EL2310 C++ Assignment

Hakan Karaoguz

For this assignment, you will use a provided dataset, a class for reading the dataset, a sample main file and also an open-source image processing library called OpenCV. You can use any functionality of OpenCV for completing this assignment. You can check the website
http://docs.opencv.org/2.4/doc/tutorials/tutorials.html for tutorials.

- Use the `Dataset` class to read the provided dataset. Please check the header file to understand how to use it.
- Create an abstract class `ImageProcessing` with necessary header and source files. This class will serve as an interface to other derived classes. Thus it should have a pure virtual function called `process()` for executing image processing tasks.
- Create a class `HistogramProcessor` derived from `ImageProcessing` to process the image color histogram. Please use the supplied code in *CalculateHistogram.txt* for calculating the color histogram of an image. This class should need to implement the virtual function `process()` to process the histogram. You should create a constructor that accepts a `Mat` variable for image and an `int` for the number of bins (should be in the interval [10-180]) . Also add the necessary get function and private member variables to hold the inputted image, number of bins and the processed histogram to be returned.
- Create a class `HistogramViewer` derived from `ImageProcessing` to view the image color histograms. Please use the supplied code in *visualizeHistogram.txt* for viewing the histogram. This class should need to implement the virtual function `process()` to view the histogram. This class should create an instance of `HistogramProcessor`, process the histogram and use the display function to display it. An example output is given in Fig. 1.
- Create a derived `SVMClassification` class from `ImageProcessing` interface class with necessary header and source files. This class should implement its own virtual `process` function to perform SVM classification. The tasks of this class are:
  - This class should be able to access to an instance of `ImageDataset` class for reading data from the provided dataset.
  - In order to train the SVM classifier, you should use the training images of the dataset.
  - For each training image, it should perform feature extraction using color histograms. The resulting histograms should be stored with

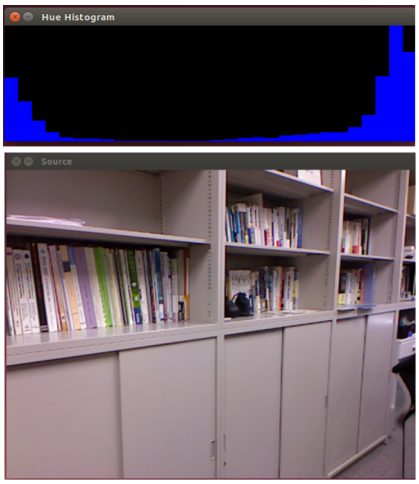

Figure 1: HistogramViewer example output

correct class label for training. Ex: If the office class has a label number of 0 and you extract the color histogram for an office training image then you should also store the label number for that image index as 0. At the end you should have a MxN matrix of histograms with M equal to the number of all training images and N is equal to the number of histogram bins. You should also have another vector Mx1 that stores the class label for each histogram row. *Hint*: You should create Mat objects for storing histograms and labels so that you can input them to the SVM.

- After feature extraction is done, the SVM classifier should be trained. You can use the supplied code in *trainSVM.txt* to train the SVM.

- After the SVM training is done, you should employ the virtual `process` function to classify the test data. You should store the number of correct and incorrect classification results in a file called *results.txt*. You can calculate the classification accuracy $|A_i|$ for each class as

$$A_i = \frac{|TP_i|}{|D_i|} \times 100$$

where $|TP_i|$ is the number of correct classifications for class i and $|D_i|$ is the number of total test samples for the same class.

- Vary the number of histogram bins you use for training the SVM. Try with 30, 90, 180 bins and see if it affects the success rate? You can draw a figure in Matlab to show the relation between the accuracy of the classification and the number of bins.

- Vary the number of training data you use for each class and see if there are any changes at the results. Try with the 30%, 50%, 70% and 100% of the training data you have. You can draw a figure in Matlab with x axis being the number of training data and the y axis being the classification accuracy.