Computer Vision: Homework 1 (Bag of Words)

Tushar Chugh (tchugh@andrew.cmu.edu)

Answer 1.0:

a. *Filter Group 1:*

```
for scale = gaussianScales
   idx = idx + 1;
   filterBank{idx} = fspecial('gaussian', 2*ceil(scale*2.5)+1, scale);
end
```

The above code creates a group of Gaussian filters with different size and sigma. The Gaussian filter when convoluted with the Image produces a blur effect and is used to remove noise from the image. In frequency domain, Gaussian filter acts as a low pass filter and hence removes high frequency noise. Increasing the value of 'sigma' restricts high frequency values to pass through (hence produces more blur effect). These filters for this assignment give similar response to flat regions and provide overall information content of the image. Figure 1 represents the surf plot of 'filterBank{1}' and 'filterBank{5}' respectively. Both have different size and sigma.

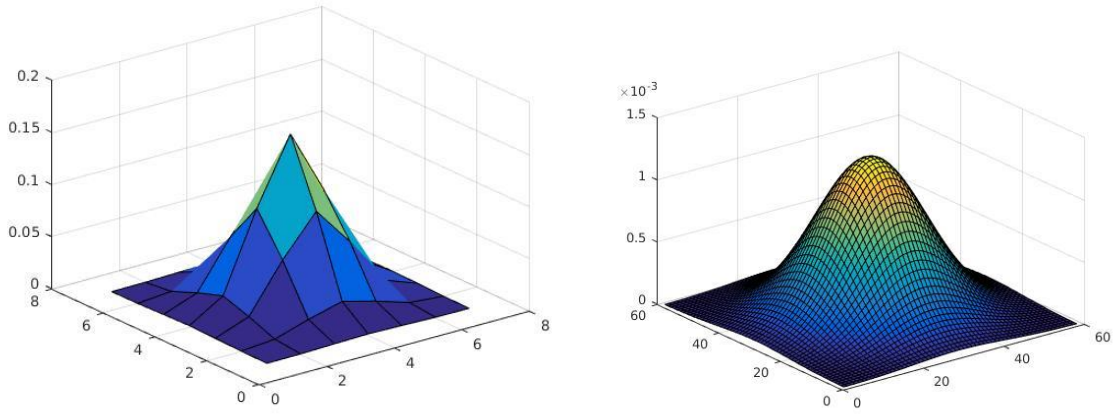

Figure 1: Gaussian Filters

b. *Filter Group 2:*

```
for scale = logScales
   idx = idx + 1;
   filterBank{idx} = fspecial('log', 2*ceil(scale*2.5)+1, scale);
end
```

The above group creates a group of 'Laplacian of Gaussian Filters'. The LoG filters gets the high frequency components of the image without the noise. Log is the second order derivative of the gaussian filters. Gaussian removes the noise first and then the 2nd order derivative detects the significant spatial change which are also the edges. Figure 2 below represents the surf plot of 'filterBank{6}' and 'filterBank{10}' which are LoG filters. LoG filter are also useful to find blobs.

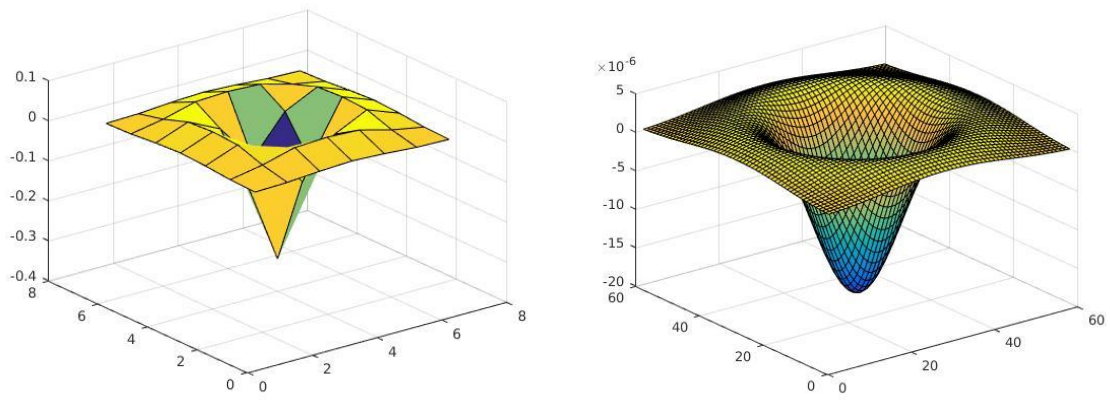

Figure 2: LoG Filters

c. *Filter Group 3:*

```
'for scale = dxScales
   idx = idx + 1;
   f = fspecial('gaussian', 2*ceil(scale*2.5) + 1, scale);
   f = imfilter(f, [-1 0 1], 'same');
   filterBank{idx} = f;
```

The above code creates a group of filters which are derivative of Gaussian in the horizontal direction (x-axis). This implies that these filters can pick up vertical edges. These filters are applied in two steps. The first one is to apply Gaussian to remove noise from the image and the second is to take the derivative in the x direction ([-1 0 1]) to find the edges. Figure 3 represents the surf plot of 'filterBank{11}' and 'filterBank{15}'.

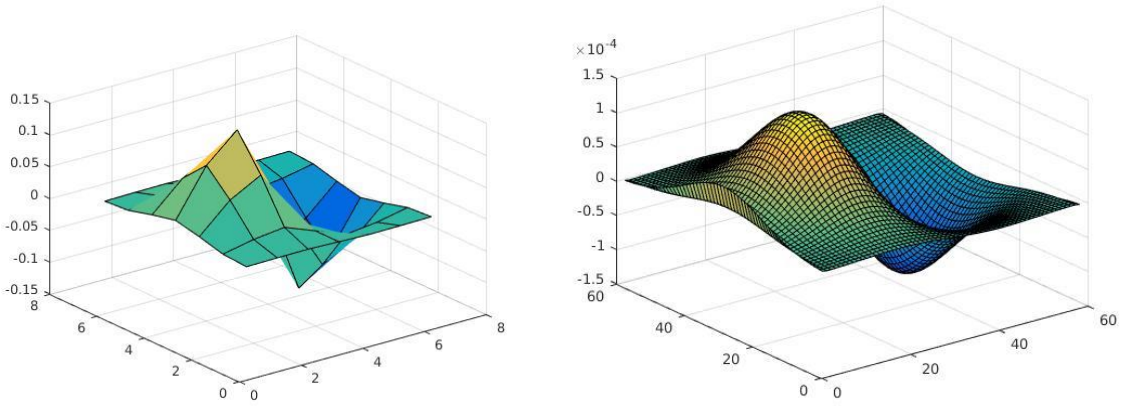

Figure 3: Horizontal derivative filters

d. Filter Group 4:

```
'for scale = dyScales
   idx = idx + 1;
   f = fspecial('gaussian', 2*ceil(scale*2.5) + 1, scale);
   f = imfilter(f, [-1 0 1]', 'same');
   filterBank{idx} = f;
end'
```

Group 4 filters are very similar to group 3 filters with the exception that these are the derivatives of Gaussian in the vertical direction (y-axis). These filters look for edges in the horizontal direction (transpose of [-1 0 1] in code). Here also the derivative is applied on the smoothened image (actually we use convolutions property to multiply the filters first and then we convolve the result with the image). Figure 4 represents the surf plot of 'filterBank{16}' and 'filterBank{20}' respectively.

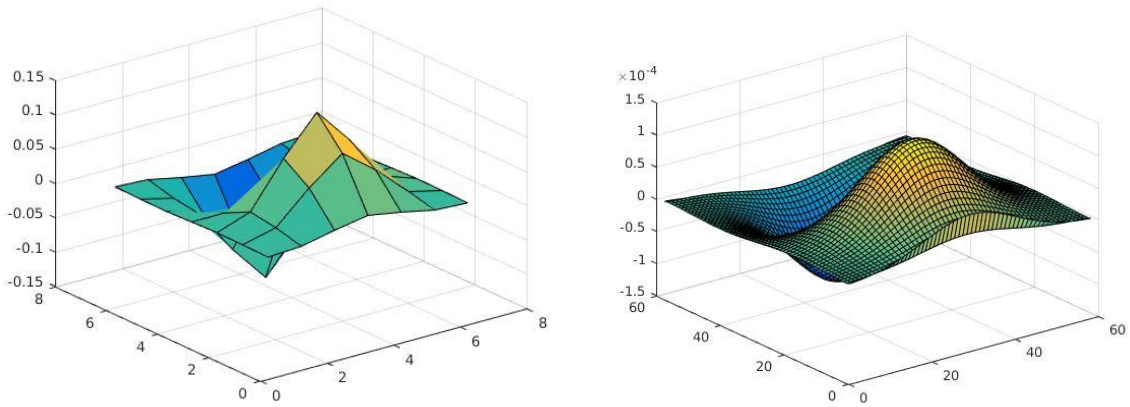

Figure 4: Vertical derivative filters

Solution 2.5

I tried with different values of alpha and K and got different results. The combinations which I tried were following (K= 100, alpha = 50; K = 150; alpha = 50; K = 200, alpha = 50, K = 200, alpha = 75, K = 200, alpha = 100). I got the best results with K = 200 and alpha = 75. The accuracies which I got in last 2 test runs with these values were 56.6250 and 60 respectively. The confusion matrices of these tests runs are shown in figure 5 and figure 6 respectively. Here horizontal axis is the target class and vertical axis is the output class.

filterBank | filterBank{1, 1} | C | test_labels | predicted_indexes

8x8 double

	1	2	3	4	5	6	7	8
1	13	3	3	0	0	0	0	1
2	3	15	0	0	0	1	1	0
3	3	1	11	1	2	2	0	0
4	2	2	0	7	1	2	5	1
5	0	2	1	0	13	0	4	0
6	0	0	0	2	3	7	6	2
7	1	1	0	4	5	0	7	2
8	2	0	0	1	0	0	1	16

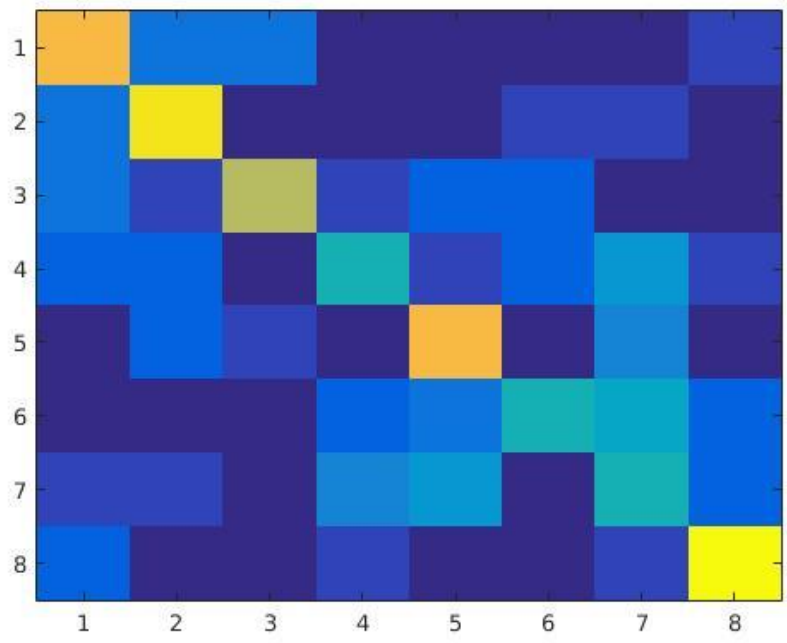

Figure5: Confusion matrix when K = 200, alpha = 75 and accuracy = 56.6250

Editor - evaluateRecognitionSystem.m Variables - C

C

8x8 double

	1	2	3	4	5	6	7	8
1	14	2	2	0	0	0	0	2
2	5	12	0	0	0	2	1	0
3	3	4	11	0	1	0	1	0
4	1	1	0	9	1	2	5	1
5	0	3	1	0	13	0	3	0
6	0	0	0	2	2	9	6	1
7	0	1	0	3	3	0	11	2
8	2	0	0	1	0	0	0	17

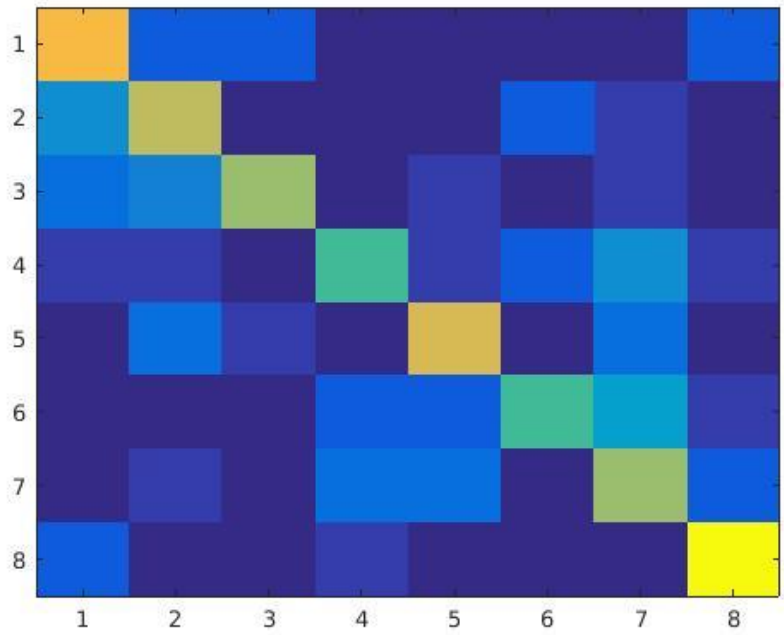

Figure 6: Confusion matrix when K = 200, alpha = 75 and accuracy = 60

Solution 2.6

By looking at both the confusion matrices we can see that the worst accuracy is for class 7 which happens to be landscape. In case one (56% accuracy) we can see that four images of campus class are wrongly predicted as landscape and five cases of landscape are misclassified as campus. This is probably because both the classes have a lot of greenery (ground with grass and trees). For the same reason, some of the images from landscape are also misclassified as football stadium (ground).

Similar trends can be interpreted from data between landscape and desert. Five images from desert were misclassifies as landscape and four images from landscape were misclassified as desert. This is because the texture is similar in some images of both the scenes.

Solution 2.7

For this part I used Gabor filters and added them to the filter bank. Here are the parameters which I used for Gabor filters (sigma = [1 2 4 8 sqrt(2)*8], theta = 0, pi/4, pi/2/, -pi/4 [in order to detected vertical, horizontal and inclined edges], lambda = 3, psi = 0 and gamma as 0.5). The accuracy that I got with this is: 61.8750. Below is the confusion matrix of the results. This result is for 40 filters (20 given and 20 Gabor filters)

C

8x8 double

	1	2	3	4	5	6	7	8
1	10	1	2	2	0	1	0	4
2	7	8	2	0	0	1	1	1
3	2	3	8	0	4	0	2	1
4	0	0	0	13	0	1	2	4
5	0	1	1	0	12	0	6	0
6	2	0	0	0	3	13	0	2
7	0	0	1	2	2	0	15	0
8	0	0	0	0	0	0	0	20

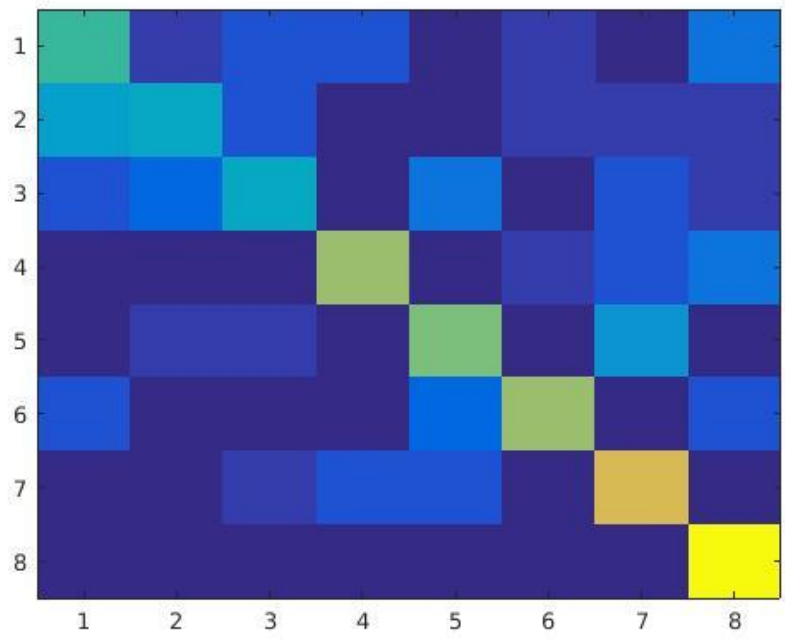

Figure 7: Confusion matrix when 40 filters including Gabor filters were used

I also tried running only 20 Gabor filters (excluded given filters). The accuracy was similar as the Gabor filters i.e. 61.250. Figure below represents the confusion matrix for this case.

C

8x8 double

	1	2	3	4	5	6	7	8
1	11	1	1	2	0	1	1	3
2	6	9	2	0	0	1	1	1
3	3	5	6	0	3	0	2	1
4	0	0	0	13	0	1	1	5
5	0	0	0	1	14	2	3	0
6	2	0	0	1	3	11	0	3
7	1	0	0	2	3	0	14	0
8	0	0	0	0	0	0	0	20

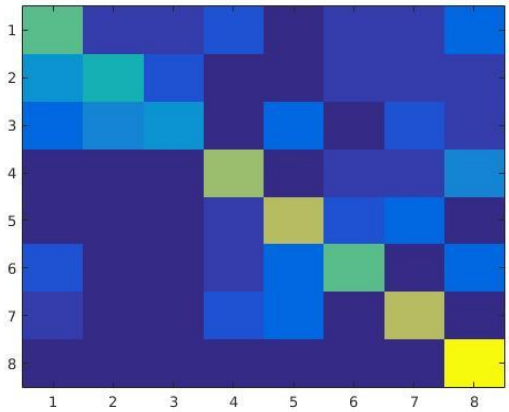

Figure 8: Confusion matrix when 20 filters Gabor filters were used

Figure 9, shows the visualization of the Gabor filters that were used to create the filter bank. Note that the filters are of following orientations 0, pi/4, pi/2 and –pi/4

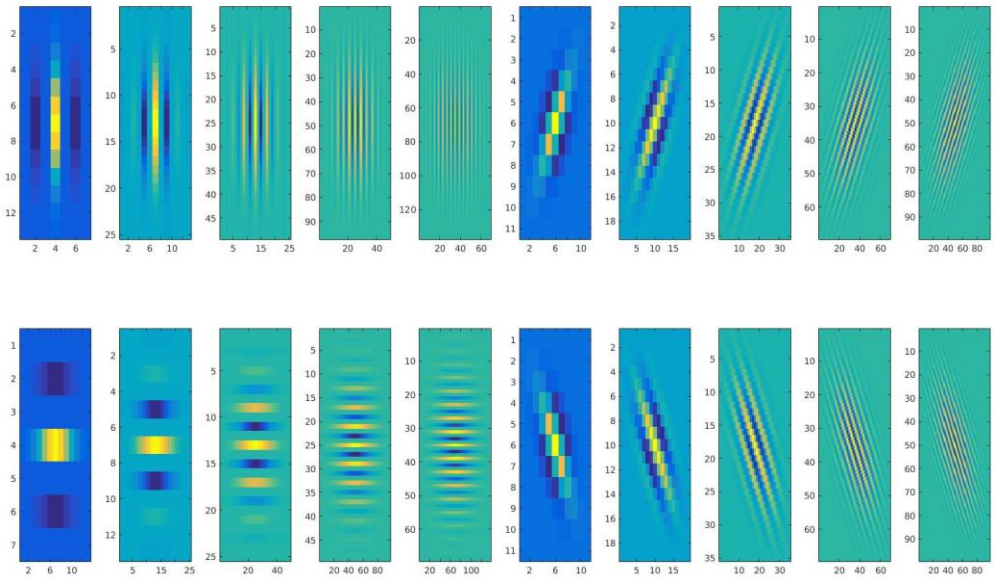

Figure 9: Filter bank of Gabor Filters