CMPE362 HW2

Spatial Filtering

Eren Kılıç

Department of Computer Engineering

TED University

ABSTRACT

Filter term in "Digital image processing" is referred to the subimage. Spatial filtering term is the filtering operations that are performed directly on the pixels of an image. The technique is used for changing the intensities of a pixel according to the intensities of the neighboring pixels. Using spatial filtering, the image is transformed based on a kernel H which has certain height and width.

I- INTRODUCTION

In this assignment, we are asked to use some MATLAB's functions related to filtering, smoothing, sharpening etc. For part 1, I implemented correlation operation which will return to same output as MATLAB's built-in imfilter function. Secondly, on part 2 I used the function recently implemented for the correlation of an image and a template. After that, I compared averaging filter and Gaussian filter for the third part. Then, to complete part 4, I applied median fitering to a corrupted image. For the next part (part 5), I implemented unsharp masking. Finally, for the bonus part I used bilateral filter function.

The detailed explanation of the steps are given below.

II- STEPS FOLLOWED

1. Implementing correlation operation

We are not asked to explain the first step. So, please go over the corresponding m-file called my_imfilter to understand the implementation.

2. Correlation of an image and a template

The main purpose of this part was related to testing our correlation implementation by computing the correlation between two images. I used one of the image as kernel on the other one.

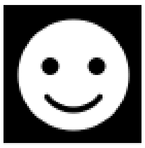

Figure 1: *Kernel image (smile.png)*

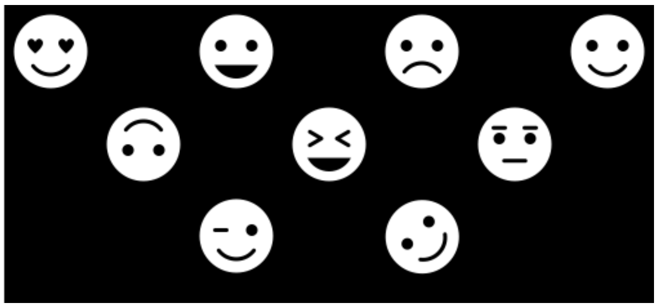

Figure 2: *Input image (emoji.png)*

I called my_imfilter function and used kernel and input images as parameters by converting these arguments to double precision. Then, I printed the result to to screen with imshow(result,[]). By using "[]" as second parameter during call, I let the minimum and the maximum value of result is mapped to black and white color, respectively. It was important to see difference between the values of the pixels.

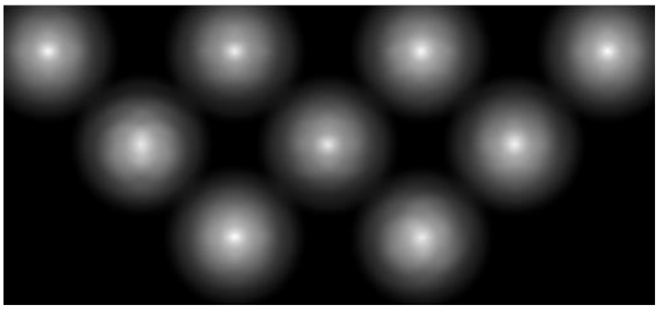

Figure 3: *Output image after correlation*

By checking figure 3, I can realize that kernel image (figure 1) is placed one of the white dots and the white dots show where the emojis are located. Hence, I can conclude that the location of a template in an image is detectable with the help of correlation.

3. Comparison of Averaging filtering and Gaussian filtering

In this part, basically we are asked to compare averaging filtering and gaussian filtering on an input image. I used different kernel sizes (7x7 and 15x15) whose weights are normalized to sum to one. First, I used imfilter function to apply averaging filtering. Note that, MATLAB's imfilter uses correlation if the parameter is not specified. A new parameter could be added as 'conv' if needed. Yet, since the values would be the same, I did not specify any extra parameter particulary. Images are smoothed with imfilter:

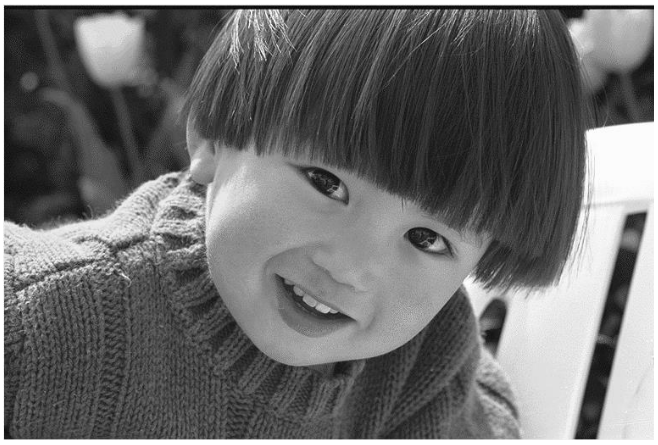

Figure 4: *Original input image (boy.png)*

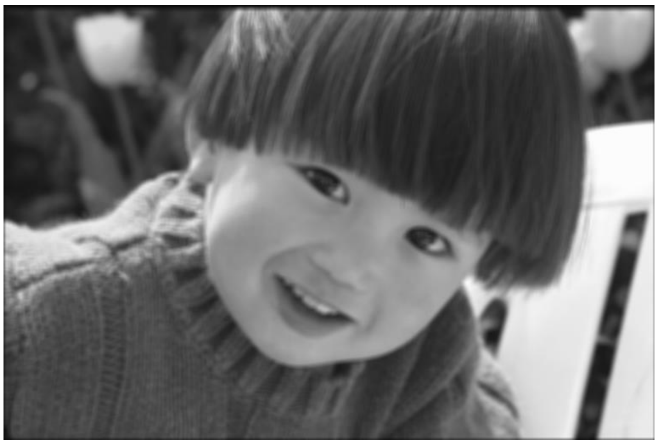

Figure 5: *Averaging filtered image (7x7 kernel)*

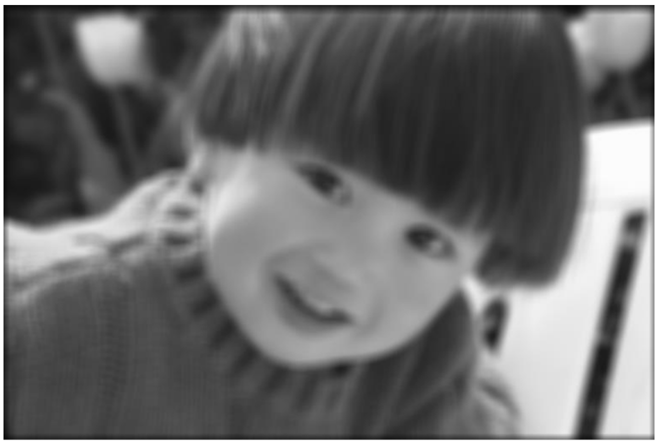

Figure 6: *Averaging filtered image (15x15 kernel)*

Then, I smoothed the image by using Gaussian filter. Again, I used two different kernels respectively (7x7 (σ=1.5) and 15x15 (σ=3.5)). Even though using fspecial for Gaussian filter is not recommended on MATLAB's offical site, I used this function anyway. Alternatively, imgaussfilt could be used. First, I calculated the filter by calling fspecial function and specified parameters in terms of type (gaussian), hsize (kernel) and sigma value. Afterwards, I called imfilter with the resulted output.

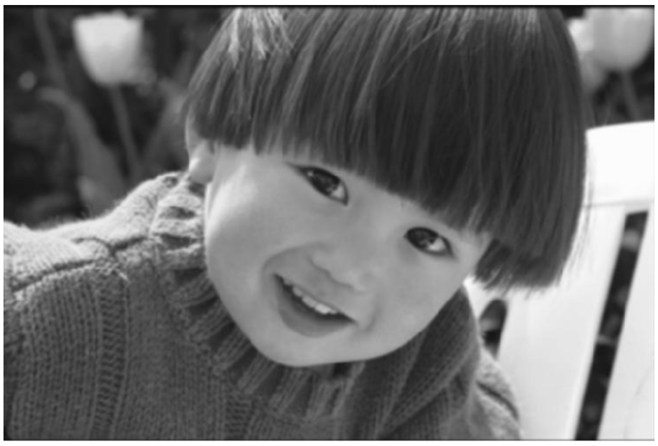

Figure 7: *Gaussian filtered image (7x7 (σ=1.5))*

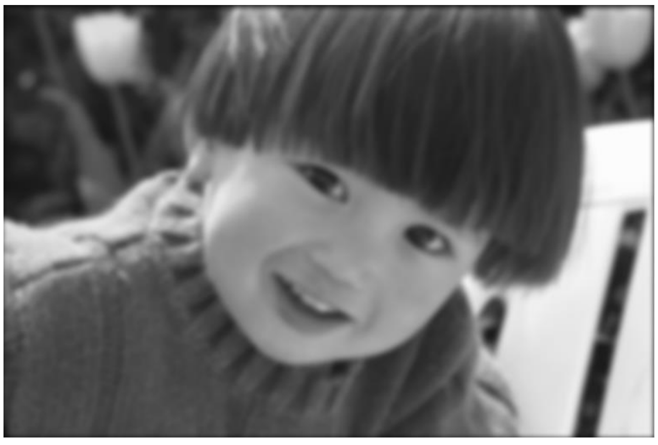

Figure 8: *Gaussian filtered image (15x15 (σ=3.5))*

As a result, if we compare figure 5 and figure, we can realize how kernel has impact on image. 7x7 kernel smoothes the image a little bit whereas 15x15 image smoothes the image noticably more. The situation is similar for figure 7 and figure 8 as well. When the sigma is 1.5 the image is less smoothed compared the case of sigma is 3.5. On the other hand, if we make a quick comparison between Gaussian filtered and averaging filtered images, we can see that Gaussian filter is more successful since the face of the boy still clear even for the 15x15(σ=3.5).

4. Median Filtering

For part 4, we are asked to corrupt the input image with a special technique called salt and pepper noise. Noise and pepper corrupts the image by changing pixels values to either black or white randomly. The desired percentage can be determined with MATLAB's imnoise function. I used 0.3 (30%) and 0.5 (50%) values for this task. I called imnoise function by specifying the the type of the noise -salt & pepper in this case- and percentage of the noise.

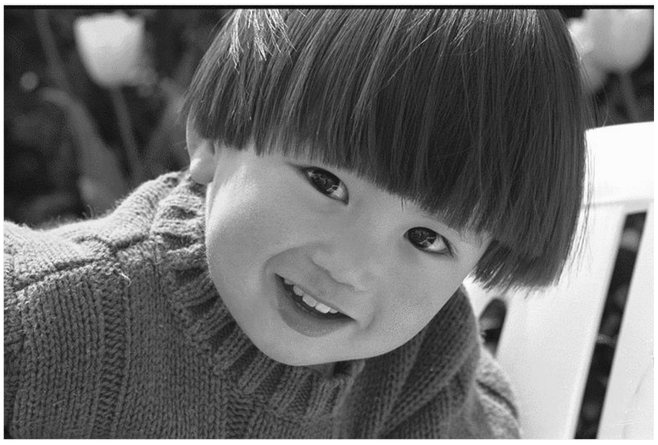

Figure 9: *Before salt&pepper (boy.png)*

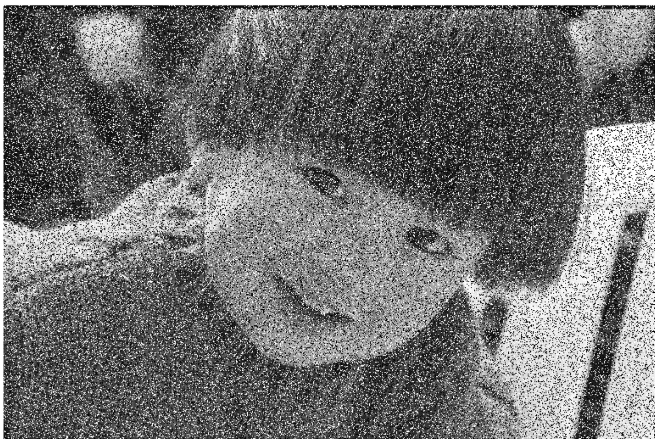

Figure 10: *Noisy image 30% (salt&pepper)*

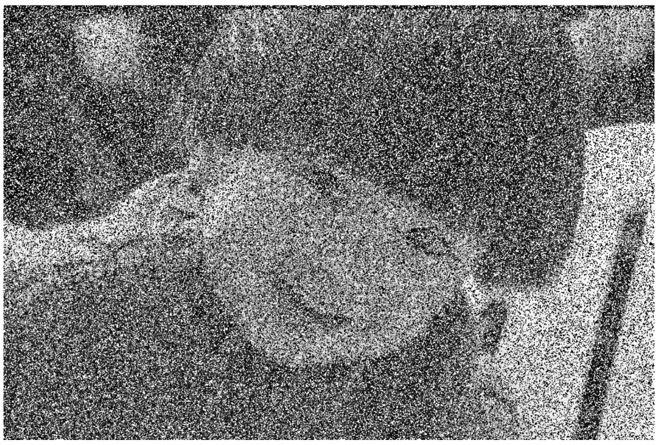

Figure 11: *Noisy image 50% (salt&pepper)*

Now I was supposed to apply filtering to noisy images in order to reduce the noise. I used both median filtering and averaging filtering for comparison purposes. I applied median filtering with medfilt2 function and averaging filtering with the combination of fspecial (type is average) and imfilter functions. Again for the comparison, I tried two different mask sizes 7x7 and 15x15.

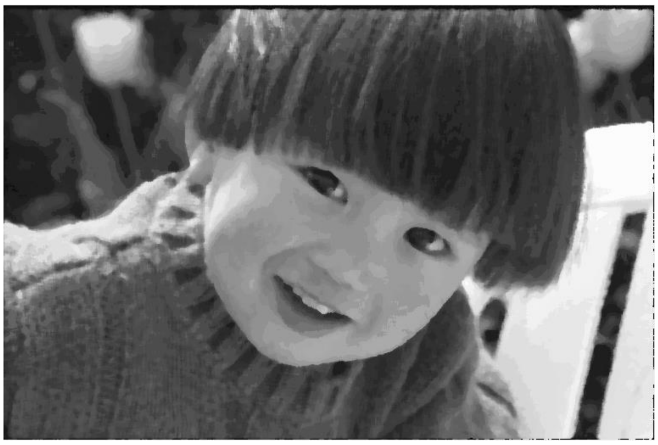

Figure 12: *Median filtered (30% noise 7x7 mask size)*

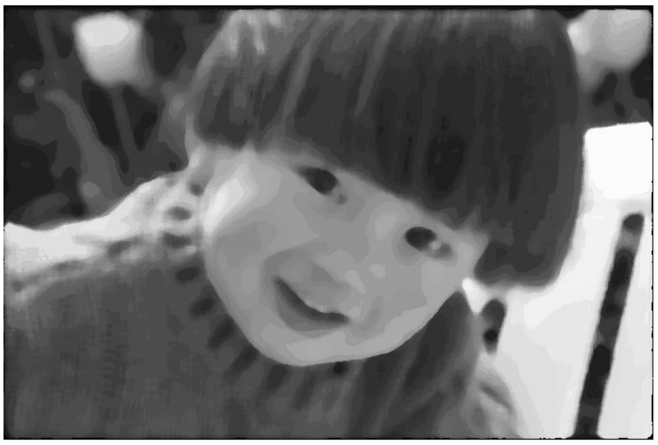

Figure 13: *Median filt. (30% noise 15x15 mask size)*

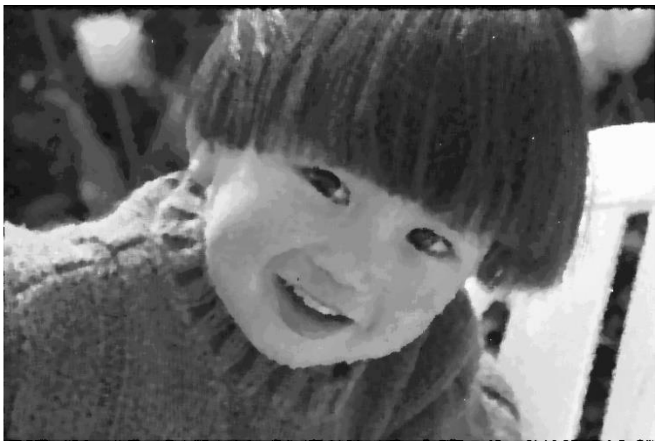
Figure 14: *Median filtered (50% noise 7x7 mask size)*

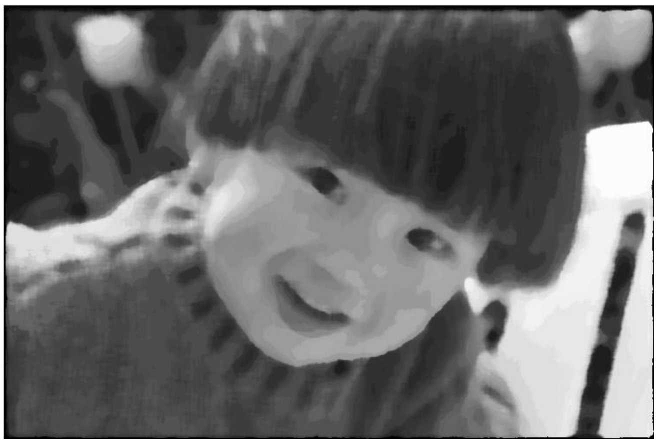
Figure 15: *Median filt. (50% noise 15x15 mask size)*

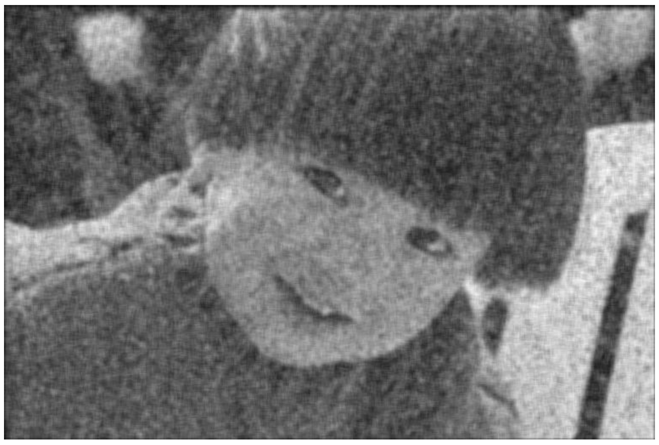
Figure 16: *Average filt. (30% noise 7x7 mask size)*

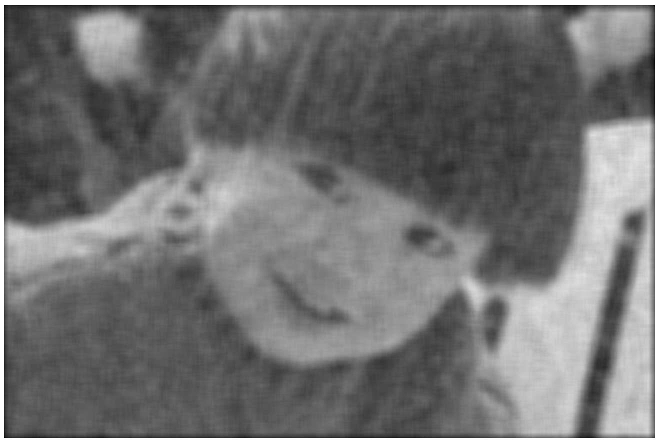
Figure 17: *Average filt. (30% noise 15x15 mask size)*

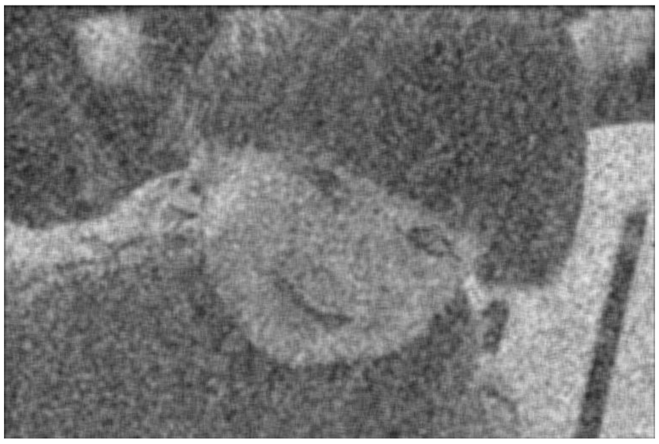
Figure 18: *Average filt. (50% noise 7x7 mask size)*

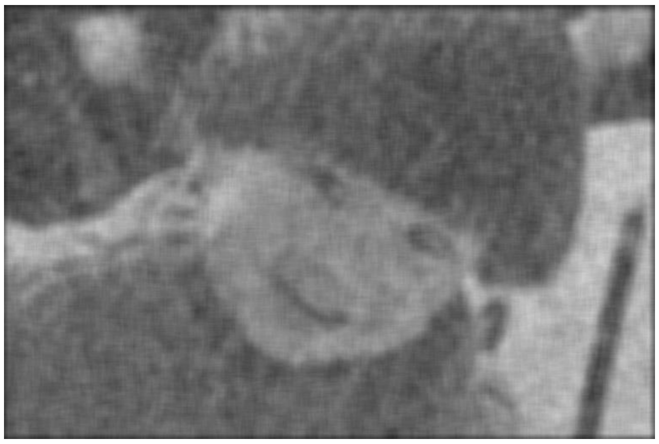
Figure 19: *Average filt. (50% noise 15x15 mask size)*

Ultimately, we can make some interpretations based on the output images. Obviously, we can see that the percentage of the imnoise changes the noise by comparing figure 10 and figure 11. In order to reduce the noises, I applied median filtering and average filtering with different mask sizes. Figure 12 proves that median filtering is quite successful to reduce 30% noise with 7x7 and the resulting image is very close the original one. However, figure 13 is more smoothed but the noises are almost dissappeared. Figure 14 shows that there are still tiny noises on 50% noisy image and it is not fully recovered. If we increase the kernel size it noises reduces as figure 15. Yet the image will be blurred. On the other hand, averaging filtering is not that successful to reduce noises unlike median filtering. Especially when the noises are distrubuted 50% randomly, applying averaging filter does not result a clear image as it shown in figure 18 and figure 19. Even increasing mask size on that situation does not fully recover the image.

5. Unsharp Masking

For this part of the assignment, we are asked to apply unsharp masking to produce a sharpened image. I implemented a function called unsharp_mask. The function first smoothes the input image by using imgaussfilt function with 7x7 Gaussian kernel (σ=1.5). According to MATLAB's documentation, the default filter size is calculated by this formula: 2*ceil(2*sigma)+1. So, if we put 1.5 here, we obtain 7 as a result. After calling imgaussfilt function I got the following output.

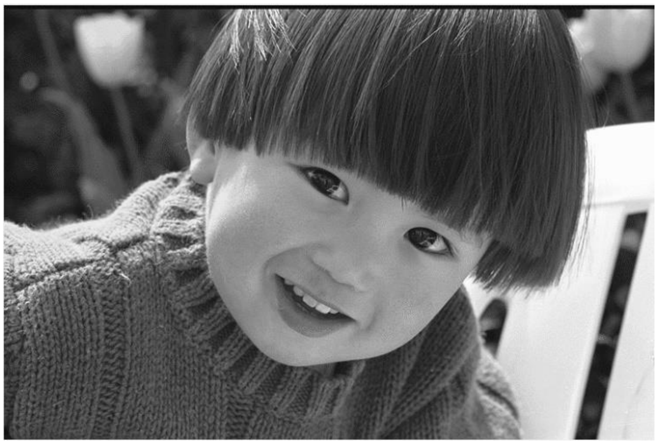
Figure 20: *Original input image (boy.png)*

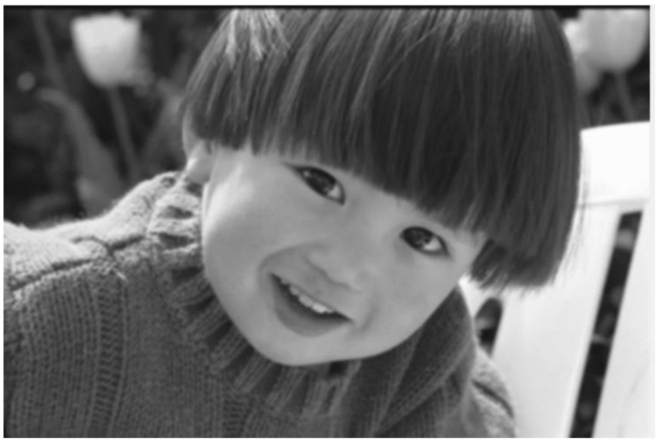
Figure 21: *Blurred* - 7x7 Gaussian kernel (σ=1.5)

After that, I subtracted the blurred image from the original image to produce an edge image.

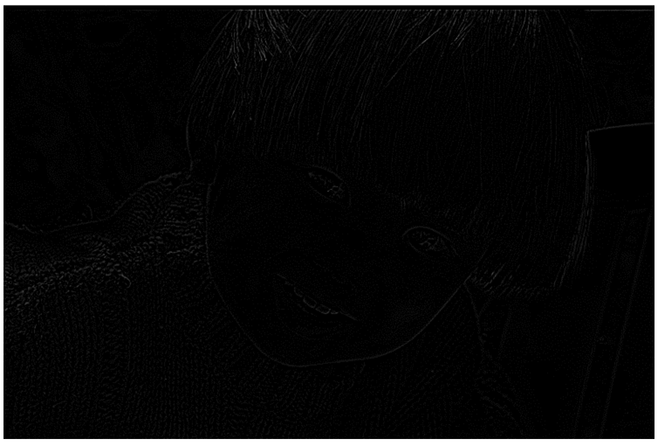
Figure 22: *Edge image (edge = input – blurred)*

Lastly, by adding edge image to the original image, I was able to get sharpened image.

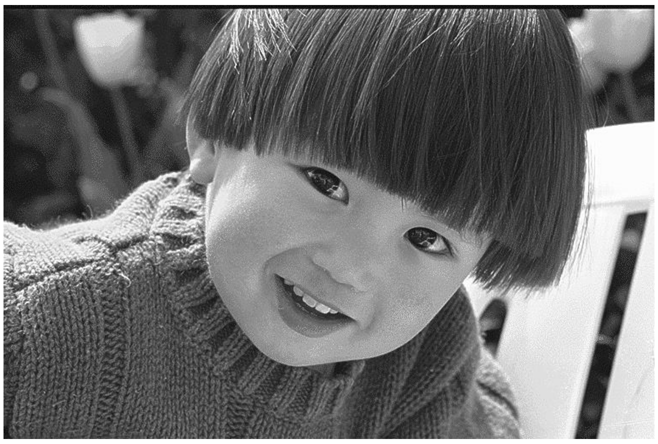
Figure 23: *Sharpened img (sharpened = edge + input)*

In conclusion, we see that producing a sharpened image is possible by applying those steps I mentioned. Unsharp masking is basically a deblurring operation. It enchances the edges in an image by subtracting off a Gaussian blurred image. First the image needs to be blurred as figure 21. Then edge image should be calculated as figure 22. At the end figure 23, the resulted image is indeed succesfully sharpened.

6. Applying Bilateral filter

The final part of the homework was required to use MATLAB's imbilatfilt function to apply bilateral filtering. This function, has 3 parameters including input image, degree of smoothing and spatial sigma. In this part, I altered degreeOfSmoothing and spatialSigma values to observe the consequence and interpret on that. I took reference a paper called Bilateral Filtering for Gray and Color Images by Tomasi and Manduchi.

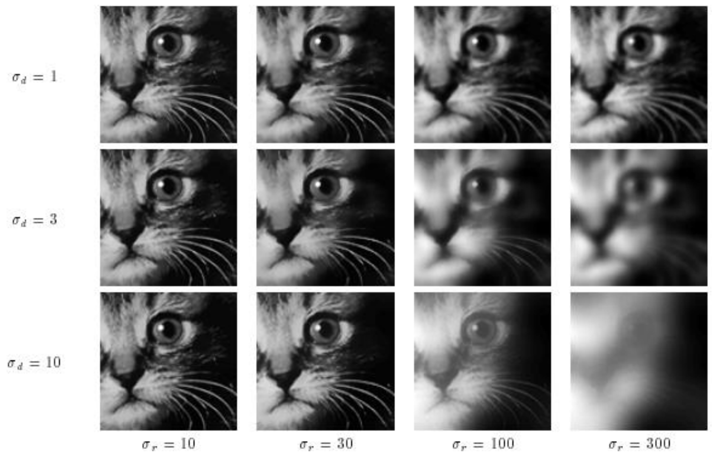

Figure 24: *An example of bilateral filtering*

During my experiment I got the outputs below.

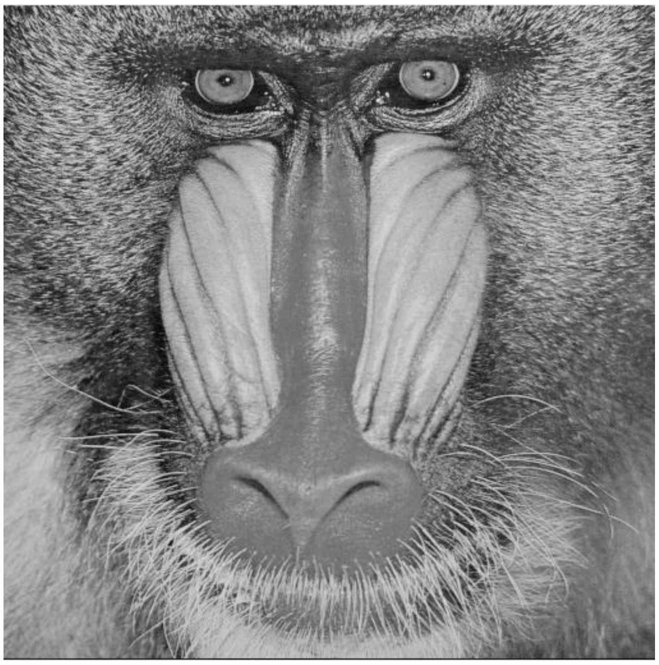

Figure 25: *Original image (baboon.png)*

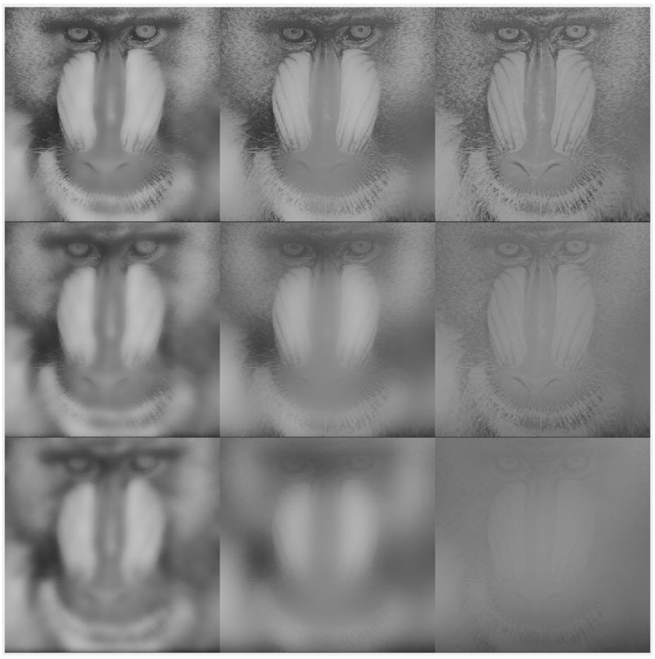

Figure 26: *Bileteral filtering on different values*

ss: 10,30,100 dos: 1,3,10 are used

In conclusion, as it shown in figure 26 when the degree of smoothing grows, the image become more sharpened. On the contrary, when we increase the spatial sigma value, the image become more blurred.

III. SUMMARY

In conclusion, MATLAB offers many functions for the filtering purposes. Filtering may be used on different fields such as noise reduction, blurring or sharpening the image. In this assignment, some of the filtering techniques are used and commented.

IV. REFERENCES

[1] Spatial Filtering. (n.d.). Retrieved from https://www.sciencedirect.com/topics/medicine-and-dentistry/spatial-filtering.

[2] Tomasi, C., & Manduchi, R. (n.d.). Bilateral filtering for gray and color images. Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271). doi: 10.1109/iccv.1998.710815.