Name : Anupriya Goyal
UB Person No.: 50287108
Date of Submission: 10/08/2018

Computer Vision and Image Processing (CSE 573)
Project Report

Task1: Edge Detection

Code for Sobel edge detection

```
import cv2
import math

#list for sobel operator
g_sob_x = [[-1,0,1],[-2,0,2],[-1,0,1]]
g_sob_y = [[1,2,1],[0,0,0],[-1,-2,-1]]
original_img = cv2.imread("path/task1.png", 0)

x_matrix_sobel =original_img.copy()
#No. of rows in original image
rows= original_img.shape[0]
print(original_img.shape)
#No. of columns in original image
columns = original_img.shape[1]
original_img[0,:] = original_img[:,0] = original_img[:,-1] = original_img[-1,:] = 0
xy_matrix_sobel=original_img.copy()

#creating a copy of image to store edge detected image values
y_matrix_sobel =original_img.copy()
res1 =0
res2 =0

for i in range(1, 598):
   for j in range (1,898):

      res1 = (  original_img[i-1][j-1]* g_sob_x[0][2]) + \
            (  original_img[i-1][j]* g_sob_x[0][1]) + \
            ( original_img[i-1][j+1]* g_sob_x[0][0] ) + \
            (original_img[i][j-1]* g_sob_x[1][2] ) + \
            (original_img[i][j]* g_sob_x[1][1] ) + \
            ( original_img[i][j+1]* g_sob_x[1][0] ) + \
            ( original_img[i+1][j-1]* g_sob_x[2][2]) + \
```

```
            (original_img[i+1][j]*g_sob_x[2][1]) + \
            ( original_img[i+1][j+1]* g_sob_x[2][0])

        res2 =   (original_img[i-1][j-1]*g_sob_y[0][2]) + \
                  ( original_img[i-1][j]*g_sob_y[0][1]) + \
                 (original_img[i-1][j+1]*g_sob_y [0][0]) + \
                 ( original_img[i][j-1]*g_sob_y [1][2] ) + \
                ( original_img[i][j]*g_sob_y [1][1]) + \
               ( original_img[i][j+1]*g_sob_y [1][0] ) + \
              ( original_img[i+1][j-1]*g_sob_y [2][2] ) +\
              (original_img[i+1][j]*g_sob_y [2][1] ) + \
            ( original_img[i+1][j+1]*g_sob_y [2][0] )
#vertical edges
        x_matrix_sobel[i-1][j-1] = res1
#horizontal edges
        y_matrix_sobel[i-1][j-1] = res2
#combining vertical and horizontal edges
        edge_magnitude =math.sqrt(res1 ** 2 + res2 ** 2)
        xy_matrix_sobel[i-1][j-1]=edge_magnitude

        #print(np.asarray(edge_x))
#print(sobelxImage)

#Eliminating zero values in vertical edges,horizontal edges and combined edges
max_of_y=0
min_of_y=0
for i in range(1, rows):
    for j in range (1,columns):
        if(min_of_y> y_matrix_sobel[i-1][j-1]):
            min_of_y= y_matrix_sobel[i-1][j-1]
        if(max_of_y< y_matrix_sobel[i-1][j-1]):
            max_of_y= y_matrix_sobel[i-1][j-1]

max_of_x=0
min_of_x=0
for i in range(1, rows):
    for j in range (1,columns):
        if(min_of_x> x_matrix_sobel[i-1][j-1]):
            min_of_x= x_matrix_sobel[i-1][j-1]
        if(max_of_x< x_matrix_sobel[i-1][j-1]):
            max_of_x= x_matrix_sobel[i-1][j-1]
max_of_xy=0
min_of_xy=0
```

```
for i in range(1, rows):
    for j in range (1,columns):
        if(min_of_xy> xy_matrix_sobel[i-1][j-1]):
            min_of_xy= xy_matrix_sobel[i-1][j-1]
        if(max_of_xy< xy_matrix_sobel[i-1][j-1]):
            max_of_xy= xy_matrix_sobel[i-1][j-1]

x_dir_edge = (x_matrix_sobel -min_of_x) / math.fabs( max_of_x-min_of_x)
cv2.imshow('Edges of x direction',x_dir_edge)
print(x_dir_edge.shape)
y_dir_edge = (y_matrix_sobel -min_of_y) / math.fabs( max_of_y-min_of_y)
cv2.imshow('Edges of y direction',y_dir_edge)
print(y_dir_edge.shape)
xy_comb_edge = (xy_matrix_sobel -min_of_xy) / math.fabs( max_of_xy-min_of_xy)
cv2.imshow('Combined edges in x and y direction',xy_comb_edge)
print(xy_comb_edge.shape)
cv2.imwrite('x_dir_edge.png',x_dir_edge)
cv2.imwrite('y_dir_edge.png',y_dir_edge)
cv2.imwrite('xy_comb_edge.png',xy_comb_edge)


cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output is given in following figures for edge detection :

Figure 1: Edges in x direction

Figure2 : Edges in y direction

Figure 3: Edges in combined x and y direction

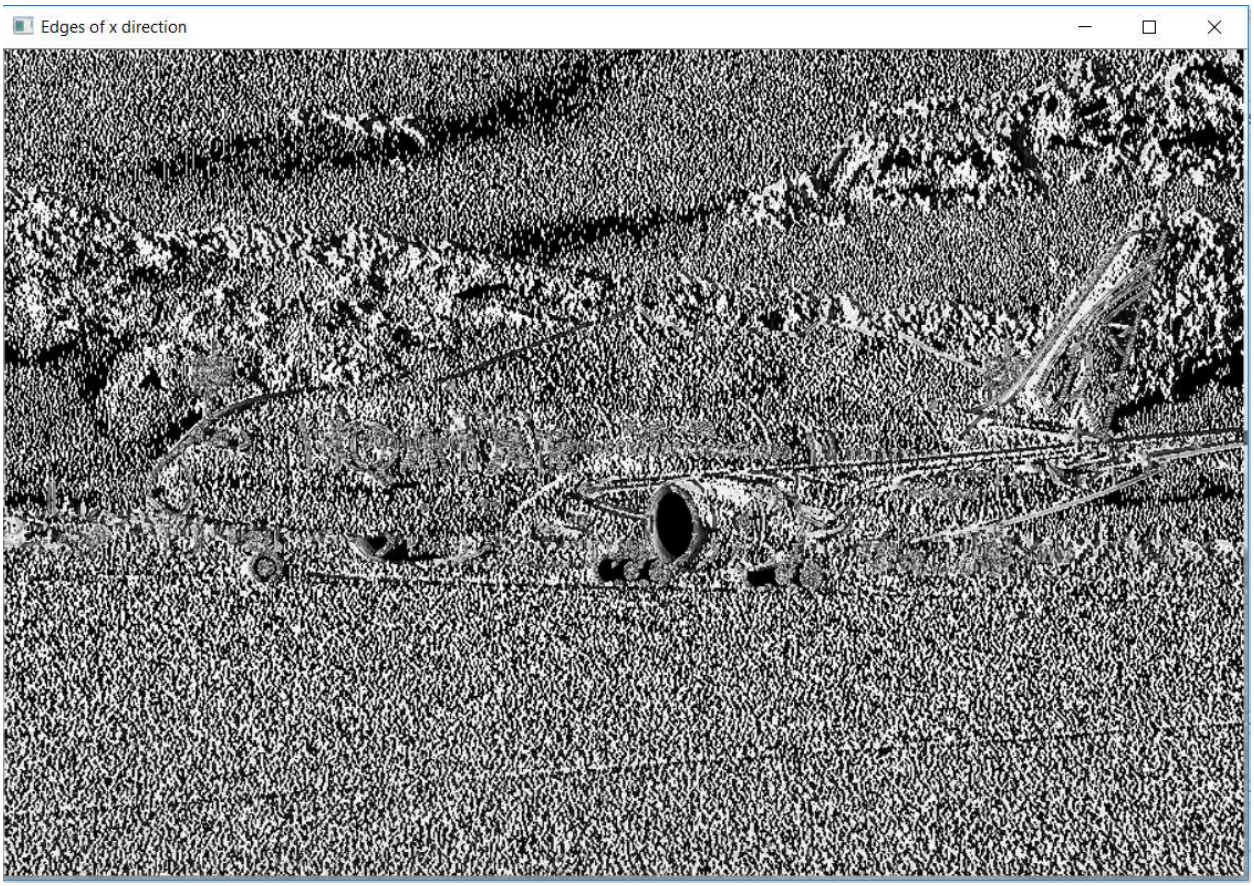


Figure 1. Edges in x direction

Figure2. Edges in y direction

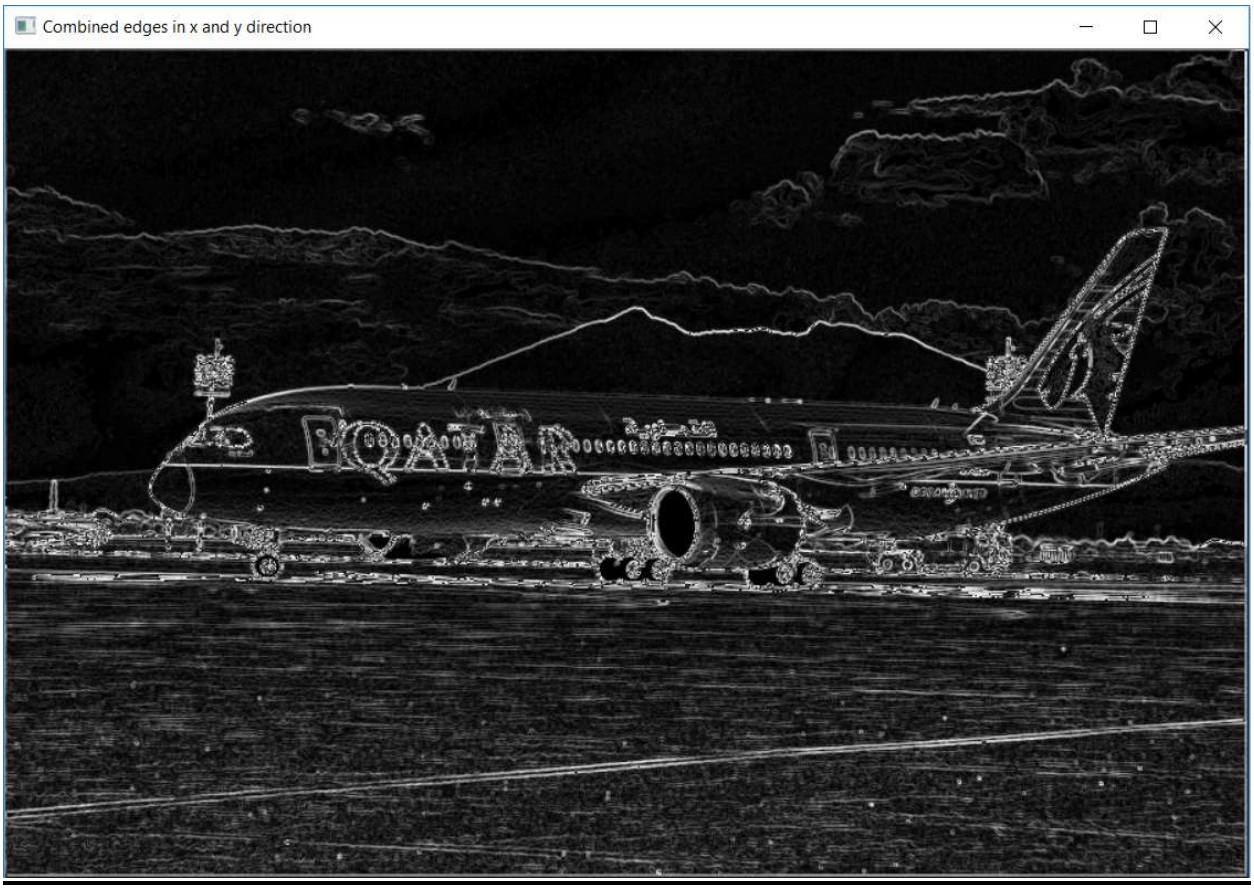


Figure3. Edges in combined x and y direction

Reference:

1. https://stackoverflow.com/questions/17815687/image-processing-implementing-sobel-filter

Task2: Keypoint Detection

Code snippets for keypoint detection program:

For first octave and key point detection code snippet

```
import math
import cv2
import numpy as np
```

```
org_img = cv2.imread("path\task2.jpg",0)
copy_org_img=org_img.copy()
val = 7;
avg = val/2;
rows=org_img.shape[0]
column=org_img.shape[1]
copy_org_img[0,:] = copy_org_img[:,0] = copy_org_img[:,-3] =  copy_org_img[-3,:] = 0
d11=copy_org_img.copy()
d12=copy_org_img.copy()
d13=copy_org_img.copy()
d14=copy_org_img.copy()
conv_res1=copy_org_img.copy()
conv_res2=copy_org_img.copy()
conv_res3=copy_org_img.copy()
conv_res4=copy_org_img.copy()
conv_res5=copy_org_img.copy()
gauss1=np.zeros((val,val));
gauss2=np.zeros((val,val));
gauss3=np.zeros((val,val));
gauss4=np.zeros((val,val));
gauss5=np.zeros((val,val));
#list of sigma values for 1st octave
sigma1=[(float(1/math.sqrt(2))),1,math.sqrt(2),2,(2*math.sqrt(2))]
add_1=0.0
add_2=0.0
add_3=0.0
add_4=0.0
add_5=0.0
```

```
R1=0

R2=0

R3=0

R4=0

R5=0

a=-3

b=4

for i in range(a,b):

    for j in range(a,b):

        gauss1[i][j] =( math.exp( -(0.5 * float((pow((i)/sigma1[0], 2.0) + pow((j)/sigma1[0],2.0)
)))))/  (2 * math.pi * sigma1[0] * sigma1[0])

        gauss2[i][j] =( math.exp( -(0.5 * float((pow((i)/sigma1[1], 2.0) + pow((j)/sigma1[1],2.0)
)))))/  (2 * math.pi * sigma1[1] * sigma1[1])

        gauss3[i][j] =( math.exp( -(0.5 * float((pow((i)/sigma1[2], 2.0) + pow((j)/sigma1[2],2.0)
)))))/  (2 * math.pi * sigma1[2] * sigma1[2])

        gauss4[i][j] =( math.exp( -(0.5 * float((pow((i)/sigma1[3], 2.0) + pow((j)/sigma1[3],2.0)
)))))/  (2 * math.pi * sigma1[3] * sigma1[3])

        gauss5[i][j] =( math.exp( -(0.5 * float((pow((i)/sigma1[4], 2.0) + pow((j)/sigma1[4],2.0)
)))))/  (2 * math.pi * sigma1[4] * sigma1[4])

        #kernel[x][y]=math.exp(-0.5*())

        #Accumulate the kernel values

        add_1 = add_1+ gauss1[i][j]

        add_2 = add_2+ gauss2[i][j]

        add_3 = add_3+ gauss3[i][j]

        add_4 = add_4+ gauss4[i][j]

        add_5 = add_5+ gauss4[i][j]

#Normalize the kernel

for x in range(0,val):

    for y in range(0,val):

        gauss1[x][y]=gauss1[x][y]/add_1
```

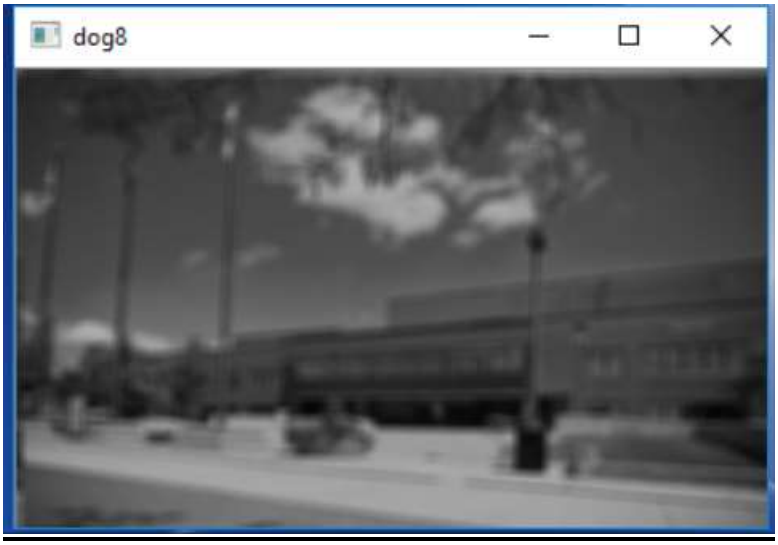
```
        gauss2[x][y]=gauss2[x][y]/add_2
        gauss3[x][y]=gauss3[x][y]/add_3
        gauss4[x][y]=gauss4[x][y]/add_4
        gauss5[x][y]=gauss5[x][y]/add_5
for x in range(0, rows):
    for y in range (0,column):
        for i in range(1,7):
            for j in range(1,7):
                R1+=gauss1[i-1][j-1]*org_img[x-(i-1)][y-(j-1)]
                R2+=gauss2[i-1][j-1]*org_img[x-(i-1)][y-(j-1)]
                R3+=gauss3[i-1][j-1]*org_img[x-(i-1)][y-(j-1)]
                R4+=gauss4[i-1][j-1]*org_img[x-(i-1)][y-(j-1)]
                R5+=gauss5[i-1][j-1]*org_img[x-(i-1)][y-(j-1)]
        conv_res1[x][y]=R1
        conv_res2[x][y]=R2
        conv_res3[x][y]=R3
        conv_res4[x][y]=R4
        conv_res5[x][y]=R5
```

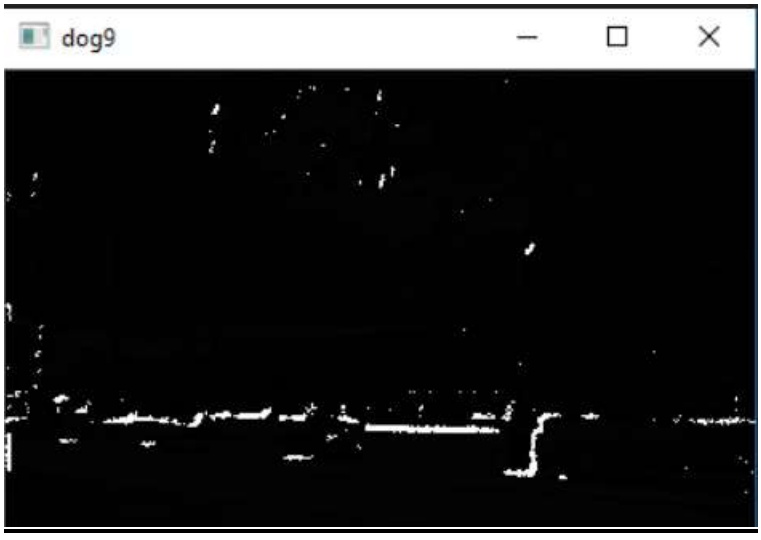
DOG's for second and third octave


dog5


dog6

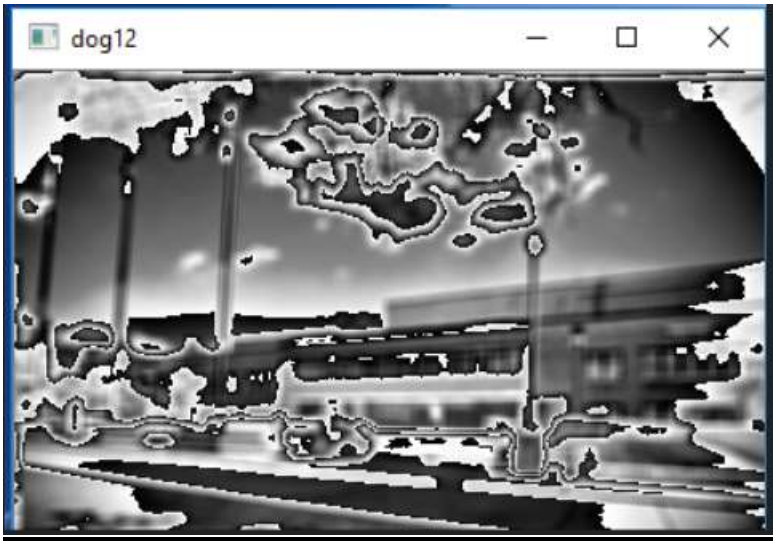

dog7

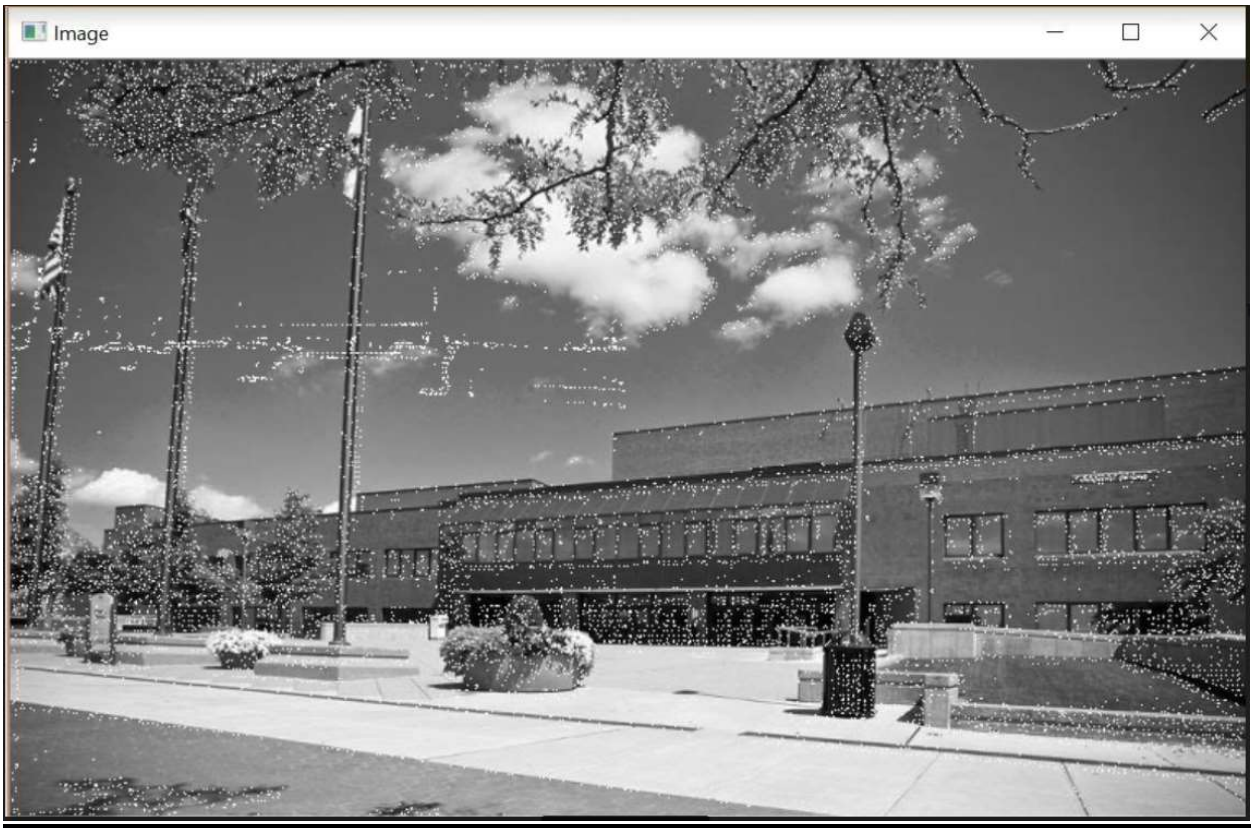

dog8


dog9


dog10


dog11





KeyPoint Detected image

```
h=copy_org_img.shape[0]
w=copy_org_img.shape[1]
findMaxima(d11,d12,d13)
```

```
findMaxima(d12,d13,d14)
h=resize1_img.shape[0]
w=resize1_img.shape[1]
findMaxima(d21,d22,d23)
findMaxima(d22,d23,d24)
h=resize2_img.shape[0]
w=resize2_img.shape[1]
findMaxima(d31,d32,d33)
findMaxima(d32,d33,d34)
h=resize3_img.shape[0]
w=resize3_img.shape[1]
findMaxima(d41,d42,d43)
findMaxima(d42,d43,d45)
cv2.imshow('Image',org_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

(Complete code for KeyPoint detection is not printed in report)

References for task2:

1. http://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/

Task 3: Cursor Detection

Code for task 3

```
import cv2
#read image of template and resize it
t_img = cv2.imread('C:/Users/Anupriya/Documents/task3/template.png')
t_img = cv2.resize(t_img , (0,0), fx=0.7, fy=0.7)
#read different images from the file and resize it
```

```
img_pos = cv2.imread('C:/Users/Anupriya/Documents/task3/pos_1.jpg')
img_pos = cv2.resize(img_pos, (0,0), fx=0.9, fy=0.9)
# convert into gray scale image
g_temp = cv2.cvtColor(t_img , cv2.COLOR_BGR2GRAY)
# Convert to grayscale image
g_img = cv2.cvtColor(img_pos, cv2.COLOR_BGR2GRAY)
# find the width and height of template
width=g_temp.shape[0]
height=g_temp.shape[1]
#sigma = 0.3*((3-1)*0.5 - 1) + 0.8
#gauss_res = cv2.GaussianBlur(img,(3,3),sigma)
#gauss_res = cv2.medianBlur(img,3)
#gauss_res = cv2.Laplacian(dst,cv2.CV_64F,0)
#gauss_template = cv2.Laplacian(template,cv2.CV_64F,0)
#result = cv2.matchTemplate(dst,dst_template,cv2.TM_CCOEFF_NORMED)
#Now,first blur the image with Gaussian Blur method and Second apply Laplacian
gauss_res = cv2.GaussianBlur(g_img,(3,3),0)
gauss_res = cv2.Laplacian(gauss_res,cv2.CV_32F,0)
gauss_temp_res = cv2.Laplacian(g_temp,cv2.CV_32F,0)
# the method used for template matching is cv2.TM_SQDIFF
k = cv2.matchTemplate(gauss_res,gauss_temp_res, cv2.TM_SQDIFF)
_,_,position,_ = cv2.minMaxLoc(k)
pos_right = (position[0] + width, position[1] + height)
cv2.rectangle(img_pos,position, pos_right,255,2)
#threshold = 0.8
#loc = np.where( res >= threshold)

#for pt in zip(*loc[::-1]):
```

```
#cv2.rectangle(img, pt, (pt[0] + w, pt[1] + h), (0,255,255), 2)


cv2.imshow("Image_result", img_pos)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

The code added in comments above is the code which was used earlier to detect cursor. It was not that efficient in detecting cursor then made changes and used new functions to detect the cursor. Earlier sigma value was used to blur the image using gaussain blur method and threshold value was set.But it was not given appropriate result. Then after that changed the method to get blur image and resized the template and image.

For task 3 bonus images the program is working but need to give template for it. These templates are taken from images itself.

Attaching the images of the template. Just need to add the path of these templates and the images on which these cursors are to be detected. Created three templates for three different cursors and then passed these template to the program for detecting of these cursors.

References for Task3: Cursor Detection

1. https://docs.opencv.org/3.4.0/d4/dc6/tutorial_py_template_matching.html
2. http://aishack.in/tutorials/image-convolution-examples/