Name- Yash Shirodkar
Net ID- yps224
N Number- N18264436

1. Source Code Name- CannyEdgeDetector.py

2. Compile Instructions- py CannyEdgeDetector.py

3. Source Code-

```
import numpy as np
import cv2
import imageio as img
import matplotlib.pyplot as plt
import math

def Gaussian_Smoothing(image, kernel):
    output = np.zeros_like(image,dtype=float) #convolution output
    image_padded = np.zeros((image.shape[0]+6, image.shape[1]+6)) #Add zero padding to the
input image
    image_padded[3:-3, 3:-3] = image
    for x in range(image.shape[0]):
        for y in range(image.shape[1]):
            #element-wise multiplication of the kernel and the image
            if (x>=3 and x<=image.shape[0]-4) and (y>=3 and y<=image.shape[1]-4):
                output[x,y]=np.sum(kernel*image_padded[x:x+7,y:y+7])/140
                #normalised by dividing by 140 which is the sum of the mask
            else:
                output[x,y]=0
                 #pixel values of first 4 rows 4 columns and last 4 rows 4 columns will be undefined
    return output


#-----------------------------------------------------------------------------------------------------------------

img = img.imread('Zebra-crossing-1.bmp')    #Load the image. Image and source code should
be in same folder. Use ‘zebra.bmp’ for other image
kernel = 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],
```

Name- Yash Shirodkar
Net ID- yps224
N Number- N18264436

```
                [2,2,4,8,4,2,2],
                [1,2,2,4,2,2,1],
                [1,1,2,2,2,1,1]]) #gaussian mask

px = np.array([[-1, 0, 1],
               [-2, 0, 2],
               [-1, 0, 1]]) #sobel horizontal operator
py = np.array([[1, 2, 1],
               [0, 0, 0],
               [-1, -2, -1]]) #sobel vertical operator


gaussian = Gaussian_Smoothing(img,kernel) #result after Gaussian Smoothing
plt.imshow(gaussian, cmap=plt.cm.gray)
plt.title("After Guassian Smoothing")
plt.axis('off')
plt.show()
cv2.imwrite('Step1-After_Guassian_Smoothing.jpg',gaussian)

#-------------------------------------------------------------------------------------------------------

def Gradient_Operator(gaussian, op):
    ans = np.zeros_like(gaussian) #gx and gy output
    image_padded = np.zeros((gaussian.shape[0]+2, gaussian.shape[1]+2)) #Add zero padding
to the input image
    image_padded[1:-1, 1:-1] = gaussian
    for x in range(gaussian.shape[0]):
        for y in range(gaussian.shape[1]):
            #element-wise multiplication of respective horizontal and vertical operator and the image
            if (x<=3 or x>=gaussian.shape[0]-4) or (y<=3 or y>=gaussian.shape[1]-4):
                ans[x,y]=0
                #pixel values of first 4 rows 4 columns and last 4 rows 4 columns will be undefined
            else:
                ans[x,y]=(np.sum(op*image_padded[x:x+3,y:y+3]))/3
                #normalised by dividing by 3
    return ans

#-------------------------------------------------------------------------------------------------------

sobel = np.zeros_like(gaussian)
gx = np.zeros_like(gaussian)
gy = np.zeros_like(gaussian)
```

Name- Yash Shirodkar
Net ID- yps224
N Number- N18264436

```
gx = Gradient_Operator(gaussian, px) #horizontal gradient
gx1=abs(gx) #we take absolute values for display purpose
gy = Gradient_Operator(gaussian, py) #vertical gradient
gy1=abs(gy)
g = (np.sqrt((gx1 * gx1) + (gy1 * gy1))/math.sqrt(2)) #normalise the magnitude by root(2)
sobel = (np.arctan2(gy, gx) * (180/np.pi)) #calculate the edge angles

plt.imshow(gx1, cmap=plt.cm.gray)
plt.title("Horizontal Gradient")
plt.xticks([]), plt.yticks([])
plt.show()
cv2.imwrite('Step2-GX.jpg', gx1)

plt.imshow(gy1, cmap=plt.cm.gray)
plt.title("Vertical Gradient")
plt.xticks([]), plt.yticks([])
plt.show()
cv2.imwrite('Step2-GY.jpg',gy1)

plt.imshow(g, cmap=plt.cm.gray)
plt.title("Edge Magnitude")
plt.xticks([]), plt.yticks([])
plt.show()
cv2.imwrite('Step3-G.jpg',g)

#-------------------------------------------------------------------------------------------------

def Non_Maxima_Suppression(sobel):
    max_sup = g.copy()
    for x in range(sobel.shape[0]):
        for y in range(sobel.shape[1]):
            if (x>=5 and x<=max_sup.shape[0]-6) and (y>=5 and y<=max_sup.shape[1]-6):
            #checking the edge angles
            #loop range is because of pixel values of first 5 rows 5 columns and last 5 rows 5
columns will be undefined
                if (sobel[x][y]<22.5 and sobel[x][y]>=0) or \
                    (sobel[x][y]>=157.5 and sobel[x][y]<202.5) or \
                    (sobel[x][y]>=337.5 and sobel[x][y]<=360):
                    sobel[x][y]=0
                elif (sobel[x][y]>=22.5 and sobel[x][y]<67.5) or \
                    (sobel[x][y]>=202.5 and sobel[x][y]<247.5):
                    sobel[x][y]=45
```

Name- Yash Shirodkar
Net ID- yps224
N Number- N18264436

```
            elif (sobel[x][y]>=67.5 and sobel[x][y]<112.5)or \
                (sobel[x][y]>=247.5 and sobel[x][y]<292.5):
                sobel[x][y]=90
            else:
                sobel[x][y]=135
        else:
            sobel[x,y]=0

    for x in range(max_sup.shape[0]):
        for y in range(max_sup.shape[1]):
            if (x>=5 and x<=max_sup.shape[0]-6) and (y>=5 and y<=max_sup.shape[1]-6):
            #finally applyting non-maxima suppression
                if sobel[x][y]==0:
                    if (g[x][y]<=g[x][y+1]) or \
                        (g[x][y]<=g[x][y-1]):
                        max_sup[x][y]=0
                elif sobel[x][y]==45:
                    if (g[x][y]<=g[x-1][y+1]) or \
                        (g[x][y]<=g[x+1][y-1]):
                        max_sup[x][y]=0
                elif sobel[x][y]==90:
                    if (g[x][y]<=g[x+1][y]) or \
                        (g[x][y]<=g[x-1][y]):
                        max_sup[x][y]=0
                else:
                    if (g[x][y]<=g[x+1][y+1]) or \
                        (g[x][y]<=g[x-1][y-1]):
                        max_sup[x][y]=0
            else:
                max_sup[x,y]=0
    return max_sup

#-----------------------------------------------------------------------------------------------

max_sup = Non_Maxima_Suppression(sobel)
plt.imshow(max_sup, cmap=plt.cm.gray)
plt.title('After Non Maxima Suppression')
plt.xticks([]), plt.yticks([])
plt.show()
cv2.imwrite('Step4-Non_Maxima_Suppression.jpg', max_sup)
```

Name- Yash Shirodkar
Net ID- yps224
N Number- N18264436

```
#-----------------------------------------------------------------------------------------------

def Double_Thresholding(NMS_Magnitude,gradient_angle,t1,t2):
    # Generating pixel count for each grayscale value
    directn = [(-1,-1),(0,-1),(-1,1),
               (-1,0),(0,1),
              (-1,1),(0,1) ,(1,1)]

    N = len(NMS_Magnitude)
    M = len(NMS_Magnitude[0])

    #Setting thresholds
    T1 = t1
    T2 = t2

    image1 = np.zeros((N, M))
    for i in range(0, N):
        for j in range(0, M):
            if (NMS_Magnitude[i][j] < T1):
                image1[i][j] = 0 #setting values less than t1 to 0
            elif (NMS_Magnitude[i][j] > T2):
                image1[i][j] = 255 #setting values greater than t2 to 255
            else:
                for v in directn:  #calculating if its 0 or 255 based on its neighbors
                    x = i + v[0]
                    y = j + v[1]

                    if x >=0 and y >=0 and x<N and y<M:
                        if NMS_Magnitude[x][y] > T2 and
abs(gradient_angle[x][y]-gradient_angle[i][j])<=45 :
                            image1[i][j] = 255
                            break
                        image1[x][y]=0

    return image1

#-----------------------------------------------------------------------------------------------

Double_Thresholding_Image = Double_Thresholding(max_sup,sobel,9,18)#selecting two
threshold values
plt.imshow(Double_Thresholding_Image, cmap=plt.cm.gray)
plt.title('After Double Threshold')
```

Name- Yash Shirodkar
Net ID- yps224
N Number- N18264436

```
plt.xticks([]), plt.yticks([])
plt.show()
cv2.imwrite("Step5-Double_thresholding.jpg", Double_Thresholding_Image)

#-------------------------------------------------------------------------------------------------------
```

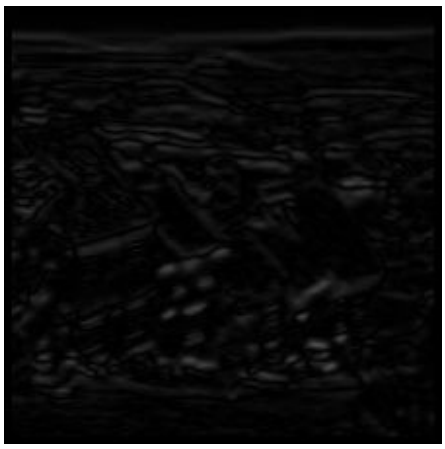
4. Output Images-

1. Output after Gaussian Smoothing-


2.Output after Horizontal Gradient


3. Output after Vertical Gradient

Name- Yash Shirodkar
Net ID- yps224
N Number- N18264436

4. Output after Gradient Magnitude


5. Output after Non Maxima Suppression


Name- Yash Shirodkar
Net ID- yps224
N Number- N18264436

6. Final Output after Double Thresholding