Michael Pedersen
10/27/16
Computer Vision

Homework #3: Scene Recognition

Accuracy for base algorithm: 0.16 or 16%
Accuracy for improved algorithm: 0.2283 or 22.83%

The base algorithm used for this assignment uses a two step approach, one for training and another for testing. When initially training I first collect the feature vectors of every image using the SURF extraction set to return a 128 dimension vector for each feature. After collecting the feature vectors of every image from the training set (1800 images) I then cluster using the k-means algorithm. When K is set to 200 the process can take a little less than an hour. The center of each cluster represents a visual word in our vocabulary. This clustering allows every image to be represented from this shared vocabulary. Next I count the number of features from each image that appear on each cluster. Thus, one row of my code book represents a histogram of features that appear in a given column(cluster number). I also save a matrix relating the ID of each training image to the label number that image originated. Each of these matricide are saved to the model.mat file.

During testing I first extract the same SURF features from each testing image. Then for each image I find the closest word (k-means cluster). I create a bag of words histogram, similar to the prices used during training) to get a histogram of how many features map into each word cluster. With the bag of words completed I find the similarity between this test image's bag of words and the bag of words histogram of each training image. In the base implementation I just took the most similar without any additional steps. Finally, I construct the matrix of predicted labels for each testing image using the process above and mapping each testing image number (row number) with the top rated category label. Overall the basic implementation got 16% accuracy on the validation set.

The base algorithm performed quite well even when implemented in my naive non-vectorized form. In order to improve the accuracy more I added a few features such as a way to take more than one nearest neighbor while comparing a test image's bag of words to the trained bag of words. By taking the mode of the 10 most similar images I could then take the most frequently occurring word (cluster) with much more confidence. In addition, I looked deeper into the documentation for the detectSURFFeatures and found through some trial and error that changing the number of scale levels per octave from the default of 4 to 10 improved the accuracy by a few full percentage points. The improved version gets 22.83% on the validation set.

Throughout this process I did have a few issues. The main one is that I was unsure of how the code was organized, originally I assumed that the feature extraction function would be able to extract the features and generate a bag of words standalone - without any additional information from the training data. This assumption along with others about how to pass data between the given template code confused me to no end. Yet again I learn hindsight is 20/20. In

addition, while scaffolding my code I used the knnsearch function as a temporary filler so that I could focus on the bag of word implementation. This proved to be an issue when I learned that the knnsearch function didn't return accurate values for the bag of words comparison. Up until I realized this mistake I had been stuck with 3% accuracy (random chance). The second I replaced it with my implementation the accuracy shot up to expected values, ~ 16%.