Name : Sarvagya Singh

SAPID : 60009200030 **BATCH: K1**

AIM: To Perform Image Enhancement(SD) Point Processing Techniques: Digital Negative, Thresholding, Grey Level Slicing with & without background

THEORY:

The principle objective of image enhancement is to process an image so that the result is more suitable than the original image for a specific application. Image enhancement approaches fall into two broad categories:

Spatial domain methods - which involve direct manipulation of pixels in an image

Frequency domain methods - which are based on modifying the Fourier transform of an image.

Point processing techniques are among the simplest of all image enhancement techniques. These techniques are basically zero memory operations where , every pixel is operated individually without the use of neighbouring pixels.

The values of pixels, before and after processing, will be called r and s, respectively. These values are related by an expression of the form

$$s = T(r)$$

where T is a transform function, it maps the pixel value r in to a pixel values.

1. Image Negatives

The negative of an image with gray levels in the range {0, L-1} is obtained by using the negative transformation given by the expression,

$s = (L – 1) – r$

where,

s = output gray level.

r = input gray level.

L – 1 = Highest gray level present in the image.

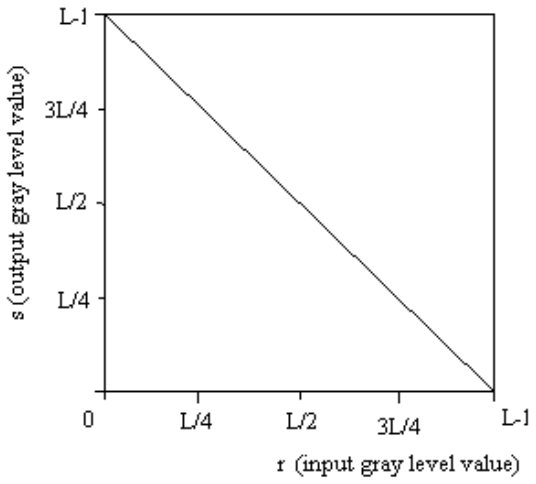


Fig 1. Negative Image Transformation

This type of processing is particularly suited for enhancing white or gray detail embedded in the dark regions of an image, especially when the black areas are dominant in size.

2.Thresholding

The input to a thresholding operation is typically a grayscale image. In the simplest implementation, the output is a binary image representing the segmentation. Black pixels correspond to background and white pixels correspond to foreground (or *vice versa*).

For example, a simple mapping function is defined by the thresholding function. In contrast stretching if a = b , v = 0 & w = L - 1, the transformation function is such that ,

s = 0 for r < a

L – 1 for r >= a

The figure below shows the transformation function:

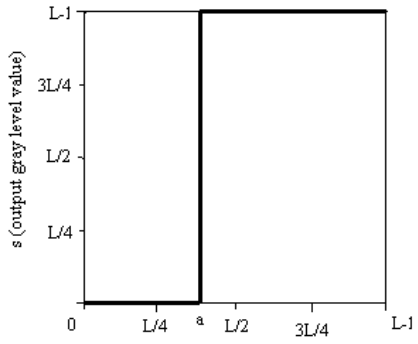


Fig 3. Thresholding

In simple implementations, the segmentation is determined by a single parameter known as the *intensity threshold*. In a single pass, each pixel in the image is compared with this threshold. If the pixel's intensity is higher than the threshold, the pixel is set to, say, white in the output. If it is less than the threshold, it is set to black.

3. Gray Level Slicing

Highlighting a specific range of gray levels in an image is often desired. Gray level slicing has two basic approaches for enhancement:

One 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 produces a binary image.

Another approach brightens the desired range of gray levels but preserves the background and gray level tonalities in the image.

gray level slicing without background		gray level slicing with background	
s = L-1	a<= r < b	s = L-1	a< r <=b
s = 0	otherwise	s = r	otherwise

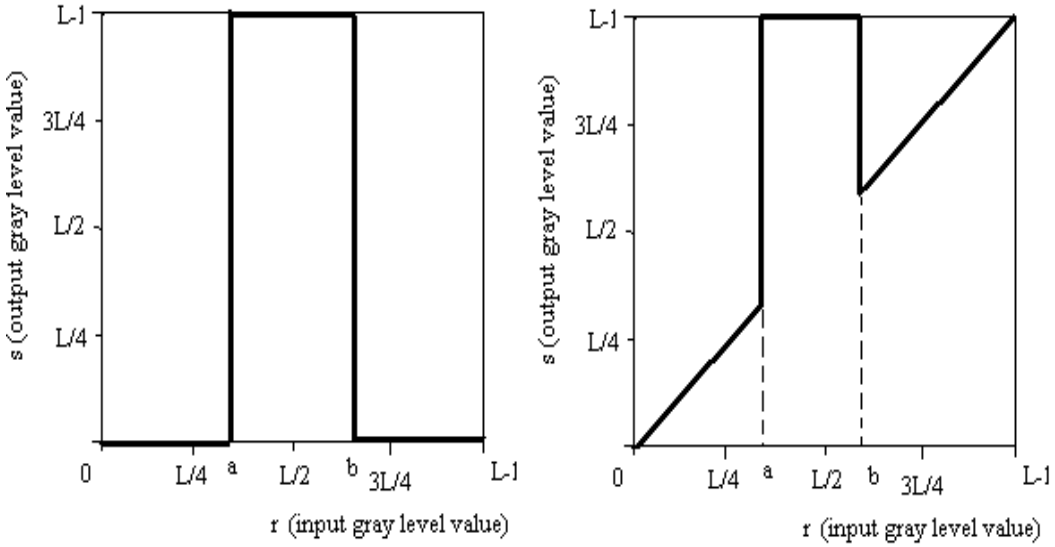


Fig 4. The gray level slicing operation (left), and with background operation (right).

RESULT:

Digital Negative

Sarvagya Singh

60009200030 - K1

IPCV -- lab2

In [1]:
```
!pip install opencv-python
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: opencv-python in /usr/local/lib/python3.8/dist-packages (4.6.0.66)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.8/dist-packages (from opencv-python) (1.22.4)
```

In [2]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

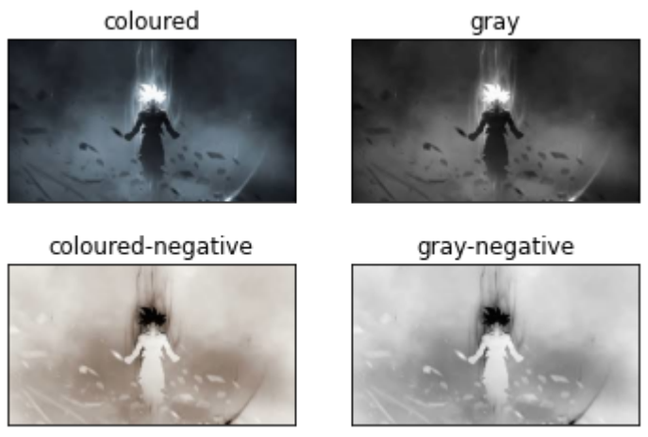
In [3]:
```
link = '/content/goku.jpeg'
image = cv2.imread(link)
```

In [4]:
```
img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

In [5]:
```
gray = cv2.imread(link, 0)
```

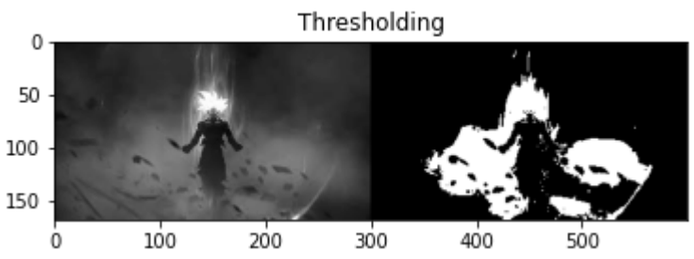
In [6]:
```
colored_negative = abs(255-img)
gray_negative = abs(255-gray)
```

In [7]:
```
imgs = [img, gray, colored_negative, gray_negative]
title = ['coloured', 'gray', 'coloured-negative', 'gray-negative']


plt.subplot(2, 2, 1)
plt.title(title[0])
plt.imshow(imgs[0])
plt.xticks([])
plt.yticks([])

plt.subplot(2, 2, 2)
plt.title(title[1])
plt.imshow(imgs[1], cmap='gray')
plt.xticks([])
plt.yticks([])

plt.subplot(2, 2, 3)
plt.title(title[2])
plt.imshow(imgs[2])
plt.xticks([])
plt.yticks([])

plt.subplot(2, 2, 4)
plt.title(title[3])
plt.imshow(imgs[3], cmap='gray')
plt.xticks([])
plt.yticks([])

plt.show()
```



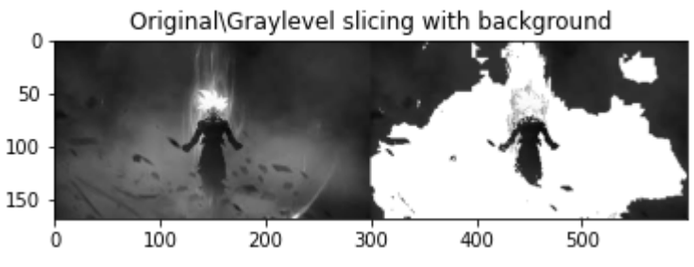
Thresholding

In [11]:
```
image = cv2.imread(link, 0)
x,y=image.shape
threshold = 100
z=np.zeros((x,y))
for i in range(0,x):
    for j in range(0,y):
        if(image[i][j]>threshold):
            z[i][j]=255
        else:
            z[i][j]=0
equ=np.hstack((image,z))
plt.title('Thresholding')
plt.imshow(equ,'gray')
plt.show()
```



Gray Level Slicing

In [12]:
```
import numpy as np
image=cv2.imread(link,0)
x,y=image.shape
z=np.zeros((x,y))
for i in range(0,x):
    for j in range(0,y):
        if(image[i][j]>50 and image[i][j]<150):
            z[i][j]=255
        else:
            z[i][j]=image[i][j]
equ=np.hstack((image,z))
plt.title('Original\Graylevel slicing with background')
plt.imshow(equ,'gray')
plt.show()
```



In [13]:
```
image=cv2.imread(link,0)
x,y=image.shape
z=np.zeros((x,y))
for i in range(0,x):
    for j in range(0,y):
        if(image[i][j]>50 and image[i][j]<150):
            z[i][j]=255
        else:
            z[i][j]=0
equ=np.hstack((image,z))
plt.title('Original\Graylevel slicing without background')
plt.imshow(equ,'gray')
plt.show()
```