Image processing and Pattern Analysis- Programming Assignment 1

Dead line-16/10/2022

Question 1: [10 Points]

Image resizing is a widely used operation in image processing. Either for scaling up or for scaling down, we use interpolation to find the corresponding pixel intensity in the resulting image. Choice of interpolation method is important as it directly affects the quality of the resulting image. The following interpolation methods were mentioned in class:

- Nearest neighbor
- Bilinear
- Bicubic

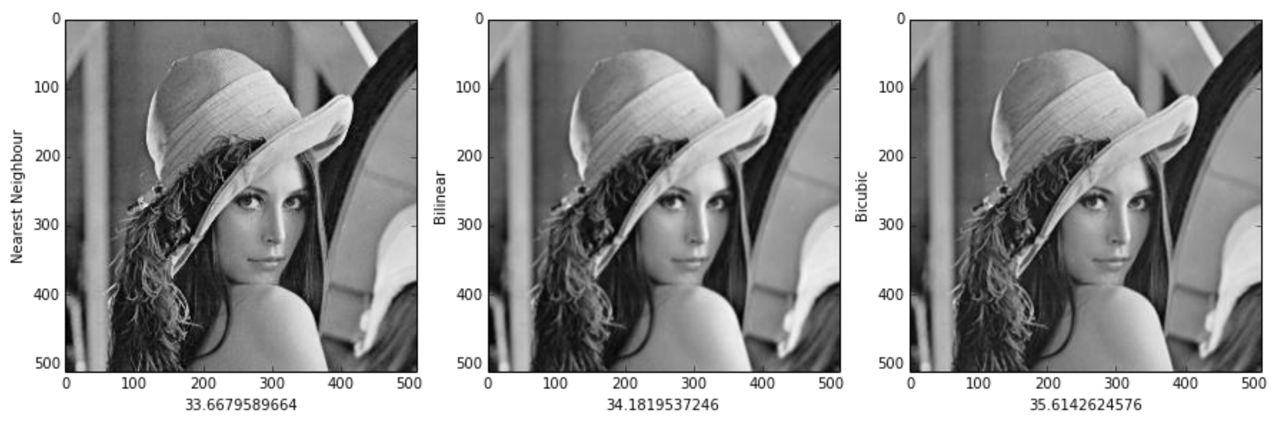
If you look at PILLOWS reference page you can see that the **`resize()`** function has predefined resampling functions:

- **`PIL.Image.NEAREST`**
- **`PIL.Image.BILINEAR`**
- **`PIL.Image.BICUBIC`**
- **`PIL.Image.LANCZOS`**

*LANCZOS is a more sophisticated method which gives better results compared to first three methods. If you're interested search the internet for more details on **LANCZOS Resampling Method**.*

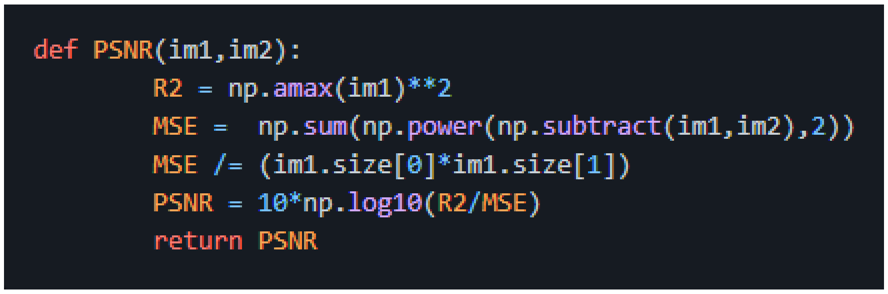
For this question, you will implement a jupyter notebook script and it will perform these operations:

1. Read color version of the image "**`lena.png`**", Convert colored **`lena`** image to grayscale image.
2. Resize (scale down) the grayscale **`lena`** image to half its original size. Then, resize (scale up) it up back to the original size using resize() with **`PIL.Image.NEAREST`**
3. Resize (scale down) the grayscale **`lena`** image to half its original size. Then, resize (scale up) it up back to the original size using resize() with **`PIL.Image.BILINEAR`**
4. Resize (scale down) the grayscale **`lena`** image to half its original size. Then, resize (scale up) it up back to the original size using resize() with **`PIL.Image.BICUBIC`**
5. Your script should display the results as follows using **`matplotlib`** along with the PSNR values w.r.t original image below the result image.



PSNR (Peak signal to Noise Ratio): PSNR computes the peak signal-to-noise ratio, in decibels, between two images. This ratio is used as a quality metric between the original and a reconstructed image. The higher the PSNR value, the better the quality of the reconstructed image.

The following function can be used to calculate PSNR between a source (im1) and target (im2) image.



```
def PSNR(im1,im2):
        R2 = np.amax(im1)**2
        MSE =  np.sum(np.power(np.subtract(im1,im2),2))
        MSE /= (im1.size[0]*im1.size[1])
        PSNR = 10*np.log10(R2/MSE)
        return PSNR
```

Question 2: [10 Points]

1. Define a generic function (**`ChSwap`**) that takes a color image as input and returns the corresponding output image in which the R,G,B channels are swapped as follows:
    - Splits the input image into R, G, B Channels
    - Red (Out) = Blue (In)
    - Green (Out) = Red (In)
    - Blue (Out) = Green(In)
    - Combine the new channels to create the new image and return
2. Read color version of the image "**`lena.png`**".
3. Use function **`ChSwap`** to create a color band swapped version of **`lena`**
4. Using **`matplotlib plot,`** display the original and new images side-by-side.

Question 3: [10 Points]

1. We have discussed intensity transformation functions in detail in class and you have gained necessary implementation skills from Lab1. Now, let's apply them to a real life challenge!

2. Although not very common among us Indians, red-eye effect is a common phenomenon in flash photography where pupils in the eyes appear red! Automatic Redeye Detection and Correction Algorithms are now very common in most cameras and cell phones!! (*Due to the inherent complexity, we will not deal with automatic detection aspect now, but will only work on correction*)

3. Come up with your own red-eye correction algorithm (**`RedEyeRemoval`**) using piece-wise intensity transformations! (Don't worry if other parts of the image with red color get distorted!)

4. Two important aspect of this algorithm are (1) Figuring out the range of intensities to modify (2) Deciding on what the modified intensity values should be!

5. Four sample images with red eyes are provided with this lab. Your function, **`RedEyeRemoval`**, should take an input image and display/output the corrected image. Using **`matplotlib`** plot, display the original and new images side-by-side. (*Test your algorithm on all the sample images.)*