Computer Vision Project 1

Joel Yuhas

February 2019

1 Introduction

Computer vision relies on the ability to transform and analyze images in order to better understand them. This project involved utilizing several techniques in order to change a grey scale image in a desired way. These techniques involved high pass filtering, low pass filtering, median filtering, edge detection, 2-D Fourier transforms, and Inverse 2-D Fourier transforms. Their effects were used in a variety of ways to see the effects of things like "salt and pepper" noise, the Meas Square Error of transformed images, and even the combination of different phases and magnitudes. Overall, the main objective was to better understand the effects these methods have on images and how they relate to one another. Python scripts were created that utilized these methodologies and produced the results that were analyzed.

2 Filtering In The Space Domain

Methods: The spacial domain is the common way images are presented to us in everyday life. In this section, a grey scale image in the spacial domain was chosen so that filters could be applied to it to demonstrate their effects. Figure 1 shows the famous "Lena" picture and that was used. This picture was chosen for its recognizability as well as its range of edges, lighting, and shapes.

Figure 1: Grey Scale Lena

In this section, a high pass and Gaussian low pass filter were applied to the image. A 3x3 filter was used with '-1's around the border and a '6' in the middle. This filter was chosen since it was able to bring out the high frequencies without loosing too much detail. This was done using the "ndimage.convolve" command, which allowed the edge calculation method to be set. The "reflect" method was chose, which mirrored the edge values outside the bounds of the original image when the convolution took place. The reflect method allowed for the texture to be preserved, vs the "constant" or "nearest" method which simply extened certain values.

After, a low pass filter was applied. This low pass filter was 5x5 with the value of 1/25 in all elements. The effect of the filter drastically blurred the image. Continued application of the filter blurred the image even further. Finally, a program was written that used Sobel edge detector masks.

Results: This section shows the resulting images mentioned above. Figure 2 shows the effects the low pass filter while figure 3 shows the effects of the high pass filter. As touched on in the method section, the low pass filter blurred the image while the high pass filter brought out some of the edges. Figure 4 shows the from the edge detector filter, with the vertical and horizontal edges placed side by side. Figure 5 shows the final result when the two images were summed together.

Figure 2: Low Pass Filter

Figure 3: High Pass Filter

Figure 4: Edge Detection Horizontal and Vertical

Figure 5: Edge Detection Sum

3 Filtering A Noisy Image

Methods: Unwanted noise can effect an images clarity. In this section the effect that certain filters had on noise was experimented with. "Salt and Pepper" as well as "Gaussian" noise was added to the greyscale Lena image as seen in Figure 6 and Figure 7 respectively. Salt and pepper noise has the effect of fully blacking or whiting out random pixels while Gaussian noise added random values to random pixels. It should be noted that the mean was set to '0' for both but the variances were independently changed for each in order for the noise to be visible. The signal to noise ratio (SNR) was calculated using the

following equation: SNR = $10\log_{10}\frac{\sigma_i^2}{\sigma_n^2}$

Figure 6: Salt and Pepper Noise

Figure 7: Gaussian Noise

First, a low pass filter and a median filter were used to attempt to filter the noise. The low pass filter was the same as before while the median filter took the median from a 3x3 grid of each pixel. Lastly, the noisy images were processed with the Sobel edge detector again in order to observe its effects.

Results: Low pass and median filters were the first set of results. Figure 8 and 9 show the low pass and median filters of the Salt and Peper noise while

Figure 10 and 11 show the filters for the Gaussian noise. As can be seen, the median filter did an excellent job at restoring the original image from the SNP noise while the lowpass filter made the image arguably worse. However, the effect seemed to be opisite with the Gaussian noise, where the median filter did worse and the low pass filter did better. This was somewhat expected, at median filters are naturally better at removing outliers while averaging, or low pass filters, are heavily effected by them. The SNR ratio was calculated to be 41.86 for the SNP and 43.73 for the Gaussian.

Figure 8: Low Pass Filtering of SNP Noise

Figure 9: Median Filtering of SNP Noise

Figure 10: Low Pass Filtering of Gaussian Noise

Figure 11: Median Filtering of Gaussian Noise

4 The 2-D Fourier Transform

Methods: Next, 2-D Fourier transforms were experimented with. The "np.fft.fft2" command was used to compute the 2-D Fourier Transform of the default Lena image. The Fourier transform effectively turns the image from its spacial domain representation into its frequency domain representation. First, the image was transformed using the "np.fft.fft2" command. This placed the image's origin

at the 4 corners. After, this process was repeated but with the "np.fft.fftshift" command, which put the origin in the center of the image. The two commands were then repeated, this time taking the log of each. Taking the log allowed the functions to be spread much more evenly over the image, and overall gave much more detailed and accurate results.

Afterwards, the inverse Fourier Transform was taken using the "np.fft.ifft2" command. This brought the image back into the space domain and reconstructed it. However, the reconstruction was not perfect. The difference between the original image and the reconstructed image was gathered by computing the Mean Square Error calculated by MSE = $1/\text{MN}\sum_{x,y}((f(x,y) - g(x,y))^2$

Results: Figure 12 and 13 show the Fourier Transforms before applying the log. Notice how the results are very hard to easily recognize. Figure 14 and 15 show the Fourier Transforms after the logs had been applied. The results were much clearer.

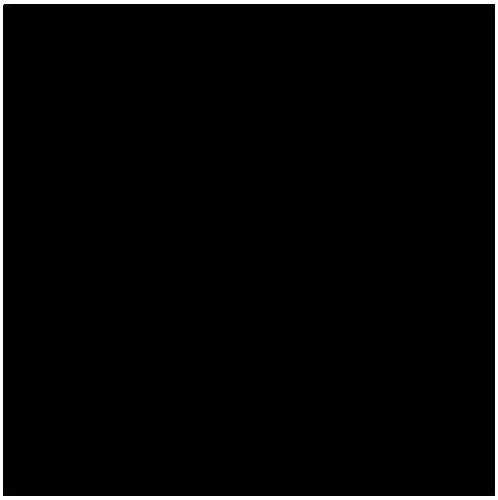

Figure 12: DFT

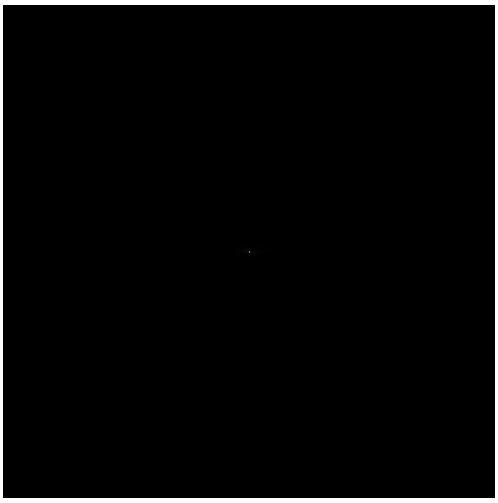

Figure 13: DFT Shifted

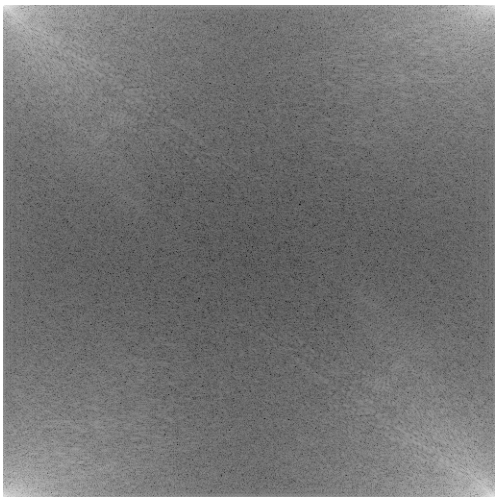

Figure 14: DFT Logged

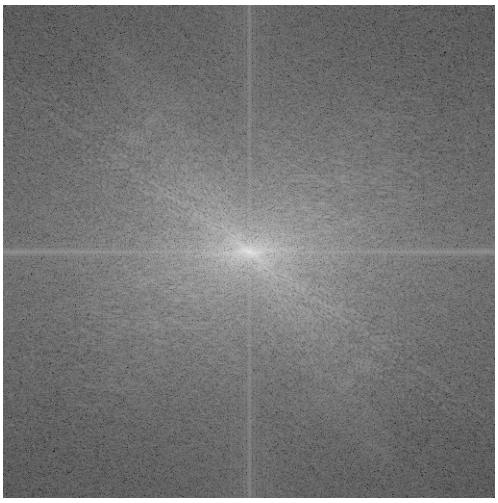

Figure 15: DFT Shifted and Logged

Figure 16 shows Lena reconstructed after taking the inverse Fourier transform of the DFT. The MSE was calculated as described in the method section and resulted in a MSE = 2388. This change from the original to reconstructed could be due to the information lost between transformations. Since the signals were digitally recreated, some of the signals could have ended up being rounded or removed.

Figure 16: IDFT Reconstructed Lena

5 The Magnitude and Phase of the 2-D DFT

Methods: In the final section, the two new images were gathered with the purpose of mixing and matching their magnitudes and phases. It was desired to have well formed images, preferable that of animals, faces, objects etc. With that in mind, the first image gathered was that of a cat as seen in Figure 17. the second image was that of a face, as seen in Figure 18. They were both 512 x 512 pixels and grey scale. Just like in the previous section, the DFT's were taken of both images. However, not only were the magnitudes taken, but the phases as well. Once they were separated, the images were reconstructed . This time however the magnitude of the other image was used in combination with its own phase. This was done to observe the effects of magnitudes and phases within images.

Figure 17: Cat Grey Scale

Figure 18: Face Grey Scale

Results: Figures 19 and 20 show the magnitude and phase plots from the cat photo, while Figures 21 and 22 show the magnitude and phase plots from the face photo. The magnitude spectrum from the cat was multiplied with the phase spectrum from the face. The result of which was ran through an inverse Fourier transform to create "cat face". The results of which can be seen in figure 23. Conversely, the magnitude of the face was multiplied with the phase from the cat. Just like previously, the result was ran through an inverse Fourier transform to create "face cat" as seen in Figure 24. It should be noted that the final results were not as intense as expected. This could be due to the fact that the cat photo has some extra detail in the background. This also could of been an issue with multiplying the spectrums and in-versing them.

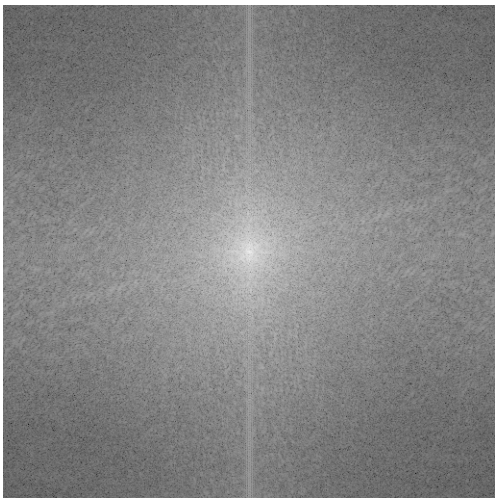

Figure 19: Cat Magnitude Spectrum

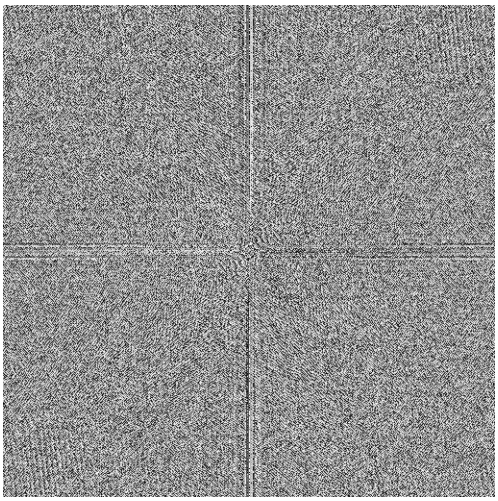

Figure 20: Cat Phase Spectrum

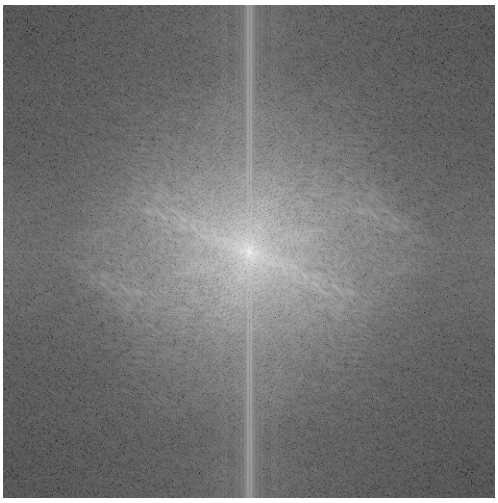

Figure 21: Face Magnitude Spectrum

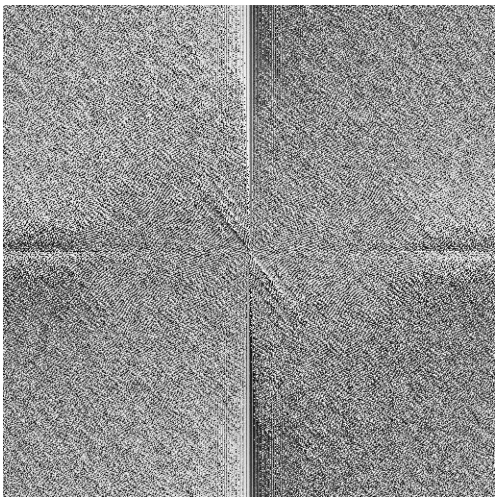

Figure 22: Face Phase Spectrum

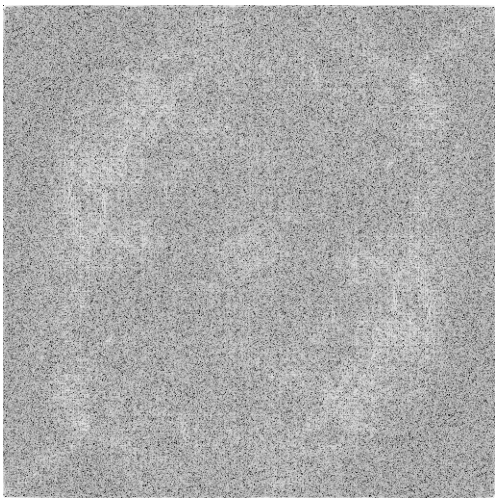

Figure 23: Cat Face

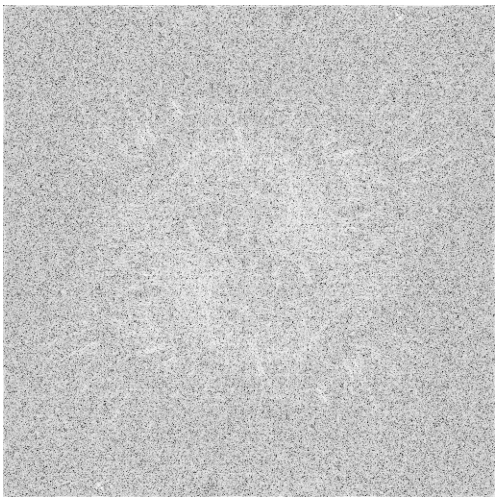

Figure 24: Face Cat

6 Discussion

Overall, the Project was able to achieve what was required and explored many different applications of image manipulations. The project also doubled in helping understand ways of how to implement these features in working code that could input and output the required results. Not only that, but by being able to create these features in a coding environment, it allowed for a more thorough

understanding of the concepts by being able to get detailed information about what each step was doing. Lastly, it should be noted that the code was written and ran using Jupyter Notebook on a local machine.

References

lena greyscale image: https://www.cosy.sbg.ac.at/ pmeerw/Watermarking/lena.html
cat greyscale image: https://www.cs.montana.edu/courses/spring2004/430/lectures/02/lect02.html
face greyscale image https://www.hlevkin.com/06testimages.htm