Mujtaba Shahid Faizi

BSCS-5A

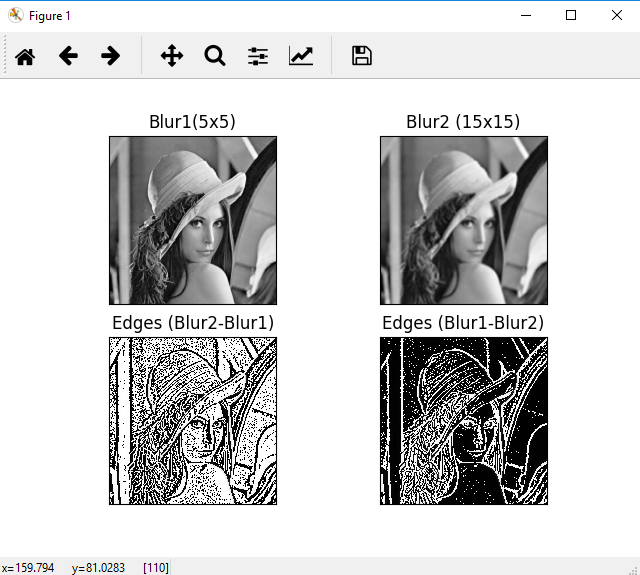
#131818

Lab 3 of Computer Vision

Task1:  
**CODE:**

**import** cv2  
**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** plt  
  
img = cv2.imread(**'lena.png'**, 0)  
blur = cv2.GaussianBlur(img,(5,5),0)  
blur2 = cv2.GaussianBlur(img,(15,15),0)  
  
edge1 = blur - blur2  
edge2 = blur2 - blur  
  
plt.subplot(2,2,1),plt.imshow(blur, cmap=**'gray'**),plt.title(**'Blur1(5x5)'**)  
plt.xticks([]), plt.yticks([])  
plt.subplot(2,2,2),plt.imshow(blur2, cmap=**'gray'**),plt.title(**'Blur2 (15x15)'**)  
plt.xticks([]), plt.yticks([])  
plt.subplot(2,2,3),plt.imshow(edge2, cmap=**'gray'**),plt.title(**'Edges (Blur2-Blur1)'**)  
plt.xticks([]), plt.yticks([])  
plt.subplot(2,2,4),plt.imshow(edge1, cmap=**'gray'**),plt.title(**'Edges (Blur1-Blur2)'**)  
plt.xticks([]), plt.yticks([])  
  
plt.show()

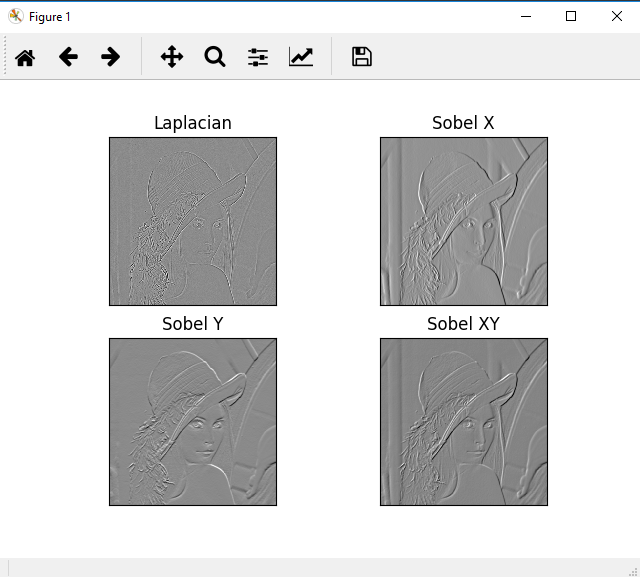
**SCREENSHOT:**



Task2:  
**CODE:**

**import** cv2  
**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** plt  
  
img = cv2.imread(**'lena.png'**, 0)  
img = cv2.GaussianBlur(img,(3,3),0)  
  
*# convolute with proper kernels*laplacian = cv2.Laplacian(img,cv2.CV\_64F)  
sobelx = cv2.Sobel(img,cv2.CV\_64F,1,0,ksize=5) *# x*sobely = cv2.Sobel(img,cv2.CV\_64F,0,1,ksize=5) *# y*sobel=sobelx+sobely *# x and y combined*plt.subplot(2,2,1),plt.imshow(laplacian,cmap = **'gray'**)  
plt.title(**'Laplacian'**), plt.xticks([]), plt.yticks([])  
plt.subplot(2,2,2),plt.imshow(sobelx,cmap = **'gray'**)  
plt.title(**'Sobel X'**), plt.xticks([]), plt.yticks([])  
plt.subplot(2,2,3),plt.imshow(sobely,cmap = **'gray'**)  
plt.title(**'Sobel Y'**), plt.xticks([]), plt.yticks([])  
plt.subplot(2,2,4),plt.imshow(sobel,cmap = **'gray'**)  
plt.title(**'Sobel XY'**), plt.xticks([]), plt.yticks([])  
  
plt.show()

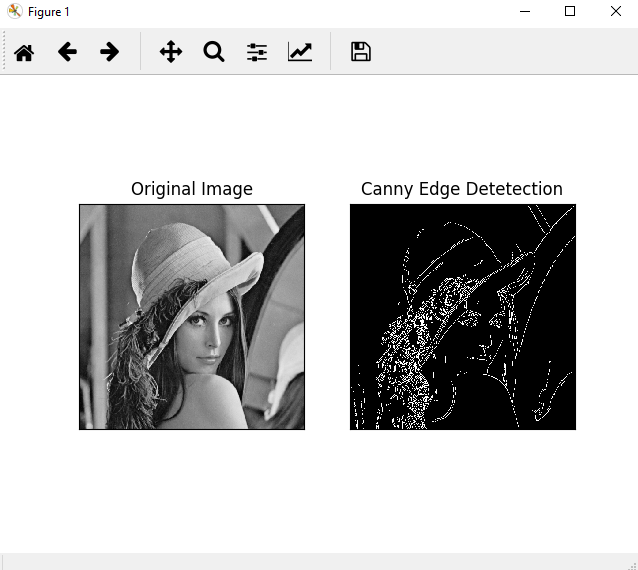
**SCREENSHOT:**



Task3:  
**CODE:**

**import** cv2  
**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** plt  
  
img = cv2.imread(**'lena.png'**,0)  
edges = cv2.Canny(img,100,200)  
  
plt.subplot(121),plt.imshow(img,cmap = **'gray'**)  
plt.title(**'Original Image'**), plt.xticks([]), plt.yticks([])  
plt.subplot(122),plt.imshow(edges,cmap = **'gray'**)  
plt.title(**'Canny Edge Detetection'**), plt.xticks([]), plt.yticks([])  
  
plt.show()

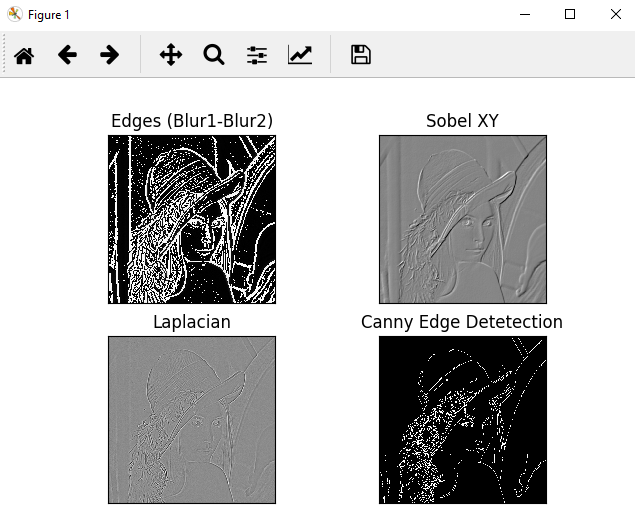
**SCREENSHOT:**



Task4:  
**CODE:**

**import** cv2  
**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** plt  
  
img = cv2.imread(**'lena.png'**, 0)  
  
gaus = cv2.GaussianBlur(img,(3,3),0)  
blur = cv2.GaussianBlur(img,(5,5),0)  
blur2 = cv2.GaussianBlur(img,(15,15),0)  
edge1 = blur - blur2  
edge2 = blur2 - blur  
  
laplacian = cv2.Laplacian(gaus,cv2.CV\_64F) *#laplacian of the gaussian of the image*sobelx = cv2.Sobel(gaus,cv2.CV\_64F,1,0,ksize=5) *# x*sobely = cv2.Sobel(gaus,cv2.CV\_64F,0,1,ksize=5) *# y*sobel=sobelx+sobely *# x and y combined*edges = cv2.Canny(img,100,200) *#canny edge detetection*plt.subplot(2,2,1),plt.imshow(edge1, cmap=**'gray'**),plt.title(**'Edges (Blur1-Blur2)'**)  
plt.xticks([]), plt.yticks([])  
plt.subplot(2,2,2),plt.imshow(sobel,cmap = **'gray'**)  
plt.title(**'Sobel XY'**), plt.xticks([]), plt.yticks([])  
plt.subplot(2,2,3),plt.imshow(laplacian,cmap = **'gray'**)  
plt.title(**'Laplacian'**), plt.xticks([]), plt.yticks([])  
plt.subplot(2,2,4),plt.imshow(edges,cmap = **'gray'**)  
plt.title(**'Canny Edge Detetection'**), plt.xticks([]), plt.yticks([])  
  
plt.show()

**SCREENSHOT:**



Task5:  
**CODE:**

**import** cv2  
**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** plt  
  
img1 = cv2.imread(**'1.png'**,0)  
img2 = cv2.imread(**'2.png'**,0)  
  
edges1 = cv2.Canny(img1,10,20)  
edges2 = cv2.Canny(img2,10,20)  
  
code = cv2.bitwise\_and(edges1, edges2) *#common edges between both images*plt.subplot(2,2,1),plt.imshow(edges1, cmap = **'gray'**)  
plt.title(**'Original Image'**), plt.xticks([]), plt.yticks([])  
plt.subplot(2,2,2),plt.imshow(edges2, cmap = **'gray'**)  
plt.title(**'Edge Image'**), plt.xticks([]), plt.yticks([])  
plt.subplot(2,2,3),plt.imshow(code, cmap = **'gray'**)  
plt.title(**'Secret Code'**), plt.xticks([]), plt.yticks([])  
  
plt.show()

**SCREENSHOT:**

