GZ720J: Homework 2 - Bag-of-Words for Scene Classification

Name - Jiaxin Chen

October 3, 2016

1 Warming up with some theory (10pts)

Question 1.1 (3pts, 2-3 lines)

Given an $N \times N$ image, and an $h \times h$ Gaussian filter, we can convolve the image and the filter in $O(h^2)$ time:

$$C[m,n] = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} I[j,k]G[m-j,n-k] \tag{1}$$

We can seperate the $h \times h$ Guassian filter to $1 \times h$ row vector g_{row} and $h \times 1$ col vector g_{col}

$$G[j,k] = g_{row}[k] * g_{col}[j] \tag{2}$$

Hence, we can convolve the image and the Gaussian filter in $O(h)$ time:

$$C[m,n] = \sum_{j=0}^{h-1} \left\{ \sum_{k=0}^{h-1} g_{row}[k]I[m-j,n-k] \right\} g_{col}[j] \tag{3}$$

Question 1.2 (2pts, 2-3 lines)

No, the bag-of-words doesn't respect spatial information.
Because it represents an image as an orderless collection of local features, which means that it disregards all information about the spatial layout of the features.

Question 1.3 (5 pts, 2-3 lines)

Because different filter responses can remove different unwanted infomation of the image and extract all kinds of useful features correspondingly. We better utilize different features of the image to complete different tasks rather than single raw pixel values, which can make our work much more efficient and effective.

2 Representing the World with Visual Words (40pts)

Question 2.1 (5 pts, 3-4 lines) Theory:

1) Gaussian Filter is the first getGaussianFilter function. It's seperable and picks up the low frequency components in the image.

2) Laplacian Filter is the second getLOGFilter function. It's not-separable and and approximated by a difference of Gaussian. It's often applied to an image that has first been smoothed with something approximating a Gaussian smoothing filter in order to reduce its sensitivity to noise.

3) Derivative of Gaussian Filter is the third filterDerivative function. It's seperable and the output of convolution is gradient at scale σ.

Question 2.2 (15 pts)

Coding question, put your implementation in `baseline/getFilterBankAndDictionary.m`

Question 2.3 (5 pts, 3-4 lines) Theory

The dictionary size K can affect the stability and the size of clusters for the representation of the bag-of-words pipeline.
If we set K too small, for example k = 10, the k-means will group clusters together, the region boundary of clusters will be in low density.
If we set K too large, for example k = 10000, the true cluster has to split into several smaller clusters, which leads to instability. And the region boundary of clusters will be in high density.

Question 2.4 (15 pts)

Coding question, put your implementation in `baseline/getVisualWords.m`

3 Building a Recognition System (95pts)

Question 3.1 (10 pts)

Coding question, put your implementation in `baseline/getImageFeatures.m`

Question 3.2 Extra credit (5 pts 2-3 lines) Theory:

Because if we normalize the histogram of visual words, the size of the image won't change the bag of feature magnitude dramatically.

Question 3.3 Extra credit (5 pts 2-3 lines) Theory

Because different visual words have different influences on the image. Some of them are very important, so they need weight largely. Some of them may be just negligible, so they can be weighted smaller. That's the reason why we need weight different levels of the histogram differently.

Question 3.4 (20 pts)

Coding question, put your implementation in `baseline/getImageFeaturesSPM.m`

Question 3.5 (10 pts)

Coding question, put your implementation in `baseline/distanceToSet.m`

Question 3.6 (5 pts)

Coding question, put your implementation in `baseline/createHistograms.m`

Question 3.7 (5 pts)

Coding question, provide any helper code you wrote for this section and list the files here. Also make sure your `.mat` is included in your submission.

The Helper Code:

1) `guessImage_implementation.m` : Randomly select certain number of testImages and invoke `guessImage.m` function to guess the name of the testImage.

2) `plotAccurary.m` : Plot the graph of k vs classification accuracy

Question 3.8 (10 pts)

Coding question, put your implementation in `baseline/knnClassify.m`

Question 3.9 (10 pts)

Coding question, put your implementation in `evaluateRecognitionSystem.m`

I test a series of k and list several representing confusionMatrix below:

k = 1

```
confusionMatrix =
    28     0     0     0     1     0     0     0     0
     0    25     4     0     0     5     0     1     0
     0     4    14     0     0     8     0     3     2
     1     0     0    32     5     0     2     0     0
     4     0     0     2    36     0     1     0     1
     0     8     2     0     0     8     0     7     1
     0     1     0     2     4     0    24     2     1
     0     1     0     0     0     7     0    21     2
     0     2     3     0     1     2     0     3    40
```

k = 2

```
confusionMatrix =
    29     0     0     0     0     0     0     0     0
     0    31     2     0     0     1     0     1     0
     0     9    17     0     0     4     0     1     0
     2     2     1    29     4     0     2     0     0
     6     0     0     2    35     0     0     0     1
     0    10     6     0     0     6     0     3     1
     1     1     0     6     4     0    21     1     0
     1     2     3     0     0    11     0    13     1
     0     3     6     0     2     0     1     3    36
```

k = 5

```
confusionMatrix =
    29     0     0     0     0     0     0     0     0
     0    32     1     0     0     1     0     1     0
     0     5    15     0     0     7     0     2     2
     1     0     0    30     7     0     1     0     1
     1     0     0     1    39     0     0     2     1
     0    11     2     0     0     8     0     3     2
     0     0     0     1     5     0    26     1     1
     2     2     0     0     0     6     0    20     1
     0     3     1     0     1     1     0     3    42
```

k = 15

```
confusionMatrix =
    28     0     0     0     1     0     0     0     0
     0    30     2     0     0     2     0     1     0
     0     7    17     0     0     4     0     1     2
     2     1     0    30     4     0     2     0     1
     1     0     0     1    40     0     0     2     0
     0    11     2     0     0    10     0     0     3
     0     0     1     1     5     1    25     1     0
     1     2     0     0     0     4     0    22     2
     0     1     2     0     0     0     0     4    44
```

k = 32

```
confusionMatrix =
    27     1     0     0     0     0     0     0     1
     0    33     2     0     0     0     0     0     0
     0    12    13     0     0     1     0     4     1
     1     2     0    28     7     0     1     0     1
     1     0     0     0    41     0     0     2     0
     0    13     6     0     0     4     0     1     2
     0     0     0     0     7     0    25     0     2
     0     2     0     0     0     1     0    26     2
     0     1     3     0     0     0     0     3    44
```

k = 64

```
confusionMatrix =
    26     1     0     0     0     0     0     0     2
     0    31     3     0     0     0     0     1     0
     0    13    15     0     0     1     0     1     1
     1     2     0    24    10     0     1     1     1
     1     0     0     0    41     0     1     1     0
     0    14     5     0     0     4     0     1     2
     0     2     0     0     7     0    23     0     2
     0     2     1     0     0     2     0    24     2
     0     1     2     0     0     1     0     4    43
```

Question 3.10 (10 pts)

I test several number for k to calculate the accuracy. The corresponding data lists in Table 1.

k	1	2	3	5	8	10	12	15	32	64
Accuracy	0.7103	0.6760	0.7290	0.7508	0.7352	0.7477	0.7570	0.7664	0.7508	0.7196

Table 1: **Accuracy versus k values.**

In my case, the accuracy increases with the increasement of k roughly. When k = 15, the accuracy reaches to the maximum 76.64%. And the accuracy decreases with the increasement of k.

My thoughts about the variation of accuracy:
If the k is too small, it will reduce the precision of classifier and amplify the noise data. If the k is too big and the sample number of the class is small, it will include many irrelevant samples in other classes. It will increase the noise and decrease the accuracy. Therefore, k = 15 is the best choice to get the maximal accuracy 76.64% for my classifier.

I implement the `plotAccuracy.m` to plot the graph of k vs the classification as Figure 1.

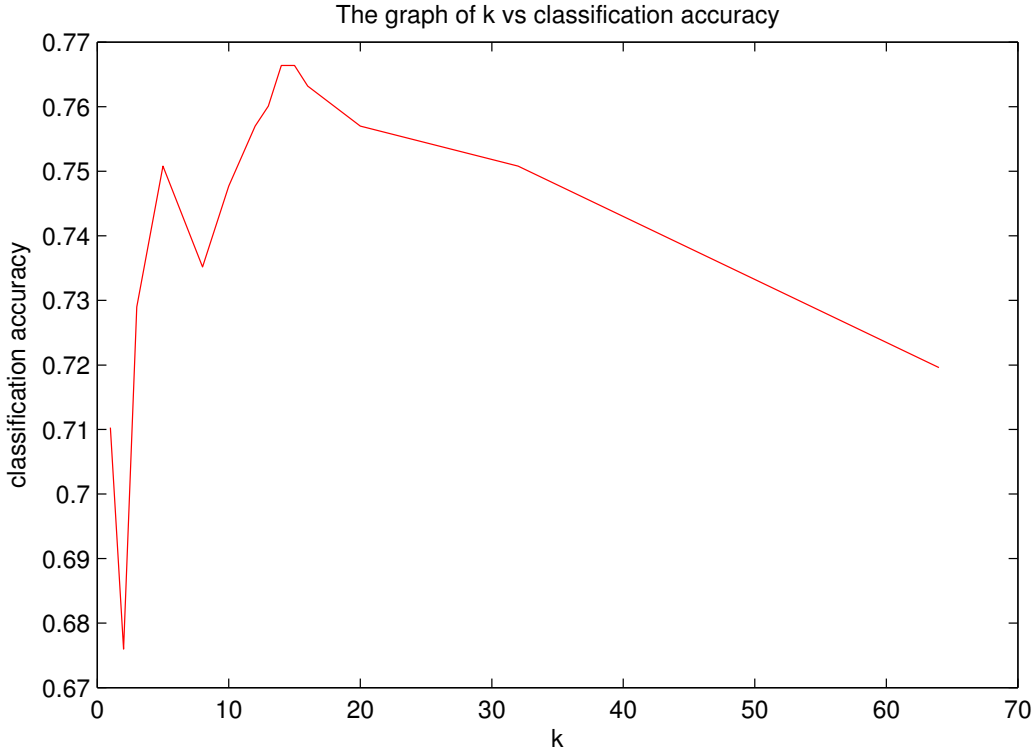

Figure 1: **k vs classification accuracy**

Question 3.11 (5 pts, 2-3 lines): Theory

In knnClassify, we need traverse all the histogram intersection by using sort and k labels by mode function in MATLAB. If the dataset is large, it's more expensive to compute the predictedLabel. And if we choose k to be large reletively in order to get the best accuracy, we'll spend even more time to compute.

4 Final thoughts on visual words (40 pts)

Question 4.1 (20 pts)

Coding question, put your implementation in `visualizeWords.m`

My understanding about visualization:
For each pixel in the image, the wordMap contains the index that corresponds the closest visual word in the dictionary. When we traverse each pixel in the image, we can get the certain 9 $\times$ 9 $\times$ 3 patch. And we take the center pixel as the visual word of this patch. We need add the patch to the corresponding visual word patch and count the number of corresponding visual words. We accumulate the patch for every image and then we can normalize them as the average patch. Therefore, we can regard the averaged patch as the visualizations for each visual word in the dictionary.

My visualization figure are presented as Figure 2.

Figure 2: **Visualizing the Visual Words**

Question 4.2 (10 pts)

The Helper Code:

1. `top10VisualWords.m` : Generating the top 10 visual words for chosen image.
2. `top10VisualWords_imagePath.m` : The image path of chosen 2 images from each class.
3. `Q4.2 Directory` : This directory contains the chosen images, their wordMap as well as the corrsponding top 10 visual words.

The explanation of top 10 visual words:
The histogram can plot the frequency of the index to the dictionary. When we sort the histogram and get the top 10 index in descending order, we can obtain the top 10 patch, which means the corresponding top 10 visual words appear most times in the image with the top 10 highest frequency.

I list my chosen images and their corresponding top 10 visual words as Figure 3.

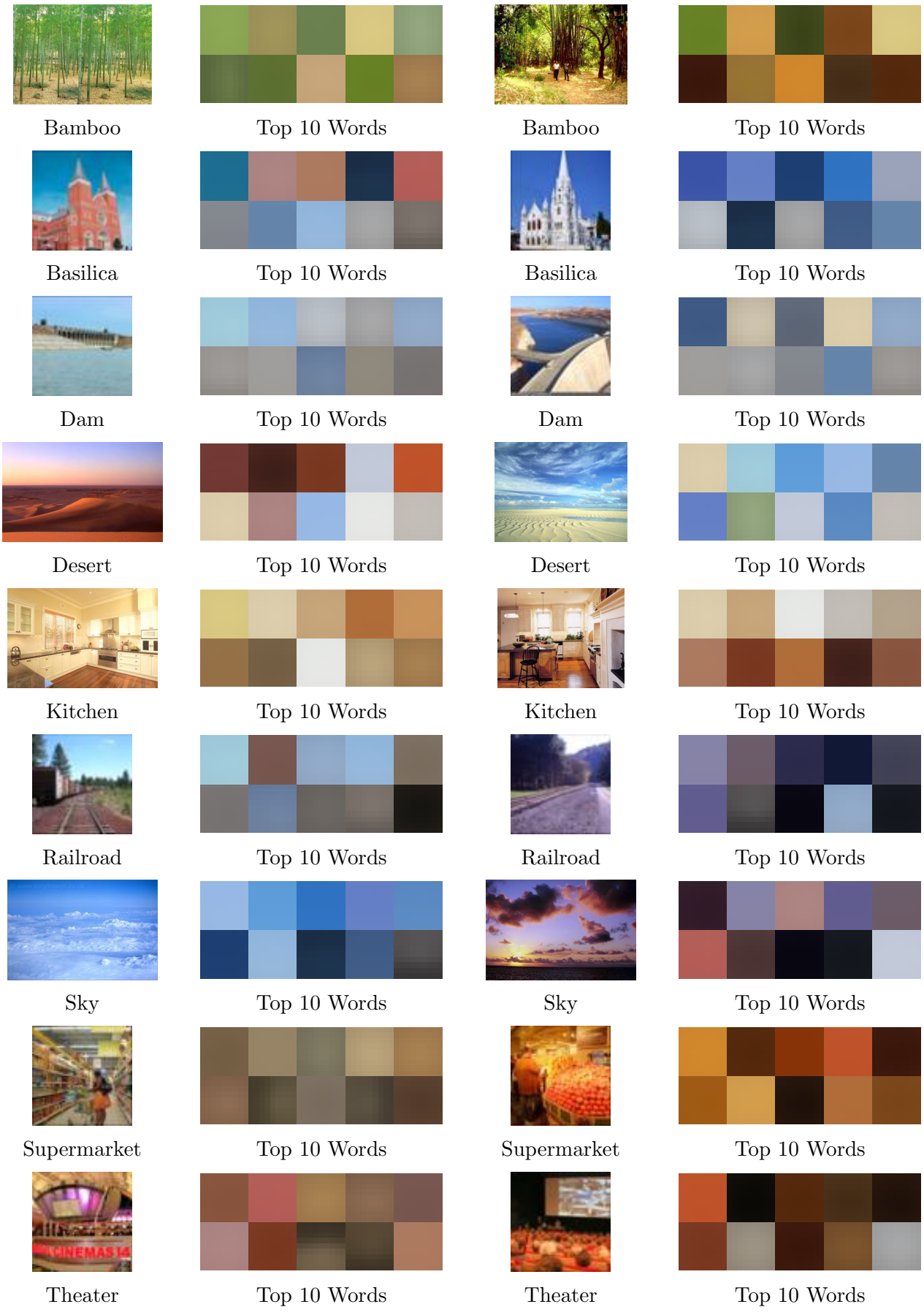

Figure 3: Input images and visualizations of top 10 words

Question 4.3 (10 pts, 4-5 lines): Dataset Expansion

The benefit of a bigger dataset: With bigger dataset, the classification model can avoid overfitting problem. And the more complex the classification is, the bigger VC dimension it has. According to the model complexity penalty, we need expand our data to reduce the model complexity penalty, which can get better generalization. Therefore, the training system can deal with more different situation and make more correct prediction.

Flipping can definitely be used as expanding the image dataset. Because the pixel in the certain location is usually different from the origin one. The training system will regard it as a new image.

Other ways to expand the dataset:

1) Rescaling the image

2) Zooming inside the image

3) Rotating the image in the certain degree

4) Shifting the image in width/height(translate the image vertically or horizontally)

5) Padding the image or filling in newly created pixels, which can used after a rotation or a width/height shift.

The ways of expanding the dataset without using external images can also work under the bag of words representation and bag of words with spatial pyramid matching. With bigger dataset, they can avoid overfitting and achieve better generalization.

5 Extra Credit: Boosting performance (45 pts)

Question 5.1 Better filter bank (10 pts)

I add two filters in the `improved/getFilterBankAndDictionary.m`.

1. Prewitt Horizontal Edge-emphasizing Filter This 3×3 filter emphasizes horizontal edges by approximating a vertical gradient.

2. Sobel horizontal edge-emphasizing filter This 3×3 filter that emphasizes horizontal edges utilizing the smoothing effect by approximating a vertical gradient.

   I create `improved/plotFilterBank.m` to plot these two filters as Figure 4.

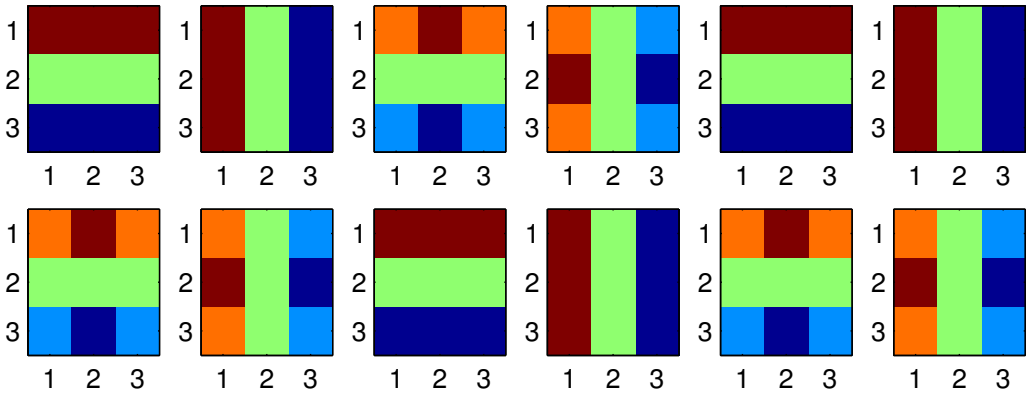

Figure 4: **Prewitt Filter and Sobel Filter**

Question 5.2 Better image similarity function (5 pts)

I modified `improved/distanceToSet.m` by using L Norm Euclidean. The basic concepts for my implementation:

1) L1 Norm Euclidean, namely Manhattan distance:

$$\| \mathbf{p} - \mathbf{q} \|_1 := \sum_{i=1}^{n} |\mathbf{p_i} - \mathbf{q_i}| \tag{4}$$

2) L2 Norm Euclidean:

$$\| \mathbf{p} - \mathbf{q} \|_2 := \sqrt{\sum_{i=1}^{n} |\mathbf{p_i} - \mathbf{q_i}|^2} \tag{5}$$

Question 5.3 Different Features (15 pts)

I utilize Speeded Up Robust Features(SURF) as the feature detector and descriptor. SURF uses square-shaped filters as an approximation of Gaussian smoothing and a blob detector based on the Hessian matrix to find points of interest, which is named the strongest points in MATLAB. I use the strongest points rather than the random pixel to get the filter responses. And it corresponding visual words can represent the image better theoretically.

Therefore, I modified the `improved/getFilterBankAndDictionary.m` to extract features by SURF. But it turns out that if the image's size is too small, the SURF even cannot return enough α strongest points. To solve this problem, I selected the left points randomly and stuff them to the rest of the strongest points matrix.

Besides, I modified the `improved/knnClassify.m` to sord the histInter in ascending order according to the L Norm Elucidean Distance.

I executed `trainSystem.m` and `improved/evaluateRecognitionSystem.m` to calculate the confusion matrix and accuracy. And I tested different parameters and the results are shown as Table 2:

α	**k**	**Selected Points**	**Distance**	**accuracy**
100	15	Strongest Points only	L2 Norm Elucidean Distance	0.5047
150	15	Strongest Points only	L2 Norm Elucidean Distance	0.5078
20	13	Strongest Points only	L2 Norm Elucidean Distance	0.5234
20	15	Strongest Points + Random Points	L2 Norm Elucidean Distance	0.5452
100	15	Strongest Points only	L1 Norm Elucidean Distance	0.7882
20	15	Strongest Points + Random Points	L1 Norm Elucidean Distance	0.7942
20	15	Strongest Points only	L1 Norm Elucidean Distance	0.7944
20	13	Strongest Points only	L1 Norm Elucidean Distance	0.8037
150	20	Strongest Points only	L1 Norm Elucidean Distance	0.8100
150	15	Strongest Points only	L1 Norm Elucidean Distance	0.8193

Table 2: **Accuracy versus different parameters.**

I represent two good results and failure cases respectively as below:

a) Failure Cases:

(1) α = 20, k = 15, Strongest Points + Random Points, L2 Norm ELucidean Distance **accuracy** = 0.5452

```
confusionMatrix =
    29     0     0     0     0     0     0     0     0
     0    30     1     0     0     0     0     4     0
     0     9     5     0     1     3     0    12     1
     1     9     1    18     9     0     0     2     0
    12     0     2     0    22     1     0     7     0
     2    10     0     0     0     3     0    10     1
     4     8     0     1    10     0     8     2     1
     0     0     0     0     0     0     0    31     0
     0     0     2     0     0     0     0    20    29
```

In this case, I choose α reletively small because I trust the strongest points the SURF give me. And for the some pictures with uneoungh strongest points, I just select random points to stuff the rest of alpha points. But the accuracy decreases dramatically, I think the L2 Norm Elucidean Distance cannot measure the distance between wordHist and histograms in the best way. So I changed it to L2 Norm Elucidean Distance and the results get much better, which the accuracy = 0.7942.

(2) $\alpha = 150$, k = 15, Strongest Points only, L2 Norm Elucidean Distance
accuracy = 0.5078

```
confusionMatrix =
    29     0     0     0     0     0     0     0     0
     0    33     1     0     0     0     1     0     0
     0     6    18     0     0     2     0     4     1
     0     1     0    32     5     0     1     1     0
     2     0     0     0    42     0     0     0     0
     0    12     5     0     0     7     0     2     0
     0     0     0     4     6     0    23     1     0
     0     2     0     0     0     2     0    27     0
     0     0     3     0     0     0     0     6    42
```

In this case, I choose the lager α to compute the dictionary. Maybe because the α is too big for L2 Norm Elucidean Distance to compute, which is much more time-consuming and the accuracy decreases dramatically.

b) Good Results:

(1) $\alpha = 20$, k = 13, Strongest Points only, L1 Norm Elucidean Distance
accuracy = 0.8037

```
confusionMatrix =
    29     0     0     0     0     0     0     0     0
     0    34     1     0     0     0     0     0     0
     0    10    16     0     0     2     0     2     1
     1     0     0    32     3     0     2     2     0
     2     0     0     0    42     0     0     0     0
     0    11     2     0     0    10     0     1     2
     2     0     0     4     3     0    25     0     0
     0     3     0     0     0     1     0    27     0
     0     0     2     0     0     1     0     5    43
```

In this case, I improved my algorithm to limit the selected points only in the strongest points range. Thanks to the SURF, even if $\alpha = 20$ is so small, SURF detects the

strongest representative points and only use those points to output our filter responses and dictionary. So we totally get rid of the unuseful random points. And due to small α, my code is much more time-saving and efficiently. And the 80.37% accuracy is satisfying.

(2) $\alpha = 150$, k = 15, Strongest Points only, L1 Norm Elucidean Distance
accuracy = 0.8193

```
confusionMatrix =
    29     0     0     0     0     0     0     0     0
     0    34     0     0     0     0     0     1     0
     0     9    18     0     0     2     0     1     1
     0     0     0    36     2     0     0     2     0
     2     0     0     0    42     0     0     0     0
     0    13     2     0     0     9     0     2     0
     0     0     0     4     5     0    24     1     0
     0     2     0     0     0     1     0    28     0
     0     0     3     0     0     0     0     5    43
```

I keep on increasing α to get more strong points, and the accuracy becomes even better to 81.93%. Maybe because larger α can get much more filter responses and more precise dictionary. But I don't recommand this setting because it's time-consumming and computative expensively.

According to observe the display between the true label and my predicted label, I found out that for the larger size of the image, such as bamboo_forest dataset, I can get more strongest points rather than repeating copying or randomly selecting, so that I can predict the label mush more precisely and get the better accuracy.

To sum up, the improved SURF can extract features amazingly, so that we can choose much smaller α to fulfill even better accuracy, which is more efficient and effective.