Homework 2

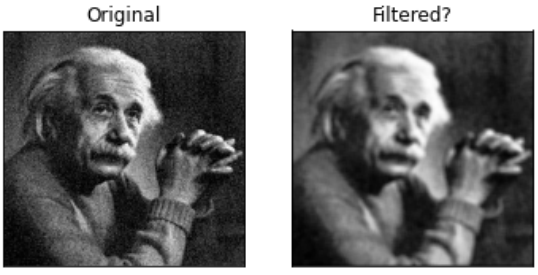
Joseph Cristiano

Problem 1)

```
def smoothingFilter(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #Must convert to grayscale in order to do operations
    kernel = np.ones((5,5),np.float32)/25 #creates 5x5 kernel
    kernelShape = kernel.shape #tuple of kernel dimensions
    imageShape = img.shape     #tuple of image dimensions

    #Zero padding below to add 2 units around the border of the image
    paddedDimensions = (imageShape[0]+kernelShape[0]-1,imageShape[1]+kernelShape[1]-1)
    paddedImage = np.zeros(paddedDimensions)
    for i in range(imageShape[0]):
        for j in range(imageShape[1]):
          paddedImage[i+int((kernelShape[0]-1)/2), j+int((kernelShape[1]-1)/2)] = img[i,j]

    #running the Filter
    for i in range(imageShape[0]):   #Create moving window
        for j in range(imageShape[1]):
            window = paddedImage[i:i+kernelShape[0],j:j+kernelShape[1]] #window matrix gathers values from image
            smoothValue = np.sum(window*kernel)
            img[i,j] = smoothValue #window gets multiplied against the kernel and the sum of the matrix replaces
    return img
```

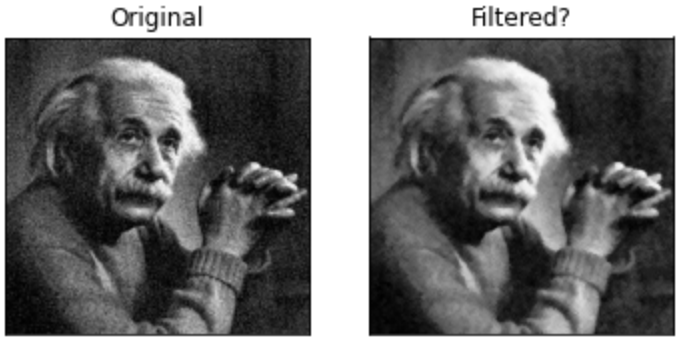
```
img1 = cv2.imread('imgnoise1.jpg')
smooth1 = smoothingFilter(img1)
plt.subplot(121),plt.imshow(img1,cmap='Greys_r'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(smooth1,cmap='Greys_r'),plt.title('Filtered?')
plt.xticks([]), plt.yticks([])
plt.show()
```


```
img2 = cv2.imread('imgnoise2.jpg')
smooth2 = smoothingFilter(img2)
plt.subplot(121),plt.imshow(img2,cmap='Greys_r'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(smooth2,cmap='Greys_r'),plt.title('Filtered?')
plt.xticks([]), plt.yticks([])
plt.show()
```



```
img1 = cv2.imread('imgnoise1.jpg')
smooth1 = centerWeightedFilter(img1)
plt.subplot(121),plt.imshow(img1,cmap='Greys_r'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(smooth1,cmap='Greys_r'),plt.title('Filtered?')
plt.xticks([]), plt.yticks([])
plt.show()
```


```
img2 = cv2.imread('imgnoise2.jpg')
smooth2 = centerWeightedFilter(img2)
plt.subplot(121),plt.imshow(img2,cmap='Greys_r'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(smooth2,cmap='Greys_r'),plt.title('Filtered?')
plt.xticks([]), plt.yticks([])
plt.show()
```



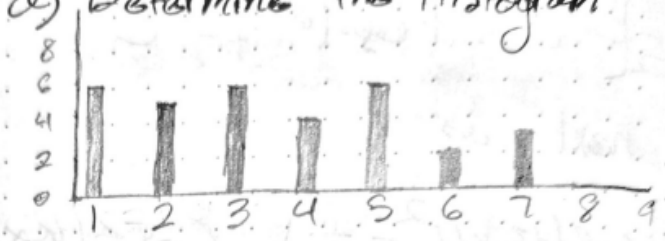
Problem 2,4 &5

Elec 401 Homework 2 3/2/22

Problem 2

$$I = \begin{bmatrix} 1 & 3 & 4 & 2 & 2 & 5 \\ 1 & 1 & 7 & 5 & 1 & 2 \\ 6 & 5 & 0 & 5 & 2 & 4 \\ 3 & 5 & 1 & 0 & 4 & 3 \\ 1 & 3 & 2 & 3 & 2 & 3 \\ 0 & 6 & 7 & 7 & 4 & 5 \end{bmatrix}$$

a) Determine the Histogram



b) Determine the Cumulative distribution

$P_r(r_k) = n_k / MN$

$MN = 36$

use table on slide 14

28	18	9	24	24	7
28	28	2	7	28	24
3	7	22	7	24	9
18	7	24	22	9	18
28	18	24	18	24	18
22	3	2	2	9	7

c) Find point transform T[I]

$S_0 = T(r_0) = 7\,pr(r_0) = 7\left(\frac{190}{36}\right) = 153$

$S_1 = 7\left(\frac{1023}{36}\right) = 198$

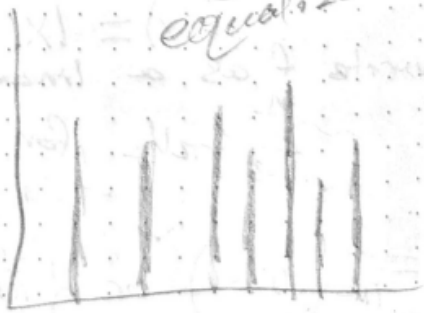
$S_2 = 165$ $S_4 = 64$ $S_6 = 24$

$S_3 = 128$ $S_5 = 48$ $S_7 = 16$

d) transformed image

198	128	64	165	165	48
198	198	16	48	128	165
24	48	153	48	165	64
128	48	128	153	64	128
198	128	165	128	165	128
153	24	16	16	64	48

e) Histogram equalized



Eleg 4041 Homework 2 2/2/22

Problem 4

Prove $g\left(\frac{t}{T}\right) \xrightarrow{\mathcal{F}} TG(uT)$

Definition

$$F\{g(ct)\} = \int_{-\infty}^{\infty} g(ct) e^{-i2\pi ft} dt \qquad u = ct \quad du = c\,dt$$

$$F\{g(ct)\} = \int_{-c\infty}^{c\infty} \frac{g(u)}{c} e^{-i2\pi f \frac{u}{c}} du$$

if $c \geq 0$

$$F\{g(ct)\} = \boxed{\frac{G\left(\frac{f}{c}\right)}{|c|}}$$

if $c < 0$

$$F(g(ct)) = \int_{\infty}^{-\infty} \frac{g(u)}{c} e^{-2\pi f \frac{u}{c}} du$$

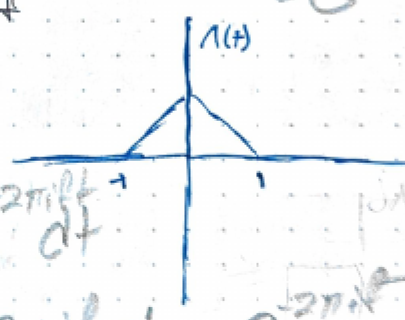
$$= -\int_{-\infty}^{\infty} \frac{g(u)}{c} e^{-2\pi f \frac{u}{c}} du$$

$$= \frac{G\left(\frac{f}{c}\right)}{-c} = \boxed{\frac{G\left(\frac{f}{c}\right)}{|c|}}$$

Problem 5

Find fourier transform of

$$\Lambda(t) = \begin{cases} 1-|t|, & \text{if } |t| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$



$$= \int_{-1}^{0} (1+t) e^{-2\pi i f t} dt + \int_{0}^{1} (1-t) e^{-2\pi i f t} dt$$

$$= \left[\frac{1+2\pi i f}{4\pi^2 f^2} - \frac{e^{2\pi i f}}{4\pi^2 f^2}\right] - \left[\frac{2\pi i f - 1}{4\pi^2 f^2} + \frac{e^{-2\pi i f}}{4\pi^2 f^2}\right]$$

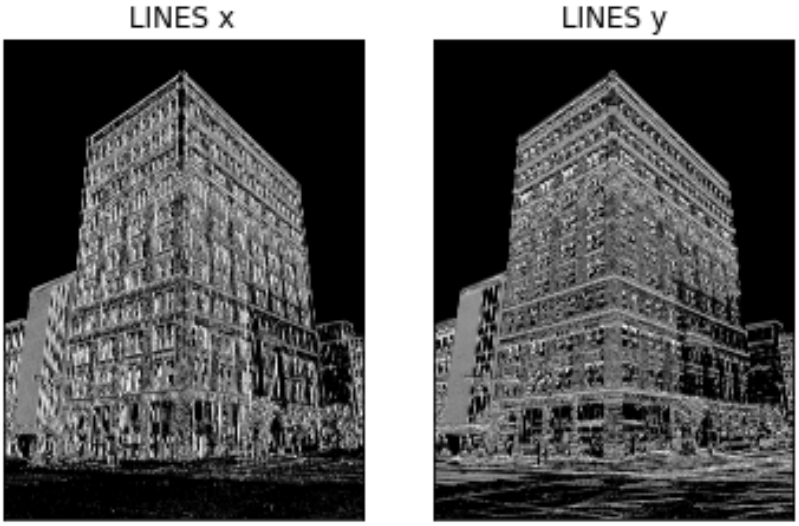
$$= \frac{e^{-2\pi i f}(e^{2\pi i f} - 1)^2}{4\pi^2 f^2} = \frac{e^{-2\pi i f}(e^{\pi i f}\{e^{\pi i f} - e^{-\pi i f}\})^2}{4\pi^2 f^2}$$

$$= \frac{e^{-2\pi i f} e^{2\pi i f} (2i)^2 \sin^2(\pi f)}{4\pi^2 f^2} = \left(\frac{\sin(\pi f)}{\pi f}\right)^2 = \boxed{\operatorname{sinc}^2 f}$$

Question 3

```python
def edgeDetectSobel(img):
    #better version of Sobel Filter from Chapter 2 slide 64
    kernel = np.array([[-3,0,3],
                       [-10,0,10],
                       [-3,0,3]])/32

    edgeX = np.zeros_like(img) #create arrays for edge data
    edgeY = np.zeros_like(img)
    kernelShape = kernel.shape #tuple of kernel dimensions
    imageShape = img.shape     #tuple of image dimensions

    #Zero padding below to add 2 units around the border of the image
    paddedDimensions = (imageShape[0]+kernelShape[0]-1,imageShape[1]+kernelShape[1]-1)
    paddedImage = np.zeros(paddedDimensions)
    for i in range(imageShape[0]):
        for j in range(imageShape[1]):
          paddedImage[i+int((kernelShape[0]-1)/2), j+int((kernelShape[1]-1)/2)] = img[i,j]

    #running the Filter
    for i in range(imageShape[0]):   #Create moving window
        for j in range(imageShape[1]):
            window = paddedImage[i:i+kernelShape[0],j:j+kernelShape[1]] #window matrix gathers values from image
            edgeX[i,j] = np.sum(window*kernel)#window gets multiplied against the kernel
            edgeY[i,j] = np.sum(window*np.flip(kernel.T,axis=0))

    gradient = np.sqrt(np.square(edgeX) + np.square(edgeY))
    gradient *= 255.0 / gradient.max() #using Gradient equation from Chapter 2 Slide 60

    return edgeX, edgeY, gradient
```

```
test1, test2, test3 = edgeDetectSobel(img_blur)
plt.subplot(121),plt.imshow(test1,cmap='Greys_r'),plt.title('LINES x')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(test2,cmap='Greys_r'),plt.title('LINES y')
plt.xticks([]), plt.yticks([])
plt.show()
```




```
plt.subplot(121),plt.imshow(img_blur,cmap='Greys_r'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(test3,cmap='Greys_r'),plt.title('gradient')
plt.xticks([]), plt.yticks([])
plt.show()
```