Assignment # 2

DIP

Name	**:**	**Arshad Habib**
Roll No	**:**	**17I-0208**
Section	**:**	**A**
Submitted To	**:**	**Dr. Akhtar Jamil**
Submission Date	**:**	**15-11-2021**

Question 1:

Solution:

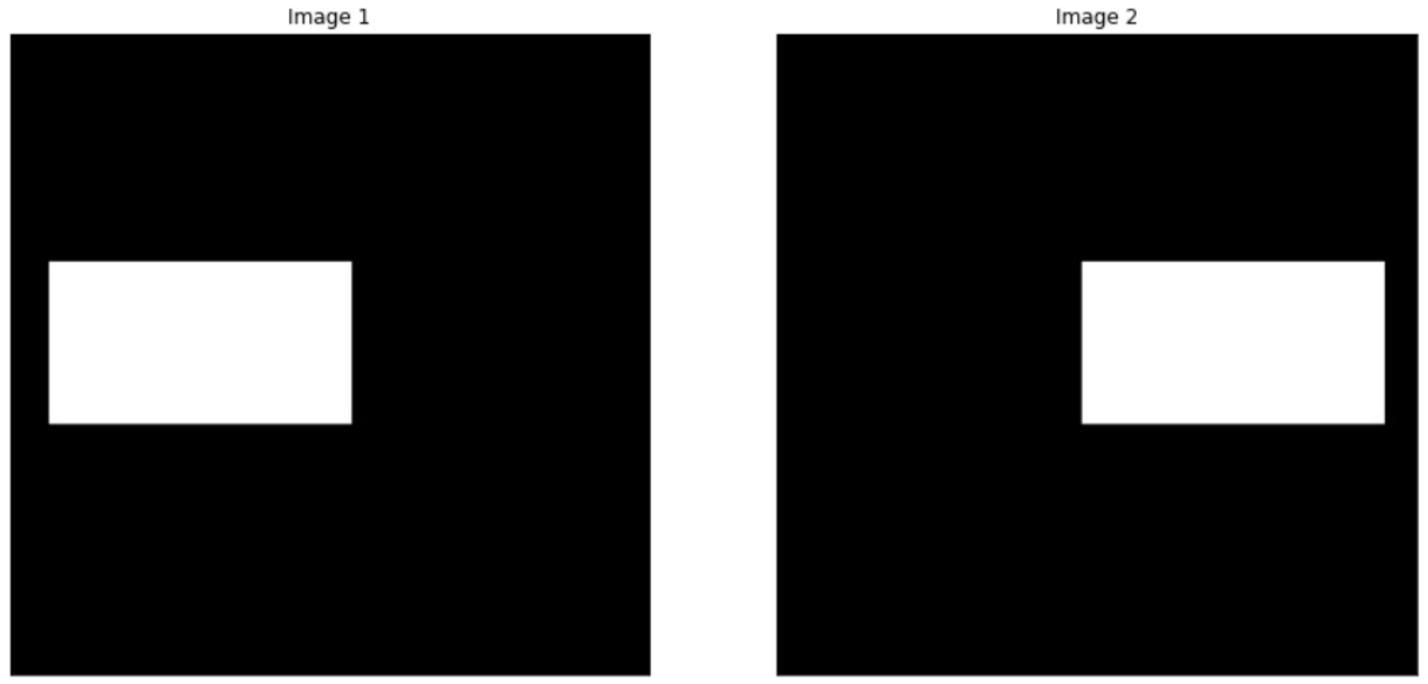
First of all importing numpy, math, shapely and matplotlib libraries as

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import math as m
from shapely.geometry import box #importing libraries
```

Reading image 1 and image 2 from path and printing both images using plt library

```
path = r'F:\Study\Fall 2021\DIP\Assignments\2\Assignment#2\img\Q-1a.jpg' #image 1 path
img1 = cv2.imread(path,cv2.IMREAD_COLOR) #reading image 1 by opencv
path = r'F:\Study\Fall 2021\DIP\Assignments\2\Assignment#2\img\Q-1b.jpg' #image 2 path
img2 = cv2.imread(path,cv2.IMREAD_COLOR) #reading image 2 by opencv
gimg1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY) #converting image 1 from BGR to Grayscale
gimg2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY) #converting image 2 from BGR to Grayscale
plt.figure(figsize=(15, 15))#showing both images using matplot library
plt.subplot(121),plt.imshow(gimg1,cmap='gray'),plt.title('Image 1')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(gimg2,cmap='gray'),plt.title('Image 2')
plt.xticks([]), plt.yticks([])
plt.show()
```

Output:

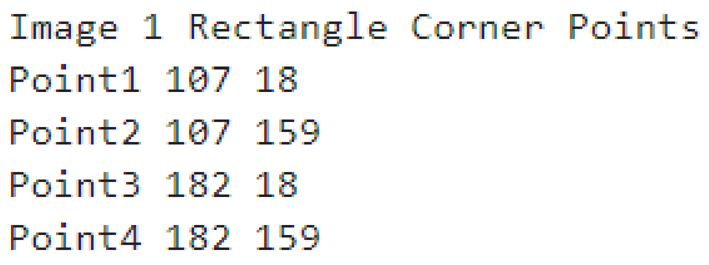


After that I am calculated image 1 rectangle corner points applying my own logic by traversing each pixel and saving corners points to the variables.

```
#calculating cordinates of rectangle object in image 1
i1px1=-1 #point 1 x
i1py1=-1 #point 1 y
i1px2=-1 #point 2 x
i1py2=-1 #point 2 y
i1px3=-1 #point 3 x
i1py3=-1 #point 3 y
i1px4=-1 #point 4x
i1py4=-1 #point 4 y
```

```
for i in range(len(gimg1)):
    for j in range(len(gimg1[i])): #nested loop for traversing image 1 pixels
        #print(gimg1[i][j])
        if(gimg1[i][j]>100): #thresh-holding
            if(i1px1==-1 and i1py1==-1): #checking if point 1 y and point 2 y are set or not
                i1px1=i #saving pixel to point 1 x
                i1py1=j #saving pixel to point 1 y
            if(gimg1[i][j+1]<100):  #thresh-holding  and checking end of rectangle columns
                if(i1px2==-1 and i1py2==-1): #checking if point 2 x and point 2 y are set or not
                    i1px2=i #saving pixel to point 2 x
                    i1py2=j #saving pixel to point 2 y
            if(gimg1[i+1][j]<100): #thresh-holding and checking then end of rectangle rows
                if(i1px3==-1 and i1py3==-1): #checking if point 3 x and point 3 y are set or not
                    i1px3=i #saving pixel to point 3 x
                    i1py3=j #saving pixel to point 3 y
i1px4=i1px3 #point 4 x is equal to point 3 x (becuase of same number of opposite rows)
i1py4=i1py2 #point 4 y is equal to point 2 y (because of same number of opposite columns)
print("Image 1 Rectangle Corner Points")
print("Point1",i1px1,i1py1) #printing rectangle co ordninates
print("Point2",i1px2,i1py2)
print("Point3",i1px3,i1py3)
print("Point4",i1px4,i1py4)
```

Output:



```
Image 1 Rectangle Corner Points
Point1 107 18
Point2 107 159
Point3 182 18
Point4 182 159
```

Them I applied the same technique of image 2 to get rectangle corner points.

```
#calculating cordinates of rectangle object in image 2
i2px1=-1 #point 1 x
i2py1=-1 #point 1 y
i2px2=-1 #point 2 x
i2py2=-1 #point 2 y
i2px3=-1 #point 3 x
i2py3=-1 #point 3 y
i2px4=-1 #point 4 x
i2py4=-1 #point 4 y
for i in range(len(gimg2)):
    for j in range(len(gimg2[i])): #nested loop for traversing image 2 pixels
        #print(gimg1[i][j])
        if(gimg2[i][j]>100): #thresh-holding
            if(i2px1==-1 and i2py1==-1): #checking if point 1 y and point 2 y are set or not
                i2px1=i #saving pixel to point 1 x
                i2py1=j #saving pixel to point 1 y
            if(gimg2[i][j+1]<100): #thresh-holding  and checking end of rectangle columns
                if(i2px2==-1 and i2py2==-1): #checking if point 2 x and point 2 y are set or not
                    i2px2=i #saving pixel to point 2 x
                    i2py2=j #saving pixel to point 2 y
            if(gimg2[i+1][j]<100): #thresh-holding and checking then end of rectangle rows
                if(i2px3==-1 and i2py3==-1):  #checking if point 3 x and point 3 y are set or not
                    i2px3=i #saving pixel to point 3 x
                    i2py3=j #saving pixel to point 3 y
i2px4=i2px3 #point 4 x is equal to point 3 x (becuase of same number of opposite rows)
i2py4=i2py2 #point 4 y is equal to point 2 y (because of same number of opposite columns)
print("Image 2 Rectangle Corner Points")
print("Point1",i2px1,i2py1) #printing rectangle co ordninates
print("Point2",i2px2,i2py2)
print("Point3",i2px3,i2py3)
print("Point4",i2px4,i2py4)
```

Output:

```
Image 2 Rectangle Corner Points
Point1 107 143
Point2 107 284
Point3 182 143
Point4 182 284
```

Now I am calling the shapely function box to draw 2 rectangles by using the corner points which I got from above code. Then I am using the shapely intersection function to calculate overlapping area between the rectangles in image 1 and 2.

```
#now using the above cordinates, calculating overlapping area
b1 = box(i1py1,i1px1,i1py4,i1px4) #image 1 rectangle points
b2 = box(i2py1,i2px1,i2py4,i2px4) #image 2 rectangle points
intersection=b1.intersection(b2) #using shapely libraray to find the overlapping area between 2 rectangles
print("Overlapping Area in Pixels : ",int(intersection.area))
```

Final Output:

```
Overlapping Area in Pixels :  1200
```

Question 2:

Solution:

Reading image.

```
path3 = r'F:\Study\Fall 2021\DIP\Assignments\2\Assignment#2\img\Q-2.jpg' #Q 2 image path
img3 = cv2.imread(path3,cv2.IMREAD_COLOR)
imgGray = cv2.cvtColor(img3,cv2.COLOR_BGR2GRAY)
img3 = cv2.cvtColor(img3,cv2.COLOR_BGR2RGB)
plt.imshow(img3)
plt.axis('off')
plt.show()
```

Output:

Now displaying the gray scale format of above image.

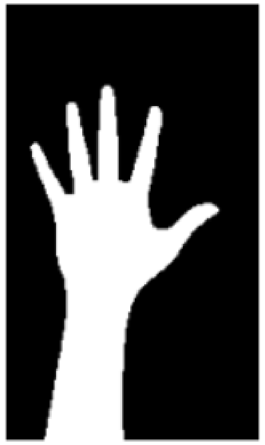
```
plt.imshow(imgGray,cmap='gray')
plt.axis('off')
plt.show()
```

Output:

Now First I converted gray image to binary then make a structuring element of size 3x3. Then I took the erosion of binary image using openCV erosion function the subtracted the eroded image from binary image. Resulted image is the boundary of hand. Then using openCV drawcircle function, I made the boundary around hand in original image. For calculating hand parameter I counted the white pixels in binary image and showed the result in the centre of hand in original image.

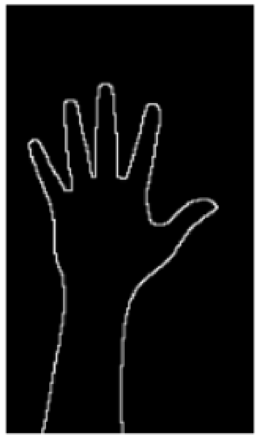
```
ret, imgthreshQ2 = cv2.threshold(imgGray,230,255,cv2.THRESH_BINARY) #applying threshholding on thresh hold value 230
img6_negQ2 = cv2.bitwise_not(imgthreshQ2) #taking negative of thresh holded image
kernel = np.ones((3,3), np.uint8) #making structurcting element of size 3x3
img_erosion = cv2.erode(img6_negQ2, kernel, iterations=1) #doing errosion
plt.imshow(img_erosion,cmap='gray') #showing the erroded image
plt.axis('off')
plt.show()
```

Output:



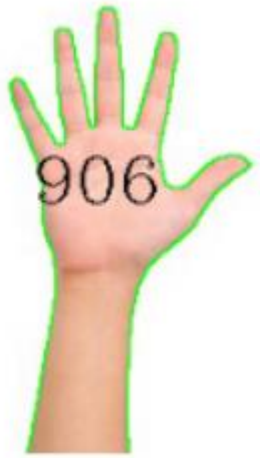
```
final_image=img6_negQ2-img_erosion #subtracting erroded image from original binary image
plt.imshow(final_image,cmap='gray') #showing the resulted image after subtraction
plt.axis('off')
plt.show()
```

Output:



```
pixel_count=0; #variable to store parimeter of hand
for i in range(len(final_image)):
    for j in range(len(final_image[i])): #nested loop
        if(final_image[i][j]==255): #if white pixel detected count it
            pixel_count=pixel_count+1 #incrementing count
for i in range(len(final_image)):
    for j in range(len(final_image[i])): #nested loop
        if(final_image[i][j]==255): #if white pixel detected
            cv2.circle(img3,(j,i),radius=0,color=(0,255,0),thickness=-1) #drawing boundary around the hand in
original image
pixel_count=str(pixel_count) #converting integer to string
cv2.putText(img3,pixel_count,((img3.shape[1]//2)-50,img3.shape[0]//2+10),cv2.FONT_HERSHEY_COMPLEX,1,(0,0,0),1)
#shoeing perimter using opencv in original image
plt.imshow(img3) #shwoing the resulted image
plt.axis('off')
plt.show()
```

Final Output:



Question 3:

Solution:

Reading image, converting to gray scale and printing it.

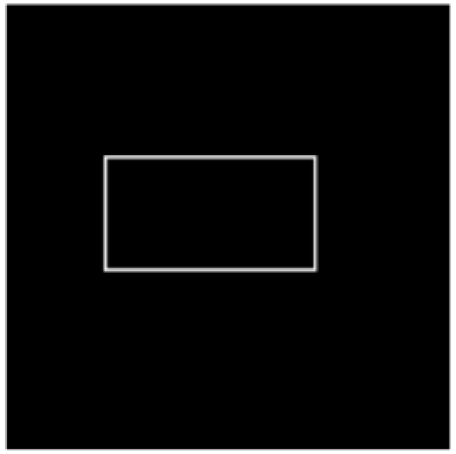
```
path5 = r'F:\Study\Fall 2021\DIP\Assignments\2\Assignment#2\img\Q-3.jpg' #Q 3 image path
img5 = cv2.imread(path5,cv2.IMREAD_COLOR) #reading image
imgGray5 = cv2.cvtColor(img5,cv2.COLOR_BGR2GRAY) #converting image to Grayscale
img5 = cv2.cvtColor(img5,cv2.COLOR_BGR2RGB) #converting image to RGB from BGR
plt.imshow(img5) #showing the original image in RGB
plt.axis('off')
plt.show()
```

Output:

Now first thresh holding the image on thresh hold value 100 by calling openCV threshold function and getting a binary image. After that applying Sobel X and Sobel Y operator on that binary image and detecting vertical and horizontal edges respectively and saving them into new images. Now adding the new images to a new image and displaying it which have all the edges.

```
ret, img5thresh = cv2.threshold(imgGray5, 100,255,cv2.THRESH_BINARY)#applying threshholding on thresh hold value 100
plt.imshow(img5thresh,cmap='gray')
#applying sobel operators on the thresh holded image to find edges
#soble operator Y
sobelY_64 = cv2.Sobel(img5thresh,cv2.CV_64F,1,0,ksize=3)
absY_64 = np.absolute(sobelY_64)
#soble operator Y
sobelX_64 = cv2.Sobel(img5thresh,cv2.CV_64F,0,1,ksize=3)
absX_64 = np.absolute(sobelX_64)
#now combining both sobel operatos
bothedges=np.zeros((img5thresh.shape[0],img5thresh.shape[1]),np.uint8)
for i in range(len(bothedges)):
    for j in range(len(bothedges[i])): #nested loops
        bothedges[i][j]=absX_64[i][j]+absY_64[i][j] #formula to convert RGB image to Grayscale Image
        if(bothedges[i][j]>255): #because both sobel operators detect corners
            bothedges[i][j]=255
plt.imshow(bothedges,cmap='gray') #showing the result image after sobel operator
plt.axis('off')
plt.show()
```

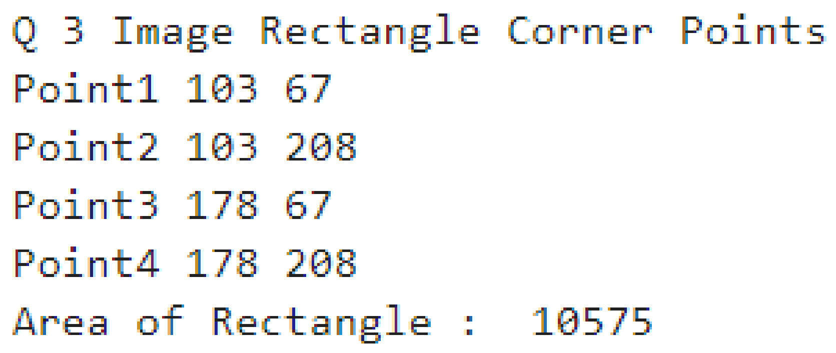
Output:



Now applying the Question 1 solution technique to get corner points of the rectangle and then using these corner points calculating the height and width and then calculating the area of rectangle.

```
#calculating cordinates of rectangle object in Q 3 image using thresh holded image
i5px1=-1 #point 1 x
i5py1=-1 #point 1 y
i5px2=-1 #point 2 x
i5py2=-1 #point 2 y
i5px3=-1 #point 3 x
i5py3=-1 #point 3 y
i5px4=-1 #point 4x
i5py4=-1 #point 4 y
for i in range(len(img5thresh)):
    for j in range(len(img5thresh[i])): #nested loop for traversing image 1 pixels
        #print(gimg1[i][j])
        if(img5thresh[i][j]==255): #thresh-holding
            if(i5px1==-1 and i5py1==-1): #checking if point 1 y and point 2 y are set or not
                i5px1=i #saving pixel to point 1 x
                i5py1=j #saving pixel to point 1 y
            if(img5thresh[i][j+1]==0):  #thresh-holding  and checking end of rectangle columns
                if(i5px2==-1 and i5py2==-1): #checking if point 2 x and point 2 y are set or not
                    i5px2=i #saving pixel to point 2 x
                    i5py2=j #saving pixel to point 2 y
            if(img5thresh[i+1][j]==0): #thresh-holding and checking then end of rectangle rows
                if(i5px3==-1 and i5py3==-1): #checking if point 3 x and point 3 y are set or not
                    i5px3=i #saving pixel to point 3 x
                    i5py3=j #saving pixel to point 3 y
i5px4=i5px3 #point 4 x is equal to point 3 x (becuase of same number of opposite rows)
i5py4=i5py2 #point 4 y is equal to point 2 y (because of same number of opposite columns)
#printing corner points and calculating area
print("Q 3 Image Rectangle Corner Points")
print("Point1",i5px1,i5py1) #printing rectangle co ordninates
print("Point2",i5px2,i5py2)
print("Point3",i5px3,i5py3)
print("Point4",i5px4,i5py4)
b5 = box(i5py1,i5px1,i5py4,i5px4) #Question 3 image rectangle points
print("Area of Rectangle : ",int(b5.area)) #prinitng Area
```

Final Output:



```
Q 3 Image Rectangle Corner Points
Point1 103 67
Point2 103 208
Point3 178 67
Point4 178 208
Area of Rectangle :  10575
```

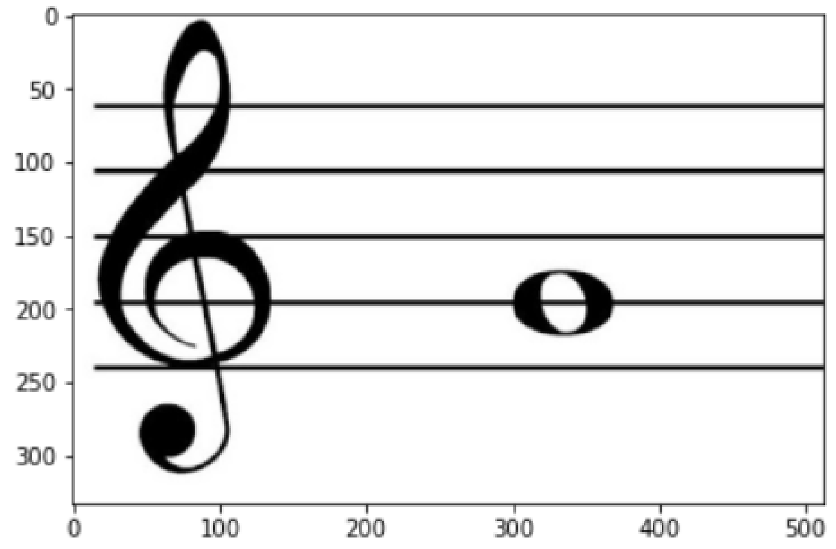
Question 4:

Solution:

Reading image, converting to gray scale and printing it.

```
path6 = r'F:\Study\Fall 2021\DIP\Assignments\2\Assignment#2\img\Q-4.jpg' #Q 4 image path
img6 = cv2.imread(path6,cv2.IMREAD_COLOR) #reading image
imgGray6 = cv2.cvtColor(img6,cv2.COLOR_BGR2GRAY) #converting image to Grayscale
img6 = cv2.cvtColor(img6,cv2.COLOR_BGR2RGB) #converting image to RGB from BGR
plt.imshow(img6) #showing the original image in RGB
plt.show()
```

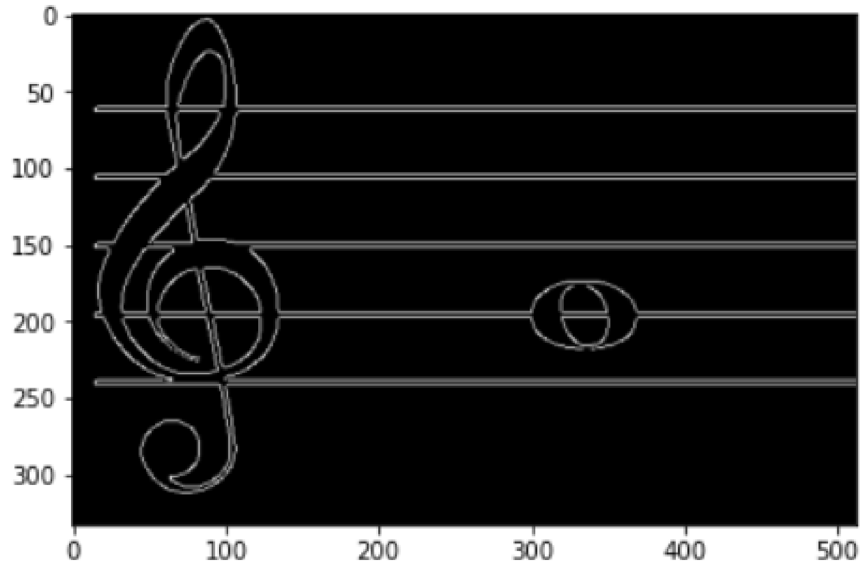
Output:



First of all I converted original image to gray scale. Then to Binary image. Then I used Canny Edge Detection. Then I used Hough Tranformation to detect only horizontal lines, Then through dilation , I filled the gap between horizontal lines, then I used erosion to reduce thickness.

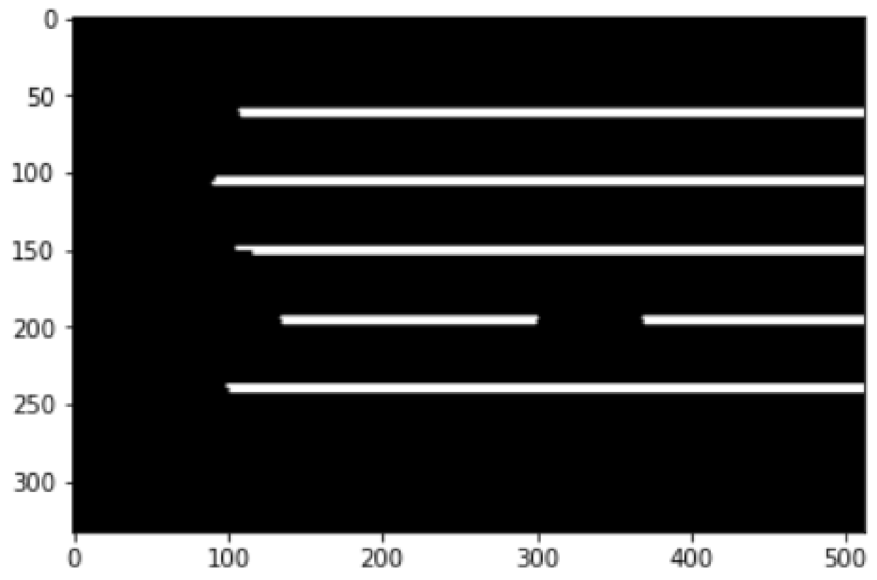
```
plt.imshow(imgGray6,cmap='gray') #showing the grayscale image
ret, img6thresh = cv2.threshold(imgGray6, 100, 255, cv2.THRESH_BINARY) #applying threshholding on thresh hold value
100
plt.imshow(img6thresh,cmap='gray') #showing the thresh holded image
imgCanny6=cv2.Canny(img6thresh,50,50) #applying canny edge detector
plt.imshow(imgCanny6,cmap='gray') #showing the thresh holded image
#plt.axis('off')
plt.show()
```

<u>Output:</u>



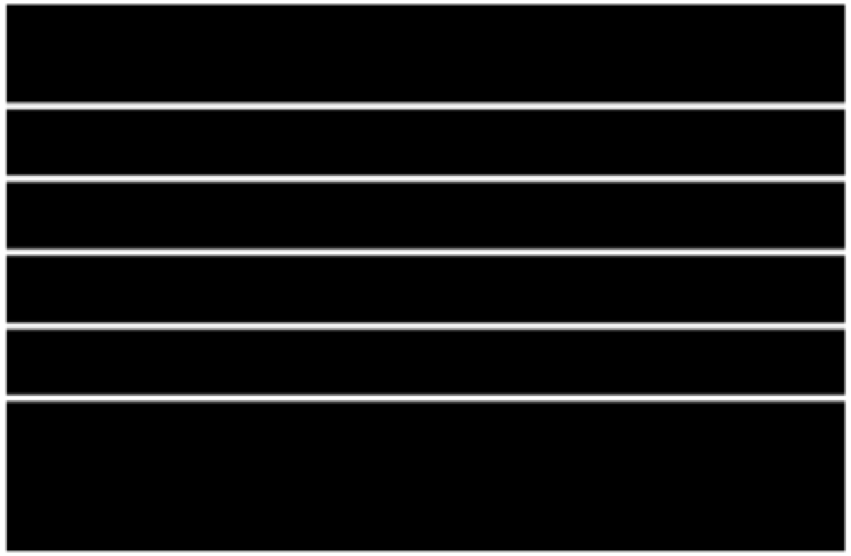
```
newimg6=np.zeros((img6thresh.shape[0],img6thresh.shape[1]),np.uint8)#making a new image of same size of original
image
lines = cv2.HoughLinesP(imgCanny6, 1,m.pi/2,2, None, 30, 1) #applying Hough Transformation on the canny image to
find horizonrtal lines, angle min=0 to angle max=90
i =0 #variable for loop iteration
while (i<len(lines)): #taking a sub array from the 3D original array
    for line in lines[i]: #getting line points
        pt1 = (line[0],line[1]) #line starting points
        pt2 = (line[2],line[3]) #line ending points
        cv2.line(newimg6, pt1, pt2, (255,255,255), 2) #using openCV line function to draw lines
    i=i+1
plt.imshow(newimg6,cmap='gray') #showing the resulted image
#plt.axis('off')
plt.show()
```

<u>Output:</u>



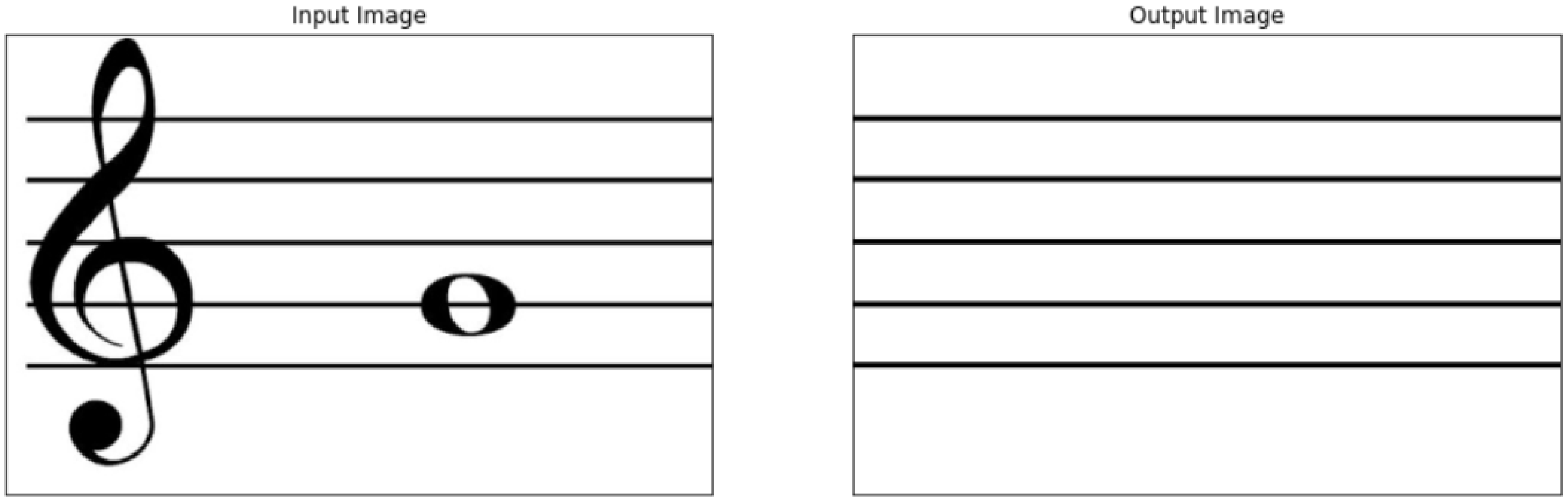
```
kernel = np.ones((1,280), np.uint8) #structuring element of size 3*3
img_erosion = cv2.dilate(newimg6, kernel, iterations=1) #doing dilation to fill edges gap
img_erosion = cv2.erode(img_erosion, kernel, iterations=1) #doing errosion to make lines thin
img_negQ4= cv2.bitwise_not(img_erosion) #taking negative of erroded image
plt.imshow(img_erosion,cmap='gray') #showing the thresh holded image
plt.axis('off')
plt.show()
```

Output:



```
plt.figure(figsize=(15, 15))#showing both images using matplot library
plt.subplot(121),plt.imshow(imgGray6,cmap='gray'),plt.title('Input Image')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_negQ4,cmap='gray'),plt.title('Output Image')
plt.xticks([]), plt.yticks([])
plt.show()
```

Final Output:


Input Image
Output Image