**CLASS : B.E. E &TC** **SUBJECT: DIVP**

**EXPT. NO. : 10** **DATE: 27/11/2020**

---

**TITLE : ENHANCE A GRAY IMAGE USING PSEUDO COLORING**

---

| | |
|---|---|
| **CO 1:** | Apply the fundamentals of digital image processing to perform various operations on an image-enhancement in spatial domain/ frequency domain, image-restoration, image compression, video filtering and video compression on a given gray image. Examine the effect of varying the mask size and density of noise in an image and comment on the obtained results. |
| **CO4:** | Carry out experiments as an individual and in a team, comprehend and write a laboratory record and draw conclusions at a technical level. |

**AIM:**

**To implement:**

- **To load gray image**
- **To perform histogram equalization on the gray image**
- **Assigning a specific color to a range of pixels have same gray intensity level**

**SOFTWARES REQUIRED:** Matlab 7.0 or above.

**THEORY:**

**10.1 Pseudo coloring**

Pseudo-color processing is a technique that maps each of the grey levels of a black and white image into an assigned color. This colored image, when displayed, can make the identification of certain features easier for the observer. The mappings

are computationally simple and fast. This makes pseudo-color an attractive technique for use on digital image processing systems that are designed to be used in the interactive mode.

## 10.2 Intensity slicing

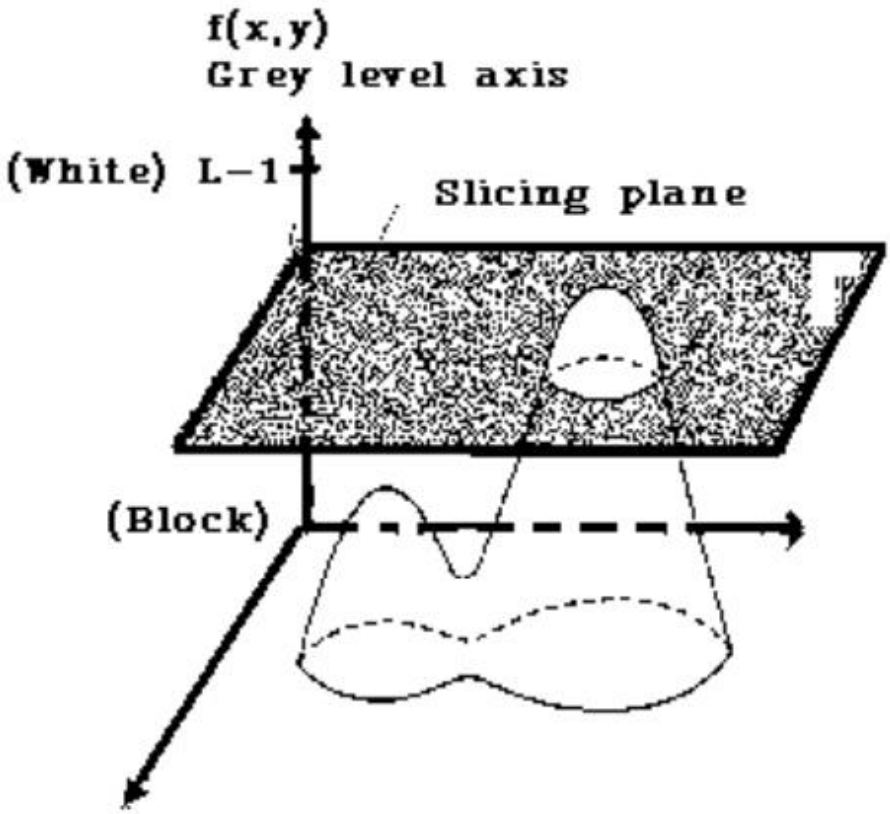


**Fig. 1: Geometric interpretation of the intensity-slicing technique**

The technique of density also called intensity slicing and color coding is the simplest example of pseudo color image processing. If an image is interpreted as a 3D function[1][2], the method can be viewed as one of placing planes parallel to the coordinate plane of the image. Each plane then slices the function in the area of intersection. The above figure shows a plane at f(x, y) = Li to slice the image function into two levels. I a different color is assigned to each side of the plane shown in Fig.1, any pixel whose gray scale is

above the plane will be coded with one color, and any pixel below the plane will be coded with the other. The result is a two-color image whose relative appearance can be controlled by moving the slicing plane up and down the gray level axis. The idea of planes is useful primarily for a geometric interpretation of the intensity-slicing technique. When more levels are used, the mapping function takes on a staircase form.

## 10.3 Gray Level to Color Transformations

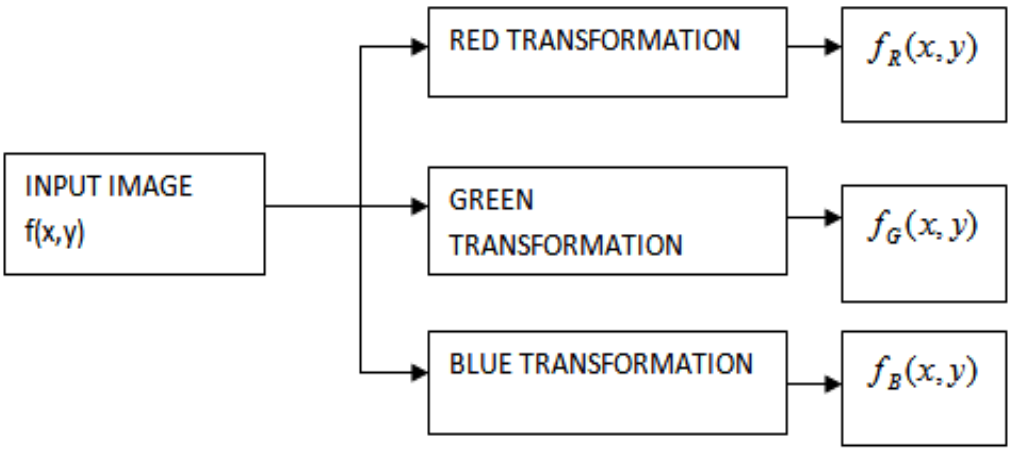


**Fig 2: Functional block diagram for pseudo color image processing $f_R$, $f_G$, $f_B$ are fed into the corresponding Red, Green, and Blue inputs of an RGB color monitor**

There are many types of transformations and are capable of achieving a wider range of pseudo color enhancement results than the simple slicing technique discussed in the preceding section. An approach that is particularly attractive is shown in Fig.2. Basically the idea underlying this approach is to perform three independent transformations on the gray level of any input pixel. The three results are then fed separately into the red, green, and blue channels of a color television monitor. This method produces a composite image,

whose color content is modulated by the nature of the transformations on the gray-level values of an image, and is not functions of position.

### 10.4 Need of pseudo coloring

Pseudo coloring results in better identification of these calcifications. The term Pseudo-color refers to using color to visualize things that aren't inherently colored. If two variables of information have to be displayed at a time in an image, the Proceedings of the International Conference on Cognition and Recognition 755 intensity (one variable) and the hue (another variable) may be varied independently. Pseudo -color can be effectively used to label or identify particularly significant sections of the image as determined by some other image processing or vision algorithm. It has to be kept in mind while using a color that a large portion of the population (estimated 10% of all males) is color blind. If isoluminant displays are created (one with uniform intensity and varying hues), color blind people can't see the displays. Normal sighted people will see them poorly.

### 10.5 Conclusion:

By means of this experiment, I have learnt about the principle of pseudo coloring well as its applications. I also implemented the code for conversion of Grayscale image to color image using Pseudo coloring technique.

### 1.5 References:

i. Gonzalez R, Woods R, "Digital image processing", Pearson Prentice Hall, 2008.

ii. Gonzalez R, Woods R, Steven E, "Digital Image Processing Using MATLAB®", McGraw Hill Education,2010.

iii. https://pdfs.semanticscholar.org/edd3/72b3bce9968b8d8f93d2ffb52df3f5b96343.pdf

_____

**(Course Teacher)**

| | | | | | |
|---|---|---|---|---|---|
| **CLASS** | **:** | **B.E (E &TC)** | **COURSE** | **:** | **DIVP** |
| **AY** | **:** | **2020-21 (SEM- I)** | **DATE** | **:** | **27/11/2020** |
| **EXPT. NO.** | **:** | **10** | **CLASS & ROLL NO** | **:** | **BE VIII 42410** |
| **TITLE** | **:** | **ENHANCE A GRAY IMAGE USING PSEUDO COLORING** | | | |

## I. CODE:

```
# -*- coding: utf-8 -*-
import cv2
import numpy as np
from matplotlib import pyplot as plt

def pseudoColor(img,row,col):
    img1 = np.zeros(shape=(row,col,3) ,dtype = 'uint8')
    for i in range(row):
        for j in range(col):
            if img[i,j]>0and img[i,j]<50:
                img1[i,j,0] = img[i,j]+125
                img1[i,j,1] = img[i,j]
                img1[i,j,2] = img[i,j]
            elif img[i,j]>50 and img[i,j]<100:
                img1[i,j,0] = img[i,j]
                img1[i,j,1] = img[i,j]+125
                img1[i,j,2] = img[i,j]
            elif img[i,j]>100 and img[i,j]<150:
                img1[i,j,0] = img[i,j]
                img1[i,j,1] = img[i,j]
                img1[i,j,2] = img[i,j]+155
            elif img[i,j]>150 and img[i,j]<200:
                img1[i,j,0] = img[i,j]
                img1[i,j,1] = img[i,j]
                img1[i,j,2] = img[i,j]+125
            elif img[i,j]>200 and img[i,j]<215:
                img1[i,j,0] = img[i,j]
                img1[i,j,1] = img[i,j]
                img1[i,j,2] = img[i,j]+200
            elif img[i,j]>215 and img[i,j]<225:
                img1[i,j,0] = img[i,j]+255
                img1[i,j,1] = img[i,j]
                img1[i,j,2] = img[i,j]
            elif img[i,j]>225 and img[i,j]<235:
                img1[i,j,0] = img[i,j]
                img1[i,j,1] = img[i,j]+255
                img1[i,j,2] = img[i,j]
            elif img[i,j]>235 and img[i,j]<245:
                img1[i,j,0] = img[i,j]
```

```
            img1[i,j,1] = img[i,j]
            img1[i,j,2] = img[i,j]+255
        else:
            img1[i,j,0] = img[i,j]+255
            img1[i,j,1] = img[i,j]
            img1[i,j,2] = img[i,j]
    return img1

def main():
    img = cv2.imread('C:/Users/Danish/Desktop/DIVP/Images/penguin.jpg',0)
    rows,cols=img.shape
    newimg=pseudoColor(img, rows, cols)
    fig = plt.figure(figsize=(30, 18))
    grid = plt.GridSpec(2, 2, wspace=0.2, hspace=0.5)

    plt.subplot(2, 3, 1), plt.imshow(img, 'gray')
    plt.subplot(2, 3, 2), plt.imshow(newimg, 'gray')
    plt.subplot(2, 3, 4),plt.hist(img.ravel(),256,[0,256])
    color = ('b', 'g', 'r')
    plt.subplot(2, 3, 5)
    for i, col in enumerate(color):
        histr = cv2.calcHist([newimg], [i], None, [256], [0, 256])
        plt.plot(histr, color = col)
        plt.xlim([0, 256])
    plt.show()
main()
```

## II. RESULTS: