Computer Vision Homework 1

Kai Li

Andrew ID: kail2

September 25, 2016

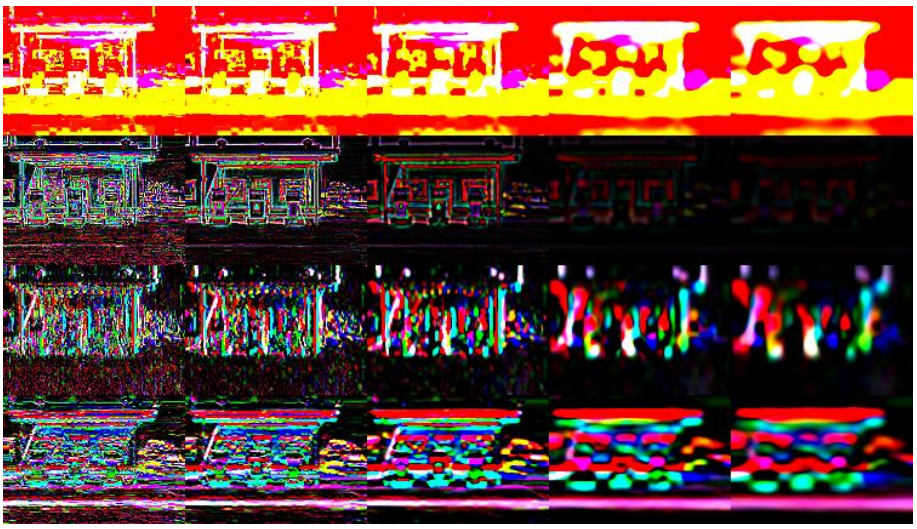
Q1.0

The filterBanks here can be grouped into manily three types:

- The first row's five filters are Gaussian filters, they are low pass filters and blur the picture, which pick up the general intensity in the neighborhood area of pixel in different scales, and thus decrease the noise.
- The second row five filters are Laplacian of Gaussian filters, they pick up the gradient intensity and thus can be used to detect edges or sharp lines.
- The last two rows are directional derivative of Gaussian filter, the third row is vertical and the fourth row is horizontal. These filters can pick up the vertical and horizontal edges around selected pixel.

Q1.1

The following figure shows the responses of 20 filters from a selected sample picture (gas_station\sun_arvnmivbiefcxugl.jpg):

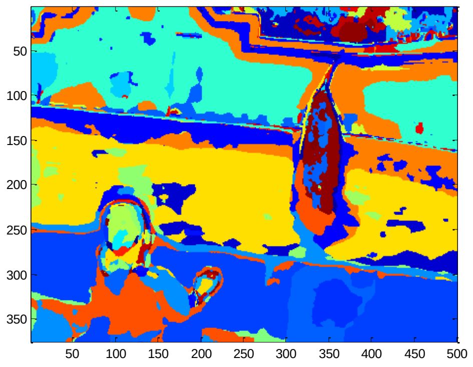


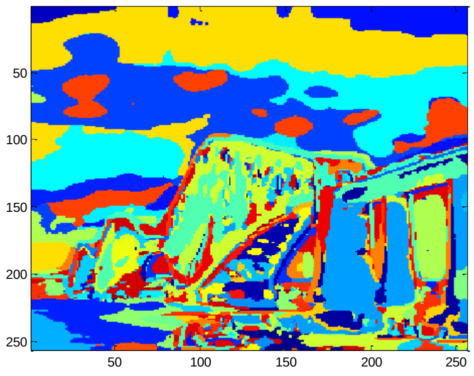
The code can be referred in extractFilterResponses.m file.

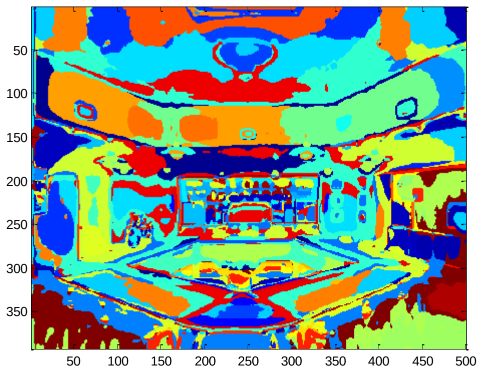
Q1.2

In this part, α is 200 and K is 150, the detailed implementation is in getFilterBankAndDictionary.m file.

Q1.3







The pictures above are selected arbitrarily and we can see the wordMaps of them contain clear features of the original pictures. The edges, different color layers as well as the main objects are extracted from original pictures, and reflected in the wordMaps of them. Please refer to getVisualWords.m file for details.

Q2.1

Please refer the code getImageFeatures.m for details.

Q2.2

Please refer the code getImageFeaturesSPM.m for details.

Q2.3

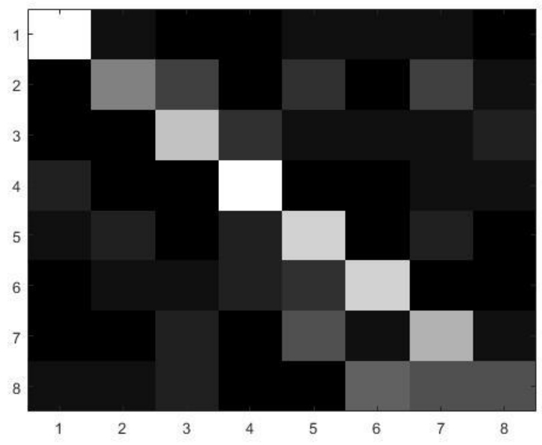
In this implementation, I used bsxfun() instead of for-loop to shorten the time. Please refer the code distanceToSet.m for further details.

Q2.4

Please refer the code buildRecognitionSystem.m for further details.

Q2.5

The confusion matrix is as follows:



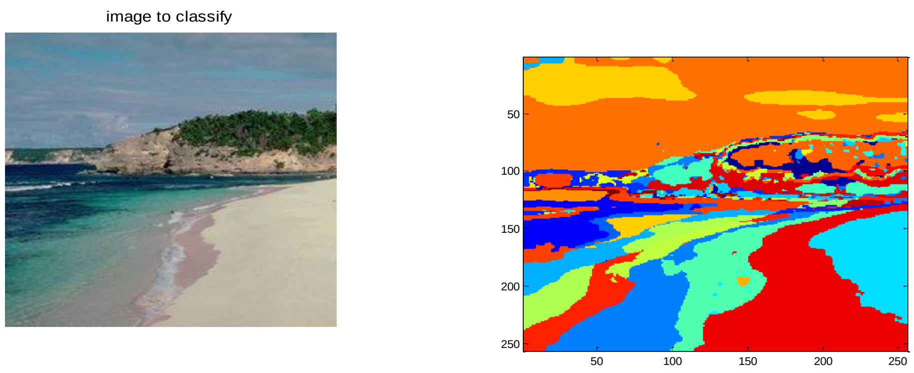
```
confusionMatrix =

    16     1     0     0     1     1     1     0
     0     8     4     0     3     0     4     1
     0     0    12     3     1     1     1     2
     2     0     0    16     0     0     1     1
     1     2     0     2    13     0     2     0
     0     1     1     2     3    13     0     0
     0     0     2     0     5     1    11     1
     1     1     2     0     0     6     5     5
```

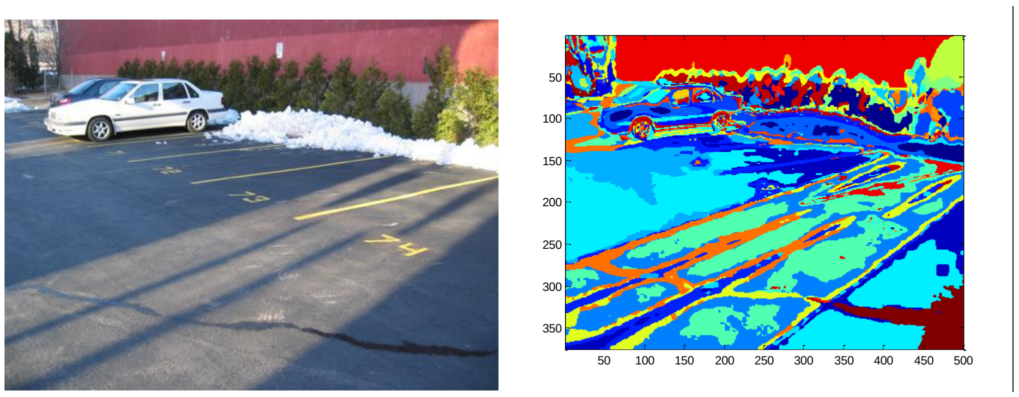
Without extra improvement, the accuracy will reach 58.8%.

Q2.6

From the confusion matrix in Q2.5, we can find class 2(beach) and class 8(waterfall) are the most difficult classes to classify. And pictures in class 'beach' is very likely to be regarded as class 'park-lot'; and 'waterfall' very likely to be mistakenly regarded as 'park'. Here are some false samples:

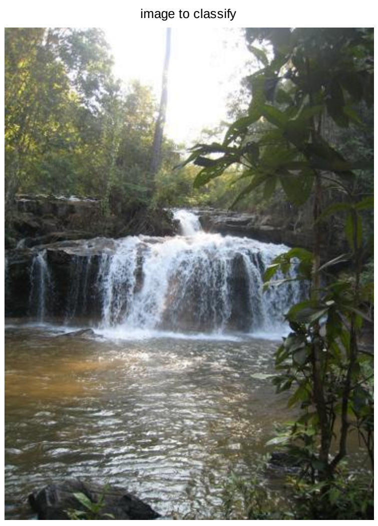


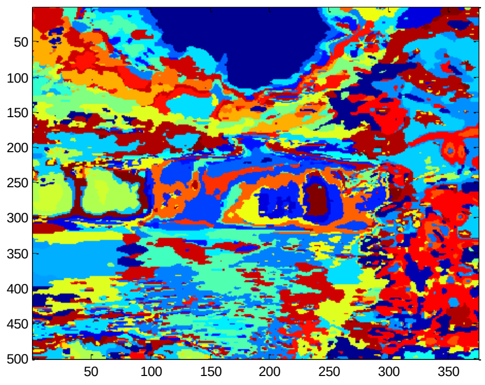
(Image of one beach picture and its wordMap)



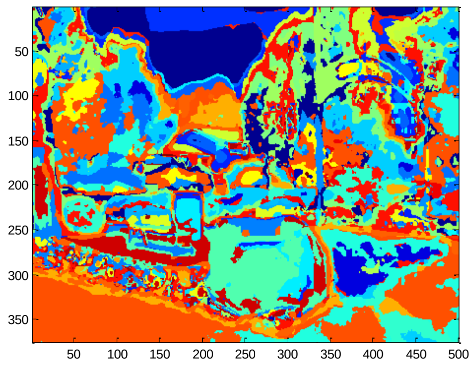
(Image of similar park-lot picture and its wordMap)

The four pictures above are one example of wrong classification between beach and parking-lot. We can see both of the original pictures are flat ground with a front obstacle. In the wordMaps, the flat ground is represented by several colored ribbons and the top part is similar; therefore, it's difficult for computer to distinguish.





(Image of one waterfall picture and its wordMap)



(Image of one park picture and its wordMap)

The four pictures above are one example of wrong classification between waterfall and park. The reasons why these two classes are hard to distinguish are: both are mainly outdoor natural scenery with many colors, which make the wordMaps filled with many small colorful blocks; also the top parts are in many cases blue sky; once the structure of waterfall is similar to park's, it's really hard to distinguish these two from wordMaps even by human eyes.

Q2.7 (extra credit)

Here I applied three methods to improve the system's performance:

- Add Gabor filters. Because we only have four kinds of filters before, the main concern here is maybe if I add different filters to my filterBank, they will extract more useful information and thus increase accuracy. After trying the x-y directional derivative of Gaussian filter, sobel filter and Gabor filter, I found Gabor filter will increase my accuracy. Thus, the first method I applied was to add fixed scale, fixed frequency with eight directions' Gabor filter to my previous filterBank, which significantly increased accuracy by 6%.Please refer to createFilterBank.m in 'custom' folder for details.

- The second thing I did is to change the weighting factors in SPM part. Based on my observation and trial and error, I found the histograms of the biggest layer (the original picture's histogram) and the finest layer (the 4*4 blocks of histograms) are more important than the middle layer (assume the layerNum==3). Therefore, I changed the layer0's weighting factor into 1, the layer1's into 1/4, and the finest layer's into 2. (Note: This doesn't reflected in my code in getImageFeaturesSPM.m because I don't quite know the exact reasons for that but I've done experiment that this alone will enhance my accuracy by 1%)
- The third thing I applied was to enhance the efficiency of my program, in distanceToSet.m, I used bsxfun() to largely save time to compare histograms instead of using for-loop.

Currently, my best result accuracy is 67% (with Gabor filters and also change the weight factor in SPM as I mentioned above.)

Thank you for this amazing homework. I learned and enjoyed a lot!