Assignment 3 – Bag of Words Categorical Matching

My method follows the pipeline described in the assignment description. I get the 128 dimensional features from each image using Matlab’s ‘detectSURFFeatures’ and ‘extractFeatures’ functions. From that point on, I implemented k-means clustering, k-nearest neighbors, tf-idf weighting, and RGB histogram comparison. I will discuss the baseline process first and save the extra steps for last. I should say early on that anytime I am comparing two models such as feature descriptors or RGB histograms, I am simply subtracting them elementwise and then summing their absolute values. Euclidean distance and cosine similarity were attempted but those methods took way longer to run. I also did not see much improvement using them.

**Training/K-means clustering**

The reason I implemented this myself instead of using Matlab’s k-means function is because I needed to preserve the image identity/label of the feature descriptors. I took all the features extracted from every image and did k-means clustering with 800 centroids(words). This number was the result of me trying different values until I noticed diminishing returns and slower runtime. The centroids after convergence or the max iteration, 25, are the words used to create the shared vocabulary codebook. For each training image, they are assigned a representation that is a histogram of all the possible words in the codebook.

**K-nearest neighbors**

After building the codebook with k-means clustering on the training data, I extract the features of the query images and generate a bag of words representation in the same histogram form as the training images have. From this point, it’s just a matter of comparing each query image’s bag of words with the training data and ranking them in terms of difference. The k nearest neighbors, which I set to 30 through manual tuning, each use their label to vote on 1 of the 30 categories. The chosen guess is the most common label of the nearest 30 training images. At this point, using the above steps, my accuracy was 20%.

**Extra Step: tf-idf weighting**

To further improve the accuracy of the algorithm, I applied tf-idf weighting. From what I read, there are different ways to determine how exactly you classify the tf and idf weights. I played around with some of them, and ended up choosing the simplest I could find. My tf weights for each word in each document were calculated by taking the number of times the word occurs divided by the total number of words in the image. The idf weight was determined for each word by taking the log of the total number of occurrences of the word throughout all documents divided by the number of documents it appears in. At this point, my accuracy was at 26%.

**Extra Step: RGB Histogram comparison**

Color is a great way of determining the category where many of the query images belong. I computed RGB histograms for each image, using 32 bins per channel to reduce processing time. Much like how I determined the 30 nearest neighbors in terms of bag of words comparison, I also computed the 30 nearest RGB histograms to each query image. These were interweaved into a structure of 60 elements, with the first element being the closest image in terms of bag of words, and the second element being the nearest RGB histogram, following this alternating pattern throughout. The most common label of these 60 images is chosen as the query image’s category. Accuracy at this point is 42%. This proved to be quite an improvement over only using the bag of words comparison.

**Conclusion**

As always, my algorithm could have been improved further as some of the manual parameters and distance measurement could have been optimized more. I cared about processing time when designing this as I know there are over 40 students in the course. My coding practices here are not the best, mainly because I was not allowed to break the skeleton code interface to do things such as RGB histogram comparison, which is fair since that is not what the assignment was built for. Overall, I enjoyed the assignment and learned a lot from it.