Experiment No 08 Image Enhancement -01

### **Name: Rebecca Dias Roll No.: 19 Class: BE CMPN A PID: 182027**

# Aim:

To perform the image enhancement techniques of Image Negative, Gray Level Slicing and Thresholding of an image signal in Python

# Theory:

Image enhancement is the process of adjusting digital images so that the results are more suitable for display or further image analysis. Image Enhancement can be done in two domains: Spatial Domain and Frequency Domain. Spatial domain enhancement involves direct manipulation of pixels of the image. Spatial domain enhancement can be carried out in two different ways: point processing and neighborhood processing.

# Point Processing:

Point processing uses only the information in individual pixels to produce new images. A transform may be computed on the basis of regional or global information and then applied to the individual points.

Some of the point processing techniques are

* Digital Negative
* Thresholding
* Gray Level Slicing
* Contrast Stretching
* Bit plane Slicing
* Dynamic Range Compression
* Power Law Transformation

# Digital Negative:

Image negative is produced by subtracting each pixel from the maximum intensity value. Thus, the transformation function used in image negative is **s = T(r) = L – 1 – r**

where L-1 is the max intensity value and s, and r are the output and input pixel values respectively.

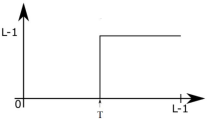
For an 8-bit gray level image, the negative transformation is:

# s = L – 1 – r = 256 – 1 – r = 255 – r

**Thresholding:**

The simplest thresholding methods replace each pixel in an image with a black pixel if the image intensity I{i,j} is less than some fixed constant T or a white pixel if the image intensity is greater than that constant. Single value thresholding can be represented mathematically as:

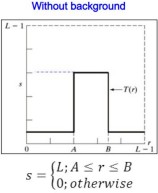


given as

# Gray Level Slicing:

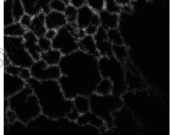
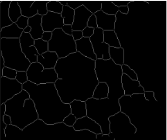
This technique is used to highlight a specific range of gray levels in a given image. It can be implemented in several ways, but the two basic approaches are:

# Approach 1:

First approach is to display a high value for all gray levels in the range of interest and a low value for all other gray levels. This transformation is given as follows:

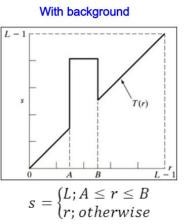
It produces a binary image.

# Example:

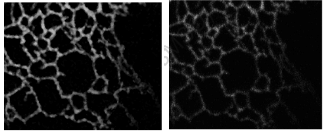
 

# Approach 2:

The second approach is based on the transformation which brightens the desired range of gray levels but preserves gray levels unchanged. This transformation is given as follows:



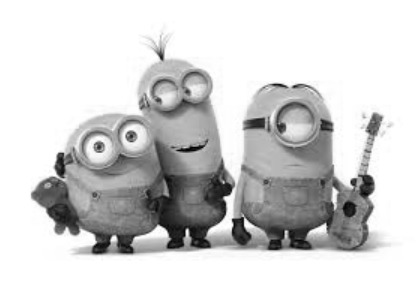
# Example:

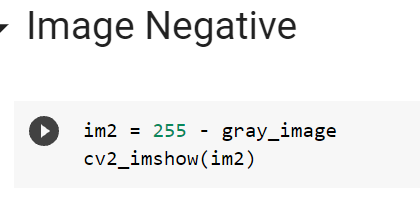


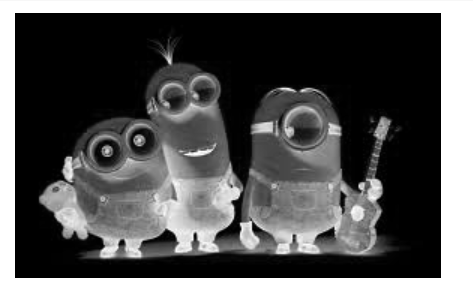
**Programming Exercises in Python**

1. Read the photo of your image.
2. Convert it into grayscale
3. Determine and show the negative of your image
4. Determine and show thresholding of your image
5. Do gray-level slicing of your image with and without background

|  |
| --- |
| import numpy as np  import pandas as pd  import cv2 as cv  from google.colab.patches import cv2\_imshow  from skimage import io  from PIL import Image  import matplotlib.pylab as plt  url = "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQli9qehh6pRJSo9X0RHF3aWEBPHw0Ef8rh-Q&usqp=CAU"  image = io.imread(url)  gray\_image = cv.cvtColor(image, cv.COLOR\_BGR2GRAY)  cv2\_imshow(gray\_image)  m,n = gray\_image.shape |

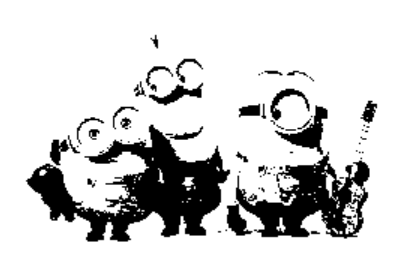






Thresholding

|  |
| --- |
| ret,th1 = cv.threshold(gray\_image,127,255,cv.THRESH\_BINARY)  cv2\_imshow(th1) |



# Gray level Slicing : Without Background

|  |
| --- |
| # the lower threshold value  T1 = 100  # the upper threshold value  T2 = 180  # create a array of zeros  img\_thresh\_noback = np.zeros((m,n), dtype = int)  for i in range(m):      for j in range(n):          if T1 < gray\_image[i,j] < T2:              img\_thresh\_noback[i,j]= 255          else:              img\_thresh\_noback[i,j] = 0    # Convert array to  png image  cv.imwrite('Thresh\_Back.png', img\_thresh\_noback)  cv2\_imshow(img\_thresh\_noback) |

# 

# Gray level Slicing : With Background

|  |
| --- |
| # the lower threshold value  T1 = 100  # the upper threshold value  T2 = 180  # create a array of zeros  img\_thresh\_back = np.zeros((m,n), dtype = int)  for i in range(m):      for j in range(n):          if T1 < gray\_image[i,j] < T2:              img\_thresh\_back[i,j]= 255          else:              img\_thresh\_back[i,j] = gray\_image[i,j]    # Convert array to  png image  cv.imwrite('Thresh\_Back.png', img\_thresh\_back)  cv2\_imshow(img\_thresh\_back) |

# 

# Conclusion:

Studied the image enhancement techniques of negative, thresholding and gray-level slicing of an image signal. Implemented it using the python language and the output was accurately obtained as well.