HOMEWORK ASSIGNMENT #2
DUE: 2:00 PM PDT, Thursday, September 24, 2020
CSCI 677: Advanced Computer Vision, Prof. Nevatia
Fall Semester, 2020

This is a programming assignment. You are asked to experiment with two methods of generating "object proposals": Selective Search and Edgeboxes, which we studied in class and are implemented in the OpenCV library.

Documentation:

Details of the *selective search* method can be found in:
https://docs.opencv.org/4.1.1/d6/d6d/classcv_1_1ximgproc_1_1segmentation_1_1SelectiveSearchSegmentation.html

Details of *Edgeboxes* can be found in:
https://docs.opencv.org/4.1.1/dd/d65/classcv_1_1ximgproc_1_1EdgeBoxes.html

Code and Variations:

Write a program to read input images, apply Selective Search and Edgeboxes functions to them, display the results and measure accuracy with respect to provided GroundTruth (GT). Both methods take a number of parameters. You are asked to experiment with a limited number of variations.

For selective search, experiment with three strategies: color only, texture only and all similarities. You may restrict experiments to use only the RGB space. To segment with strategy using color similarity, for example, do the following:

1) Create SS strategy object by calling createSelectiveSearchSegmentationStrategyColor(),
2) Create SS object by calling createSelectiveSearchSegmentation()
   and use addStrategy() in SS object to add strategies.

To feed an image into SS object, you can do the following:

```
ss = cv2.ximgproc.segmentation.createSelectiveSearchSegmentation()
ss.setBaseImage(img)
```

To get the output bounding boxes, you may call

```
bboxes = ss.process()
```

For Edgeboxes, experiment with different values of alpha and beta (both were briefly explained in class). Leave other parameters with default values.
Edgeboxes need an initial set of edges for which you can do the following:

1) Create edge detector:
   Model can be downloaded from here:
   https://github.com/opencv/opencv_extra/blob/master/testdata/cv/ximgproc/model.yml.gz

```
edge_detection = cv.ximgproc.createStructuredEdgeDetection(model)
```

```
    # get the edges
    edges = edge_detection.detectEdges(…)
    # create an orientation map
    orimap = edge_detection.computeOrientation(…)
    # suprress edges
    edges = edge_detection.edgesNms(…)
```

2) Create edge box:

```
    edge_boxes = cv.ximgproc.createEdgeBoxes()
    boxes, scores = edge_boxes.getBoundingBoxes(…)
```

NOTE:- (…) means function takes some parameter/parameters

Results:

Apply the two algorithms to the provided data (details of data are given below) and experiment with various settings of parameters as described above. Display results on all images for each method with just one setting of parameters. Show top scoring 100 proposal boxes for each method; selective search does not compute an explicit score so you may just display the first 100. Also highlight the boxes that have IoU > 0.5 with GT (ground truth is given with images, see below); you can do this by showing these boxes, and the GT, in different colors or on a separate image for clarity.

Compute the "recall" of ground truth (GT) bounding boxes by computing the fraction of GT boxes that are overlapped with at least one proposal box with Intersection over Union (IoU) > 0.5. Also display the proposal boxes that have this property, along with the corresponding GT, on each image. IoU definition, and an illustration were provided in class. It may be helpful to note that the area of the union area can be computed as the combined area of the two boxes minus the intersection area.

Image Data:

The image data is in HW2_VOC_Subset.zip folder in the DEN class page. Four color images are in "JPEGImages" folder; their corresponding ground truth bounding boxes are in the "Annotations" folder.

An example of the bounding box coordinates (saved in .xml file) is in the following:

```
<object>
    <name>aeroplane</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>68</xmin>
      <ymin>76</ymin>
      <xmax>426</xmax>
      <ymax>209</ymax>
    </bndbox>
  </object>
```

where the object label is “aeroplane”, and the bounding box is described in the format of (xmin, ymin, xmax, ymax) = (68, 76, 426, 209), Namely, the left-top and right-bottom points of the box. You can visualize it by using cv2.rectangle function.

What to Submit?

You should submit the following:

1. A brief description of the programs you write (include the source listing). Include comments in your code.

2. Test results on the given images with the variations suggested in the problem statement.

3. An analysis of your test results including where the methods work well and where they fail. Base your conclusions on your test results rather than on the observations made by the instructor or the authors of the books and papers.