**Department of Computing**

**Digital Image Processing**

**Class: BSCS-9ABC**

**Lab 3**

**Submitted By: Fatima Seemab**

**CMS ID: 291310**

**BSCS 9B**

Lab 4

Primitive Transformations

**Task #1: Image Negatives**

Implement a function for displaying negative of an input image. Note that the function must handle binary, grayscale, and RGB images. Example of RGB negative:



**Code:**

from PIL import Image,ImageOps

#For processing image class PIL is imported from library PIL

im=Image.open('lenna.jpg')

#For opening the image

x,y=im.size

# This will return no of rows and no of cols in an image

print(x,y)

#To convert image to a grayscale having level of quantization between 0 -1

grayscale=ImageOps.grayscale(im)

#To iterate over all the rows

for i in range (0,x):

#To iterate over all the cols in an image

for j in range (0,y):

cordinate=i,j

# finding the intensity of current pixel

value=grayscale.getpixel((cordinate))

#For calculating negative/complement of a pixel

s1=255-value

newcoordinates=s1

# Pixel is assigned the complemented value

negative=grayscale.putpixel((cordinate),(newcoordinates))

#For displaying modified image

grayscale

**Output:**

A person wearing a hat

Description automatically generated with low confidence

**Code:**

################For rgb #########################################

from PIL import Image

#For processing image class PIL is imported from library PIL

im=Image.open('lenna.jpg')

#For opening the image

x,y=im.size

# This will return no of rows and no of cols in an image

#To iterate over all the rows

for i in range (0,x):

#To iterate over all the cols in a row

for j in range (0,y):

cordinate=i,j

# finding the intensity of current pixel

r,g,b=im.getpixel((cordinate))

#For calculating negative/complement of a pixel

s1=255-r;

s2=255-g

s3=255-b

newcoordinates=s1,s2,s3

# Pixel is assigned the complemented value

negative=im.putpixel((cordinate),(newcoordinates))

im

**Output:**

A picture containing text

Description automatically generated

**Code:**

#####################Tak 1#########################

############Binary Negative #######################

from PIL import Image,ImageOps

#For processing image class PIL is imported from library PIL

im=Image.open("lenna.jpg")

#For opening the image

x,y=im.size

#This will return no of rows and no of cols in an imag

#Converting RGB to grayscale image

im=ImageOps.grayscale(im)

for i in range(x):

#To iterate over all the rows

for j in range(y):

#To iterate over all the columns in a row

cordinate=i,j

# finding the intensity of current pixel

a=(im.getpixel((cordinate)))

#condition when intensity level is

#above 180 new image will

#map to lightest shade otherwise darker

if a>120:

a=265

else:

a=0

threshold=a

#for assigning new values to current pixels

im.putpixel((cordinate),(threshold))

r=im.getpixel((cordinate))

s=255-r

negative=im.putpixel((cordinate),(s))

im

**Output:**

A picture containing text

Description automatically generatedA picture containing text, plant

Description automatically generated

**Task #2: Image Gradients**

The horizontal gradient image can be used to detect vertical edges in an image. Implement a function for displaying the horizontal gradient of a grayscale image. The gradient can be approximated by forward differences:

Note that the gradient values can be both positive and negative! So you need to find a way to display the gradient values in the range: 0, 1, 2, …, 255. The following link can be helpful here:

<https://www.cis.rit.edu/people/faculty/rhody/EdgeDetection.htm>

The resulting image should look something like this:

Graphical user interface, application

Description automatically generated

***Code:***

from PIL import Image,ImageOps

#For processing image class PIL is imported from library PIL

import numpy

im=Image.open('lenna.jpg')

#For opening the image

x,y=im.size

#This will return no of rows and no of cols in an image

print(x,y)

arr=numpy.zeros((x,y))

im=ImageOps.grayscale(im)

#To iterate over all the rows

for i in range (x):

#To iterate over all the cols in an image

for j in range (y):

cordinate=i,j

# For clipping values

if i+1<225:

gradientcoordinate=i+1,j

a=im.getpixel((i,j))

b=im.getpixel((i+1,j))

# subtracting to get intensity given by equation

c=b-a

if c<0:

c=abs(c)

#Putting the modified value to an image.

im.putpixel((i,j),(c))

im

***Output :***

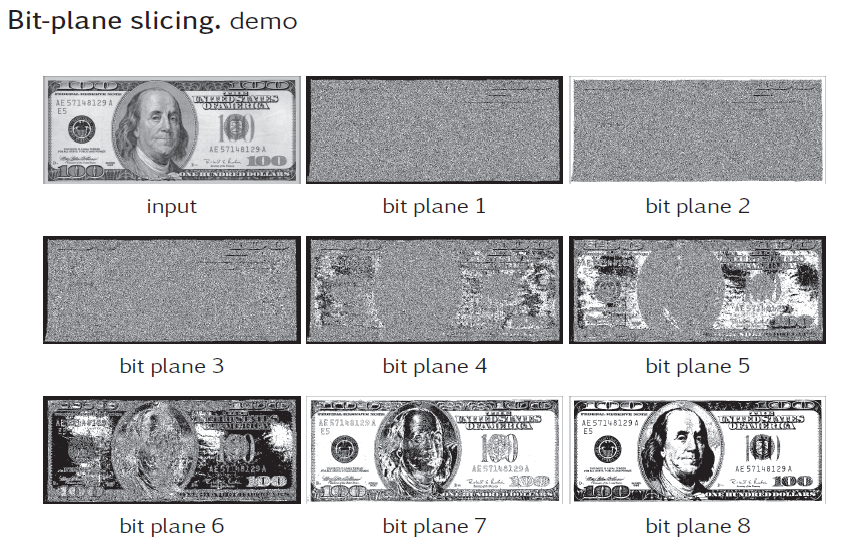
A picture containing text, dark

Description automatically generatedA person wearing a hat

Description automatically generated with medium confidence

**Task #3: Bit Plane Slicing**

Perform bit slicing of an 8 bit greyscale image as discussed in the lecture. Start from the least significant bit and move towards the most significant bit. You will get eight binary images of the input image as demonstrated below.



***Code:***

from PIL import Image

#For processing image class PIL is imported from library PIL

import sys

import numpy

import imageio

#For opening the image

im=Image.open('bitplane.tif')

x,y=im.size

arr\_size=y\*x

# initializing arrays to contain bit plane slices

arr1=numpy.zeros((y,x))

arr2=numpy.zeros((y,x))

arr3=numpy.zeros((y,x))

arr4=numpy.zeros((y,x))

arr5=numpy.zeros((y,x))

arr6=numpy.zeros((y,x))

arr7=numpy.zeros((y,x))

arr8=numpy.zeros((y,x))

# To converting image to array

img=numpy.array(im)

print(x,y)

#To iterate over all the rows

for j in range (0,y):

#To iterate over all the columns

for i in range (0,x):

cordinate=j,i

# converting to uint format to be used in unpack

intensity=numpy.uint8(img[j][i])

# finding the intensity of current pixel.

binary=numpy.unpackbits(intensity)

# Assigning bit values in binary format to corresponding arrays

arr1[j][i]=binary[7]

arr2[j][i]=(binary[6])

arr3[j][i]=(binary[5])

arr4[j][i]=(binary[4])

arr5[j][i]=(binary[3])

arr6[j][i]=(binary[2])

arr7[j][i]=(binary[1])

arr8[j][i]=(binary[0])

# To converting array to images and save all the images in directory

imageio.imwrite('image1.png',arr1)

imageio.imwrite('image2.png',arr2)

imageio.imwrite('image3.png',arr3)

imageio.imwrite('image4.png',arr4)

imageio.imwrite('image5.png',arr5)

imageio.imwrite('image6.png',arr6)

imageio.imwrite('image7.png',arr7)

imageio.imwrite('image8.png',arr8)

***Outputs:***

***Bit Plane 1***

A picture containing furniture

Description automatically generated

***Bit Plane 2***

A picture containing furniture, rug

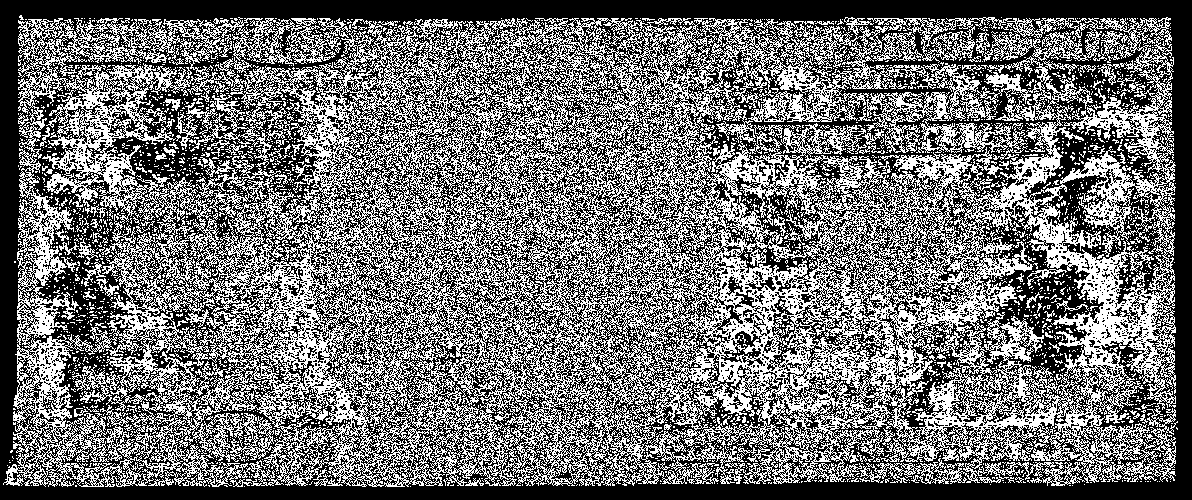
Description automatically generated

***Bit Plane 3***

A picture containing furniture, rug

Description automatically generated

***Bit Plane 4***

******

***Bit Plane 5***

A picture containing text

Description automatically generated

***Bit Plane 6***

A picture containing text

Description automatically generated

***Bit Plane 7***

A close-up of some money

Description automatically generated with low confidence

***Bit Plane 8***

A close-up of a dollar bill

Description automatically generated with medium confidence