Q1.1

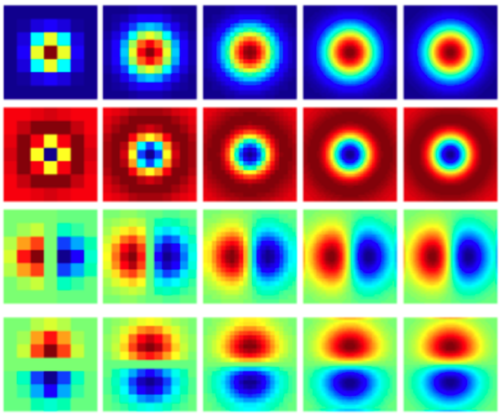

Figure 1. Filter Banks

The first-row filters are different-size Gaussian Filters, which are used to smooth the image on different scale.

The second-row filters are Laplacian of Gaussian Filters, which are used to highlight the edges of the images on different scale.

The third-row filters are Gaussian and y-direction Sobel filters, which are used to find gradient changes on y direction on different scale.

The third-row filters are Gaussian and x-direction Sobel filters, which are used to find gradient changes on x direction on different scale.

Q1.2

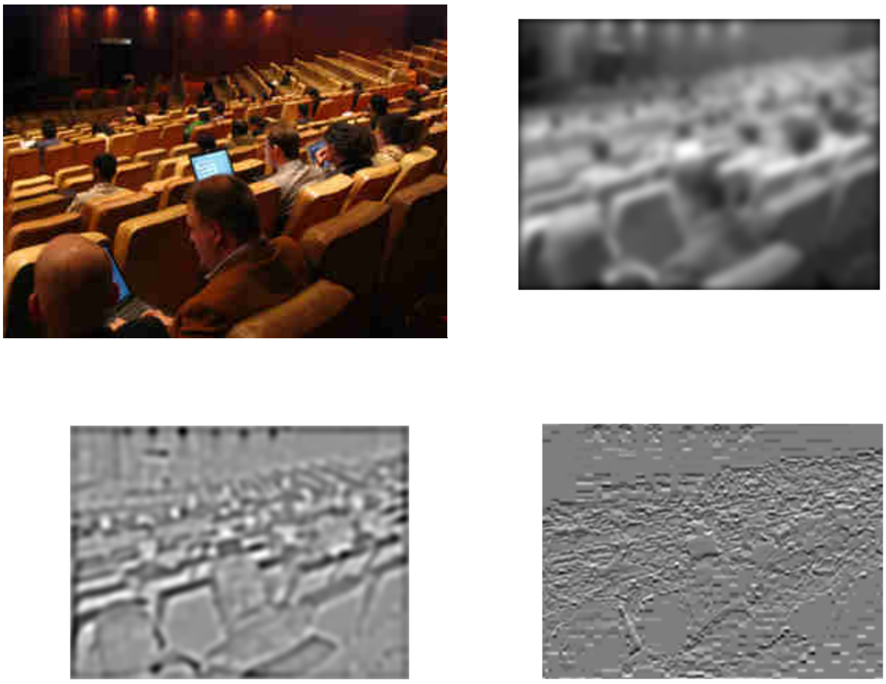

Figure 2. Filter Response of an image from the dataset

CIE Lab color space has three channels, L for lightness, a for green-red and b for blue-yellow. It has wider range of color than other color spaces.

It was designed for human perception. A large difference in lab space means a very different colors for human eyes and a small difference in lab means similar colors. As a result, it makes sense when we do computer vision tasks evaluated by people.

Q1.3

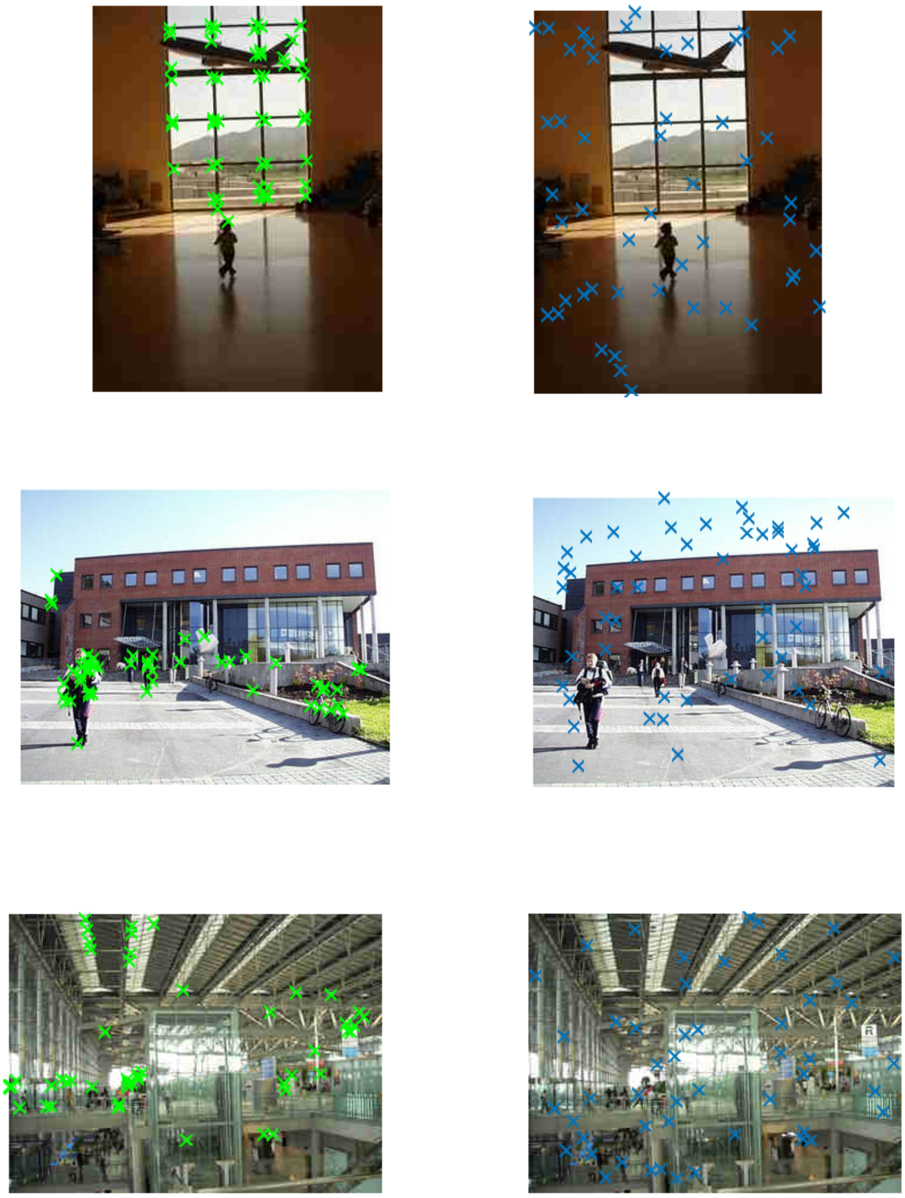

Figure 3. Harris and Random points generator on 3 images

Q2.1

Scene: football_stadium

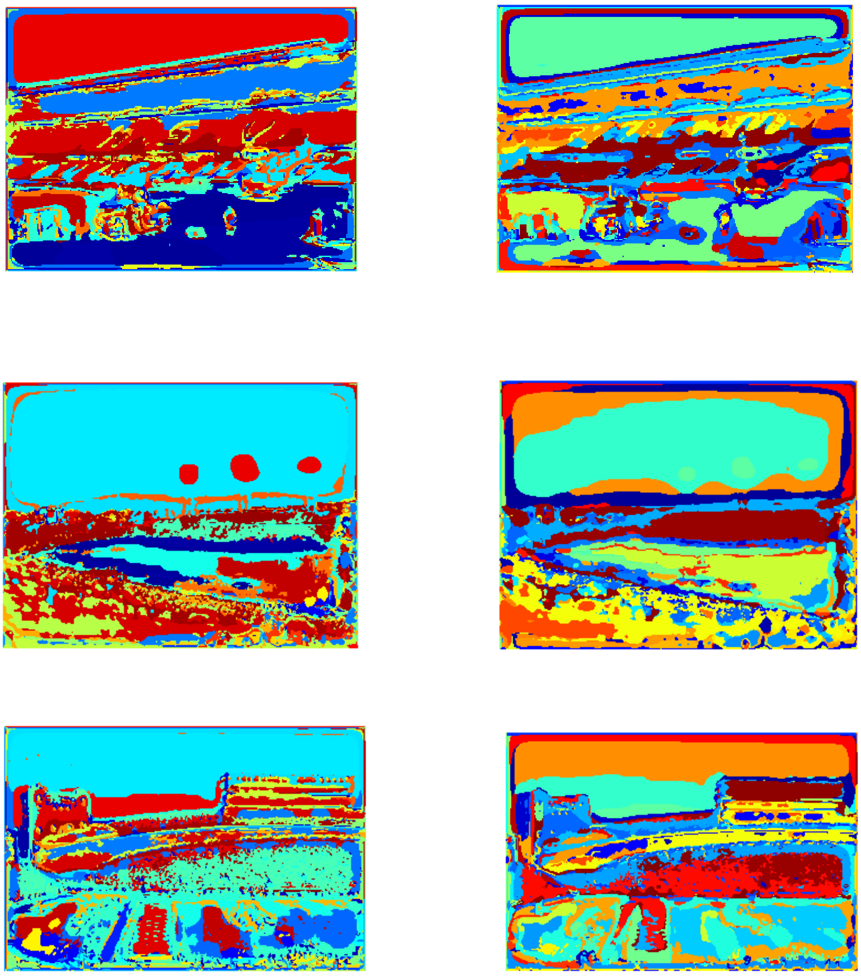

Figure 4. Harris(left) and Random(right) WordMaps

Figure 5. Original Images

Scene: campus

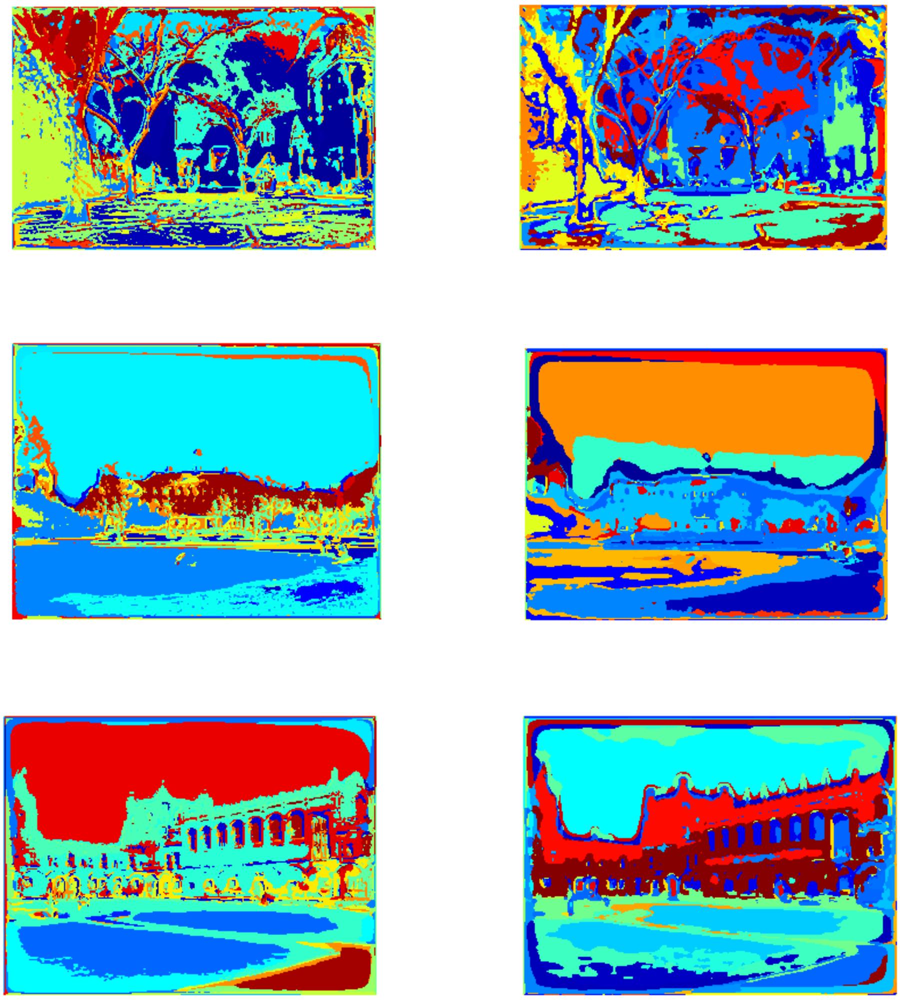

Figure 6. Harris(left) and Random(right) WordMaps

Fugure 7. Original Images

2.1 Conclusion:

These visual words are able to capture the semantic meanings since we can roughly separate the colors on different objects.

Harris dictionary seems to have a better performance. The edges are more clear to tell. There is only one color for sky. And similar objects seems to have the same color.

Q3.2 Evaluation

Harris, 50 points, 50 clusters, chi2:

accuracy: 51.25%

Confusion matrix:

10	3	0	2	1	1	2	4
1	13	5	2	2	1	1	0
4	2	12	2	3	2	2	1
1	0	0	9	0	3	5	1
0	1	2	0	12	1	1	0
0	1	0	3	0	7	0	0
1	0	0	1	2	2	5	0
3	0	1	1	0	3	4	14

Harris, 50 points, 50 clusters, Euclidean:

accuracy: 44.37%

Confusion matrix:

8	5	3	0	1	2	3	3
2	12	4	3	3	1	1	1
3	2	8	1	1	0	1	0
1	0	2	8	1	5	4	2
0	1	1	0	11	2	3	0
0	0	1	3	0	6	1	0
2	0	0	3	3	2	4	0
4	0	1	2	0	2	3	14

Random, 50 points, 50 clusters, chi2:

accuracy: 49.38%

Confusion matrix:

13	4	5	2	0	1	1	3
1	12	3	3	3	2	1	0
4	2	11	1	3	3	2	1
0	0	0	6	1	1	4	0
0	1	1	1	9	0	5	0
0	1	0	3	0	7	0	0
0	0	0	4	4	5	5	0
2	0	0	0	0	1	2	16

Random, 50 points, 50 clusters, Euclidean:

accuracy: 46.88%

Confusion matrix:

11	5	4	3	0	1	2	2
1	11	4	3	3	1	2	0
5	1	10	1	4	1	2	1
1	1	0	6	2	3	3	3
0	1	2	1	7	1	2	0
0	1	0	0	0	8	0	0
1	0	0	6	4	3	8	0
1	0	0	0	0	2	1	14

Q3.2 Conclusion:

The performance of Harris dictionary should be better than random dictionary since the wordMap outputs of Harris dictionary seem to have more organized information than random dictionary outputs. For “chi2” method, I get the expected output, 51.25% for Harris and 49.38% for random. However, for “Euclidean” method, random dictionary has a better performance than Harris dictionary, 44.37% for Harris and 46.88% for random, which surprised me a lot. This may be result from the method I implement for Harris corner detection. Using non-maximal suppression or not and some details will result in a change of accuracy.

For “chi2” and “Euclidean” methods, “chi2” absolutely has a better performance. The reason is that chi-squared distance is a kind of weighted Euclidean distance. In Euclidean distance, each entry has the same weight in the distance calculating.

$$Euclidean\ Distance = \sqrt{\sum (a_i - b_i)^2}$$

However, it is more reasonable if we put less emphasis on the entries with more elements.

$$Chi - squared\ Distance = \sqrt{\sum \frac{(a_i - b_i)^2}{a_i + b_i}}$$

so that if one specific entry’s frequency is high, it doesn’t matter if we have some noise around it, the distance on this entry keeps small.

QX.1 SVM for Harris dictionary, 50 points, 50 clusters

1) For Gaussian Kernel:

Accuracy =

0.5125

Confusion Matrix =

15	5	0	3	0	5	3	3
2	13	6	1	3	2	0	1
1	0	13	1	3	2	2	0
0	0	0	6	0	3	3	1
0	1	0	0	10	1	2	0
0	0	0	0	0	2	0	0
0	1	0	9	4	4	9	1
2	0	1	0	0	1	1	14

2) For Linear Kernel:

Accuracy =

0.4563

Confusion Matrix =

14	5	0	3	0	6	4	4
2	11	5	1	4	2	0	1
1	1	14	1	4	2	2	0
0	0	0	4	0	3	1	1
0	2	0	0	8	1	3	0
0	0	0	0	0	0	0	0
0	1	0	11	4	5	9	1
3	0	1	0	0	1	1	13

2) For Polynomial Kernel:

Accuracy =

0.5500

Confusion Matrix =

15	5	1	3	0	2	2	3
2	13	5	1	3	3	0	1
1	0	14	1	3	1	2	0
0	0	0	8	0	4	4	1
0	1	0	0	11	1	2	0
0	0	0	0	0	4	0	0
0	1	0	7	3	4	9	1
2	0	0	0	0	1	1	14

Conclusion: Linear kernel has a poor performance on the test set since it’s impossible to separate nonlinear data. Polynomial Kernel has the highest performance since it projects the original data dimension to a higher dimensional space where they are separate, but it’s computational expensive. Gaussian takes less computational resources so that it has a lower performance than Polynomial Kernel.

QX.2

`computeIDF.m`

First, I calculated each words' frequency using a loop for all training images and calculated IDF according to the expression in the handout.

`getImageFeature_IDF.m`

I have to revise the getImageFeature function since the histogram should multiply the IDF to get updated.

`evaluateRecognitionSystem_IDF.m`

Use SVM to train and classify the data

Conclusion:

After I implemented the Inverse Document Frequency method, the accuracy actually became worse.

Linear: 0.4563->0.4563
Gaussian: 0.5125->0.4750
Polynomial: 0.5500->0.5000

One possible reason for this is that there were too few clusters. When I checked the wordCounts of each entry, most of them are above 1000 out of 1331 which means that most of the entries get small IDF afterwards and then ignored.

	1	2
2	1069	
3	1228	
4	405	
5	1261	
6	992	
7	935	
8	1152	
9	1170	
10	973	
11	677	
12	1325	
13	1239	
14	1249	
15	1160	
16	1057	
17	1296	
18	742	
19	753	
20	597	
21	498	
22	611	
23	1331	
24	1206	
25	1034	
26	1293	
27	1002	

	1
1	0.0038
2	0.2192
3	0.0805
4	1.1898
5	0.0540
6	0.2940
7	0.3531
8	0.1444
9	0.1289
10	0.3133
11	0.6760
12	0.0045
13	0.0716
14	0.0636
15	0.1375
16	0.2305
17	0.0266
18	0.5843
19	0.5696
20	0.8018
21	0.9831
22	0.7786
23	0
24	0.0986
25	0.2525
26	0.0290

Figure 8. word counts and IDF