

# 16720-B HW2

Jacob Miller

October 2020

**1.1.1** What visual properties do each of the filter functions pick up? You should group the filters into categories by its purpose/functionality. Also, why do we need multiple scales of filter responses?

The Gaussian filter is used primarily for de-noising and blurring images. This filter can be used to blur images on its own, or used with another filter to de-noise the image for detecting edges. The Laplacian of Gaussian,  $DoG_x$  and  $DoG_y$  are all derivative filters that can be used to detect edges in images. Since derivative filters are sensitive to noise, they also perform some smoothing. Multiple scales are needed to detect features of different sizes within the images.

**1.1.2** Collage of filter responses

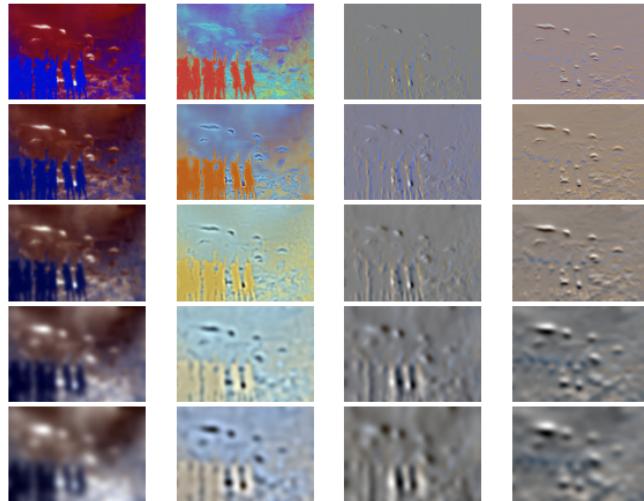


Figure 1: The filter responses of the four different filters with five different scales.

**1.2.1** The corner detection results on 3 random images from the provided dataset

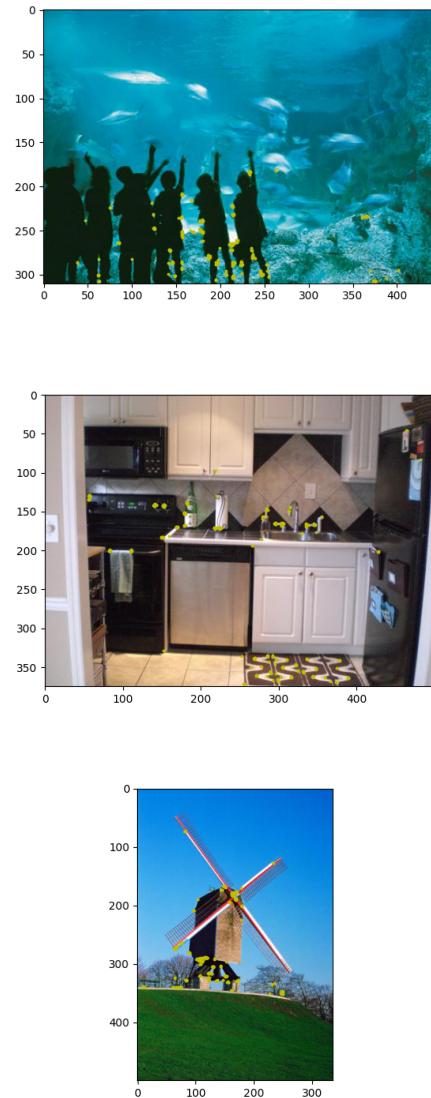


Figure 2: The corner detection results on 3 random images from the provided dataset.

### 1.3.1 Visualizations of 3 wordmaps with their original image. Do the word boundaries make sense?

Many of the word boundaries align relatively well with different parts of each scene. For instance, the children in the aquarium image are all colored yellow in the wordmap while much of the background is blue. However, there are clear limitations. Because of differences in the lighting of the blue sky in the windmill scene, the pixels corresponding to the sky are of various colors in wordmap. Ideally, these pixels would share a common color since they are all representing the same thing. There can also be quite a bit of noise with the wordmap boundaries, as can be seen in the kitchen scene.



Figure 3: Visualizations of wordmaps (left) and their original images (right).

### 2.1.1 Visualization of a wordmap histogram

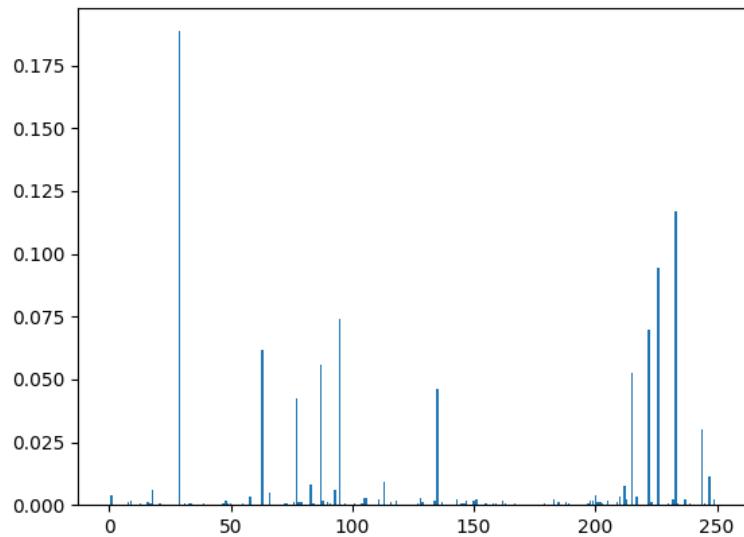


Figure 4: The histogram representing the wordmap of an image.

### 3.1.1 BoW confusion matrix and accuracy

12	1	0	0	0	0	0	1
0	11	1	2	0	2	1	1
1	1	12	4	3	1	0	3
1	4	1	13	0	1	1	5
2	1	0	0	10	0	0	0
0	0	0	1	8	10	4	1
1	6	1	2	1	1	8	1
1	2	1	4	1	0	0	10

Table 1: Confusion matrix for BoW image classification.

The accuracy of the BoW image classifier was 53.75%.

**3.1.2** As there are some classes/samples that are more difficult to classify than the rest using the bags-of-words approach, they are more easily classified incorrectly into other categories. List some of these classes/samples and discuss why they are more difficult.

The most difficult class for my BoW classifier was the waterfall. Only 8 of 21 test images depicting waterfalls were correctly classified. Waterfalls are tricky to classify because there can be drastic variance between the appearance of waterfalls. Additionally, some characteristics of a scene with a waterfall might also be seen in other settings. This is likely why 6 waterfall pictures were incorrectly classified as parks.

Other classes that were difficult included laundromats and highways. Again, this is likely because there can be a lot of variance between images of these classes and they may contain common features with other classes such as kitchens for laundromats or parks for highways.

### 3.1.3 Improving classification performance

First, I experimented with different classifiers, keeping all other parameters consistent. I compared the results of K nearest neighbors (KNN), multi-class support vector machine (SVM), and naive Bayes. Accuracy dropped for the KNN and SVM classifiers, but the naive Bayes classifier reached an accuracy of 56.875%.

Next, I tried increasing the number of spatial period levels ( $L$ ) using the naive Bayes classifier. Increasing  $L$  had no increase in accuracy.

Next, I increased  $K$  to 300 and  $\alpha$  to 500 and tested the naive Bayes classifier with the new, larger dictionary. These changes to the parameters increased the accuracy to 66.25% with the confusion matrix shown below.

14	0	0	0	0	0	0	0	0
0	15	0	0	0	2	1	0	
0	4	9	4	0	4	1	3	
0	1	1	18	0	2	2	2	
0	0	0	0	10	3	0	0	
1	2	0	0	2	19	0	0	
1	5	0	0	1	3	10	1	
1	0	0	3	3	1	0	11	

Table 2: Confusion matrix for BoW naive Bayes image classification with improved parameters.

### 3.1.4 How does Inverse Document Frequency affect performance?

Using the nearest neighbor classifier with  $K=300$  and  $\alpha=500$ , the accuracy without IDF was 55.625%. With the same parameters using IDF, the accuracy went up to 58.125%. The classifier accuracy likely went up because visual words that were appearing often in all images were no longer being weighted as heavily. This would come into play while calculating the distance between sets to find the nearest neighbor.

Testing IDF with the naive Bayes classifier that obtained 66.25% accuracy from question 3.1.3, however, yielded different results. When IDF was used, the accuracy went down to 56.25%. This is likely because naive Bayes calculates the probability of each image belonging to each class based on the calculated features rather than computing the similarity between histograms. Scaling these features hurts rather than helps performance in this case.

**4.1.1** Difference between feature extractor results  
The error is 2976.64, which is larger than expected.

**4.2.1** Confusion matrix and accuracy. Are the results better or worse than classical BoW? Why do you think that is?

12	0	1	0	0	1	0	1
0	17	0	0	0	0	1	0
0	0	23	0	0	0	1	1
0	0	0	25	0	0	1	0
0	0	0	0	12	1	0	0
0	0	0	0	4	20	0	0
0	1	0	0	0	0	20	0
0	0	0	0	0	0	0	10

Table 3: Confusion matrix for BoW image classification using a CNN.

The accuracy of the CNN image classifier was 92.5%. The CNN was much more accurate at classifying images than BoW. This is likely because CNNs can extract a more diverse feature set than BoW.