**Intensity Transformation:**  
import cv2

import numpy as np

import matplotlib.pyplot as plt

**# Image Negative:**

import imageio

import matplotlib.pyplot as plt

image = cv2.imread("C:\\Users\\kambl\\Computer\_Vision\_OpenCV\\Sample Images\\sceneBeach.jpg")

plt.imshow(image, 'gray')

plt.figure(figsize = (6,6))

plt.imshow(255 - image);

**# Intensity level slicing without background**

image = cv2.imread("C:\\Users\\kambl\\Computer\_Vision\_OpenCV\\Sample Images\\sceneBeach.jpg")

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

gray = cv2.cvtColor(image, cv2.COLOR\_RGB2GRAY)

x,y = image.shape[:2]

plt.imshow(gray,'gray')

plt.title('Original Image')

plt.show()

z = np.zeros((x,y))

for i in range(0,x):

for j in range(0,y):

if(gray[i][j]>50 and gray[i][j]<150):

z[i][j]=255

else:

z[i][j]=0

plt.imshow(z, 'gray')

plt.title('Sliced Original Image')

plt.show()

# **Intensity level slicing with background**

image = cv2.imread("C:\\Users\\kambl\\Computer\_Vision\_OpenCV\\Sample Images\\sceneBeach.jpg")

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

gray = cv2.cvtColor(image, cv2.COLOR\_RGB2GRAY)

x,y = image.shape[:2]

plt.imshow(gray,'gray')

plt.title('Original Image')

plt.show()

z = np.zeros((x,y))

for i in range(0,x):

for j in range(0,y):

if(gray[i][j]>50 and gray[i][j]<150):

z[i][j]=255

else:

z[i][j]=gray[i][j]

plt.imshow(z, 'gray')

plt.title('Sliced with background')

plt.show()

**# Log transformation**

image = cv2.imread("C:\\Users\\kambl\\Computer\_Vision\_OpenCV\\Sample Images\\sceneBeach.jpg")

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

gray = cv2.cvtColor(image, cv2.COLOR\_RGB2GRAY)

x,y = image.shape[:2]

plt.imshow(gray,'gray')

plt.title('Original Image')

plt.show()

c = 255/(np.log(1+np.max(gray)))

log = c\*np.log(1+gray)

log = np.array(log,dtype=np.uint8)

plt.imshow(log, 'gray')

plt.show()

**# Power Law**

image = cv2.imread("C:\\Users\\kambl\\Computer\_Vision\_OpenCV\\Sample Images\\sceneBeach.jpg")

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

gray = cv2.cvtColor(image, cv2.COLOR\_RGB2GRAY)

x,y = image.shape[:2]

plt.imshow(gray,'gray')

plt.title('Original Image')

plt.show()

z = np.zeros((x,y))

gamma = float(input("Enter gamma value: "))

c = 255/(np.log(1+np.max(gray)))

z = np.array(c\*(gray\*\*gamma), dtype = 'uint8')

plt.imshow(z, 'gray')

plt.show()

**# HISTOGRAM PLOTTING**

1. **Histogram Plotting**

import cv2

import matplotlib.pyplot as plt

import numpy as np

from skimage import exposure

from skimage.exposure import match\_histograms

from matplotlib.colors import NoNorm

# reads an input image

img = cv2.imread(r"C:\Users\hp\Downloads\rose.jpg")

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

gray = cv2.cvtColor(img, cv2.COLOR\_RGB2GRAY)

plt.imshow(gray,'gray')

plt.title('Original Image')

plt.show()

# find frequency of pixels in range 0-255

histr = cv2.calcHist([gray],[0],None,[50],[0,256])

# show the plotting graph of an image

plt.plot(histr)

plt.xlabel('Number of Bins')

plt.ylabel('Number of pixels')

plt.title('Histogram')

plt.show()

for color histogram

img = cv2.imread(r".\rose.jpg")

color = ('b','g','r')

for i,col in enumerate(color):

    histr = cv2.calcHist([img],[i],None,[256],[0,256])

    plt.plot(histr,color = col)

    plt.xlabel('Intensity value')

    plt.ylabel('Number of pixels')

    plt.show()

1. **Linear stretching**

def pixelVal(pix, r1, s1, r2, s2):

if (0 <= pix and pix <= r1):

return (s1 / r1)\*pix

elif (r1 < pix and pix <= r2):

return ((s2 - s1)/(r2 - r1)) \* (pix - r1) + s1

else:

return ((255 - s2)/(255 - r2)) \* (pix - r2) + s2

plt.imshow(img.astype('uint8')) # function is used to cast a pandas object to a specified data type

plt.title('Original Image')

plt.show()

# Define parameters.

r1 = 18

s1 = 0

r2 = 213

s2 = 255

# Vectorize the function to apply it to each value in the Numpy array.

pixelVal\_vec = np.vectorize(pixelVal)

# Apply contrast stretching.

linear\_stretched = pixelVal\_vec(img, r1, s1, r2, s2)

plt.imshow(linear\_stretched.astype('uint8'))

plt.title('Result')

plt.show()

plt.hist(linear\_stretched.flatten())

plt.title('Original Image Histogram')

1. **Histogram Equalization**

#Apply histogram equalization

equl = cv2.equalizeHist(gray)

plt.imshow(equl,'gray')

plt.title('Equalized Image')

plt.show()

histr2 = cv2.calcHist([equl],[0],None,[256],[0,256])

plt.plot(histr2)

plt.xlabel('Intensity value')

plt.ylabel('Number of pixels')

plt.title('Equalised histogram')

plt.show()

1. **Histogram matching**

import cv2

import matplotlib.pyplot as plt

from skimage.exposure import match\_histograms

from skimage import exposure

# Reading main image

img1 = cv2.imread(r"C:\Users\hp\Downloads\girl.jpg")

# Checking the number of channels

print('No of Channel is: ' + str(img1.ndim))

# Reading reference image

img2 = cv2.imread(r"C:\Users\hp\Downloads\rose.jpg")

# Checking the number of channels

print('No of Channel is: ' + str(img2.ndim))

image = img1

reference = img2

# Matching histograms with updated argument

matched = match\_histograms(image, reference, channel\_axis=-1)

fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3,

figsize=(8, 3),

sharex=True, sharey=True)

for aa in (ax1, ax2, ax3):

aa.set\_axis\_off()

ax1.imshow(cv2.cvtColor(image, cv2.COLOR\_BGR2RGB))

ax1.set\_title('Source')

ax2.imshow(cv2.cvtColor(reference, cv2.COLOR\_BGR2RGB))

ax2.set\_title('Reference')

ax3.imshow(cv2.cvtColor(matched, cv2.COLOR\_BGR2RGB))

ax3.set\_title('Matched')

plt.tight\_layout()

plt.show()

fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(8, 8))

for i, img in enumerate((image, reference, matched)):

for c, c\_color in enumerate(('red', 'green', 'blue')):

img\_hist, bins = exposure.histogram(img[..., c],

source\_range='dtype')

axes[c, i].plot(bins, img\_hist / img\_hist.max())

img\_cdf, bins = exposure.cumulative\_distribution(img[..., c])

axes[c, i].plot(bins, img\_cdf)

axes[c, 0].set\_ylabel(c\_color)

axes[0, 0].set\_title('Source')

axes[0, 1].set\_title('Reference')

axes[0, 2].set\_title('Matched')

plt.tight\_layout()

plt.show()