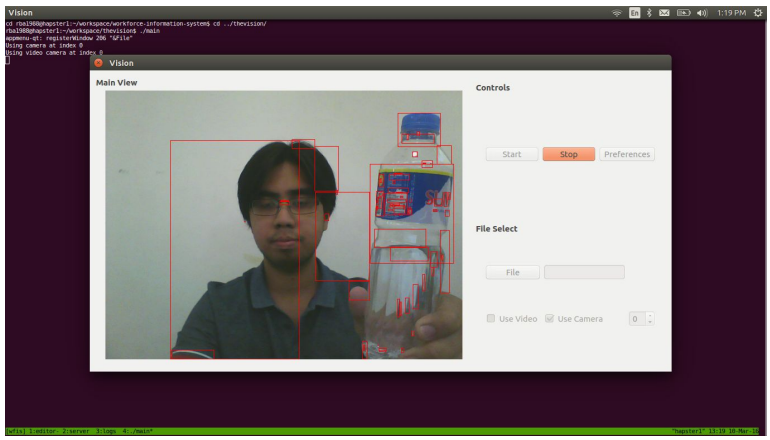
CS 129.18 Lab 2

Objectives:

1. To apply pattern recognition algorithms and methodologies in solving a computer vision problem
2. To evaluate the performance of a classification algorithm

Instructions:

1. Install the OpenCV library in your computers and make sure you are able to do the following:
    a. Read an image file using the **`imread()`** and store it as a Mat object.
    b. Perform the **`Canny()`** function that creates another Mat object representing the edges found in the image. **`Canny()`** has two parameters called the low and high threshold. You may hard code these values or opt to automatically generate these values as found in this article (http://www.pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/)
    c. Perform the **`findContours()`** that looks for contours and returns an array of contours for referencing.
    d. Call the **`rectangle()`** function to draw a rectangle around the contours.
    e. Create a **`Rect()`** object that defines a rectangle given 4 points (there is also a data type called Point in OpenCV but you can opt to pass the starting x, y and ending x, y in the Rect function).
    f. Call the **`Mat(r)`** function that returns a new Mat from the original Mat that's bound by r which is an instance of Rect. This essentially creates a region of interest from an original Mat. For more information, refer to this link: https://jayrambhia.wordpress.com/2012/09/20/roi-bounding-box-selection-of-mat-images-in-opencv/
2. Given an image and the function mentioned above, you should be able to produce something like this:



3. The rectangles displayed here represents the contours found in an image. Our objective is to be able to say that the region of interest represented by those rectangles are “objects” or “non-objects”. For a given image, you should be able to store several Mat instances representing these regions of interests (i.e. an array of Mat objects).
4. The array of Mat objects are essentially your data set. You can write a Mat to an image file by using the **`imwrite("somename.jpg")`** function. For each Mat (region of interest Mat) extracted from an image, determine manually what your group thinks is an “object” or “non-object”. Build your set of “object” and “non-object” data. This will serve as your ground truth (for a high likelihood and better classification results, it would be nice if you are able to extract 60 positive and 60 negative samples).
5. Once you have your ground truth, it is time to create feature vectors for both positive and negative images using the BING features. The methodology is as follows:
    a. For each Mat, create an image gradient intensity version of the Mat by using the **`Sobel()`** operator. This new Mat will have the same size as the original Mat but its pixel values will contain gradient intensity values (also from 0 to 255). Sample code can be found here: http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html
    b. For each gradient Mat, resize the image to an 8x8 image. You may automate this by using the resize function of OpenCV. This 8x8 image can already be considered as your feature vector for BING. The feature vector for each Mat in your data set should be 64 dimensions long.
    c. Store the data in a csv file in the same format as your previous lab wherein the last column is the label for the sample (1 for object and 0 for non-object).
    d. To make things easier, I uploaded the code for producing the BING feature vector in moodle (ng.cpp). You may use it or port your own version to be integrated in your workflow.
6. Perform naive bayes to train a model with the corresponding variance and mean parameters for objects and non-objects.
7. Visually display your result by taking a new image and drawing blue rectangles for regions that the system thinks are objects and red rectangles for regions that the system thinks are non-objects.
8. Answer the following questions:
    a. How would you determine the accuracy of your Naive Bayes model in this case? Use a confusion matrix as your final result but also explain your methodology.
    b. Will Naive Bayes work for this case? What improvements might you suggest to improve this classification system?