CMPS327 IMAGE PROCESSING

Revision Session

IMAGE PROCESSING

- *This lab is a review, going over* ***some*** *examples from all previous labs. full whereas all lab content is required for the final.*

LIBRARIES AND BASICS

1. Import OpenCV, Pillow and NumPy

____

____

____

Which Library Allows for opening, manipulating, and saving various image file formats? __________

1. Open the image 02.jpeg located in Images file in local disk C and transform it into grayscale and then display under the title “Lena Image”:

____

____

____

LIBRARIES AND BASICS

- Import OpenCV, Pillow and NumPy

```
import cv2
from PIL import Image
import numpy as np
```

- Which Library Allows for opening, manipulating, and saving various image file formats? OpenCV, Pillow
- Open the image 02.jpeg located in Images file in local disk C and transform it into grayscale and then display under the title “Lena Image”:

```
image = Image.open(‘c:/Images/02.jpeg').convert('L')
display(image," Lena Image")
```

TRANSFORMATION

1. Complete the following function of Normalization given by the expression:

$$s = 255 * \frac{r - min}{max - min}$$

```
def linear(x,maxi,mini):
    ________________________
    ________________________
    ________________________
    ________________________
    return x
```

TRANSFORMATION

1. Complete the following function of Normalization given by the expression:

$$s = 255 * \frac{r - min}{max - min}$$

```
def linear(x,maxi,mini):
        x1 = x-mini
         x1 = 255*x1
        d = maxi-mini
        x = x1/d
        return x
```

TRANSFORMATION

1. Write the following function of Power – Law transformations given by the expression:

$$s = cr^{\gamma}$$

TRANSFORMATION

1. Write the following function of Power – Law transformations given by the expression:

$$s = cr^{\gamma}$$

```
def power(c,gamma,r):
  return c*r**gamma
```

FILTERING

$$\hat{f}(m,n) = \frac{1}{ab} \sum_{S_{ab}} g(s,t)$$

- Apply the Arithmetic mean filtering,
- Let Sab be a rectangular window of size a=3 * b=3. The arithmetic mean filter computes the average value of the pixels in g(m,n) over the window Sab.

```
data = np.array(image)
n=_____

____________________
for i in range(______, ______):
    for j in range(______, ______):
        value = 0
        for l in range(______, ______):
            for w in range(______, ______):
                _____________________________
        ____________________
        data_result[i][j]= value
```

1. Create Pillow Image after arithmetic mean:

   image2 = ________________________

FILTERING

$$\hat{f}(m,n) = \frac{1}{ab}\sum_{S_{ab}} g(s,t)$$

- Apply the Arithmetic mean filtering,
- Let Sab be a rectangular window of size a=3 * b=3. The arithmetic mean filter computes the average value of the pixels in g(m,n) over the window Sab.

```
data = np.array(image)
n= 3
n2=pow(n,2)
for i in range(int(n/2), row-int(n/2)):
    for j in range(int(n/2), col-int(n/2)):
        value = 0
        for l in range(-1*int(n/2),int(n/2)+1):
            for w in range(-1*int(n/2),int(n/2)+1):
                value += int(data[i+l][j+w])
        value = int(value/n2)
        data_result[i][j]= value
```

1. Create Pillow Image after arithmetic mean:

```
image2 = image.fromarray(data_result)
```

IMAGE RESTORATION IN PRESENCE OF NOISE ONLY

1. **Given the Contra-Harmonic Mean Filter:**

It is well suited for reducing the effects of salt-and-pepper noise, if Q>0 then it is used for elimination of pepper noise and if Q<0 then for elimination of salt noise

$$\hat{f}(x,y)=\frac{\sum_{(s,t)\in S_{x,y}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{x,y}} g(s,t)^{Q}}$$

Define **Value1** and **Value2** in terms of:

```
data = np.array(image)
n= 5
Q=-1
for i in range….
    for j in range….
        Value1 = 0
        Value2 = 0
        for l in range….
            for w in range…
          Value1 ______________________

          Value2 ______________________
```

IMAGE RESTORATION IN PRESENCE OF NOISE ONLY

$$\hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{x,y}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{x,y}} g(s,t)^{Q}}$$

1. **Given the Contra-Harmonic Mean Filter:**

It is well suited for reducing the effects of salt-and-pepper noise, if Q>0 then it is used for elimination of pepper noise and if Q<0 then for elimination of salt noise

Define **Value1** and **Value2** in terms of:

```
data = np.array(image)
n= 5
Q=-1
for i in range (int(n/2), row-int(n/2)):
   for j in range(int(n/2), col-int(n/2)):
        Value1 = 0
        Value2 = 0
        for l in range(-1*int(n/2),int(n/2)+1):
            for w in range(-1*int(n/2),int(n/2)+1):
           Value1 += int(data[i+l,j+w])**(Q+1)

           Value2 +=int(data[i+l,j+w])**Q
```

EDGE DETECTION

Prewitt operator

a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. At each point in the image, the result of the Prewitt operator is either the corresponding gradient vector or the normal of this vector. The Prewitt operator is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction and is therefore relatively inexpensive in terms of computations.

$\boldsymbol{Mx_{3x3}} = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$	$\boldsymbol{My_{3x3}} = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$

EDGE DETECTION

Prewitt operator

In each edge detection the kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (Gx and Gy). These can then be combined together to find the absolute magnitude of the gradient at each point. The gradient magnitude is given by:

$$|\boldsymbol{G}(\boldsymbol{x},\boldsymbol{y})| = \sqrt{\boldsymbol{G_x}^2 + \boldsymbol{G_y}^2}$$

After computing the gradient magnitude, the threshold can be applied as follows:

$$\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) = \begin{cases} 255, & for \ \left||\boldsymbol{G}(\boldsymbol{x},\boldsymbol{y})|\right| \geq T \\ 0, & for \ \left||\boldsymbol{G}(\boldsymbol{x},\boldsymbol{y})|\right| < T \end{cases}$$

EDGE DETECTION

Prewitt operator

Complete the edge_detection method to compute the **prewitt** operator taking into consideration the threshold:

```
def edge_detection(img,mask,threshold,withSharp):
    maskX = ________
    maskY = ________
    # load the image
    n=np.shape(mask)[0]
    # convert image to numpy array
    data = np.array(image)
    data_result = np.array(image)
    row,col = _______________
 for i in range…..
       for j in range….
          valueX = 0
          valueY = 0
          for l in range….
             for w in range….
                valueX += ______________________________
                valueY += ______________________________
          value = __________________

          #put threshold condition
          ________________
          ________________
          ________________
          ________________
             if withSharp==True:
                data_result[i,j] = _______________
             else:
                data_result[i,j] = value

 # create Pillow image..
```

Write Main Method of image 11.jpeg in images folder in local disk C with 100 threshold and sharp is False

#define prewitt matrix:

____

#call function edge_detection:

____

EDGE DETECTION

Prewitt operator

Complete the edge_detection method to compute the **prewitt** operator taking into consideration the threshold:

```
def edge_detection(img,mask,threshold,withSharp):
    maskX = mask
    maskY = np.transpose(mask)
    # load the image
    n=np.shape(mask)[0]
    # convert image to numpy array
    data = np.array(image)
    data_result = np.array(image)
    row,col = data.shape
  for i in range(int(n/2), row-int(n/2)):
        for j in range(int(n/2), col-int(n/2)):
            valueX = 0
            valueY = 0
            for l in range(-1*int(n/2),int(n/2)+1):
                for w in range(-1*int(n/2),int(n/2)+1):
                    valueX +=int(data[i+l,j+w])*maskX[l+int(n/2)][w+int(n/2)]
                    valueY += int(data[i+l,j+w])*maskY[l+int(n/2)][w+int(n/2)]
           value = = int(np.sqrt(valueX**2+valueY**2))
#put threshold condition
if value<threshold:
        value = 0
 else:
        value = 255
if  withSharp==True:
            data_result[i,j] = value +data_result[i,j]
else:
            data_result[i,j] = value
# create Pillow image..
```

Write Main Method of image 11.jpeg in images folder in local disk C with 100 threshold and sharp is False

```
#define prewitt matrix:
prewitt3  = [[-1,-1,-1],[0,0,0],[1,1,1]]
#call function edge_detection:
edge_detection('C:/Images/11.jpg',prewitt3,100,False)
```

IMAGE SEGMENTATION

We consider an image $\boldsymbol{f(x,y)}$ and we want to apply some application in order to produce a segmented image.

Thresholding:

- ✓ *Thresholding* separates an image $f(x,y)$ into two meaningful regions: background and object, using optimal threshold value t_0.

$$\text{Thresholded image } g(x,y) = \begin{cases} 255 & if\ f(x,y) >= t_0 \\ 0 & if\ f(x,y) < t_0 \end{cases}$$

Let h(i) be the image histogram.

i is the gray-level between [0...255].

For the bimodal thresholding, image pixels will be divided into two classes:

C_1 {0,1,..., t}, and C_2{t,,..., 255}, where *t* is the threshold value.

C_1 that represents the class of dark pixels (Background)

C_2 that represents the class of bright pixels (Object)

The mean of the Classes C1, and C2 if we want segmented the image belong two classes only is:

$$m_1 = \frac{\sum_{i=0}^{t_0} h(i)*i}{\sum_{i=0}^{t_0} h(i)} \qquad m_2 = \frac{\sum_{i=t_0}^{255} h(i)*i}{\sum_{i=t_0}^{255} h(i)} \qquad t_{new} = (m_1 + m_2)/2$$

HISTOGRAM

1. Write a method called **histogram** that compute the histogram of an image.

```
____________________ // define function
    hist = np.zeros(256)
    row,col = data.shape
    for i in range(0, row):
        for j in range(0, col):
            ____________________________
    return hist
```

HISTOGRAM

1. Write a method called **histogram** that compute the histogram of an image.

```
def histogram(data):  // define function
    hist = np.zeros(256)
    row,col = data.shape
    for i in range(0, row):
        for j in range(0, col):
            hist[int(data[i,j])]+=1
   return hist
```

BIMODEL-THRESHOLDING

1. Write a method called **bimodelthreshold** to compute the optimal threshold of bimodal histogram.

```
def bimodelthreshold(data):

    row,col = __________
    hist = _______________
    mini = 255
    maxi = 0
    for i in range(0, row):
        for j in range(0, col):
             value = int(data[i,j])
            mini = min(mini,value)
            maxi = __________________
    prevT=____________________
    t=1
```

```
while abs(________)>1:
        avg1 = 0
        count1 = 0
        for i in range(________):
            avg1 += hist[i]*i
            count1 += hist[i]
        avg1 /= count1
        #compute value for avg2
        __________________
        _________________
        _________________
        _________________

        # compute the optimal threshold of bimodal histogram.
        prevT= _____
        t=__________
return t
```

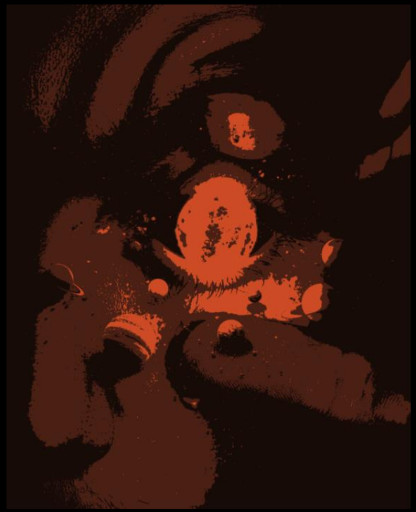
BIMODEL-THRESHOLDING

1. Write a method called **bimodelthreshold** to compute the optimal threshold of bimodal histogram.

```
def bimodelthreshold(data):

    row,col = data.shape
    hist = histogram(data)
    mini = 255
    maxi = 0
    for i in range(0, row):
        for j in range(0, col):
             value = int(data[i,j])
            mini = min(mini,value)
            maxi = max(maxi,value)
    prevT=(mini + maxi)/2
    t=1

while abs(prevT- t)>1:
        avg1 = 0
        count1 = 0
        for i in range(0, prevT):
            avg1 += hist[i]*i
            count1 += hist[i]
        avg1 /= count1
        #compute value for avg2
       avg2 = 0
      count2 = 0
      for i in range(prevT,256):
         avg2 += hist[i]*i
         count2 += hist[i]
      avg2 /= count2

        # compute the optimal threshold of bimodal histogram.
        prevT= t
        t= (avg1+avg2)/2
```

IMAGE SEGMENTATION WITH CLUSTERING

Original Image



Segmented image after Clustering

IMAGE SEGMENTATION WITH CLUSTERING

How Many Clusters were applied in the segmentation **?** ____________

Which below line is responsible for disabling (turning the pixels into black) the pixels belonging to a cluster?

a) masked_image_1 = np.copy(image)
b) masked_image_1 = masked_image_1.reshape((-1, 3))
    cluster = 0
a) masked_image_1[labels == cluster] = [0, 0, 0]
b) masked_image_1 = masked_image_1.reshape(image.shape)

Answer ==>> Line number (___)

IMAGE SEGMENTATION WITH CLUSTERING

How Many Clusters were applied in the segmentation **?** ***We have 3 colors, so we have 3 clusters***

Which below line is responsible for disabling (turning the pixels into black) the pixels belonging to a cluster?

a) masked_image_1 = np.copy(image)
b) masked_image_1 = masked_image_1.reshape((-1, 3))
c) cluster = 0
masked_image_1[labels == cluster] = [0, 0, 0]
d) masked_image_1 = masked_image_1.reshape(image.shape)

Answer ==>> Line number (c)