Sai Surya Salalith Mantha 9463553709

Mean Shift Segmentor

```
# Importing neccessary libraries
import numpy as np
import cv2
print(cv2.__version__)
```

```
3.3.1
```

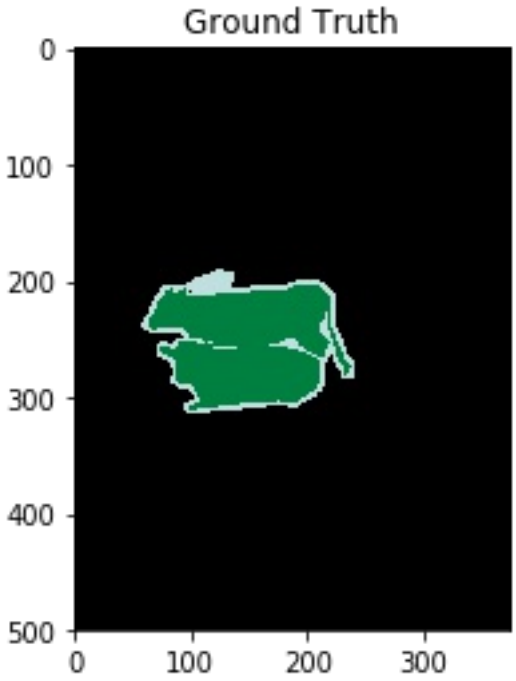
steps involved:

1. Read the input image and convert to the LAB Color space.
2. Apply the pyrMeanShiftFiltering() with varying spatial and color window radius with pyramid-1.
3. Converted the image back to RGV for better understanding and displayed the results.

```
# Reading Images

img1=cv2.imread('2007_000464.jpg')
img2=cv2.imread('2007_001288.jpg')
img3=cv2.imread('2007_002953.jpg')
img4=cv2.imread('2007_005989.jpg')

imgG1=cv2.imread('2007_000464.png')
imgG2=cv2.imread('2007_001288.png')
imgG3=cv2.imread('2007_002953.png')
imgG4=cv2.imread('2007_005989.png')



# convert the images to LAB color space

img1=cv2.cvtColor(img1,cv2.COLOR_BGR2LAB)
img2=cv2.cvtColor(img2,cv2.COLOR_BGR2LAB)
img3=cv2.cvtColor(img3,cv2.COLOR_BGR2LAB)
img4=cv2.cvtColor(img4,cv2.COLOR_BGR2LAB)
```

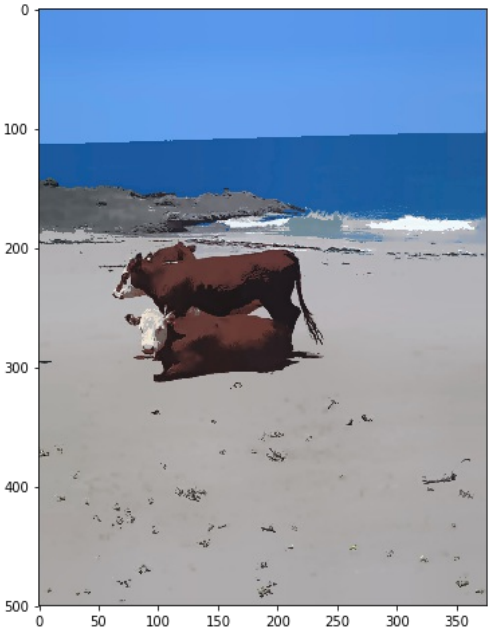
- using pyrMeanShiftFiltering(src,sp,sr,maxLevel)
- src----image
- sp-----spatial window radius.
- sr-----color window radius.
- maxLevel----Maximum level of the pyramid for the segmentation.

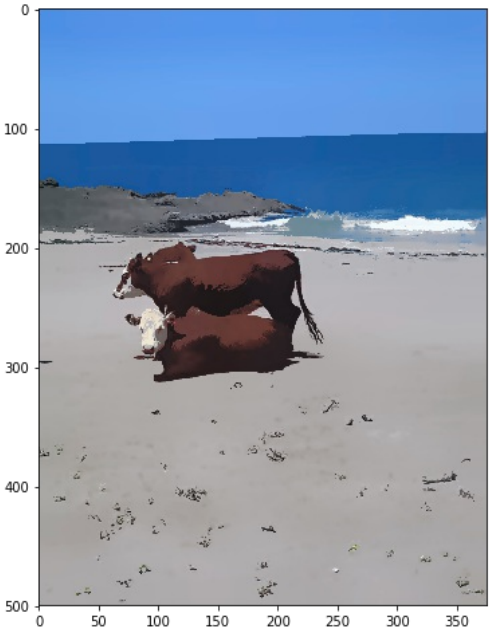
```
# Ground truth image

plt.imshow(imgG1)
plt.title("Ground Truth")
plt.show()
```

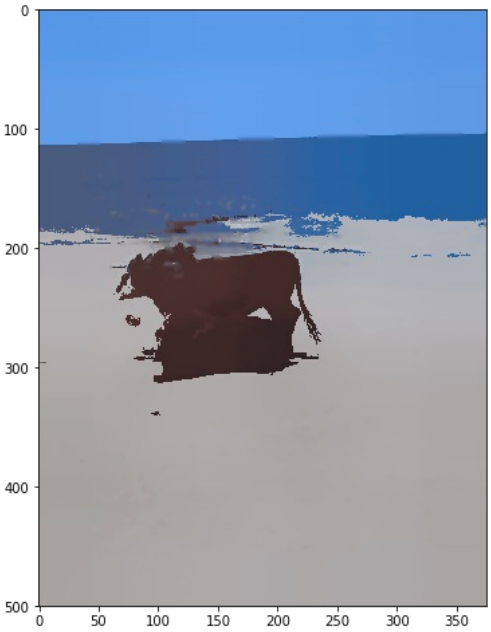


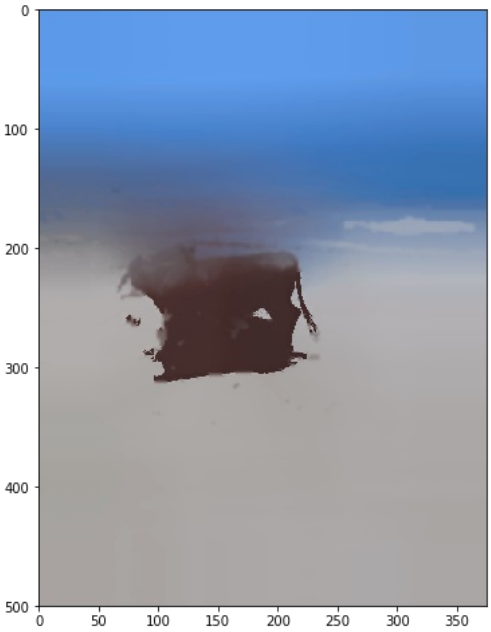

```
# Applying for various values for obesrving the changes

img_1_0=cv2.pyrMeanShiftFiltering(img1,30,20,1)
img_1_1=cv2.pyrMeanShiftFiltering(img1,20,20,1)
img_1_2=cv2.pyrMeanShiftFiltering(img1,40,50,1)
img_1_3=cv2.pyrMeanShiftFiltering(img1,50,100,1)


# Displaying the result using matplotlib

from matplotlib import pyplot as plt
f, axarr = plt.subplots(2,2,figsize=(18, 18))
axarr[0,0].imshow(cv2.cvtColor(img_1_0,cv2.COLOR_LAB2RGB))
axarr[0,1].imshow(cv2.cvtColor(img_1_1,cv2.COLOR_LAB2RGB))
axarr[1,0].imshow(cv2.cvtColor(img_1_2,cv2.COLOR_LAB2RGB))
axarr[1,1].imshow(cv2.cvtColor(img_1_3,cv2.COLOR_LAB2RGB))
```

```
plt.show()
```

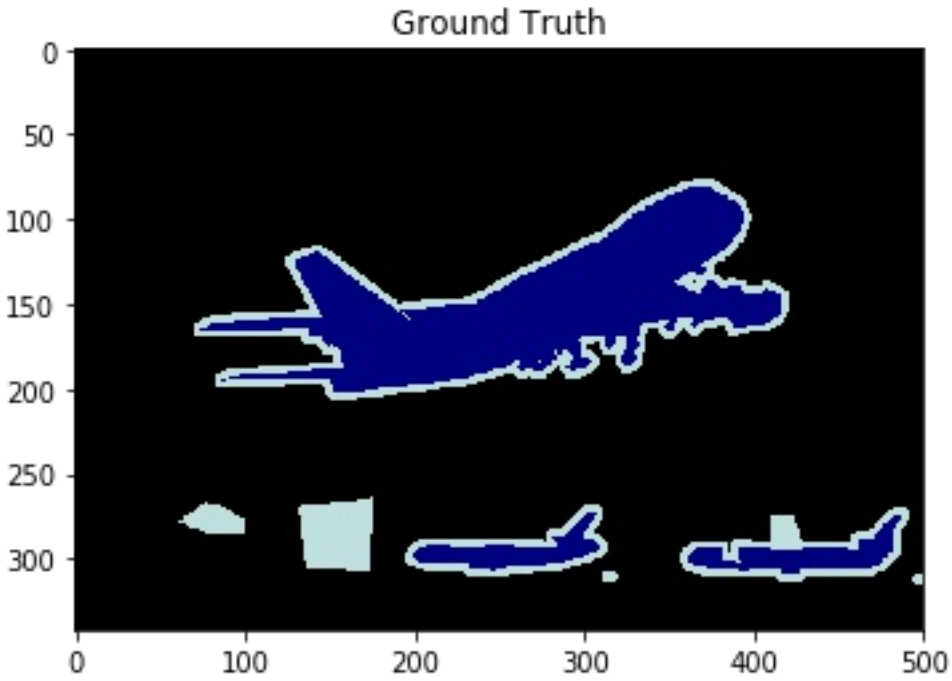






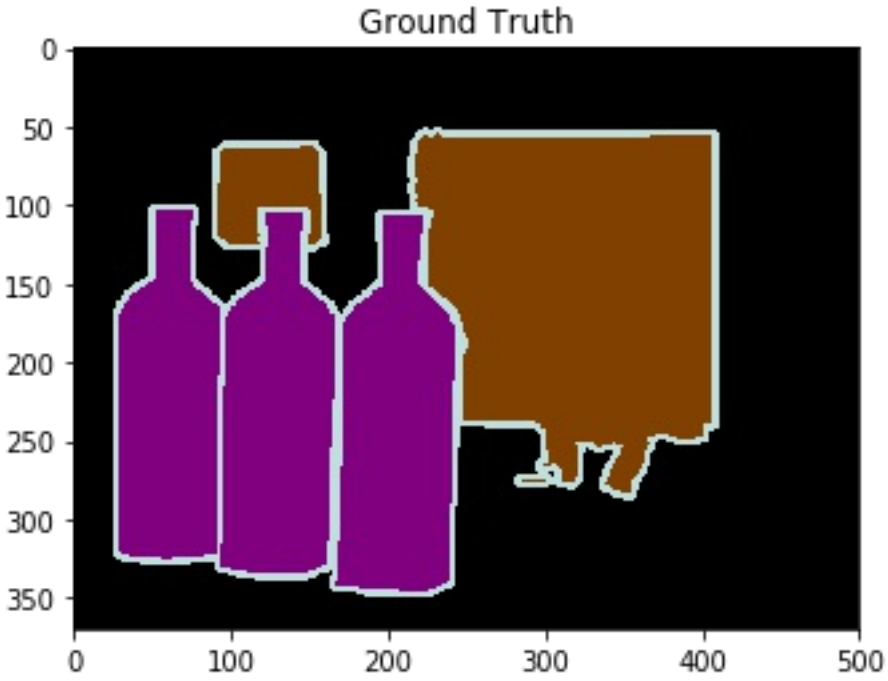



```
# Ground truth image

plt.imshow(imgG2)
plt.title("Ground Truth")
plt.show()
```



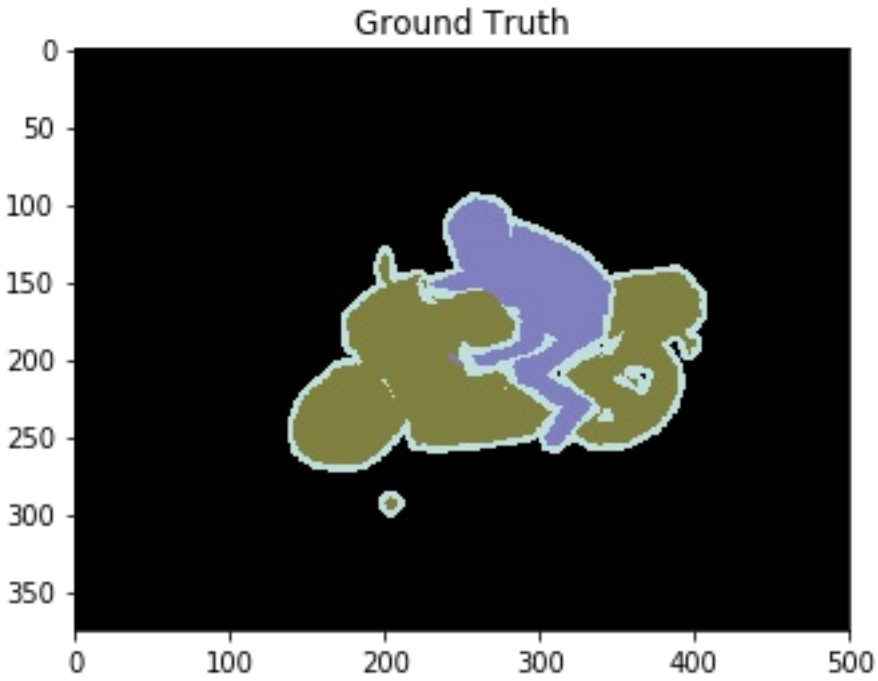

```
# Applying for various values for obesrving the changes

img_2_0=cv2.pyrMeanShiftFiltering(img2,30,20,1)
img_2_1=cv2.pyrMeanShiftFiltering(img2,20,20,1)
img_2_2=cv2.pyrMeanShiftFiltering(img2,30,30,1)
img_2_3=cv2.pyrMeanShiftFiltering(img2,50,50,1)


# Displaying the result using matplotlib

from matplotlib import pyplot as plt
f, axarr = plt.subplots(2,2,figsize=(18, 12))
axarr[0,0].imshow(cv2.cvtColor(img_2_0,cv2.COLOR_LAB2RGB))
axarr[0,1].imshow(cv2.cvtColor(img_2_1,cv2.COLOR_LAB2RGB))
axarr[1,0].imshow(cv2.cvtColor(img_2_2,cv2.COLOR_LAB2RGB))
axarr[1,1].imshow(cv2.cvtColor(img_2_3,cv2.COLOR_LAB2RGB))
plt.show()
```

```
# Ground truth image

plt.imshow(imgG3)
plt.title("Ground Truth")
plt.show()
```



```
# Applying for various values for obesrving the changes

img_3_0=cv2.pyrMeanShiftFiltering(img3,30,20,1)
img_3_1=cv2.pyrMeanShiftFiltering(img3,20,20,1)
img_3_2=cv2.pyrMeanShiftFiltering(img3,50,70,1)
img_3_3=cv2.pyrMeanShiftFiltering(img3,50,90,1)

# Displaying the result using matplotlib


from matplotlib import pyplot as plt
f, axarr = plt.subplots(2,2,figsize=(18, 12))
axarr[0,0].imshow(cv2.cvtColor(img_3_0,cv2.COLOR_LAB2RGB))
axarr[0,1].imshow(cv2.cvtColor(img_3_1,cv2.COLOR_LAB2RGB))
axarr[1,0].imshow(cv2.cvtColor(img_3_2,cv2.COLOR_LAB2RGB))
axarr[1,1].imshow(cv2.cvtColor(img_3_3,cv2.COLOR_LAB2RGB))
plt.show()
```

```
# Ground truth image

plt.imshow(imgG4)
plt.title("Ground Truth")
plt.show()
```




```
# Applying for various values for obesrving the changes

img_4_0=cv2.pyrMeanShiftFiltering(img4,30,20,1)
img_4_1=cv2.pyrMeanShiftFiltering(img4,20,30,1)
img_4_2=cv2.pyrMeanShiftFiltering(img4,50,50,1)
img_4_3=cv2.pyrMeanShiftFiltering(img4,50,70,1)


# Displaying the result using matplotlib

from matplotlib import pyplot as plt
f, axarr = plt.subplots(2,2,figsize=(18, 12))
axarr[0,0].imshow(cv2.cvtColor(img_4_0,cv2.COLOR_LAB2RGB))
axarr[0,1].imshow(cv2.cvtColor(img_4_1,cv2.COLOR_LAB2RGB))
axarr[1,0].imshow(cv2.cvtColor(img_4_2,cv2.COLOR_LAB2RGB))
axarr[1,1].imshow(cv2.cvtColor(img_4_3,cv2.COLOR_LAB2RGB))
plt.show()
```

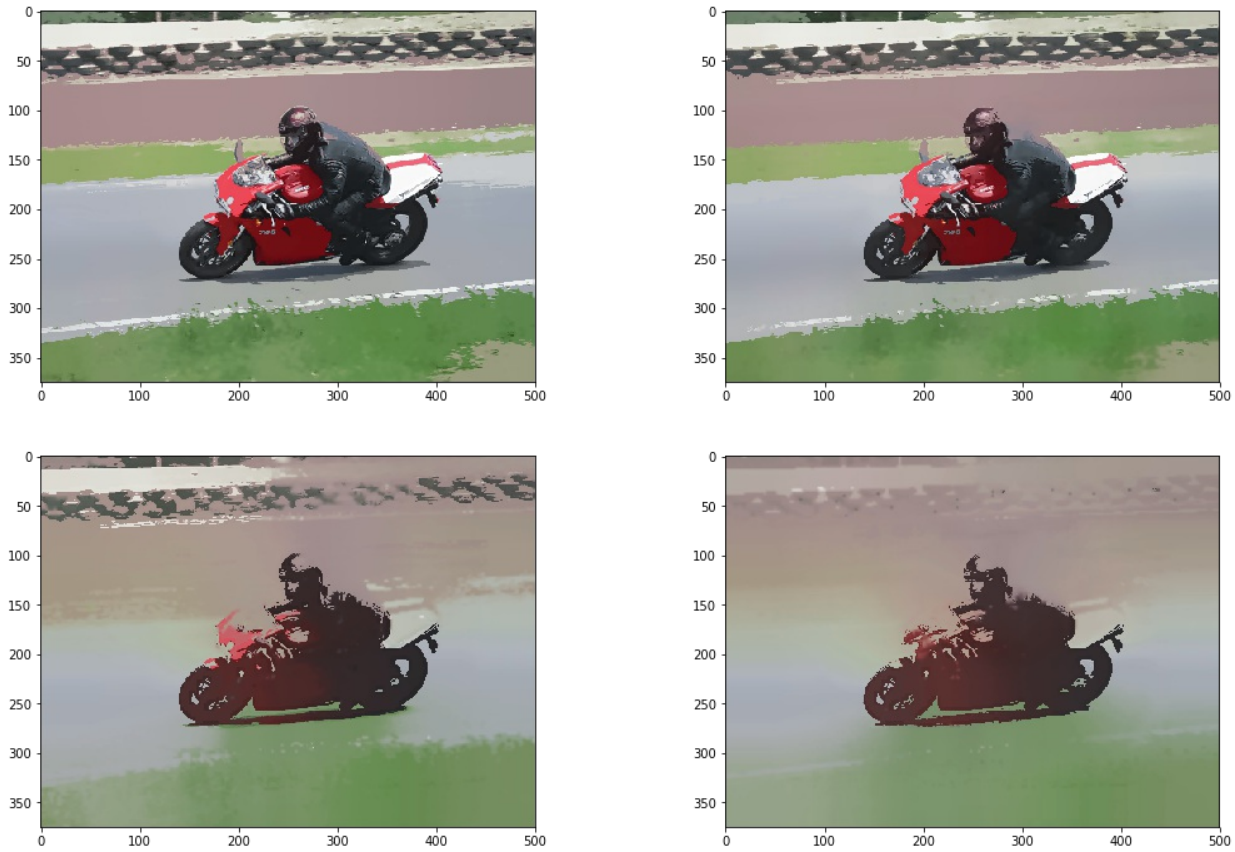


Image-1

- Segmentation ground truth for the above image is two cows.
- For the above image increasing the spatial radius is improving the results while if we increase the color radius initially it has improved the results but after certain values, the image got blurred.

Image-2

- Segmentation ground truth for the above image is three flights.
- For the above image increase in the parameters is making the segmentation bad and out of the images the second image in the first row is segmenting properly but in remaining images the flights on the ground were not segmented properly and got blurred.

Image-3

- ground truth for this image is bottle and screen.
- In this case, with the increase in parameters the results are improving. In the second row first image is segmenting the bottles and screens properly. In this, if we still increase the

parameters the regions around the bottle got blurred while the screen was still improved.

Image-4

- Ground truth is bike along with person.
- For this image increase in color window radius has more effect than the increase in the spatial window radius. The results got improved in the second image while the fourth image got blurred.

Analysis of Algorithm:

- The Algorithm is robust to outliers and does not assume spherical clusters, but the problem is the output depends entirely on window size which is hyperparameter which we need to tune and also the algorithm is computationally expensive and it increases with the size of the image.

- The parameters spatial window radius, Color window radius, and pyramid level were purely hyperparameters which we need to fine tune to get the better results.

Selective Search

```
%reset
import numpy
import cv2
from matplotlib import pyplot as plt
from matplotlib.pyplot import figure
print(cv2.__version__)
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
3.3.1
```

steps involved:

1. Read the images in RGB color space.
2. Get the ground truth Bounding Box values.
3. Apply SelectiveSearchSegmentation for the image using various strategies like color,

texture, size.

4. consider only first 100 bounding boxes.
5. calculate IOU for each bounding box with ground truth bounding boxes and only consider which has >0.5.
6. draw the boxes on the image.

```
# calculates the IOU and returns the value.
# Takes two lists as input.

def IOU(box1,box2):

    xA = max(box1[0], box2[0])
    yA = max(box1[1], box2[1])
    xB = min(box1[2]+box1[0], box2[2]+box2[0])
    yB = min(box1[3]+box1[1], box2[3]+box2[1])

    intersection = max(0, xB - xA + 1) * max(0, yB - yA + 1)
    boxAArea = box1[2]*box1[3]
    boxBArea = box2[2]*box2[3]

    iou = intersection / float(boxAArea + boxBArea - intersection)
    return iou


# Takes Image,bounding box and Color as input
# returns image with bounding box

def boxes(img, rects,color):
    imOut = img.copy()
    # itereate over all the region proposals
    for rect in rects:
        x, y, w, h = rect
        cv2.rectangle(imOut, (x, y), (x+w,y+h), color, 2)
    # show output
    return imOut

# shows the image

def show(image,title="",x=12,y=16):
#     figure(num=None, figsize=(x, y), dpi=80, facecolor='w', edgecolor='k')
    plt.imshow(image)
    plt.show()
```

```
# Reads the input images

img1=cv2.imread('2007_000464.jpg')
img2=cv2.imread('2007_001288.jpg')
img3=cv2.imread('2007_002953.jpg')
img4=cv2.imread('2007_005989.jpg')
```

```
# converts the images to RGB color space

img1=cv2.cvtColor(img1,cv2.COLOR_BGR2RGB)
img2=cv2.cvtColor(img2,cv2.COLOR_BGR2RGB)
img3=cv2.cvtColor(img3,cv2.COLOR_BGR2RGB)
img4=cv2.cvtColor(img4,cv2.COLOR_BGR2RGB)
```

```
# Ground truth bounding boxes

cow=[(71,252,216-71,314-252),(58,202,241-58,295-202)]
aeroplane=[(68,76,426-68,209-76),(357,272,497-357,317-272),
    (196,272,316-196,314-272)]
juice=[(25,102,93-25,295-102),(94,104,166-94,337-104),
        (165,103,247-165,347-103),(216,52,416-216,304-52),
        (89,61,162-89,141-61)]
racer=[(140,130,408-140,273-130),(213,96,355-213,260-96)]
```

```
# Takes image and GroundTruth bounding box
# calulates the Selective search regions and shows the images with boxes
# Considers only one strategy color space.

def onlyColor(img,GT):
    imageG=boxes(img,GT,(0, 255, 0))
    ss= cv2.ximgproc.segmentation.createSelectiveSearchSegmentation()
#     strategy Color
    color=cv2.ximgproc.segmentation.createSelective
                        SearchSegmentationStrategyColor()
#     adding only one strategy
    ss.addStrategy(color)
    ss.setBaseImage(img)
    ss.switchToSelectiveSearchQuality()
    rects = ss.process()
    temprects=boxes(img,rects[0:100],(0,0,255))
    mzx=[]
    uniqueMax=[]
    for j in GT:
        tempMax=0
        tempList=[0]
        count=0
        for i in rects:
            temp=IOU(j,i)
            if(tempMax<temp):
                tempMax=max(tempMax,temp)
                tempList[0]=i
            if(temp>=0.5):
                count+=1
                if(count>=100):
                    break
                mzx.append(i)
        uniqueMax.append(tempList[0])
```

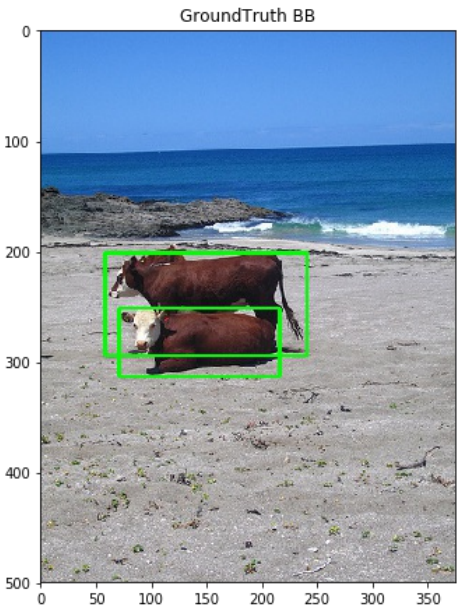
```
    imgUnique=boxes(img,uniqueMax,(255,0,0))
    imgGB=boxes(imageG,mzx,(0,0,255))
    print("_____________________________________________")
    print("Number of Bounding boxes generated:",len(rects))
    print("Number of Bounding boxes with IOU > 0.5 is ", len(mzx))
    recall=1
    if(len(GT)==len(uniqueMax)):
        recal=1
    print("recall = 1")
    print("_____________________________________________")
    from matplotlib import pyplot as plt
    f, axarr = plt.subplots(2,2,figsize=(18, 16))
    axarr[0,0].imshow(imageG)
    axarr[0,0].set_title("GroundTruth BB")
    axarr[0,1].imshow(temprects)
    axarr[0,1].set_title("All genrated BB")
    axarr[1,0].imshow(imgGB)
    axarr[1,0].set_title("BB > 0.5")
    axarr[1,1].imshow(imgUnique)
    axarr[1,1].set_title("BB Top IOU")
    plt.show()
```

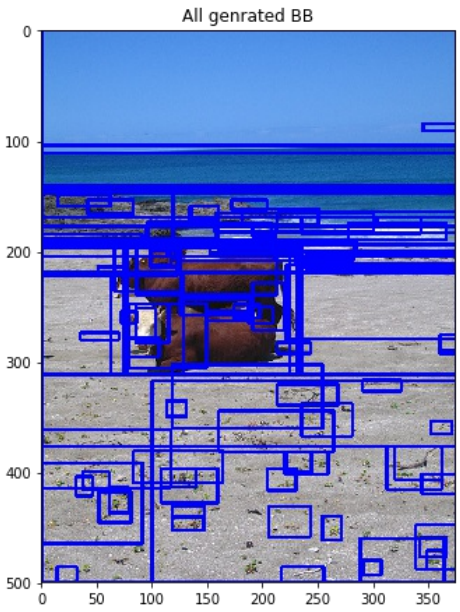
```
# Takes image and GroundTruth bounding box
# calulates the Selective search regions and shows the images with boxes
# Considers three strategies color,texture and Size.

def all_sta(img,GT):
    imageG=boxes(img,GT,(0, 255, 0))


    ss= cv2.ximgproc.segmentation.createSelectiveSearchSegmentation()
#     strategy Color
    color=cv2.ximgproc.segmentation.createSelectiveSearchSegmentationStrategyColor(
#     strategy texture
    stra_texture = cv2.ximgproc.segmentation.createSelectiveSearch
                SegmentationStrategyTexture()
#     strategy size
    size=cv2.ximgproc.segmentation.createSelectiveSearchSegmentationStrategySize()
    fill=cv2.ximgproc.segmentation.createSelectiveSearchSegmentationStrategyFill()
    stra_multi = cv2.ximgproc.segmentation.createSelectiveSearch
                SegmentationStrategyMultiple(stra_texture, color,size,fill)

#     adding all the strategies
    ss.addStrategy(stra_multi)
    ss.setBaseImage(img)
    ss.switchToSelectiveSearchQuality()
    rects = ss.process()
    temprects=boxes(img,rects[0:100],(0,0,255))
    mzx=[]
    uniqueMax=[]
    for j in GT:
```

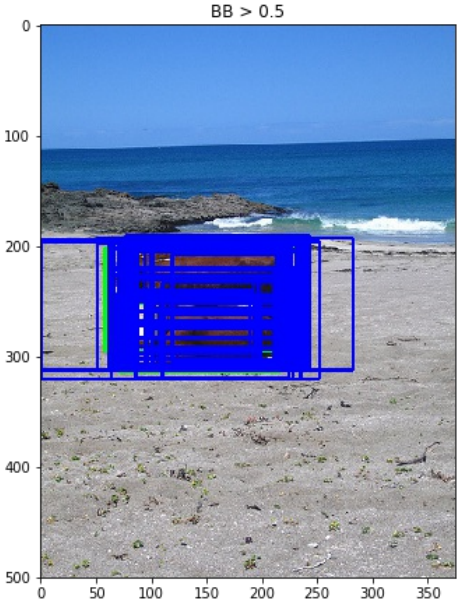
```
        tempMax=0
        tempList=[0]
        count=0
        for i in rects:
            temp=IOU(j,i)
            if(tempMax<temp):
                tempMax=max(tempMax,temp)
                tempList[0]=i
            if(temp>=0.5):
                count+=1
                if(count>=100):
                    break
                mzx.append(i)
        uniqueMax.append(tempList[0])


    imgUnique=boxes(img,uniqueMax,(255,0,0))
    imgGB=boxes(imageG,mzx,(0,0,255))
    print("_____________________________________________")
    print("Number of Bounding boxes generated:",len(rects))
    print("Number of Bounding boxes with IOU > 0.5 is ", len(mzx))
    recall=1
    if(len(GT)==len(uniqueMax)):
        recal=1
    print("recall = 1")
    print("_____________________________________________")

    from matplotlib import pyplot as plt
    f, axarr = plt.subplots(2,2,figsize=(18, 16))
    axarr[0,0].imshow(imageG)
    axarr[0,0].set_title("GroundTruth BB")
    axarr[0,1].imshow(temprects)
    axarr[0,1].set_title("All genrated BB")
    axarr[1,0].imshow(imgGB)
    axarr[1,0].set_title("BB > 0.5")
    axarr[1,1].imshow(imgUnique)
    axarr[1,1].set_title("BB Top IOU")
    plt.show()
```

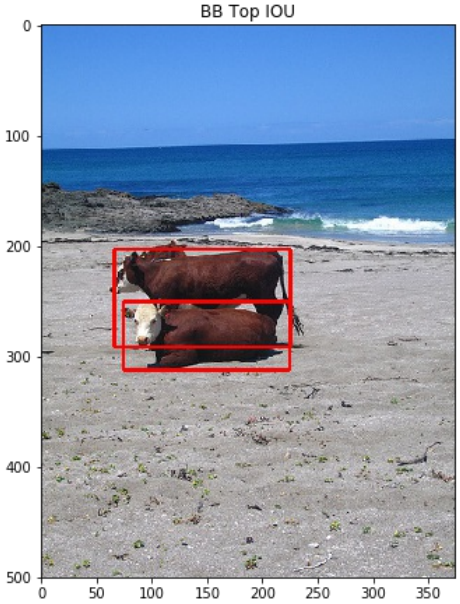
```
onlyColor(img1,cow)
```

```
_____________________________________________
Number of Bounding boxes generated: 3283
Number of Bounding boxes with IOU > 0.5 is  180
recall = 1
_____________________________________________
```



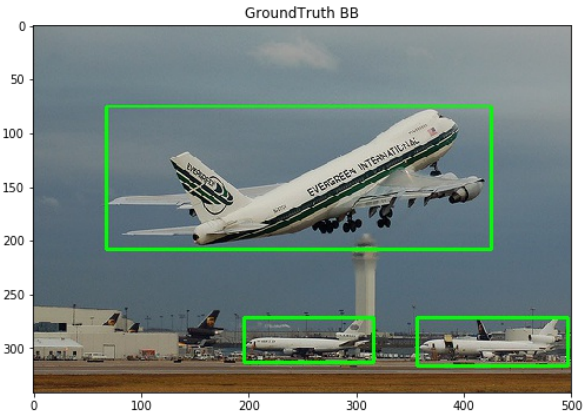


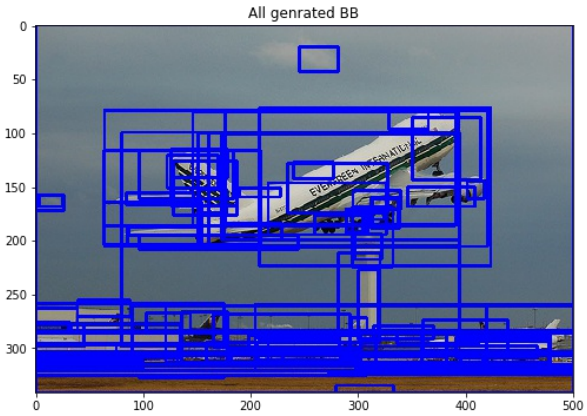


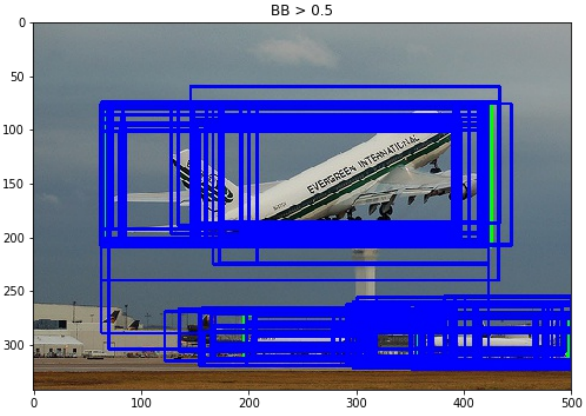


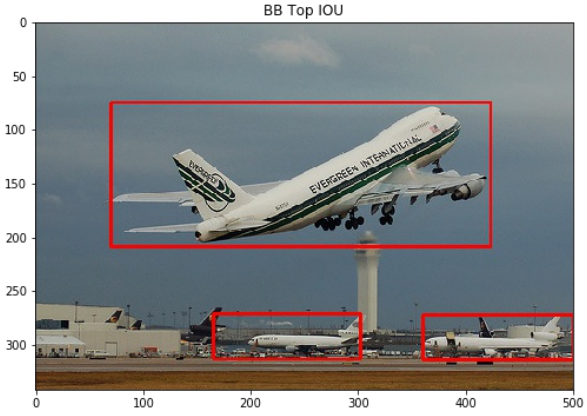

```
onlyColor(img2,aeroplane)
```

```
_____________________________________________
Number of Bounding boxes generated: 3542
Number of Bounding boxes with IOU > 0.5 is  233
recall = 1
_____________________________________________
```



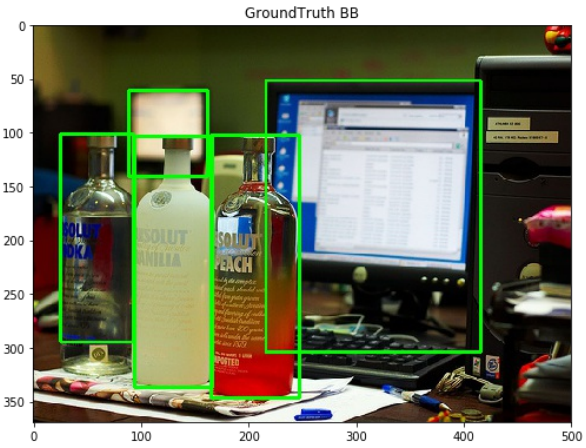


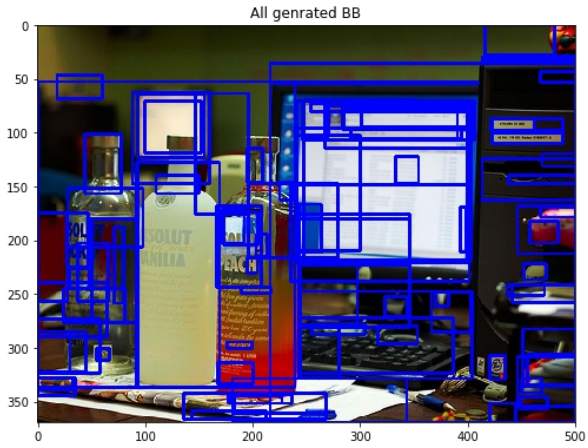


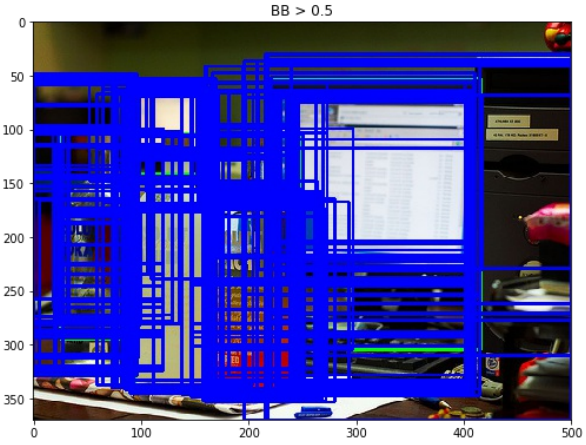


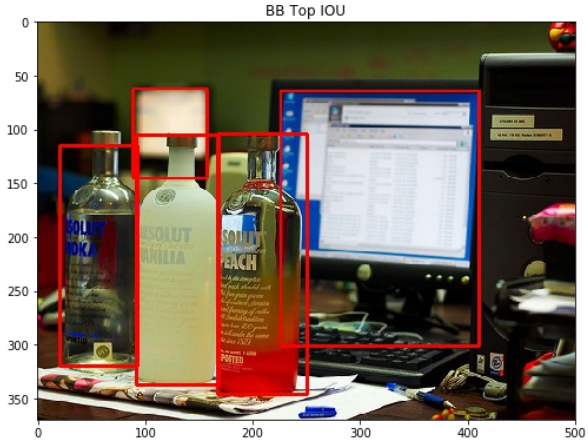

```
onlyColor(img3,juice)
```

```
_____________________________________________________
Number of Bounding boxes generated: 8429
Number of Bounding boxes with IOU > 0.5 is  481
recall = 1
_____________________________________________________
```



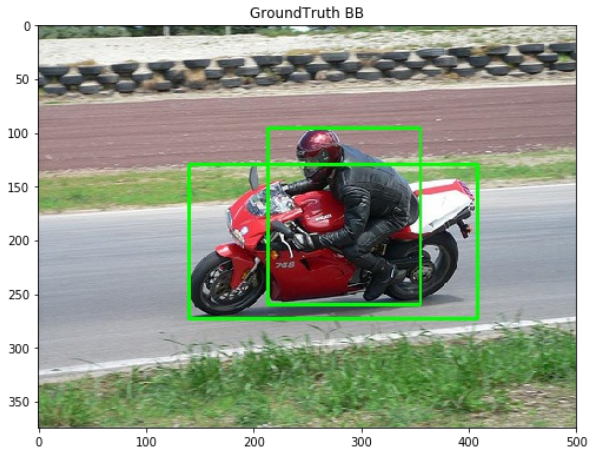


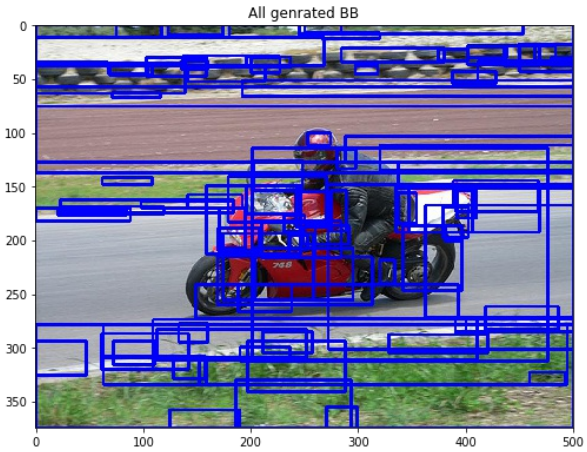


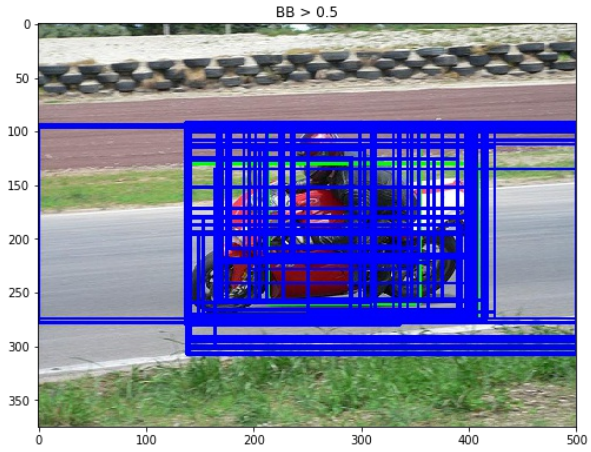


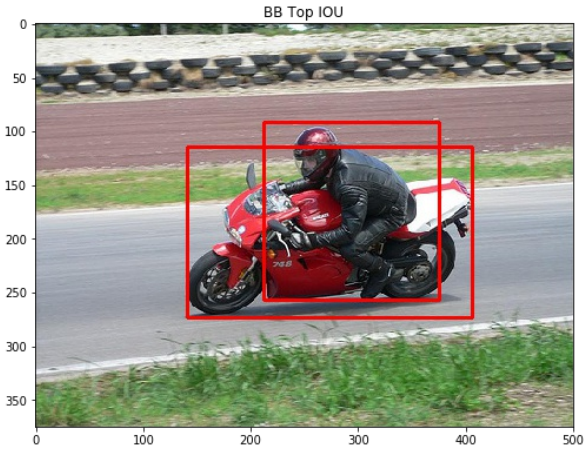

```
onlyColor(img4,racer)
```

```
_____________________________________________________
Number of Bounding boxes generated: 5034
Number of Bounding boxes with IOU > 0.5 is  162
recall = 1
_____________________________________________________
```



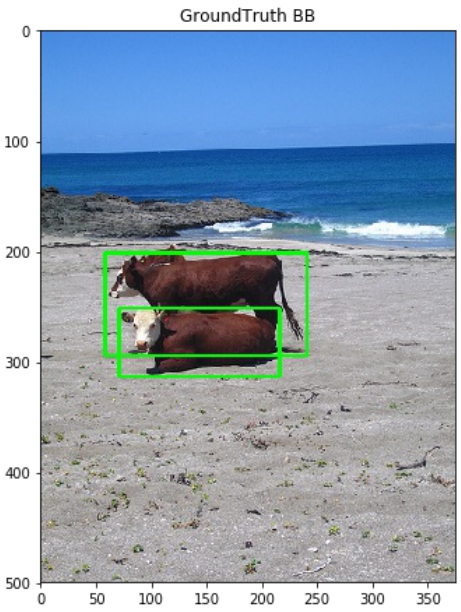


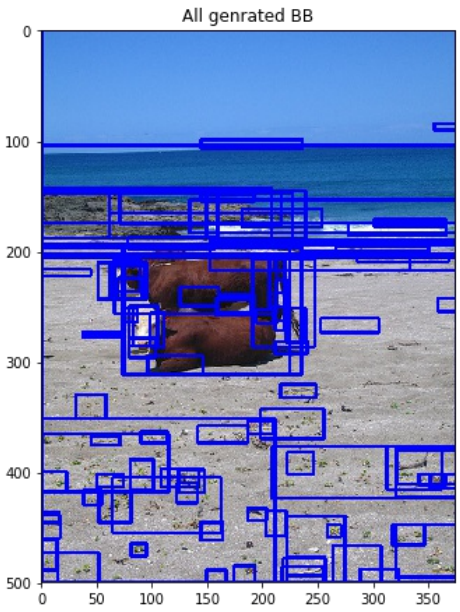


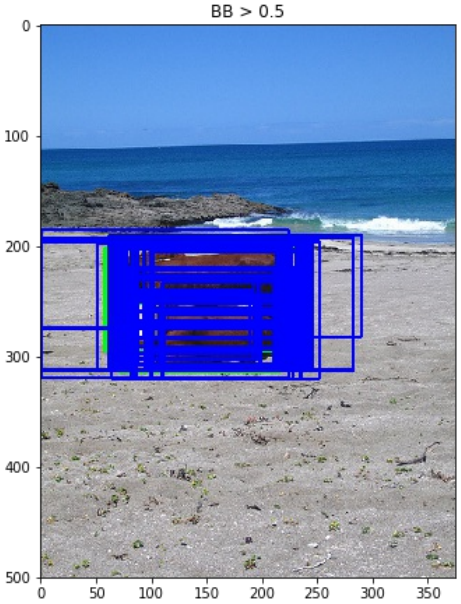


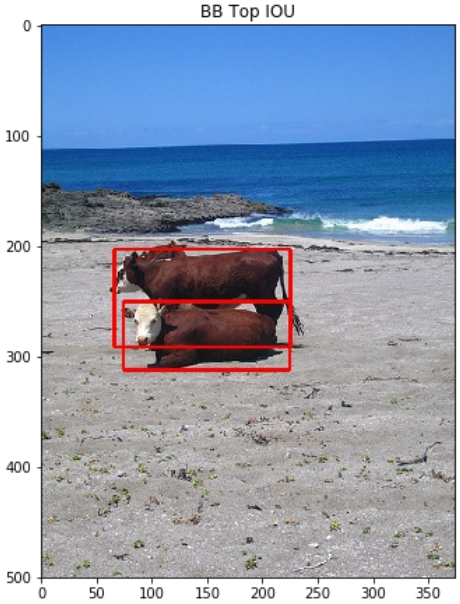

```
all_sta(img1,cow)
```

```
_____________________________________________________
Number of Bounding boxes generated: 3283
Number of Bounding boxes with IOU > 0.5 is  180
recall = 1
_____________________________________________________
```



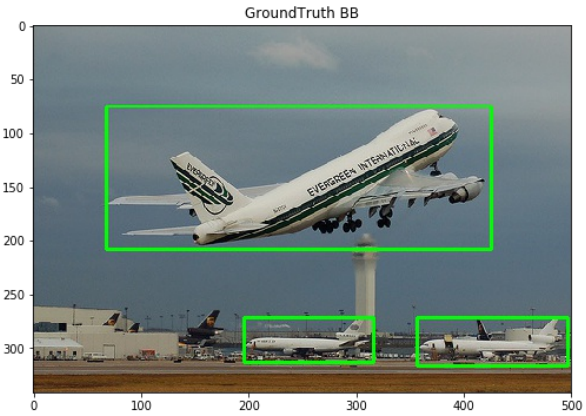


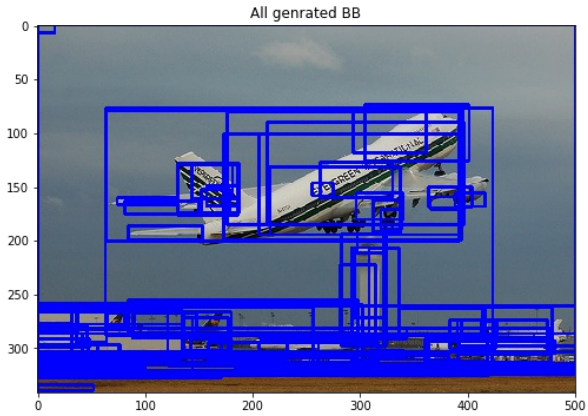


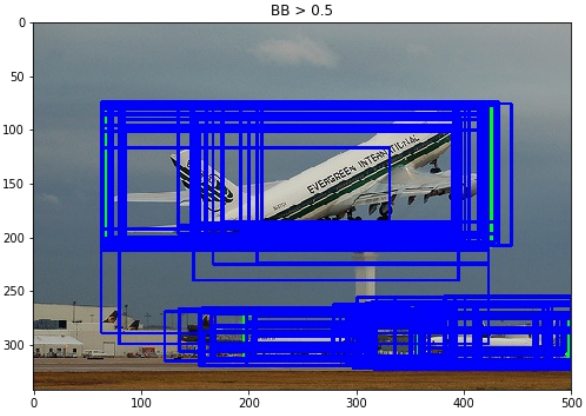


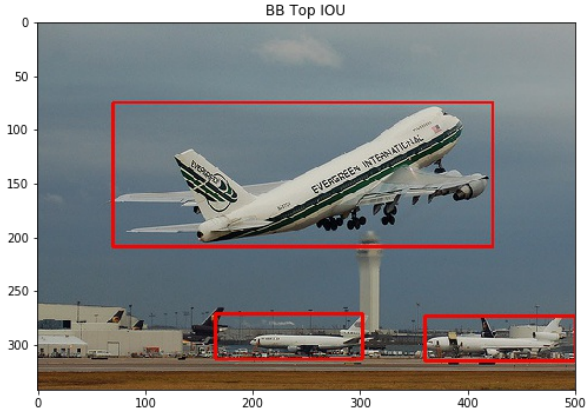

```
all_sta(img2,aeroplane)
```

```
_____________________________________________________
Number of Bounding boxes generated: 3542
Number of Bounding boxes with IOU > 0.5 is  233
recall = 1
_____________________________________________________
```



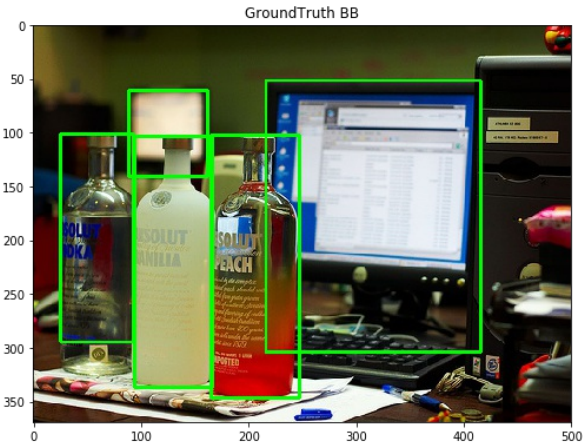


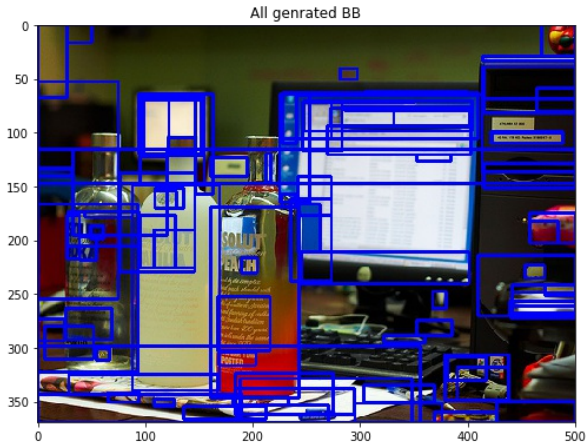


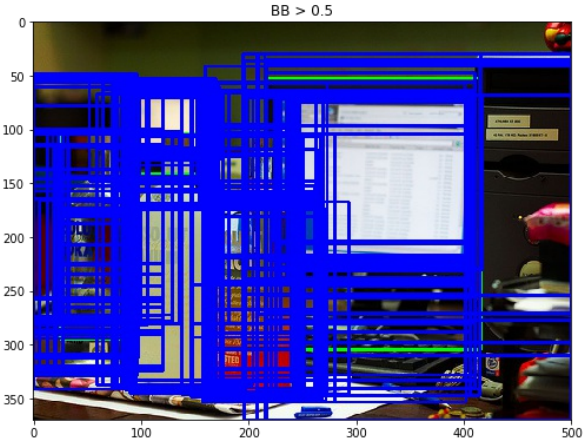


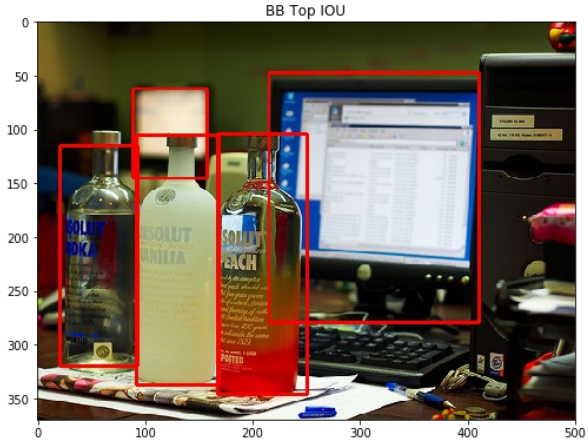

```
all_sta(img3,juice)
```

```
_________________________________________________
Number of Bounding boxes generated: 8429
Number of Bounding boxes with IOU > 0.5 is  481
recall = 1
_________________________________________________
```



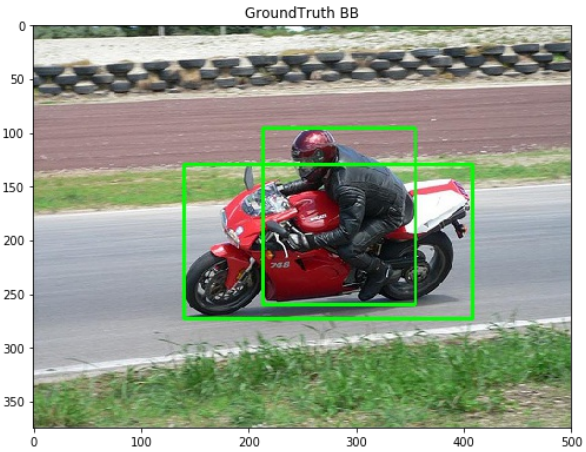


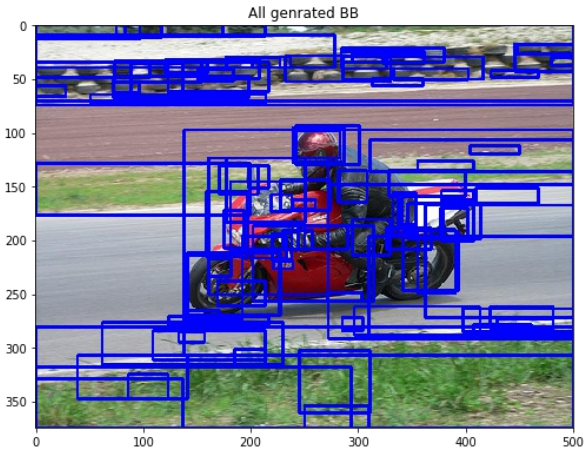


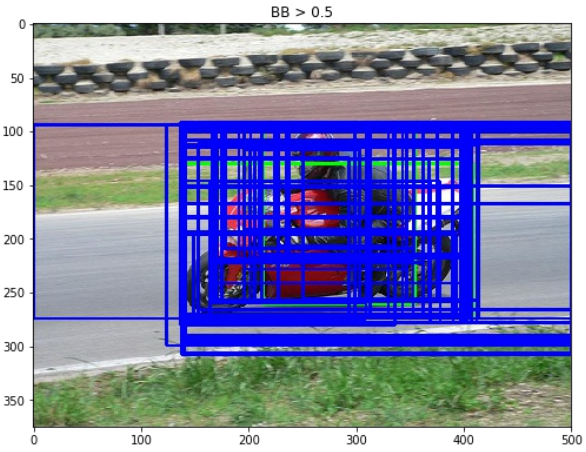


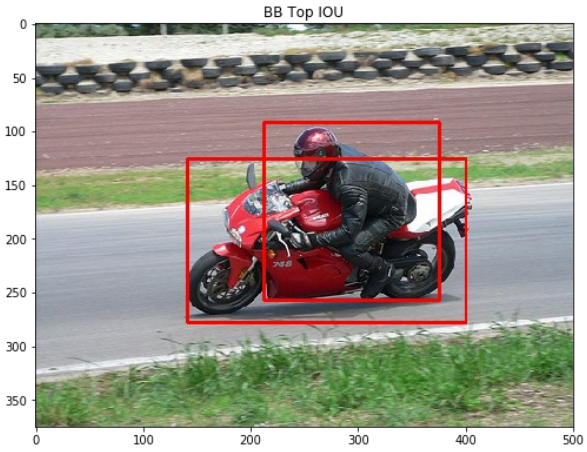

```
all_sta(img4,racer)
```

```
_____________________________________________________
Number of Bounding boxes generated: 5034
Number of Bounding boxes with IOU > 0.5 is  162
recall = 1
_____________________________________________________
```









Recall:

$$Recall = \frac{truePositives}{truePositives + falseNegatives}$$

Image-1

using only color strategy

- There are 2 ground truth bounding boxes and the predicted boxes also segmenting correctly. so the recall will be 1.

using only color strategy

- There are 2 ground truth bounding boxes and the predicted boxes also segmenting correctly. so the recall will be 1.

Image-2

using only color strategy

- There are 3 ground truth bounding boxes and the predicted boxes also segmenting correctly. so the recall will be 1.

using only color strategy

- There are 3 ground truth bounding boxes and the predicted boxes also segmenting correctly. so the recall will be 1.

Image-3

using only color strategy

- There are 5 ground truth bounding boxes and the predicted boxes also segmenting correctly. so the recall will be 1.

using only color strategy

- There are 5 ground truth bounding boxes and the predicted boxes also segmenting correctly. so the recall will be 1.

Image-4

using only color strategy

- There are 2 ground truth bounding boxes and the predicted boxes also segmenting correctly. so the recall will be 1.

using only color strategy

- There are 2 ground truth bounding boxes and the predicted boxes also segmenting correctly. so the recall will be 1.

Comparision

- Comparing color only approach and multiple strategies, the bounding boxes generated were slightly different. The multiple strategy approach is better compared to the color only. For example, we can consider the first image in only color approach, tail of cow is not covered in bounding box while multiple approache has covered the tail of the cow. Same way, we can see a slight difference in every image where multiple strategies are better than the color the only approach.

Conclusion:

- I have tried color only strategy first on all images then applied all strategies on same images,clearly multiple strategies gave better result.
- This algorithm is very much useful in classification as this reduces the number of bouding boxes drasstically from millions to few thousands.