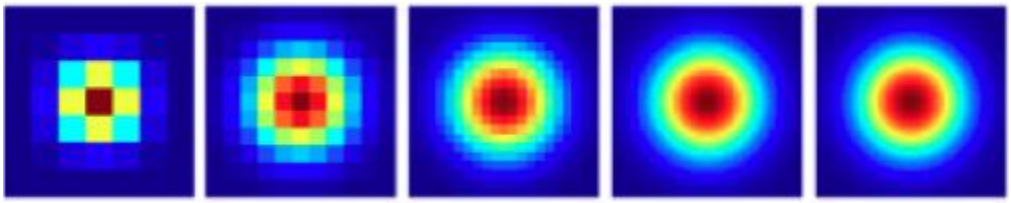
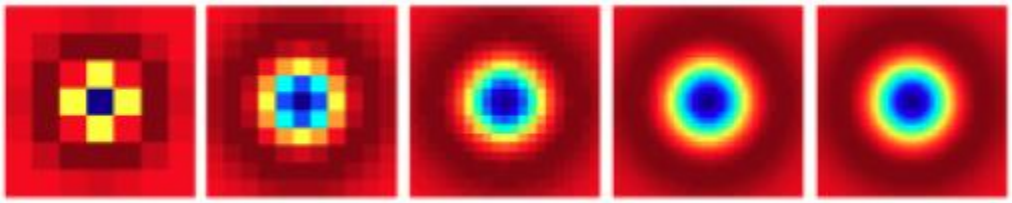
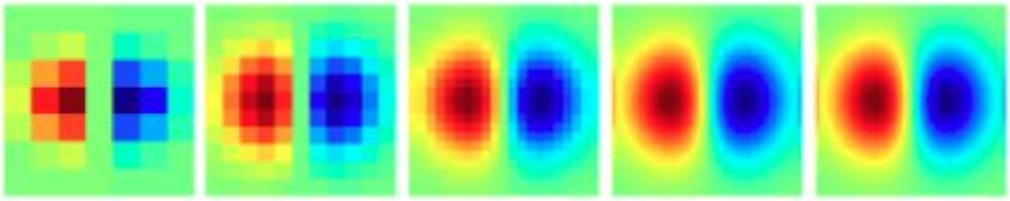
1.1



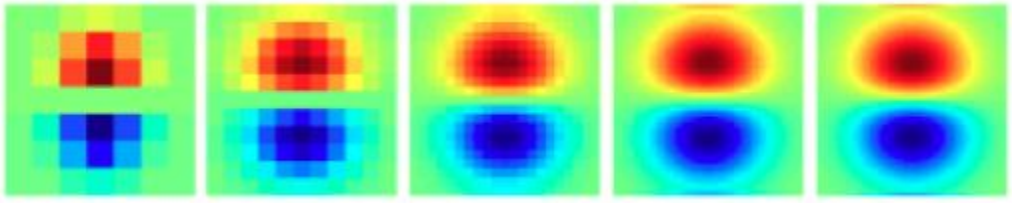
The first five filters are Gaussian filters of increasing granularity. They pick up local intensity information, blurred normally. As the order of the filter increases, its size increases, capturing intensity information over a larger local area centered around the pixel.



The next 5 filters are LoG filters or Laplacians of Gaussian filters. Essentially they capture edge information. A simple second derivative is very noisy, which is why the image is first smoothened with a Gaussian and then the Laplacian is applied. Edges have a zero crossing. From 6-1, the scale of the filter increases, ie, it looks for thicker edges.

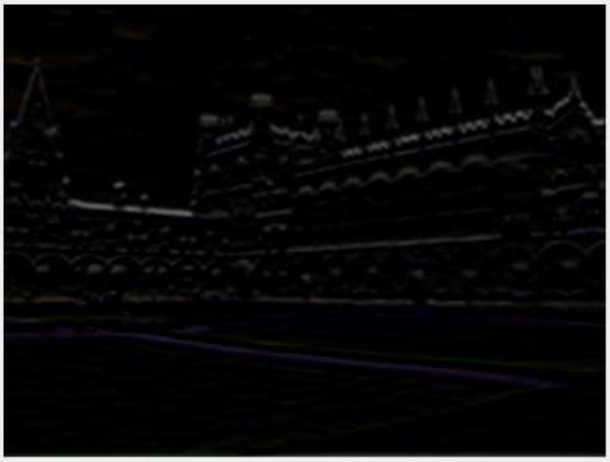


Filters 11-15 capture x derivatives, ie, vertical edges. The 11th filter captures fine, very thin vertical edges while the larger filters capture vertical edges pronounced over a wider area.



Filters 16-20 capture y derivatives, ie, horizontal edges. The 16th filter captures fine, very thin horizontal edges spread over few pixels while the larger filters capture horizontal edges pronounced over a wider area.

1.2



From left to right and top to bottom:

Image 1: original image

Image 2: Output of second filter. One can see the gaussian blur. Intensity information is spread over space.

Image 3: Output of 6th filter. LoG operation applied. One can see the edges highlighted. On close observation, the edges are double edges since LoG gives a zero crossing at edges.

Image 4: Output of 11th filter: vertical edge detection. One can see the vertical edges highlighted.

Image 5: Output of 16th filter: horizontal edge detection. One can see the horizontal edges highlighted.

CIE Lab Colorspace

The Lab color space has three components.

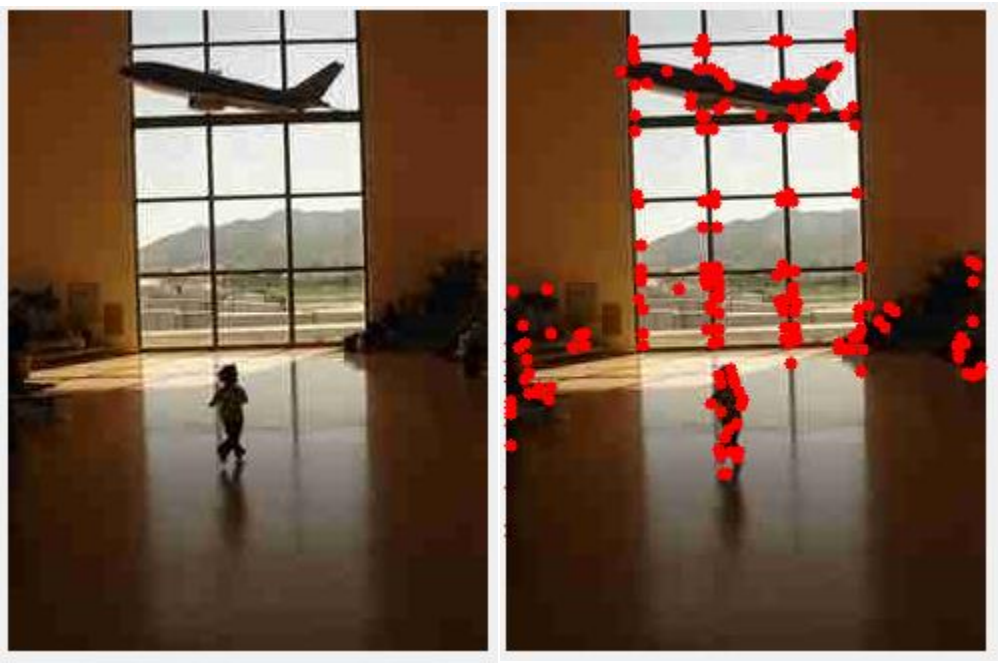
1. L – Lightness (Intensity).
2. a – color component ranging from Green to Magenta.
3. b – color component ranging from Blue to Yellow.

In Lab color space, the L channel is independent of color information and encodes brightness only. The other two channels encode color. It has the following properties.

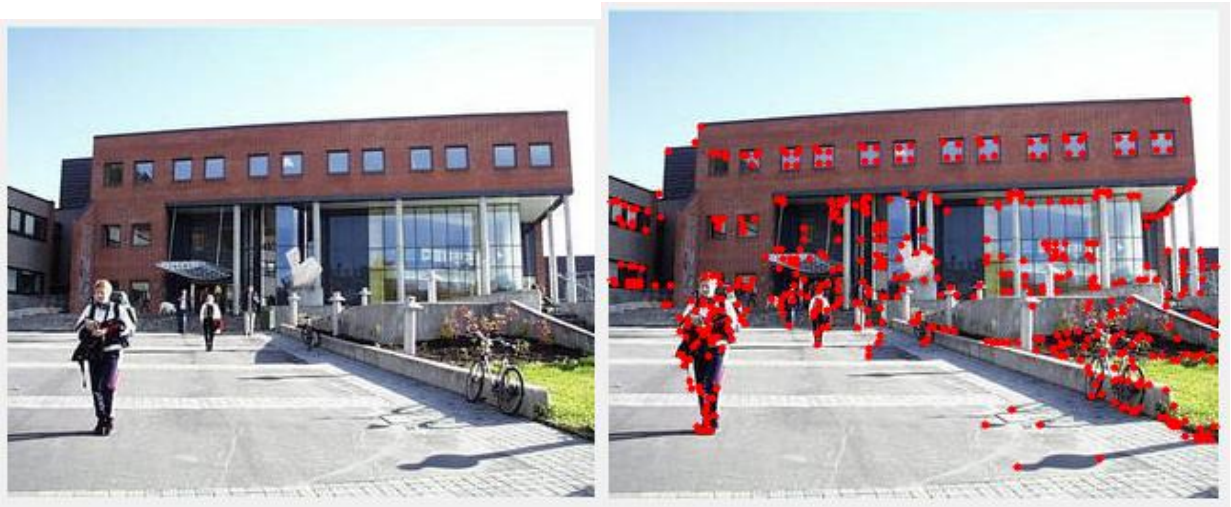
Perceptually uniform color space which approximates human vision and how we perceive color. The output is nearly independent of device and change in illumination mostly only affects the L component, unlike in RBG where all channels encode brightness. Therefore, the same images taken under different lights would have similar A and B components while this is not the case in RGB space. Thus, it can be used to make accurate color balance corrections by modifying output curves in the *a* and *b* components, or to adjust the lightness contrast using the *L* component.

1.3

Response of Harris Corner detecter:



With 200 points

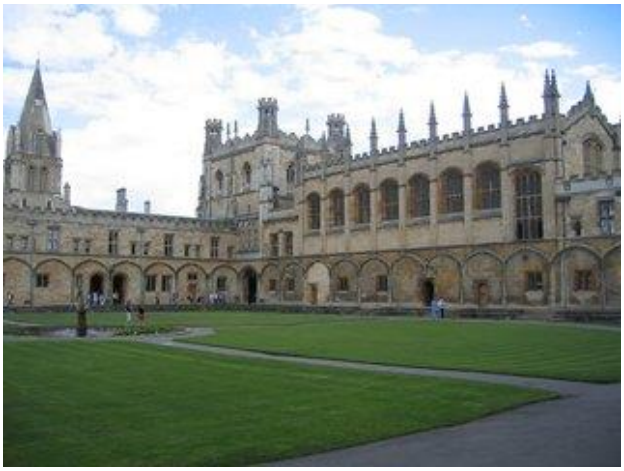


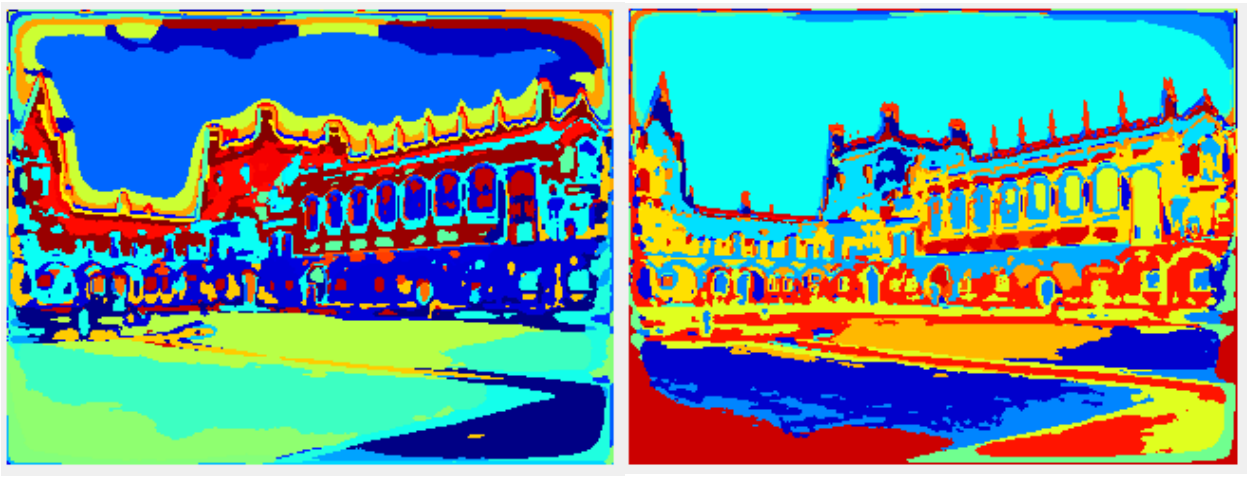
With 500 points

With 200 points

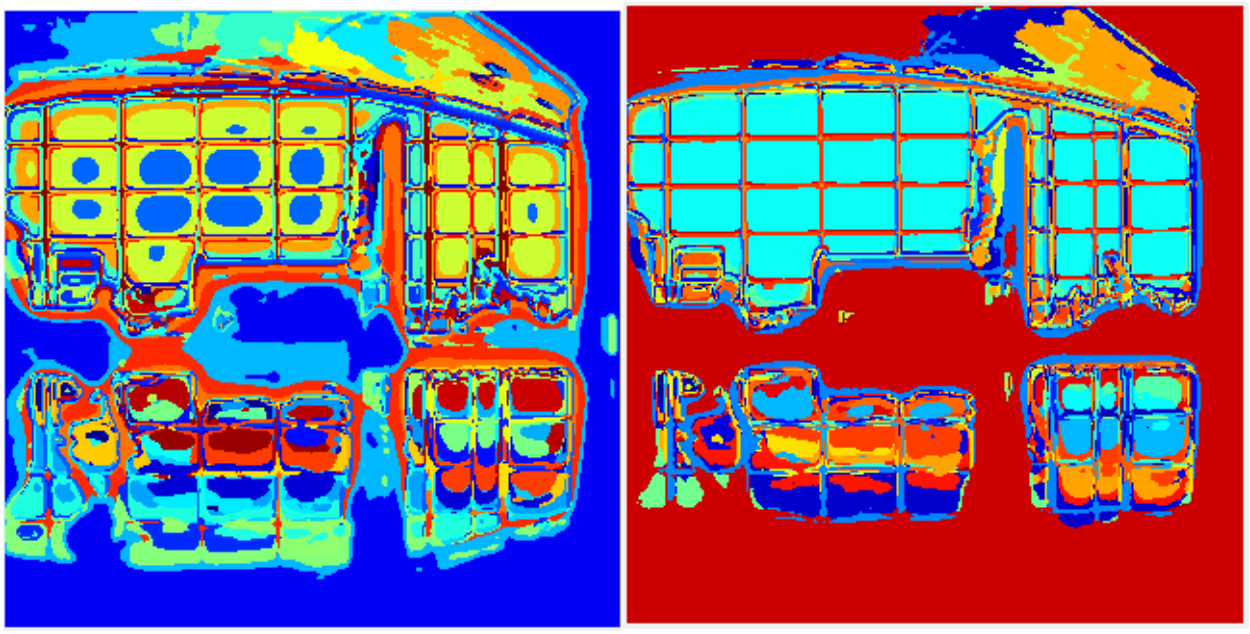
2.2

Alpha=50, K=100

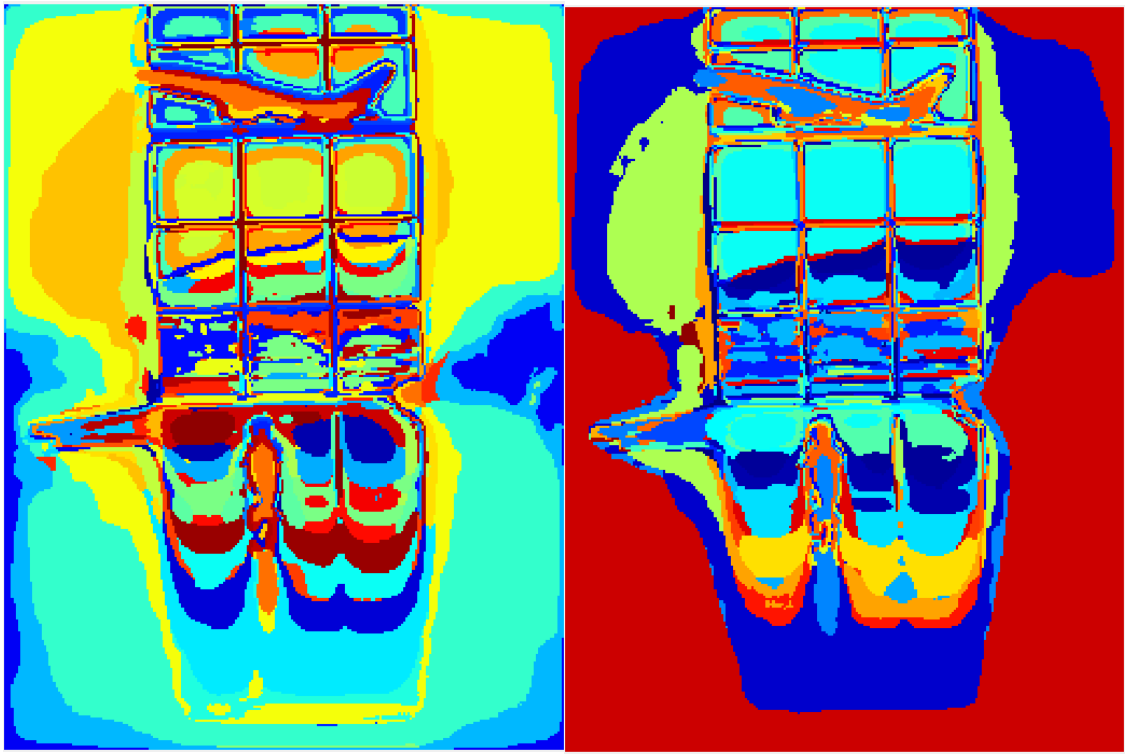




Left: Random, Right: Harris



Left: Random, Right: Harris



Left: Random, Right: Harris

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

Yes, the visual words capture semantic meaning. One can see that different colors are being applied to what is windows, grounds, doors, sky, aeroplane, etc respectively. This shows that the wordMap has classified these spaces as different “words” and captured the semantic information of these objects. Harris is a better dictionary because it is much more discriminative. If you see a single object, Harris often gives it a single classification compared to random, which segments the same object into different “words”. This is because Harris samples corners, and many corners put together define object boundaries. Corner points therefore are better features/”words” than random points, this is why Harris performs better in segmentation as well as classification.

3.2

<u>Random Points & Euclidean Distance with 500 points and 200 words</u>

correct = 69

wrong = 91

accuracy = 0.4313

confusion =

9	5	4	6	5	3	1	3
3	9	4	3	0	3	2	1
2	4	10	0	5	0	2	0
2	0	1	5	0	2	3	3
0	0	1	0	7	1	1	0
0	2	0	1	0	7	1	0
0	0	0	2	2	0	9	0
4	0	0	3	1	4	1	13

<u>Random Points & Chisq Distance with 500 points and 200 words</u>

correct = 82

wrong = 78

accuracy = 0.5125

confusion =

12	5	6	5	0	3	3	3
1	11	2	3	0	2	1	1

1	1	11	0	4	2	2	0
2	0	0	5	0	3	4	2
0	1	1	0	15	0	0	0
0	0	0	0	0	6	0	0
0	1	0	5	1	0	8	0
4	1	0	2	0	4	2	14

Harris Corner Points & Euclidean Distance with 500 points and 200 words

correct =72

wrong = 88

accuracy = 0.4500

confusion =

7	4	3	5	0	5	1	1
4	11	4	2	3	4	2	0
1	2	13	0	2	1	3	0
1	0	0	6	2	2	4	3
0	0	0	0	6	0	0	0
1	0	0	1	3	6	1	0
2	1	0	3	4	1	9	2
4	2	0	3	0	1	0	14

Harris Corner Points & ChiSq Distance with 500 points and 200 words

correct = 80

wrong = 80

accuracy = 0.5000

confusion =

10	8	1	3	0	2	1	3
3	8	4	2	3	2	2	0
1	1	14	0	3	3	4	0

1	0	0	8	1	4	2	3
0	1	1	0	10	0	0	0
0	0	0	0	0	6	0	0
1	1	0	4	3	0	11	1
4	1	0	3	0	3	0	13

Harris Corner Points & ChiSq Distance with 600 points and 200 words

correct = 86

wrong =74

accuracy = 0.5375

confusion =

15	4	1	5	1	4	2	2
1	13	5	2	2	3	1	0
1	1	13	1	1	1	1	1
0	0	0	6	0	2	4	2
0	1	1	1	11	1	2	0
0	0	0	0	0	4	1	0
0	1	0	4	5	2	9	0
3	0	0	1	0	3	0	15

Harris performs better than random and chi-square performs better than Euclidean.

It is not surprising that Harris performs better than Random, because corners constitute essential features/"words" to define objects and thereby perform better for feature matching for scene classification. The textons collected at corners are more significant to decision making than ones collected at random locations. This difference is extremely evident when the number of sample points is small. For 50 points, Harris performs about 8% better than Random points. The difference becomes less significant as more and more points are sampled, since Harris points are clumped near corners and samples many points from the same local region while Random covers a larger area.

Chi-square distance performs better than Euclidean distance.

Euclidean distance between two vectors x and y, where i is an orthogonal direction = $\sqrt{\sum(xi - yi)^2}$.

Chi-square distance, on the other hand is, $\sum \frac{(xi-yi)^2}{2(xi+yi)}$. It is the square of the difference between the observed and expected values for a cell, divided by the expected value for that cell. This works better for

correspondence analysis than Euclidean because essentially, we want to compare similarity/correlation of histograms rather than absolute distance. Because Chi-square is a weighted Euclidean it is smoothened and is less affected by extremes. Chi-square is a better metric for comparing histograms.

X.1

The following were observed with Harris Corner Points & ChiSq Distance with 600 points and 200 words

SVM with gaussian kernel

correct = 75

wrong = 85

accuracy = 0.4688

confusion =

13	0	1	0	2	0	0	0
1	11	2	5	10	0	1	1
0	0	0	0	0	0	0	0
0	3	0	9	0	0	2	0
0	1	0	1	2	0	0	0
0	0	4	1	1	12	0	1
1	0	2	3	0	5	11	1
5	5	11	1	5	3	6	17

SVM with polynomial kernel of order 8

correct = 82

wrong = 78

accuracy = 0.5125

confusion =

13	0	1	0	0	0	0	1
1	10	3	3	7	0	1	0
0	0	1	0	0	0	0	0
0	5	1	11	0	0	0	0

1	2	3	1	7	0	0	0
0	0	3	1	0	13	1	1
1	0	2	3	2	5	13	4
4	3	6	1	4	2	5	14

SVM with gaussian kernel is worse than Harris with chisq and SVM with polynomial kernel is comparable to the Harris with chisq. The above experiments were conducted with 600 points and 200 words in dictionary. SVm works very well for binary classification, however since this task is multi-class classification, one had to combine many binary classifiers. This may be one reason why performance is worse than K-nearest neighbor. SVM works well on linearly separable data, while K-nearest neighbours is good for clustering in general. Another reason may be that the wordMaps are generated by Kmeans clustering which gives a good prior for KNN classification while it may not give a good prior for SVM classification.

Polynomial kernel of order 8 works better than gaussian kernel because it has a higher generalizing capability than gaussian for multiclass classification. Its greater flexibility allows it to fit better into the training data.

X.2

<u>IDF with Harris 600 points and 200 words where IDF is applied after histogram step.</u>

correct = 75

wrong = 85

accuracy =

0.4688

confusion =

11	4	5	6	1	4	3	2
2	12	3	2	4	0	0	0
3	1	9	2	1	2	0	1
1	1	0	7	0	2	4	2
0	1	1	0	10	1	5	0
0	0	1	1	0	5	1	1
0	1	1	2	4	3	7	0
3	0	0	0	0	3	0	14

<u>IDF with Harris 600 points and 200 words where IDF is applied before histogram step</u>

Here, word frequency is multiplied with IDF and then histogram is generated.

correct = 67

wrong = 93

accuracy = 0.4188

confusion =

9	5	1	5	4	2	5	2
5	12	4	2	4	0	1	1
3	2	12	2	0	5	0	1
1	0	0	3	0	1	5	1
0	1	2	1	5	1	4	0
0	0	1	2	1	8	0	2
0	0	0	5	6	1	5	0
2	0	0	0	0	2	0	13

Applying IDF actually sees a drop in performance. While IDF should intuitively increase the performance, in this case the results may have been reversed because the dictionary is very sparse. Since there are only 200 words in the dictionary, weighting corner features this way means that corners that are very prominent and picked up often by Harris are given lower weightage than a corner which may be a noise point simply because it is rare. In a large dictionary with a lot of words, it may make sense to ignore common prominent words and weigh less frequent words higher, but here, this may just amplify the effect of noisy points.

X.3

Please check file tryBetterFeatures.m. I tried HOG descriptors. However, the size of the HOG vector depended on the size of image giving rise to feature vectors of different sizes making distance comparison impossible. To counter this, I clipped the features and applied an SVM, however, I couldn't fully debug the model.