CSCI-631: Foundations of Computer Vision

Submitted by: - Yash Jain (yj8359)

HOMEWORK – 5

NOTE: I have used imshow instead of imagesc because imagesc performs scaling while showing the image and since we are comparing output image from manual filter and imfilter, imshow will help us comparing the output we have generated.

A. fltr = [1 2 1 ; 0 0 0 ; -1 -2 -1] / 8
Does this filter have its highest response (highest positive value) at places where the image goes light to dark vertically, or diagonally, or is it dark to light? What input conditions cause the largest response.
Ans: It's a vertical gradient sobel filter. It gives horizontal edges. It gives highest response going vertically for light to dark as well as dark to light.

The image is darker because we are working in range 0 to 1 and filter is also having values less than 1 as we have normalization constant, so when we multiply both of values we will get even lesser values which is close to 0 and hence darker image. The matrix values of the vertical gradient sobel filter cause the largest response.

B. fltr = [1 0 -1 ; 2 0 -2 ; 1 0 -1] / 8
How does this filter's response differ from part A?
Ans: It's a horizontal gradient sobel filter. It gives vertical edges.
Since it is horizontal gradient, it will detect the changes as we move the filter from left to right. And since it will detect the changes going from left to right, we will get all the vertical edges.

The output image is darker because we are working in range 0 to 1 and filter is also having values less than 1 as we have normalization constant, so when we multiply both of values we will get even lesser values which is close to 0 and hence darker image.

C. fltr = [1 0 0 0 -1 ; 2 0 0 0 -2 ; 1 0 0 0 -1] / 16
How does this filter's response differ from part B?
Ans: This is also a horizontal gradient filter. It will also give us the vertical edges in picture. But since we have increased the width of our filter by putting more 0's in between, we will get thicker edges as compare to what we were getting in part B.

The output image is even more darker because normalization constant is large as compare to B, so when we multiply both of values we will get even smaller values which is close to 0 and hence darker image.

D. fltr = [1 0 0 0 0 0 0 0 -1 ; 2 0 0 0 0 0 0 0 -2 ; 1 0 0 0 0 0 0 0 -1] / …norm_constant
a. Find the correct normalization constant for norm_constant. What is the value? Describe how you arrived at this and why.
Ans: Normalization constant is 32. We can break the above filter into two matrices as:

$$A = \begin{bmatrix} 1; \\ 2; \\ 1 \end{bmatrix} \quad \text{and } B = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ -1]$$

Now sum of elements of A is **4**(1+2+1) and distance of points in B that is ΔI(delta I) is **8**(1 0 0 0 0 0 0 0 -1). Multiplying 4 and 8 we get **32**(4*8) which is the normalization value for the given filter.

b. How does this filter's response differ from part B?
Ans: Majorly there are two differences, one is the edges are thicker than what we are getting in B. Secondly, the edges are little bit darker as the normalization constant is even bigger in this case, which means even smaller filter values. When we multiply them to original image while applying filter, we will get even more smaller

values which is more close to 0 which is black hence the edges are not brighter as compare to part B.

E. [0 -1 0 ; -1 4 -1 ; 0 -1 0]
How does this filter differ from part B? How does the response differ? What do you notice?
Ans: Filter in part B is a horizontal gradient which detects vertical edges. In this case, I think it is working as diagonal gradient and hence its detecting both vertical as well as horizontal edges. It's a laplacian filter. It is second order derivate of original sobel filter.
If we have noise in image then higher order derivatives filter on that image will add more noise in output image. That's why we can see grass and other objects too when we apply laplacian. Also, edges are clear in this output.

F. fltr = fspecial('laplacian', 1)
Again, describe what you notice. Which other filter is the response of this filter most similar to?
Ans: The fspecial command returns laplacian filter in this part. Output for F is most similar to the output in part E. Laplacian is second order derivate and if we have noise in image then higher order derivatives filter on that image will add more noise in output image. That's why we can see grass and other objects too when we apply laplacian.
Here parameter is 1 while in question **E**, based on the filter matrix given I found that its parameter value is 0. I read that this parameter gives the shape of filter.

G. fltr = fspecial('log')
Again, describe what you notice. Which other filter is the response of this filter most similar to? What is the log filter?
Ans: It is laplacian of Gaussian filter. The output is similar to **F**. Log filter is precalculated in advance so that only one convolution operations is performed while running the filter. Otherwise we have to compute Gaussian first and then laplacian. Also, as mentioned in class the output image has negative as well as non negative values.

I think if we apply Gaussian filter to the output of F, we must get same output as we have in here (G).

Here we get the clearest output image to detect edges.

3. Built-in Filtering:
Re-run all of the above filters using the commands:
image_out = Imfilter(image_in, fltr, 'same', 'replicate');
a. Fill out a table of timing results to show how the times compared.

Filter	Manual Time	Imfilter time	Discussions and observations
A	0.1760	0.0209	Manual is taking more time then imfilter
B	0.2201	0.0226	Manual is taking more time then imfilter
C	0.2335	0.0245	Manual is taking more time then imfilter
D	0.3819	0.0192	As the size of filter increases it takes more time to apply the filter to the given image as we have more computations to perform.
E	0.1738	0.0174	Manual is taking more time then imfilter
F	0.1751	0.0280	Manual is taking more time then imfilter
G	0.3852	0.0203	Like D, the size of filter is larger so it took time to apply the filter as we have more computations to perform.

CSCI-631: Foundations of Computer Vision

Submitted by: - Yash Jain (yj8359)

b. Test to see if imfilter( ) is doing exactly what you think it should be doing? Does it do convolution or local weighting?

Ans: Yes imfilter() is giving the exact same output as manual filter. The only difference is we are replicating borders of images in imfilter by saying 'repl' while in manual filter we are ignoring the borders. If we look closely we can see the difference at borders. Other difference is imfilter is taking less time than manual filter to perform the calculations and generate the output image. Also, the sum of all elements in filter comes out to 0.

Based on the filters given in this homework, it does convolution/filtering but not local weighting and the Normalization factor is not the sum of all elements in given filter but its calculated in different fashion as mentioned in part **D.a**

But when I look for 'help' about imfilter I found that if we give 'conv' then it performs convolution otherwise by default it will perform correlation 'corr' which is just applying the filter.

4. Conclusions:

Write a paragraph or two about your learnings on this homework.

Ans: This homework helped in learning how edge detection filters are applied on image. In the last homework, we computed the local averaging or local weighing. Also, helped learning how to compute normalization constant for edge detection filters.

Also, learnt about the cell type data structure. It is similar to multidimensional array in other languages. It also helped in learning about laplacian filter which is a second order derivative. Learnt that if we get higher order derivative filter then it will end up adding more noise to image.

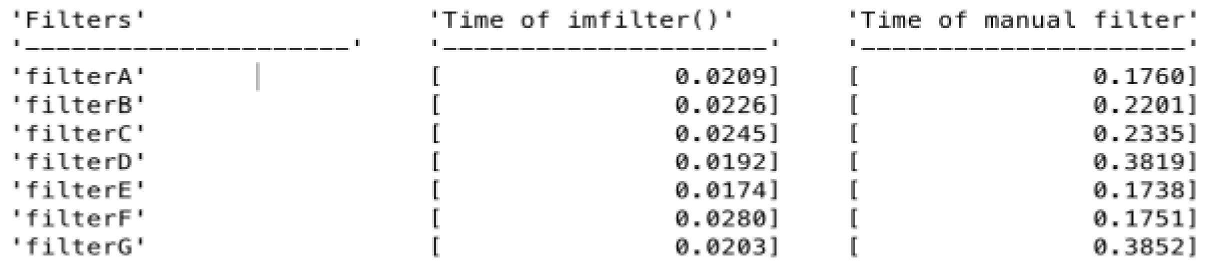
My conclusion is output image by applying manual filter as well as imfilter is same, hence manual filter implementation is correct.

s

5. Challenge:

Always feel free to exceed expectations. Perhaps you have done image processing in the past. If you need a challenge you can write a Matlab program to use cell arrays ("doc cell") which collects all of the above timings and results automatically.

Ans:



```
'Filters'                     'Time of imfilter()'       'Time of manual filter'
'_______________________'     '_____________________'    '_____________________'
'filterA'                     [              0.0209]     [              0.1760]
'filterB'                     [              0.0226]     [              0.2201]
'filterC'                     [              0.0245]     [              0.2335]
'filterD'                     [              0.0192]     [              0.3819]
'filterE'                     [              0.0174]     [              0.1738]
'filterF'                     [              0.0280]     [              0.1751]
'filterG'                     [              0.0203]     [              0.3852]
```

Fig a. Screenshot of the final values of elapsed time in cells for given filters

It's a 9*3 cell array, where column 1 is the type of filter and column 2 and 3 will tell the elapsed time when we apply imfilter() and manual filter respectively.