The first part of the code is mostly to check if the VL_SIFT library works as expected and also has the command to run the setup of the library which is needed before using functions of VL_SIFT.

1. Codebook generation

In the second section of code I use my function descriptorsImageFolder to load images from the 101_ObjectCategories dataset. The resulting descriptors are stored in two matrices: trainDescriptors and testDescriptors.

The function descriptorsImageFolder has the following input parameters: folderName which is the path to the folder, tag which is the index for the category of images in that folder, the two matrices trainDescriptors and testDescriptors in which the new descriptors are added, nrTrain and nrTest are the numbers of images from which we read trainDescriptors or testDescriptors. The function implementation is quite ugly and has much redundant code but it works as expected. At the end descriptors having 128 columns and 1 row each are stored in the corresponding matrices. At the end the trainDescriptors and testDescriptors matrices have 129 columns because I also add a column with the index of images category; I thought it might be useful later. Right now I have loaded descriptors from each 30 train and 10 test images of 12 categories, mostly for purpose of testing the implementation. We can add more later.

After this step, I have code to also save the descriptors in 2 local csv files for later use. In the next section, I load the descriptors from those two files and apply kmeans clustering. Since the Matlab kmeans implementation takes ages to run for about 80000 descriptors and k=300 I have used the vl_kmeans which has the algorithm ANN implemented and this allows to obtain clustering for large amounts of data in a shorter time, but I guess it relies more on approximations and it is not same accurate as normal algorithm but it is a lot faster. For k=128 I need about 5 minutes to execute the function on my machine. The vl_kmeans expects the descriptors to be on columns so I have to give as input the inverse of trainDescriptors matrix. The resulting parmaters are: C which are the cluster centers and A which are the cluster indices for each descriptor. In the next step I use knn to assign each descriptors to the closest centroid and store the result in closestCentroid. I think this result should be used for retrieving the images but I do not understand how to do it.

2. Indexing

After making the clustering, I create the database pretty much as described in the assignment text. The database is actually an array of struct with the following fields: filename, category, set, BagOfWords which is the word histrogram. The structs are created inside the folderToDatabase function which takes

as parameters the database array to which new structs are added, foldeName, centroids that were obtained previously from vl_kmeans, the category name and the number of train and test images to be loaded from the folder.

The implementation of this function is also quite ugly but it works like this: for each descriptor of the image it searches for the closest centroid using knn and it creates a histogram with 128 bins (one for each cluster that was obtained previously) where I count how many descriptors are assigned to each centroid. At the end I store the information in a struct as described previously.

3. Retrieving

I did not have much time to also implement this but here we have to use the resulting database to retrieve the images category. I have tried fast to take the image of a barrel, extract the histogram from its struct and compute Euclidean distance to each other word histogram from every entry of the database. At the end I sorted the results according to the distance. The results look bad but I think I did something wrong at this code or maybe there is also something wrong before this stage. I did not understand very well how to make this retrieving using the algorithms described in the lecture slides but I think it is just to compute a value using the values from each word histogram and use the smallest values resulting to assign the image to a category, without using any learning as I saw at other implementations of Content Based Image Retrieval that use SVMs or other algorithms that actually need training.