Q1.1

Below is an original image, with three filter responses

A. Original Image

B. Gaussian Filter

C. X Gradient

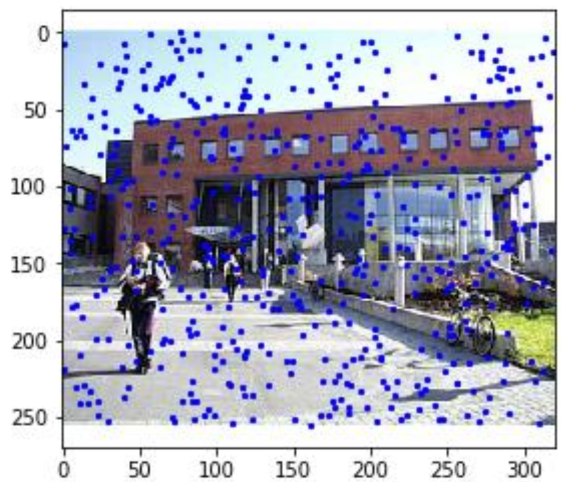
D. Y Gradient

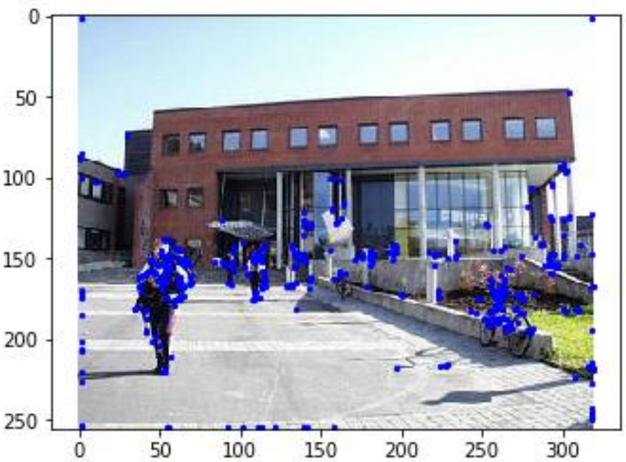
Image B shows the typical Gaussian blur. Both images C and D show the effect of the different gradient filters at their initial scale.

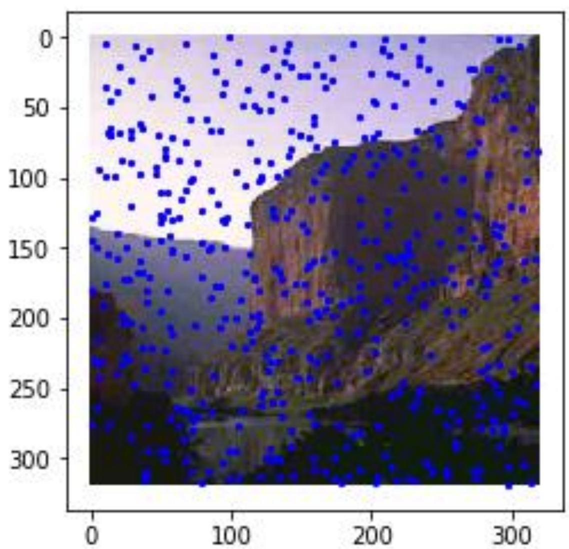
CIE Lab mathematically describes all perceivable colors in three dimensions L,a,b. L is for lightness, a is the color components of green-red and b is the color components blue-yellow. A benefit is that colorimetric distances between the individual colors correspond to perceived color differences. Also, the subset of colors represented is larger than that of RGB, this allows for more accurate filtered results.

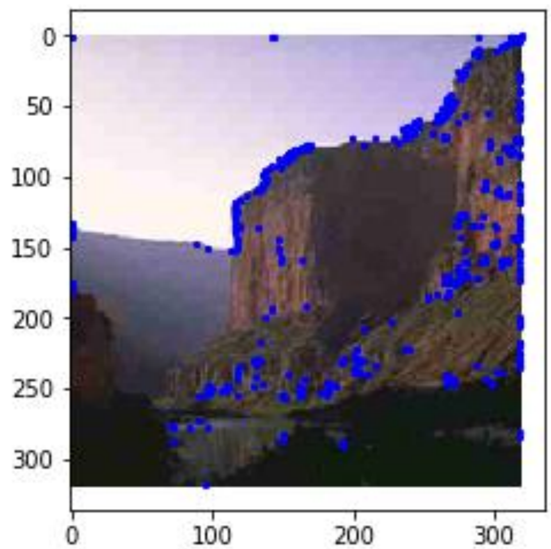
Q1.2

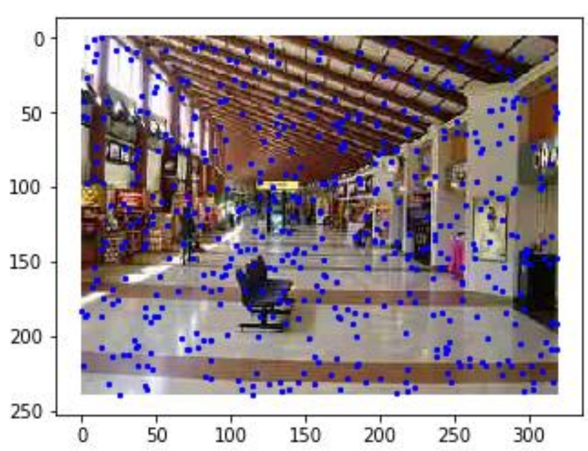
500 Random Points **500 Harris Points**

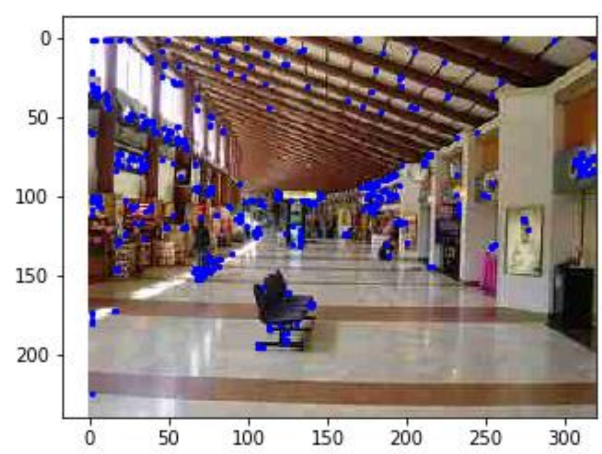








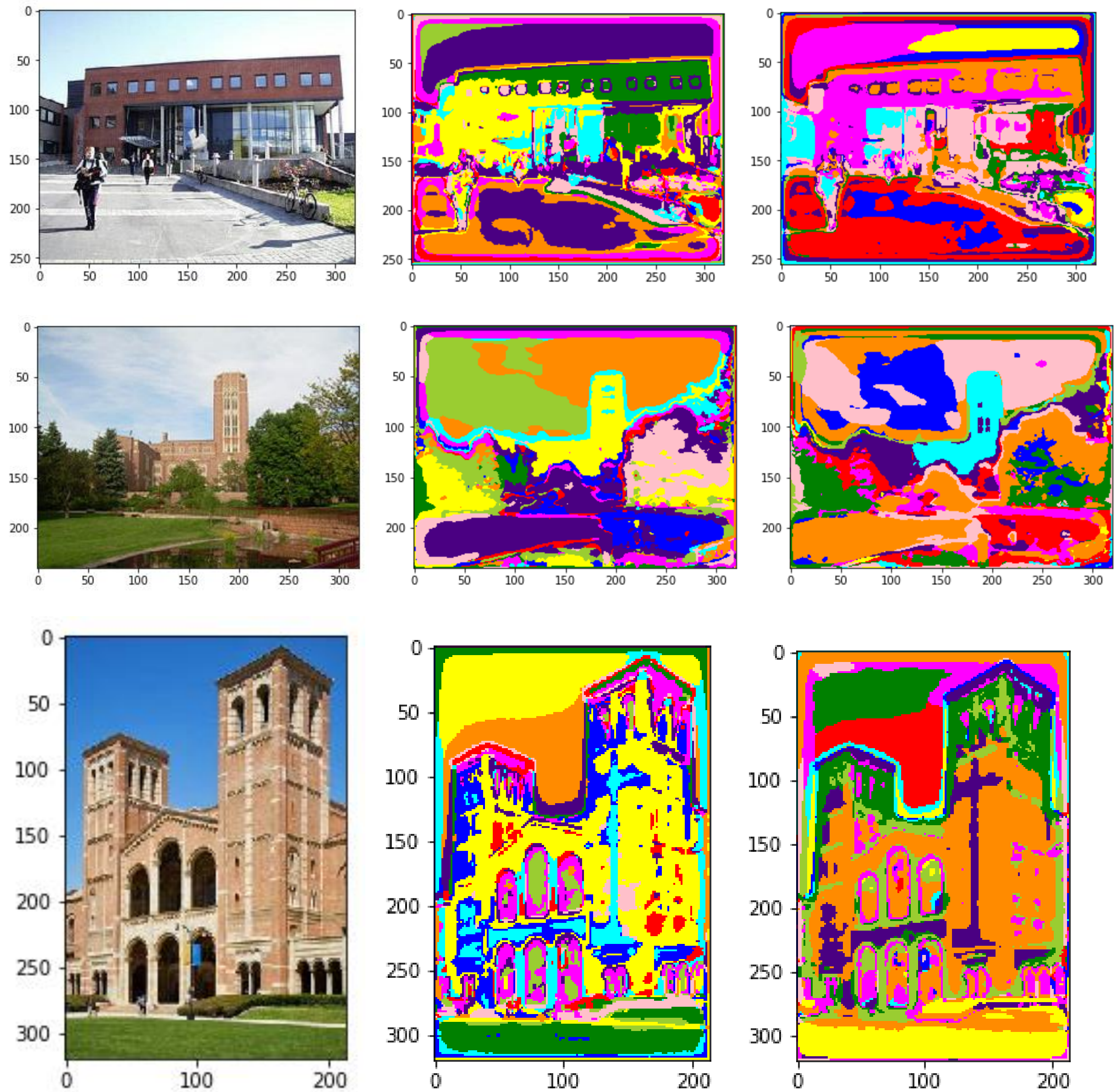




Q2.2

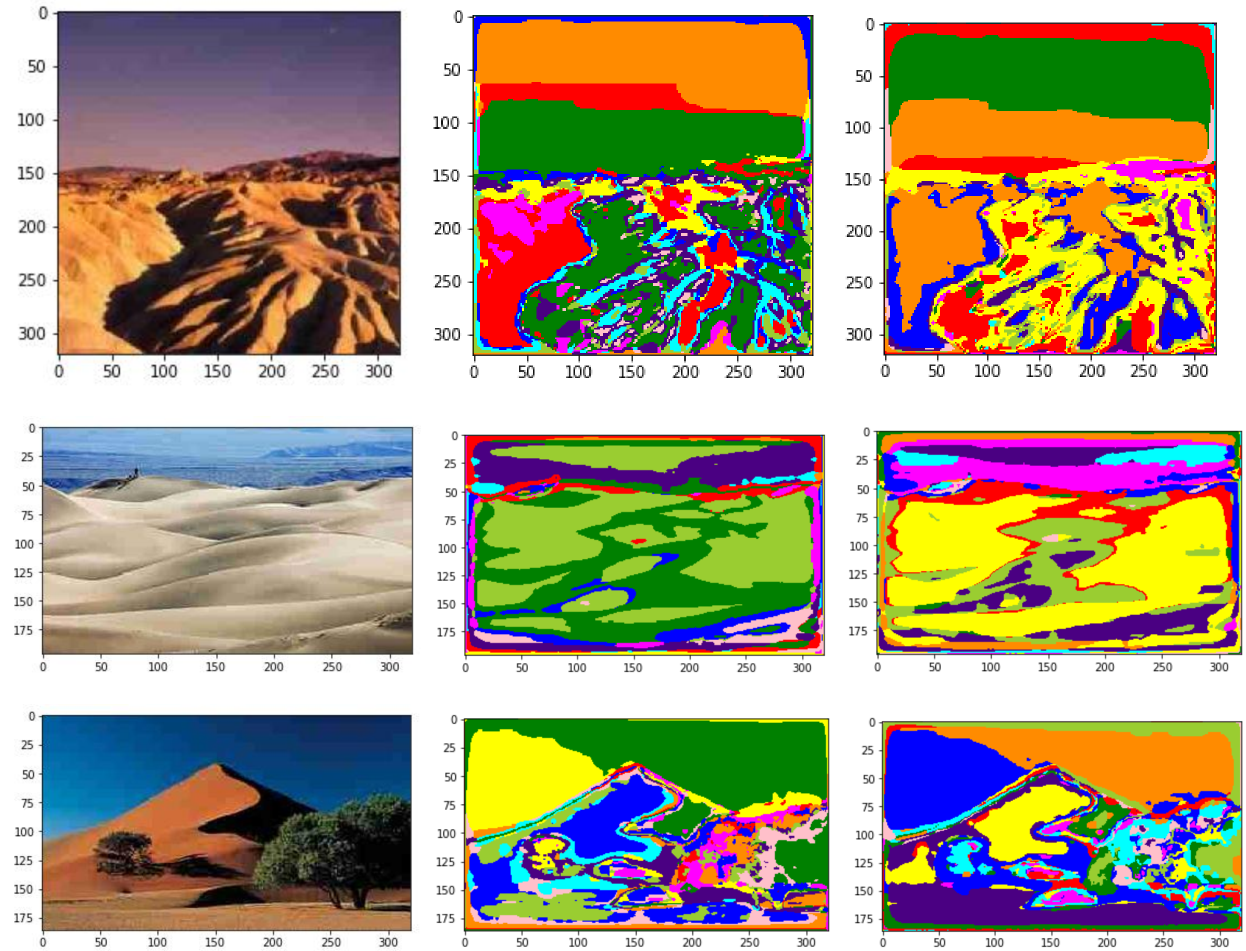
Campus

Original **Random** **Harris**



Desert

Original **Random** **Harris**



The visual words are sort of capturing semantic meanings. In Harris, orange seems to be something that is sort of flat and darker. Light blue seems to be picking out somewhat high contrast areas. In regards to which dictionary is better, the random dictionary, looking at the campus images, every building has yellow, where it seems like the majority color for each building in the Harris dictionary are all different. It’s quite hard to quantify the differences between them. With the alpha and k they are currently at, I would just have to side with the data and pick the random dictionary.

Q3.2

Euclidean **Chi-Squared**

Random Dictionary

```
Using Random Dictionary and Eucldean distance metric
Accuracy = 0.4625
[[ 14.   2.   2.   0.   0.   0.   0.   2.]
 [  5.  11.   3.   0.   1.   0.   0.   0.]
 [  5.   3.  10.   1.   0.   1.   0.   0.]
 [  3.   0.   2.   7.   1.   1.   4.   2.]
 [  1.   3.   2.   2.   9.   0.   2.   1.]
 [  1.   3.   1.   2.   2.   4.   4.   3.]
 [  2.   3.   1.   5.   3.   0.   3.   3.]
 [  2.   0.   0.   1.   0.   0.   1.  16.]]
```

```
Using Random Dictionary and ChiSquared distance metric
Accuracy = 0.50625
[[ 13.   1.   3.   0.   0.   0.   0.   3.]
 [  3.  11.   3.   0.   1.   1.   1.   0.]
 [  5.   4.  11.   0.   0.   0.   0.   0.]
 [  0.   0.   5.   8.   1.   3.   3.   0.]
 [  0.   3.   3.   0.  11.   0.   3.   0.]
 [  1.   3.   3.   1.   2.   5.   3.   2.]
 [  2.   3.   2.   2.   0.   0.   8.   3.]
 [  3.   0.   0.   2.   0.   0.   1.  14.]]
```

Harris Dictionary

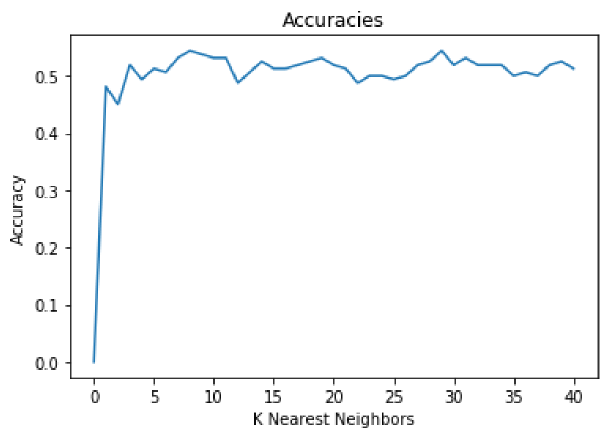
```
Using Harris Dictionary and Eucldean distance metric
Accuracy = 0.4
[[ 10.   1.   5.   1.   0.   0.   0.   3.]
 [  5.  10.   2.   1.   1.   0.   1.   0.]
 [  6.   7.   6.   0.   1.   0.   0.   0.]
 [  3.   0.   2.   7.   1.   1.   4.   2.]
 [  2.   4.   0.   2.   7.   1.   3.   1.]
 [  1.   2.   3.   4.   1.   5.   1.   3.]
 [  3.   3.   3.   3.   0.   1.   4.   3.]
 [  5.   0.   0.   0.   0.   0.   0.  15.]]
```

```
Using Harris Dictionary and ChiSquared distance metric
Accuracy = 0.48125
[[ 10.   2.   5.   1.   0.   0.   0.   2.]
 [  3.  13.   2.   0.   1.   0.   1.   0.]
 [  4.   3.  12.   1.   0.   0.   0.   0.]
 [  3.   0.   2.   7.   1.   1.   5.   1.]
 [  0.   4.   2.   4.   8.   0.   2.   0.]
 [  2.   1.   5.   2.   0.   6.   1.   3.]
 [  4.   1.   2.   5.   0.   0.   5.   3.]
 [  2.   0.   0.   2.   0.   0.   0.  16.]]
```

Surprisingly, the random dictionary was slightly better than the Harris dictionary. While this could be due to an implementation mistake, I think a larger a much larger alpha and K value would have shown the favor toward the Harris dictionary. Due to time constraints, alpha was kept at 50 and K was kept at 100.

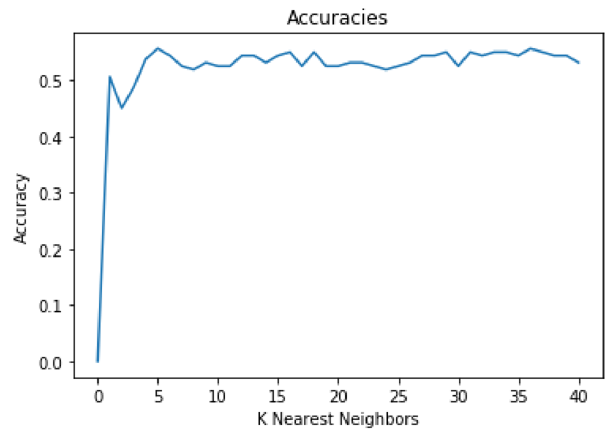
Compared to the variance between dictionaries, the distance metric played a larger role. The Chi-Squared metric performed better. This increase in performance is likely from weighting the difference squared by the sum of the two points rather than a generic square root, used in the Euclidean distance. As points x,y move further away from each other, the Chi-Squared distance doesn't grow as quickly as the Euclidian distance.

Harris




```
Best k number of nearest neighbors was 8 with accuracy 0.54375
[[ 14.   1.   2.   0.   0.   0.   0.   3.]
 [  5.  13.   2.   0.   0.   0.   0.   0.]
 [  3.   4.  13.   0.   0.   0.   0.   0.]
 [  4.   0.   2.   9.   0.   1.   4.   0.]
 [  0.   2.   3.   1.  11.   0.   2.   1.]
 [  1.   1.   5.   1.   1.   6.   2.   3.]
 [  8.   1.   1.   3.   2.   1.   4.   0.]
 [  3.   0.   0.   0.   0.   0.   0.  17.]]
```

Random




```
Best k number of nearest neighbors was 5 with accuracy 0.55625
[[ 13.   3.   2.   0.   0.   0.   0.   2.]
 [  4.  13.   3.   0.   0.   0.   0.   0.]
 [  3.   4.  13.   0.   0.   0.   0.   0.]
 [  4.   1.   1.   7.   0.   1.   5.   1.]
 [  0.   3.   4.   1.  11.   0.   1.   0.]
 [  2.   3.   3.   2.   1.   6.   1.   2.]
 [  4.   1.   2.   2.   1.   0.   9.   1.]
 [  2.   0.   1.   0.   0.   0.   0.  17.]]
```

Since the accuracies between the dictionaries was close, both were ran to find an optimum k nearest neighbors. The best value of k for Harris Dictionary was 8. Looking at the graph above, the accuracy does seem to oscillate as k is increased. In theory, as k continues to get larger, accuracy will go down because K-Nearest-Neighbors will simply favor the clusters with the most visual words. K = 8 is reasonable considering it is relatively small, good for run time, and also is not a sharp point on the graph but rather the peak of a somewhat smooth maximum region. This smoothness is a safer k value than say k=28, where the accuracy is nearly the same but the curvature is much sharper. The sharper the maximum, the larger probability for accuracy fluctuation. Interestingly, the random dictionary accuracies seem to vary less as k increases compared to the Harris dictionary.

Note: Running joblib.Parallel on windows requires some extra steps to avoid recursive spawning. These include having to surround the code that contains the call to Parallel with an if statement, as well as having to bring in all of your own functions that you would simply import.