HOMEWORK ASSIGNMENT #2
DUE: 5 PM, Tuesday, September 25, 2018
CSCI 677: Advanced Computer Vision, Prof. Nevatia
Fall Semester, 2018

This is a programming assignment. You are asked to experiment with two methods implemented in the OpenCV library. One is the Mean Shift based segmentor; another is the selective search proposal generator. Details of the two are given below.

Problem 1:

Implement and test a *Mean Shift Segmentor* program. The Mean shift algorithm is given by the function **PyrMeanShiftFiltering**. Following link provides the declaration of the function in OpenCV documentation:

https://docs.opencv.org/3.4.1/d4/d86/group__imgproc__filter.html#ga9fabdce9543bd602445f5db3827e4cc0

Note 1: Use LAB color space for this problem.

Note 2: You only need to use a level-1 pyramid for this assignment with different spatial window radius and different color window radius values.

Write a program to invoke the functions given above, and to display the results. Apply the code to the given image data. Vary the key parameters involved in the algorithm to see the effects on the segmentation results. Also provide a qualitative comparison between the outputs of the algorithm and the provided ground truth. Note that the ground truth does not provide segmentation of the background but multiple regions may exist in the background.

Problem 2:

Implement and test a *selective search* program. Following link provides the declaration of the function in OpenCV documentation:

https://docs.opencv.org/3.4.2/d5/df0/group__ximgproc__segmentation.html

OpenCV provides "strategies" to change similarity measures in Selective Search (SS).
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()
```

Draw the output bounding boxes on the input image and show this result in your report.

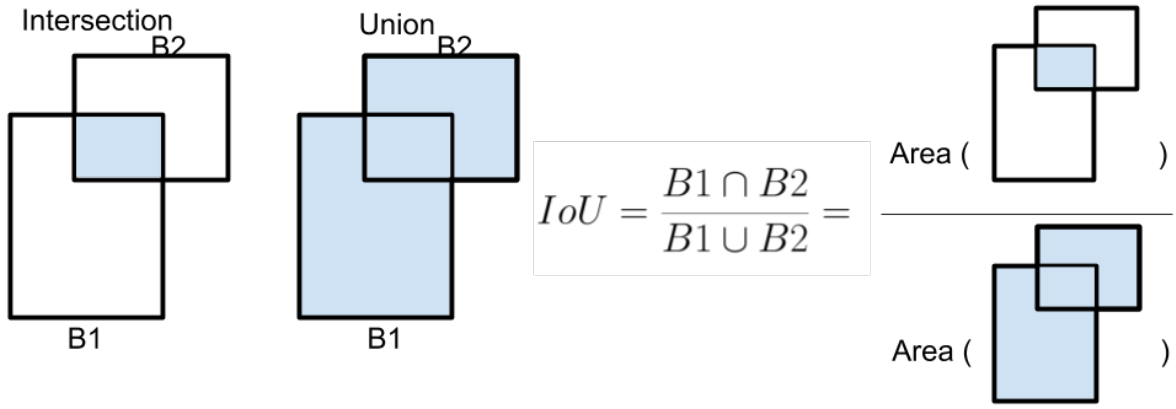
The visualized result will be as shown below. Green rectangles are bounding box proposals provided by SS.

Detailed documentation for using Selective Search in OpenCV can be found here:
https://docs.opencv.org/3.4.2/dc/d7a/classcv_1_1ximgproc_1_1segmentation_1_1SelectiveSearchSegmentationStrategy.html
https://docs.opencv.org/3.4.2/d6/d6d/classcv_1_1ximgproc_1_1segmentation_1_1SelectiveSearchSegmentation.html

Write a program to invoke the functions given above, and to display the results. Apply the code to the given image data. Limit the number of proposals displayed to 100. Experiment with two strategies: color only and all similarities. You may restrict experiments to use only the RGB space.

To quantitatively evaluate the results, compute the “recall” of ground truth (GT) bounding boxes by computing what fraction of GT boxes are overlapped with at least one proposal box with Intersection over Union (IoU) > 0.5. Display the proposal boxes that have this property, either by using a different color on in a separate image.

For your convenience, intersection over union is visualized by an example below:



Note that the union area can be computed as combined area of the two boxes minus the intersection area.

Image Data:

The image data is in HW2_ImageData.zip folder in the DEN class page. Four color images are in “Images” folder, their corresponding ground truth segmentation images are in “Segmentation_groundtruths” folder (only segmentation of objects is given), and the ground truth bounding boxes are in “boundingbox_groudntruths” 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 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 statements. For problem 1, one can easily create hundreds of results by varying parameters; just include a few that you think illustrate the key differences.

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.