Report on
Image Processing and Interpretation
Project 4

Submitted to
Dr. Bebis
for
CS474

October 16th, 2015

By:
Aaron Whitehead
and
Jeremiah Berns

This report examines several introductory computer processing practice programs. The programs cover such things as band filtering in the frequency domain, inverse filtering for motion blur, and homomorphic filtering methods.

Technical Discussion

As the overall structure of the project is broken up into three sections, the technical writeup will also be broken into three sections.

Problem one:

The first problem is a noise removal one. The basic idea is to take an image that has had a certain pattern used on it to generate "corruption" and then apply some sort of band filtering method to the frequency domain. The basic algorithm goes:

1. Take an image and apply some sort of modification to it that is based on a pattern, such as a sine wave.
2. Convert the image into the frequency domain.
3. Find a band range that has decent results and apply it to the frequency domain by removing the data in that band range.
4. Convert back to the spatial domain.

Essentially, by being able to go in and remove a single "band" in the frequency domain, we can remove a single instance of corruption without having to take much data from the overall image.

Program two:

The second program is a image restoration problem, that uses different types of image restoration in order to compare. The different methods are applied to an image that has a known corruption value H(u,v) along with a set of random data. This is a difficult problem to approach due to the unknown random data on top of the known corruption algorithm. The algorithm is:

1. Take an image and convert it to the frequency domain.
2. Apply a motion blur algorithm on the image in order to simulate a blurred photo
3. Convert back to the spatial domain
4. Apply random Gaussian noise.
5. Test different types of filters.

Program three:

The third problem is a homomorphic filtering technique one. This problem essentially boils down to simplifying light gradiants in an image. The algorithm for this is as thus:

1. Apply the ln function
2. Take the Fourier Transform of the image.

3. Apply High-Pass filtering
4. Take inverse Fourier
5. Apply the exp function
6. Display the magnitude

Results and Discussion

Once again, the results section, like the technical discussion section, will be broken into three core parts.

Problem one:

Problem one was a noise removal function. The desired result was an image that did not have the obvious corruptions on it that had been added previously by applying a sine wave in the frequency domain. By converting it to the frequency domain and narrowing down a particular band of complex data to remove, it became possible for us to remove only the corrupted section without leaving big blurs or blank spots on the image at the end. However, in order to show the difference, a spatial Gaussian filter was also applied with mask sizes 7 and 15, with a side by side comparison below.

Given

Band filtered

Gaussian 7

Gaussian 15

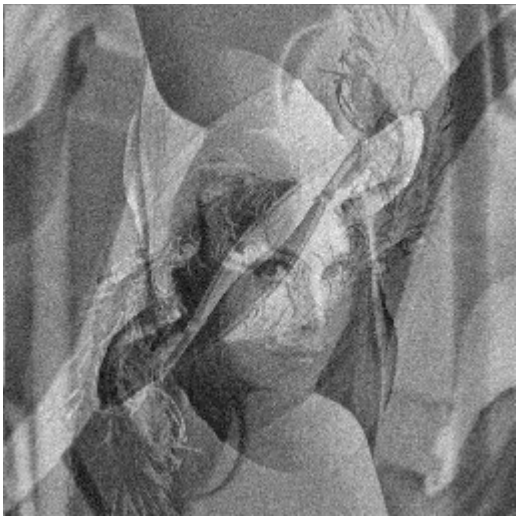
So as can be seen by comparing above, the band frequency rejection was a great success. The given image is covered in what can be described as a criss-cross pattern on lines caused by the sine wave corruption. In the Gaussian 7x7 filter, you can see that things have actually gotten worse, as the lines have become larger and have spread out over more of the previously untouched image. This is only worsened by increasing the the mask to 15x15 as can be seen, the entire image starts to look darker since the lines have spread so far, and there is almost no part in the image that is untouched now, definitely a move in the complete wrong direction. However, when you consider the band reject filtering method, the lines are basically gone. On careful observation, one can see a sort of heat haziness affect where the lines were at one point, but it is very pleasing to the eye and the details are sharp and crisp. Definitely a successful process.

Problem two:

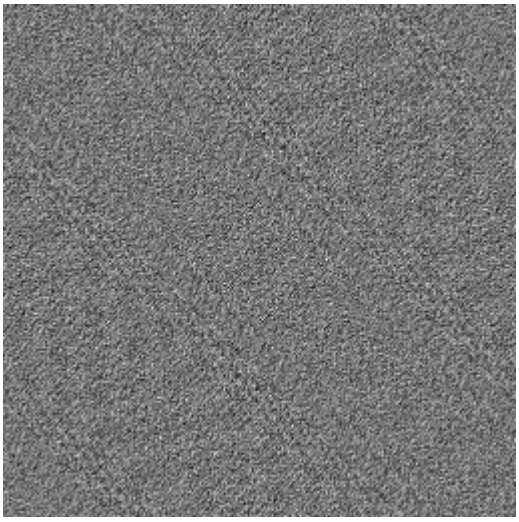
Problem two caused some problems that were somewhat dealt with, leading to interesting results. The main idea here was to create a h(u,v) set which is the information of the corruption for a motion blur affect. Then simply multiply it into the frequency domain of the image, convert back, throw some random noise in n(u,v), and then attempt to remove it. However, since you are aware of the h(u,v) an inverse approach of just dividing by the motion offset will achieve the result generally, minus the data that has been randomly messed with as we do not know what that random data is. However, in my implementation of the code there was a small quirk in my motion blur. The motion blur filter almost works, except the “secondary” image, IE the offset image that will make things look blurry, happens to be flipped on both the vertical and horizontal axis. Thus, when applying the inverse and the Weiner filter, interesting things happen.

The given

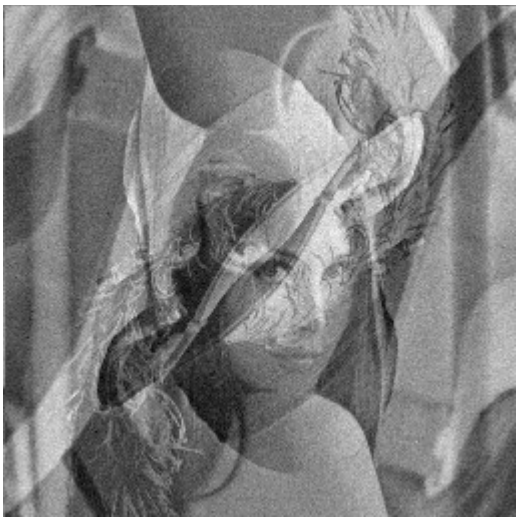
Given with Gauss noise



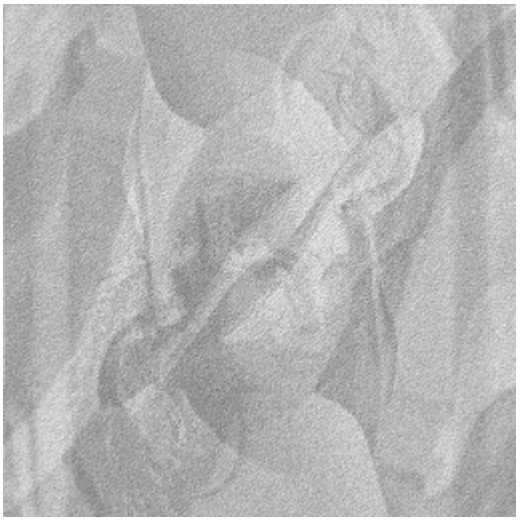
The blurred image with noise



Inverse filter on left image



Wiener filter k=1000



Weiner filter k=0.001

So as can be seen, there is some sort of strange flip going on with the data for the blur that I, Aaron, was unable to overcome. By dividing each point with h(u,v) I get the inverse filter image, which does not seem to be correct. However, the Wiener filter with k being very large, while very very subtle, does seem to have been a success. To begin with the random noise seems to be diminished, or faded away. Furthermore, the lady in the image seems to be much more substantial and less see through, which I deduce would be a very promising thing if the motion blur was applied correctly. Finally, in the Weiner filter with a very small K, it can be seen that the image mostly just seems to be fading away, not really doing anything overly noteworthy, beyond the fact that it does not look too horrible, simply that every data point has been overly normalized downwards, or some similar issue.

Problem three:

The third problem was an interesting problem in that the results are not as clear cut, due to having to try several different values of Yh and Yl in order to understand how their changes were affecting the overall outcome. Below you can see the given image, along with several different Yh and Yl pair results.

Given image

0.5/1.5 for yl/yh

0.5/1.9 for yl/yh

0.1/1.9 for yl/yh

0.1/1.5 for yl/yh

0.9/1.9 for yl/yh

So to my personal untrained eye, it seems that the changing of the values of yl and yh simply change how big the difference is between sections of light differences, along with how much light is in each section at the end. The original image is a picture of a smiling young girl in what seems to be a dark room with a spot light on her face. In using 0.5/1.5 for the input values, one can see the

different sections of light that have been defined. As yh increases in the first example, the only discernible difference to my eye is a overall darkening of the middle section, while the surrounding sections seem to be the same shade. As yl drops in the next example, a darkening of the main white splotch on her face can be observed. As yh starts to drop, the overall image begins to brighten. This set of input values seems to offer the greatest visual result, in that the difference between each section is not too stark, and each layer of the light can be clearly seen. In the final example with both values raised to max, it can be said that while the middle is very bright, the difference of the first layer with the rest is much to big, leaving everything but the very front layer of light to be almost unnoticeable.

Division of Labor

Jeremiah Berns was mostly in charge of designing and writing problems one and three, while Aaron Whitehead was in charge of problem two. Aaron was also in charge of creating and writing the write up. As problem two is the only problem that does not seem done at first glance, I, Aaron, would like to request that Jeremiah be penalized as little as possible due to me not being able to finish what we both agreed was my section. I would also like to not receive a 0 since my section did not function correctly as I seem to be off by only some strange value on my motion function and that caused the inverse code to fail even though it seems to be written correctly. Honestly, this was my mistake, and I would appreciate it if my parter did not suffer for my lack of planning. If you are interested, you can find the code on my github:
http://github.com/BigMacStorm/imageProcessing