**Practical No: 7**

Page | 15

**Aim:** Write program to implement point/pixel intensity transformations

**A:** Log and Power-law transformations.

**Log Transformation:**

During log transformation, the dark pixels in an image are expanded as compare to the higher pixel values. The higher pixel values are kind of compressed in log transformation.

**Program code:**

from PIL import Image

import pylab

def plot\_image(image, title=''):

pylab.title(title, size=20), pylab.imshow(image)

pylab.axis('off')

im = Image.open(r'C:\Users\LJP\_IT\_LAB\Desktop\Ravi Singh\Image Processing\Docs\nature.jpg')

im1=im.point(lambda i: 255\*np.log(1+i/255))

#im\_r, im\_g, im\_b = im.split()

pylab.style.use('ggplot')

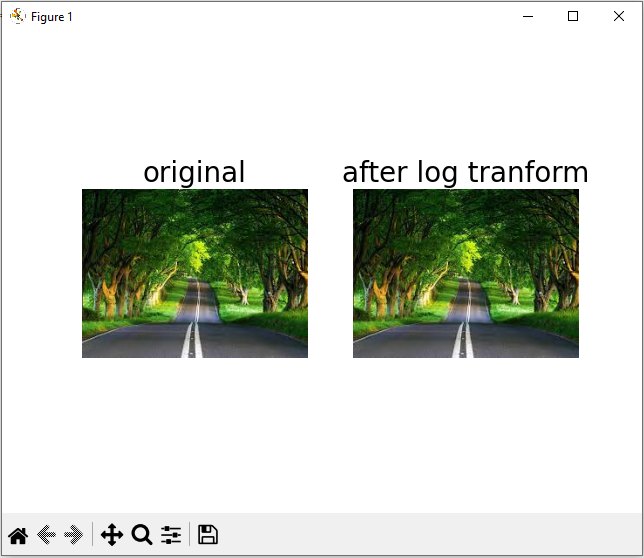
pylab.subplot(121), plot\_image(im, 'original')

pylab.subplot(122), plot\_image(im1, 'after log tranform')

pylab.imshow(im)

pylab.show()

**Output:**



**Power-Law Transformation:**

Page | 16

This type of transformation is used for enhancing images for different type of display devices. The gamma of different display devices is different.

**Program Code:**

im = img\_as\_float(imread('../images/earthfromsky.jpg'))

gamma = 5

im1 = im\*\*gamma

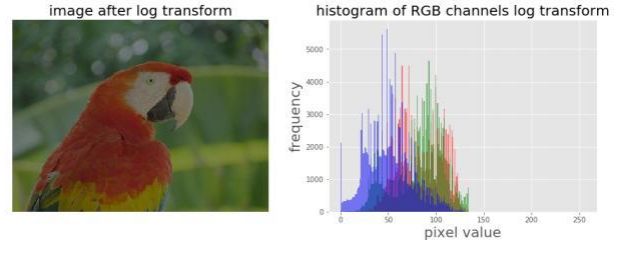
pylab.style.use('ggplot')

pylab.figure(figsize=(15,5))

pylab.subplot(121), plot\_hist(im[...,0], im[...,1], im[...,2],'histogram for RGB channels (input)')

pylab.subplot(122), plot\_hist(im1[...,0], im1[...,1], im1[...,2],'histogram for RGB channels (output)') pylab.show()

**Output:**



Page 13

Page | 17

**B:** Histogram equalization with scikit-image

Histogram Equalization is a computer image processing technique used to improve contrast in images. It accomplishes this by effectively spreading out the most frequent intensity values

**Program Code:**

img = rgb2gray(imread('../images/earthfromsky.jpg'))

img\_eq = exposure.equalize\_hist(img)

img\_adapteq = exposure.equalize\_adapthist(img, clip\_limit=0.03)

pylab.gray()

images = [img, img\_eq, img\_adapteq]

titles = ['original input (earth from sky)','after histogram equalization','after adaptive histogramequalization']

for i in range(3):

pylab.figure(figsize=(20,10)), plot\_image(images[i], titles[i])

pylab.figure(figsize=(15,5))

for i in range(3):

pylab.subplot(1,3,i+1), pylab.hist(images[i].ravel(), color='g'), pylab.title(titles[i], size=15) pylab.show()

**Output:**



**C**: Contrast Adjustments

Page | 18

Contrast adjustment remaps image intensity values to the full display range of the data type. An image with good contrast has sharp differences between black and white.

**Program Code:**

im = Image.open('../images/cheetah.png')

im\_r, im\_g, im\_b, \_ = im.split()

pylab.style.use('ggplot')

pylab.figure(figsize=(15,5))

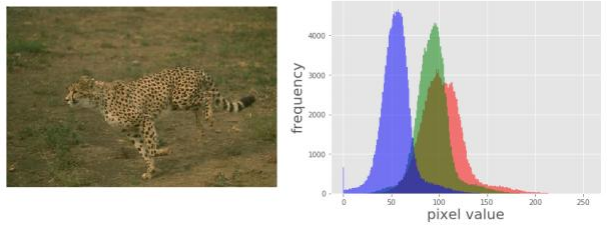
pylab.subplot(121)

plot\_image(im)

pylab.subplot(122)

plot\_hist(im\_r, im\_g, im\_b)

pylab.show()

 def contrast(c): return 0 if c < 70 else (255 if c > 150 else (255\*c - 22950) / 48)

# piece-wise linear

functionim1 = im.point(contrast)

im\_r, im\_g, im\_b, \_ = im1.split()

pylab.style.use('ggplot')

pylab.figure(figsize=(15,5))

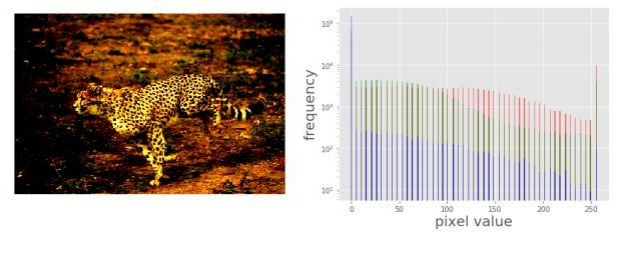
pylab.subplot(121)

plot\_image(im1)

pylab.show()

**Output:**

Page | 19



**D:** Half-toning

Halftone is the reprographic technique that simulates continuous-tone imagery through the use of dots, varying either in size or in spacing, thus generating a gradient-like effect.

**Program Code:**

im = Image.open('../images/swans.jpg').convert('L')

im = Image.fromarray(np.clip(im + np.random.randint(- 128, 128, (im.height, im.width)), 0, 255).astype(np.uint8))

pylab.figure(figsize=(12,18))

pylab.subplot(221), plot\_image(im,'original image (with noise)')

th = [0, 50, 100, 150, 200]

for i in range(2, 5):

im1 = im.point(lambda x: x > th[i])

pylab.subplot(2,2,i), plot\_image(im1,'binary image with threshold=' + str(th[i]))

pylab.show()

**Output:**

Page | 20

