IPPR Lab 6

Name : Arya Shah

Roll No. E071

Class : BTech CSBS

Aim: To Apply Otsu Thresholding Technique To Segment The Objects And To Apply Canny Edge Detection Algorithm To Detect The Edges To Segment The Object

```
from skimage import io
import matplotlib.pyplot as plt
from skimage.color import rgb2gray
import numpy as np
from scipy import signal
from random import randint
import cv2
```

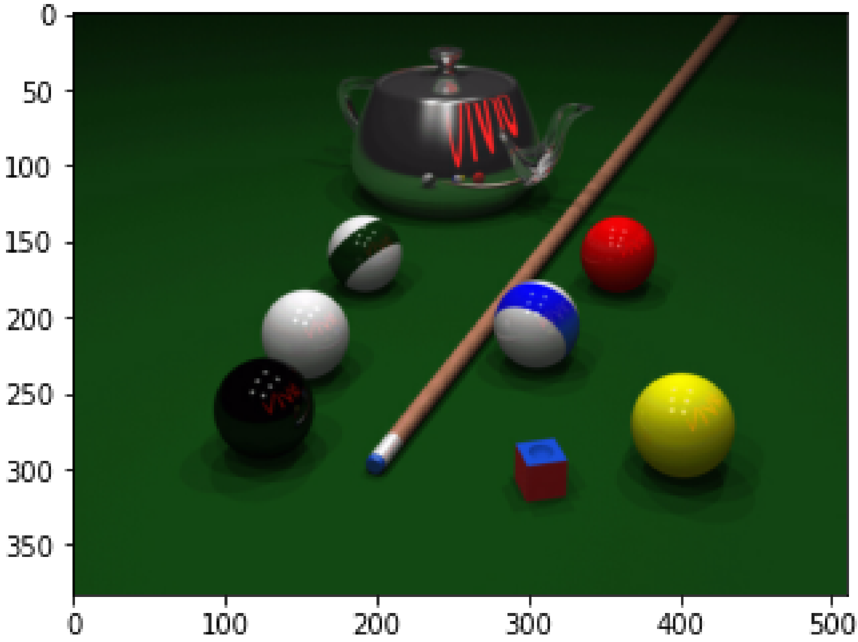
```
#Import Image
image=io.imread('pool.png')
```

```
image.shape
```

```
(383, 510, 3)
```

```
plt.imshow(image, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x209fb257c88>
```

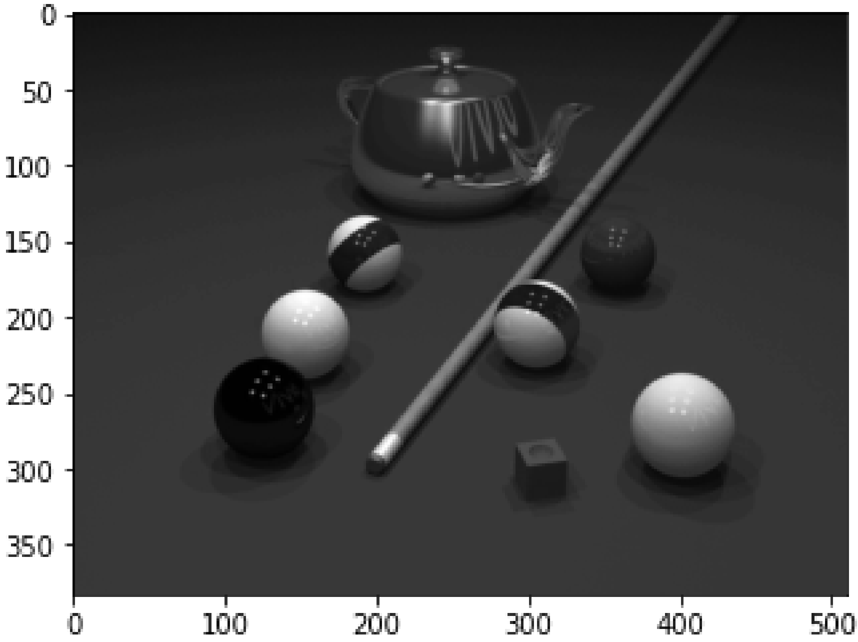


```
#Convert rgb to gray
image=rgb2gray(image)
```

```
image=image*255
```

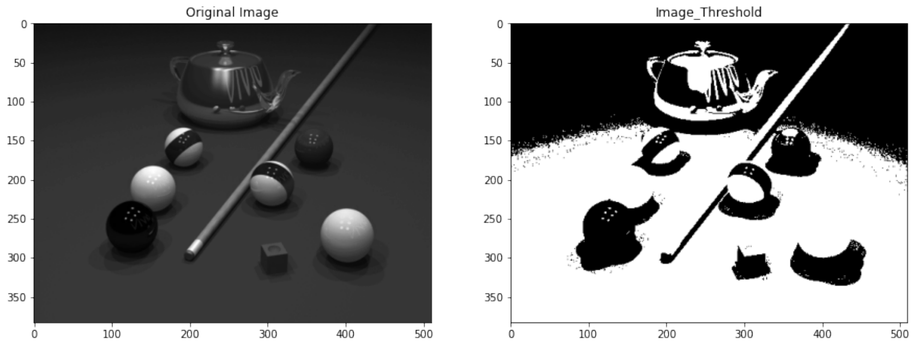
```
plt.imshow(image, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x209fb4b3c08>
```



```
# All pixels more than 150 become 255 and rest 0
# Given image made binary to segment the balls in the pool image

image_s=image.copy()

T=50

thresh,image_s = cv2.threshold(image,T,200,cv2.THRESH_BINARY)

#Display original and sobel image
plt.figure(figsize=(15,15))
plt.subplot(1,2,1)
plt.imshow(image, cmap = 'gray')
plt.title('Original Image')

plt.subplot(1,2,2)
plt.imshow(image_s, cmap = 'gray')
plt.title('Image_Threshold = 50')
```

```
Text(0.5, 1.0, 'Image_Threshold')
```



```
image_s[0:10,0:10]
```

```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

```
image_s[380:398,440:450]
```

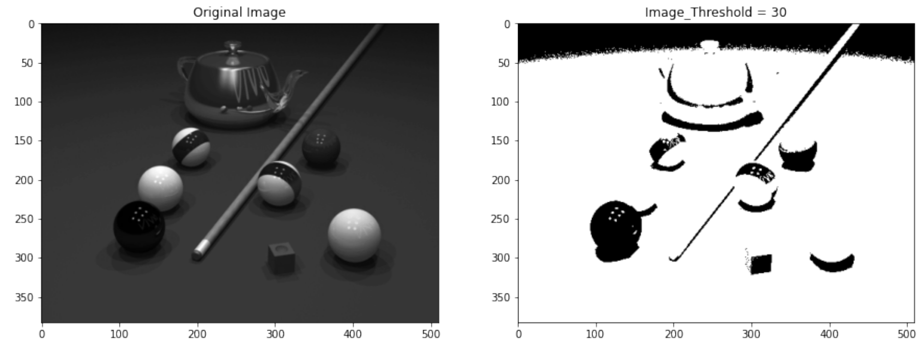
```
array([[200., 200., 200., 200., 200., 200., 200., 200., 200., 200.],
       [200., 200., 200., 200., 200., 200., 200., 200., 200., 200.],
       [200., 200., 200., 200., 200., 200., 200., 200., 200., 200.]])
```

```
# Increase/Decrease Threshold Value
T1 = 30
```

```
thresh,image_s = cv2.threshold(image,T1,200,cv2.THRESH_BINARY)
```

```
#Display original and sobel image
plt.figure(figsize=(15,15))
plt.subplot(1,2,1)
plt.imshow(image, cmap = 'gray')
plt.title('Original Image')

plt.subplot(1,2,2)
plt.imshow(image_s, cmap = 'gray')
plt.title('Image_Threshold = 30')
```

```
Text(0.5, 1.0, 'Image_Threshold = 30')
```



Conclusion

- For image threshold using threshold = 50 to segment the objects of the given image
- To segment more objects with little darker intensity than the original objects, threshold should be reduced further

(Initially threshold is 50, Reducing threshold to 30,)

```
from skimage import io
import matplotlib.pyplot as plt
from skimage.color import rgb2gray
import numpy as np
from scipy import signal
from random import randint
import cv2
```

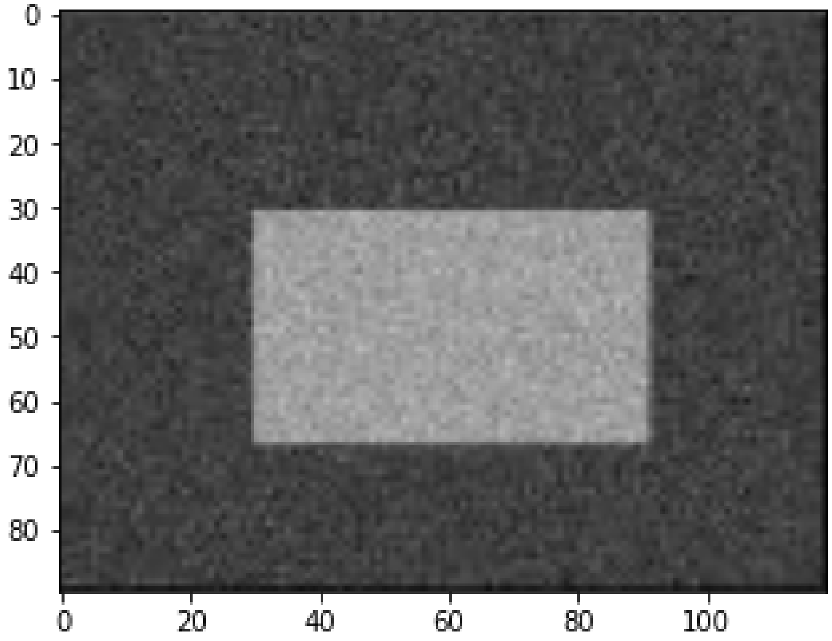
```
#Import Image
image=io.imread('noisy_image.png')
```

```
image.shape
```

```
(90, 119, 4)
```

```
plt.imshow(image, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x156c805de88>
```

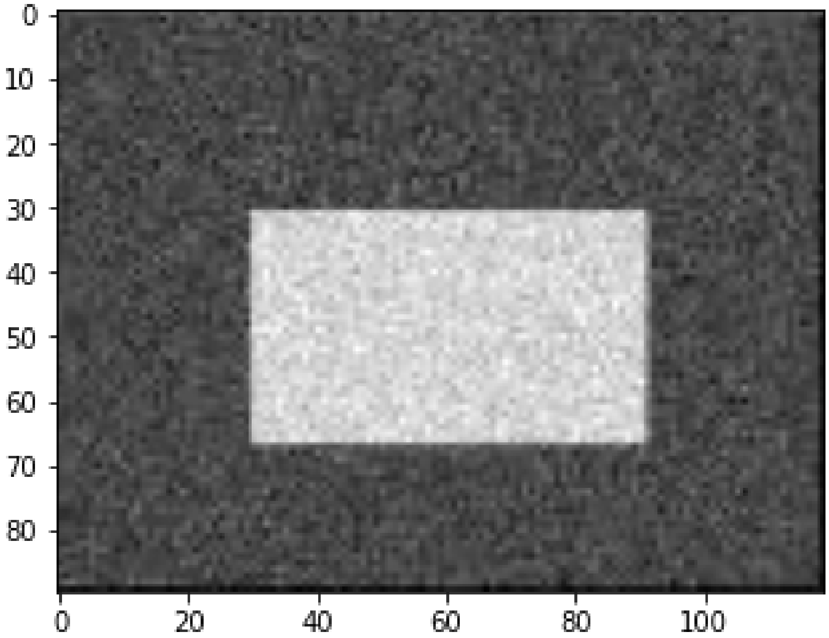


```
#Convert rgb to gray
from skimage.color import rgba2rgb
image = rgb2gray(rgba2rgb(image))
```

```
image=image*255
```

```
plt.imshow(image, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x156c8143c88>
```



```
sh = image.shape
```

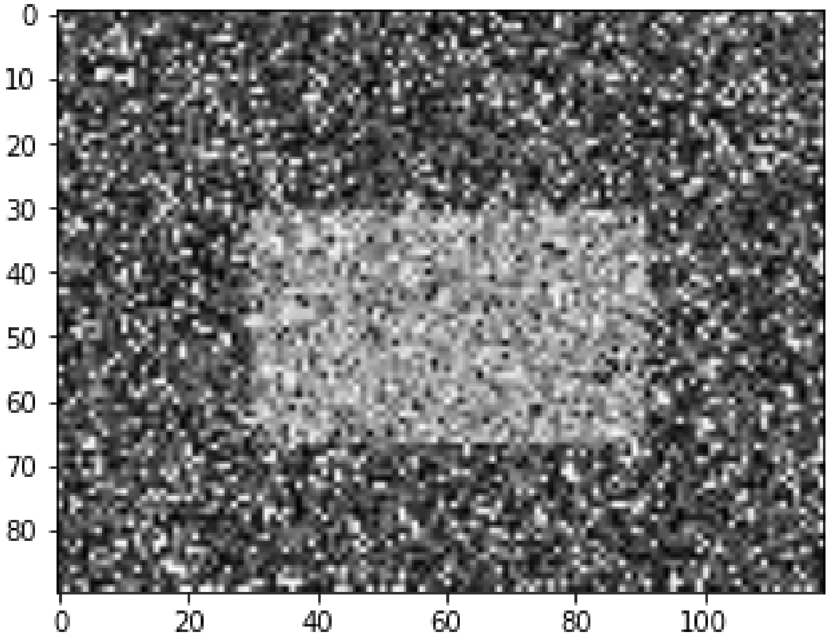
```
rows = sh[0]
cols = sh[1]
```

```
gn = np.random.normal(0,50,(rows,cols))
img1 = image+gn
```

```
img1 = np.uint8(img1)
```

```
plt.imshow(img1, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x156c81e1488>
```



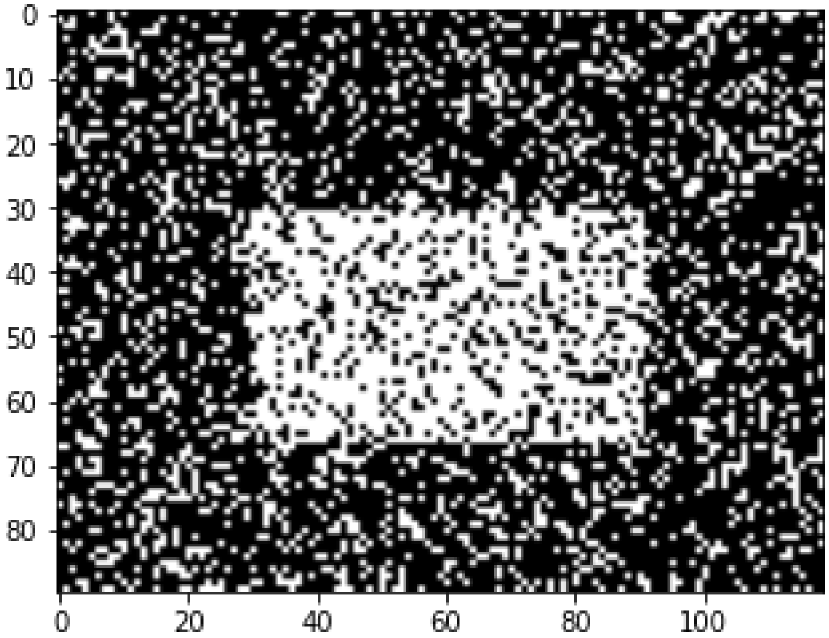
```
# Threshold and the Image it will be generating
th1,img1_th1 = cv2.threshold(img1,120,255,cv2.THRESH_BINARY)
```

```
th1
```

```
120.0
```

```
plt.imshow(img1_th1, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x156c826a608>
```



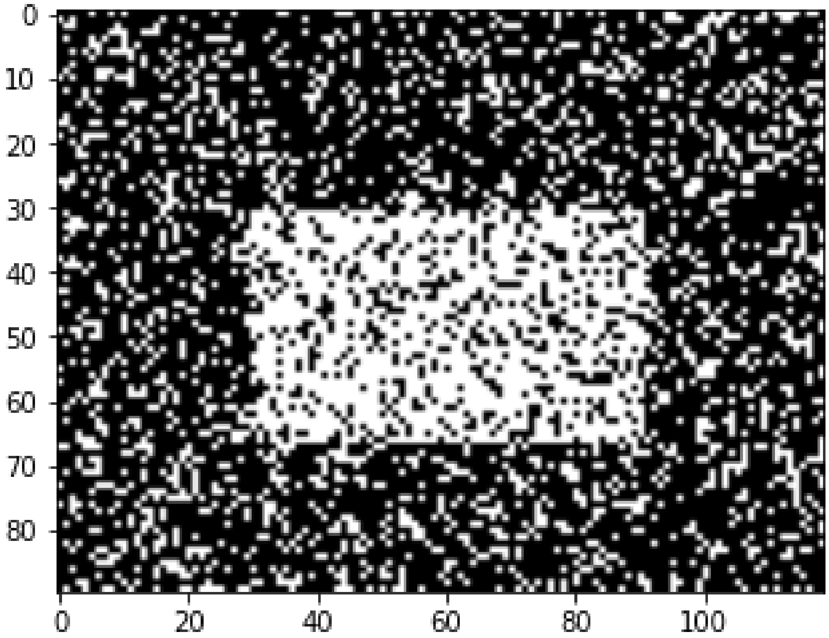
```
th2,img1_th2 = cv2.threshold(img1,100,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
```

```
th2
```

```
121.0
```

```
plt.imshow(img1_th2, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x156c82e7c08>
```



```
# Change 100 to 50
th2,img1_th2 = cv2.threshold(img1,50,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
```

```
th2 #  Gettiing the same threshold value
```

```
121.0
```

```
img_blur = cv2.GaussianBlur(img1,(5,5),0)
```

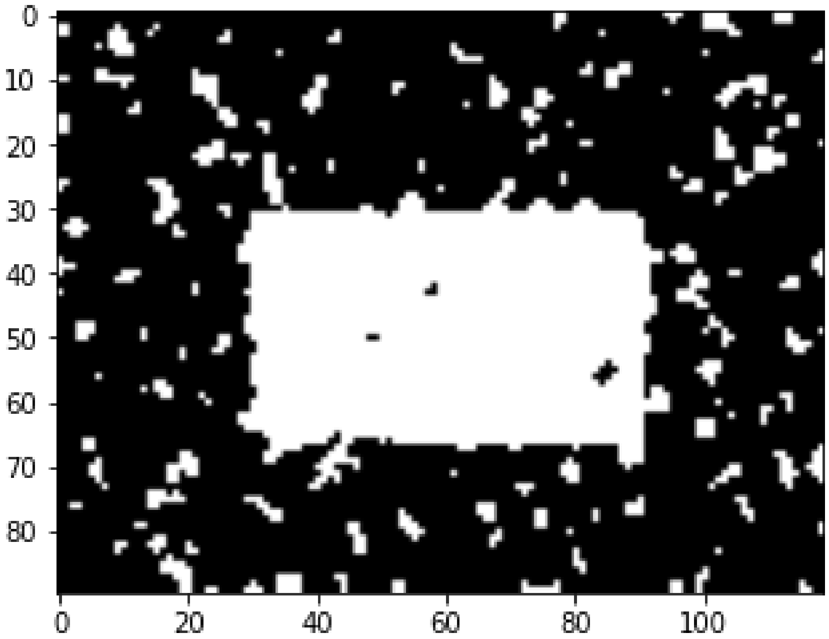
```
th3, img1_th3 = cv2.threshold(img_blur,100,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
```

```
th3
```

```
115.0
```

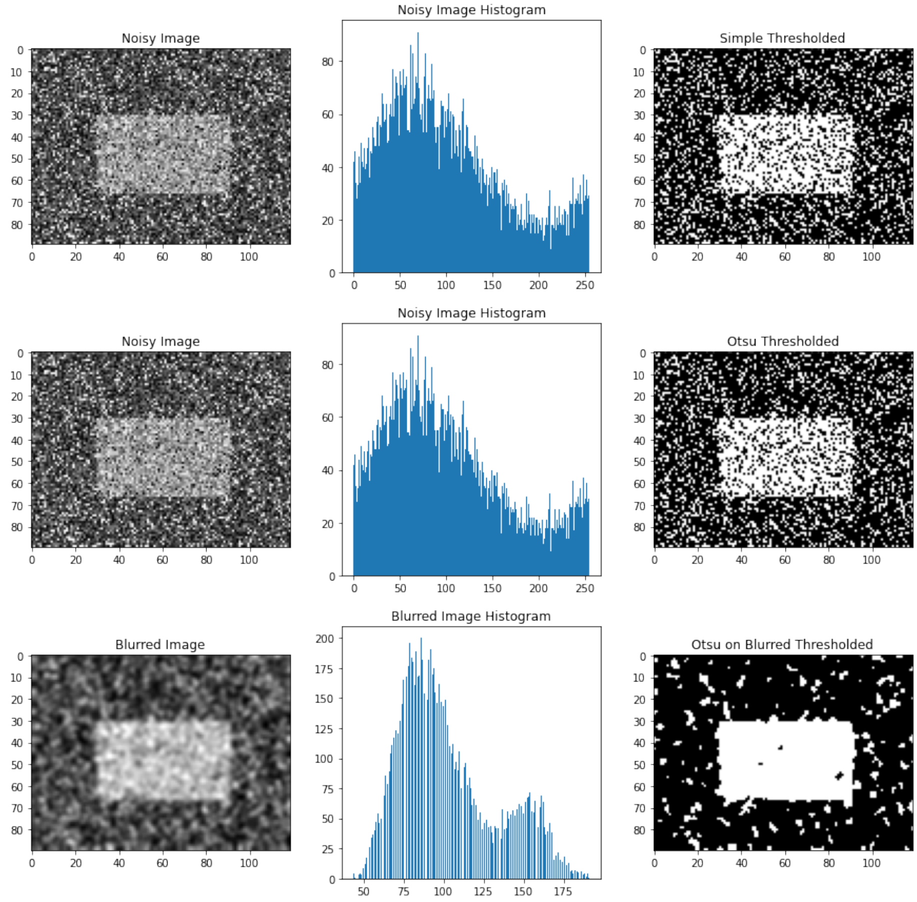
```
plt.imshow(img1_th3, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x156c836af48>
```



```
plt.figure(figsize=(15,15))
plt.subplot(3,3,1)
plt.imshow(img1, cmap = 'gray')
plt.title("Noisy Image")
plt.subplot(3,3,2)
plt.hist(img1.ravel(),256)
plt.title("Noisy Image Histogram")
plt.subplot(3,3,3)
plt.imshow(img1_th1, cmap = 'gray')
plt.title("Simple Thresholded")

plt.subplot(3,3,4)
plt.imshow(img1, cmap = 'gray')
plt.title("Noisy Image")
plt.subplot(3,3,5)
plt.hist(img1.ravel(),256)
plt.title("Noisy Image Histogram")
plt.subplot(3,3,6)
plt.imshow(img1_th2, cmap = 'gray')
plt.title("Otsu Thresholded")

plt.subplot(3,3,7)
plt.imshow(img_blur, cmap = 'gray')
plt.title("Blurred Image")
plt.subplot(3,3,8)
plt.hist(img_blur.ravel(),256)
plt.title("Blurred Image Histogram")
plt.subplot(3,3,9)
plt.imshow(img1_th3, cmap = 'gray')
plt.title("Otsu on Blurred Thresholded")
```

```
Text(0.5, 1.0, 'Otsu on Blurred Thresholded')
```



```
# Exam Q: Level of standard deviation till which the threshold is working
# Increased the gaussian noise std deviation
# Originally it was 20 made it 50
```

Conclusion

- If Image is corrupted with the noise with standard deviation of 20, the simple & Otsu method of thresholding show some noisy dots in the segmented

image.
- To reduce the effect of noise, the given image is blurred by using Gaussian Filter before applying Otsu's Thresholding
- It is observed that Otsu's method is effetive on the blurred image.
- If standard deviation of noise is increased to 50, Otsu method is not effective in segmenting the object

Canny Edge Detection

```
#Import Image
img=io.imread('pool.png')
```

```
img.shape
```

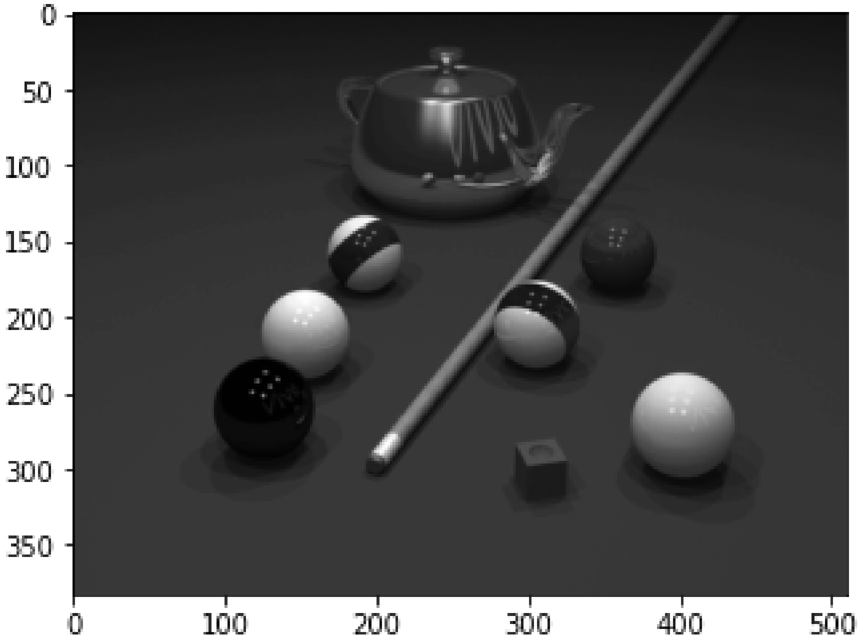
```
(383, 510, 3)
```

```
#Convert rgb to gray
img=rgb2gray(img)
```

```
img=img*255
```

```
plt.imshow(img, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x156c93f5ec8>
```



```
'''
If gradient more than threshold, they are all strong edge points
If gradient less than threshold, no edge point
In between is unsure
Canny edge decides which goes to which part of the threshold
'''
```

```
'\nIf gradient more than threshold, they are all strong edge points\nIf gradient less than t
```

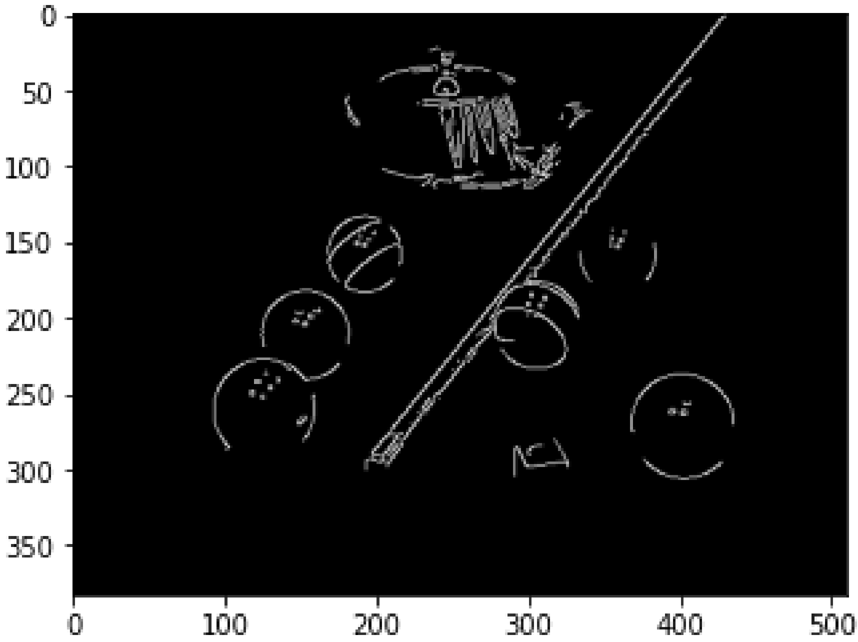
```
# Above 150 strong edge points, below 100 no edge points
th_l = 100
th_h = 150
```

```
img = np.uint8(img)
```

```
img_canny = cv2.Canny(img,th_l,th_h)
```

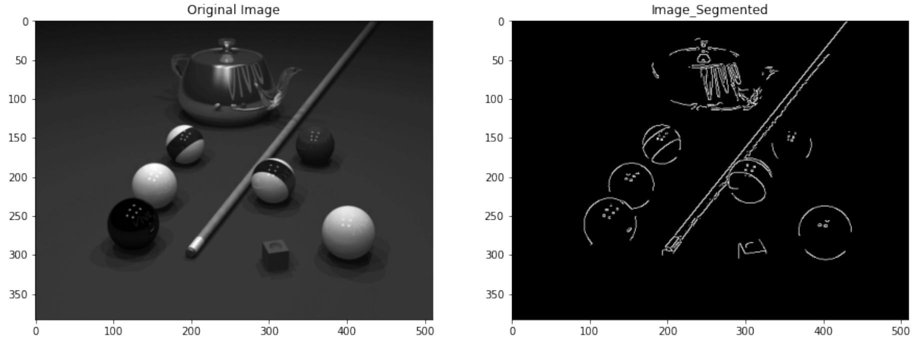
```
plt.imshow(img_canny, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x156c93c9c88>
```



```
#Display original and segmented image
plt.figure(figsize=(15,15))
plt.subplot(1,2,1)
plt.imshow(img, cmap = 'gray')
plt.title('Original Image')

plt.subplot(1,2,2)
plt.imshow(img_canny, cmap = 'gray')
plt.title('Image_Segmented')
```

```
Text(0.5, 1.0, 'Image_Segmented')
```

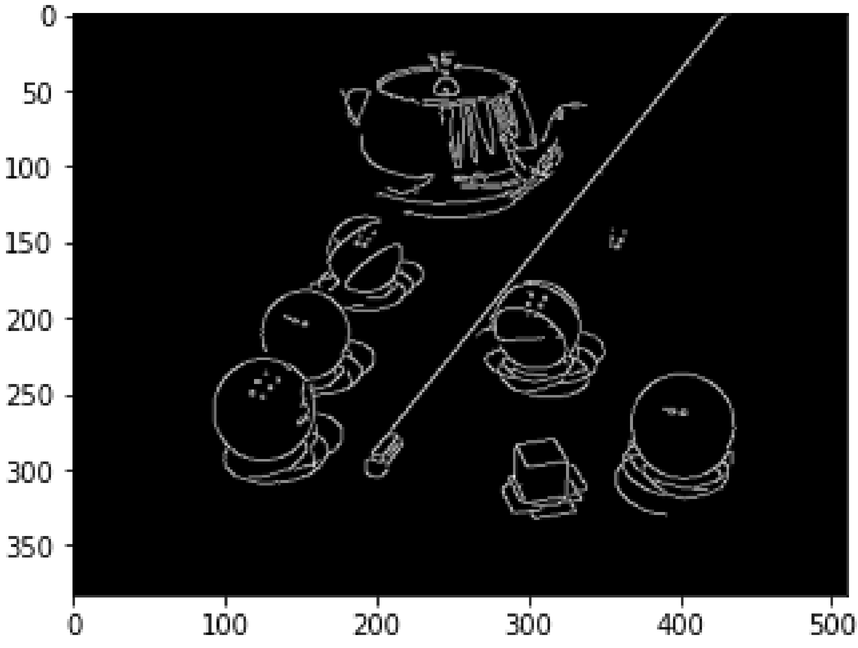


```
# Try Changing Threshold
th_l = 10
th_h = 250
```

```
img_canny = cv2.Canny(img,th_l,th_h)
```

```
plt.imshow(img_canny, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x156ca737988>
```

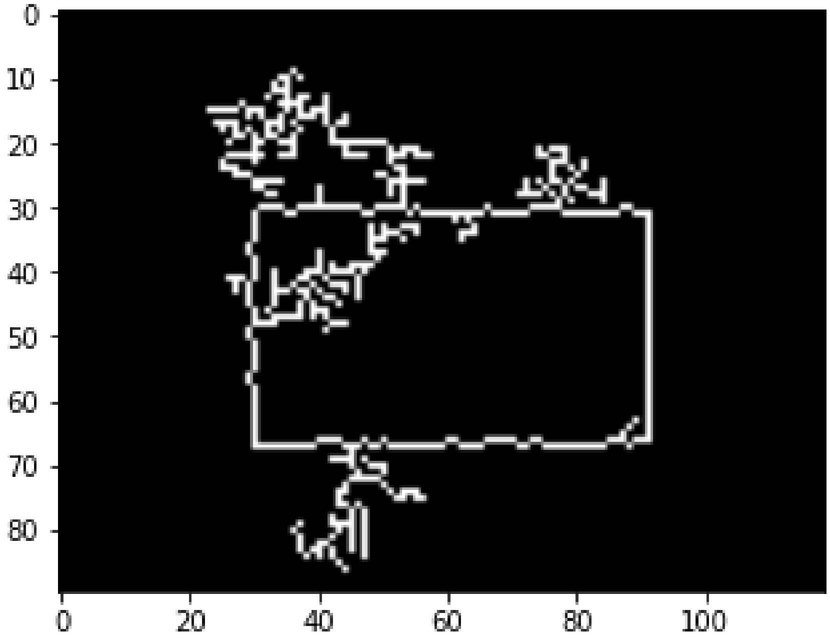


```
image = np.uint8(image)
```

```
img_canny1 = cv2.Canny(image,th_l,th_h)
```

```
plt.imshow(img_canny1, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x156ca878048>
```



Conclusion

- Cany Edge Detection is applied on the image with the hysteresis range of 100 to 150.
- It is observed that not all the objects are segmented using this range.
- If this range is increased to 50 to 200, more objects can be segmented