Digital Image processing

PIL library

What is PIL

- Stands for Python Imaging Library. Pillow is similar
- The Python Imaging Library adds image processing capabilities to your Python interpreter.
- This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.
- The core image library is designed for fast access to data stored in a few basic pixel formats.
- Installation: https://pillow.readthedocs.io/en/stable/installation/basic-installation.html
- Alternatives : OpenCV

Processing Gray images

```python
## to install library: pip install pillow
from PIL import Image
import numpy as np

tree_gray = Image.open("/m/Camera_team/01_team_members/mohamed/DIP/trees_gray.jpeg")
tree_gray_data = np.array(tree_gray)
print(' gray image size : ', tree_gray_data.shape)   #  gray image size :  (177, 187)
tree_gray.show()

# process the image
tree_gray_data[100,110] = 0
tree_gray_data[0:-1, 5:6] = 0

# create an image from numpy array
tree_copy = Image.fromarray(tree_gray_data)
tree_copy.show()
```

0 dark

255 light

Creating images

Grayscale image:

```
im = PIL.Image.new(mode = "L", size = (100, 200),
        color = (150))
im.show()
```

Colored image:

```
im = PIL.Image.new(mode = "RGB", size = (200, 200),
        color = (153, 153, 255))
im.show()
```

1 (1-bit pixels, black and white, stored with one pixel per byte)
L (8-bit pixels, grayscale)
P (8-bit pixels, mapped to any other mode using a color palette)
RGB (3x8-bit pixels, true color)
RGBA (4x8-bit pixels, true color with transparency mask)
CMYK (4x8-bit pixels, color separation)
YCbCr (3x8-bit pixels, color video format)
Note that this refers to the JPEG, and not the ITU-R BT.2020, standard
LAB (3x8-bit pixels, the L*a*b color space)
HSV (3x8-bit pixels, Hue, Saturation, Value color space)
Hue's range of 0-255 is a scaled version of 0 degrees <= Hue < 360 degrees
I (32-bit signed integer pixels)
F (32-bit floating point pixels)

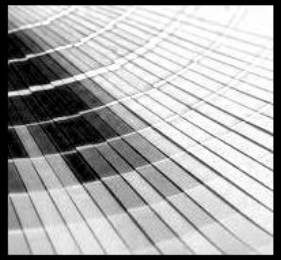
Color Image loading

Color = R + G + B

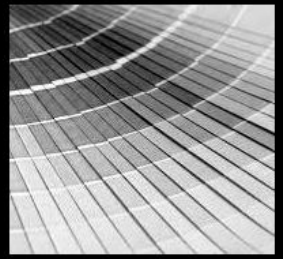
W = 255 + 255 + 255

B = 0 + 0 + 0

Red channel



Green channel



Blue channel

Color Image loading

```python
from PIL import Image
import numpy as np
# open the image
im = Image.open("/m/Camera_team/01_team_members/mohamed/DIP/colors.jpeg")

# copy image data into numpy array access image pixels
data = np.asarray(im)
print(data.shape) # (183, 195, 3)
im.show()

r = data[:,:,0] # red channel
g = data[:,:,1] # green channel
b = data[:,:,2] # blue channel

new_image_r = Image.fromarray(r)
new_image_r.show()

new_image_g = Image.fromarray(g)
new_image_g.show()

new_image_b = Image.fromarray(b)
new_image_b.show()
```

Exercise

- **Load any colored image from your choice**
- **Convert the image to grayscale image using PIL library**
- **Test the function Image.split on the colored image**
- **Display the output of the split function**
- **combine the output of the split function using merge function and display it.**
- **Replace the green channel with zeros and display the output of merge function**