**EN2550 - Fundamentals of Image Processing & Machine Vision**

**Assignment 1**

**Intensity Transformations and Neighbourhood Filtering**

**Question-01**

**Code:**

c=np.array([(50,50),(50,100),(150,150)])

t1=np.linspace(0,c[0,1],c[0,0]+1).astype('uint8')

t2=np.linspace(c[1,1]+1,255,c[2,0]-c[1,0]).astype('uint8')

t3=np.linspace(c[2,1]+1,255,255-c[2,0]).astype('uint8')

transform= np.concatenate((t1,t2),axis=0).astype('uint8')

transform= np.concatenate((transform,t3),axis=0).astype('uint8')

fig,ax=plt.subplots()

ax.plot(transform)

ax.set\_xlabel(r'Input , $f ( \mathbf { x } ) $ ')

ax.set\_ylabel('Output , $\mathrm{T } [ f ( \mathbf { x } ) ] $')

ax.set\_xlim(0,255)

ax.set\_ylim(0,255)

ax.set\_aspect('equal')

plt.show()

img\_org =cv.imread('emma\_gray.jpg', cv.IMREAD\_GRAYSCALE)

image\_transformed = cv.LUT(img\_org, transform)

fig, ax =plt.subplots(1,2, sharex='all', sharey='all', figsize=(18,18))

ax[0].imshow(img\_org,cmap='gray')

ax[0].set\_title('original image')

ax[1].imshow(image\_transformed,cmap='gray')

ax[1].set\_title('transformed image')

**Output results:**

**A collage of a person

Description automatically generated with low confidence**

**Chart, line chart

Description automatically generated**

**Question 02**

**Code:**

c=np.array([(50,50),(50,100),(150,150)])

t1=np.linspace(0,c[0,1],c[0,0]+1).astype('uint8')

t2=np.linspace(c[1,1]+1,255,c[2,0]-c[1,0]).astype('uint8')

t3=np.linspace(c[2,1]+1,255,255-c[2,0]).astype('uint8')

transform= np.concatenate((t1,t2),axis=0).astype('uint8')

transform= np.concatenate((transform,t3),axis=0).astype('uint8')

c=np.array([(105,150),(205,100),(205,50)])

t\_1=np.linspace(255,c[0,1],c[0,0]+1).astype('uint8')

t\_2=np.linspace(255,c[1,1],c[1,0]-c[0,0]).astype('uint8')

t\_3=np.linspace(c[2,1],0,c[1,1]-c[2,1]).astype('uint8')

t\_w= np.concatenate((t\_1,t\_2),axis=0).astype('uint8')

t\_w= np.concatenate((t\_w,t\_3),axis=0).astype('uint8')

fig,ax=plt.subplots(1,2,figsize=(10,10))

ax[0].plot(transform)

ax[0].set\_xlabel(r'Input , $f ( \mathbf { x } ) $ ')

ax[0].set\_ylabel('Output , $\mathrm{T } [ f ( \mathbf { x } ) ] $')

ax[0].set\_xlim(0,255)

ax[0].set\_ylim(0,255)

ax[0].set\_aspect('equal')

…………

img\_org =cv.imread('brain\_proton\_density\_slice.png', cv.IMREAD\_GRAYSCALE)

image\_trans\_white = cv.LUT(img\_org, transform)

image\_trans\_gray = cv.LUT(img\_org, t\_w)

fig, ax  = plt.subplots(1,3, sharex='all', sharey='all', figsize=(18,18))

ax[0].imshow(img\_org,cmap='gray')

ax[0].set\_title('original image')

……………….

**Output results:**

**Chart, line chart

Description automatically generated**

**A picture containing text, old, set, ceramic ware

Description automatically generated**

**Question 03**

**Code:**

def gammaCorrection(src, gamma):

    invGamma = 1 / gamma

    table = [((i / 255) \*\* invGamma) \* 255 for i in range(256)]

    table = np.array(table, np.uint8)

    return cv.LUT(src, table)

img = cv.imread('highlights\_and\_shadows.jpg')

for i in range(10):

    gamma = i+rand.randint(0,9)\*0.1

    gammaImg = gammaCorrection(img,gamma)

    fig,ax=plt.subplots(1,2)

    ax[0].hist(gammaImg.flatten(),256,[0,256],color = 'r')

    ax[0].set\_xlim([0,256])

    ax[0].set\_title('Histogram of gamma corrected Image with $\gamma$ = {0}'.format(gamma))

    ax[1].imshow(cv.cvtColor(gammaImg,cv.COLOR\_BGR2RGB))

    ax[1].axis('off')

    plt.show()

**Output results:**

**Graphical user interface

Description automatically generated**

**Chart, histogram

Description automatically generated**

**Question 04**

**Code:**

def histogramEqualization(image):

    hist ,bins = np.histogram(image.ravel(), 256, [0, 256])

    cdf = hist.cumsum()

    cdf\_normalized = cdf\*(len(hist)-1)/cdf.max()

    transformation = cdf\_normalized.astype(int)

    equalized = cv.LUT(image, transformation)

    return equalized

shells = cv.imread(r'shells.png', cv.IMREAD\_GRAYSCALE)

equalized = histogramEqualization(shells)

fig, ax = plt.subplots(1,2, figsize=(15,7))

ax[0].set\_xlim([0, 256])

ax[0].title.set\_text('Histogram of the orignal image')

ax[0].hist(shells.flatten(), 256, [0, 256]);

…………………

fig, ax = plt.subplots(1,2, figsize=(15,7))

ax[0].axis('off')

ax[0].title.set\_text('orginal Image')

ax[0].imshow(shells, cmap = 'gray', vmin =0, vmax=255)

………………