CMPT 412

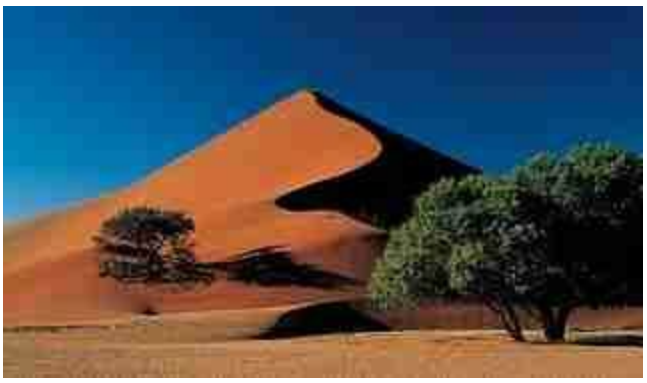
Project 4

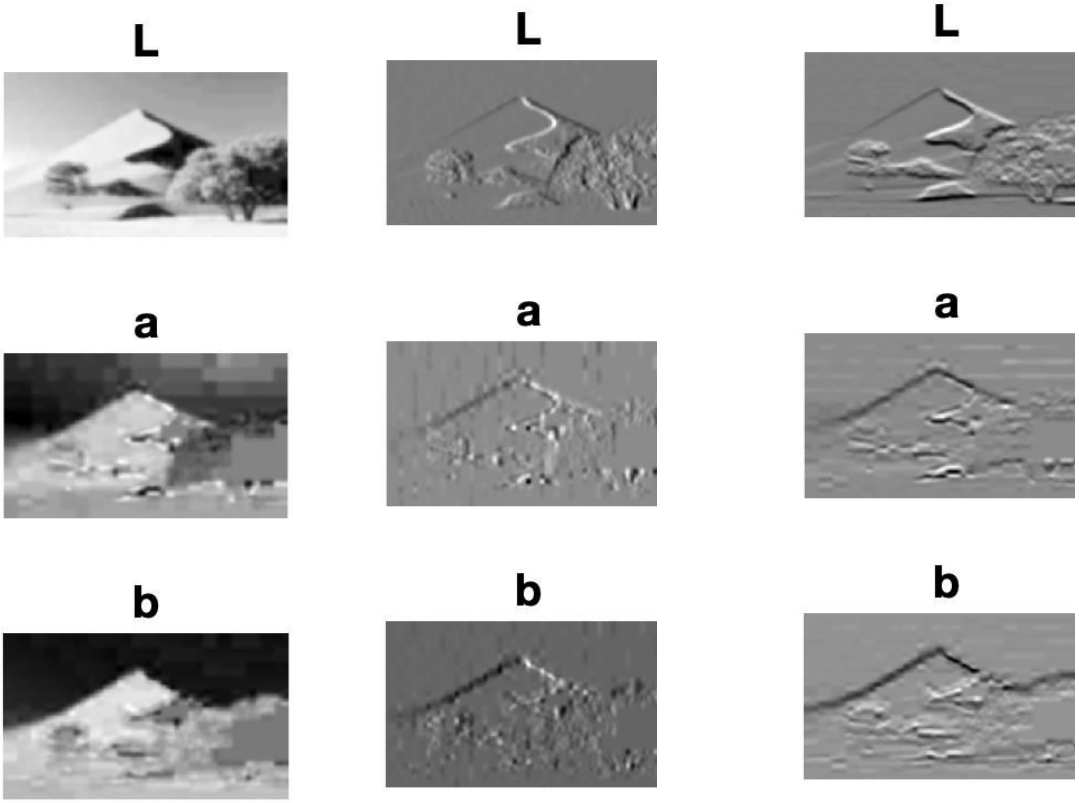
Scene Recognition with Bag of Words

Q1.1 Extract Filter Responses

In this section we are going to extract the filter responses for a given image . This is done after converting image from RGB to Lab space and then applying the filter to each L, a, and b layer. We use Lab colour space for this task as it allows for better feature extraction, which ultimately dictates the accuracy of the system. The better the harris features, the better the accuracy at the end. It is defined as one a,b plane on which we define intensities at every point according to an L parameter. This allows for a uniform distribution of the

colours on the same plane with just different intensities as opposed to having one plane for each colour channel as is the case in RGB. Below are 3 examples of 3 different filters applied to the same image. Each image is itself 3 channels L, a, and b. Therefore we have 9 total images. The original image is the one shown in RGB below.

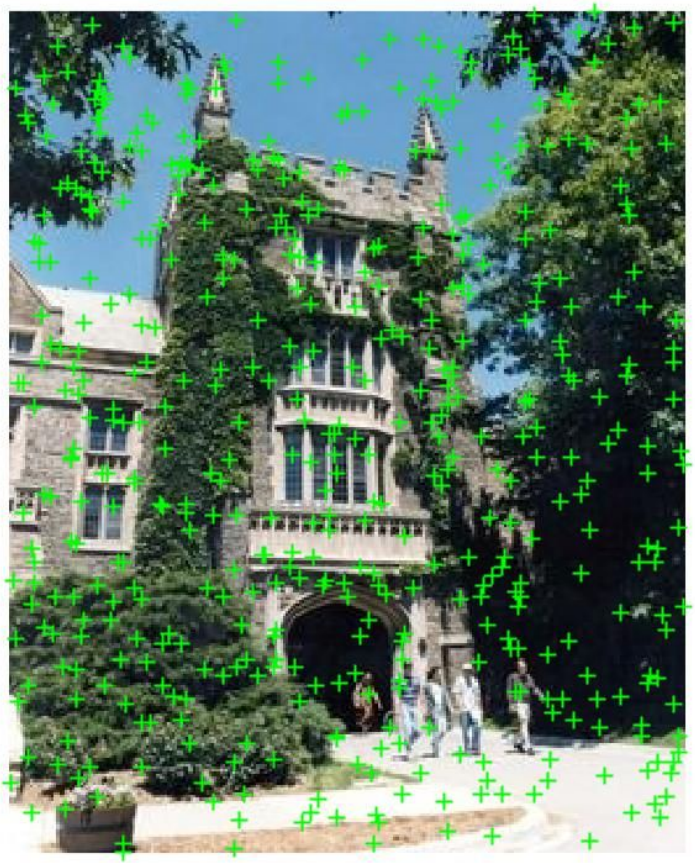




Some artifacts which are obvious right away, are the horizontal or vertical blocks. These are due to the fact that we are using the gradients of the gaussian which respond to vertical or horizontal edges. Because of the low quality of the images and sudden change in intensities from one pixel to the next, these artifacts become apparent.

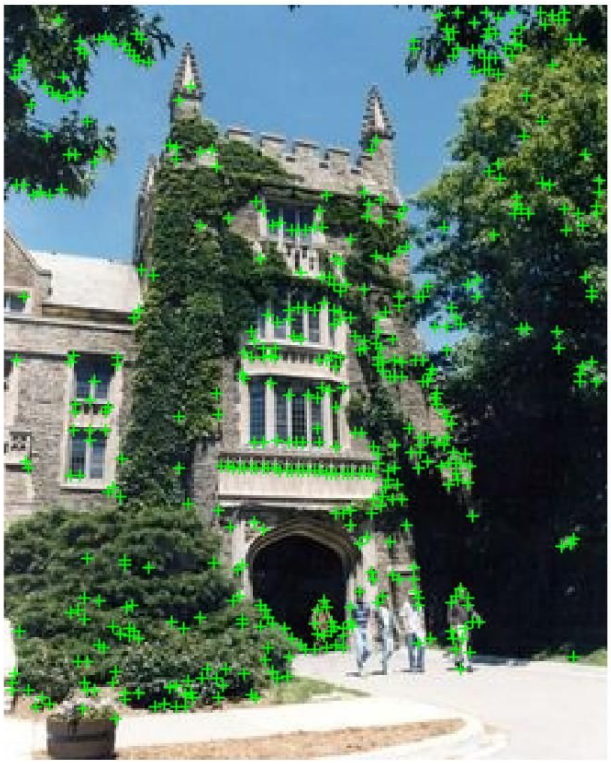
Q1.2 Collect Sample of Points

In this section we attempt to extract both random as well as harris features. A sample random point extraction is shown below. As we can see, the randomness will not give us any meaningful features. This will be more evident later when we show accuracies.



However, using harris corner detector algorithm will result in “meaningful” feature points. The harris algorithm uses the x and y image gradients which have been smoothed using a gaussian blur technique in advance. The borders of the gradients are also set to 0 intensity to prevent any feature extraction from the borders. After the harris points are extracted, we do NMS to discard points that are too close and add noise to our features. We

ultimately want the best harris responses, so we suppress the low intensities locally using matlab's imregionalmax function. If at least alpha points still exist after NMS, then we grab the top alpha points, otherwise we grab all the existing suppressed points. This guarantees that are features are high in quality. Below are examples of the harris detector results on 3 different images all from the campus category.



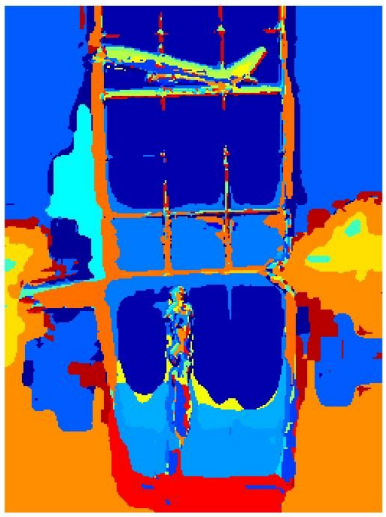
Q1.3 Compute Dictionary of Visual Words

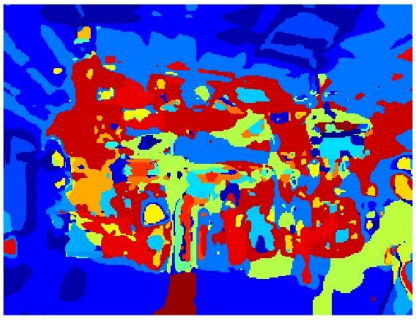
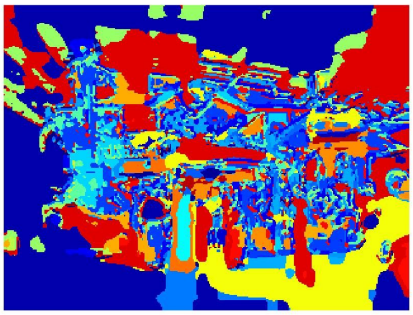
This section is about creating the visual dictionary using the extracted features and matlab's kmeans function. I originally had more dictionaries to submit, but to save space, I settled on the one with the best result, which would be one with alpha set to 100 and k set to 100 for both random and harris method. The dictionaries are in the matlab directory in the submitted zip folder.

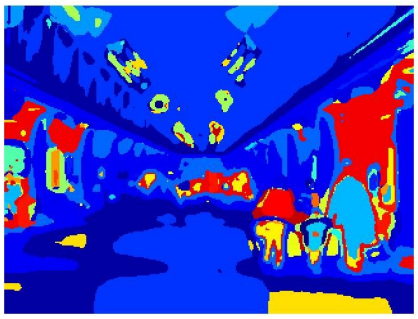
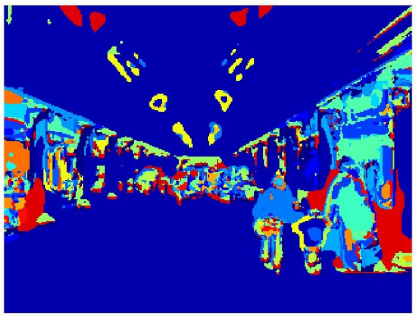
Q2.1 Convert Image to Word Map

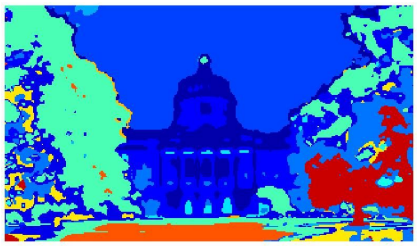
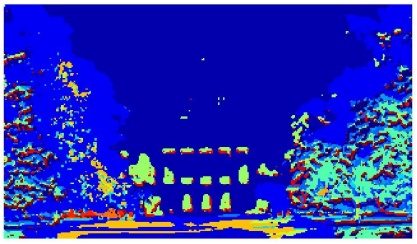
This section is about converting images to their respective word maps. Essentially every image turns into a bunch of labels for each of its pixels. Then we are able to visualize this by converting the word map to rgb using label2rgb in matlab. Also when computing each pixel's label, we use the pdist2 function. This function takes the dictionary as well as the response. However, the response is a 3D matrix and in order to make the columns match the columns of the dictionary, matlab's reshape function is used to turn the 3D matrix into 2D with 60 columns to match the dictionary. Below are some results from visualizing the resulting word maps of some images using both random and harris method. The left image is the original, middle contains the word map from random, and right is from harris.

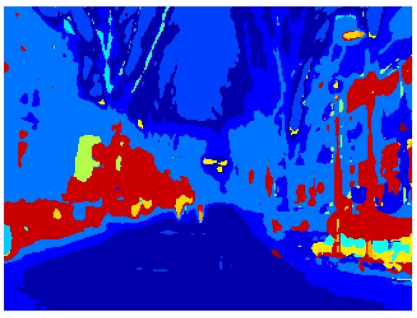
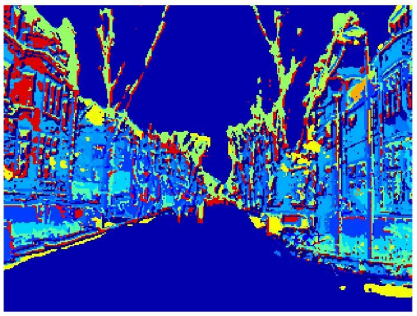


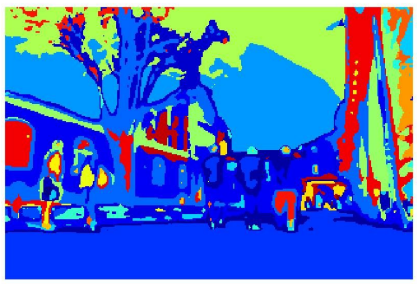
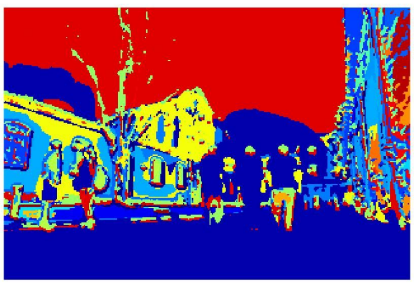


At first glance, it might not be entirely obvious which dictionary is doing a better job at creating these word maps. However, upon close inspection, we can see that harris is actually partitioning the image in a meaningful way. Though, random is also partitioning, there is less detail in its partitioning. For example, the sky is sometimes partitioned as two different entities, or trees are blended in with the building or the background. People aren't as evident in the word map either. Whereas, Harris does a good job in the areas mentioned. So, Harris dictionary is most likely going to be performing down the pipeline as we will see in a bit. Also, we can see that the areas that are similar are uniformly partitioned in most cases. Which, we can argue that the meaning is being captured in some sense. For example trees in the same image are similarly coloured. Same for people, buildings, the sky, etc. So in a way the meaning is contained in the word map, especially for the ones created using the Harris dictionary.

Q2.2 Get Image Features

This part is very trivial and only captures all the labeling in the word maps into histograms or the "bag of words". The histograms are then L1 normalized to adjust height.

Q3.1 Image Feature Distance

In this part, two different distance methods are implemented, namely chi squared distance and euclidean. The chi squared implementation follows the formula provided in course notes directly, while the euclidean method uses the pdist2 matlab function.

Q3.2 Evaluate Recognition System - NN and kNN

NN method

In this part, we use two different methods of evaluating our system. The first method is going to determine what we ultimately use kNN on. The winner in NN is going to be the harris method along with the chi squared metric. This is evident from the output of the evaluateRecognitionSystem_NN script. The output is given below. It contains the accuracies for 4 different combinations of feature extraction method and metric used as well as the confusion matrix for each. The output below is directly copied over from matlab.

Accuracy of harris using metric chi2 is: 0.481250 | Confusion matrix below:

12	0	4	0	0	0	1	3
6	9	3	0	1	1	0	0
2	2	12	1	3	0	0	0
5	0	2	8	1	1	3	0
2	3	2	0	11	0	2	0
2	0	3	7	1	4	1	2
3	1	1	4	5	1	5	0
2	0	0	1	0	0	1	16

Accuracy of harris using metric euclidean is: 0.437500 | Confusion matrix below:

7	4	3	2	0	0	2	2
4	13	0	0	1	0	2	0
3	2	9	2	2	0	2	0
1	2	5	10	0	0	2	0
2	2	4	0	8	0	4	0
1	3	0	5	1	5	4	1
2	1	3	4	3	1	4	2
2	1	0	3	0	0	0	14

Accuracy of random using metric chi2 is: 0.468750 | Confusion matrix below:

9	2	4	0	0	0	1	4
4	12	1	0	2	1	0	0
3	3	10	1	2	0	1	0
1	1	3	7	0	2	5	1
1	2	4	0	9	0	4	0
0	1	3	3	1	5	3	4
3	0	0	4	3	0	8	2
2	0	2	1	0	0	0	15

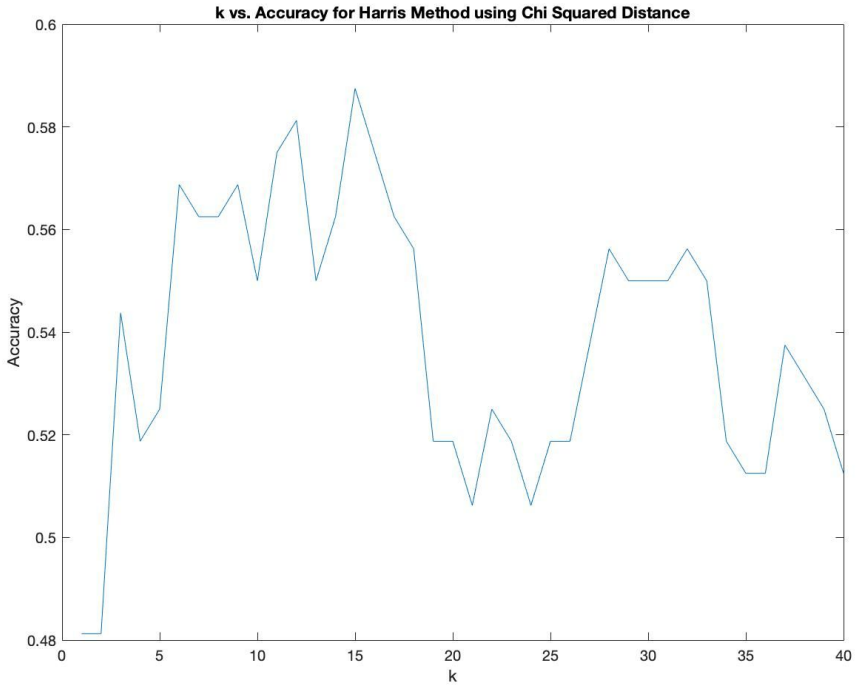
Accuracy of random using metric chi2 is: 0.468750 | Confusion matrix below:

9	2	4	0	0	0	1	4
4	12	1	0	2	1	0	0
3	3	10	1	2	0	1	0
1	1	3	7	0	2	5	1
1	2	4	0	9	0	4	0
0	1	3	3	1	5	3	4
3	0	0	4	3	0	8	2
2	0	2	1	0	0	0	15

We can see that Harris and Chi squared is the best combination with ~48% accuracy. It is not surprising that Harris is performing better. We essentially used much more meaningful and higher quality features to build our dictionary, whereas, random samples the image space randomly and could easily grab many meaningless and useless patches and build a dictionary out of them. Random performed surprisingly well, regardless. To visualize the difference, we can refer back to section 2.1 of the report and get an intuition on how different the dictionaries actually are. This is explained more thoroughly above in section 2.1. The metric, increased the accuracy of Harris by about 5 percent. In general, chi squared is used to compare distributions in statistics. Here, we are comparing histograms which are essentially distributions, so it makes sense that the chi squared method would give better results in this context as opposed to simple euclidean distance between the two histograms.

kNN method

Now we are going to use kNN on the chi squared and harris combination and determine the best k value to use in the range of 1 - 40. By plotting all the different k values the below graph can be demonstrated.



We can see that the optimum k value in this case is 15 giving an accuracy of about 59%. This is a great improvement from using k = 1 which is just the NN method which is about 10 percent lower. It is interesting how if we keep increasing k, we don't necessarily improve performance. At some point, we are taking into account neighbours which are too far away and distorting our prediction. In order to resolve ties, I essentially let matlab decide on the winner by calling the max function, which essentially returns the first max which was observed. The confusion matrix for k = 15 is copied over from matlab and can be seen below.

Maximum accuracy occurs at k=15 with accuracy of 0.587500 | Confusion matrix below:

13	1	3	0	0	0	1	2
5	12	2	0	0	0	1	0
2	3	15	0	0	0	0	0
4	1	2	7	2	0	4	0
1	2	2	0	15	0	0	0
2	3	2	5	1	6	0	1
6	0	1	0	5	0	8	0
2	0	0	0	0	0	0	18

- *Some information about the above system in case they get lost in the text or are missing:*
    - *Alpha and K were set to 100*
    - *My dictionary names have the suffix of _100.mat because initially different sized dictionaries were created so had to be named differently.*
    - *randomVision and harrisVision also have suffix 100.mat*
    - *All the word maps also have a suffix of either _harris100.mat or _random100.mat*
    - *The batchToVisualWords file was modified*