**Department of Computing**

**Digital Image Processing**

**Class: BSCS-9ABC**

**Lab 9**

**Submitted By: Fatima Seemab and** [Mahum Samar](mailto:msamar.bscs19seecs@seecs.edu.pk)

**CMS ID: 291310,290647**

**BSCS 9B**

**Task #1:-**

Corrupt the input images with different types of noise models such as:-

1. 'Gaussian'

2. 'poisson'

3. 'salt & pepper'

4. 'speckle'

5. Salt only noise

6. Pepper only noise

**CODE**

| import cv2  import numpy as np  from skimage.util import random\_noise  # opening the image using cv2 library  image\_1 = cv2.imread("pears.png")  gray\_image = cv2.cvtColor(image\_1, cv2.COLOR\_BGR2GRAY)  def addNoise(gray\_img, mode):  noise = random\_noise(gray\_img, mode=mode)  noise = np.array(255 \* noise, dtype='uint8')  return noise  gaussian\_noise = addNoise(gray\_image, "gaussian")  cv2.imwrite("gaussian\_noise.png", gaussian\_noise)  poisson\_noise = addNoise(gray\_image, "poisson")  cv2.imwrite("poisson\_noise.png", poisson\_noise)  salt\_noise = addNoise(gray\_image, "salt")  cv2.imwrite("salt\_noise.png", salt\_noise)  pepper\_noise = addNoise(gray\_image, "pepper")  cv2.imwrite("pepper\_noise.png", pepper\_noise)  salt\_n\_pepper\_noise = addNoise(gray\_image, "s&p")  cv2.imwrite("salt\_n\_pepper\_noise.png", salt\_n\_pepper\_noise)  speckle\_noise = addNoise(gray\_image, "speckle")  cv2.imwrite("speckle\_noise.png", speckle\_noise) |
| --- |

**Image Corrupted with Gaussian Noise:**

****

**Image Corrupted with Poisson Noise:**

****

**Image Corrupted with Pepper Noise:**

****

**Image Corrupted with Salt Noise:**

****

**Image Corrupted with Salt and Pepper Noise:**

****

**Image Corrupted with Speckle Noise:**

****

**Task # 2:-**

Apply different kinds of noise removal filters as given below;

1. Arithmetic mean

2. Geometric mean

3. Harmonic mean

4. Contra harmonic mean, The contra harmonic mean filter is used for

filtering an image with either salt or pepper noise (but not both).

5. Max filters

6. Min filters

7. Median filters

**CODE:**

| import cv2  import numpy as np  import math  from scipy import ndimage  def remove\_noise(image, name\_suffix):  max\_filter = ndimage.filters.maximum\_filter(image, size=3)  name = "maximum" + name\_suffix + ".png"  cv2.imwrite(name, max\_filter)  min\_filter = ndimage.filters.minimum\_filter(image, size=3)  name = "minimum" + name\_suffix + ".png"  cv2.imwrite(name, min\_filter)  median\_filter = ndimage.filters.median\_filter(image, size=3)  name = "median" + name\_suffix + ".png"  cv2.imwrite(name, median\_filter)  arithmetic(image,name\_suffix)  geometric(image,9,name\_suffix)  Contra\_harmonic(image,name\_suffix,9,3)  harmonic(image,9,name\_suffix)  def geometric(im,filter,name\_suffix):  x=im.shape[0]  y=im.shape[1]  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))  arr\_size=filter\*filter  img\_output=np.zeros((x,y))  arr=np.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):  # print(img\_pad[i+f][j+s])  arr[m]=img\_pad[i+f][j+s]  s=s+1  m=m+1  s=-math.floor(filter/2)  f=f+1  value=int(np.prod(arr))  value=int(pow(value,1/arr\_size))  img\_output[i][j]=(value)  name = "geometric" + name\_suffix + ".png"  cv2.imwrite(name, img\_output)  cv2.waitKey(0)  def arithmetic(img,name\_suffix):  kernel = np.ones((7,7),np.float32)/49  dst = cv2.filter2D(img,-1,kernel)  name = "arithmetic" + name\_suffix + ".png"  cv2.imwrite(name, dst)  def Contra\_harmonic(im,name\_suffix,filter,q):  x=im.shape[0]  y=im.shape[1]  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))  arr\_size=filter\*filter  img\_output=np.zeros((x,y))    arr=np.zeros(arr\_size)  arr2=np.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]=float(pow(img\_pad[i+f][j+s],q+1))  arr2[m]=float(pow(img\_pad[i+f][j+s],q))  s=s+1  m=m+1  s=-math.floor(filter/2)  f=f+1    value=int(np.sum(arr)/np.sum(arr2))  img\_output[i][j]=(value)  name = "contra\_harmonic" + name\_suffix + ".png"  cv2.imwrite(name, img\_output)  t  def harmonic(im,filter,name\_suffix):  x=im.shape[0]  y=im.shape[1]  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))  arr\_size=filter\*filter  img\_output=np.zeros((x,y))    arr=np.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]=float(1/img\_pad[i+f][j+s])  s=s+1  m=m+1  s=-math.floor(filter/2)  f=f+1  value=float(np.sum(arr))  value=int(arr\_size/value)  img\_output[i][j]=(value)  name = "harmonic" + name\_suffix + ".png"  cv2.imwrite(name, img\_output)  cv2.waitKey(0)  # opening the image using cv2 library  image\_1 = cv2.imread("gaussian\_noise.png")  image\_1 = np.array(image\_1)  remove\_noise(image\_1, "\_gaussian\_removal")  image\_2 = cv2.imread("poisson\_noise.png")  image\_2 = np.array(image\_2)  remove\_noise(image\_2, "\_poisson\_removal")  image\_3 = cv2.imread("salt\_n\_pepper\_noise.png")  image\_3 = np.array(image\_3)  remove\_noise(image\_3, "\_salt\_n\_pepper\_removal")  image\_4 = cv2.imread("speckle\_noise.png")  image\_4 = np.array(image\_4)  remove\_noise(image\_4, "\_speckle\_removal")  image\_5 = cv2.imread("salt\_noise.png")  image\_5 = np.array(image\_5)  remove\_noise(image\_5, "\_salt\_removal")  image\_6 = cv2.imread("pepper\_noise.png")  image\_6 = np.array(image\_6)  remove\_noise(image\_6, "\_pepper\_removal") |
| --- |

**Arithmetic Filter**

| **Gaussian Noise** | **Poisson Noise** |
| --- | --- |
|  |  |
| **Salt Noise** | **Pepper Noise** |
|  |  |
| **Salt And Pepper** | **Speckle Noise** |
|  |  |

**Contra Harmonic**

| **Gaussian Noise** | **Poisson Noise** |
| --- | --- |
|  |  |
| **Salt Noise** | **Speckle Noise** |
|  |  |
| **Salt And Pepper** | **Pepper Noise** |
|  |  |

**Harmonic**

| **Gaussian Noise** | **Poisson Noise** |
| --- | --- |
|  |  |
| **Salt Noise** | **Speckle Noise** |
|  |  |
| **Salt And Pepper** | **Pepper Noise** |
|  |  |

**Geometric Filter:**

| **Gaussian Noise** | **Poisson Noise** |
| --- | --- |
|  |  |
| **Salt Noise** | **Speckle Noise** |
|  |  |
| **Salt And Pepper** | **Pepper Noise** |
|  |  |

**Minimum Filter**

| **Gaussian Noise** | **Poisson Noise** |
| --- | --- |
|  |  |
| **Salt Noise** | **Speckle Noise** |
|  |  |
| **Salt And Pepper** | **Pepper Noise** |
|  |  |

**Maximum Filter**

| **Gaussian Noise** | **Poisson Noise** |
| --- | --- |
|  |  |
| **Salt Noise** | **Speckle Noise** |
|  |  |
| **Salt And Pepper** | **Pepper Noise** |
|  |  |

**Median Filter**

| **Gaussian Noise** | **Poisson Noise** |
| --- | --- |
|  |  |
| **Salt Noise** | **Speckle Noise** |
|  |  |
| **Salt And Pepper** | **Pepper Noise** |
|  |  |