ICE 472: Digital Speech & Image Processing

Summer 2020

Experiment 3: Gray Level Transformations

Name: Suraya Ahlam

Student ID: 2016-1-50-014

Assignment:

(1) Running this notebook and producing the outputs.

Library Imports

```
import skimage
from skimage import data
import matplotlib.pyplot as plt
import numpy as np
```

Loading the Image

```
img = data.camera()
img_flat = img.flatten()    # A flat version of the image is also prepared, in order to easil
plt.imshow(img, cmap = 'gray')
plt.show()
```

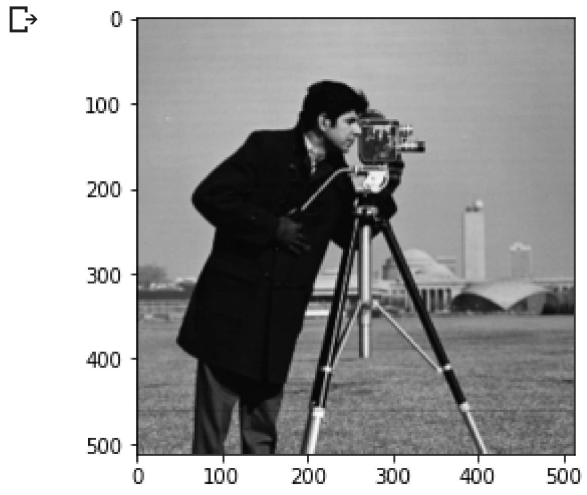


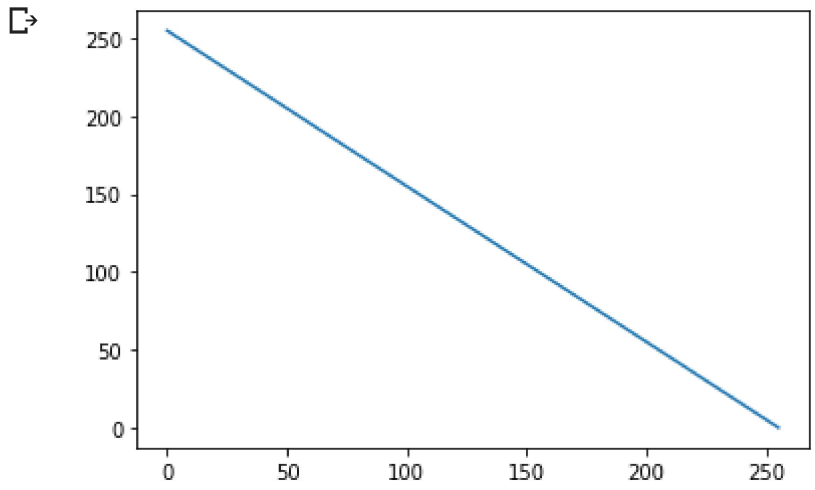
Image Negatives

$$s = L - 1 - r$$

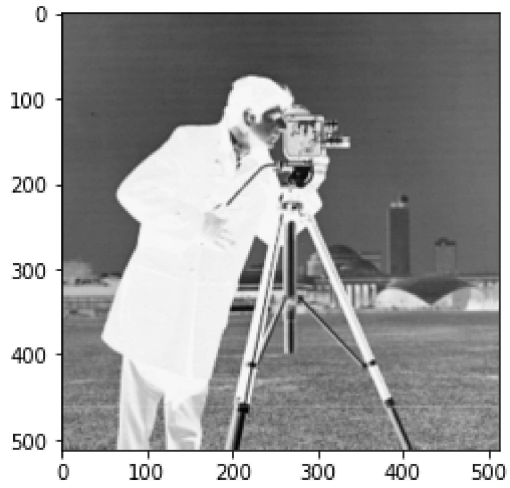
```
L = 256                     # Number of Gray Levels
neg = lambda r: L - 1 - r     # Lambda Function for the Transformation Operation
```

Transformation Function:

```
r = np.arange(L)
s = [neg(x) for x in r]
plt.plot(r,s)
plt.show()
```




```
neg_img = neg(img)
plt.imshow(neg_img, cmap = 'gray')
plt.show()
```



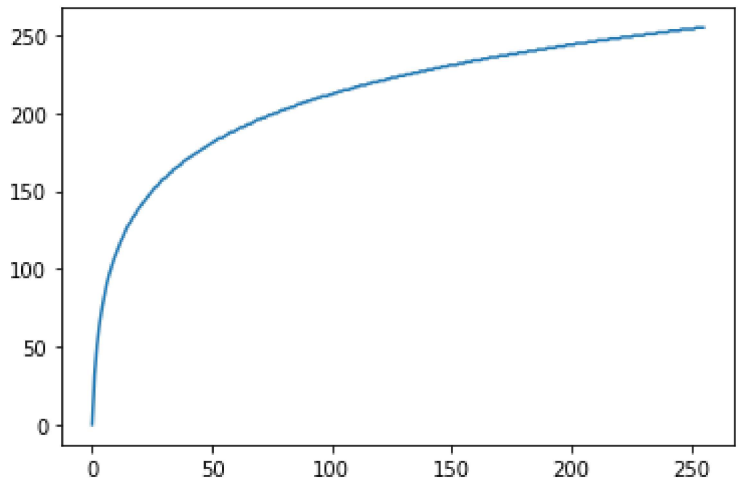
Log Transformation

$$s = c * log(1 + r)$$

```
L = 256
c = (L-1)/np.log(L-1)                                  # Automatically calculate the value of c
logt = lambda r: int(np.round(c * np.log(1+r)))      # Lambda Function for Log Transformation
```

Transformation Function

```
r = np.arange(L)
s = [logt(x) for x in r]
plt.plot(r,s)
plt.show()
```



Producing the log-transformed image is done in a step/by step process:

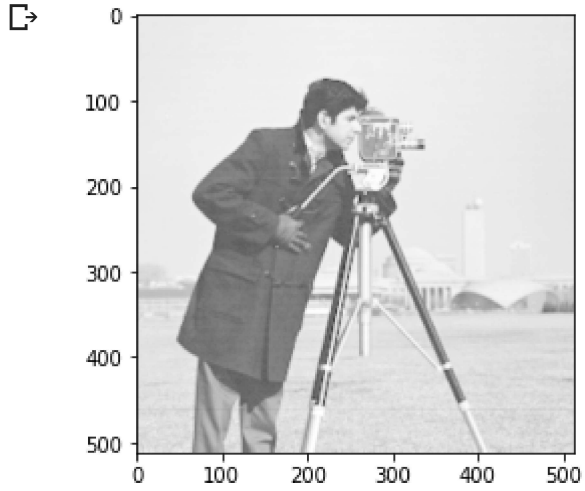
`[f(x) for x in img_flat]` -> Calculates all the values of pixels individually and stores them into a list

`np.asarray( ... list ... )` -> Converts the list of pixels into a numpy array

`(....).reshape(img.shape)` -> reshape the flat numpy array to the original image shape

```
log_img = np.asarray([logt(x) for x in img_flat]).reshape(img.shape)
```

```
plt.imshow(log_img, cmap = 'gray')
plt.show()
```

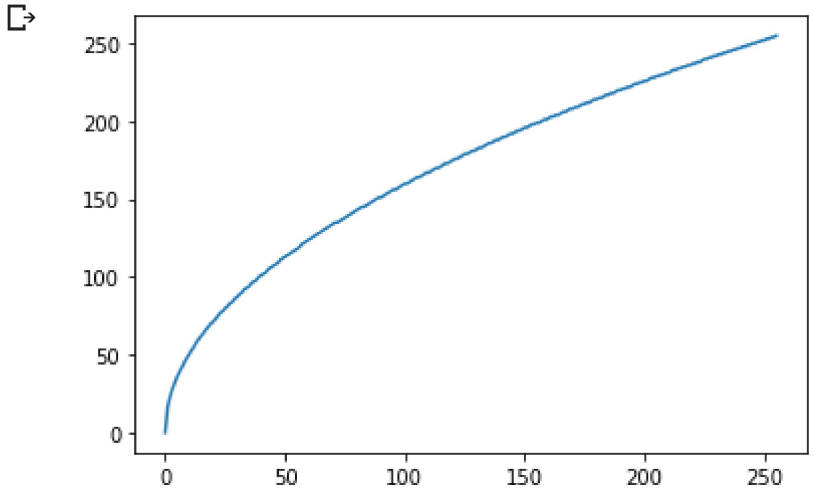


Power Law Transformations

$s = cr^{\gamma}$

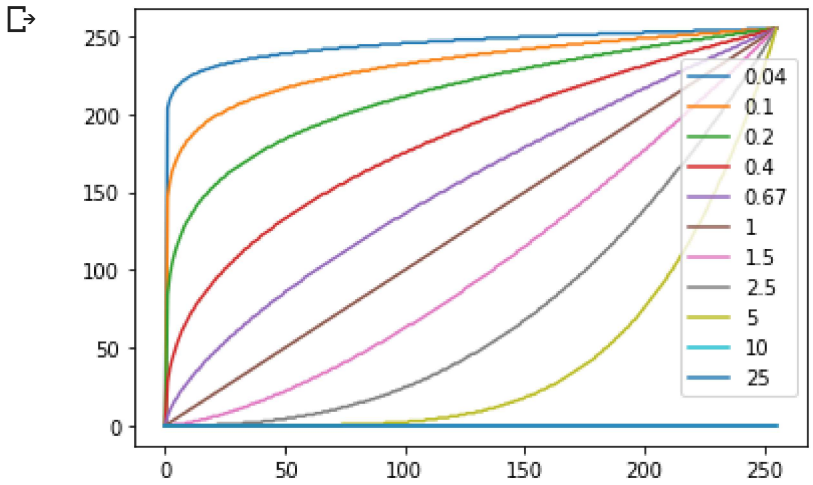
```
L = 256
gamma = 1/2
c = (L-1)**(1-gamma)                          # Automatically calculate the value of c
powt = lambda r: int(np.round(c*r**gamma))    # Lambda function for the power law transforma
```

```
r = np.arange(L)
s = [powt(x) for x in r]
plt.plot(r,s)
plt.show()
```



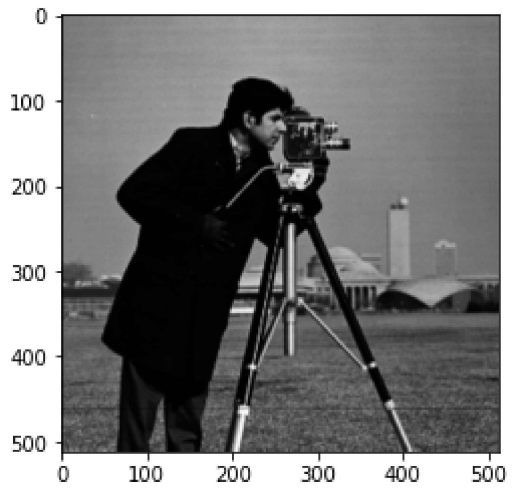
Effect of Gamma on Transformation function

```
gammas = [0.04, 0.10, 0.20, 0.40, 0.67, 1, 1.5, 2.5, 5, 10, 25]       # Array containing a nu


# Loop over gamma values and plot their transformation functions
L = 256
r = np.arange(L)
for gamma in gammas:
  c = (L-1)**(1-gamma)
  powt = lambda r: int(np.round(c*r**gamma))
  s = [powt(x) for x in r]
  plt.plot(r,s)
  plt.legend(gammas)
plt.show()
```




```
gamma = 1.5
c = 255**(1-gamma)
powt = lambda r: int(np.round(c*r**gamma))
```

```
power_img = np.asarray([powt(x) for x in img_flat]).reshape(img.shape)


plt.imshow(power_img, cmap = 'gray')
plt.show()
```



Contrast Stretching

```
(r1, s1) = (100, 20)
(r2, s2) = (150, 220)
```

Equation of Straight Line from (0,0) to (r1, s1):

$y = \frac{s1}{r1}x$

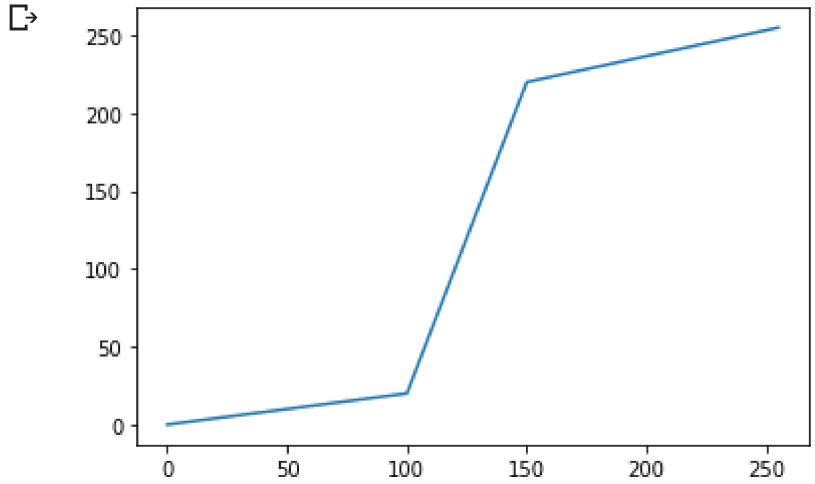
Equation of Straight Line from (r1, s1) to (r2, s2):

$y = \frac{s2-s1}{r2-r1}(x - r1) + s1$

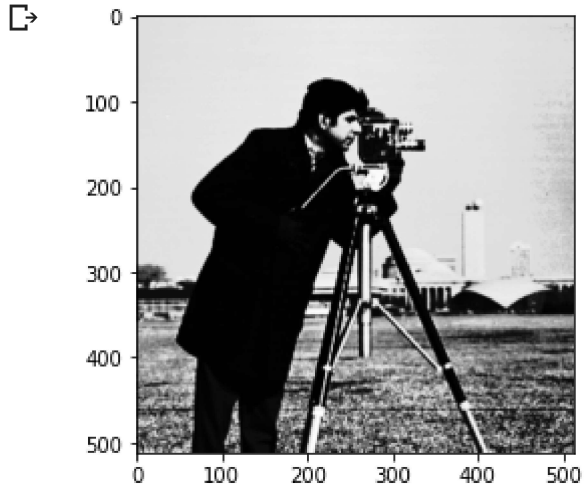
Equation of Straight Line from (r2, s2) to (255, 255):

$y = \frac{s2-255}{r2-255}(x - 255) + 255$

```
# Function for contrast stretching
def cont_stretch(x):
  if (x <= r1):
    return s1/r1*x
  if (r1 < x <= r2):
    return (s2 - s1)/(r2 - r1)*(x - r1) + s1
  if (x > r2):
    return (s2 - 255)/(r2 - 255)*(x - 255) + 255


r = np.arange(L)
s = [cont_stretch(x) for x in r]
plt.plot(r,s)
plt.show()
```




```
cont_img = np.asarray([cont_stretch(x) for x in img_flat]).reshape(img.shape)


plt.imshow(cont_img, cmap = 'gray')
plt.show()
```

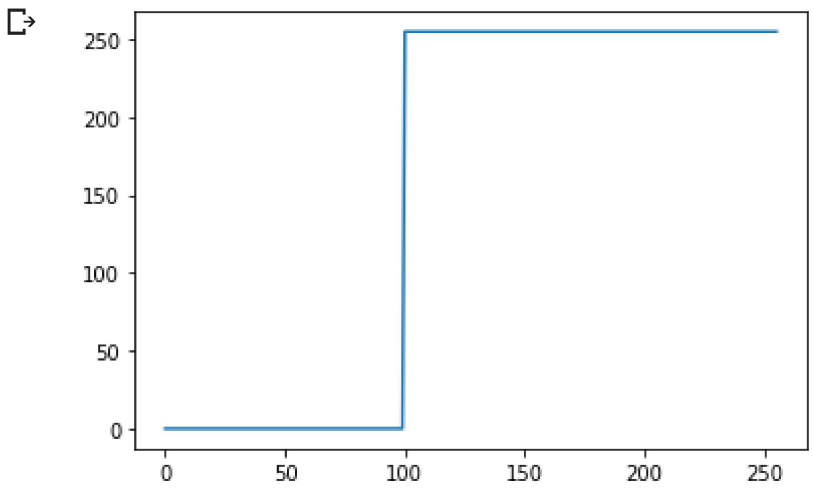


Thresholding

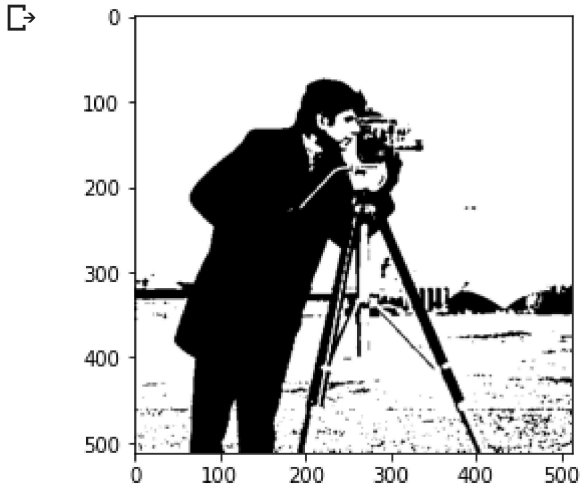
```
m = 100           # Value of Threshold
def thresh(x):
  if(x < m):
    return 0
  else:
    return 255
```

```
r = np.arange(L)
```

```
s = [thresh(x) for x in r]
plt.plot(r,s)
plt.show()
```




```
thresh_img = np.asarray([thresh(x) for x in img_flat]).reshape(img.shape)


plt.imshow(thresh_img, cmap = 'gray')
plt.show()
```



Assignment:

(1) Run this notebook and produce the outputs.

(2) Choose a Gray Image (not color image), or input a color image as a gray image (to do this, set the **as_gray** argument to **True** while using the **imread()** function. You can either load the image by uploading it or use a link.

(a) Apply a gamma of 2.5 to obtain a new image. Then, apply gamma correction to this image to obtain the original image. Show both images.

(b) Apply the following transformation to the original image and show the image:

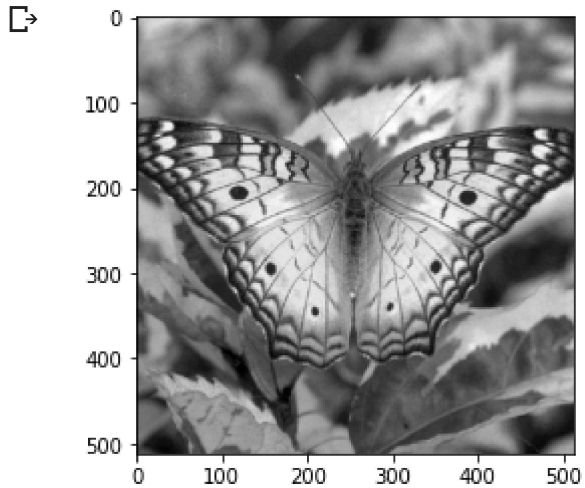
$s = \frac{c}{1+r+r^2}$

[Calculate the value of c by yourself, assume 256 Level quantization]

(3) Enhance the contrast of the following image: https://homepages.inf.ed.ac.uk/rbf/HIPR2/images/wom1.gif so that the pixels in the range 100 ~ 120 are boosted in contrast. Show the input & output image.

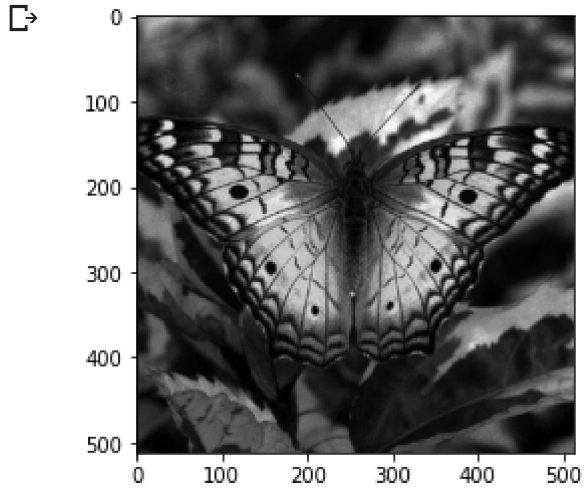
Loading the image:

```
from skimage import io


pic = io.imread('butterfly.gif', as_gray=True)
pic_flat = pic.flatten()
plt.imshow(pic, cmap = 'gray')
plt.show()
```

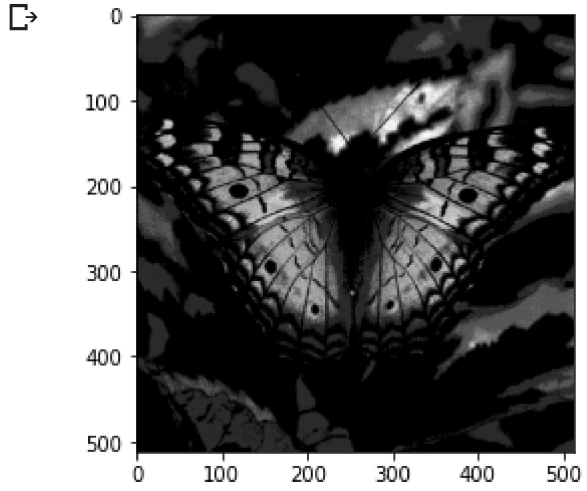


Problem 2(a): Gamma Correction

```
L=256
gamma = 2.5
c = (L-1)**(1-gamma)
powt = lambda r: int(np.round(c*r**gamma))
power_pic = np.asarray([powt(x) for x in pic_flat]).reshape(pic.shape)
plt.imshow(power_pic, cmap = 'gray')
plt.show()
```

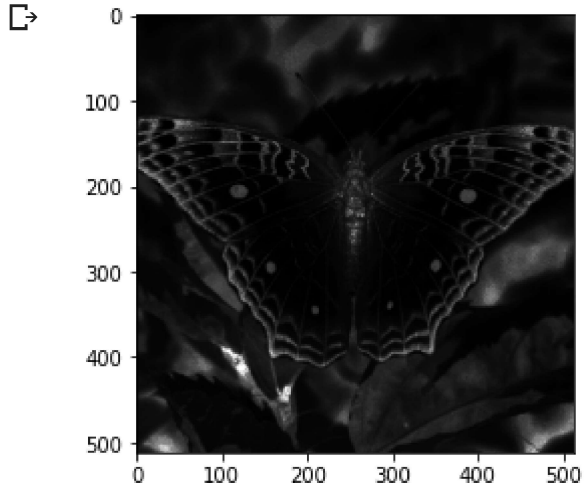



```
power_flat = power_pic.flatten()
gamma = 1/2.5
powt = lambda r: int(np.round(c*r**gamma))
corrected_pic = np.asarray([powt(x) for x in power_flat]).reshape(pic.shape)
plt.imshow(corrected_pic, cmap= 'gray')
plt.show()
```



Problem 2(b): Custom Transformation

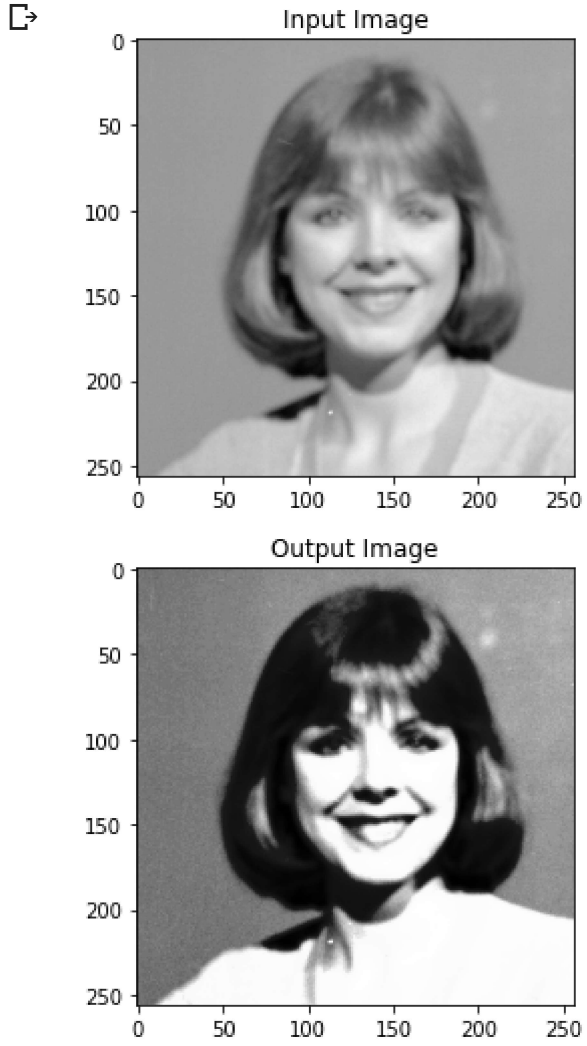
```
L = 256
c = (L-1)*(1+(L-1)+((L-1)**2))
st = lambda r: int(np.round(c / (1+r+(r**2))))


custom_pic = np.asarray([st(x) for x in pic_flat]).reshape(pic.shape)
plt.imshow(custom_pic, cmap = 'gray')
plt.show()
```



Problem 3: Contrast Enhancement

```
lady = io.imread('https://homepages.inf.ed.ac.uk/rbf/HIPR2/images/wom1.gif')
lady_flat = lady.flatten()


(r1, s1) = (100, 50)
(r2, s2) = (120, 230)
```

```
cont_lady = np.asarray([cont_stretch(x) for x in lady_flat]).reshape(lady.shape)
plt.imshow(lady, cmap = 'gray')
plt.title('Input Image')
plt.show()
plt.imshow(cont_lady, cmap = 'gray')
plt.title('Output Image')
plt.show()
```



Discussion:

In this experiment, I have learn about various gray level transformations such as, image negatives, log transformation, power law transformation, effect of gamma on law transformation, contrast streching and thresholding. I have saw the effect of those transformation. I have done the assignment part, where I have upload an image as gray image, for that I have also import io.skimage. Then I apply gamma 2.5 in that image and showed the image. After that, I corrected the gamma by 1/2.5 and showed the image. I have also made custom transformation as given in the question and showed the image. It work as image negative transformation. I have enhanced the contrast of the following image: https://homepages.inf.ed.ac.uk/rbf/HIPR2/images/wom1.gif so that the pixels in the range 100 ~ 120 are boosted in contrast. And showed the input and output image. At last, now I can apply any gray transformation function in any image.