Digital Image Processing
Homework 4

Steven Jennings

05 March 2018

Overview

The goal of this assignment was to understand how noise can affect digital images, and how to restore images corrupted with noise. This goal was to be accomplished through the following parts:

- Part I: Generate a “perfect” image
- Part II: Generate noise signals
- Part III: Apply those noise signals to the “perfect” image
- Part IV: Attempt to restore the noise-corrupted images with known image processing techniques

Procedure and Analysis

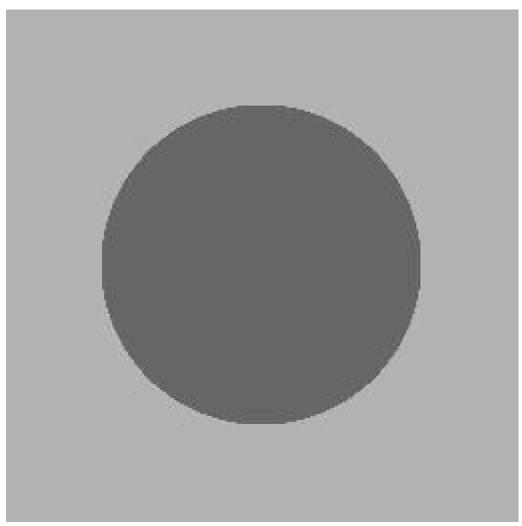
Part I: “Perfect” Image Generation

The first part was to generate an image that we can use for this assignment under the following conditions:

- Create a circle at the center of the image of radius 80
    - Must be a normalized value of 0.4
- The rest of the image must be a flat background color
    - Must be a normalized value of 0.7

I decided to create an image with dimensions 256x256.

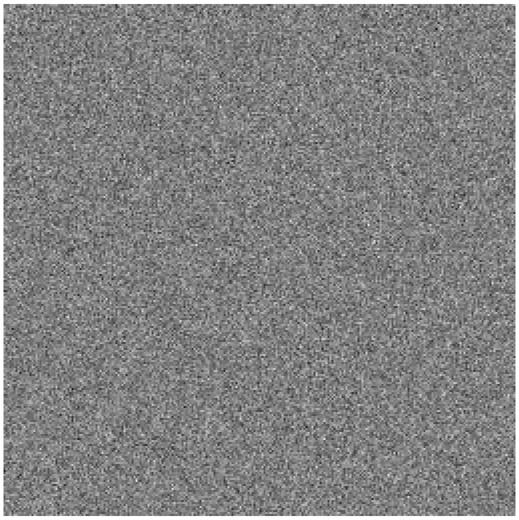
Here’s the resulting “perfect” image.



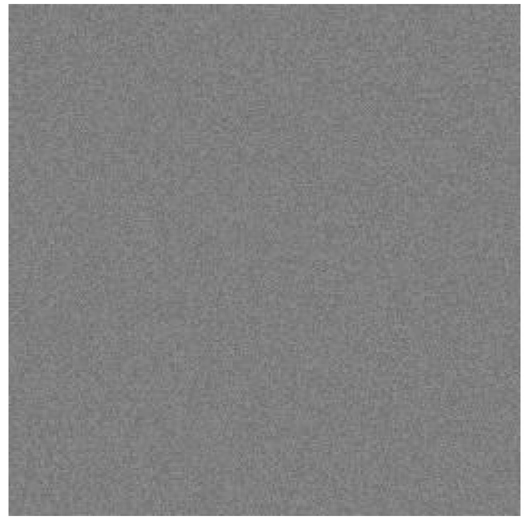
Part II: Noise Generation

The second part was to create random noise signals to apply to the image from part I.

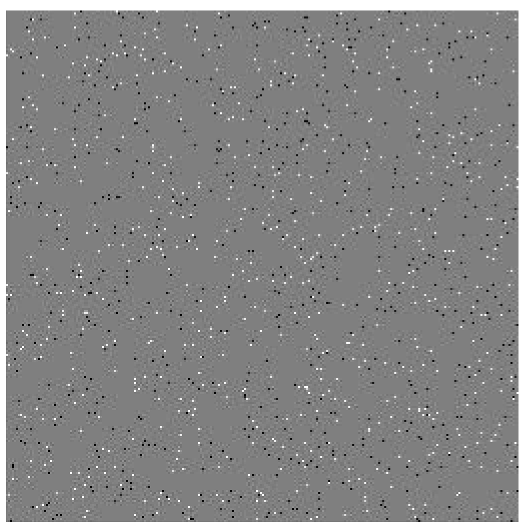
Gaussian noise generated.



Uniform noise generated.



Salt & Pepper noise generated.



Part III: Adding Noise to Image

The third part of this assignment was to add the noise from part II to the original image generated in part I. Here's each of the three new images, with noise.

Image + Gaussian noise.

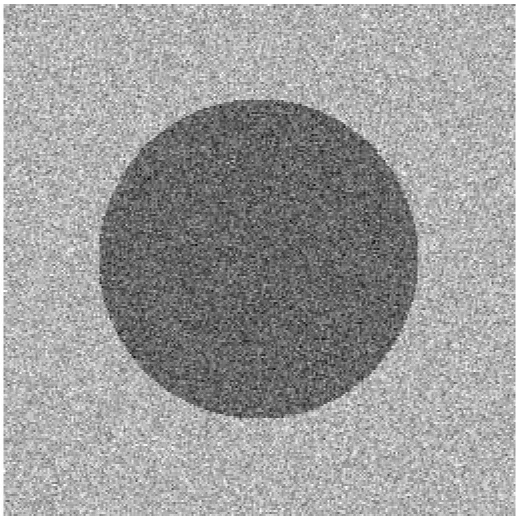


Image + Uniform noise.

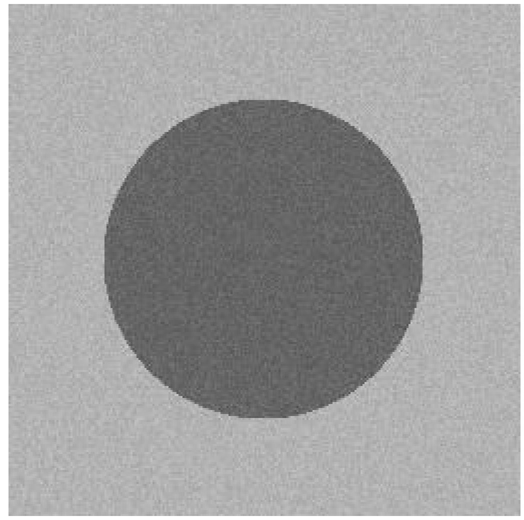
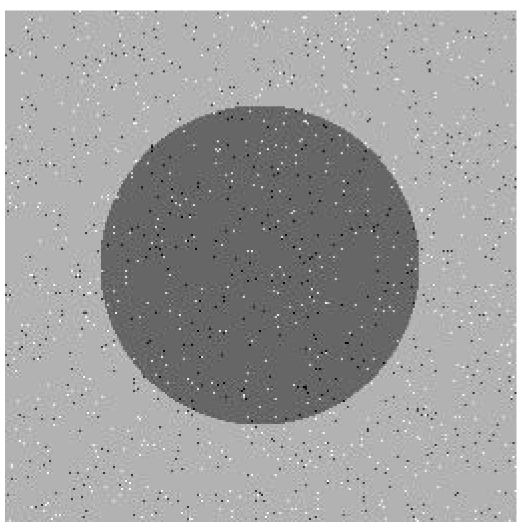


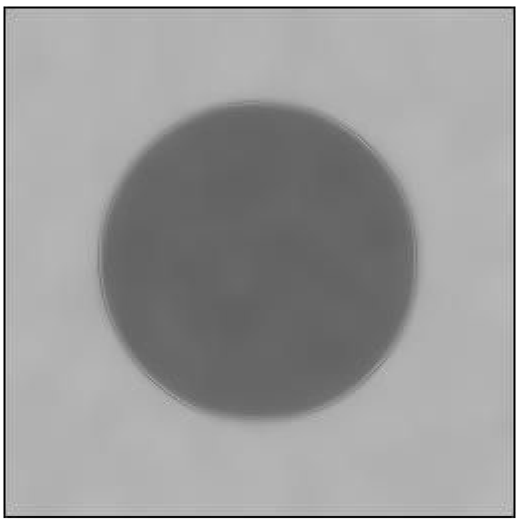
Image + Salt & Pepper noise.



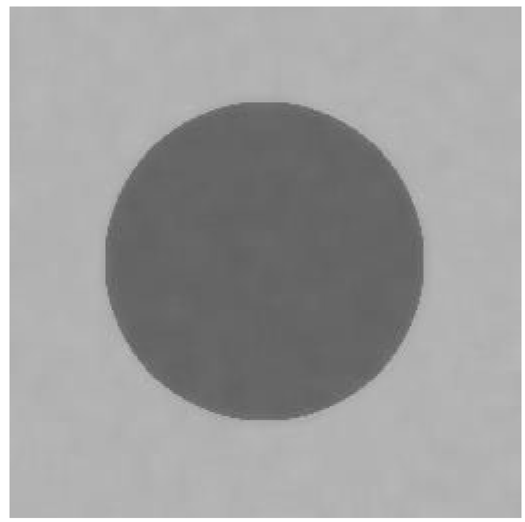
Part IV: Restoring Each Noisy Image

The final part of this assignment was to attempt to restore the noisy images from part III using existing knowledge of image processing. I mostly used the median filter to smooth out the noise before trying to enhance the edge information in the images. This approach worked almost perfectly with the uniform noise and salt & pepper noise, but it was only fairly decent with the Gaussian noise. Here are the results of the 3 restorations.

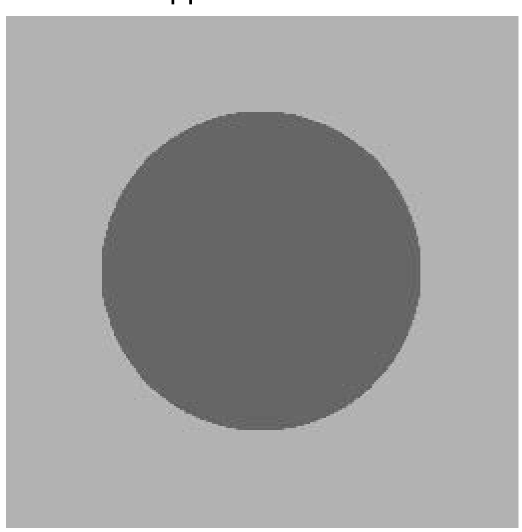
Gaussian noise restoration.



Uniform noise restoration.

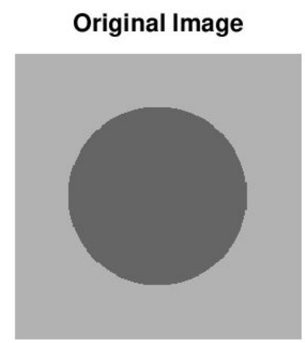


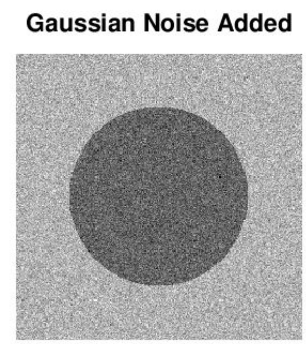
Salt & Pepper noise restoration.

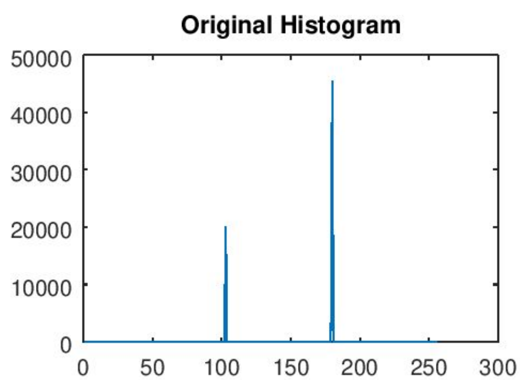


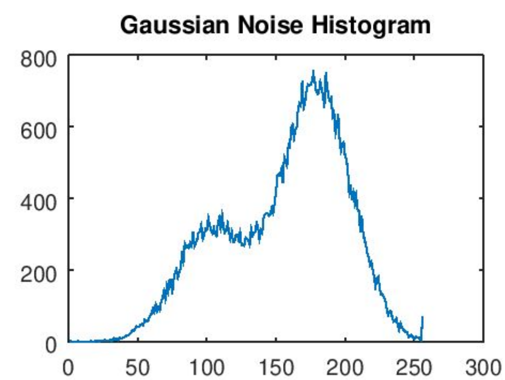
Extra Information

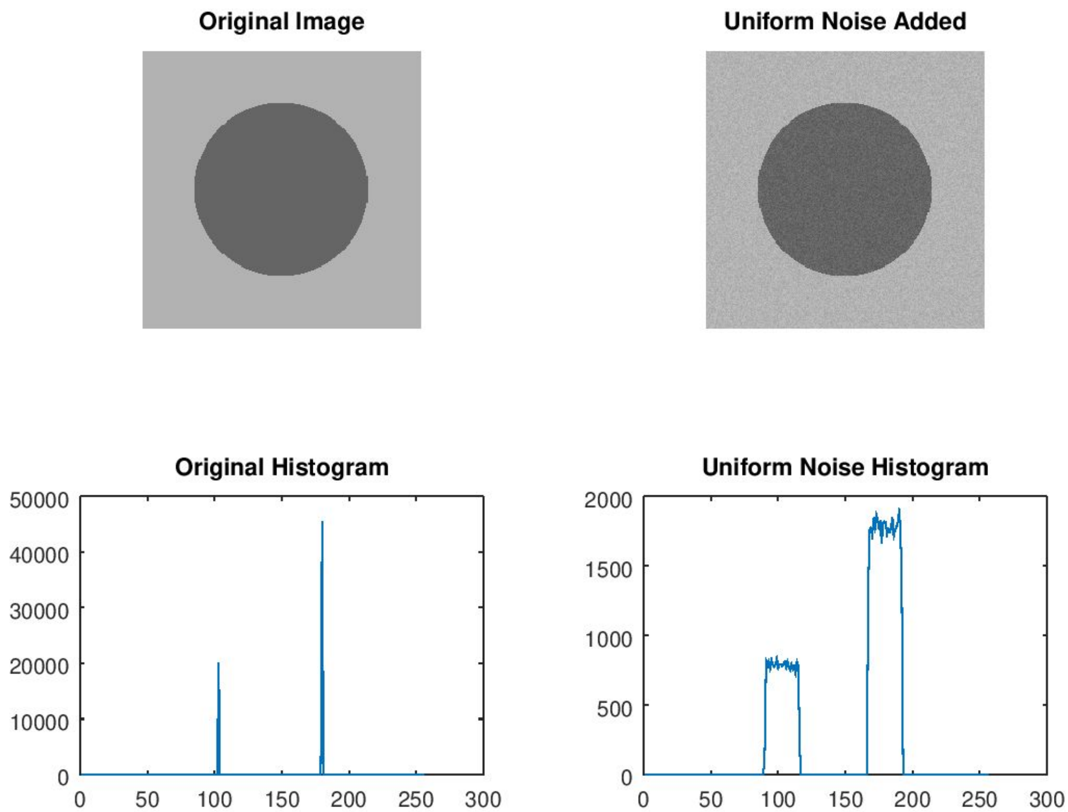
The following pages are a lot of extra information generated by my code. All of this information can be found in the “Figures” folder in my .zip archive.

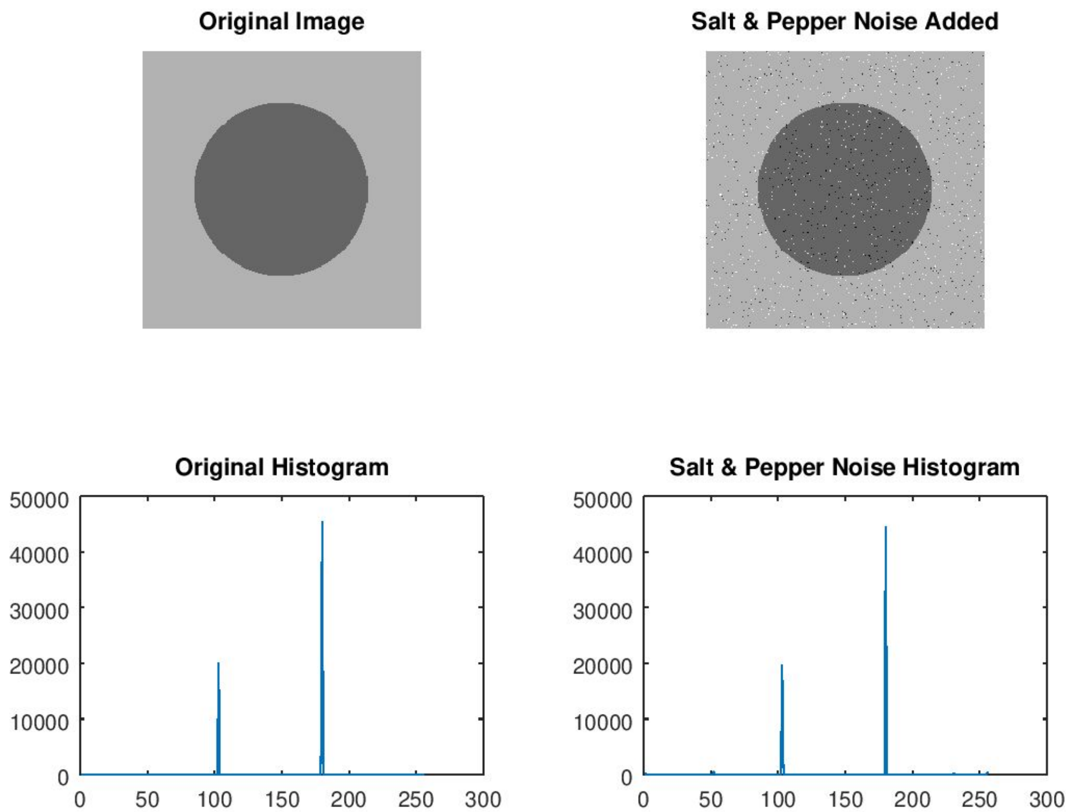




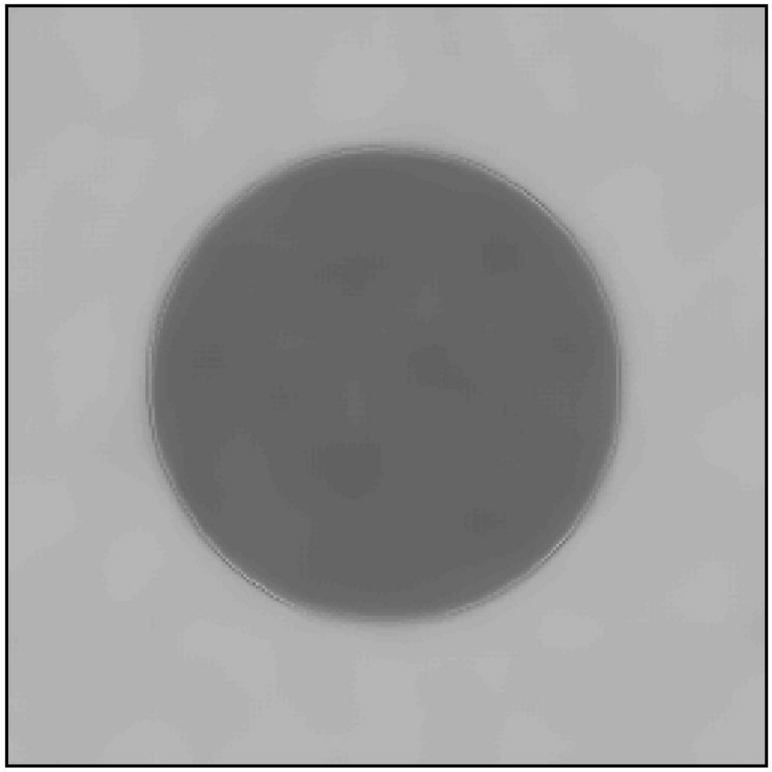


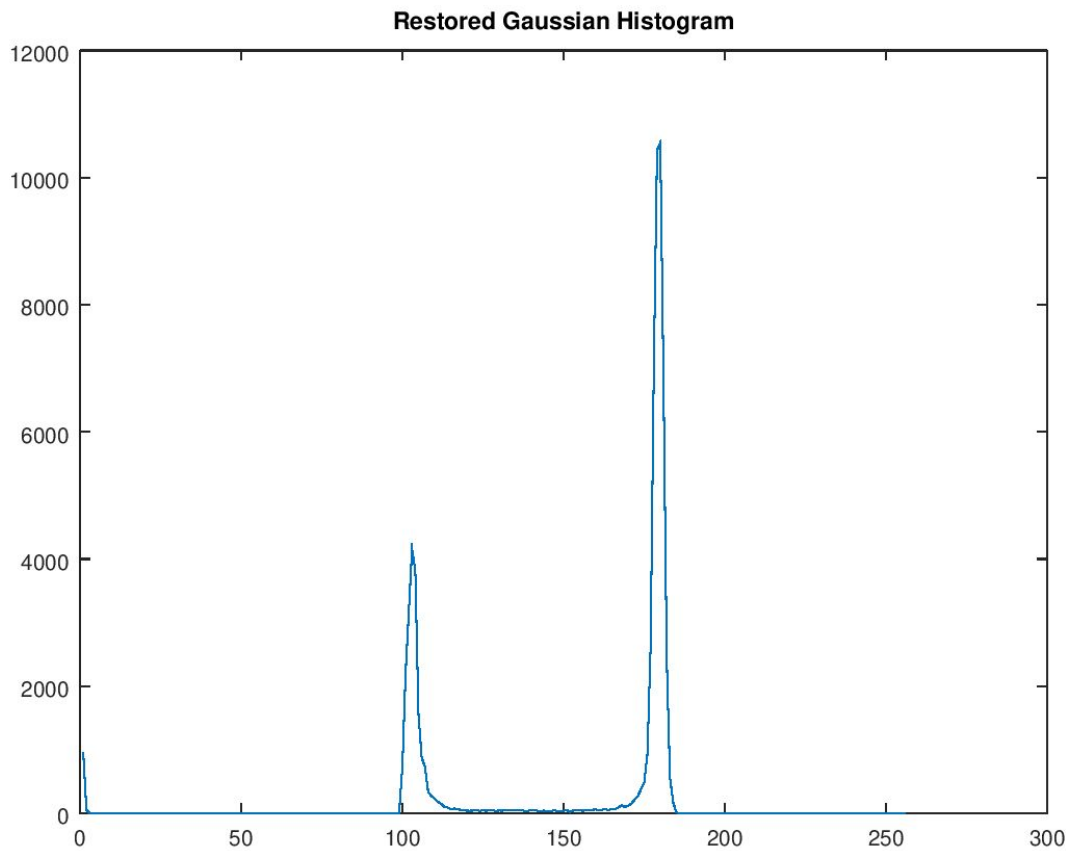



Original Image
Uniform Noise Added
Original Histogram
50000
40000
30000
20000
10000
0
0 50 100 150 200 250 300
Uniform Noise Histogram
2000
1500
1000
500
0
0 50 100 150 200 250 300

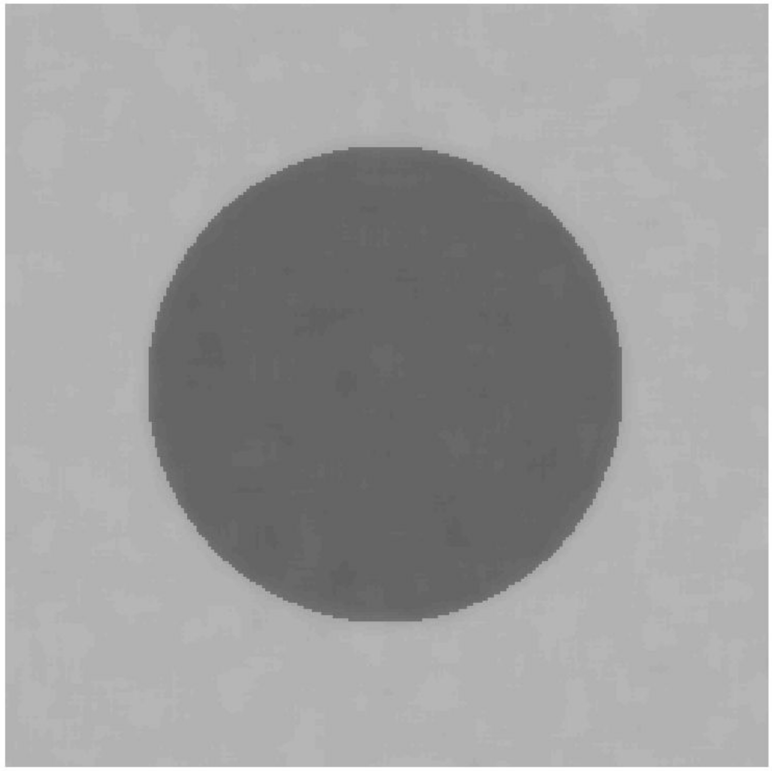

Original Image
Salt & Pepper Noise Added
Original Histogram
Salt & Pepper Noise Histogram
50000
40000
30000
20000
10000
0
0
50
100
150
200
250
300

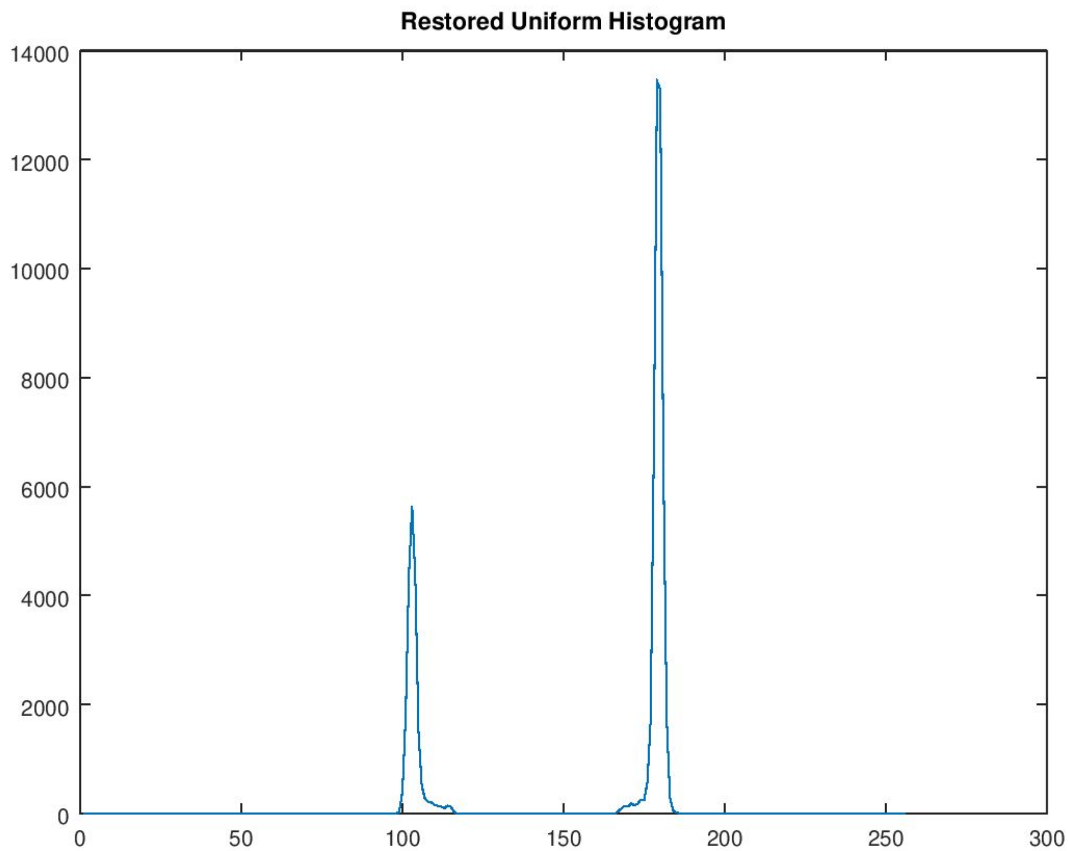
Restored Gaussian Noise



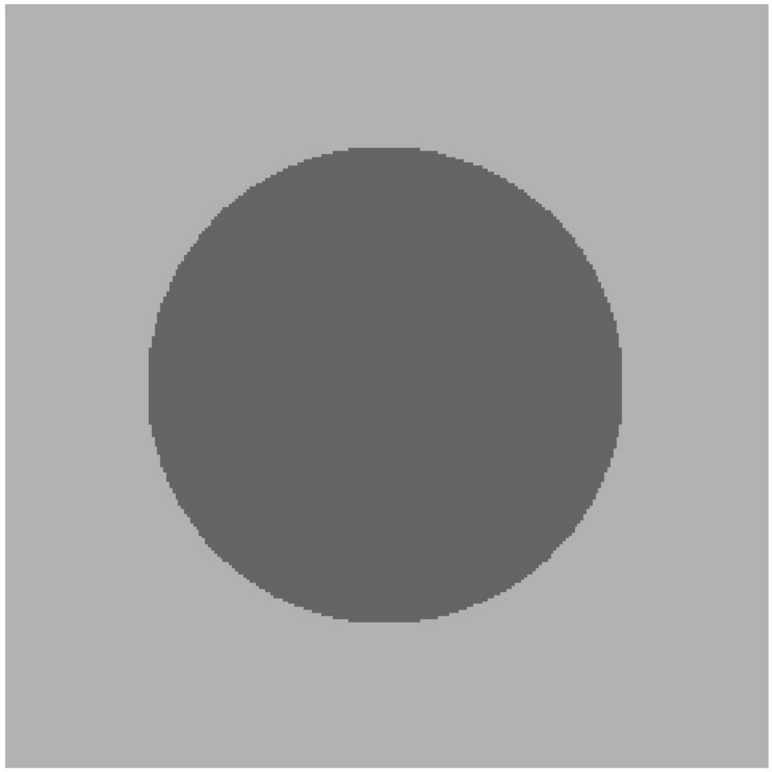

Restored Gaussian Histogram
12000
10000
8000
6000
4000
2000
0
0
50
100
150
200
250
300

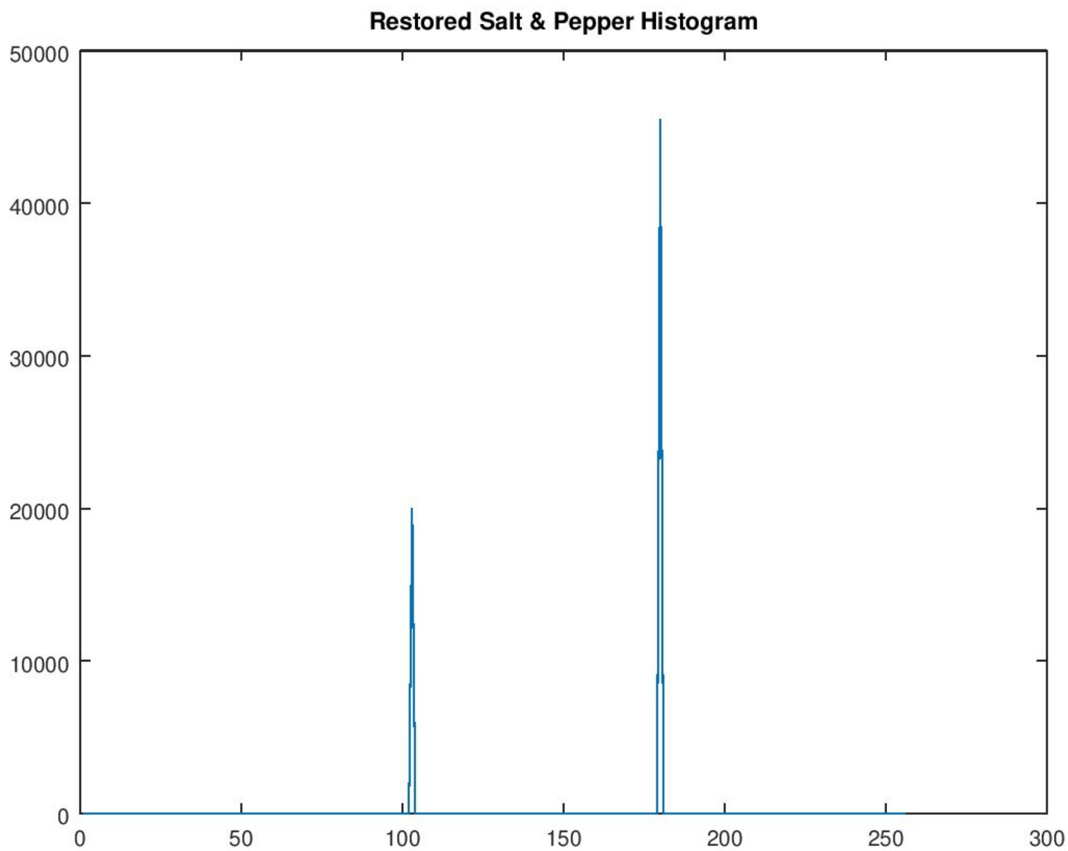
Restored Uniform Noise




Restored Uniform Histogram
14000
12000
10000
8000
6000
4000
2000
0
0
50
100
150
200
250
300

Restored Salt & Pepper Noise




Restored Salt & Pepper Histogram
50000
40000
30000
20000
10000
0
0
50
100
150
200
250
300

Conclusion

Dr. Jiang, you should have had us try implementing our own type of HDR+ processing, the way Google's newest phones have been doing it. You mentioned it in class a while back, but it would have been neat to generate an image, create several different versions of the image with noise, and then use pixel-by-pixel averaging techniques to create a noise-free image.

Regarding this homework assignment, it was good practice. The only result I am not really satisfied with is the Gaussian restoration result. It looks decent, but not as good as the other two results. Considering all things, the histogram for the Gaussian noisy image was horrific and the Gaussian noisy image itself was very bad, so the restoration result was pretty good in perspective.