Assignment -2-
Computer Vision

Naman Kumar Andrew ID : namank

Q1.0

The filters are basically grouped in to different categories:
1. **Gaussians:** There are 9 different gaussian filters at different scales which are applied on the input image to obtain a filter response.
i) Its impulse response is a gaussian function.
ii) It is used to smooth out the image and remove noise/
2. **Derivatives of Gaussians:** Two types of Derivatives of Gaussians were used:
a) Derivative with respect to x : Total 6 were used at different scales.
b) Derivative with respect to y : Total 6 were used at different scales.
i) It is used to find Edges and sudden changes in intensity in the image.
3. **Laplacian of Gaussian(LOG):** 12 LOG filters were used on the input image.
i) It is used for Blob Detection.
ii) Also, when we need to compare pixels from two image, we compare their laplacian values because it becomes invariant to lighting and illumination when we take the second derivative.

Q1.1

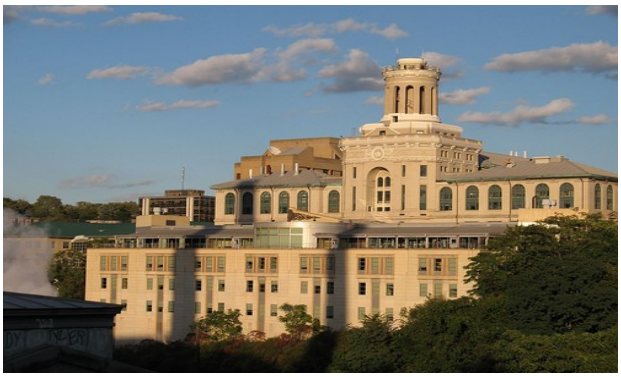
In this question, I created a dictionary from the list of images.
1. First, create a filter bank.
2. Take alpha = 100 and K = 100.
3. Then extracted the filter-response of each image and selected alpha random responses from the response.
4. Repeat it for all images to get final filter response.
5. Finally, use k-means to form the dictionary and save it.

Q1.2

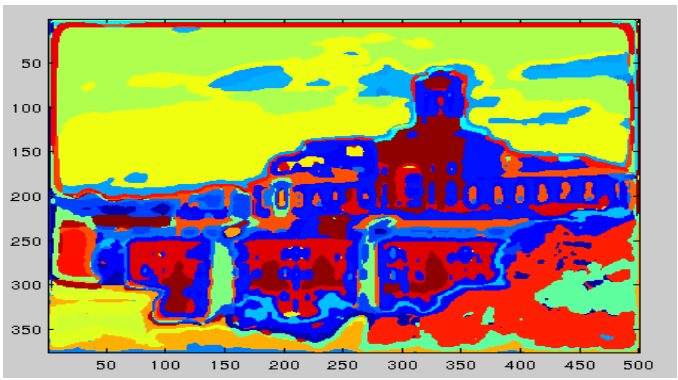
In getVisualWords, we have to create a wordmap corresponding to the image and each value in it should represent the index of the closest visual word which is present in the dictionary.
1. So, here first create a filter response of the image and find the closest visual word using pdist2.
2. Then, replace the value in wordmap with the index of the visual word.
3. Reshape the wordmap to the size of the original image.

I have shown Original Image and its WordMap with K = 100;

Original CMU Image



Word Map



Q2.1

1. In getImageFeatures, we need to see how often each visual word is appearing in the image and the create a histogram to represent that. We will get a K*1 vector where each value represents how many times that particular visual word is appearing in the image.
2. Then , normalize it using L-norm to get the final histogram.

Q2.2

1. In last case, we noticed that it does not contain any spatial information. To overcome this issue, we are going to use Spatial Pyramid Matching.
2. Basically, we will have different layers (layer 0, 1, 2) and in each layer we will be subdividing the image and then computing histograms for each sub-image. For example in layer 2, Image is divided into 16 parts and histogram is computed for each of them.
3. Then concatenate all the histograms from different layers after multiplying each of them with their respective weights to get the final histogram. For example: weight assigned to layer 2 is 1/2. The dimension of the final histogram will be 2100 * 1 if K = 100. Note that the final histogram should be normalized.

Q2.3

In distanceToSet, we have to find the nearest instance from the training set and for that, there are different techniques to find similarity between histograms like chi^2, euclidean, histogram similarity and here histogram similarity technique is used.

Q2.4

1. In buildRecognitionSystem, we need to create a model from training set which can be used to predict the output for any testing image.
2. For that, we create a featureTrs which basically contains all the histograms of all training images and then can be done by computing histogram using getImageFeaturesSPM and then concatinating all the histograms to form featureTrs. Its dimension is 2100*1483 if K = 100 and number of training images is 1483.
3. Then create classTrs which contains labels of all the training images. It will be a 1*1483 vector.
4. Finally, save filterBank, dictionary, featureTrs and classTrs in vision.mat.

Q2.5

In evaluateRecognitionSystem, we have to guess all the testing images and then create a confusion matrix and find accuracy.
1. Firstly, find all the predicted labels using guessImage.
2. Then find the confusion matrix and accuracy.

For alpha = 100 and K = 100, my confusion matrix is

```
Confusion_Matrix =

    13     0     0     0     4     1     0     2
     0    11     0     0     2     0     6     1
     0     1    19     0     0     0     0     0
     1     0     1    15     1     1     1     0
     0     0     0     1    18     0     1     0
     2     2     1     8     0     6     0     1
     0     2     2     0     1     0    15     0
     0     2     0     0     0     0     2    16


Accuracy =

    0.7063
```

Q3.1

1.The perfomance of classifiers is expected to improve if size of the dataset increases. Without using external images and only using the dataset given to us, there are lot of ways to expand the data set.

2. **Left – Right flip of the image.**

a) Flipping the image will not work in case of bag of words representation that is it will not help in expanding the dataset because as we know, bag of words representation looses all the spatial information and it does not know the location of the pixels in the image. So, after flipping the image, BOW representation will consider it as the same image, therefore, it will not help in case of BOW represntation.

b) In case of BOW represntation with Spatial Pyramid Matching, we are considering Spatial Information also by subdividing the image into sub-images and using them to find histograms. So, in this case flipping the image will give us a complete new image because spatially it will be different from the original image. Both images will have a different histogram represntation and can be considered as different images. So, this will help in expanding our dataset under BOW representation with Spatial Pyramid Matching.

3. Apart from flipping the image to get a new image in case of BOW representation with SPM, there can be other ways by which dataset can be expanded:

a) Rotating the image.

b) Scaling and cropping the image.

c) Changing illumination and lighting on the image (maybe by artificially modifying it).

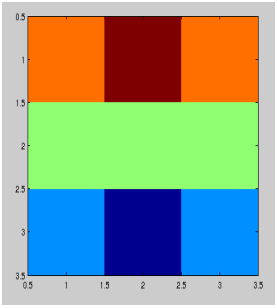
Q3.2

1. Adding more Filters

For the purpose of this question, I added three filters:

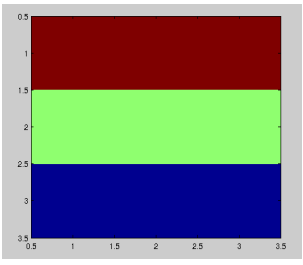
a. Sobel Filter.

It is a differentiation operator which computes the approximation of gradient of the image intensity function.

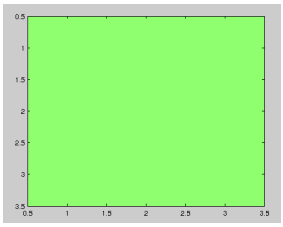


b. Prewitt Filter.

It is almost similar to Sobel Filter and is used particularly for edge detection.



c. Average Filter.



The confusion matrix and accuracy is shown in **3. First Method.**

1. As expected, adding more filters led to the increase in the accuracy because we are adding more filters which will make our filter Bank more diverse.

2. Better Image Similarity Function

Earlier, we used histogram intersection and for this question, I will be using **euclidean Distance** to compute the similarity between two histograms.

```
histInter = 1./sqrt(sum(bsxfun(@minus,wordHist,histograms).^2));
```

Confusion Matrix and Accuracy

```
Confusion_Matrix =

    12     0     0     3     1     2     1     1
     0    14     0     1     0     0     5     0
     0     2    16     2     0     0     0     0
     1     4     1    10     1     1     2     0
     0     3     0     2    11     1     3     0
     2     2     2     8     0     5     1     0
     0     4     0     1     0     0    15     0
     1     5     1     2     0     0     1    10

Accuracy =

    0.5813
```

a) As expected, accuracy will decrease because euclidean does not consider the relative difference between two values.
b) For example, in reality going from 1 to 2 is not same as going from 999 to 1000 but euclidean gives a value of 1 for both the cases which is not good as compared to Histogram similarity intersection or Chi Square distance. These take care of this issue and therefore performs better than Euclidean Distance.

3.

FirstMethod

1.Instead of using Filter Response, I used SIFT to find the descriptor for the interest points which was 128 Dimensional Vector and used these points to form the response for a particular image. Note that I am not using any Filters or FilterResponse in this method and is only using SIFT.

2. After getting response from all the images, I formed the dictionary whose size was 100 * 128 where K = 100.

3. a) Then I used Dense SIFT to create a wordmap for the image.
b) Using SIFT for creating wordmap will not give us good performance because it only selects few interest points and calculate descriptors only at those points and majority of the points are ignored.

Therefore, we will not get a good perspective of the complete scene. As expected, I got an accuracy of 0.4 when using SIFT for creating dictionary and also using SIFT to create the wordmap
c) To overcome this issue, I used Dense SIFT (dsift) to create a wordmap which generates descriptors for large number of points and gives a very good perspective of the complete scene of the image. For scene classification, it is better to use Dense Sift as suggested in [1].
d) Please note that to create a dictionary, I am still using SIFT so that it is less computationally expensive.
e) Once we get the wordmap for all the images, then we Build the recognition System and then run the evaluation to check the results.

4.
The confusion matrix and accuracy is as shown below:

```
Confusion_Matrix =

     5     1     0     4     5     4     0     1
     3    12     0     1     2     0     2     0
     0     0    19     1     0     0     0     0
     0     1     1    17     1     0     0     0
     4     0     0     0    15     1     0     0
     1     2     0     2     5     8     0     2
     0     1     0     2     0     0    16     1
     0     0     0     0     0     0     0    20


Accuracy =

    0.7000
```

5. e) Accuracy in case of using DSIFT for wordmap and using SIFT to generate a dictionary is 0.7 (better than 0.4 which we obtained when using SIFT both for creating dictionary and creating wordmap) because we are now considering large number of points from the image to create a wordmap and ignoring very less number of points. So, we get a very good perspective of the image.
f) Also, there was not much difference in the accuracy calculated using SIFT(/DSIFT) Features and accuracy calculated using Filter Responses.

Second Method

a. Earlier, we used to select alpha random points from the filter response but in this case we will be using SIFT to find the interest points and selecting filter responses corresponding to those interest points.

b. It is expected to give a better performance because we are selecting good (interest) points rather than selecting any alpha random points and we will be using those points to form the dictionary.

c. Confusion Matrix and Accuracy

Note : This is after adding Filter Banks and modifying selection of interest points in case of Filter Responses and using histogram intersection similarity.

```
Confusion_Matrix =

    14     0     0     2     2     2     0     0
     0    14     0     0     1     0     4     1
     0     1    19     0     0     0     0     0
     0     0     1    17     2     0     0     0
     2     2     0     1    15     0     0     0
     2     1     0     8     0     8     0     1
     0     2     1     2     0     0    15     0
     0     0     0     0     0     0     2    18


Accuracy =

    0.7500
```

d. As expected accuracy increased because we are now considering good interest points to form the dictionary.

For the purpose of submission of this question in **custom,** I applied the First Method.

Q4

In Application section, I implemented two things:
1. I took two different data-sets of images:
a) Berkeley Segmentation DataSet (BSDS 500) - (Berkeley)
[http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html]
b) CalTech 101 [http://www.vision.caltech.edu/Image_Datasets/Caltech101/] - (caltech)
Then took 50 images from each dataset and kept 40 of them for Training Set and 10 for Testing set.
So, in total I have 80 training Images and 20 testing images.
2. I took a dataset of 50 images taken by a professional cameraman using a professional camera and 50 images taken by a normal person using a normal camera.
Similarly, 80 were taken as training images and 20 as testing images.
I ran my framework on these two things.

For the purpose of this assignment, I will be demonstrating only the first part where I took two datasets and checked whether the framework distinguishes images from one dataset and images from another.

Firstly, traintest.mat was modified according to the new training set.
Then, I ran the complete framework and finally computed confusion matrix and accuracy and it showed very good accuracy

The confusion Matrix and Accuracy

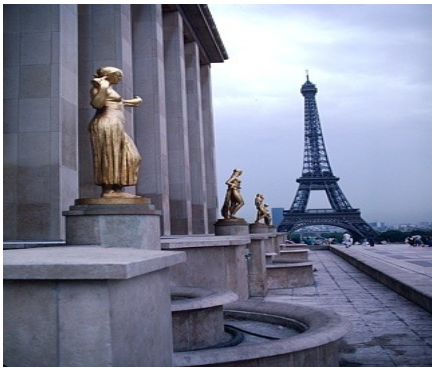
```
Confusion_Matrix =

     9     1
     2     8


Accuracy =

    0.8500
```

1. Here, apart from doing what we did earlier, we can also add some extra information regarding to particular characteristics which belongs to all the images from a particular dataset. For example: All of the images of a particular dataset were taken under certain conditions of lightning, illumination, etc..

Failure Cases:

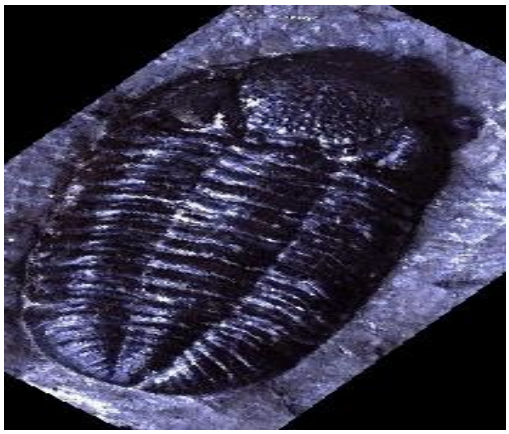
1. Actual = Caltech. Predicted = Berkeley.

2. Actual = Berkeley. Predicted = Caltech



Success Cases:

1. Actual = Caltech. Predicted = Caltech.



2. Actual = Berkeley. Predicted = Berkeley

1. As can be seen from the failure cases, first image was a slight degression from the natural characteristics of the dataset(caltech) and was more similar to the other dataset(Berkeley).

<u>POINTS TO NOTE</u>

1. There are three folders:

a) Baseline : It contains all the files related to Q2.

b) Custom : It contais all the files related to Q3 and apart from that, it also contains a folder vlfeat which I included to use SIFT and DSIFT for Q3. **Please Note: To run Q3, we need to add vlfeat toolbox to the Matlab Path.**

c) Application : It contains a folder **images** which contains images from 2 dataset “Berkeley” and “Caltech”. I used this for Q4.

2. Also, there is a writup describing all the questions from Q1 to Q4.

REFERENCES

1. S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In Computer Vision and Pattern Recognition (CVPR), 2006 IEEE Conference on, volume 2, pages 2169–2178, 2006.