Assignment 1: Filtering and Smoothing

Name : Suozhi Qi

Dept: Computer Science

Part I
Exercise 1:
Original tiger image downloaded:

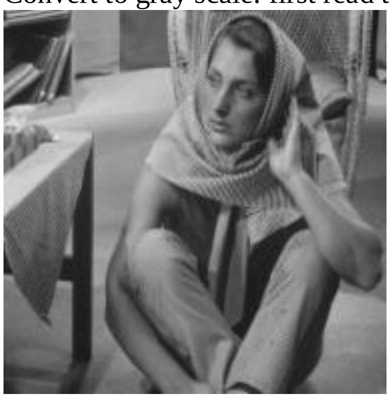
Crop image: I use imcrop() function, and the user need to manually selection out the head of the provide tiger image. The user can finish the selection process by double clicking the selected region. Here is the cropped image:

Display red components: basically just set the matrixes associated with green and blue to zero, and save the results to a new image object. Here is the red image:

Change the order of color: just assign different matrixes associated with R, G or B colors to the new order provided. Here is the color changed image:

Exercise 2:

Convert to gray scale: first read the image and then use rgb2gray function. Here is the grayscale image:



Blur the image using Gaussian filter: first I created a Gaussian filter using fspecial function. Then I use imfilter function to blur the image. Here is the image:

Subtract image: I use imsubtract function. Here is the image:

Threshold image and display the image: I use max(max()) function to get the largest pixel value, and times the result by 0.05 to get the threshold. I use the find() function to get the indexes of all the locations that have pixels lower than the threshold. I set those locations to 0 in pixel value, and get the resultant image shown below.

You can clearly see that the final image looks like a total darkness with some dim white bands in it.

Part II:
Filtering:
Filter manually the matrices I1 and I2: I manually input I1 and I2, and input the three filters as instructed. Then I use imfilter function and provides I1 in combination with each of the three filters, and repeat the same with I2. Results are attached:

```
disp(filt11);
   85.0000  125.0000  120.0000   93.3333   53.3333
   90.0000  133.3333  106.6667   75.0000   31.6667
   78.3333   95.0000   65.0000   35.0000   18.3333
   40.0000   53.3333   33.3333   23.3333   10.0000
   21.6667   28.3333   18.3333   10.0000    3.3333

disp(filt12);
   88.3333   86.6667   83.3333   56.6667   28.3333
  130.0000  123.3333  100.0000   68.3333   35.0000
  115.0000   93.3333   73.3333   38.3333   21.6667
   80.0000   60.0000   36.6667   20.0000   11.6667
   38.3333   23.3333   20.0000    8.3333    5.0000

disp(filt13);
   58.3333   86.1111   75.5556   56.1111   28.3333
```

84.4444	117.7778	97.2222	67.7778	34.4444
69.4444	93.8889	68.3333	44.4444	20.0000
46.6667	58.8889	38.8889	22.7778	10.5556
20.5556	27.2222	17.2222	11.1111	4.4444

disp(filt21);

85.0000	125.0000	120.0000	118.3333	78.3333
90.0000	133.3333	120.0000	116.6667	73.3333
41.6667	60.0000	55.0000	48.3333	30.0000
21.6667	35.0000	30.0000	23.3333	10.0000
8.3333	15.0000	11.6667	10.0000	3.3333

disp(filt22);

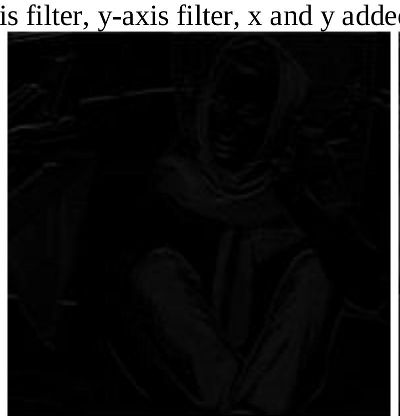
88.3333	86.6667	83.3333	70.0000	81.6667
110.0000	106.6667	101.6667	86.6667	95.0000
80.0000	73.3333	75.0000	56.6667	56.6667
38.3333	33.3333	38.3333	25.0000	18.3333
16.6667	13.3333	20.0000	8.3333	5.0000

disp(filt23);

58.3333	86.1111	80.0000	78.3333	50.5556
72.2222	106.1111	98.3333	94.4444	60.5556
51.1111	76.1111	68.3333	62.7778	37.7778
23.8889	36.6667	32.2222	27.2222	14.4444
10.0000	16.6667	13.8889	11.1111	4.4444

Apply following filters on the gray scale image of Barbara:

1. Central difference gradient filter: I create a X-axis filter [-1, 0, 1], and Y-axis filter [-1;0;1]. I then filtered the image using both filters. Next I square both of the images, add them up, and then get the square root value. Here are the images, left to right, x-axis filter, y-axis filter, x and y added up:


2. Sobel filter: I use fspecial function to create a sobel filter, and apply this filter to the image using imfilter function. Here is the image:

3. Mean filter: I repeat the same process as in sobel filter but change the parameter to average to generate the filter. Here is the image:

4. Median filter: I directly use the medfilt2 function to generate the resultant image. Here is the image:

Smoothing:
Read the image: I use imread() function
Box filters of different sizes: I use fspecial function to create boxfilters and the type is “average”. Here are the filterd image. Left to right, top to bottom: size 2, 4, 8, 16

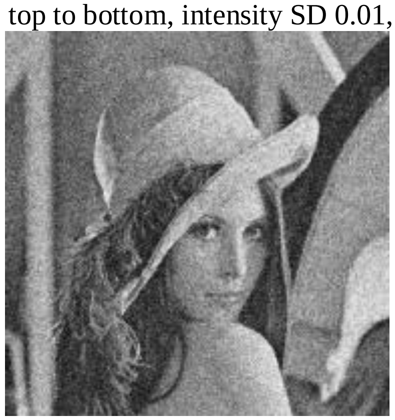
Gaussian filter of standard deviations of 2, 4, 8, 16:
According to the instructions, the size should be 4 times the SD. So the size of the each filters should be 8, 16, 32, 64. Here are the images. Left to right, top to bottom: SD 2, 4, 8, 16:

It looks like the Gaussian filter has a stronger effect compared with the box filter. In both the box filters and the Gaussian filters, with the increase of box filter size in box filters, or increase of SD in Gaussian filters, the image become more and more blurry.

Grad credits:
I first use imread() function to read the image and transform it to double precision in order for the bilateral filter to work.
Then I introduce the AWGN as instructed by the run demo.
Next I use the same parameters as the demo, but adjust the intensity standard deviation. For the “large” one I change it to 1000, and the “small” one I change it to 0.001. Then I apply the filter to each image and save the resulting file. Here are the images, left to right, original, large intensity SD, small intensity SD:



Clearly, we can see that when intensity SD gets large, the noise is getting small. I also tested other values of intensity SD. The results showed that when SD goes above 1, the noise level stays almost the same as it continues to increase. On the other hand, when SD becomes smaller, close to 0, the noise background increases. Here are the images, left to right, top to bottom, intensity SD 0.01, 0.1, 1, 10, 100:

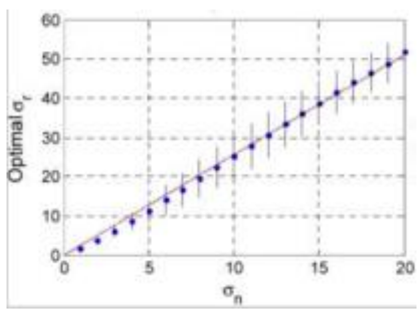




Looks like an intensity SD = 1 is optimal according to my eyes.

Compared with Gaussian filter. Here are the images, left to right, Gaussian filter sd = 4, sd = 8, bilateral intensity sd = 1:

I would say Gaussian filter seems to work better in denoising compared with the bilateral filter according to the above pics.

As to the optimal choice of intensity standard deviation related to the noise standard deviation, I found the data from a paper by Zhang et al. [1], showing that optimal intensity SD increases with the noise standard SD. Below is the data from the paper, x-axis is noise SD and y-axis is optimal intensity SD:



Reference:

[1] M. Zhang and B. K. Gunturk (2008). “Multiresolution Bilateral Filtering for Image Denoising”

IEEE Trans Image Process. 2008 Dec; 17(12): 2324–2333.