number with the use of slider.

In [73]: