16-720 Computer Vision: Homework 1
Spatial Pyramid Matching for Scene Classification

Abhishek Bhatia (abhatia1@andrew.cmu.edu)

2/2/2016

1) Representing the world with visual words

1.0)

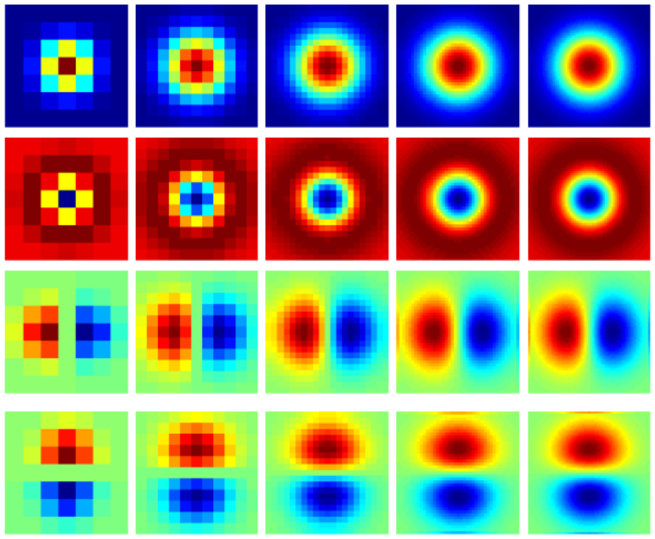


Figure 1: Provided multi-scale filter bank (taken from assignment handout)

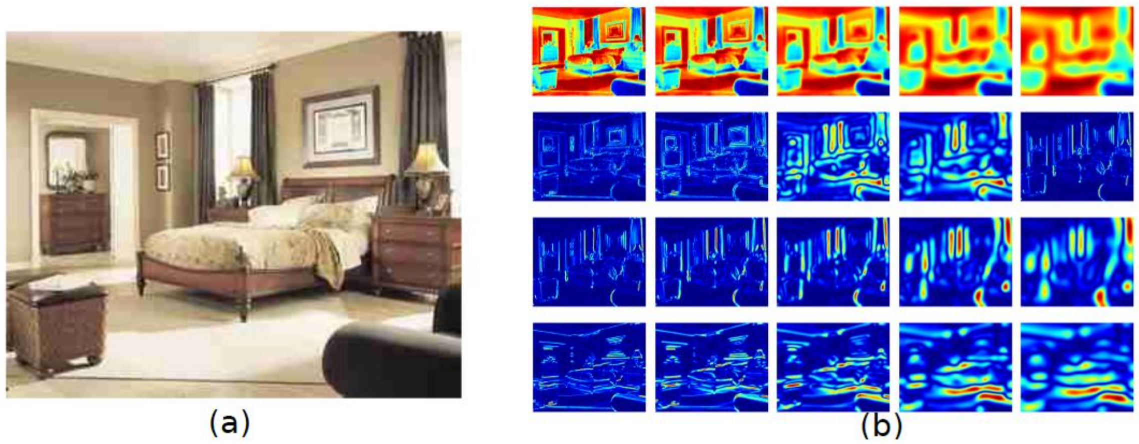


Figure 2: Input image and filter responses from the filter bank (taken from assignment handout)

From the above figures (Figure 1 and 2), it can be observed that:

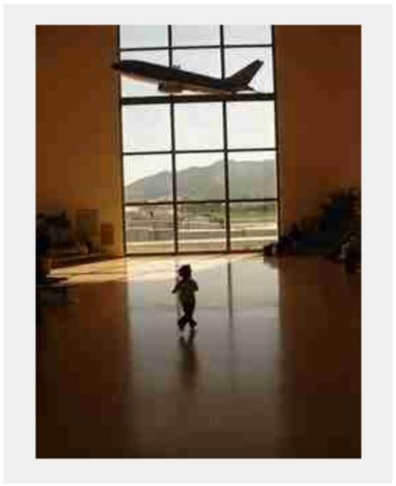
Filter 1-5: Gaussian with varying scale from 1-5. Gaussian smoothing filters are known to blur images and remove noise and detail. As shown in Figure 2, Figure 2.a is the input image and Figure 2.b first row shows the filter responses of this image with the Gaussian filters. The blurring increases from left to right indicating the increase in scale factor from left to right.

Filter 6-10: Laplacian of Gaussian (LoG) with varying scale from 6-10. Laplacian of Gaussian is the double derivative of the Gaussian function, it highlights the regions of rapid intensity change and hence is used for edge detection. Similar to the Gaussian filter, as the scale increases, the blurring also increases with the LoG filter, highlighting the prominent edges (Figure 2.b second row).

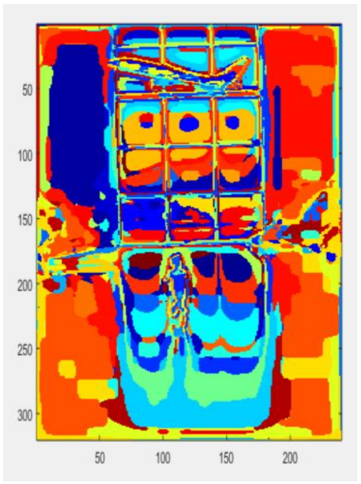
Filter 11-15: Derivative of Gaussian in x direction with varying scale from 11-15. After the Gaussian filter is applied to the input image, to get the blurred output, it's derivative is calculated along the x axis. This combination detects the edges along the vertical direction. As the scale factor is increased, the prominent edges are highlighted (Figure 2.b third row).

Filter 16-20: Derivative of Gaussian in y direction with varying scale from 16-20. After the Gaussian filter is applied to the input image, to get the blurred output, it's derivative is calculated along the y axis. This combination detects the edges along the horizontal direction. As the scale factor is increased, the prominent edges are highlighted (Figure 2.b fourth row).

1.3) Computing Visual Words



a)



b)

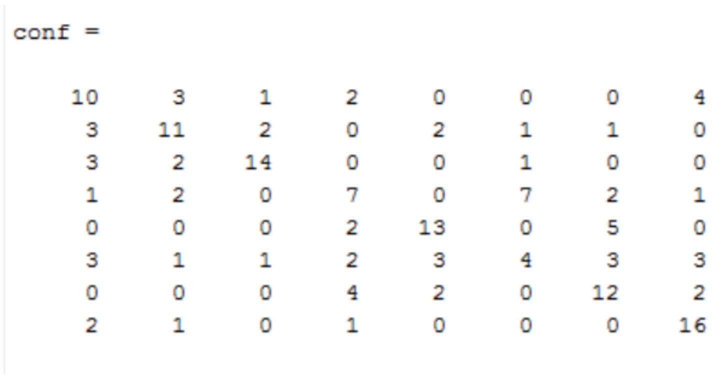
Figure 3: a) is the original input image and b) is the generated visual word output

2) Building a Recognition System

2.5) Quantitative Evaluation

After loading the test images and their labels and comparing them with predicted labels of each test image, I get the below mentioned accuracies and confusion matrix:

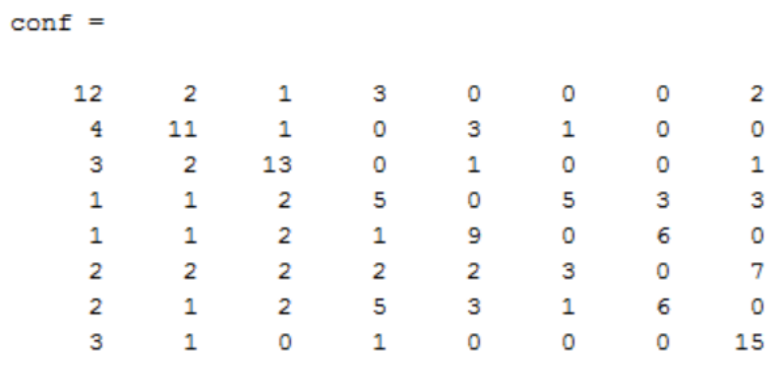
a) With Spatial Pyramid Matching:
Accuracy: 54.3750 (K_words = 300, alpha = 150)
Confusion Matrix:



```
conf =

    10     3     1     2     0     0     0     4
     3    11     2     0     2     1     1     0
     3     2    14     0     0     1     0     0
     1     2     0     7     0     7     2     1
     0     0     0     2    13     0     5     0
     3     1     1     2     3     4     3     3
     0     0     0     4     2     0    12     2
     2     1     0     1     0     0     0    16
```

Without Spatial Pyramid Matching:
Accuracy: 46.2500 (K_words = 300, alpha = 150)
Confusion Matrix:



```
conf =

    12     2     1     3     0     0     0     2
     4    11     1     0     3     1     0     0
     3     2    13     0     1     0     0     1
     1     1     2     5     0     5     3     3
     1     1     2     1     9     0     6     0
     2     2     2     2     2     3     0     7
     2     1     2     5     3     1     6     0
     3     1     0     1     0     0     0    15
```

Few Observations:

- I tried out different configurations by varying the value of alpha and k_words, but there was no major change in accuracy overall, the best accuracy I achieved was with K_words = 300 and alpha = 150, for which the confusion matrix has been pasted above.
- I was doing a mistake in my implementation of generating filter responses, I was converting the RGB image to Lab format, but was not using the Lab output during calculating filter responses. I got the error for the first time while using the guess_image function, where im2double(image) as an input to extractFilterResponses was not giving

me the correct responses. After debugging for a lot of time, when I still couldn't figure out my mistake, I modified the guess_imge function to not convert image using im2double before sending the input to extractFilterResponses function. However, after careful observation, I found my mistake and changed the guess-image function to normal.

- Another mistake that I was doing was selecting the alpha random pixels from the image before calculating the filter response. The correct approach is to calculate the filter response of the image before and then select alpha random values from the filter response. My accuracy improved after fixing this issue.
- Regenerating the dictionary and visual words again for same alpha and K_words value could lead to different accuracy values, for obvious reasons as we select alpha pixels randomly from the image and generate the filter responses which could differ for same alpha and K_words values.

2.6) Failed Cases

Observations:

- The category that gave me the maximum accuracy was 'rainforest'.
- The categories where the system had maximum confusion were between desert and landscape, and auditorium and football stadium.
- The confusion between desert and landscape could mostly be because of similar outdoor conditions, basically similar lighting, texture and mostly the environment (large empty spaces).
- The confusion between auditorium and football stadium could be because of similar geometry. Auditorium and football stadium both have huge seating areas with lots of chairs providing similar texture and environment.
- Other categories where I observed some confusion includes bedroom and airport mostly because of similar lighting conditions. There was also some confusion between campus and rainforest mostly because of presence trees/grass which have similar texture.
- One interesting case I observed was an image of airport with a huge tree was classified as rainforest, which makes sense because the tree provides the texture similar to that of a rainforest.

2.7) Improving Performance

- Besides varying the values of alpha and K_words to find a perfect combination that could generate the maximum accuracy, one thing to try for improving performance is to use Gabor Filters and generate a filter bank, instead of using the filters described above.

Gabor filters are linear filters that are used for edge detection and are similar to human visual system. Gabor are found to be particularly appropriate for texture representation and discrimination [https://en.wikipedia.org/wiki/Gabor_filter]. Based on this information, I would definitely try using these filters for our recognition pipeline.

- Secondly, playing with weights of the layers or maybe even increasing the number of layers in the Spatial Pyramid Method could help boost accuracy by some percentage.