**Department of Computing**

**Digital Image Processing**

**Class: BSCS-9ABC**

**Lab 9**

**Submitted By: Fatima Seemab**

**CMS ID: 291310**

**BSCS 9B**

**Program 1:**

**Perform un-sharp masking on the following image. Does this enhance the image? Try with at least two different smoothing methods. How does that affect the results?**

| from PIL import Image,ImageOps  import numpy  import math  import cv2  import IPython.display as display  def smooth(img,filter):  im=cv2.imread(img)  im = cv2.cvtColor(im, cv2.COLOR\_BGR2GRAY)    filter\_one\_side=math.floor(filter/2)  img\_pad = cv2.copyMakeBorder(im, filter\_one\_side, filter\_one\_side, filter\_one\_side, filter\_one\_side, cv2.BORDER\_CONSTANT, (0,0,0))  x=img\_pad.shape[0]  y=img\_pad.shape[1]  img\_output=numpy.zeros((x,y))  print(x,y)  arr\_size=filter\*filter      arr=numpy.zeros(arr\_size)  for i in range(filter\_one\_side,x-filter\_one\_side):  for j in range(filter\_one\_side,y-filter\_one\_side):  f=-math.floor(filter/2)  s=-math.floor(filter/2)  m=0  for k in range (filter):  for l in range (filter):  arr[m]=img\_pad[i+f][j+s]  s=s+1  m=m+1  s=-math.floor(filter/2)  f=f+1  value=int(numpy.mean(arr))  img\_output[i][j]=(value)  return img\_output  def unsharpmasking(input,filter):  im=smooth(input,filter)  filter\_one\_side=math.floor(filter/2)  input=cv2.imread(input)  input = cv2.cvtColor(input, cv2.COLOR\_BGR2GRAY)  input = cv2.copyMakeBorder(input, filter\_one\_side, filter\_one\_side, filter\_one\_side, filter\_one\_side, cv2.BORDER\_CONSTANT, (0,0,0))  x=int(input.shape[0])  y=int(input.shape[1])  img\_mask=numpy.zeros((x,y))  img\_output=numpy.zeros((x,y))  print(x,y)    for i in range(0,x):  for j in range(0,y):  value=input[i][j]-im[i][j]  img\_mask[i][j]=value  img\_output[i][j]=int(0.9\*img\_mask[i][j]+input[i][j])  cv2.imwrite("Lab\_9\_task1.png", img\_output)  cv2.imshow("Lab9\_task1.png", img\_output)  cv2.waitKey(0)  unsharpmasking('unsharpmasking.png',3) |
| --- |

**Input:**

****

**Output:**

****

Unsharp masking produces sharp image then the original.As smoothened version is removed from the original image and added to original image produces more sharp image

**Program 2:**

**An image having salt and pepper noise is given below. Which filtering approach**

**do you consider will suit best to remove the said noise? Apply the approach and**

**show results.**

**Median is the best approach for removing the noise**

| def denoiseMedian(img):  from PIL import Image  import numpy  import IPython.display as display  im=Image.open(img)  x,y=im.size  arr=numpy.zeros(25)  # to iterate all the rows  for i in range(2,x-2):  # to iterate over the columns  for j in range(2,y-2):  # for iterating over the neighbors indices from i-2 to i+2  f=-2  # for iterating over the neighbours idices from j-2 to j+2  s=-2  #counter for array indices  m=0  for k in range (5):  for l in range (5):  arr[m]=im.getpixel((i+f,j+s))  s=s+1  m=m+1  s=-2  f=f+1  # median of the array to denoise  value=int(numpy.median(arr))  im.putpixel((i,j),(value))  display.display(im)  denoiseMedian('saltandpaper.png') |
| --- |

**Output:**

|  |
| --- |

**Program 3:**

| from PIL import Image,ImageOps  import numpy  import math  import cv2  import IPython.display as display  def edges(img,filter):  im=cv2.imread(img)  im = cv2.cvtColor(im, cv2.COLOR\_BGR2GRAY)    filter\_one\_side=math.floor(filter/2)  img\_pad = cv2.copyMakeBorder(im, filter\_one\_side, filter\_one\_side, filter\_one\_side, filter\_one\_side, cv2.BORDER\_CONSTANT, (0,0,0))  x=img\_pad.shape[0]  y=img\_pad.shape[1]  y\_edges=numpy.zeros((x,y))  x\_edges=numpy.zeros((x,y))  print(x,y)  for i in range(filter\_one\_side,x-filter\_one\_side):  for j in range(filter\_one\_side,y-filter\_one\_side):  x\_upaxis=(img\_pad[i-1][j-1]+2\*img\_pad[i-1][j]+img\_pad[i-1][j+1])  x\_downaxis=(img\_pad[i+1][j-1]+2\*img\_pad[i+1][j]+img\_pad[i+1][j+1])  x\_edges[i][j]=x\_upaxis-x\_downaxis  y\_upaxis=(img\_pad[i-1][j-1]+2\*img\_pad[i][j-1]+img\_pad[i+1][j-1])  y\_downaxis=(img\_pad[i-1][j+1]+2\*img\_pad[i][j+1]+img\_pad[i+1][j+1])  y\_edges[i][j]=y\_upaxis-y\_downaxis    return x\_edges,y\_edges  def edgesoutput(input,filter):  x\_edge,y\_edge=edges(input,filter)  filter\_one\_side=math.floor(filter/2)  x=int(x\_edge.shape[0])  y=int(x\_edge.shape[1])    img\_output=numpy.zeros((x,y))    for i in range(0,x):  for j in range(0,y):  value=x\_edge[i][j]+y\_edge[i][j]  img\_output[i][j]=value    cv2.imwrite("Lab\_9task3.png", img\_output)  cv2.imshow("Lab9\_task3.png", img\_output)  cv2.waitKey(0)  edgesoutput('two\_cats.jpg',3) |
| --- |

**Output:**

****