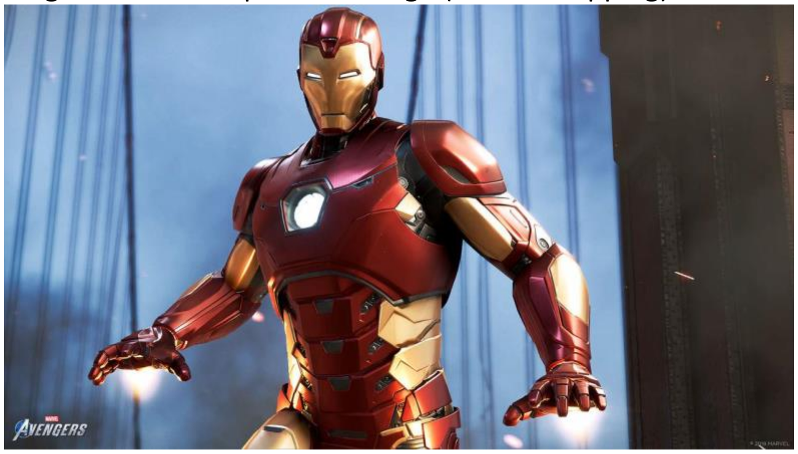
# Computer Vision Assignment 2

Kai-Po Lin (kl72); Department of Computer Science

[Part 1]

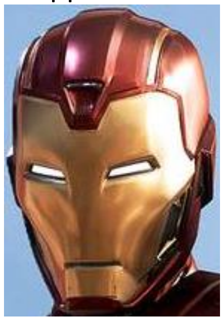
Exercise 1:

1. Original Marvel Superhero Image (before cropping).

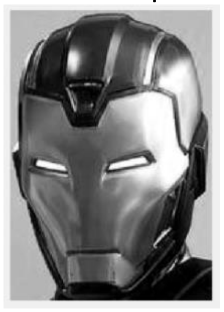
   IronMan.jpg (https://www.google.com/url?sa=i&url=https%3A%2F%2Fsquare-enix-games.com%2Fen_GB%2Fnews%2Fmarvels-avengers-pro-tips-iron-man&psig=AOvVaw0CgnMcB1_Gsd7E83Kg2pKW&ust=1643226482615000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCOi-xpLWzfUCFQAAAAAdAAAAABAD)

2. Cropped Head.

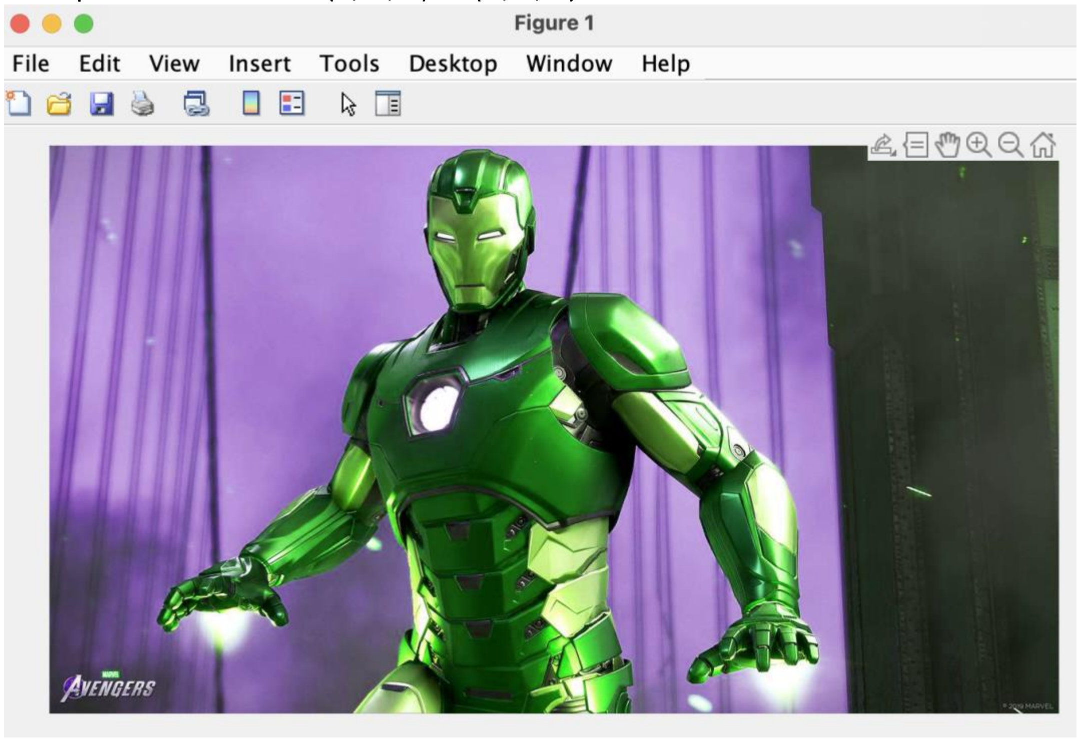
   

   head.png

3. Green Component of Head.

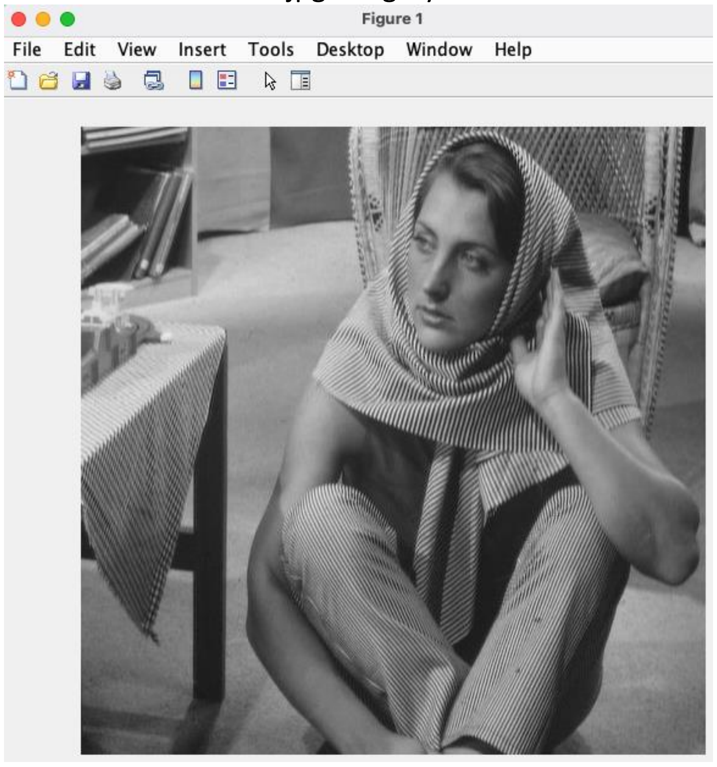
   

4. Transpose channel from (R, G, B) to (G, R, B).

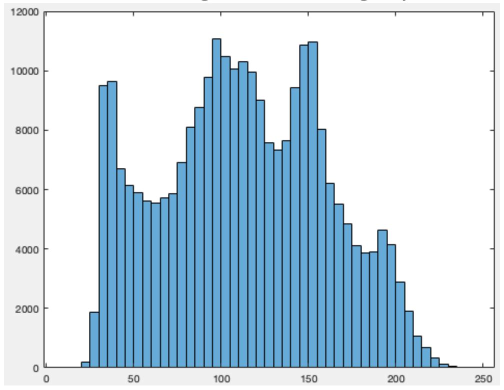


Exercise 2:

1. Convert “barbara.jpg” to gray scale.



2. Plot a histogram of the gray scale image with bin-size of 5 in intensity.



3. Blur the gray scale image by using Gaussian filters of size 15 × 15 with standard deviations 2 and 8.

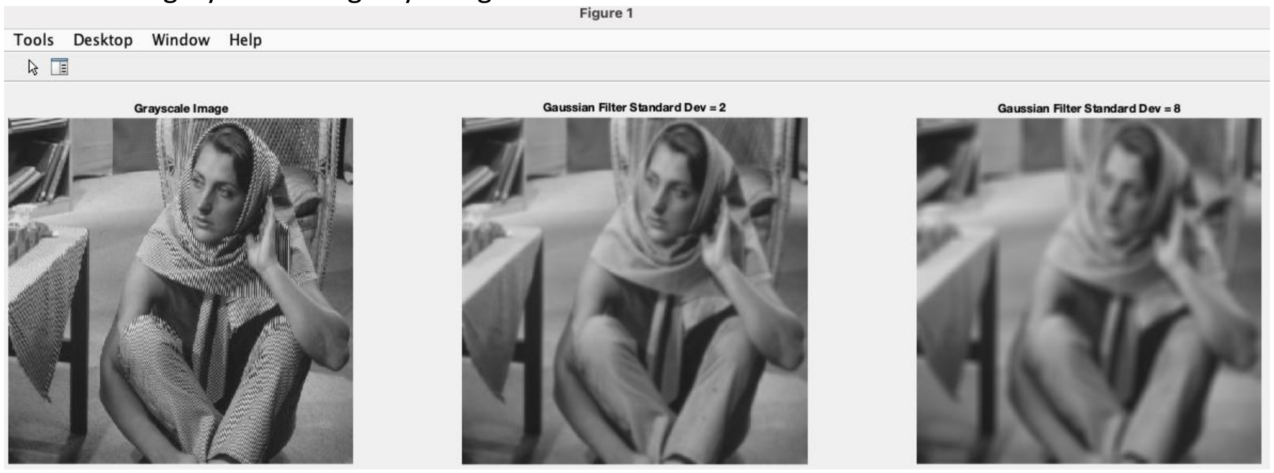
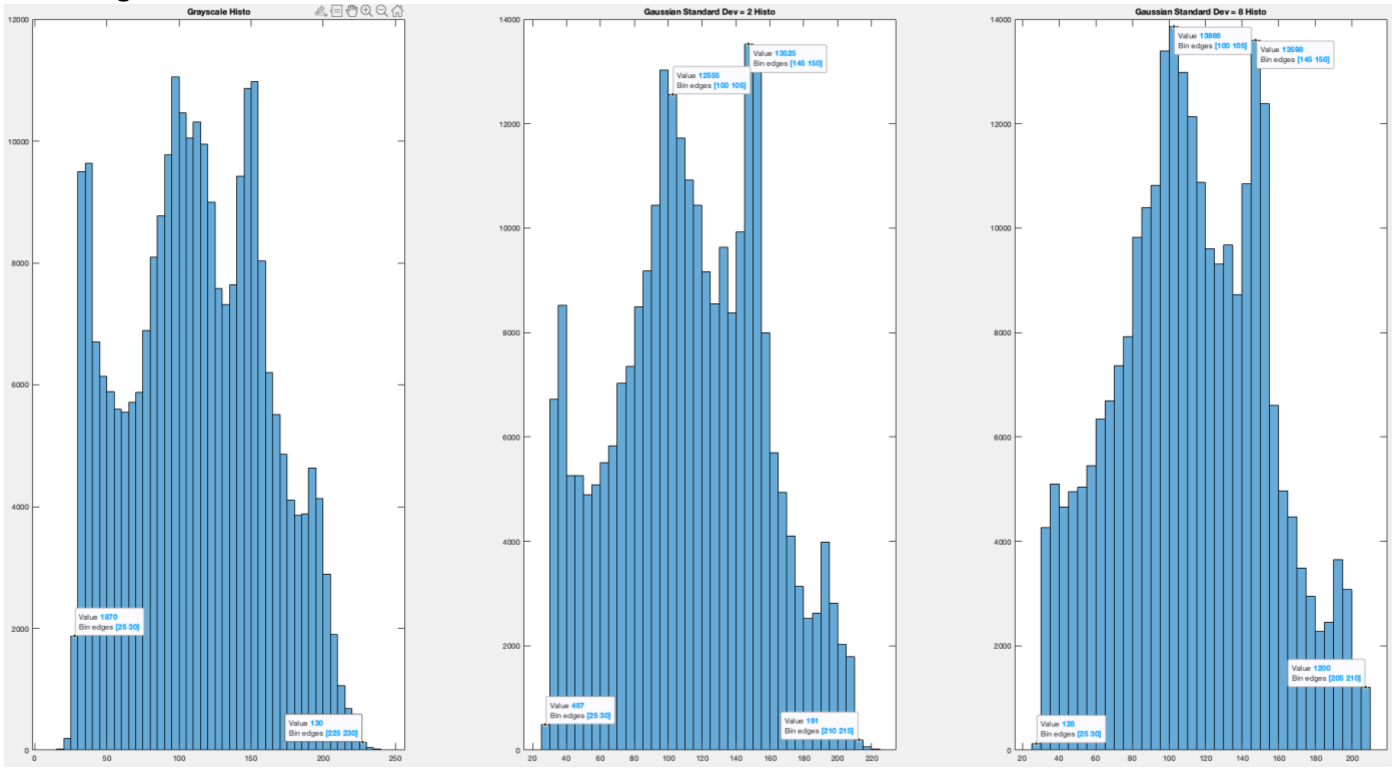


Figure 1
Tools
Desktop
Window
Help
Grayscale Image
Gaussian Filter Standard Dev = 2
Gaussian Filter Standard Dev = 8

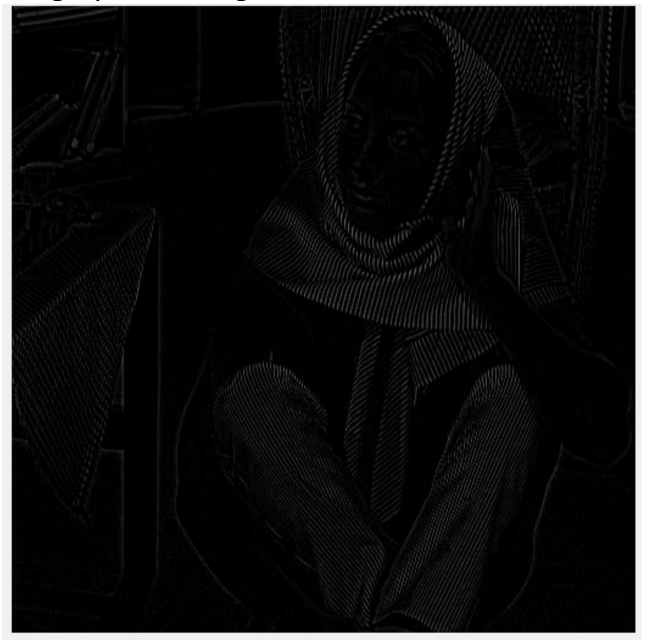
4. Plot histograms for the blurred images. How do they differ from each other and the original histogram?



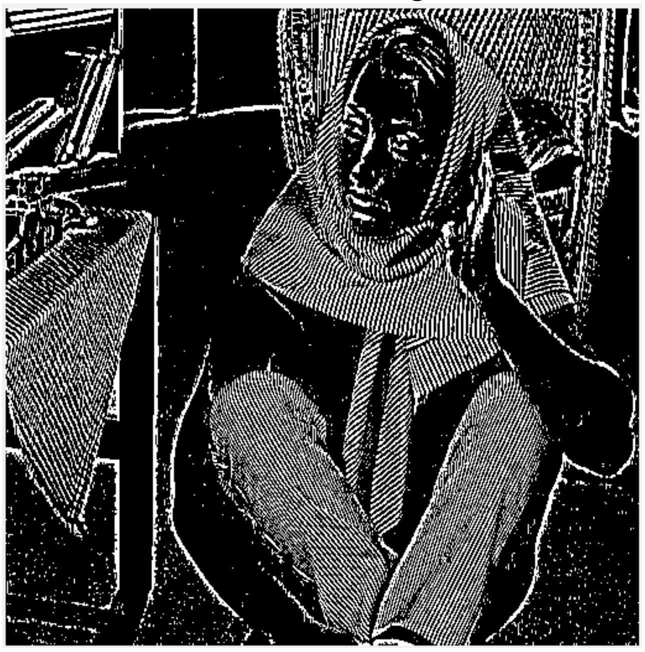
According to the histograms above, we can find out that as the standard deviation of the Gaussian filter increases (the original grayscale image can be treated as a 0-standard deviation), **the intensities of the image are more likely to centralize**. That means that a Gaussian filter with a higher standard deviation is more likely to average the intensity of data, causing the image to blur more intensively.

Intensity (Approximately)
- ⇨ Original Image: Start from intensity=20 with value 198 to intensity=235 with value 45
- ⇨ 2 Standard Deviation: Start from intensity=25 with value 487 to intensity=215 with value 191
- ⇨ 8 Standard Deviation: Start from intensity=25 with value 128 to intensity=210 with value 1200

5. Subtract the blurred image obtained using the filter with standard deviation of 2 from the original gray scale image.



6. Threshold the resultant image at 5% of its maximum pixel value and display the final image.



[Part 2]
Filtering:

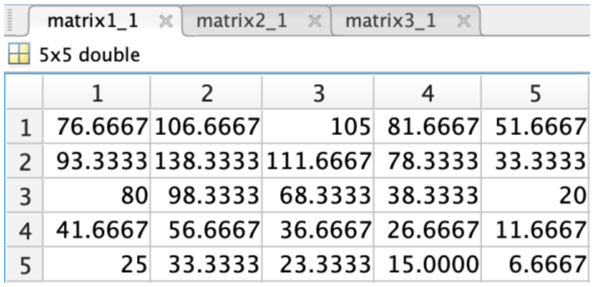
Filter the following matrices $I_1$ and $I_2$ without using built-in MATLAB functions.

$$I_1 = \begin{bmatrix} 120 & 110 & 90 & 115 & 40 \\ 145 & 135 & 135 & 65 & 35 \\ 125 & 115 & 55 & 35 & 25 \\ 80 & 45 & 45 & 20 & 15 \\ 40 & 35 & 25 & 10 & 10 \end{bmatrix} \quad I_2 = \begin{bmatrix} 125 & 130 & 135 & 110 & 125 \\ 145 & 135 & 135 & 155 & 125 \\ 65 & 60 & 55 & 45 & 40 \\ 40 & 35 & 40 & 25 & 15 \\ 15 & 15 & 20 & 15 & 10 \end{bmatrix}$$

(I1, I2 used zero padding before filtering)

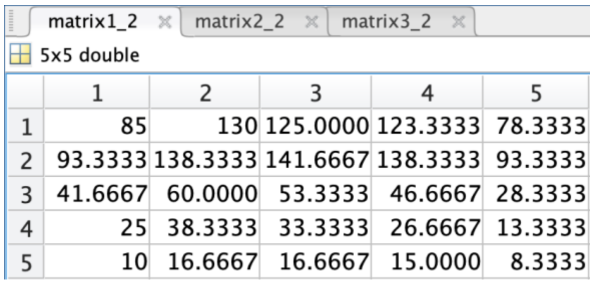
1. 1/3 [1 1 1]

(1) Filter1 on I1



| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 76.6667 | 106.6667 | 105 | 81.6667 | 51.6667 |
| 2 | 93.3333 | 138.3333 | 111.6667 | 78.3333 | 33.3333 |
| 3 | 80 | 98.3333 | 68.3333 | 38.3333 | 20 |
| 4 | 41.6667 | 56.6667 | 36.6667 | 26.6667 | 11.6667 |
| 5 | 25 | 33.3333 | 23.3333 | 15.0000 | 6.6667 |

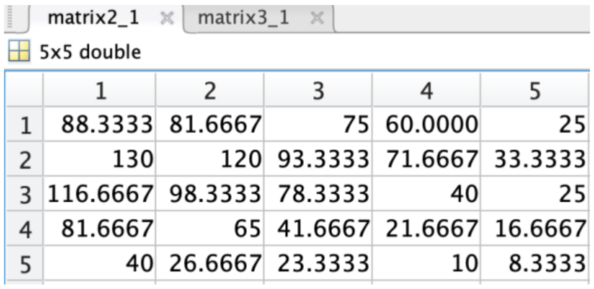
(2) Filter1 on I2



| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 85 | 130 | 125.0000 | 123.3333 | 78.3333 |
| 2 | 93.3333 | 138.3333 | 141.6667 | 138.3333 | 93.3333 |
| 3 | 41.6667 | 60.0000 | 53.3333 | 46.6667 | 28.3333 |
| 4 | 25 | 38.3333 | 33.3333 | 26.6667 | 13.3333 |
| 5 | 10 | 16.6667 | 16.6667 | 15.0000 | 8.3333 |

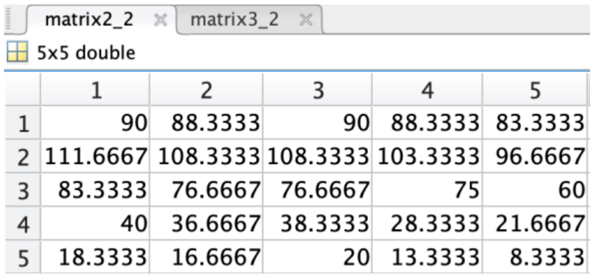
2. 1/3 [1; 1; 1]

(1) Filter2 on I1



| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 88.3333 | 81.6667 | 75 | 60.0000 | 25 |
| 2 | 130 | 120 | 93.3333 | 71.6667 | 33.3333 |
| 3 | 116.6667 | 98.3333 | 78.3333 | 40 | 25 |
| 4 | 81.6667 | 65 | 41.6667 | 21.6667 | 16.6667 |
| 5 | 40 | 26.6667 | 23.3333 | 10 | 8.3333 |

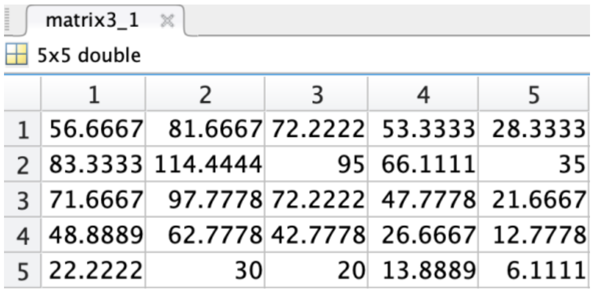
(2) Filter2 on I2



| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 90 | 88.3333 | 90 | 88.3333 | 83.3333 |
| 2 | 111.6667 | 108.3333 | 108.3333 | 103.3333 | 96.6667 |
| 3 | 83.3333 | 76.6667 | 76.6667 | 75 | 60 |
| 4 | 40 | 36.6667 | 38.3333 | 28.3333 | 21.6667 |
| 5 | 18.3333 | 16.6667 | 20 | 13.3333 | 8.3333 |

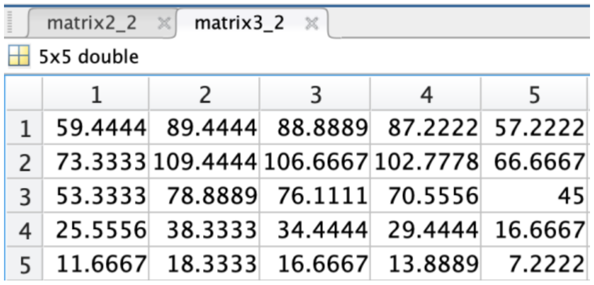
3. 1/9 [1 1 1; 1 1 1; 1 1 1]

(1) Filter3 on I1


matrix3_1
5x5 double

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 56.6667 | 81.6667 | 72.2222 | 53.3333 | 28.3333 |
| 2 | 83.3333 | 114.4444 | 95 | 66.1111 | 35 |
| 3 | 71.6667 | 97.7778 | 72.2222 | 47.7778 | 21.6667 |
| 4 | 48.8889 | 62.7778 | 42.7778 | 26.6667 | 12.7778 |
| 5 | 22.2222 | 30 | 20 | 13.8889 | 6.1111 |

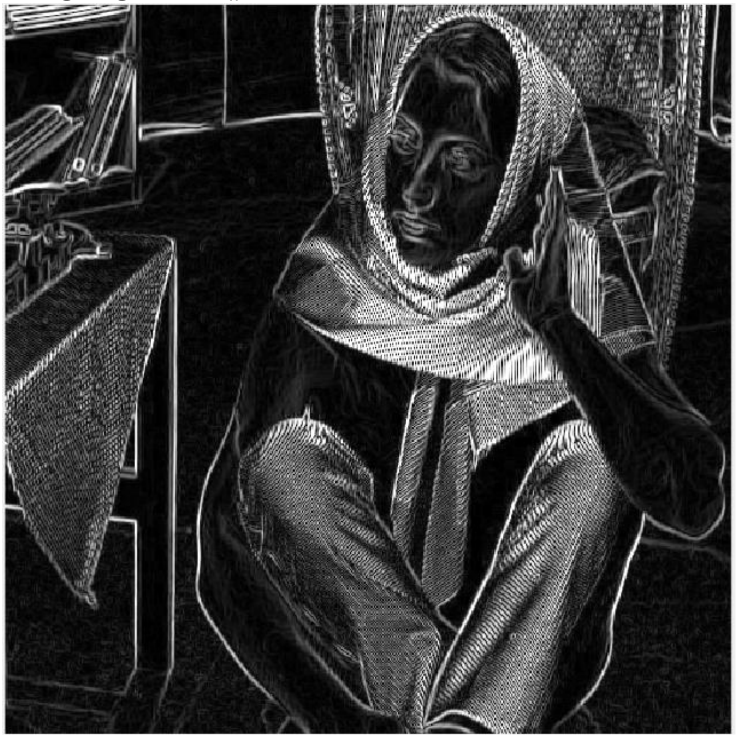
(2) Filter3 on I2


matrix2_2
matrix3_2
5x5 double

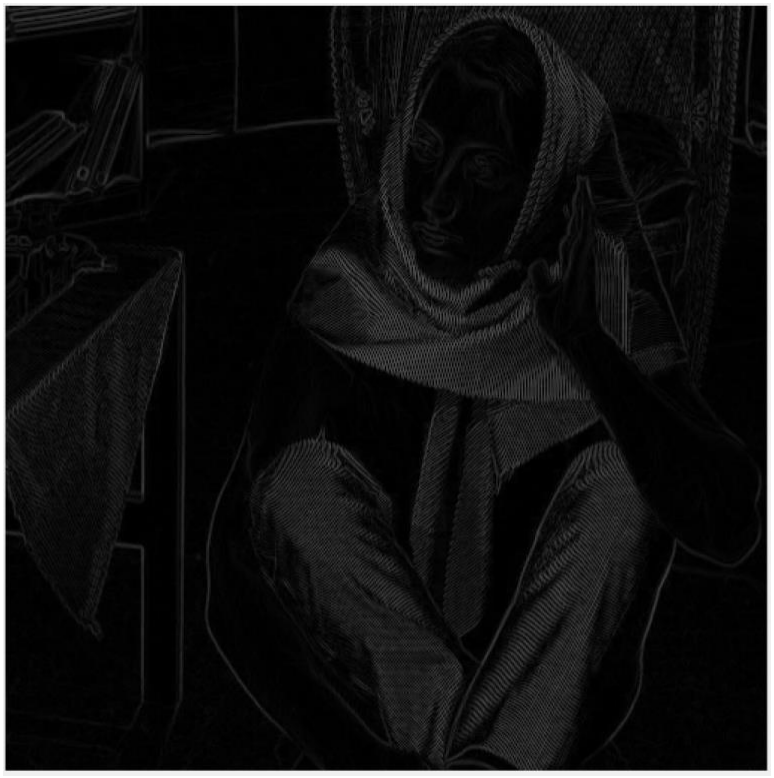
| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 59.4444 | 89.4444 | 88.8889 | 87.2222 | 57.2222 |
| 2 | 73.3333 | 109.4444 | 106.6667 | 102.7778 | 66.6667 |
| 3 | 53.3333 | 78.8889 | 76.1111 | 70.5556 | 45 |
| 4 | 25.5556 | 38.3333 | 34.4444 | 29.4444 | 16.6667 |
| 5 | 11.6667 | 18.3333 | 16.6667 | 13.8889 | 7.2222 |

Apply following filters on the gray scale image of barbara from Part I.
1. Central difference Gradient filter
2. Sobel filter
3. Mean filter
4. Median filter
Note: You can use fspecial, imfilter functions in Matlab to accomplish the above and below tasks. Do not use imgradient.

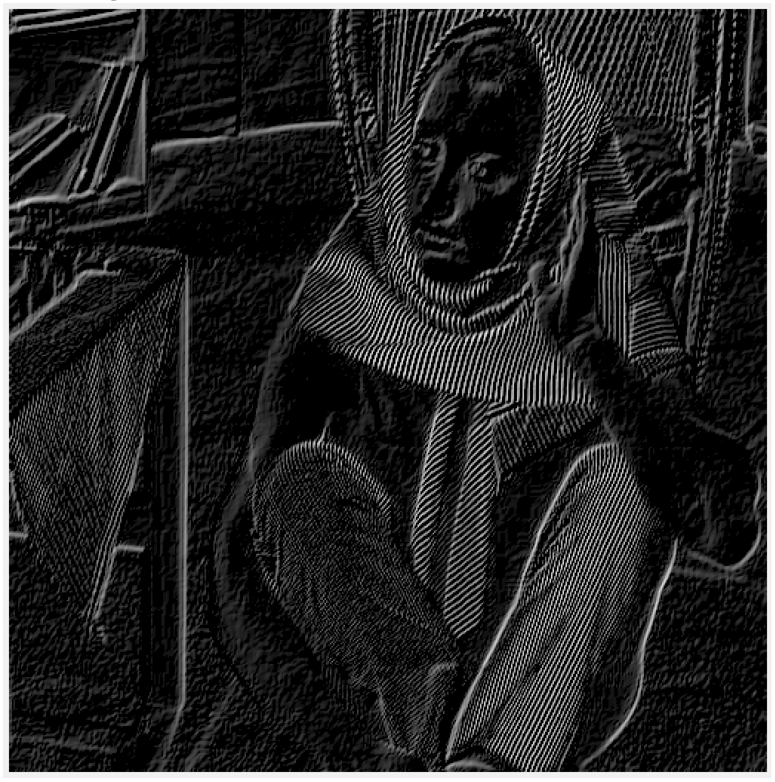
1. Central difference Gradient filter
   Using imgradient() => Should be the correct answer



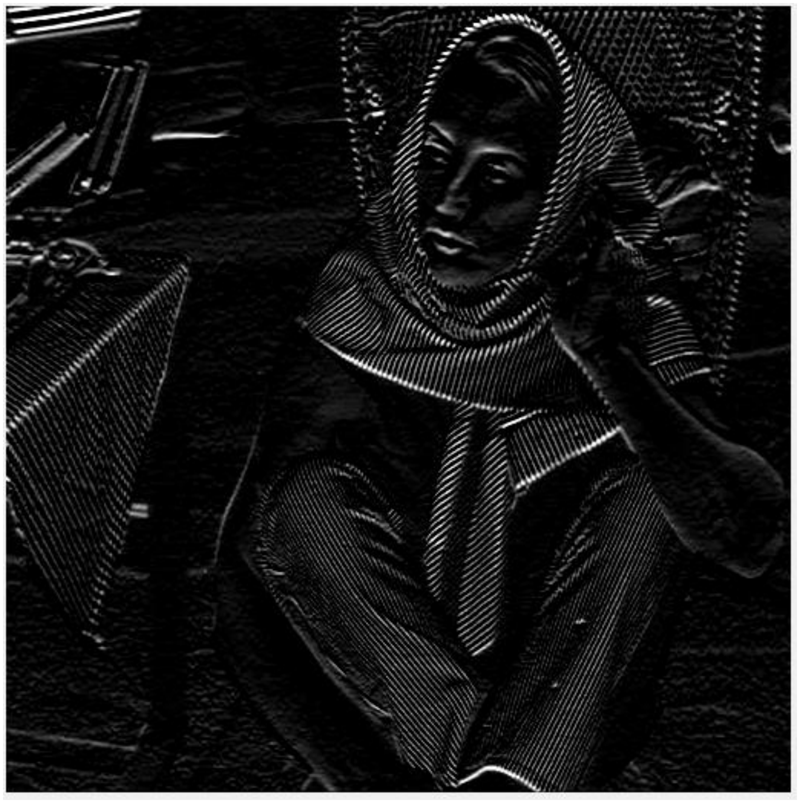
Using x_filter=[-1, 0, 1]/2, y_filter=[-1; 0; 1]/2, gradient_image = sqrt(x_image .* 2 + y_image .* 2);
=> **Mathematically correct**, **but the output image seems too dark**.



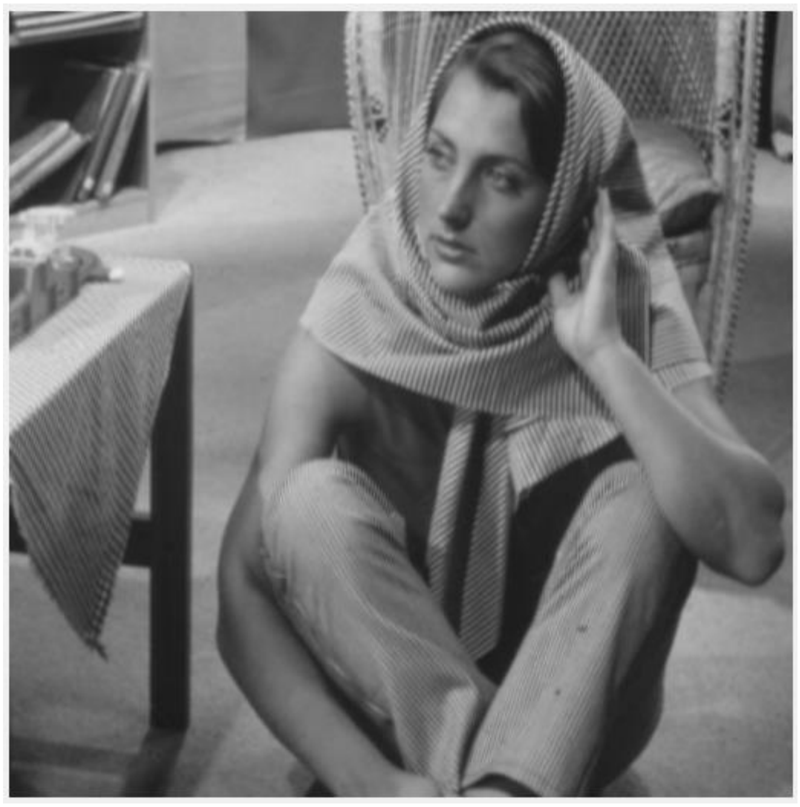
Using x_filter=[-1, 0, 1]/2, y_filter=[-1; 0; 1]/2, gradient_image = sqrt(x_image .* 2 + y_image .* 2);
=> **Mathematically wrong**, but **the output image is more similar to the expected image** that result from imgradient()



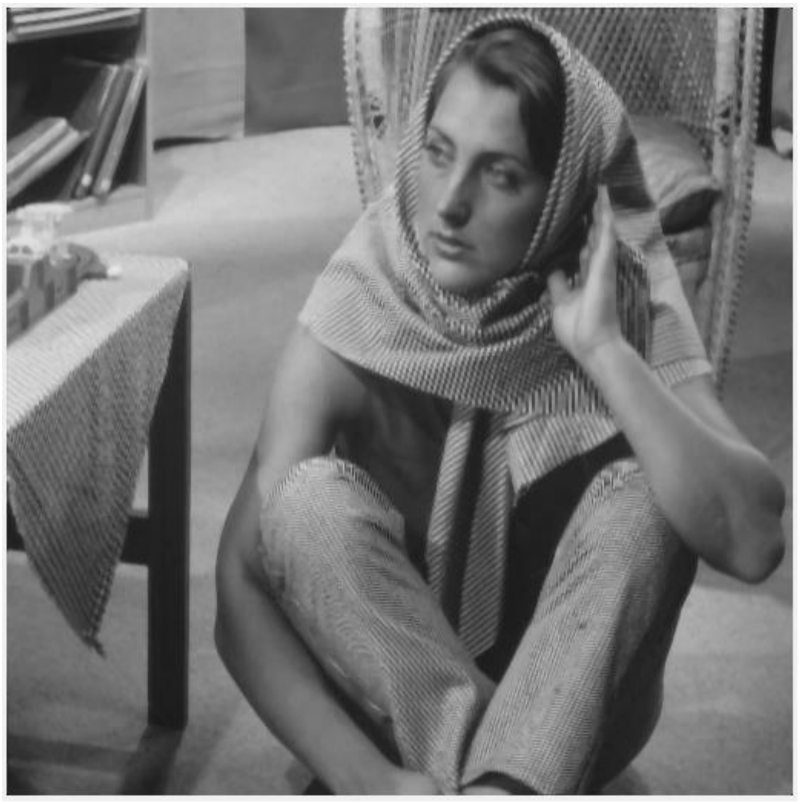
2. Sobel filter



3. Mean filter

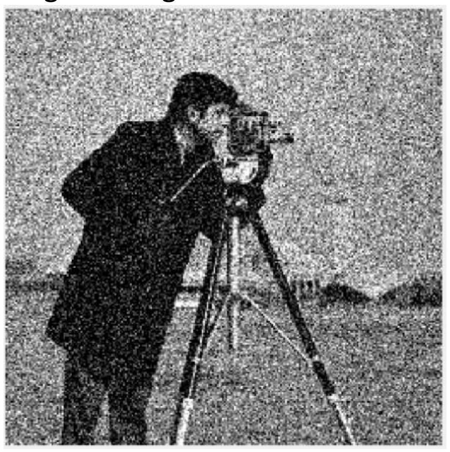


4. Median filter

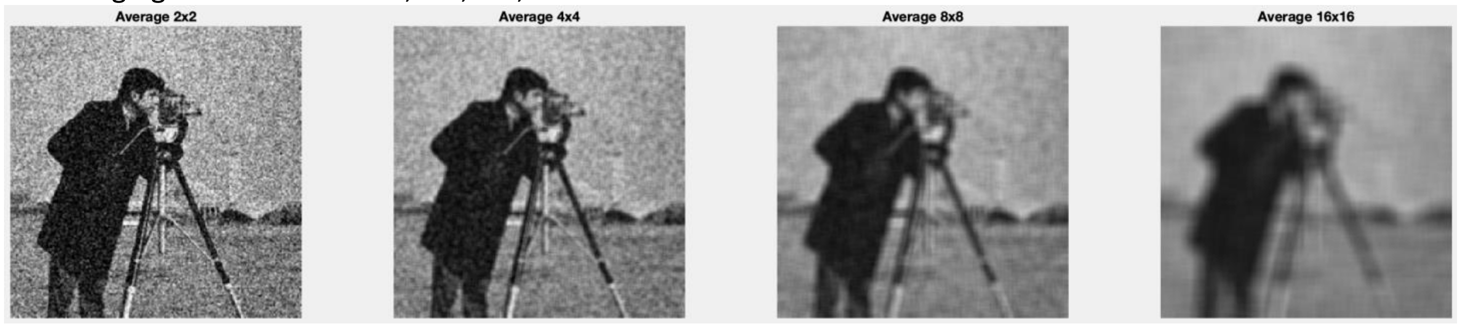


Smoothing:

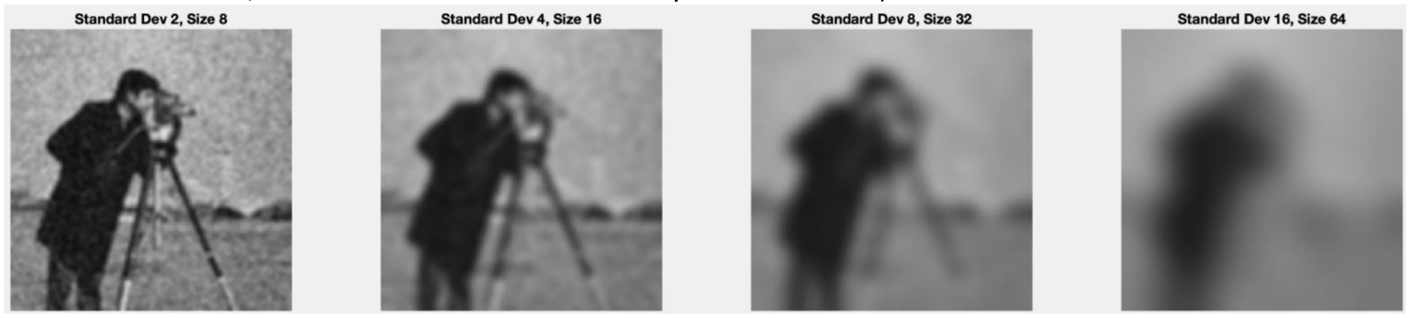
Original Image:



1. Averaging filters of sizes 2×2,4×4,8×8,16×16.

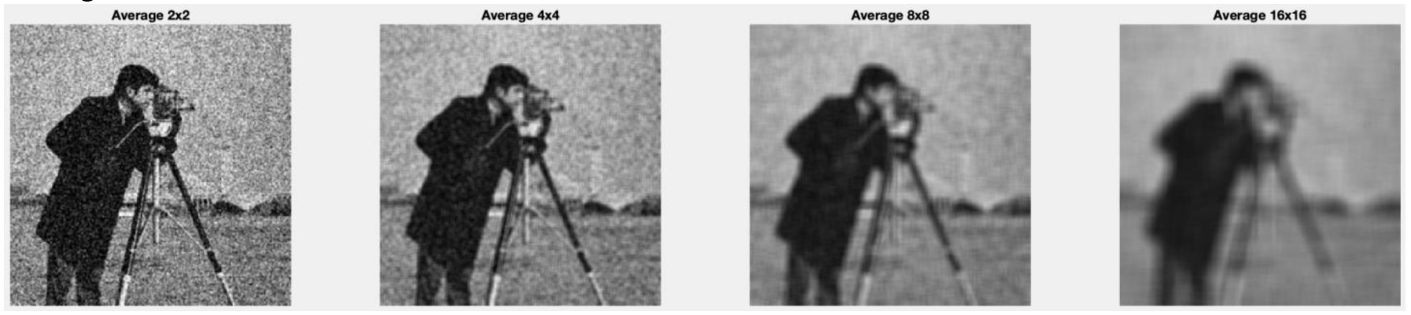


2. Gaussian filter of standard deviations 2, 4, 8, 16 (A good choice of gaussian filter size is 4 times its standard deviation, to include most of the variability within the box).

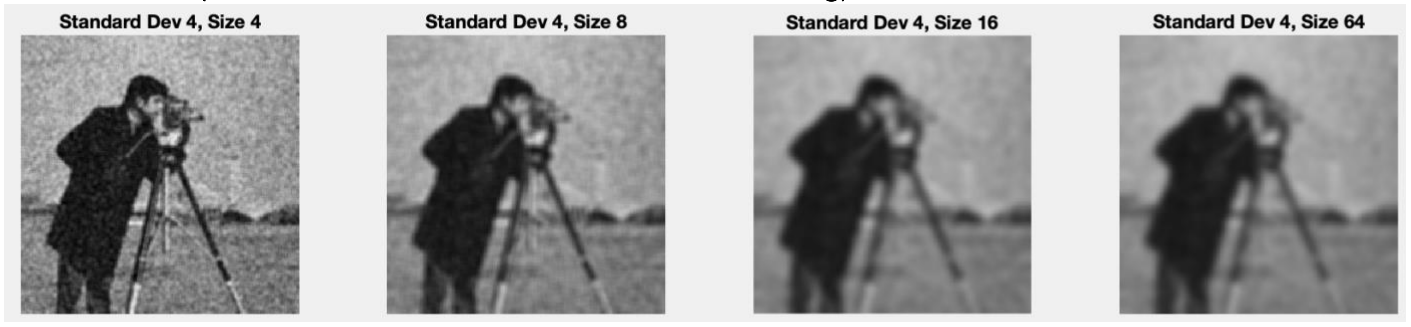


(1) Which filter works best?
In my opinion, both filters did not perform well on this task. However, I think **the average filter works better than the Gaussian filter** since average filter still **preserves some edges while alleviated most of the salt-and-pepper problem** (like the average 4x4 filter). Gaussian filter is more like blurring both the image information (like edges) and the noise simultaneously.

(2) What happens when you vary the box filter size?
Average filter

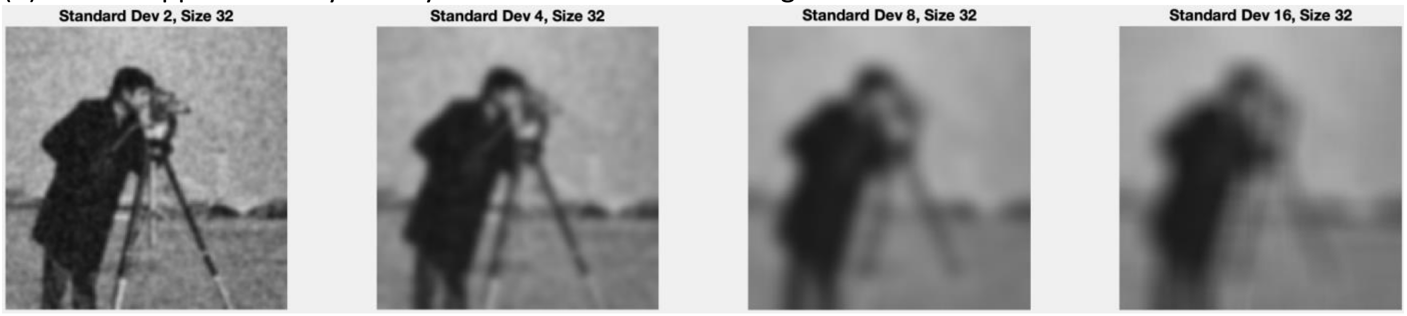


Gaussian filter (standard deviation fixed at 4 with size increasing)



From the pictures above, we can know that **both the Average filter and Gaussian filter will become blurrier if the box filter size increases**.

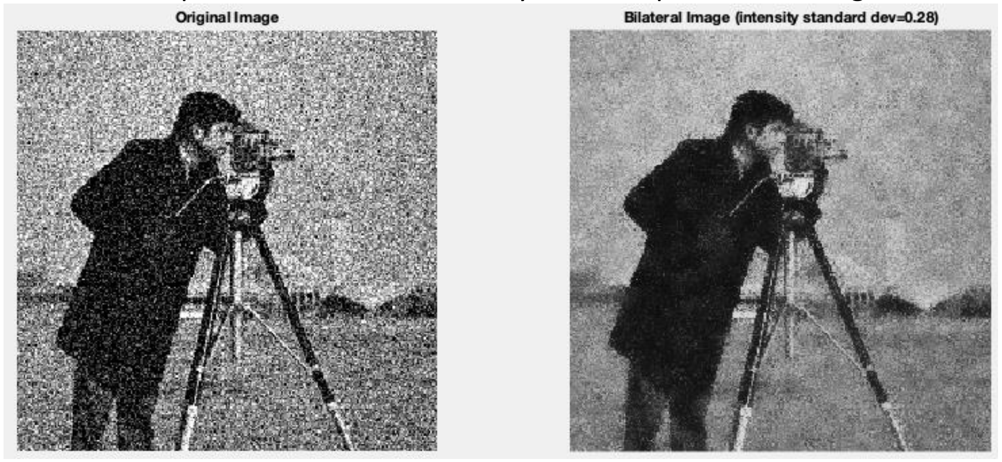
(3) What happens when you vary the standard deviation of gaussian filter?



Increasing the standard deviation of the Gaussian filter will make the picture **becomes blurrier**. Moreover, compared the picture above with the picture in the previous question that fixed the standard deviation but increased the size, we can find out that **fixing the size and increasing the standard deviation will make the picture blurry more dramatically**. Merely 4 times the standard deviation the output picture will be blurrier than 16 times the size of the box filter size.

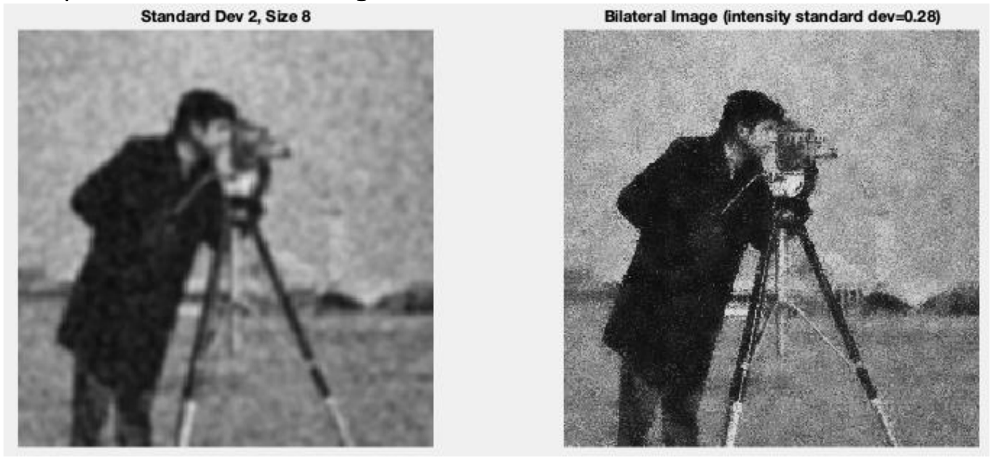
Grad credits: Edge preserving smoothing

(1) Find the best parameters that work for you and report them along with the denoised image.



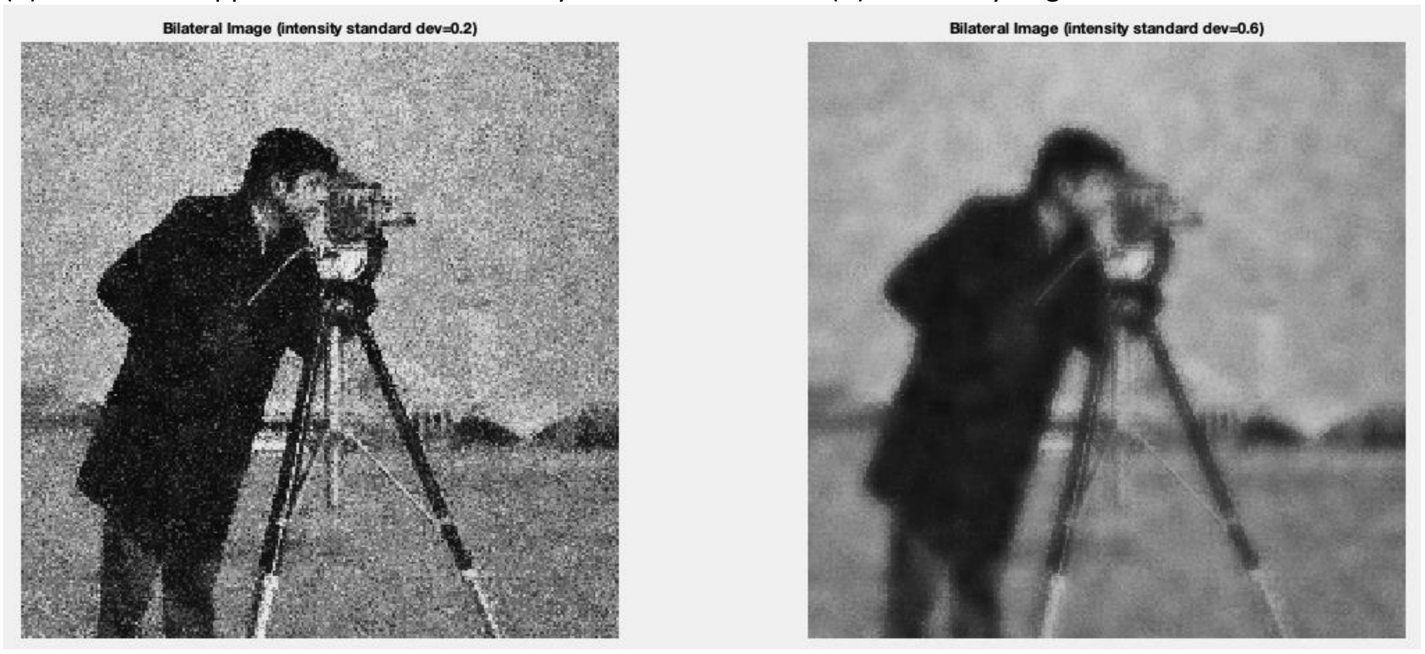
Weight = 5
Spatial standard deviation = 5
Intensity standard deviation = 0.28

(2) Compare the denoised image from bilateral filter with the one from the best Gaussian filter.



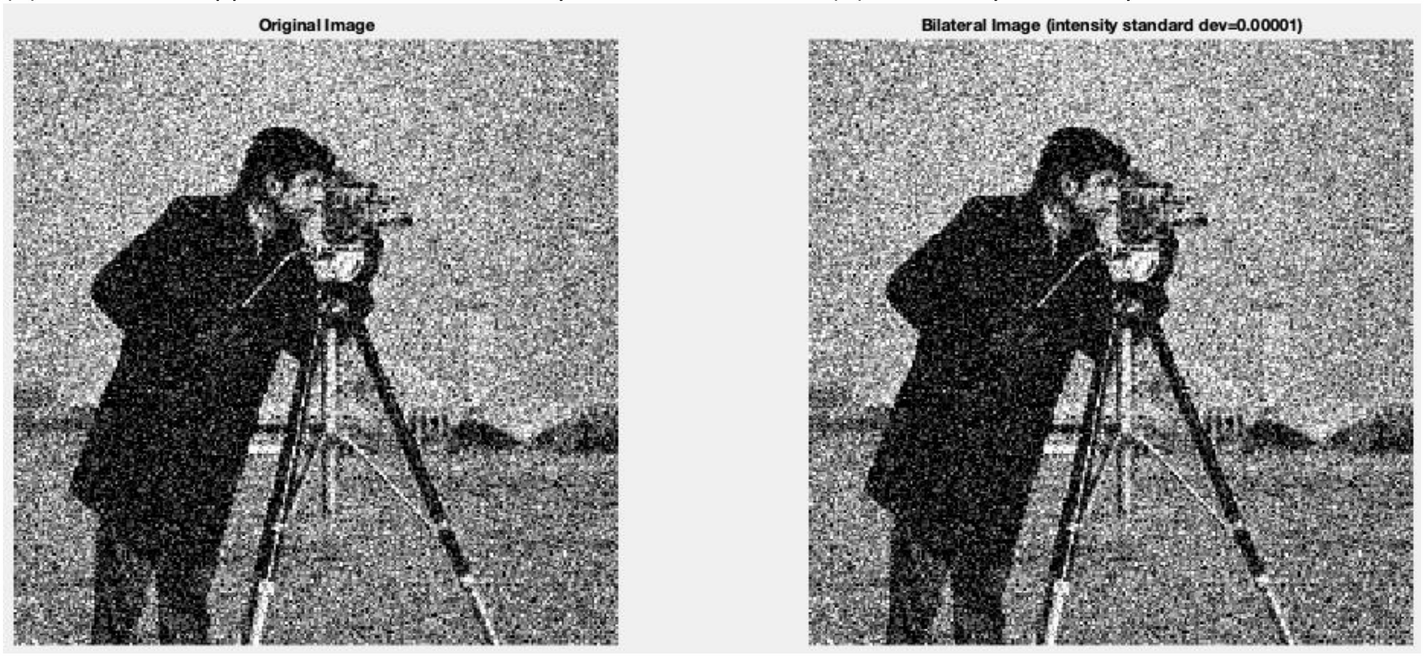
We can clearly see that the result after **Bilateral filtering outperforms the picture after going through the best Gaussian filter with less noisy points and less blur**.

(3) What will happen if we choose intensity standard deviation (σ) to be very large?


Bilateral Image (intensity standard dev=0.2)
Bilateral Image (intensity standard dev=0.6)

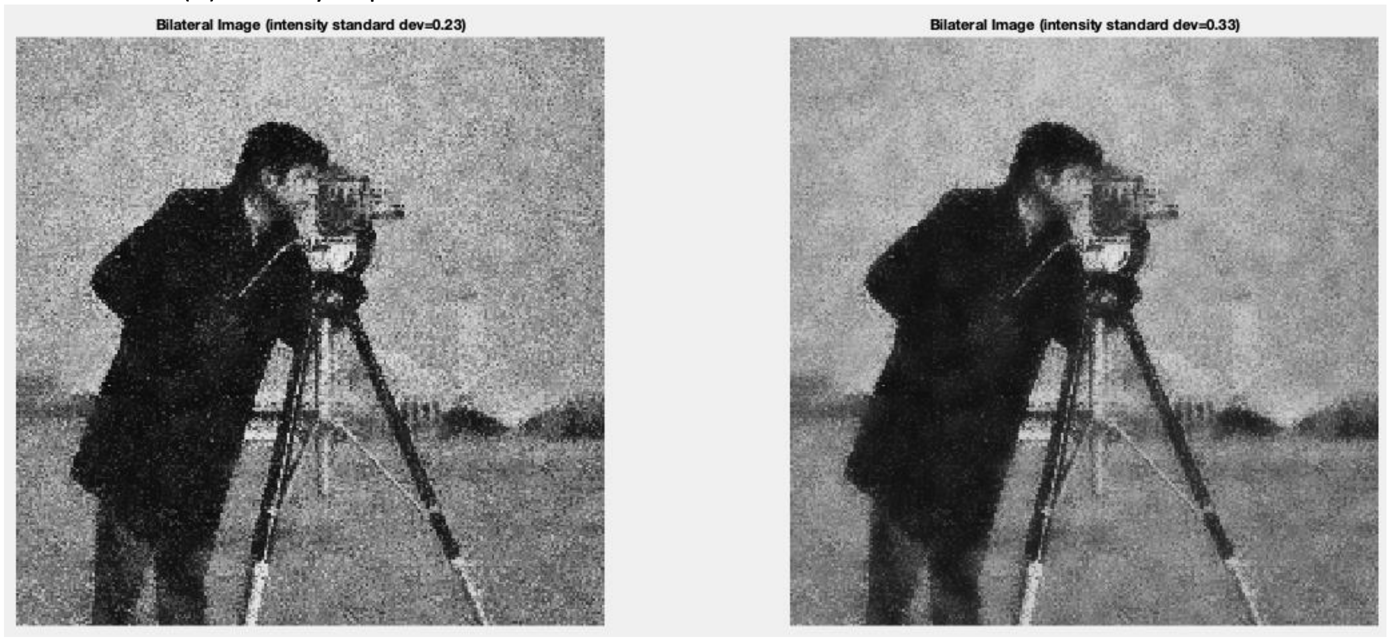
If the intensity standard deviation (σ) is larger, the filter **is more powerful at alleviating the noisy points. However, the output image is blurrier**.

(4) What will happen if we choose intensity standard deviation (σ) to be very small, say almost zero?



The output image will **likely to be identical to the original image**.

(5) For a particular noise standard deviation, what value (or range of values) of intensity standard deviation (σ) would you prescribe?


Bilateral Image (intensity standard dev=0.23)
Bilateral Image (intensity standard dev=0.33)

I would prescribe a **value range from 0.23~0.33 since the output image will be not that noisy and preserves most of the original image's information**.

Problems encountered:

First, I think the tutorial has well-defined the basic functions we will use in this assignment, which helps me to better find out what specific function to use by checking the function on the MATLAB official website and helps me to forestall some function finding efforts.

The most challenging part for me in this assignment is to **implement the Central difference Gradient filter**. Since we cannot use imgradient() directly, we must find out how to combine the x filter and the y filter to present the result. **I eventually found out that utilizing the correct mathematical form of the filter makes the result seems a little bit darker**. Thus, I've tried lots of different ways to modify the operations and functions and eventually found out that **changing the .^ operation to the .* operation makes the output more similar to the result coming from imgradient()**.

Also, the piecewise element calculation in MATLAB causes me some time to find out the bug of getting the wrong value since it is totally different to operate calculation element by element and to calculate it with matric calculation.

Surprises:

**I am amazed by the robustness of the bilateral filter**. It is super powerful in denoising images but preserves most of the edge details. Moreover, I considered that the **Gaussian filter will be also useful if we want to hide some important information in some specific area**. Lastly, I am impressed by the **Sobel filter which efficiently highlighted the edges of the picture perfectly**.