HACETTEPE UNIVERSITY
COMPUTER ENGINEERING DEPARTMENT
BBM 418 COMPUTER VISION LABORATORY
PROGRAMMING ASSIGNMENT-1

Image Retrieval Basics

MELTEM TOKGÖZ 21527381

April 14, 2019

1 PART - SIMPLE IMAGE REPRESENTATIONS

In this assignment, I found the categories of query images given to us and calculate the accuracy by applying different algorithms.(Gabor filter bank, Average SIFT, BoW, BoW with spatial tiling)
Dataset Information: The dataset I use contains the following: There are 10 object categories that is selected from. In the training set, you have 30 images whereas in the query set you have 5 images for each object category. Therefore, you will calculate the average and class based accuracies over 50 query images. Note that, all the images are named with a related category number.
Now let's talk about how I applied these algorithms and analyzed with achieved results.

1.1 a) Gabor Filter Bank

Gabor filter is one of the important methods used in image analysis. With the help of the filter, the distances extending to a certain direction on an image can be detected. Thanks to this feature, plate recognition, fingerprint recognition, iris recognition, face recognition, such as image processing techniques are used in many methods. We use the gabor filter bank to categorize images in this assignment.

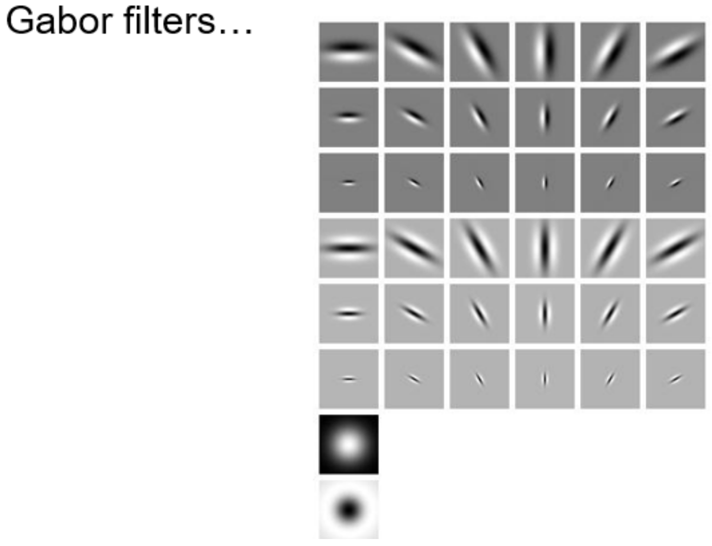


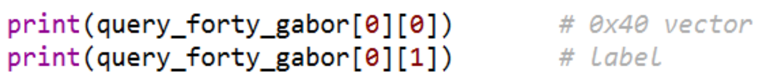
Figure 1.1: Gabor Filter

The functions and explanations I made in this step are as follows:

1. **build filters() function:** In this function, 40 gabor filters were produced and an array called filters was added.The function returns this array. There are certain parameters that controls how Gabor filter will be and which features will it respond to.A 2D Gabor filter can be viewed as a sinusoidal signal of particular frequency and orientation, modulated by a Gaussian wave.
   The gabor filters are manufactured in different ways by giving different parameters.At the end of this section you will see a comparison of different gabor filters and their results.Let me explain exactly what these parameters mean.

Lambda: Wavelength of the sinusoidal component.
Theta: The orientation of the normal to the parallel stripes of Gabor function.
Psi: The phase offset of the sinusoidal function.
Sigma: The sigma/standard deviation of the Gaussian envelope
Gamma:The spatial aspect ratio and specifies the ellipticity of the support of Gabor function.

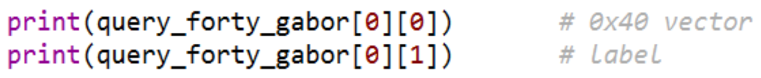
2. **process(img, filters):**
   In this function, gabor filters are applied for each image. The mean amplitude of each filtered image is calculated and added to the fourtymeanvector, so that a vector of 1x40 is obtained for each image. The function returns this vector.I used to built-in functions for creating a gabor filter, doing convolution.
3. **train gabor(train forty gabor):**
   In this function, each train image is read and the filters created for each image are applied (filter application function is call here.) With the 1x40 vector rotating in the filter reading function, the name of the image has been added to an arrand and it is added to the train forty gabor array to contains all the train pictures in the array. The function train returns the forty gabor array.



```
print(query_forty_gabor[0][0])        # 0x40 vector
print(query_forty_gabor[0][1])        # Label
```

Figure 1.2: train forty gabor

4. **query gabor(query forty gabor):**
   The same function as the 3rd step is applied this time for query imagery.

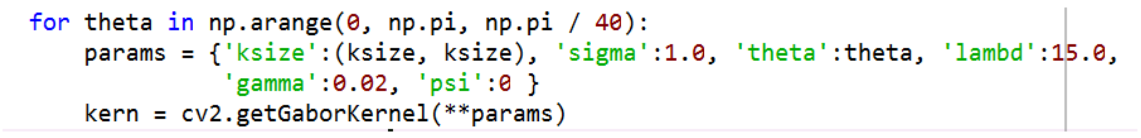


```
print(query_forty_gabor[0][0])        # 0x40 vector
print(query_forty_gabor[0][1])        # Label
```

Figure 1.3: query forty gabor

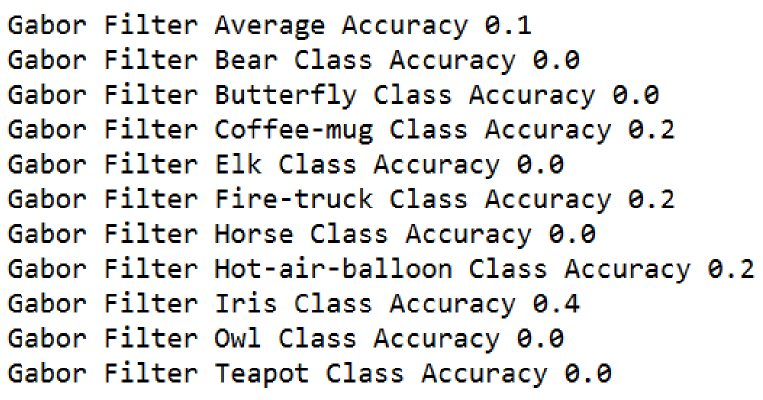
5. **accuracy gabor(train forty gabor,query forty gabor):**
   Using the 1x40 vectors from the above functions result, I applied the k-nn algorithm between each query picture and the train images (k = 1). Then I made mean and class based accuracy calculations based on the found label of each query image.

The results of the 3 different filter banks I tried and the parameters I used were shown in figures 1.3, 1.4, 1.5, 1.6, 1.7, 1.8.According to these results, the best result is the filter bank 3 and I used it in my code.



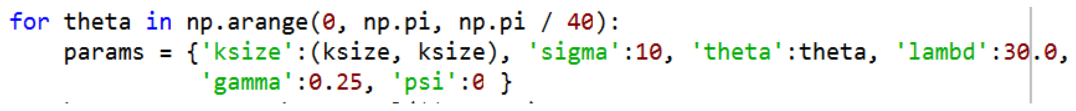
```
for theta in np.arange(0, np.pi, np.pi / 40):
    params = {'ksize':(ksize, ksize), 'sigma':1.0, 'theta':theta, 'lambd':15.0,
              'gamma':0.02, 'psi':0 }
    kern = cv2.getGaborKernel(**params)
```

Figure 1.4: Gabor Filter 1



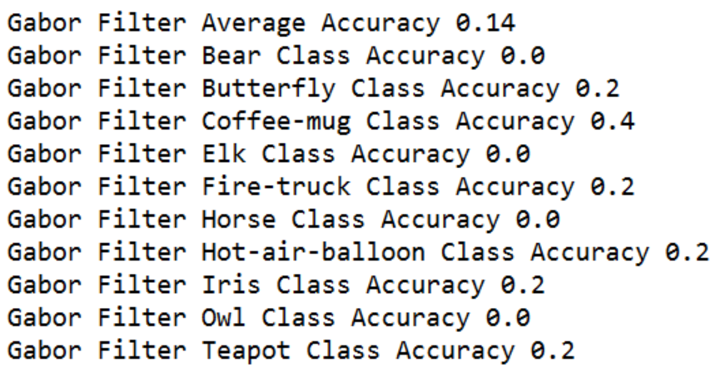
```
Gabor Filter Average Accuracy 0.1
Gabor Filter Bear Class Accuracy 0.0
Gabor Filter Butterfly Class Accuracy 0.0
Gabor Filter Coffee-mug Class Accuracy 0.2
Gabor Filter Elk Class Accuracy 0.0
Gabor Filter Fire-truck Class Accuracy 0.2
Gabor Filter Horse Class Accuracy 0.0
Gabor Filter Hot-air-balloon Class Accuracy 0.2
Gabor Filter Iris Class Accuracy 0.4
Gabor Filter Owl Class Accuracy 0.0
Gabor Filter Teapot Class Accuracy 0.0
```

Figure 1.5: Gabor Filter 1 Result



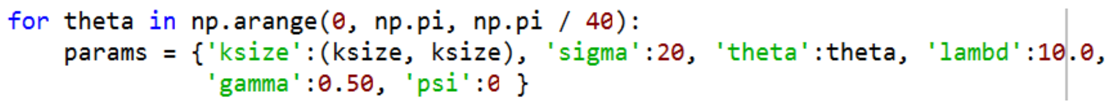
```
for theta in np.arange(0, np.pi, np.pi / 40):
    params = {'ksize':(ksize, ksize), 'sigma':10, 'theta':theta, 'lambd':30.0,
              'gamma':0.25, 'psi':0 }
```

Figure 1.6: Gabor Filter 2



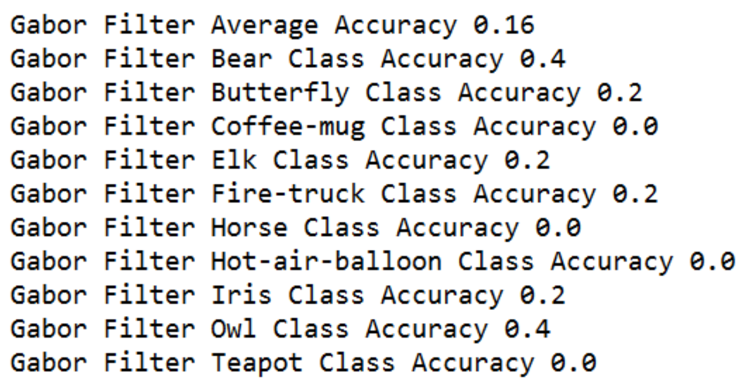
```
Gabor Filter Average Accuracy 0.14
Gabor Filter Bear Class Accuracy 0.0
Gabor Filter Butterfly Class Accuracy 0.2
Gabor Filter Coffee-mug Class Accuracy 0.4
Gabor Filter Elk Class Accuracy 0.0
Gabor Filter Fire-truck Class Accuracy 0.2
Gabor Filter Horse Class Accuracy 0.0
Gabor Filter Hot-air-balloon Class Accuracy 0.2
Gabor Filter Iris Class Accuracy 0.2
Gabor Filter Owl Class Accuracy 0.0
Gabor Filter Teapot Class Accuracy 0.2
```

Figure 1.7: Gabor Filter 2 Result



```
for theta in np.arange(0, np.pi, np.pi / 40):
    params = {'ksize':(ksize, ksize), 'sigma':20, 'theta':theta, 'lambd':10.0,
              'gamma':0.50, 'psi':0 }
```

Figure 1.8: Gabor Filter 3



```
Gabor Filter Average Accuracy 0.16
Gabor Filter Bear Class Accuracy 0.4
Gabor Filter Butterfly Class Accuracy 0.2
Gabor Filter Coffee-mug Class Accuracy 0.0
Gabor Filter Elk Class Accuracy 0.2
Gabor Filter Fire-truck Class Accuracy 0.2
Gabor Filter Horse Class Accuracy 0.0
Gabor Filter Hot-air-balloon Class Accuracy 0.0
Gabor Filter Iris Class Accuracy 0.2
Gabor Filter Owl Class Accuracy 0.4
Gabor Filter Teapot Class Accuracy 0.0
```

Figure 1.9: Gabor Filter 3 Result

You can also see the pictures that are 5 closest to the 3 query images I randomly selected for the gabor filter algorithm in my code.

1.2 b) Average SIFT

SIFT property is a selected image region (also known as the parent point) with an associated identifier. Important points are removed by the SIFT detector and the identifiers are calculated by the SIFT identifier.In this section, descriptors have been determined by using SIFT and the descriptor vector has been created by means of averaging them. Then the classification is done for the query pictures with the k-nn algorithm.

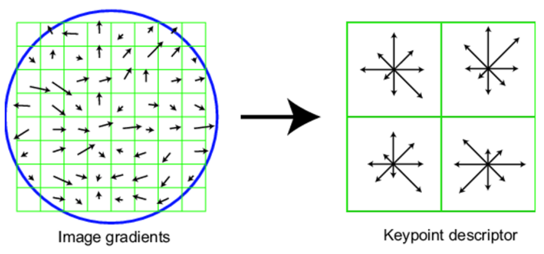


Figure 1.10: SIFT

1. **ToGray(color img)**
   I used to convert images in RGB format to gray (to make them colorless).Takes image as parameters.
2. **sift features(gray img)**
   By using SIFT i (with opencv library), each image has a descriptors, key point.Directly I find keypoints and descriptors in a single step with the function, sift.detectAndCompute().It also returns a vector of 1x128 with the mean of the descriptors.

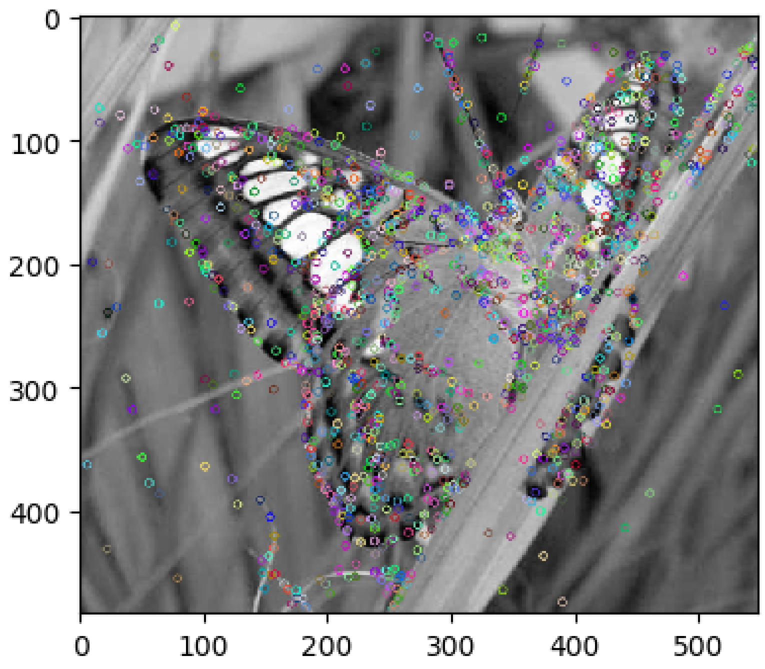
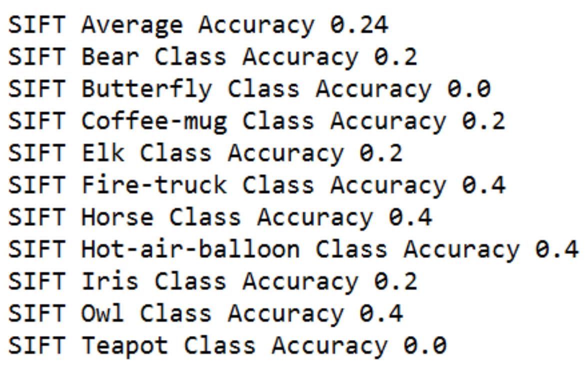


Figure 1.11: Example SIFT descriptor

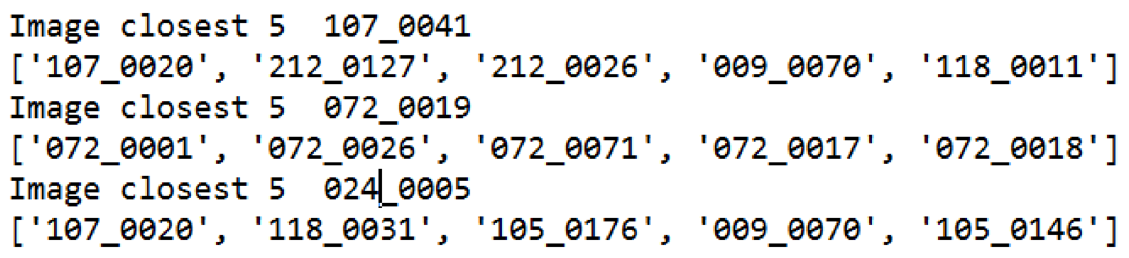
3. **train sift mean():** Returns an array of [1x128 mean shift descriptor vector, image name] for each of the train images. (The functions that image to convert to gray and the sift features function are call in this section.)
4. **query sift mean():** The same as in step 3 is work done, for query images.
5. **accuracy sift(train mean desc,query mean desc):**
   Using the 1x128 mean descriptor vectors from the above functions result, I applied the k-nn algorithm between each query picture and the train images (k = 1). Then I made mean and class based accuracy calculations based on the found label of each query image.
   The results of this process are shown in Figure 1.10.



```
SIFT Average Accuracy 0.24
SIFT Bear Class Accuracy 0.2
SIFT Butterfly Class Accuracy 0.0
SIFT Coffee-mug Class Accuracy 0.2
SIFT Elk Class Accuracy 0.2
SIFT Fire-truck Class Accuracy 0.4
SIFT Horse Class Accuracy 0.4
SIFT Hot-air-balloon Class Accuracy 0.4
SIFT Iris Class Accuracy 0.2
SIFT Owl Class Accuracy 0.4
SIFT Teapot Class Accuracy 0.0
```

Figure 1.12: SIFT Result

Also, the pictures that come closest to the 3 query pictures I randomly selected are as follows for SIFT algorithm.You can view pictures when you run the code.



```
Image closest 5  107_0041
['107_0020', '212_0127', '212_0026', '009_0070', '118_0011']
Image closest 5  072_0019
['072_0001', '072_0026', '072_0071', '072_0017', '072_0018']
Image closest 5  024_0005
['107_0020', '118_0031', '105_0176', '009_0070', '105_0146']
```

Figure 1.13: SIFT closest 5 image for random 3 query image

2 PART - Bag of Visual Words

Bag of visual words (BOVW) is commonly used in image classification.We have the same concept in bag of visual words (BOVW), but instead of words, we use image features as the “words”. Image features are unique pattern that we can find in an image. The general idea of bag of visual words (BOVW) is to represent an image as a set of features. Features consists of keypoints and descriptors. Keypoints are the “stand out” points in an image, so no matter the image is rotated, shrink, or expand, its keypoints will always be the same. And descriptor is the description of the keypoint. We use the keypoints and descriptors to construct vocabularies and represent each image as a frequency histogram of features that are in the image. From the frequency histogram, later, we can find another similar images or predict the category of the image.

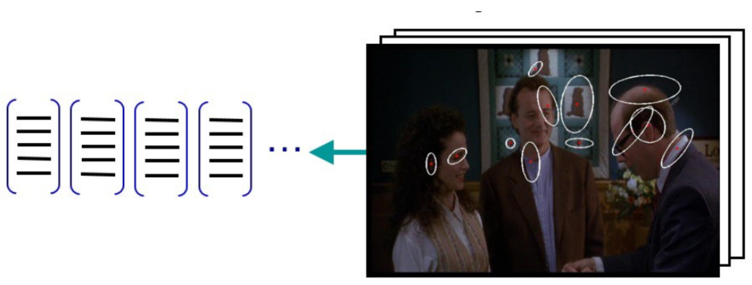


Figure 2.1: Detecting features and extracting descriptor

In this assignment, I used descriptors of BoW pictures and created histograms. I made the accuracy calculations for the categories according to these histograms.

2.1 Step 1 - Codebook

1. **sift desc vector():**
   In order to create the codebooks at this stage, I created a nx128 vector by extending the sift descriptors.
2. **codebook(k, des):**
   In this function, I applied the k-means algorithm to the vector nx128 I created. (According to the desired k value)K is number of centroid and the center of each cluster will be used as the codebook’s vocabularies. So I created the codebooks.

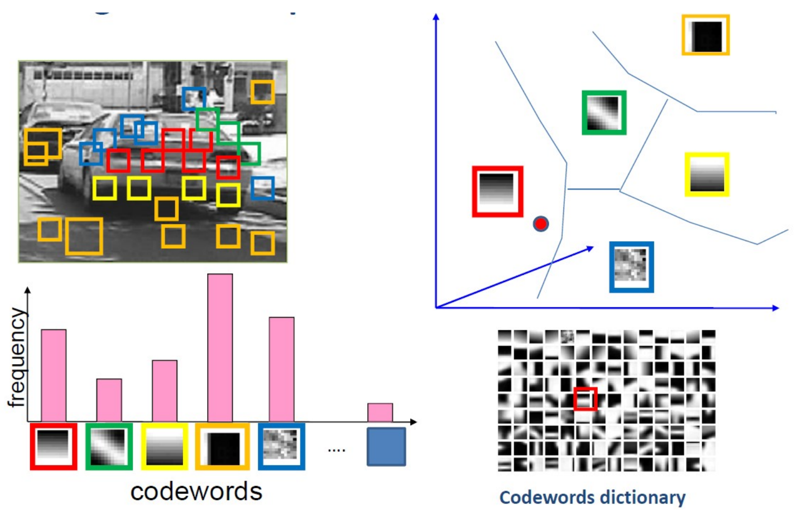


Figure 2.2: Create CodeBook and Histogram

2.2 Step 2 - Quantization

I send the codebooks as a parameter to this function and compared these code books with the descriptors of each query image (k-nn) and created histograms. There is an array of 1xk, which consists of 0s that it originally created (because there are cluster centers up to k).
Then I increased the value on the histogram by the center, which is closer codebook for each descriptor. So, I calculated the frequencies.To do this, I used the two functions I mentioned below.

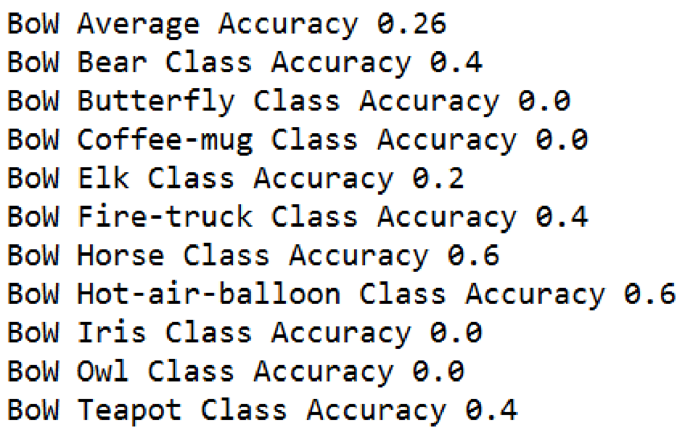
1. **histogram query(codebk):**
2. **histogram train(codebk):**

2.3 Step 3 - Histogram

In this step, I applied the k-nn algorithm between the histograms I created for the query and training images and calculate accuracy.Here I observed that the k value increases as the average accuracy increases.When the k value increases, I think that there is an increase in the number of visual words in codebook. So, average accuracy is increased.

1. **accuracy bow(all qHistogram,all tHistogram):**

Here are some of the results of BoW.



```
BoW Average Accuracy 0.26
BoW Bear Class Accuracy 0.4
BoW Butterfly Class Accuracy 0.0
BoW Coffee-mug Class Accuracy 0.0
BoW Elk Class Accuracy 0.2
BoW Fire-truck Class Accuracy 0.4
BoW Horse Class Accuracy 0.6
BoW Hot-air-balloon Class Accuracy 0.6
BoW Iris Class Accuracy 0.0
BoW Owl Class Accuracy 0.0
BoW Teapot Class Accuracy 0.4
```

Figure 2.3: k=100 accuracy

k value	Average Accuracy
2	% 18
10	% 24
50	% 26
100	% 26
250	% 28

Table 2.1: BoW Algorithm Result

NOTE: I couldn't try with k = 500 because the time was too long and the hardware was not enough.

3 Compare

When I tried different filters in the gabor, I saw that the changed accuracs came in. When the filters were changed, the accuracy of the algorithm is changing. I chosed the best parameters from what I tried.Among the algorithms I tried, I observed that the lowest result was the gabor. It was also average as the working time.
I have observed that Bow gives better accuracy than other algorithms. However, I see slow as a disadvantage because it works very slowly at high k values (ex. K = 250, 500 and etc.). On the other hand, if we make a comparison, I think sift is the best algorithm that gives close results to BoW in a much shorter time. In the algorithms I tried, the fastest-running sift.Accuracy gave good accuracy even though it was not as good as BoW. Overall, the accuracy was lower than I expected.

Algorithm	**Average Accuracy**	**Speed**
Gabor Filter	% 16	medium
SIFT	% 24	high
BoW	% 26	low

Table 3.1: Algorithms Average Accuracy

NOTE: MY TEACHER, I TRIED TO DO SPATIAL TILING, BUT I COULDN'T.THAT'S WHY I CAN'T DO THE ANALYSIS ON THEM. ONLY MY SPATIAL TILING WAS MISSING IN MY ASSIGNMENT.