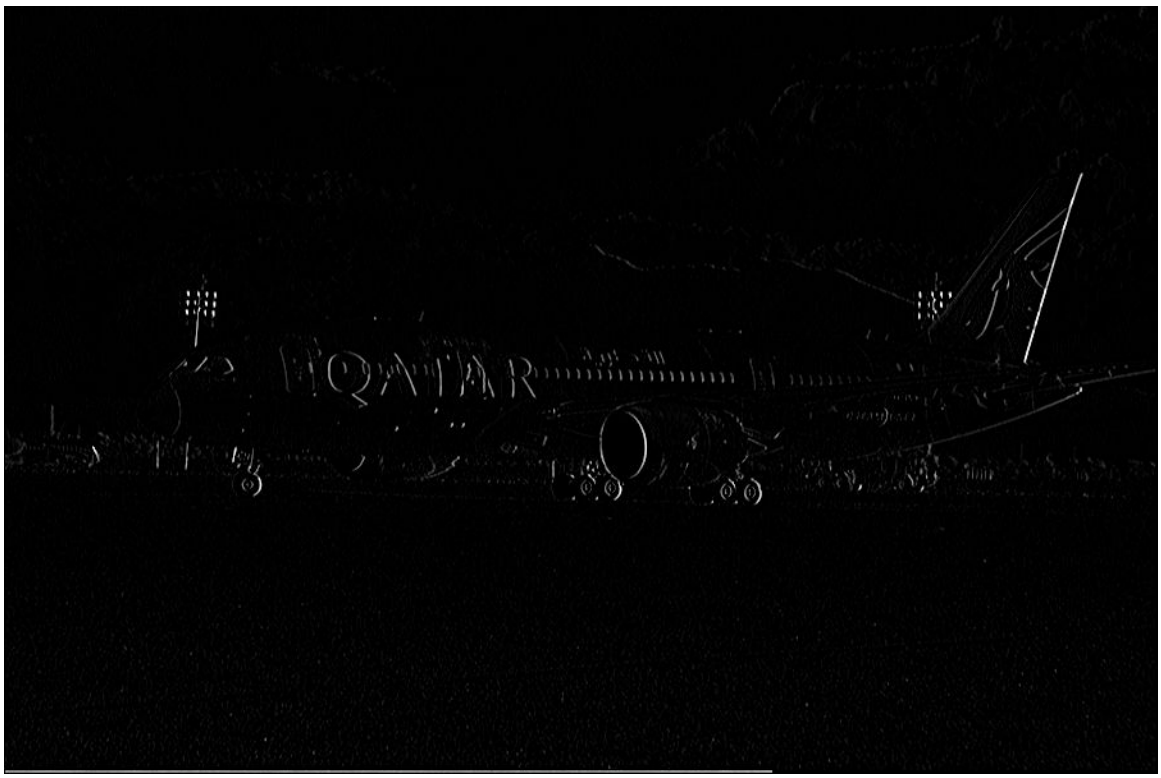
CSE 573 : PROJECT 1 REPORT

NAME : MANASI KULKARNI
UBIT NAME : mkulkarn
STUDENT NUMBER : 50288702

TASK 1: EDGE DETECTION



Edges along X-direction

Edges along Y-direction

Edges along X and Y direction

Source Code :

```
import cv2
import math
import numpy
readImage = cv2.imread("/Users/manasikulkarni/Desktop/task1.png",cv2.IMREAD_GRAYSCALE)
imgArray = numpy.asarray(readImage)
row = len(imgArray)
col = len(imgArray[0])
newimgX = numpy.array([[0 for i in range(int(col+2))] for j in range(int(row+2) )],dtype = numpy.float)
newimgY = numpy.array([[0 for i in range(int(col+2))] for j in range(int(row+2) )],dtype = numpy.float)
newimgCommon = numpy.array([[0 for i in range(int(col+2))] for j in range(int(row+2) )],dtype = numpy.float)
for i in range(0,row+1):
    for j in range(0,col+1):
        if i == 0 or j==0 or i >= col or j >= row :
            newimgX[i,j] = 0
            newimgY[i,j] = 0
            newimgCommon[i,j] = 0
        else:
            newimgX[i,j] = readImage[i-1,j-1]
            newimgY[i,j] = readImage[i-1,j-1]
            newimgCommon[i,j] = readImage[i-1,j-1]
pixel = readImage[0,0]
def pixel00(i,j,val):
    pixel = readImage[i,j]
    if val == 0:
        return (-1 *(pixel))
    elif val == 1:
        return (-1 *(pixel))
def pixel01(i,j,val):
    pixel = readImage[i,j]
```

```
    if val == 0:
        return (-2 * (pixel))
    elif val == 1:
        return 0
def pixel02(i,j,val):
    pixel = readImage[i,j]
    if val == 0:
        return (-1 * (pixel))
    elif val == 1:
        return pixel
def pixel10(i,j,val):
    if val == 0:
        return 0
    elif val == 1:
        return (-2 * (pixel))
def pixel11(i,j,val):
    if val == 0:
        return 0
    elif val == 1:
        return 0
def pixel12(i,j,val):
    if val == 0:
        return 0
    elif val == 1:
        return (2 * (pixel))
def pixel20(i,j,val):
    pixel = readImage[i,j]
    if val == 0:
        return pixel
    elif val == 1:
        return (-1 * (pixel))
def pixel21(i,j,val):
    pixel = readImage[i,j]
    if val == 0:
        return (2 * (pixel))
    elif val == 1:
        return 0
def pixel22(i,j,val):
    pixel = readImage[i,j]
    if val == 0:
        return  pixel
    elif val == 1:
        return  pixel
for x in range(1, row-1):
    for y in range(1, col-1):
        gradientX = 0
        gradientY = 0
        gradientX += pixel00(x-1,y-1,0)
        gradientX += pixel10(x-1,y,0)
        gradientX += pixel20(x-1,y+1,0)
        gradientX += pixel01(x,y-1,0)
        gradientX += pixel11(x,y,0)
        gradientX += pixel21(x,y+1,0)
        gradientX += pixel02(x+1,y-1,0)
        gradientX += pixel12(x+1,y,0)
        gradientX += pixel22(x+1,y+1,0)
        gradientY += pixel00(x-1,y-1,1)
```

```
        gradientY += pixel10(x-1,y,1)
        gradientY += pixel20(x-1,y+1,1)
        gradientY += pixel01(x,y-1,1)
        gradientY += pixel11(x,y,1)
        gradientY += pixel21(x,y+1,1)
        gradientY += pixel02(x+1,y-1,1)
        gradientY += pixel12(x+1,y,1)
        gradientY += pixel22(x+1,y+1,1)
        lengthX = gradientX
        if lengthX < 0 :
            lengthX = 0
        lengthY = gradientY
        if lengthY < 0 :
            lengthY = 0
        lengthCommon = math.sqrt((lengthX * lengthX)+(lengthY * lengthY))
        lengthX = int(lengthX/800 * 255)
        lengthY = int(lengthY/400 * 255)
        newimgX[x,y] = lengthX
        newimgY[x,y] = lengthY
        newimgCommon[x,y] = lengthCommon
cv2.imwrite('imageX.png', newimgX)
cv2.imwrite('imageY.png', newimgY)
cv2.imwrite('imageCommon.png', newimgCommon)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

TASK 2: KEYPOINT DETECTION - (1)

The images in the second octave are of size 379 * 229 pixel. The images in the second octave are as follows:

Image 1 in Octave 2

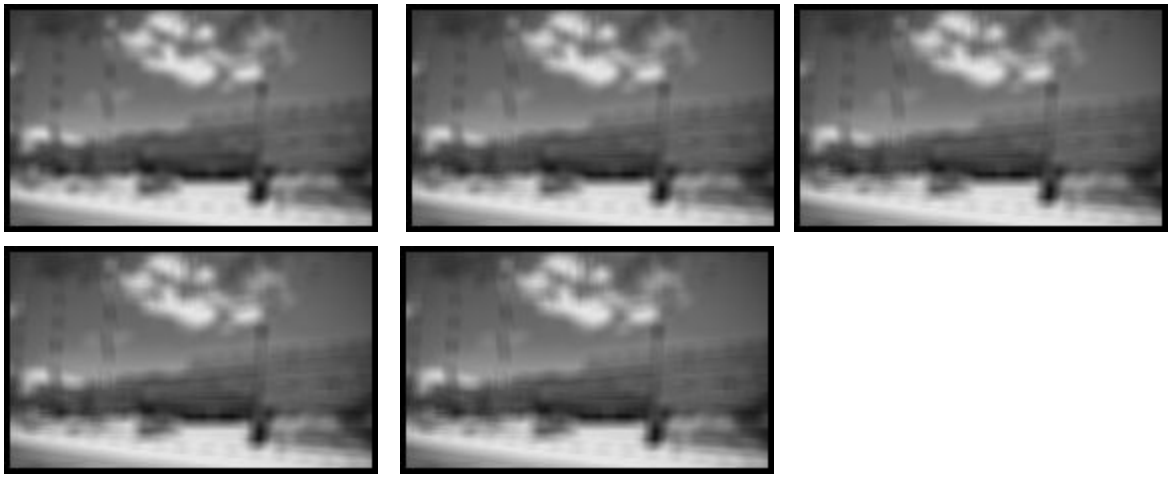

Image 2 in Octave 2


Image 3 in Octave 2


Image 4 in Octave 2


Image 5 in Octave 2

The images in the third octave are of size 187 * 114 pixel. The images in the third octave are as follows:



(2):

The DoG images obtained using the second octave are as follows:


DoG 1 in Octave 2


DoG 2 in Octave 2

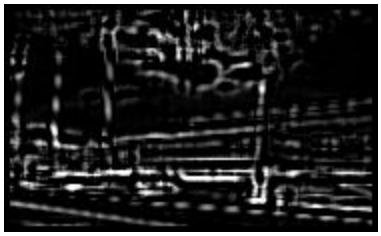

DoG 3 in Octave 2

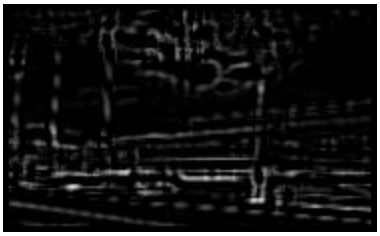

DoG 4 in Octave 2

The DoG images obtained using the third octave are as follows:

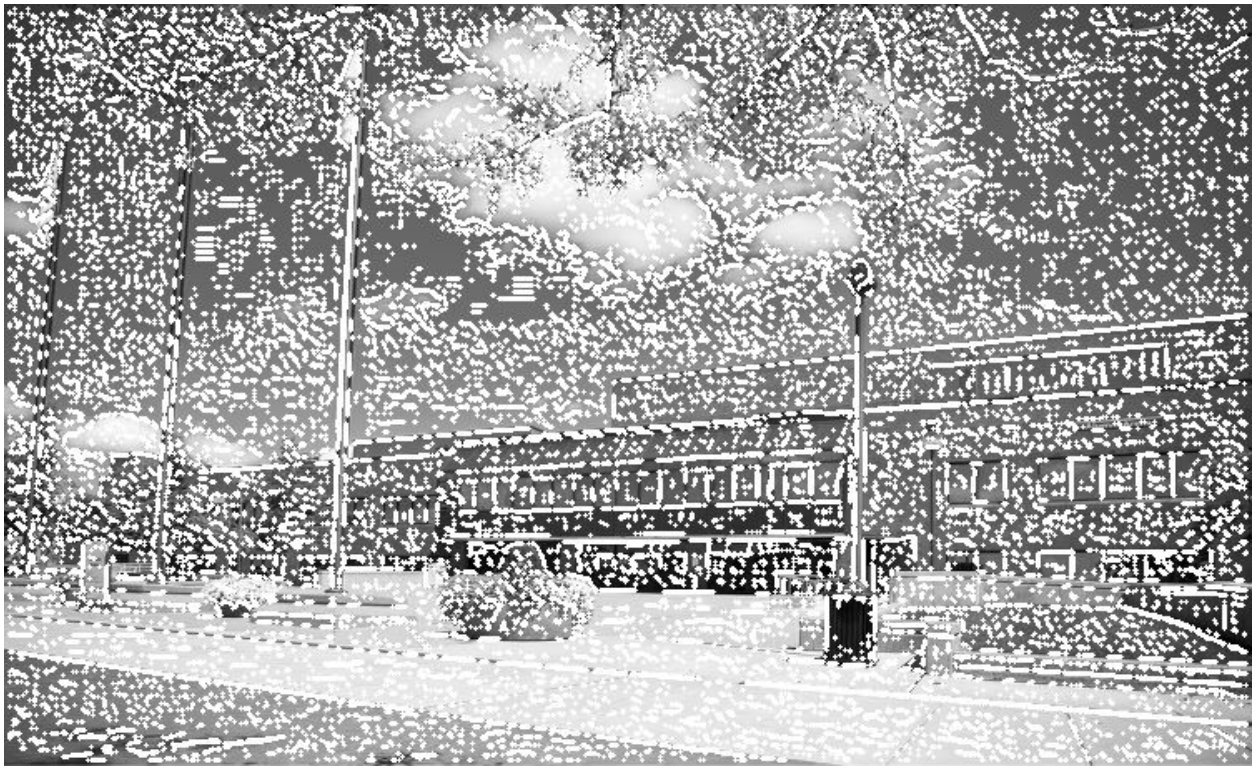

DoG 1 in Octave 3


DoG 2 in Octave 3


DoG 3 in Octave 3


DoG 4 in Octave 3

(3):



Detected Keypoints

(4): The 5 left-most detected keypoints are : (3,6), (3,14), (3,77), (3,108), (4,102)

Source Code : (for Octave 2 and Octave 3)

```
import cv2
import math
import numpy

def Octave2Computation():
    readImage = cv2.imread("/Users/manasikulkarni/Desktop/task2.jpg",cv2.IMREAD_GRAYSCALE)
    height, width = readImage.shape

    imageBy2 = numpy.array([[0 for i in range(int(width/2))] for j in range(int(height/2))],dtype = numpy.float)
    imageBy21 = numpy.array([[0 for i in range(int(width/2))] for j in range(int(height/2))],dtype = numpy.float)
    imageBy22 = numpy.array([[0 for i in range(int(width/2))] for j in range(int(height/2))],dtype = numpy.float)
    imageBy23 = numpy.array([[0 for i in range(int(width/2))] for j in range(int(height/2))],dtype = numpy.float)
    imageBy24 = numpy.array([[0 for i in range(int(width/2))] for j in range(int(height/2))],dtype = numpy.float)
    imageBy25 = numpy.array([[0 for i in range(int(width/2))] for j in range(int(height/2))],dtype = numpy.float)
    dogOctave2Image1 = numpy.array([[0 for i in range(int(width/2))] for j in range(int(height/2) )],dtype = numpy.float)
    dogOctave2Image2 = numpy.array([[0 for i in range(int(width/2))] for j in range(int(height/2) )],dtype = numpy.float)
    dogOctave2Image3 = numpy.array([[0 for i in range(int(width/2))] for j in range(int(height/2) )],dtype = numpy.float)
    dogOctave2Image4 = numpy.array([[0 for i in range(int(width/2))] for j in range(int(height/2) )],dtype = numpy.float)
    for x in range(0,int(height/2)):
        for y in range(0,int(width/2)):
            imageBy2[x,y] = readImage[x*2,y*2]
    octave2 = [math.sqrt(2) , 2 , 2 * math.sqrt(2) , 4 , 4 * math.sqrt(2)]

    gaussianSum1 = 0.0
    gaussianSum2 = 0.0
    gaussianSum3 = 0.0
    gaussianSum4 = 0.0
    gaussianSum5 = 0.0
    gaussianKernel1 = []
    gaussianKernel2 = []
    gaussianKernel3 = []
    gaussianKernel4 = []
    gaussianKernel5 = []
    for x in range(-3,4):
        for y in range(3,-4,-1):
            gaussianResult1 = float(1/float(2 * math.pi * octave2[0] * octave2[0])) * numpy.exp(-((float(x * x + y * y))/(2 * octave2[0] *
octave2[0])))
            gaussianSum1 += gaussianResult1
            gaussianKernel1.append(gaussianResult1)
            gaussianResult2 = float(1/float(2 * math.pi * octave2[1] * octave2[1])) * numpy.exp(-((float(x * x + y * y))/(2 * octave2[1] *
octave2[1])))
            gaussianSum2 += gaussianResult2
            gaussianKernel2.append(gaussianResult2)
            gaussianResult3 = float(1/float(2 * math.pi * octave2[2] * octave2[2])) * numpy.exp(-((float(x * x + y * y))/(2 * octave2[2] *
octave2[2])))
            gaussianSum3 += gaussianResult3
            gaussianKernel3.append(gaussianResult3)
            gaussianResult4 = float(1/float(2 * math.pi * octave2[3] * octave2[3])) * numpy.exp(-((float(x * x + y * y))/(2 * octave2[3] *
octave2[3])))
            gaussianSum4 += gaussianResult4
            gaussianKernel4.append(gaussianResult4)
            gaussianResult5 = float(1/float(2 * math.pi * octave2[4] * octave2[4])) * numpy.exp(-((float(x * x + y * y))/(2 * octave2[4] *
octave2[4])))
            gaussianSum5 += gaussianResult5
            gaussianKernel5.append(gaussianResult5)
```

```
    for i in range(0,len(gaussianKernel1)):
        gaussianKernel1[i] = gaussianKernel1[i] / float(gaussianSum1)
        gaussianKernel2[i] = gaussianKernel2[i] / float(gaussianSum2)
        gaussianKernel3[i] = gaussianKernel3[i] / float(gaussianSum3)
        gaussianKernel4[i] = gaussianKernel4[i] / float(gaussianSum4)
        gaussianKernel5[i] = gaussianKernel5[i] / float(gaussianSum5)
    for x in range(3, int(height/2)-3):
        for y in range(3, int(width/2)-3):
            GResult1 = 0
            GResult2 = 0
            GResult3 = 0
            GResult4 = 0
            GResult5 = 0
            var = 0
            for i in range(-3,4):
                for j in range(-3,4):
                    pixel = imageBy2[x+i,y+j]
                    GResult1 += pixel * gaussianKernel1[var]
                    GResult2 += pixel * gaussianKernel2[var]
                    GResult3 += pixel * gaussianKernel3[var]
                    GResult4 += pixel * gaussianKernel4[var]
                    GResult5 += pixel * gaussianKernel5[var]
                    var = var + 1
            imageBy21[x,y] = GResult1
            imageBy22[x,y] = GResult2
            imageBy23[x,y] = GResult3
            imageBy24[x,y] = GResult4
            imageBy25[x,y] = GResult5
            dogOctave2Image1[x,y] = ((imageBy22[x,y] - imageBy21[x,y])*255)
            dogOctave2Image2[x,y] = ((imageBy23[x,y] - imageBy22[x,y])*255)
            dogOctave2Image3[x,y] = ((imageBy24[x,y] - imageBy23[x,y])*255)
            dogOctave2Image4[x,y] = ((imageBy25[x,y] - imageBy24[x,y])*255)
    cv2.imwrite('image1octave2.png', imageBy21)
    cv2.imwrite('image2octave2.png', imageBy22)
    cv2.imwrite('image3octave2.png', imageBy23)
    cv2.imwrite('image4octave2.png', imageBy24)
    cv2.imwrite('image5octave2.png', imageBy25)
    cv2.imwrite('dog1octave2.png', dogOctave2Image1)
    cv2.imwrite('dog2octave2.png', dogOctave2Image2)
    cv2.imwrite('dog3octave2.png', dogOctave2Image3)
    cv2.imwrite('dog4octave2.png', dogOctave2Image4)

    keyPointsList = []
    for x in range(3, int(height/2)-3):
        for y in range(3, int(width/2)-3):
            isMaximum = True
            isMinimum = True
            for i in range(-1,2):
                for j in range(-1,2):
                    if dogOctave2Image2[x,y] > dogOctave2Image2[x+i,y+j] and dogOctave2Image2[x,y] > dogOctave2Image1[x,y] and
dogOctave2Image2[x,y] > dogOctave2Image1[x+i,y+j] and dogOctave2Image2[x,y] > dogOctave2Image3[x,y] and
dogOctave2Image2[x,y] > dogOctave2Image3[x+i,y+j]:
                        if x==x+i and y==y+j:
                            break
                        isMaximum = True
                        isMinimum = False
                        break
```

```
                else:
                    isMaximum = False
                    isMinimum = False
                    break
                if dogOctave2Image2[x,y] < dogOctave2Image2[x+i,y+j] and dogOctave2Image2[x,y] < dogOctave2Image1[x,y] and
dogOctave2Image2[x,y] < dogOctave2Image1[x+i,y+j] and dogOctave2Image2[x,y] < dogOctave2Image3[x,y] and
dogOctave2Image2[x,y] < dogOctave2Image3[x+i,y+j]:
                    if x==x+i and y==y+j:
                        break
                    isMinimum = True
                    isMaximum = False
                    break
                else:
                    isMinimum = False
                    isMaximum = False
                    break
            if isMaximum:
                keyPointsList.append((y,x))
            elif isMinimum:
                keyPointsList.append((y,x))
    for i in range(0,len(keyPointsList)):
        cv2.circle(imageBy2,(keyPointsList[i][0],keyPointsList[i][1]), 1, (255,255,255), -1)
    cv2.imwrite('keypoints1octave2.png', imageBy2)

    keyPointsList = []
    for x in range(3, int(height/2)-3):
        for y in range(3, int(width/2)-3):
            isMaximum = True
            isMinimum = True
            for i in range(-1,2):
                for j in range(-1,2):
                    if dogOctave2Image3[x,y] > dogOctave2Image3[x+i,y+j] and dogOctave2Image3[x,y] > dogOctave2Image2[x,y] and
dogOctave2Image3[x,y] > dogOctave2Image2[x+i,y+j] and dogOctave2Image3[x,y] > dogOctave2Image4[x,y] and
dogOctave2Image3[x,y] > dogOctave2Image4[x+i,y+j]:
                        if x==x+i and y==y+j:
                            break
                        isMaximum = True
                        isMinimum = False
                        break
                    else:
                        isMaximum = False
                        isMinimum = False
                        break
                    if dogOctave2Image3[x,y] < dogOctave2Image3[x+i,y+j] and dogOctave2Image3[x,y] < dogOctave2Image2[x,y] and
dogOctave2Image3[x,y] < dogOctave2Image2[x+i,y+j] and dogOctave2Image3[x,y] < dogOctave2Image4[x,y] and
dogOctave2Image3[x,y] < dogOctave2Image4[x+i,y+j]:
                        if x==x+i and y==y+j:
                            break
                        isMinimum = True
                        isMaximum = False
                        break
                    else:
                        isMinimum = False
                        isMaximum = False
                        break
            if isMaximum:
                keyPointsList.append((y,x))
```

```
            if isMinimum:
                keyPointsList.append((y,x))
    for i in range(0,len(keyPointsList)):
        cv2.circle(imageBy2,(keyPointsList[i][0],keyPointsList[i][1]), 1, (255,255,255), -1)
    cv2.imwrite('keypoints2octave2.png', imageBy2)

def Octave3Computation():
    readImage = cv2.imread("/Users/manasikulkarni/Desktop/task2.jpg",cv2.IMREAD_GRAYSCALE)
    height, width = readImage.shape

    imageBy4 = numpy.array([[0 for i in range(int(width/4 ))] for j in range(int(height/4) )],dtype = numpy.float)
    imageBy41 = numpy.array([[0 for i in range(int(width/4 ))] for j in range(int(height/4) )],dtype = numpy.float)
    imageBy42 = numpy.array([[0 for i in range(int(width/4 ))] for j in range(int(height/4) )],dtype = numpy.float)
    imageBy43 = numpy.array([[0 for i in range(int(width/4 ))] for j in range(int(height/4) )],dtype = numpy.float)
    imageBy44 = numpy.array([[0 for i in range(int(width/4 ))] for j in range(int(height/4) )],dtype = numpy.float)
    imageBy45 = numpy.array([[0 for i in range(int(width/4 ))] for j in range(int(height/4) )],dtype = numpy.float)
    dogOctave3Image1 = numpy.array([[0 for i in range(int(width/4 ))] for j in range(int(height/4) )],dtype = numpy.float)
    dogOctave3Image2 = numpy.array([[0 for i in range(int(width/4 ))] for j in range(int(height/4) )],dtype = numpy.float)
    dogOctave3Image3 = numpy.array([[0 for i in range(int(width/4 ))] for j in range(int(height/4) )],dtype = numpy.float)
    dogOctave3Image4 = numpy.array([[0 for i in range(int(width/4 ))] for j in range(int(height/4) )],dtype = numpy.float)
    for x in range(0,int(height/4)):
        for y in range(0,int(width/4)):
            imageBy4[x,y] = readImage[x*4,y*4]
    octave3 = [2 * math.sqrt(2) , 4 , 4 * math.sqrt(2) , 8 , 8 * math.sqrt(2)]
    gaussianSum1 = 0.0
    gaussianSum2 = 0.0
    gaussianSum3 = 0.0
    gaussianSum4 = 0.0
    gaussianSum5 = 0.0
    gaussianKernel1 = []
    gaussianKernel2 = []
    gaussianKernel3 = []
    gaussianKernel4 = []
    gaussianKernel5 = []
    for x in range(-3,4):
        for y in range(3,-4,-1):
            gaussianResult1 = float(1/float(2 * math.pi * octave3[0] * octave3[0])) * numpy.exp(-((float(x * x + y * y))/(2 * octave3[0] *
octave3[0])))
            gaussianSum1 += gaussianResult1
            gaussianKernel1.append(gaussianResult1)
            gaussianResult2 = float(1/float(2 * math.pi * octave3[1] * octave3[1])) * numpy.exp(-((float(x * x + y * y))/(2 * octave3[1] *
octave3[1])))
            gaussianSum2 += gaussianResult2
            gaussianKernel2.append(gaussianResult2)
            gaussianResult3 = float(1/float(2 * math.pi * octave3[2] * octave3[2])) * numpy.exp(-((float(x * x + y * y))/(2 * octave3[2] *
octave3[2])))
            gaussianSum3 += gaussianResult3
            gaussianKernel3.append(gaussianResult3)
            gaussianResult4 = float(1/float(2 * math.pi * octave3[3] * octave3[3])) * numpy.exp(-((float(x * x + y * y))/(2 * octave3[3] *
octave3[3])))
            gaussianSum4 += gaussianResult4
            gaussianKernel4.append(gaussianResult4)
            gaussianResult5 = float(1/float(2 * math.pi * octave3[4] * octave3[4])) * numpy.exp(-((float(x * x + y * y))/(2 * octave3[4] *
octave3[4])))
            gaussianSum5 += gaussianResult5
            gaussianKernel5.append(gaussianResult5)
    for i in range(0,len(gaussianKernel1)):
```

```
        gaussianKernel1[i] = gaussianKernel1[i] / float(gaussianSum1)
        gaussianKernel2[i] = gaussianKernel2[i] / float(gaussianSum2)
        gaussianKernel3[i] = gaussianKernel3[i] / float(gaussianSum3)
        gaussianKernel4[i] = gaussianKernel4[i] / float(gaussianSum4)
        gaussianKernel5[i] = gaussianKernel5[i] / float(gaussianSum5)
    for x in range(3, int(height/4)-3):
        for y in range(3, int(width/4)-3):
            GResult1 = 0
            GResult2 = 0
            GResult3 = 0
            GResult4 = 0
            GResult5 = 0
            var = 0
            for i in range(-3,4):
                for j in range(-3,4):
                    pixel = imageBy4[x+i,y+j]
                    GResult1 += pixel * gaussianKernel1[var]
                    GResult2 += pixel * gaussianKernel2[var]
                    GResult3 += pixel * gaussianKernel3[var]
                    GResult4 += pixel * gaussianKernel4[var]
                    GResult5 += pixel * gaussianKernel5[var]
                    var = var + 1
            imageBy41[x,y] = GResult1
            imageBy42[x,y] = GResult2
            imageBy43[x,y] = GResult3
            imageBy44[x,y] = GResult4
            imageBy45[x,y] = GResult5
            dogOctave3Image1[x,y] = ((imageBy42[x,y] - imageBy41[x,y]) * 255)
            dogOctave3Image2[x,y] = ((imageBy43[x,y] - imageBy42[x,y]) * 255)
            dogOctave3Image3[x,y] = ((imageBy44[x,y] - imageBy43[x,y]) * 255)
            dogOctave3Image4[x,y] = ((imageBy45[x,y] - imageBy44[x,y]) * 255)
    cv2.imwrite('image1octave3.png', imageBy41)
    cv2.imwrite('image2octave3.png', imageBy42)
    cv2.imwrite('image3octave3.png', imageBy43)
    cv2.imwrite('image4octave3.png', imageBy44)
    cv2.imwrite('image5octave3.png', imageBy45)
    cv2.imwrite('dog1octave3.png', dogOctave3Image1)
    cv2.imwrite('dog2octave3.png', dogOctave3Image2)
    cv2.imwrite('dog3octave3.png', dogOctave3Image3)
    cv2.imwrite('dog4octave3.png', dogOctave3Image4)

    keyPointsList = []
    for x in range(3, int(height/4)-3):
        for y in range(3, int(width/4)-3):
            isMaximum = True
            isMinimum = True
            for i in range(-1,2):
                for j in range(-1,2):
                    if dogOctave3Image2[x,y] > dogOctave3Image2[x+i,y+j] and dogOctave3Image2[x,y] > dogOctave3Image1[x,y] and
dogOctave3Image2[x,y] > dogOctave3Image1[x+i,y+j] and dogOctave3Image2[x,y] > dogOctave3Image3[x,y] and
dogOctave3Image2[x,y] > dogOctave3Image3[x+i,y+j]:
                        if x==x+i and y==y+j:
                            break
                        isMaximum = True
                        isMinimum = False
                        break
                    else:
```

```
                    isMaximum = False
                    isMinimum = False
                    break
                if dogOctave3Image2[x,y] < dogOctave3Image2[x+i,y+j] and dogOctave3Image2[x,y] < dogOctave3Image1[x,y] and
dogOctave3Image2[x,y] < dogOctave3Image1[x+i,y+j] and dogOctave3Image2[x,y] < dogOctave3Image3[x,y] and
dogOctave3Image2[x,y] < dogOctave3Image3[x+i,y+j]:
                    if x==x+i and y==y+j:
                        break
                    isMinimum = True
                    isMaximum = False
                    break
                else:
                    isMinimum = False
                    isMaximum = False
                    break
            if isMaximum:
                keyPointsList.append((y,x))
            elif isMinimum:
                keyPointsList.append((y,x))
    for i in range(0,len(keyPointsList)):
        cv2.circle(imageBy4,(keyPointsList[i][0],keyPointsList[i][1]), 1, (255,255,255), -1)
    cv2.imwrite('keypoints1octave3.png', imageBy4)

    keyPointsList = []
    for x in range(3, int(height/4)-3):
        for y in range(3, int(width/4)-3):
            isMaximum = True
            isMinimum = True
            for i in range(-1,2):
                for j in range(-1,2):
                    if dogOctave3Image3[x,y] > dogOctave3Image3[x+i,y+j] and dogOctave3Image3[x,y] > dogOctave3Image2[x,y] and
dogOctave3Image3[x,y] > dogOctave3Image2[x+i,y+j] and dogOctave3Image3[x,y] > dogOctave3Image4[x,y] and
dogOctave3Image3[x,y] > dogOctave3Image4[x+i,y+j]:
                        if x==x+i and y==y+j:
                            break
                        isMaximum = True
                        isMinimum = False
                        break
                    else:
                        isMaximum = False
                        isMinimum = False
                        break
                    if dogOctave3Image3[x,y] < dogOctave3Image3[x+i,y+j] and dogOctave3Image3[x,y] < dogOctave3Image2[x,y] and
dogOctave3Image3[x,y] < dogOctave3Image2[x+i,y+j] and dogOctave3Image3[x,y] < dogOctave3Image4[x,y] and
dogOctave3Image3[x,y] < dogOctave3Image4[x+i,y+j]:
                        if x==x+i and y==y+j:
                            break
                        isMinimum = True
                        isMaximum = False
                        break
                    else:
                        isMinimum = False
                        isMaximum = False
                        break
            if isMaximum:
                keyPointsList.append((y,x))
            if isMinimum:
```

```
            keyPointsList.append((y,x))
    for i in range(0,len(keyPointsList)):
        cv2.circle(imageBy4,(keyPointsList[i][0],keyPointsList[i][1]), 1, (255,255,255), -1)
    cv2.imwrite('keypoints2octave3.png', imageBy4)
```

Same approach is used for Octave 1 and Octave 4

TASK 3: CURSOR DETECTION

Approach:

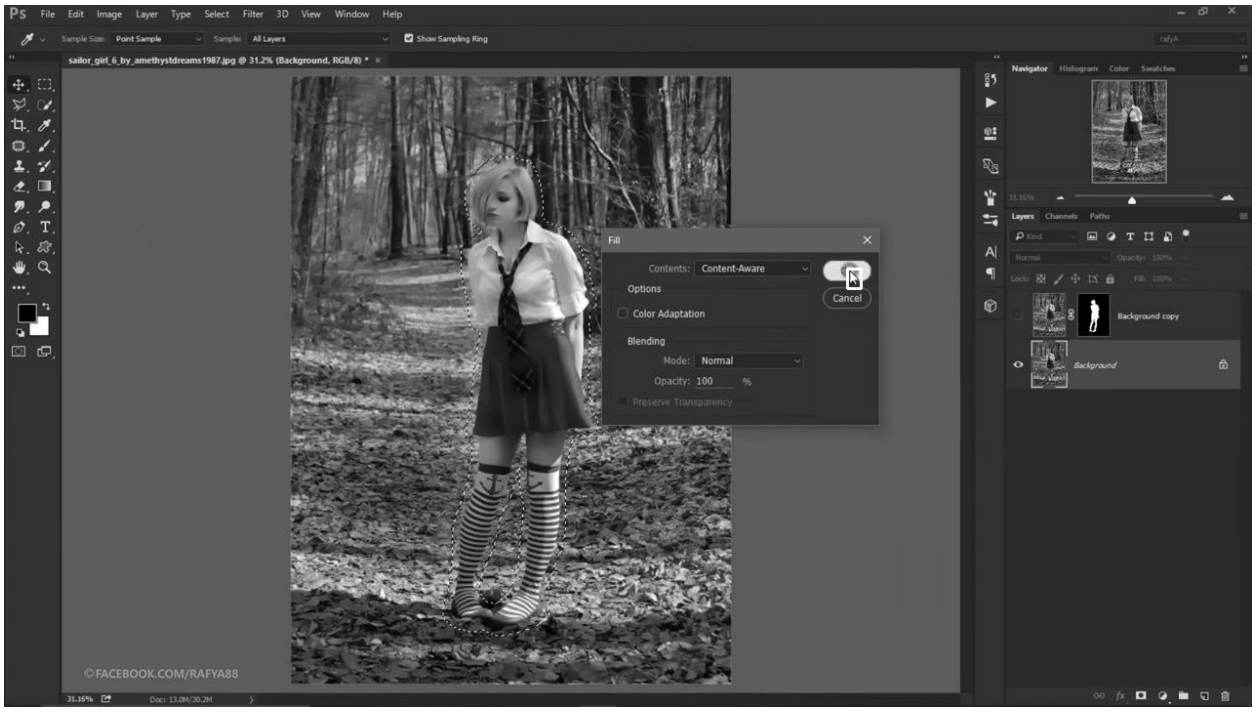
1. Read the image on which the cursor is to be detected
2. Read the template image
3. Resize the template image into half
4. Use 3*3 Gaussian Kernel to blur both the images. I have used Gaussian Blur to reduce the image noise. The image is smoothened. This is done by using the OpenCV function cv2.GaussianBlur. This function takes the parameters as the image, size of Kernel and the sigma value. I have used 3*3 kernel
5. Apply Laplacian transformation to both the images. I have used cv2.Laplacian function which takes parameters as the image and ddepth. I am passing the image obtained after the Gaussian blur and passing ddepth as 0.
6. Use template matching to match both the images. I have made use of cv2.tenplateMatch function, which takes parameters as the 2 images which need to be compared and the method. The method can be either of the following types : cv2.TM_CCOEFF, cv2.TM_CCOEFF_NORMED, cv2.TM_CCOR, cv2.TM_CCOR_NORMED, cv2.TM_SQDIFF, cv2.TM_SQDIFF_NORMED.
7. I have used cv2.TM_CCOEFF_NORMED, because it gives the best match.
8. Then I have used cv2.mimMaxLoc, which gives the minimum and maximum values and their positions in the array

Source Code:

```
import cv2
import numpy as np

readImage = cv2.imread('/Users/manasikulkarni/Desktop/Task3/pos_11.jpg',cv2.IMREAD_GRAYSCALE)
template = cv2.imread('/Users/manasikulkarni/Desktop/template.png',cv2.IMREAD_GRAYSCALE)

resized_image = cv2.resize(template, (0,0), fx=0.5, fy=0.5)
w, h= resized_image.shape[::-1]
mainImageBlur = cv2.GaussianBlur(readImage,(3,3),0)
templateImageBlur = cv2.GaussianBlur(resized_image,(3,3),0)

main_Laplacian = cv2.Laplacian(mainImageBlur,0)
template_Laplacian = cv2.Laplacian(templateImageBlur,0)
res = cv2.matchTemplate(main_Laplacian.astype(np.uint8),template_Laplacian.astype(np.uint8),cv2.TM_CCOEFF_NORMED)
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
top_left = max_loc
print(top_left)
bottom_right = (top_left[0] + w, top_left[1] + h)
print(bottom_right)
cv2.rectangle(readImage, top_left, bottom_right, 255, 2)
```

```
cv2.imwrite('abc.jpg', readImage)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:



The detected cursor is highlighted with a white-bordered rectangle