ESE650 Project 1: Color Segmentation

Due Date: **1/25/2018 at 1:20pm** on Canvas, and in class

In this project, you will train a model to detect barrels in images and find the relative world coordinates of the barrel from images. To do this, you will need to implement algorithms that learn the color model, segment the target color, and finally localize the target object from images. Given a set of training images, you will hand-label some training examples. From these training examples, you will build a color classifier and red barrel detector. You will then use your algorithms to mark the center of the detected barrel and display the distance to the barrel on new test images.

Training Data Download: Now available at https://upenn.box.com/v/ese650Proj1-train
Test Data Release: 1/25/2018 1:20pm, https://upenn.box.com/v/ese650Proj1-test
Upload: on Canvas **(Late submission is not allowed!)**

(1) Code (due 1/25/2018 1:20pm, <pennkeyID>_project1.zip)
: Do not include image data.
: Include a README file with clear instructions for running it
: Include trained models for the test

(2) Write-up (due 1/25/2018 11:59pm, <pennkeyID>_project1.pdf)
: Write the summary of your approach, result, and discussion.
: Make sure your result includes (a) the bounding box of the barrel, (b) the center coordinate of the barrel, (c) the distance estimate to the barrel for each test image.

Grading: Rubrics can be found on the Canvas assignment page.

Instructions and Tips

1. Hand-label appropriate regions in the training images with discrete color labels. For this project, we will be especially interested in regions containing the red barrel. If you are more ambitious, you could try to implement more automated ways of labeling images - perhaps by an unsupervised image segmentation, or an adaptive region flooding algorithm.

   Tips: You may or may not use RGB. There are useful built-in functions if you want to use a different color space. Lighting invariance will be an issue, so you should think carefully about the best color space to use, and perhaps some low-level adaptation on the image.

2. Use a learning algorithm to partition the color space into appropriate class color regions. You should first implement the approach discussed in class, but you can also try other machine learning approaches, i.e. decision trees, support vector machines, etc. You need to make your algorithm so that it is able to robustly generalize to new images. To do this, you should use cross validation (hold out some of the training images to test this). This will allow you to compare different parameters for the probabilistic models and different color space representations.

3. Once the red barrel regions are identified, you can use shape statistics and other higher-level features to decide where the barrel is located in the images. The filename gives [the number of meters away from the barrel].[no meaning].png. Use your algorithms to label and identify the coordinates of the centroid of the barrel in a new test image. You should also compute an estimate of the distance to the barrel. You'll be expected to quickly be able to classify and display your results on a new set of test images. Write a test script to display your result as follows:

```
import cv2, os
folder = "Test_Set"
for filename in os.listdir(folder):
        # read one test image
        img = cv2.imread(os.path.join(folder,filename))
        # Your computations here!
        x, y, d = myAlgorithm(img)
        # Display results:
        # (1) Segmented image
        # (2) Barrel bounding box
        # (3) Distance of barrel
        # You may also want to plot and display other diagnostic information
        cv2.waitKey(0)
        cv2.destroyAllWindows()
```

4. During the presentation in class, you are expected to bring your own laptop or use the classroom computer. The projector has a HDMI port and you may need a HDMI adaptor for your laptop. You will be asked to run your code on the test set images which will be released both online and on a USB flash disk prior to the presentations.

5. Your write-up upload should include your test result in the following manner:

   ImageNo = [01], CentroidX = 662.0208, CentroidY = 506.7571, Distance = 5.4418
   ImageNo = [02], CentroidX = 507.0605, CentroidY = 826.7004, Distance = 2.3840
   ….

6. You may use some useful python functions for this project:
   - hand-labeling: roipoly: https://github.com/jdoepfert/roipoly.py
   - color conversion: cvtColor: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html
   - region analysis: regionprops: http://scikit-image.org/docs/dev/api/skimage.measure.html

   However, please do not use any built-in function that would be the core part of this project. If you are not sure, then ask TAs. Examples of allowed code:

```
import cv2
img2 = cv2.cvtColor(img, cv2.COLOR_BGR2YCR_CB)

from skimage import data, util
from skimage.measure import label, regionprops
img = util.img_as_ubyte(data.coins()) > 110
label_img = label(img, connectivity=img.ndim)
props = skimage.measure.regionprops(label_img)
```

   - hand-labeling: roipoly
   - color conversion: rgb2ycbcr
   - region analysis: regionprops