Project 4

Scene recognition with bag of words

Computer Vision - CMPT 762

3. Programming

Part 1: Build Visual Words Dictionary

Run testPart1.m to see all the results for part1.

Q1.1 Extract Filter Responses

There are 4 filters, and each one is made at 5 different scales, for a total of 20 filters. The filters are:

- The first five filters are Gaussian filters. They respond strongly to constant regions, and suppresses edges, corners and noise. They pick up local intensity information and blurred. By increasing the order of the filter, intensity information over a larger local area centered around the pixel is captured.
- The next 5 filters are LoG filters or Laplacians of Gaussian filters. Responds strongly to blobs of similar size to the filter
- Filters 11-15 capture x derivatives. They respond strongly to vertical edges. The 11th filter captures fine, very thin vertical edges while the larger filters capture vertical edges over a wider area.
- Filters 16-20 capture y derivatives. They respond strongly to horizontal edges. The 16th filter captures fine, very thin horizontal edges while the larger filters capture horizontal edges over a wider area.

Q1.1 Show an image from the data set and 3 filter responses. Explain any artifacts you may notice.

Here is the result for extractFilterResponses.m.

CIE Lab color space has three components:

- L: Lightness (Intensity).
- a: color component ranging from Green to Magenta.
- b: color component ranging from Blue to Yellow.

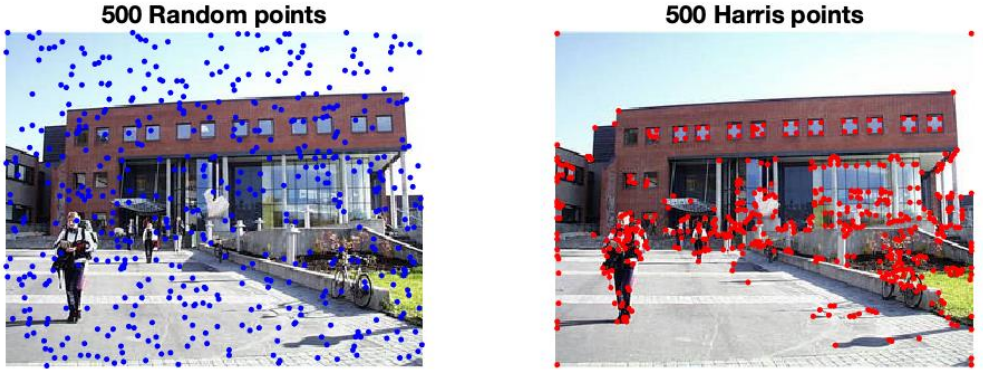
In CIELAB, the same amount of numerical change in these values corresponds to nearly the same amount of visually perceived change. In Lab color space, the L channel is independent of color information and encodes brightness only. The other two channels encode color. So, it defines

colors independent of how they are created or displayed. As a result, it is useful when images are captured using different cameras. In fact, even our images are taken by different cameras, after converting to the LAB space, we can remove that effect.

The left image is the original. The second one is the output of Gaussian filter for L channel (Gaussian blur). The third one is the output of x gradient Gaussian filter for L channel (vertical edge detection). The last one is the output of y gradient Gaussian filter for L channel (horizontal edge detection).

Q1.2 Collect sample of points from image

Here is the result for getRandomPoints.m and getHarrisPoints.m. (alpha = 500 and k = 0.05)



Here you can see the results of corner detector on 3 different images. (alpha = 150, k = 0.05)

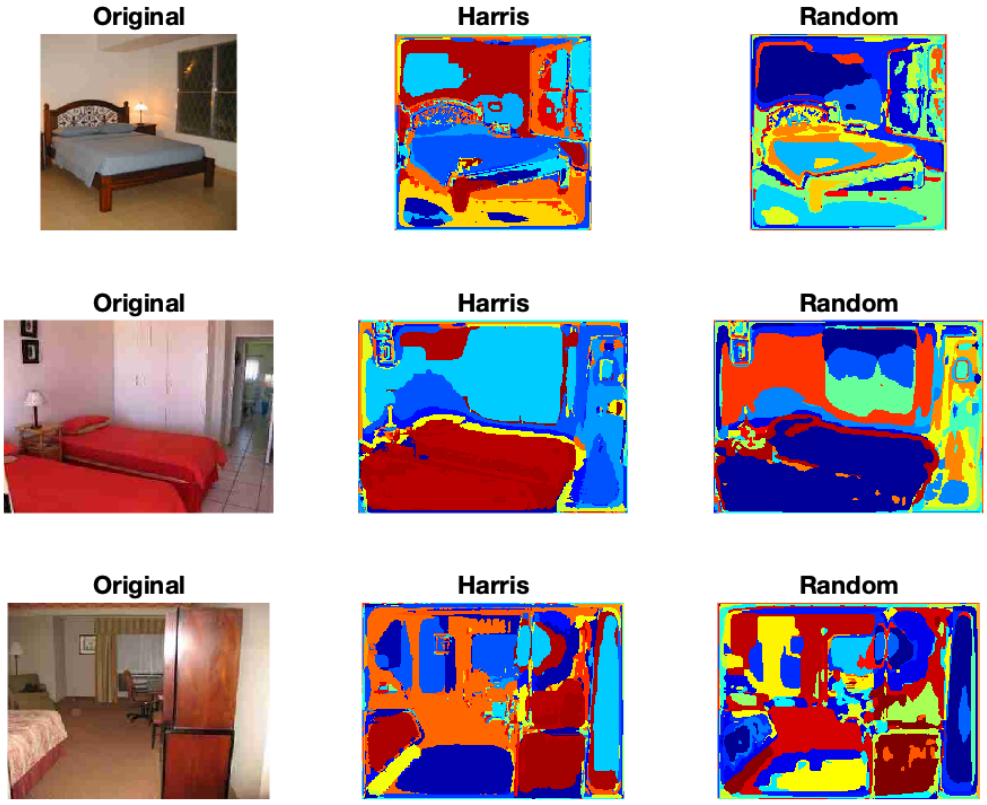
Q1.3 Compute Dictionary of Visual Words

You can run computeDictionary.m to test getDictionary.m. (alpha = 75 and K = 100)
The results (dictionaryHarris.mat and dictionaryRandom.mat) are saved in matlab directory.

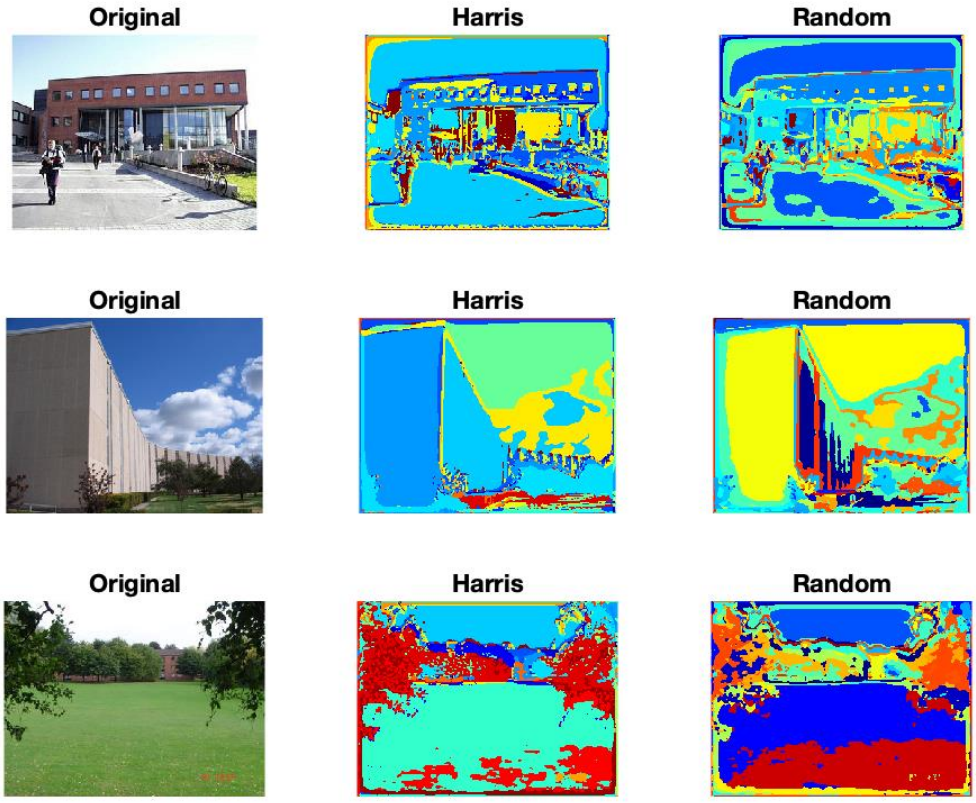
Part 2: Build Visual Scene Recognition System

Q2.1 Convert image to word map

Two sets of images from campus and bedroom are shown.



Bedroom



Campus

Are the visual words capturing semantic meanings? Which dictionary seems to be better in your opinion? Why?

Yes, the visual words capture the semantic meaning. You can see that different colors are assigned to sky, building, bed, ground and so on. So, the word map can classify these spaces as different words.Harris works better here to capture semantic meanings. Because Harris samples corners and these corners can help to define object boundaries. So corner points are better than random points in this part.

At the end, I used batchToVisualWords.m to save wordmap for every image. I saved the results for Harris with _h.mat an Random with _r.mat.

Q2.2 Get Image Features

getImageFeatures.m

Q2.3 Build Recognition System - Nearest Neighbors

Run buildRecognitionSystem.m to save visionRandom.mat and visionHarris.mat in matlab folder.

Part 3: Evaluate Visual Scene Recognition System

Q3.1 Image Feature Distance

I have implemented this function in getImageDistance.m: [dist] = getImageDistance(hist1, histSet, method)

For Chi2 distance if the denominator is equal to zero, we put zero for distance.

Q3.2 Evaluate Recognition System - NN and kNN

- Include the output of evaluateRecognitionSystem_NN.m (4 confusion matrices and accuracies).

Sampling = Random
Distance metric = Chi2
Accuracy = 0.4813
Confusion Matrix =

14	2	2	0	0	0	1	1
8	11	0	0	1	0	0	0
3	6	10	0	1	0	0	0
2	1	3	7	1	1	4	1
1	3	0	0	11	1	3	1
2	0	1	4	1	4	2	6
3	1	0	3	1	2	6	4
2	0	0	2	0	1	1	14

Sampling = Random
Distance metric = Euclidean
Accuracy = 0.4188
Confusion Matrix =

12	3	3	0	0	0	1	1
5	11	0	0	1	0	0	3
3	6	7	0	2	0	2	0
3	2	1	7	1	1	5	0
0	5	3	1	7	1	3	0
3	1	1	6	0	4	3	2
0	1	1	5	4	1	6	2
3	0	0	3	0	0	1	13

Sampling = Harris
Distance metric = Chi2
Accuracy = 0.4750
Confusion Matrix =

16	0	2	0	0	0	1	1
6	10	2	0	1	1	0	0
1	4	9	0	3	0	2	1
3	1	2	6	0	2	4	2
1	3	2	0	11	0	2	1
2	1	1	6	1	3	2	4
4	1	3	2	2	1	6	1
2	0	1	1	0	0	1	15

Sampling = Harris
Distance metric = Euclidean
Accuracy = 0.4250
Confusion Matrix =

14	0	4	0	0	0	0	2
6	6	5	1	1	1	0	0
1	4	8	2	3	0	1	1
2	2	4	6	0	1	3	2
0	1	3	2	9	1	3	1
1	3	2	6	1	3	0	4
1	1	2	6	1	1	7	1
2	0	1	2	0	0	0	15

- How do the performances of the two dictionaries compare? Is this surprising?

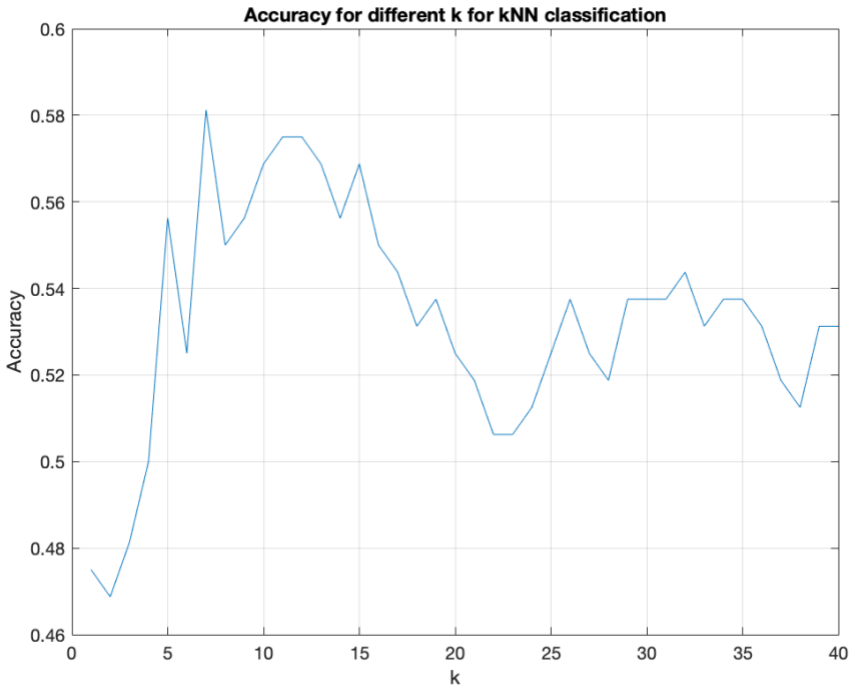
Harris sampling should perform better than random sampling. As we mentioned before, corner points are better in capturing semantic information as the sampling points are corresponded to regions with more variations in the neighbourhood. If we decrease number of sample points, we can see the effects of dictionary that Harris is better option for sampling. But this difference become less when we increase the number of sampling points since Harris points are clumped near corners but random cover larger areas and more regions.

- How do the two distance metrics affect the results? Which performed better?
- Why do you think this is?

Chi-square works better than Euclidean here for Random and Harris sampling. Because we want to compare similarity/correlation of histograms rather than absolute distance. Because Chi-square is a weighted Euclidean distance and it is less affected by extremes and noises. Chi-square is a better metric for comparing histograms.

- Also include output of evaluateRecognionSystem_kNN.m (plot and confusion matrix). Comment on the best value of k. Is larger k always better? Why or why not? How did you choose to resolve ties?

Run evaluateRecognitionSystem_kNN.m for Harris Sampling and Chi2 distance metric.



Best k = 7
Accuracy = 0.58125
Confusion Matrix =

17	1	0	0	0	0	1	1
4	12	2	0	2	0	0	0
1	3	11	1	1	0	1	2
4	1	1	11	0	1	2	0
0	1	1	1	15	0	1	1
5	2	1	3	1	4	3	1
4	2	4	0	2	0	8	0
3	0	1	0	1	0	0	15

The best k is 7. Using kNN can help to solve outlier's problem. If we use a very large k, it means that almost all the points can vote for deciding the test point class and it is also time-consuming during evaluation. We also need to find this hyperparameter large enough to ignore outliers. So, it should not be too small as well. In other word, A small value of k means that noise will have a higher influence on the result and a large value make it computationally expensive. In case of having ties, we will choose the first occurrence of the class label with respect to the order. (smaller valued class)

Part 4: Advanced techniques

Q4.1 Evaluate Recognition System - Support Vector Machine

I saved visionHarris.mat as visionSVM.mat because it works better. I used fitcocec function in Matlab and used gaussian, polynomial, linear and rbf kernels.

Are the performances of the SVMs better than nearest neighbor? Why or why not? Does one kernel work better than the other? Why?

Yes, SVM is better than NN here especially if we choose a good kernel, we can get a good accuracy. Because in SVM we have larger boundaries between classes which increase accuracy.

KNN has some nice properties: it is automatically non-linear, it can detect linear or non-linear distributed data, it tends to perform very well with a lot of data points. But, on the minus side KNN needs to be carefully tuned, the choice of K and the metric (distance) to be used are critical. KNN is also very sensitive to bad features (attributes) so feature selection is also important. It is also sensitive to outliers and removing them before using KNN tends to improve results.

SVM can be used in linear or non-linear ways with the use of a Kernel, when you have a limited set of points in many dimensions SVM tends to be very good because it should be able to find the linear separation that should exist. SVM is good with outliers as it will only use the most relevant points to find a linear separation (support vectors). So that's why I think SVM works better here.

The polynomial kernel seems to work better for our data. One reason is that it has a higher generalizing capability than other for multiclass classification. This greater flexibility helps to fit better into the training data. Also, it can define better non-linear boundaries.

SVM 1 with gaussian kernel
Accuracy for SVM1: 0.6
Confusion Matrix for SVM1:

12	2	2	1	0	0	0	3
2	14	0	0	2	0	1	1
2	2	14	0	1	0	0	1
3	2	1	7	0	3	3	1
0	3	1	0	15	0	1	0
1	0	0	7	1	7	1	3
0	0	1	2	6	1	9	1
1	0	0	1	0	0	0	18

SVM 2 with polynomial kernel
Accuracy for SVM2: 0.6625
Confusion Matrix for SVM2:

13	3	1	1	0	0	0	2
2	14	2	0	1	0	1	0
1	2	17	0	0	0	0	0
2	2	0	10	0	1	4	1
1	0	1	1	14	1	2	0
1	0	1	5	1	10	0	2
1	1	1	0	2	1	13	1
1	2	0	1	0	0	1	15

SVM 3 with linear kernel
Accuracy for SVM3: 0.56875
Confusion Matrix for SVM3:

11	4	1	1	0	0	1	2
3	12	2	0	2	0	1	0
0	2	17	0	0	0	1	0
4	2	0	5	2	3	4	0
0	2	0	2	11	0	4	1
2	1	0	6	0	8	2	1
0	1	1	2	3	0	12	1
1	3	0	1	0	0	0	15

SVM 4 with rbf kernel
Accuracy for SVM4 = 0.59375
Confusion Matrix for SVM4 =

12	2	2	1	0	0	0	3
2	14	0	0	2	0	1	1
2	2	14	0	1	0	0	1
3	2	1	7	0	2	4	1
0	3	1	0	15	0	1	0
1	1	0	7	0	7	1	3
0	0	1	2	7	1	8	1
1	0	0	1	0	0	0	18

Q4.2 Inverse Document Frequency

How does Inverse Document Frequency affect the performance? Better or worse? Does this make sense?

It helps to improve the accuracy a little bit when we use SVM and polynomial kernel. Because in IDF we are scaling down highly repetitive words in the dictionary so we can consider features that is important in classification more.

The reason that it can't increase the accuracy too much is that maybe appearing a feature more frequently does not have a complete positive relationship to the importance of that feature.

using SVM and polynomial kernel
Accuracy = 0.66875
Confusion Matrix =

13	2	1	1	1	0	0	2
2	14	2	0	1	0	1	0
1	2	17	0	0	0	0	0
3	1	0	10	0	1	4	1
1	0	0	1	15	1	2	0
1	0	1	5	1	10	0	2
1	1	1	1	2	1	12	1
2	1	0	1	0	0	0	16

Q4.3 Better pixel features

In your write-up: What did you experiment with and how did it perform. What was the accuracy?

Actually, I got a good accuracy (67%) in previous section. But I want to try other filters and see how they can affect the accuracy here.

By using HOG and libsvm library, I got 80.625% accuracy which is really good, and the computation is fast as well. You can reproduce the result by running tryBetterFeatures.m.

Configuration for SVM in libsvm: `'-t 0 -c 5000 -q'`

Filter: HOG, precision = 15 (size of interval in degree)
Classifier = SVM, kernel type: Linear
Accuracy = 80.625