Objective: *In this assignment, we are going to experiment with two "object proposal" methods:* ***Selective Search and Edge Boxes.***

1) **Selective Search:** Selective search is a segmentation based region proposal method where instead of exhaustive search for the object window, the algorithm uses graph based F-H segmentation algorithm to reduce the sample space for searching.

   In this method, the algorithm first over-segments the image based on pixel intensity and then hierarchically groups the pixels based on color, texture, shape, size and any linear combination of the features [1].

   Algorithm: (Complete code is attached in .pynb submission)

   a) **Get bounding boxes from the opencv implementation of selective search.**

```
#Module which performs Selective Search based on given parameters -> returns bounding boxes
def SelectiveSearch(image):
 #Create base object
      ss=cv2.ximgproc.segmentation.createSelectiveSearchSegmentatio()
      #Use different strategy
      Strategy=cv2.ximgproc.segmentation.createSelectiveSearchSegmentationStrategyMultiple()
      ss.addStrategy(strategy)
      ss.setBaseImage(image)
      #High recall but slow selective search
      ss.switchToSelectiveSearchFast()
      #Run Selective search process
      bb = ss.process()
      return bb
```

b) Subroutine to calculate IoU

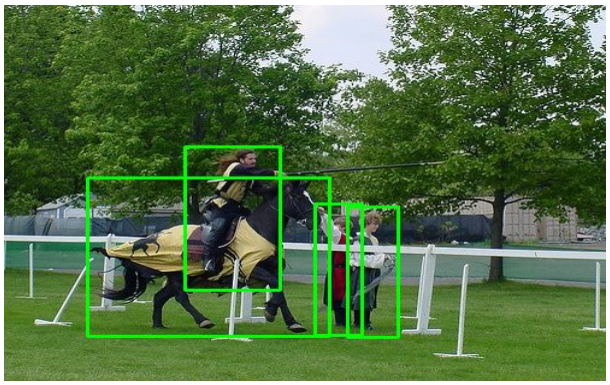
```
#Calculating IOU
def IoUCalculation(box, gt):
 #Find the area of 2 bounding boxes
 area_box = box[2]*box[3]
 area_gt = (gt[2] - gt[0] + 1) * (gt[3] - gt[1] + 1)
 union = area_box + area_gt

 #Finding the coordinates of the intersection rectangle
 topX = max(box[0], gt[0])
 topY = max(box[1], gt[1])
 bottomX = min(box[0]+box[2], gt[2])
 bottomY = min(box[1]+box[3], gt[3])
 intersection = max(0, bottomX - topX + 1) * max(0, bottomY -
topY + 1)
 IoU = intersection/(union - intersection)
 return IoU
```

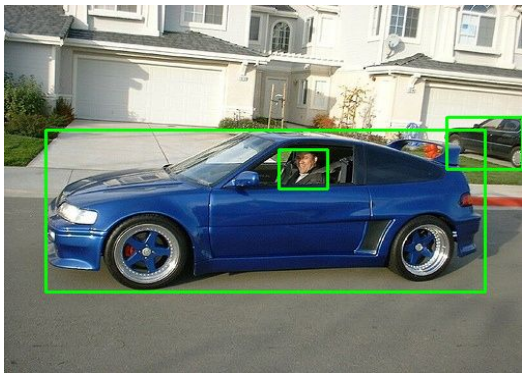
c) Subroutine to calculate recall:

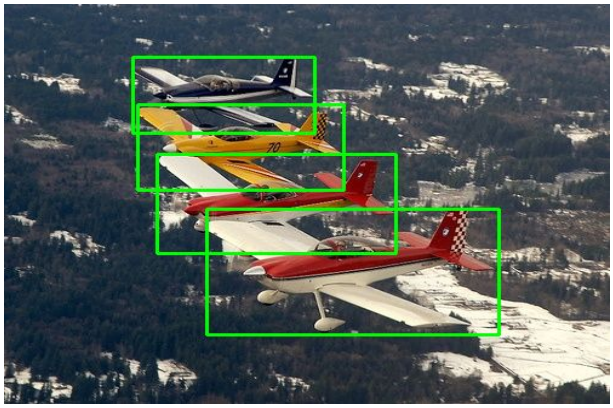
```
def Recall():
      count = 0
      for groundTruth in gt2:
            for proposal in bb:
                  IoU = IoUCalculation(proposal,groundTruth)
                  if(IoU > 0.5):
                        count += 1
                        Break
      recall = count/len(gt2)
      return recall
```

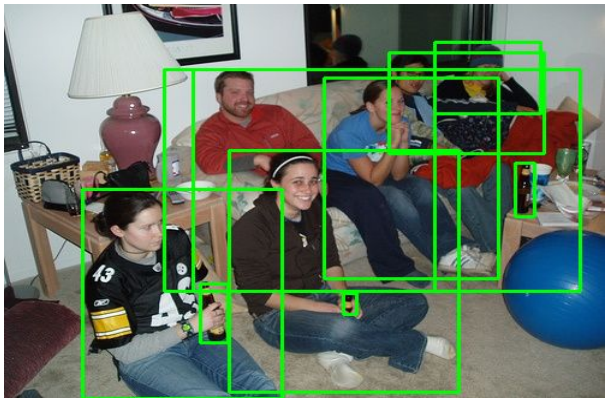
Results:

a) **Images with ground truths marked.**

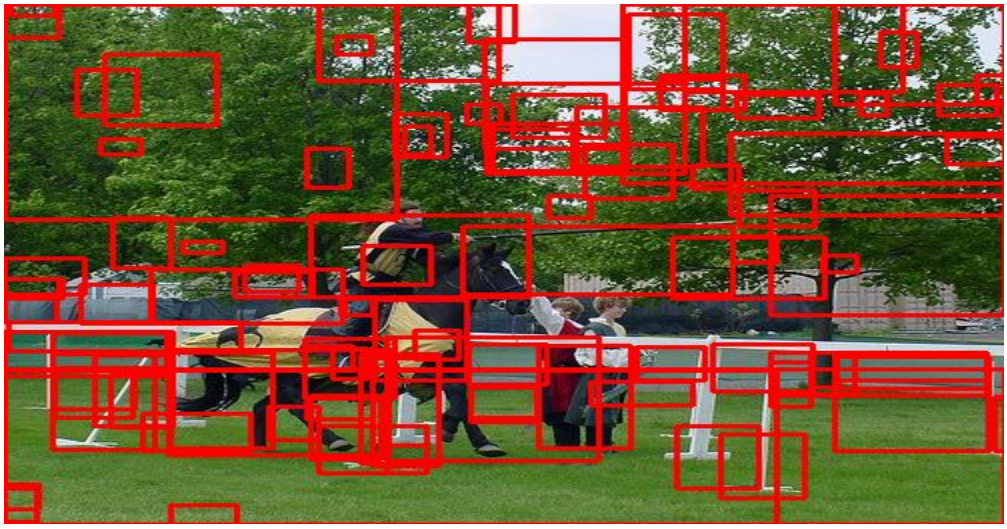


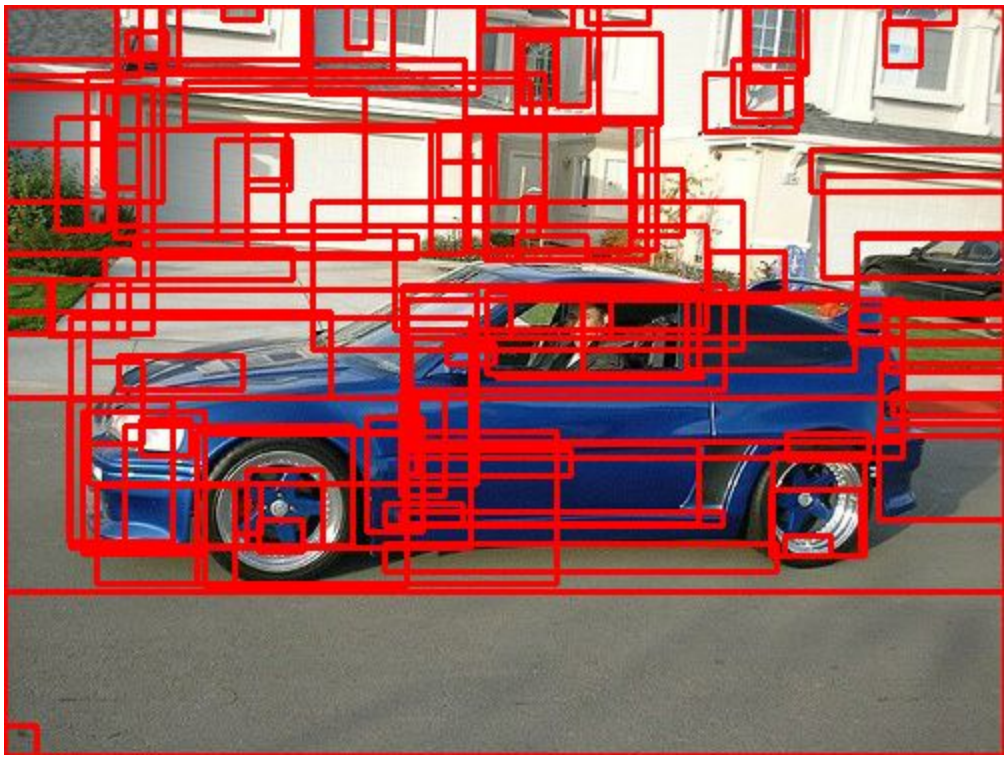


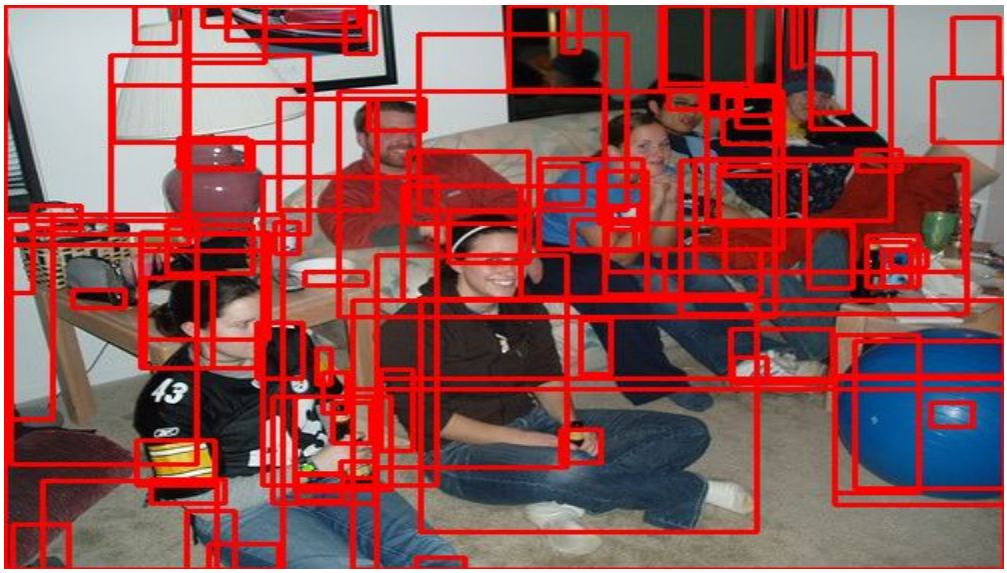




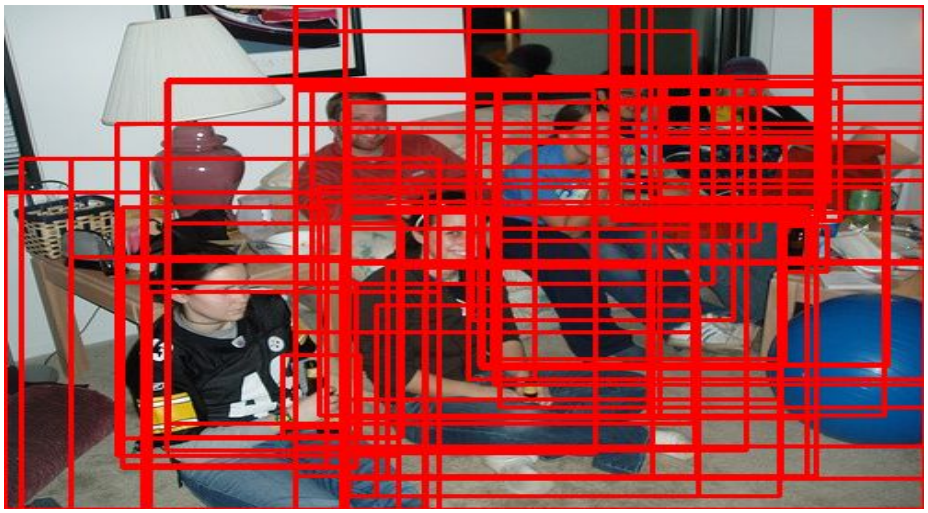
b) Images with first 100 proposals:

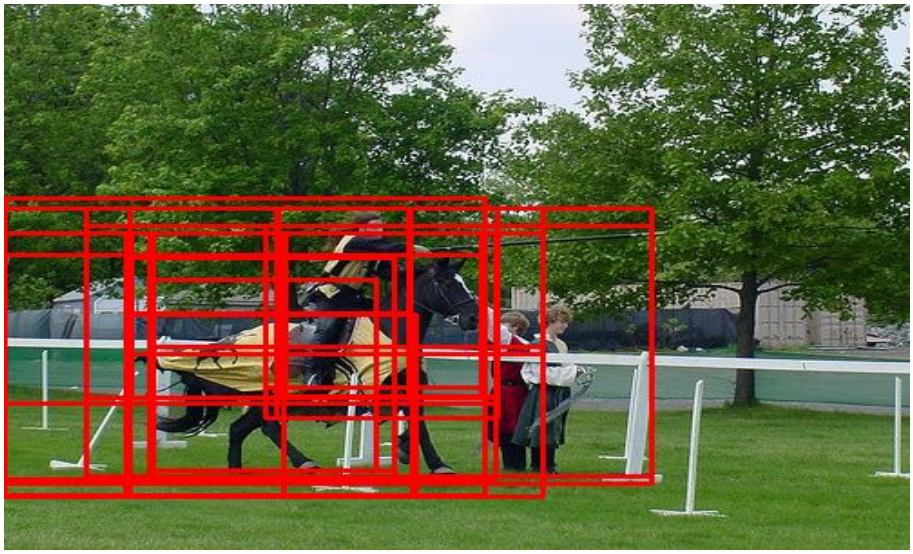


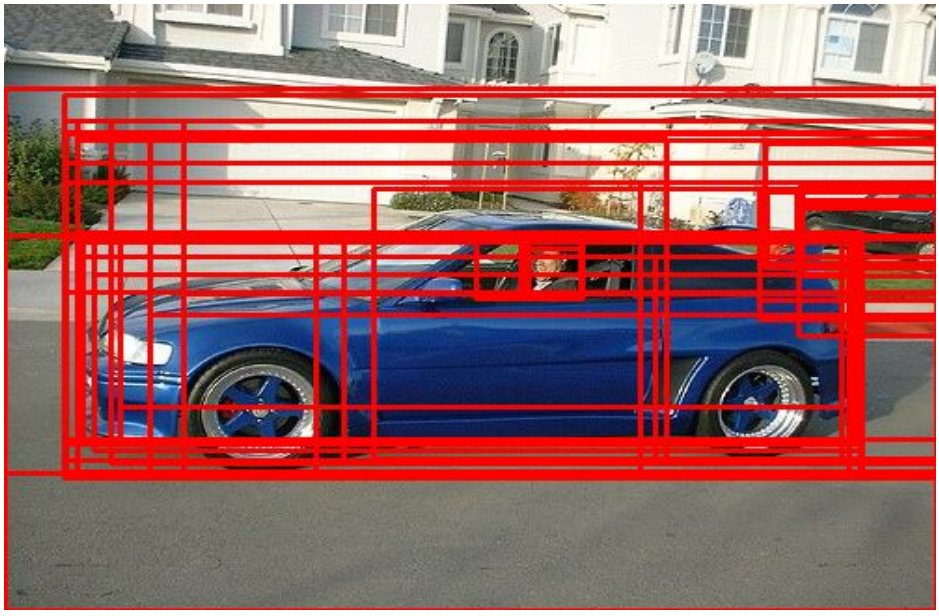




c) Images with IoU > 0.5:

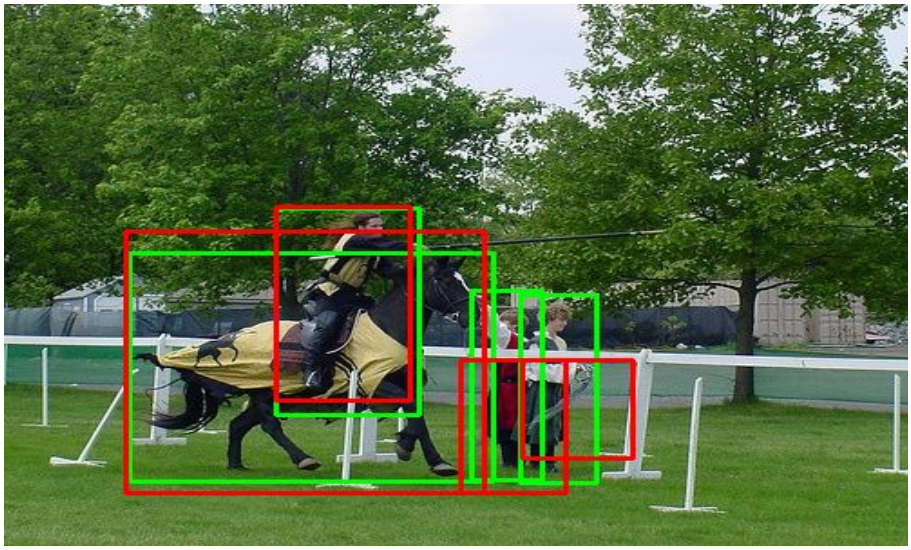


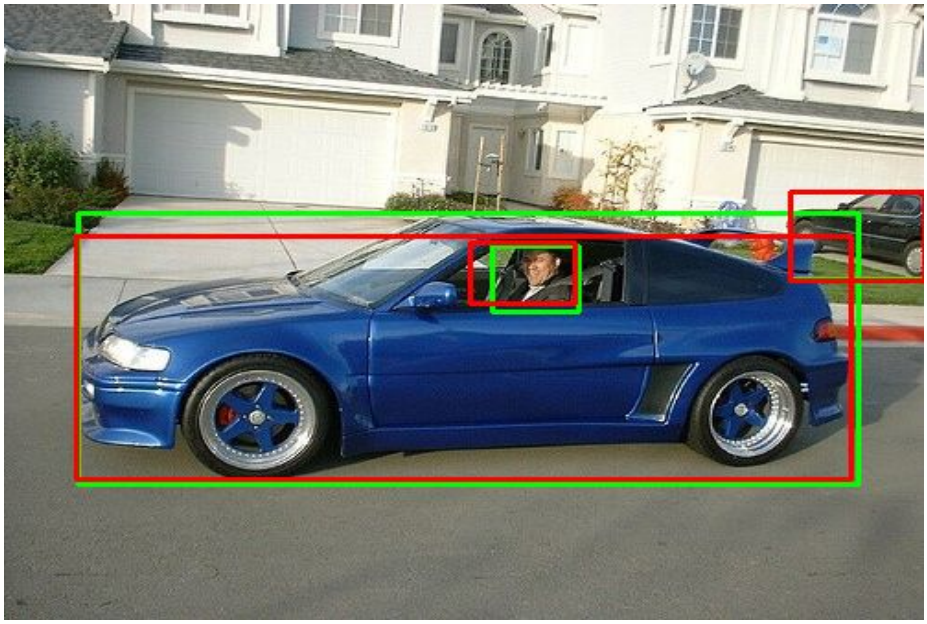


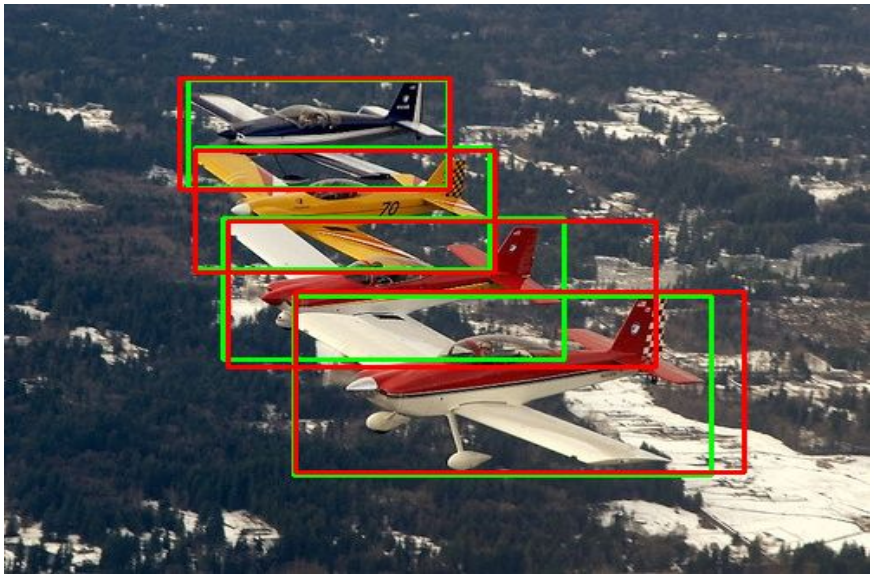


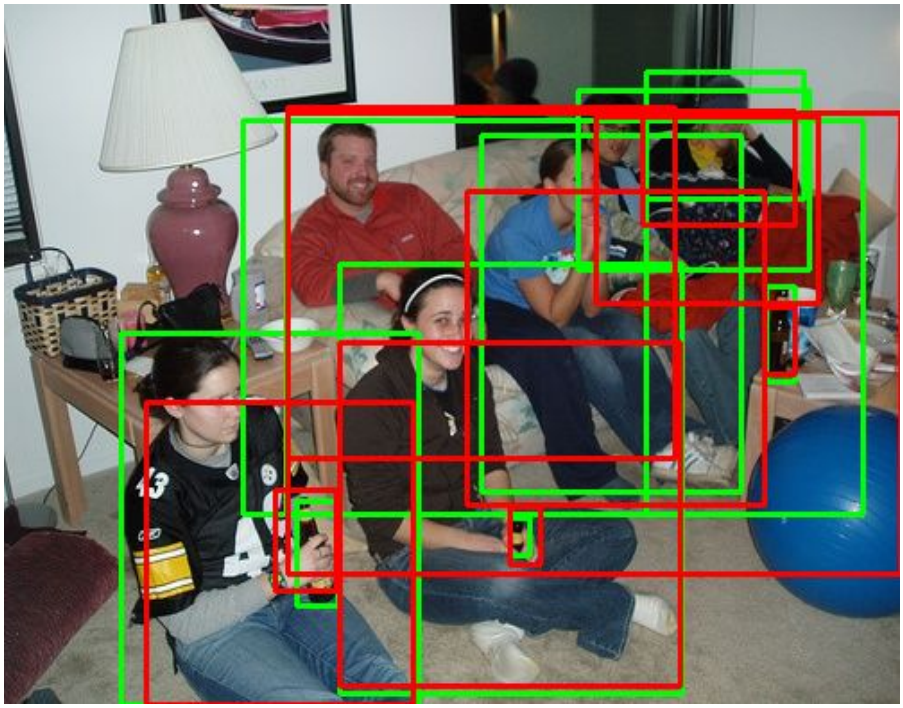
d) Final Bounding boxes and comparison with the ground truth.

(recall = 0.5 bounding boxes with respect to kids have IoU < 0.5)









Test results on the given images: Number of proposals, recall values are noted for all the images with different strategies like color, texture, size and multiple (lineas combination). Table 1 shows the results of experiments.

Table1 : Results of Selective Search on given images

Images	Total number of proposals	# of proposals for which IoU>0.5	Color (recall)	Texture (recall)	Size (recall)	Multiple (recall)
Image 1	2081	27	0.5	0.5	0.5	0.5
Image 2	1579	72	1.0	1.0	1.0	1.0
Image 3	1918	116	1.0	1.0	1.0	1.0
Image 4	2032	122	0.9	0.9	0.9	0.9

Conclusion: From the results in the above sections, it is evident that selective search worked well on all the images. However, for “Image 1”, we have a recall of 0.5 because the kids standing in front of the horse are not detected accurately by the detector. This is because of the occlusion with the background. Also for “Image 4”, we have 10 ground truths and the object detector did a great job with 9 objects. In this case, the overlapping objects confuses the segmentation algorithm.

Different strategies are implemented like Color, texture, size but the recall values and the number of proposal boxes didn’t change. This result seems to be unusual because the texture features are expected to be different than color.

Edge Boxes: Edge boxes use edge maps and orientation maps to obtain object bounding boxes. Edge maps can be generated by using any of the edge detection techniques. In this assignment, we are using a data driven approach to generate edge maps which are called **Structure edges (extension of sketch tokens).**

After generating edge map and orientation map, the edge points are connected using **Iterative end point fit algorithm.** The contour map is then used to generate a score to measure whether there should be a bounding box around the contour or not. This method gives better results compared to the Selective Search algorithm.

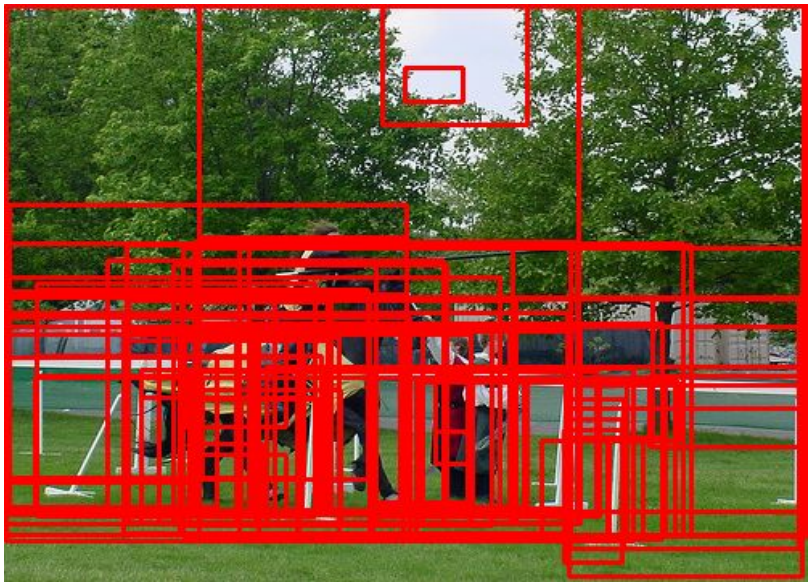
a) **Algorithm:**
   Step1: Use structured edges to generate edge maps and then use the edge maps to get edge boxes

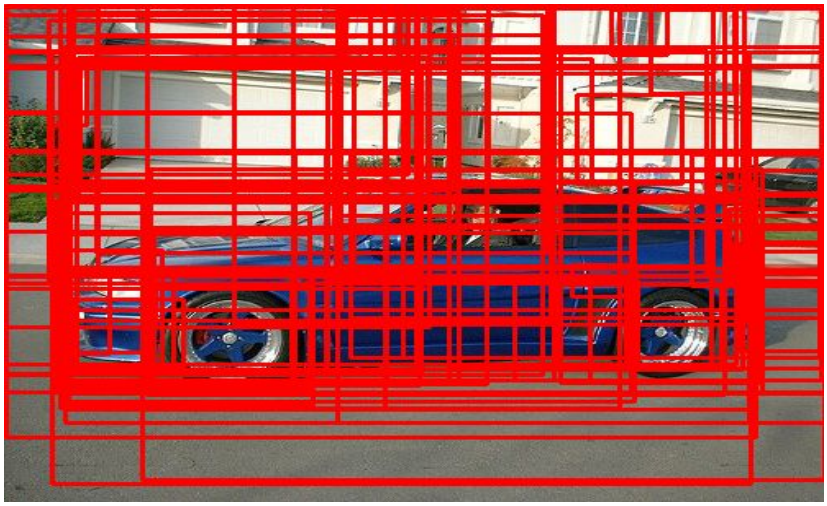
```
def EdgeBoxes(image, model):
 #Create Structure Edge Object
 find_edges = cv2.ximgproc.createStructuredEdgeDetection(model)
 #Convert BGR to RGB
 image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
 #Detect Edgemap
 edge_map = find_edges.detectEdges(np.float32(image_rgb) / 255.0)
 #Orientation map
 orientation_map = find_edges.computeOrientation(edge_map)
 #Non-maxima supression on edge map
 edges = find_edges.edgesNms(edge_map, orientation_map)
 #Edge boxe
 edge_boxes = cv2.ximgproc.createEdgeBoxes(alpha = 0.65, beta = 0.7)
 #edge_boxes.setMaxBoxes(100)
 boxes = edge_boxes.getBoundingBoxes(edges, orientation_map)
 boxes, scores = edge_boxes.getBoundingBoxes(edges, orientation_map)
 return boxes, scores
```

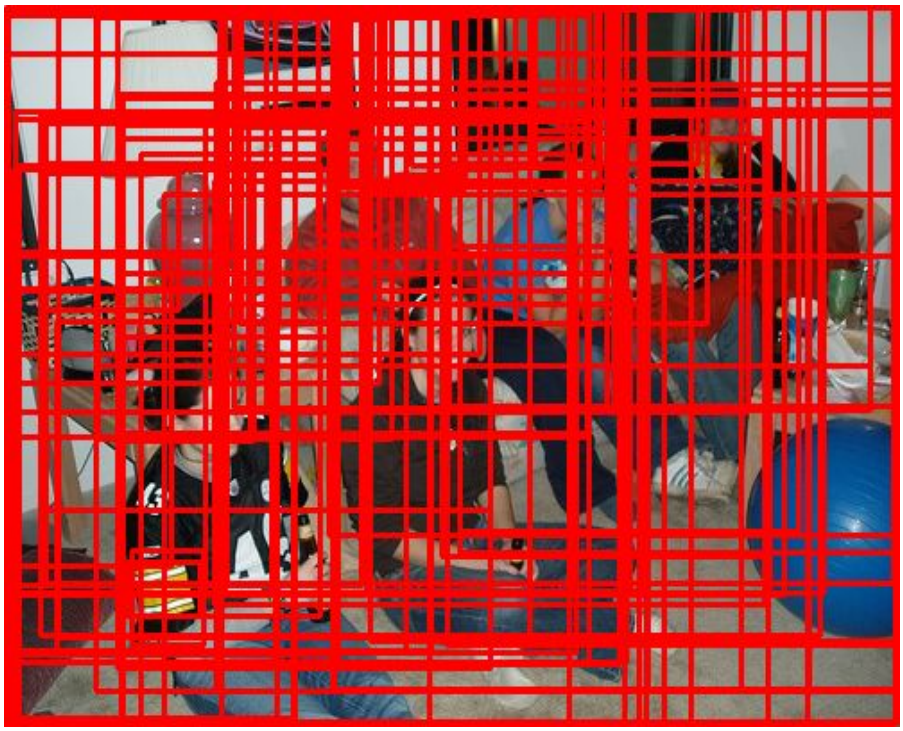
(**Rest of the code is similar to what we have used in Selective Search method)**

Results:

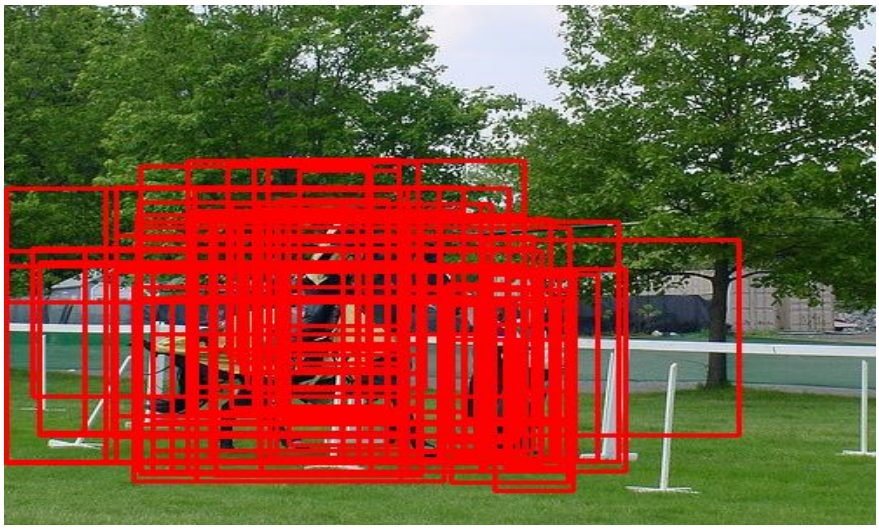
a) **Image with top 100 scoring edge boxes.**

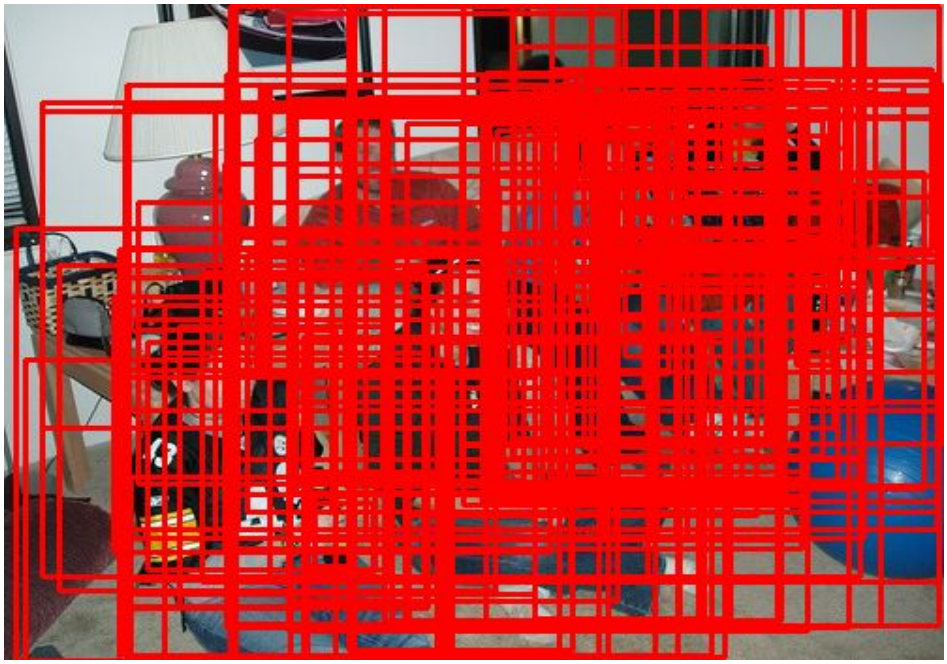




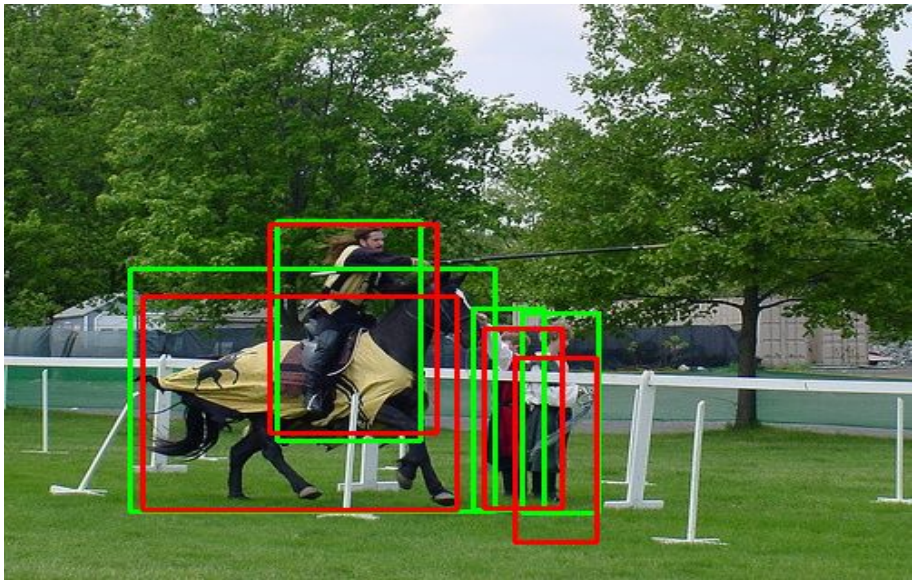


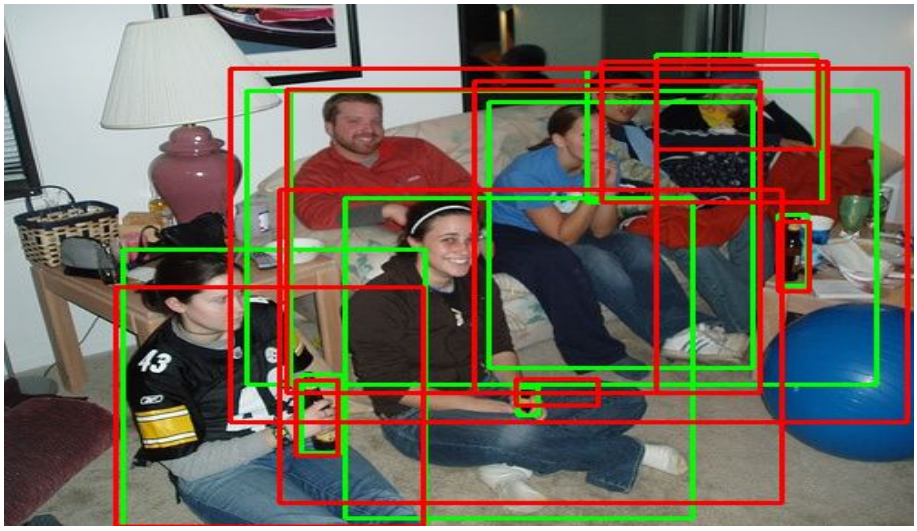
b) **Image with bounding boxes which have IoU>0.5 with ground truth.**

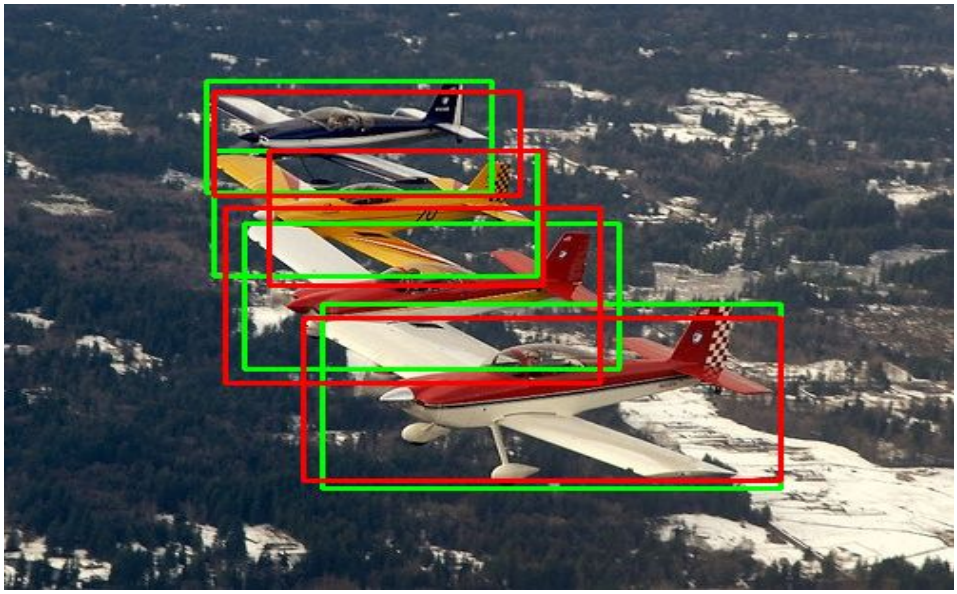


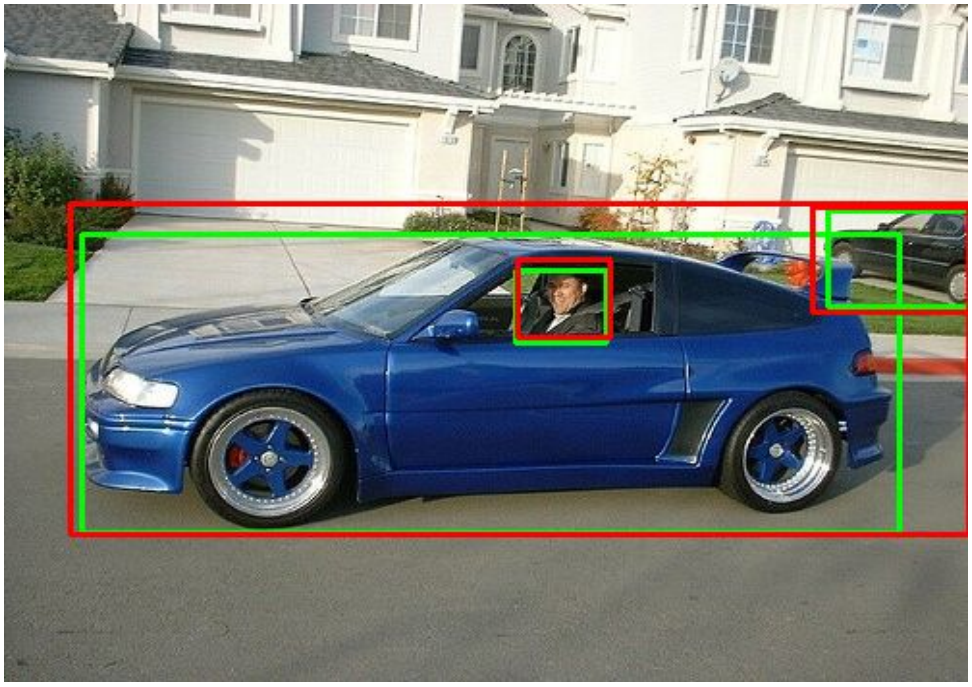


c) **Final results with comparison:**









Test results on the given images: Number of proposals, recall values are noted for all the images with different values of alpha and beta. Table 2-4 shows the results of experiments.

Table2 : Edge Boxes (alpha = 0.65, beta = 0.7)

Images	Total number of proposals	# of proposals for which IoU>0.5	Recall
Image 1	3849	55	1.0
Image 2	3024	32	1.0
Image 3	3184	59	1.0
Image 4	3693	133	0.9

Table3 : Edge Boxes (alpha = 0.35, beta = 0.4)

Images	Total number of proposals	# of proposals for which IoU>0.5	Recall
Image 1	3849	45	0.75
Image 2	3026	38	1.0
Image 3	3478	78	1.0
Image 4	3789	115	0.7

Table4 : Edge Boxes (alpha = 0.8, beta = 0.9)

Images	Total number of proposals	# of proposals for which IoU>0.5	Recall
Image 1	3849	45	1
Image 2	3026	38	1.0
Image 3	3478	78	1.0
Image 4	3789	115	0.9

Conclusion: ***Edge box gives better results than Selective Search method.*** We can actually see the difference in results of Image1 where the recall values have been increased from 0.5 to 1. Varying the alpha and beta parameters made a significant difference in the number of proposal boxes. However better results are achieved for **alpha = 0.65 and beta = 0.7 (all the results are shown for these values).**

Edge boxes usually fail where there are overlapping contours. We can see the performance degradation in Image 4 where we have some occlusion.