Mujtaba Shahid Faizi

BSCS-5A

#131818

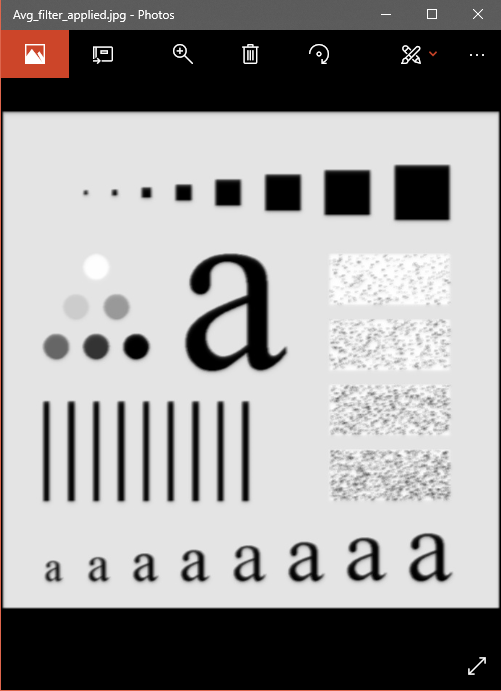
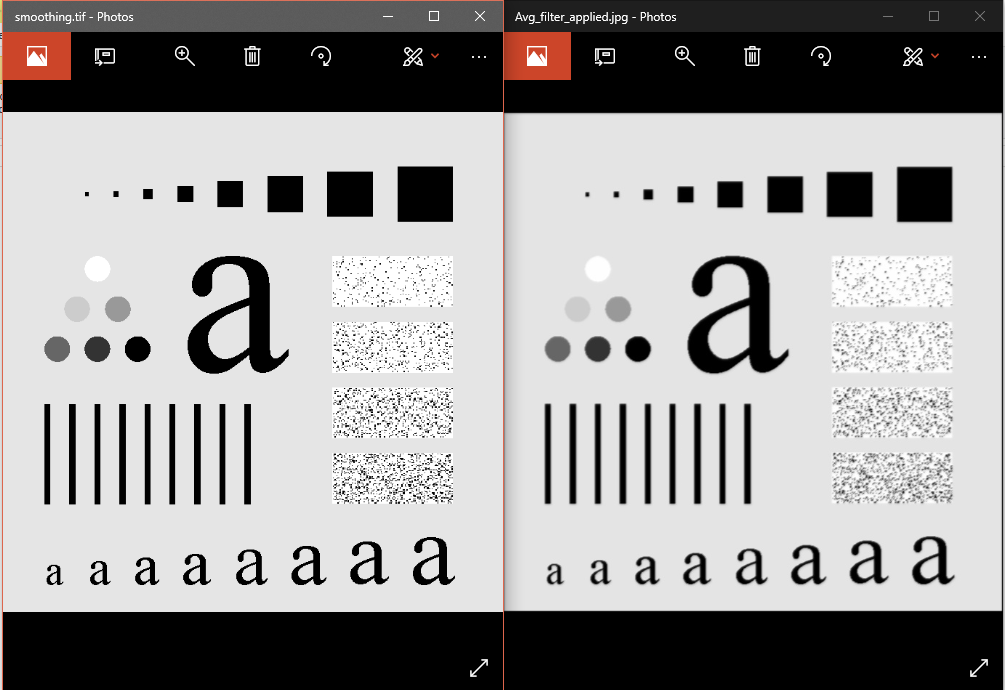
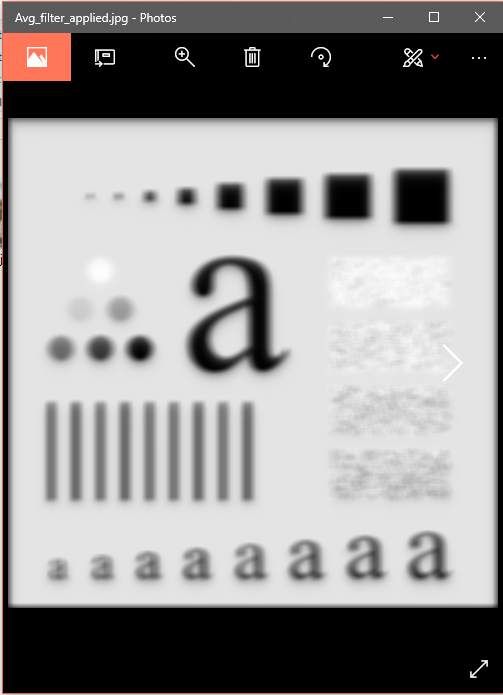
Lab 8 of DIP

**Task 1:**

**Code:**

**import** numpy **as** np  
**import** cv2  
  
**def** apply\_filter(img,filter):  
 filter\_sum=0  
 **for** x **in** range(0, filter.shape[0]):  
 **for** y **in** range(0, filter.shape[1]):  
 filter\_sum = filter\_sum + filter[x, y]  
 arr=np.asarray(img)  
  
 filter\_row=filter.shape[0]  
 filter\_col=filter.shape[1]  
 padding\_size=filter\_col/2 *#do zero padding acc. to the filer size to avoid out of bound errors* arr=np.pad(arr, int(padding\_size), mode=**'constant'**) *#applying zero padding* rows = arr.shape[0] *# width & height of the image i.e. no. of pixels* cols = arr.shape[1]  
 c = 0  
 d=0  
 g=filter.shape[1]  
 **for** a **in** range(0,(rows\*cols)):  
 **if** filter\_col>cols: *#for last column pixel, move the window to the next row* d+=1  
 filter\_row+=1  
 c=0  
 filter\_col=g  
 **if** filter\_row>rows: *#after last row limit of window, exit* **break** b=arr[int(d):int(filter\_row), int(c):int(filter\_col)]  
 sum=np.sum(filter\*b)  
 avg=sum/filter\_sum  
 arr[int(d+padding\_size),int(c+padding\_size)]=avg *#replacing with the avg pixel value* c+=1 *#for moving the window ahead one pixel at a time (same for rows also)* filter\_col+=1  
 img = np.array(arr \* 255, dtype=np.uint8) *#converting array to image* **return** arr  
  
img = cv2.imread(**'smoothing.tif'**,0)  
a = np.array([[1,1,1], [1,1,1], [1,1,1]])  
img=apply\_filter(img,a)  
cv2.imwrite(**'Avg\_filter\_applied.jpg'**, img)

**Screenshot (original, 3x3 filter, 5x5 filter, 15x15 filter i.e. all filters have every constant as 1):**

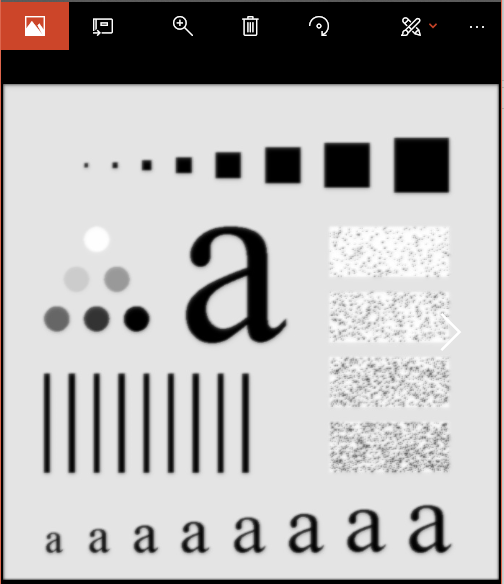
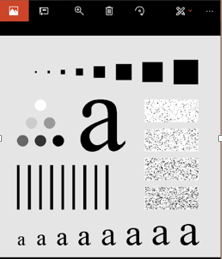
Increasing the size of the filter haves a more/strong blurring/smoothing effect since the filter covers more area for averaging the image pixels.

**Task 2:** 

**Code (same as task 1, just the filter specification is different ):**

**import** numpy **as** np  
**import** cv2  
  
**def** apply\_filter(img,filter):  
 filter\_sum=0  
 **for** x **in** range(0, filter.shape[0]):  
 **for** y **in** range(0, filter.shape[1]):  
 filter\_sum = filter\_sum + filter[x, y]  
 arr=np.asarray(img)  
  
 filter\_row=filter.shape[0]  
 filter\_col=filter.shape[1]  
 padding\_size=filter\_col/2 *#do zero padding acc. to the filer size to avoid out of bound errors* arr=np.pad(arr, int(padding\_size), mode=**'constant'**) *#applying zero padding* rows = arr.shape[0] *# width & height of the image i.e. no. of pixels* cols = arr.shape[1]  
 c = 0  
 d=0  
 g=filter.shape[1]  
 **for** a **in** range(0,(rows\*cols)):  
 **if** filter\_col>cols: *#for last column pixel, move the window to the next row* d+=1  
 filter\_row+=1  
 c=0  
 filter\_col=g  
 **if** filter\_row>rows: *#after last row limit of window, exit* **break** b=arr[int(d):int(filter\_row), int(c):int(filter\_col)]  
 sum=np.sum(filter\*b)  
 avg=sum/filter\_sum  
 arr[int(d+padding\_size),int(c+padding\_size)]=avg *#replacing with the avg pixel value* c+=1 *#for moving the window ahead one pixel at a time (same for rows also)* filter\_col+=1  
 img = np.array(arr \* 255, dtype=np.uint8) *#converting array to image* **return** arr  
  
img = cv2.imread(**'smoothing.tif'**,0)  
a = np.array([[1,1,2,2,2,1,1], [1,2,2,4,2,2,1], [2,2,4,8,4,2,2], [2,4,8,16,8,4,2],[2,2,4,8,4,2,2],[1,2,2,4,2,2,1],[1,1,2,2,2,1,1]])  
img=apply\_filter(img,a)  
cv2.imwrite(**'gausian\_filter\_applied.jpg'**, img)

**Screenshot:**

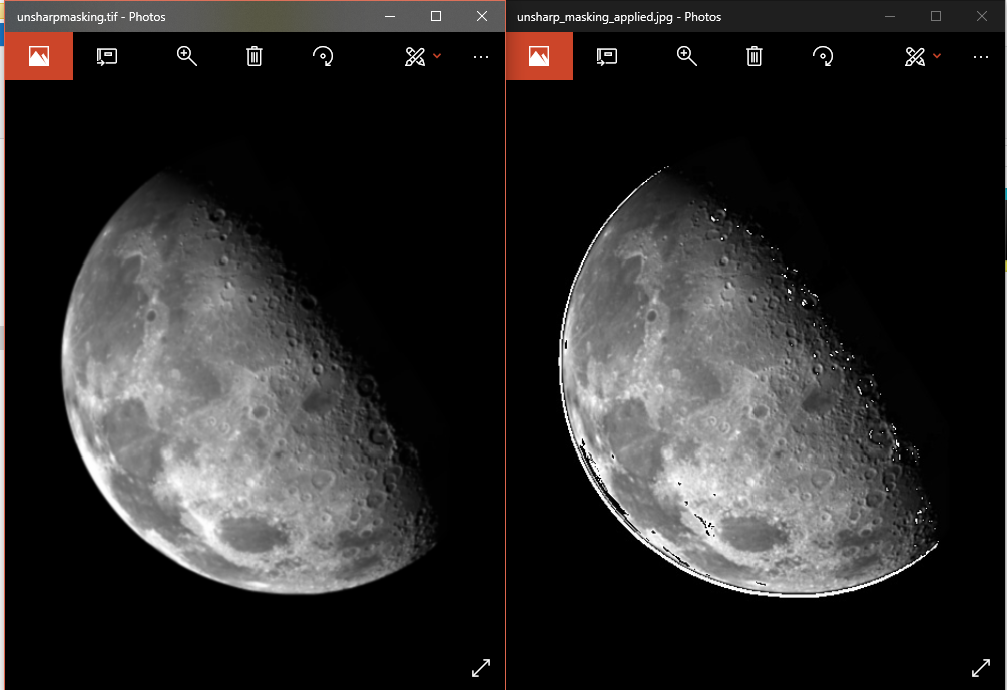


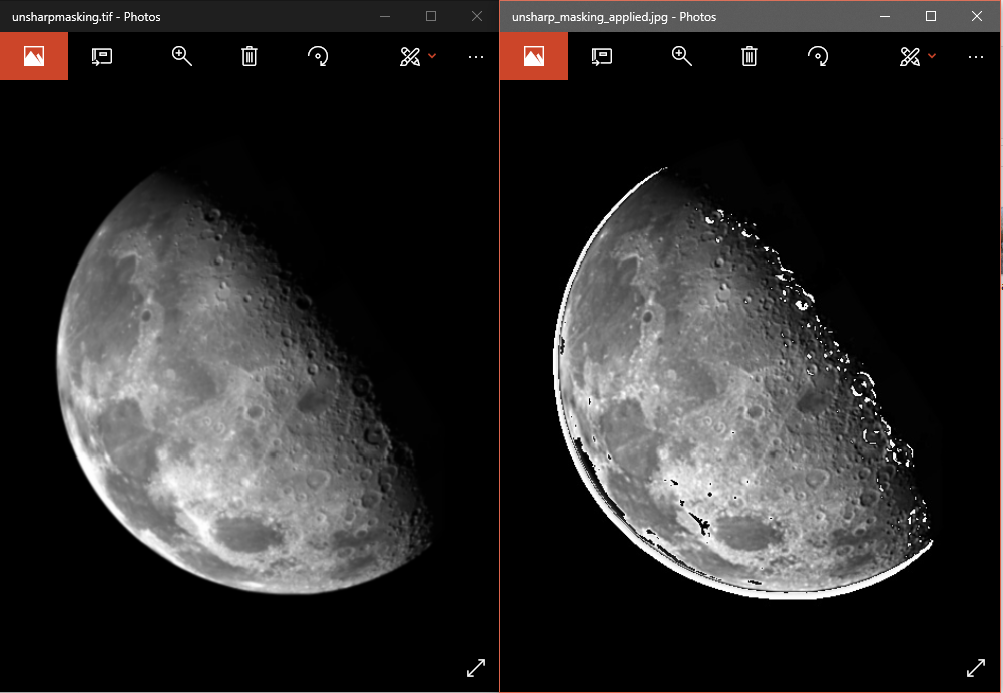
When the value of sigma increases, the blurring/smoothing effect increases.

**Task 3:**

**Code:**

**import** numpy **as** np  
**import** cv2  
  
**def** apply\_filter(img,filter):  
 filter\_sum=0  
 **for** x **in** range(0, filter.shape[0]):  
 **for** y **in** range(0, filter.shape[1]):  
 filter\_sum = filter\_sum + filter[x, y]  
 arr=np.asarray(img)  
  
 filter\_row=filter.shape[0]  
 filter\_col=filter.shape[1]  
 padding\_size=filter\_col/2 *#do zero padding acc. to the filer size to avoid out of bound errors* arr=np.pad(arr, int(padding\_size), mode=**'constant'**) *#applying zero padding* rows = arr.shape[0] *# width & height of the image i.e. no. of pixels* cols = arr.shape[1]  
 c = 0  
 d=0  
 g=filter.shape[1]  
 **for** a **in** range(0,(rows\*cols)):  
 **if** filter\_col>cols: *#for last column pixel, move the window to the next row* d+=1  
 filter\_row+=1  
 c=0  
 filter\_col=g  
 **if** filter\_row>rows: *#after last row limit of window, exit* **break** b=arr[int(d):int(filter\_row), int(c):int(filter\_col)]  
 sum=np.sum(filter\*b)  
 avg=sum/filter\_sum  
 arr[int(d+padding\_size),int(c+padding\_size)]=avg *#replacing with the avg pixel value* c+=1 *#for moving the window ahead one pixel at a time (same for rows also)* filter\_col+=1  
 **return** arr  
  
a = np.array([[1,1,1], [1,1,1], [1,1,1]])  
img = cv2.imread(**'unsharpmasking.tif'**,0)  
arr1=np.asarray(img)  
filter\_col = a.shape[1]  
padding\_size = filter\_col / 2 *# do zero padding acc. to the filer size to keep the sizes equal with the filtered image*arr1 = np.pad(arr1, int(padding\_size), mode=**'constant'**) *# applying zero padding*arr2=apply\_filter(img,a)  
  
  
cv2.imwrite(**'unsharp\_masking\_applied.jpg'**, np.add(np.absolute(np.subtract(arr1,arr2)),arr1)) *#highpass = (original - lowpass) + original* **Screenshot (3x3 average filter, 7x7 gaussian filter):**





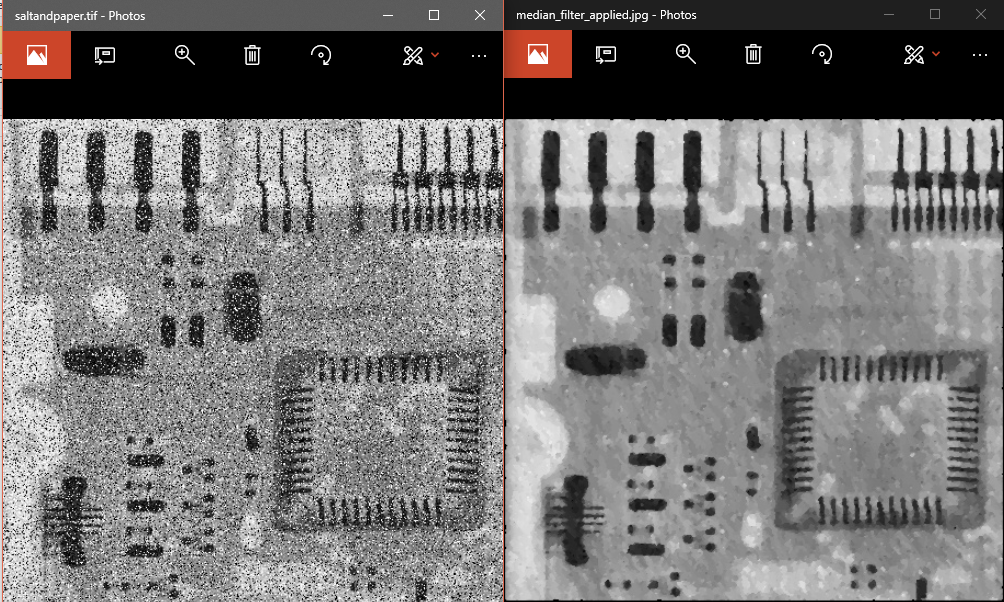
The edges are more visible and sharpened but there some white noise needed to be removed

**Task 4:**

**Code:**

**import** numpy **as** np  
**import** cv2  
  
**def** apply\_filter(img,filter):  
 filter\_sum=0  
 **for** x **in** range(0, filter.shape[0]):  
 **for** y **in** range(0, filter.shape[1]):  
 filter\_sum = filter\_sum + filter[x, y]  
 arr=np.asarray(img)  
  
 filter\_row=filter.shape[0]  
 filter\_col=filter.shape[1]  
 padding\_size=filter\_col/2 *#do zero padding acc. to the filer size to avoid out of bound errors* arr=np.pad(arr, int(padding\_size), mode=**'constant'**) *#applying zero padding* rows = arr.shape[0] *# width & height of the image i.e. no. of pixels* cols = arr.shape[1]  
 c = 0  
 d=0  
 g=filter.shape[1]  
 **for** a **in** range(0,(rows\*cols)):  
 **if** filter\_col>cols: *#for last column pixel, move the window to the next row* d+=1  
 filter\_row+=1  
 c=0  
 filter\_col=g  
 **if** filter\_row>rows: *#after last row limit of window, exit* **break** b=arr[int(d):int(filter\_row), int(c):int(filter\_col)]  
 arr[int(d+padding\_size),int(c+padding\_size)]=np.median(b) *#replacing with the median pixel value* c+=1 *#for moving the window ahead one pixel at a time (same for rows also)* filter\_col+=1  
 img = np.array(arr \* 255, dtype=np.uint8) *#converting array to image* **return** arr  
  
img = cv2.imread(**'saltandpaper.tif'**,0)  
a = np.array([[1,1,1], [1,1,1], [1,1,1]])  
img=apply\_filter(img,a)  
cv2.imwrite(**'median\_filter\_applied.jpg'**, img)

**Screenshot (3x3 median filter):**



Using median filter removes the salt & pepper noise in the given image.

**Task 5:**

**Code:**

**import** cv2  
**import** numpy **as** np  
  
img = cv2.imread(**'two\_cats.jpg'**,0)  
sobelx = cv2.Sobel(img,cv2.CV\_64F,1,0,ksize=5) *#vertical edges by calculating gradient along x axis*sobely = cv2.Sobel(img,cv2.CV\_64F,0,1,ksize=5) *#horizontal edges by calculating gradient along y axis*arr1 = np.asarray(sobelx)  
arr2 = np.asarray(sobely)  
cv2.imwrite(**'Sharpening filter.jpg'**, np.add(arr1,arr2))

**Screenshot:**

