# CSCI677: HW2 Report

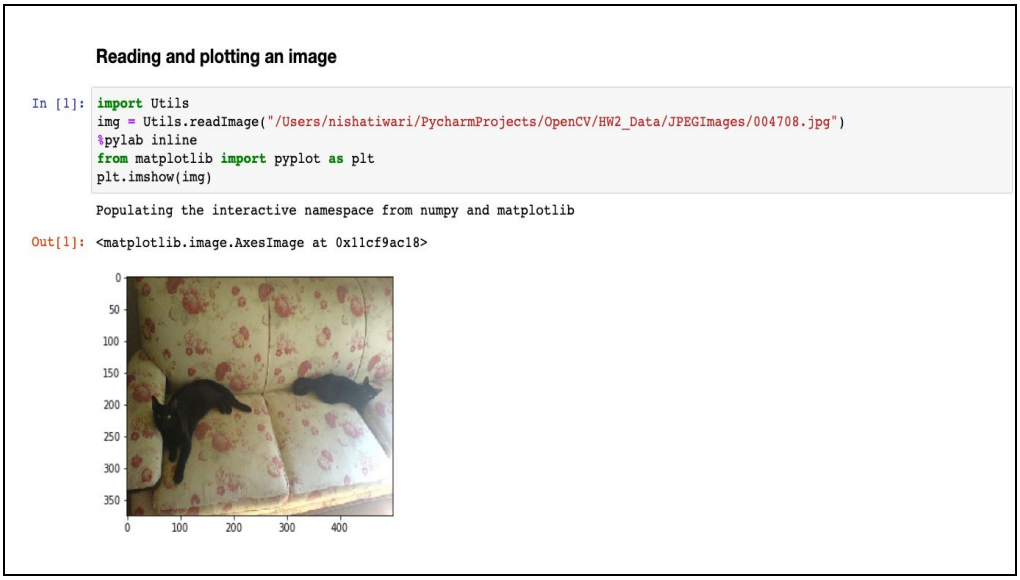
Nisha Tiwari | nishatiw@usc.edu | USCID-7888495181



## Code Description

The code for this assignment is organized in following main files:

1. **Utils.py** : holds the common methods required for both Selective Search and Edge Box detection algorithms
    a. **readImage** : Reads the image and converts it to RGB space.

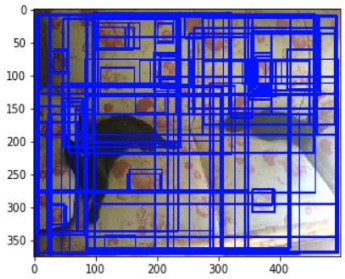



    b. **drawBoxesInImage** : Given the dimensions of a box, plots the boxes in the given image

**Plotting the top 100 bounding boxes from Edge Detection**

```
img_with_bboxes=Utils.drawBoxesInImagePredBox(img, 100, boxes,(0,0,255))
%pylab inline
plt.imshow(img_with_bboxes)
plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
```

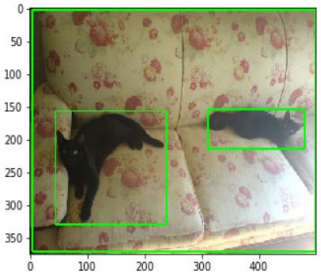



c. **parseXMLAndFindObjects** : Parses the provided annotations for an image and returns the bounding boxes dimensions.

**Parsing Annotations for the given image and plotting the ground truths on the image**

In [4]:
```
xmlPath='/Users/nishatiwari/PycharmProjects/OpenCV/HW2_Data/Annotations/004708.xml'
GTimageObjects=Utils.parseXMLAndFindObjects(xmlPath)
img_with_gt_bboxes=Utils.drawBoxesInImage(img, len(GTimageObjects), GTimageObjects, (0,255,0))
%pylab inline
plt.imshow(img_with_gt_bboxes)
plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
```

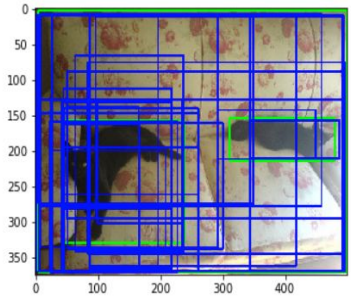



d. **calculateIOU** : Calculates the Intersection Over Union (IOU) for source and target bounding boxes and returns Precision, Recall, Maximum average Best Overlap, boxes with IOU greater than a given threshold, and the best-overlapped boxes for each ground truth image box.

**Calculating IOU and plotting all the bounding boxes greater than the IOU threshold**

```
threshold=0.5
mabo, precision, recall, selectedBoxes, bestOverlapBoxes =Utils.calculateIOU( boxes, GTimageObjects, threshold)
img_with_selectedBboxes=Utils.drawBoxesInImagePredBox(img_with_gt_bboxes, len(selectedBoxes), selectedBoxes, (0,0,255))
%pylab inline
plt.imshow(img_with_selectedBboxes)
plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
```




e. **calculateOverlapArea** : Given two bounding boxes, calculates the intersection area.
f. **calculateArea** : Given the dimension of a box, calculates its area.

2. **SelectiveSearchUtils.py**: holds methods specific to Selective Search
   a. **applySelectiveSearch**: Given an image and if combined strategy required or not, it runs the selective search on the image and returns the generated proposals. By default, the strategy is the color strategy.

**Calling Selective Search for the image above with isCombined flag (default as false)**

```
import SelectiveSearchUtils
bboxes=SelectiveSearchUtils.applySelectiveSearch(img, True)
```

3. **EdgeDetectionUtils.py**: holds methods specific to Edge Detection
   a. **applyEdgeDetection**: Given an image and the model, it runs the Edge Detection on the image and returns the generated proposals and respective scores.

**Calling Edge Detection for the image**

```
import EdgeDetectionUtils
model_path="/Users/nishatiwari/PycharmProjects/OpenCV/model.yml.gz"
boxes, scores=EdgeDetectionUtils.applyEdgeDetection(img, model_path)
```

4. **SelectiveSearch.ipynb**: Jupyter Notebook to run the selective search for an image
5. **EdgeBoxes.ipynb**: Jupyter Notebook to run the Edge Detection for an image

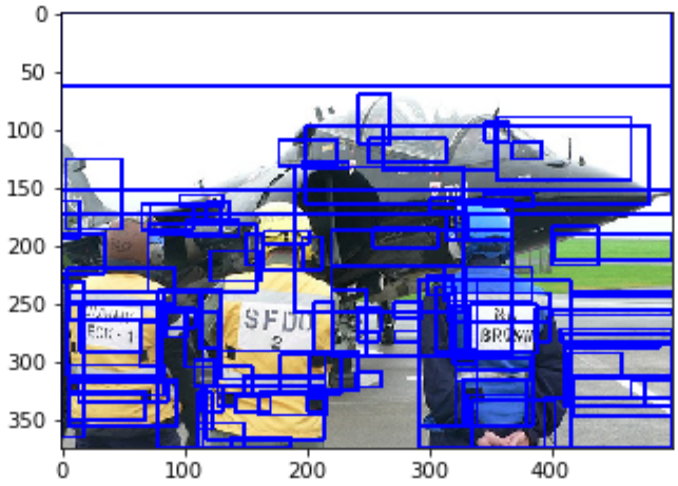
# Selective Search

Following is the image with the ground truth bounding boxes which I have chosen for my comparative analysis between Selective Search and Edge Box detection algorithms

**Image 1**


## Top 100 bounding boxes

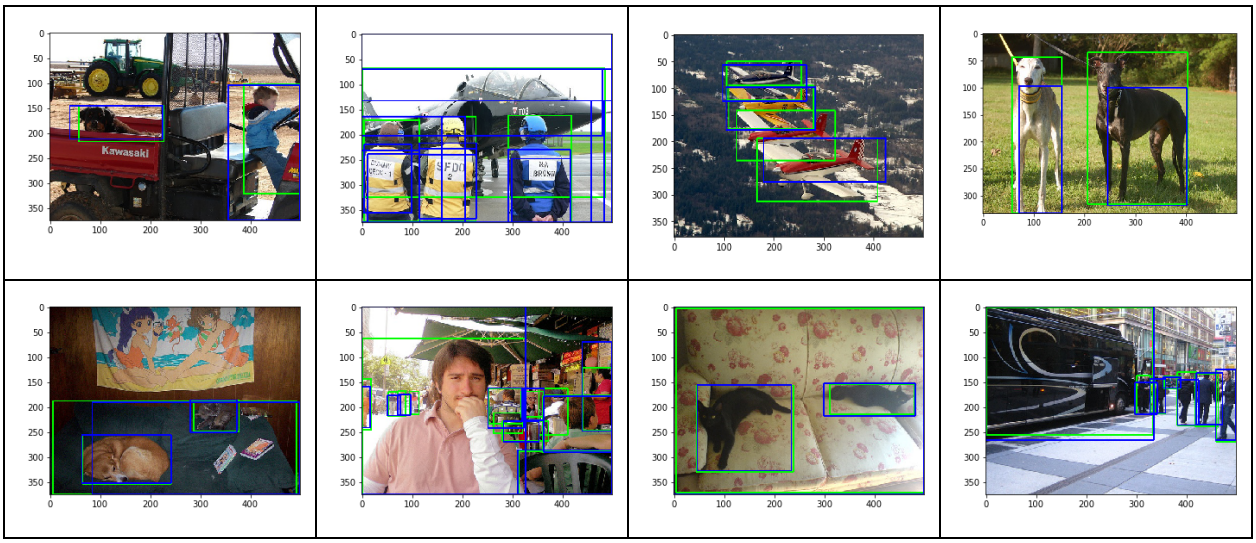
As already mentioned in the assignment's description, the Selective search does not have any explicit score for the bounding boxes hence, I display here the first 100 bounding boxes returned by the algorithm.




## Object Proposals for each image

The following are the best proposals based on the combined strategy and have IOU>0.5.

**Green-Ground Truth, Blue- SS bounding box**



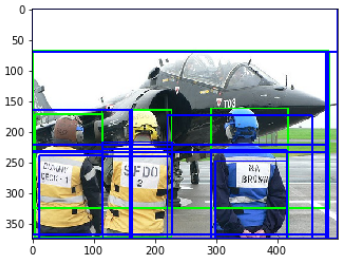
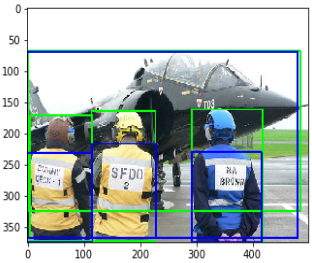
## Results and Analysis

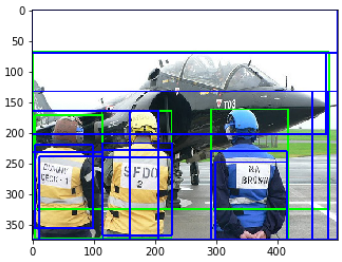
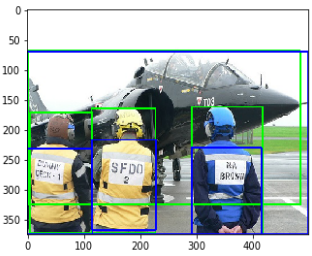
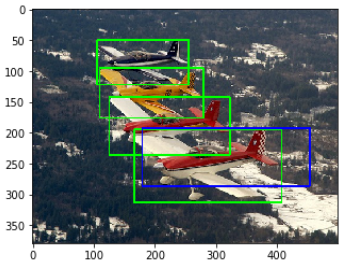
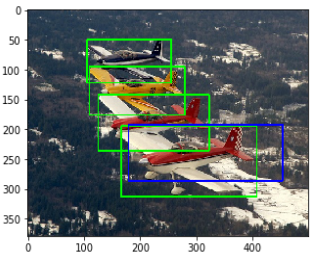
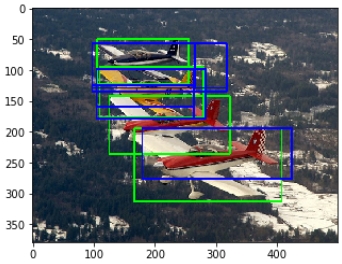
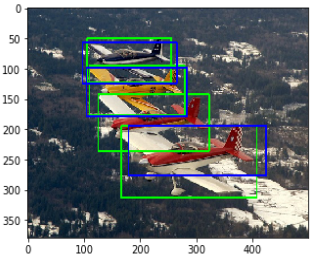
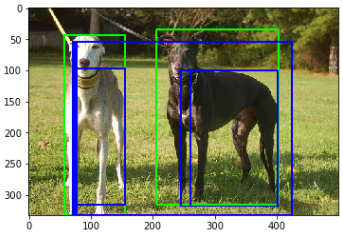
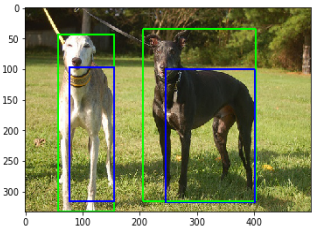
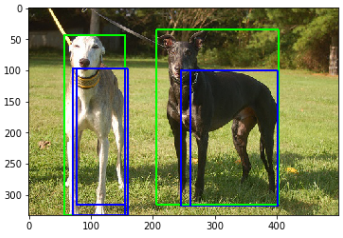
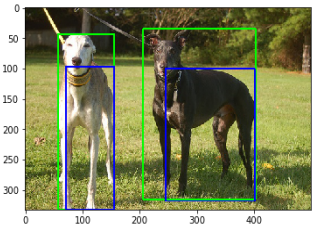
Table 1 shows the results for the Selective Search algorithm for the image above (and 2 more images for better analysis ).

1. As we see from the table, the results are better when using the combined strategy.
2. Recall for the color strategy varies between 25 and 100 and for the combined strategy, it varies from 75 to 100. The lower recall for both strategies is for the same image. This implies that the performance of the algorithm depends on the given image.
3. Highest Maximum Average Best Overlap (MABO) score computed over the bounding boxes with IOU>0.5 is ~72%.

**Table 1**

MABO =sum of IOU for the best overlap for each ground truth image/ number of ground truth images

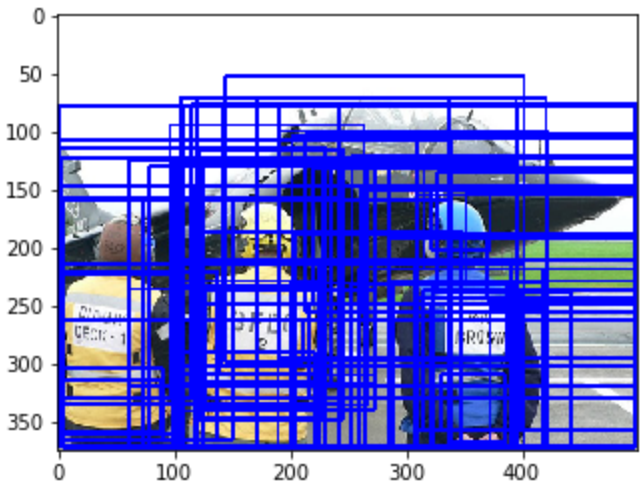
| **Strategy** | **Image (IOU>0.5)** | **Best IOU overlap boxes (Green-Ground Truth, Blue- SS bounding box)** | **MABO** | **Recall (%)** | **#bounding boxes (IOU>0.5)** |
|---|---|---|---|---|---|
| **Color** |  |  | 0.723 | 100 | 17 |

| | | | | | |
|---|---|---|---|---|---|
| **Combined** |  |  | 0.713 | 100 | 16 |
| **Color** |  |  | 0.157 | 25 | 1 |
| **Combined** |  |  | 0.57 | 75 | 8 |
| **Color** |  |  | 0.60 | 100 | 6 |
| **Combined** |  |  | 0.651 | 100 | 5 |

# Edge Detection

## Top 100 Bounding Boxes

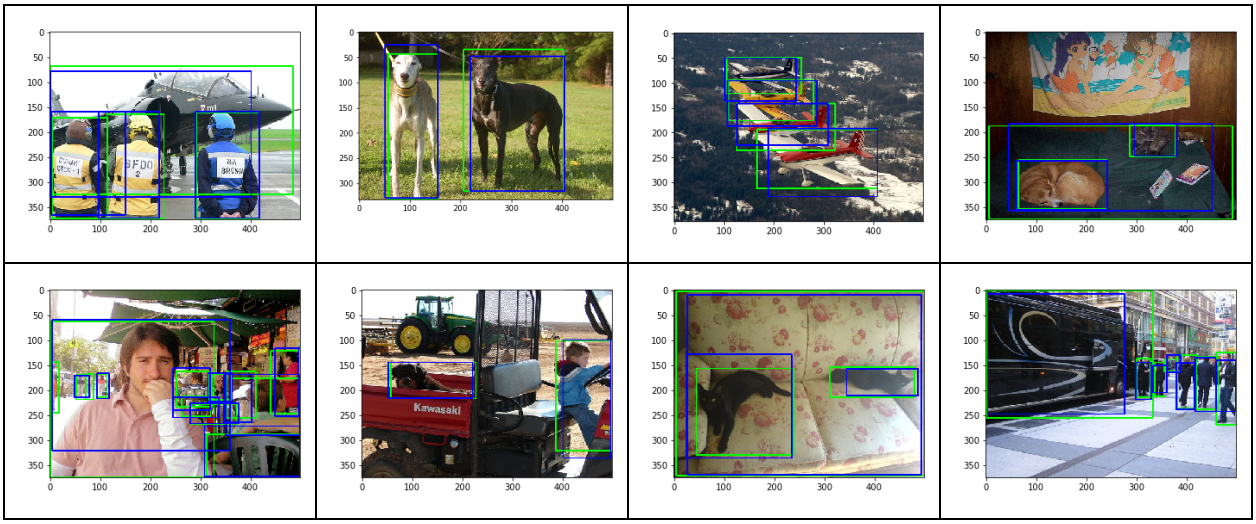
Edge detection algorithms return the scores along with the bounding boxes. Below is the image with top scorer (100)  bounding boxes.



## Object Proposals for each image

The following are the best proposals that have IOU>0.5 and default values of alpha and beta.

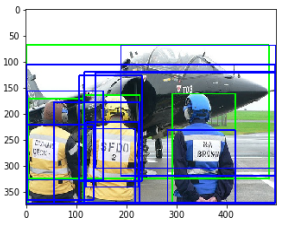
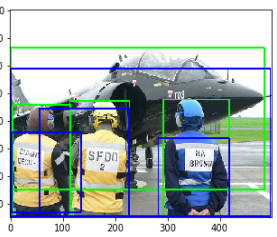
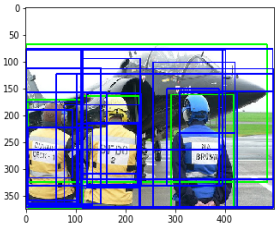
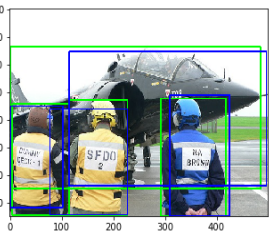
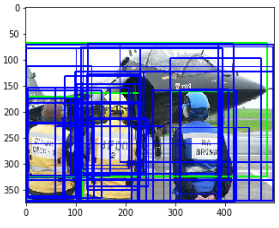
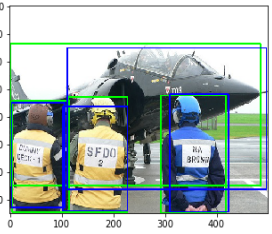
**Green-Ground Truth, Blue- SS bounding box**

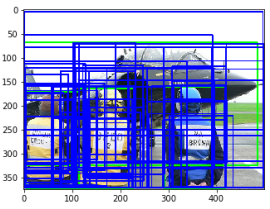
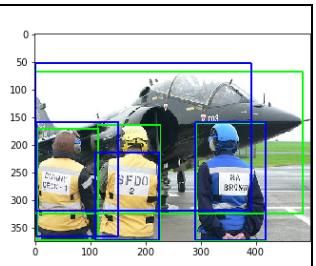
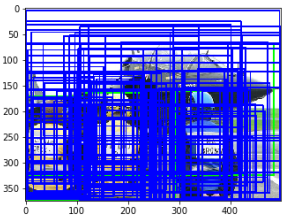
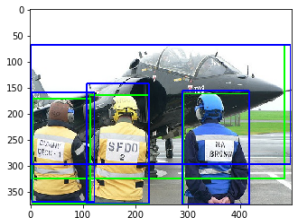
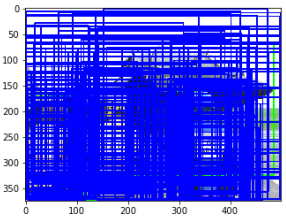
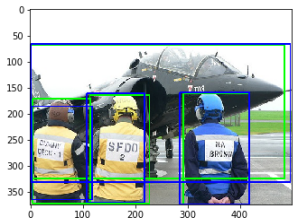
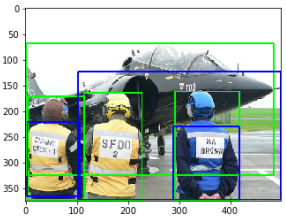
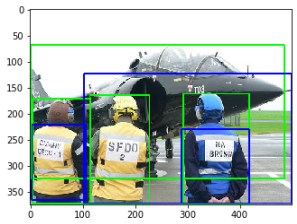
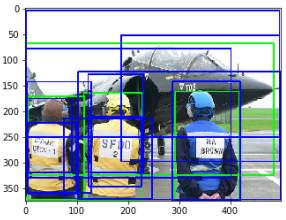
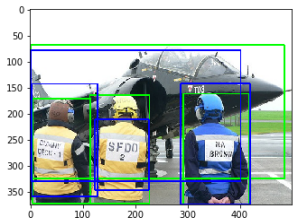
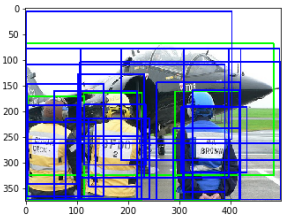
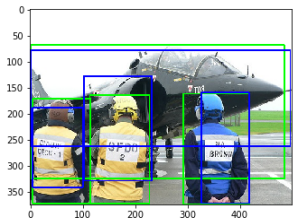


## Results and Analysis

1. From chart 1, we see that as we increase the alpha value, the algorithm finds more bounding boxes and takes more time to generate them (execution times are not mentioned in the chart)
2. From table 2 we can observe that, as the alpha and beta value increases, the MABO value also increases.
3. We see a similar observation (as in point 1) from table 2 that as we increase beta, the number of bounding boxes above the IOU threshold increases.
4. As per the runtimes for different values of alpha and beta (not shown in table 2), I have also observed that as the value of alpha and beta increases, the algorithm takes more time to return all the proposals.

**Table 2**

| α | β | Recall (%) | MABO | #bounding boxes(IOU>0.5) | Image with all bounding boxes(IOU>0.5) | Image with the best overlap Boxes(Green-Ground Truth, Blue- SS bounding box) |
|---|---|---|---|---|---|---|
| 0.4 | 0.75 | 100 | 0.65 | 11 |  |  |
| 0.5 | 0.75 | 100 | 0.75 | 26 |  |  |
| 0.6 | 0.75 | 100 | 0.81 | 39 |  |  |

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.65 | 0.75 | 100 | 0.78 | 59 |  |  |
| 0.8 | 0.75 | 100 | 0.88 | 151 |  |  |
| 0.9 | 0.75 | 100 | 0.90 | 262 |  |  |
| 0.65 | 0.3 | 75 | 0.46 | 3 |  |  |
| 0.65 | 0.5 | 100 | 0.74 | 11 |  |  |
| 0.65 | 0.6 | 100 | 0.67 | 21 |  |  |

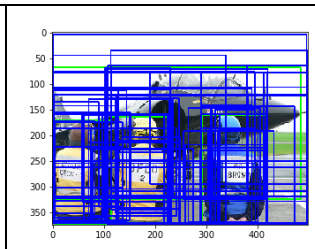
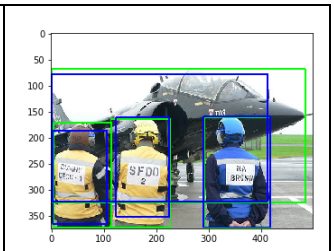
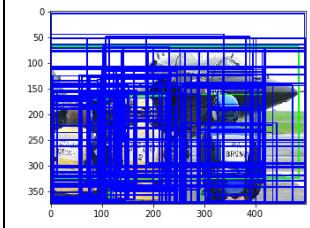
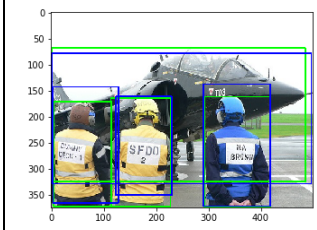
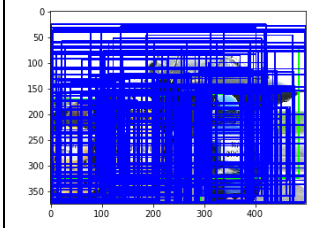
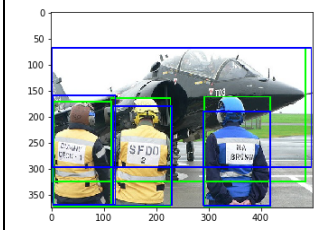
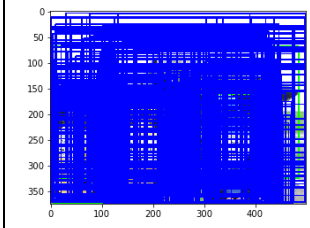
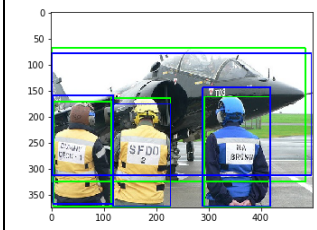
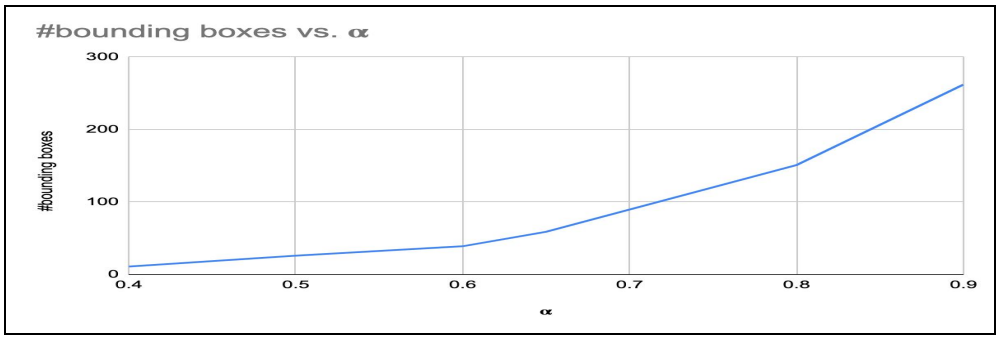
| | | | | | | |
|---|---|---|---|---|---|---|
| 0.65 | 0.8 | 100 | 0.84 | 74 |  |  |
| 0.65 | 0.9 | 100 | 0.83 | 146 |  |  |
| 0.8 | 0.8 | 100 | 0.86 | 262 |  |  |
| 0.9 | 0.8 | 100 | 0.89 | 551 |  |  |

**Chart 1**



## Comparative Analysis of the two methods

1. Both Selective Search and Edge detection algorithms are high recall algorithms in most of the cases (As the performance of Edge detection is dependent on alpha and beta settings, the Selective Search's performance also depends on the parameter settings for createGraphSegmentation() method).
2. The MABO value for **Image 1** found for Selective search (for the default configuration of createGraphSegmentation() method) is 0.72 and for Edge Detection (highest among all alpha and beta configurations) is 0.89.
    a. The number of bounding boxes generated is 17 and 551 respectively for Selective Search and Edge Detection.
3. MABO for both algorithms mostly increases as the number of proposals generated by them increases.