CSC 391 Computer Vision
Project 1 Report
Tianen (Benjamin) Liu
Feb 6, 2019

- **3. Spatial Filtering**
- 3.1 Implementation

For this part, I create a 27x27 box filter and apply it to `image`.
Output:

Original

With Box Filter

- 3.2.1 Smoothing and Denoising

In this part, I apply Gaussian filters of 3 sizes, 3x3, 9x9, and 27x27, and the outputs are shown below.

Gaussian 3x3

Gaussian 9x9

Gaussian 27x27

By comparing and contrasting the 3 Gaussian filtered images above, we can tell the trade-off between denoising and resolution. The “Gaussian 3x3” image is much clearer in resolution but has much more noise than the other two. The “Gaussian 27x27” has less noise but is lower in resolution.

Then I apply median smoothing filters of the same sizes as above. The output images are shown below. By comparing with the Gaussian filtered images, I find that Gaussian filtered images has a smoother blur.

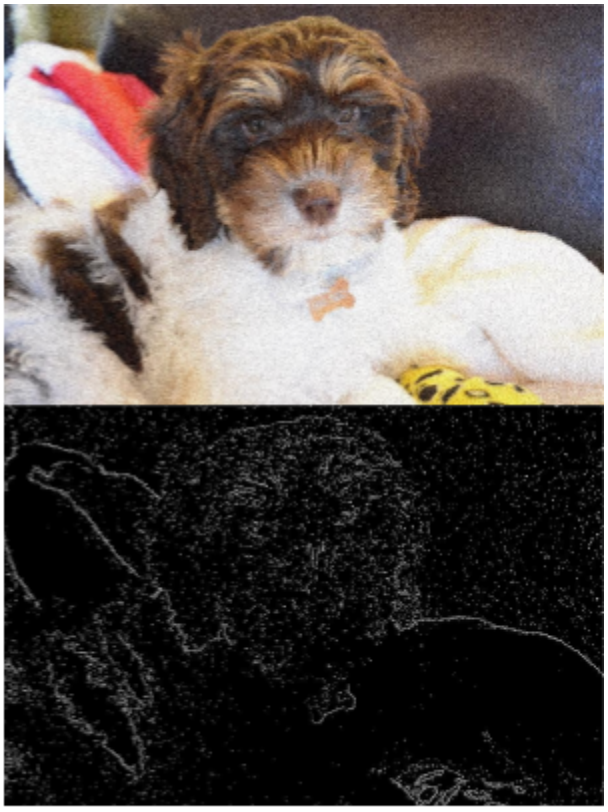
Median 3x3

Median 9x9

Median 27x27

- 3.2.2 Edge Detection

In this part I implement the Canny edge detector to pull out edges of an image. Below shows a comparison of Canny edge detector's effect on original image and noisy image. It is easy to see that the noisy image still contain a lot of noisy dots after pulling out the edge.



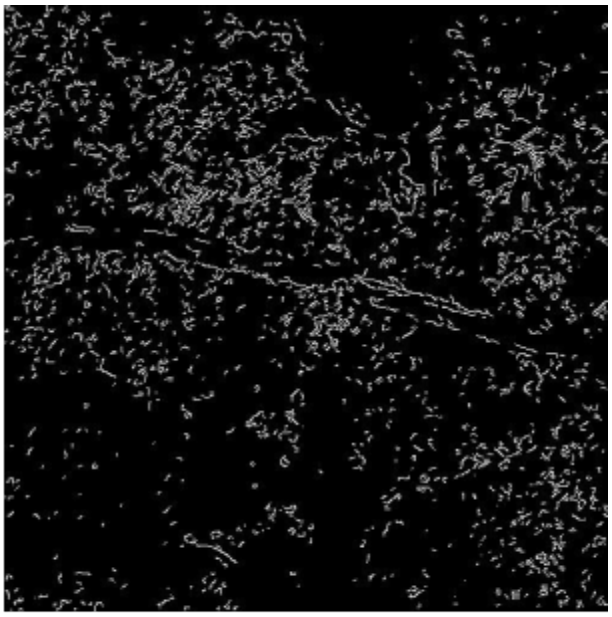
Canny Original

Canny Noisy

While implementing the command `cv2.Canny(image, 100, 200)` on original image, a clear edge shows up. However, using the same command for noisy image will result in many noise. Thus I adjust the frequencies and instead use `cv2.Canny(image, 300, 350)` for noisy image.

Implementing on another image using command `cv2.Canny(image, 150, 200)`

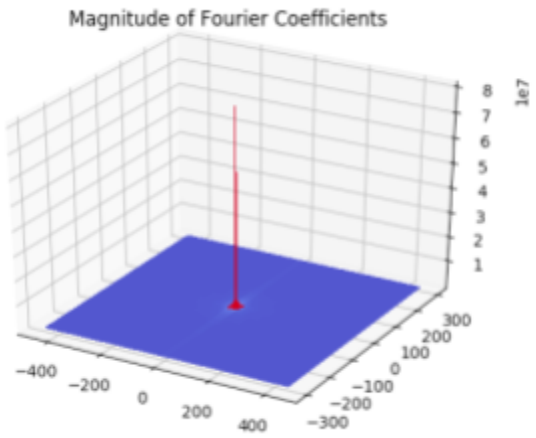
Nature

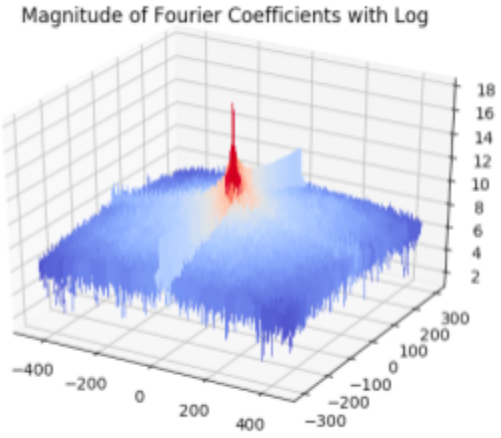


Canny Nature

- **<u>4. Frequency Analysis</u>**
- 4.1 Implementation

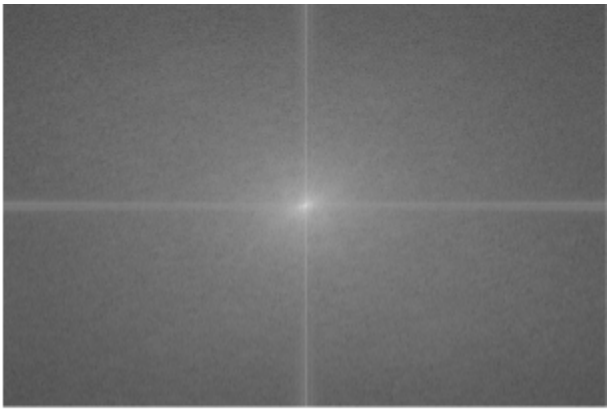
In this part, I first convert `image` file to `image_grey` and then apply 2-D Fourier Transformation on `image_grey`. To visualize the magnitude of each Fourier coefficient, I use a 3-D plot for Fourier coefficients with low frequencies in the center. According to the plot they are much higher in value, which means they contribute more in forming the overall frequency of the image. To further visualize the contribution of mid and high frequencies, I appy log of the magnitude + 1 to shrink the range of those values. The 3-D plot of magnitude of Fourier coefficients is shown below:





Below are the corresponding 2-D magnitude plot viewing from the top:

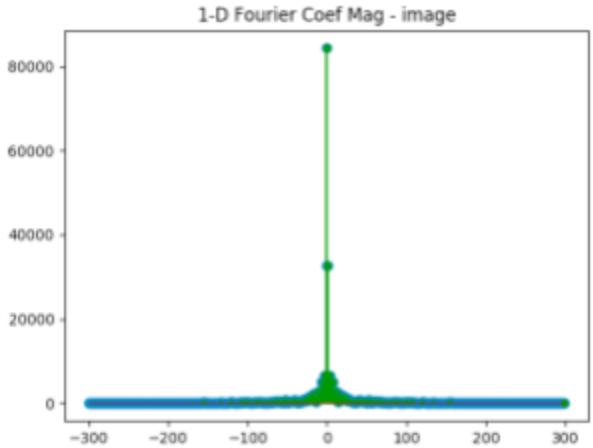
2-D Magnitude

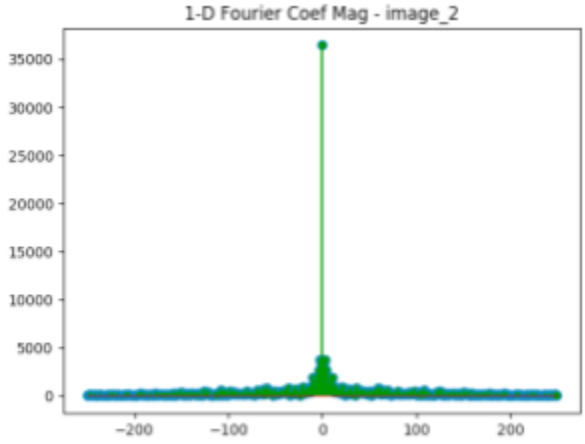


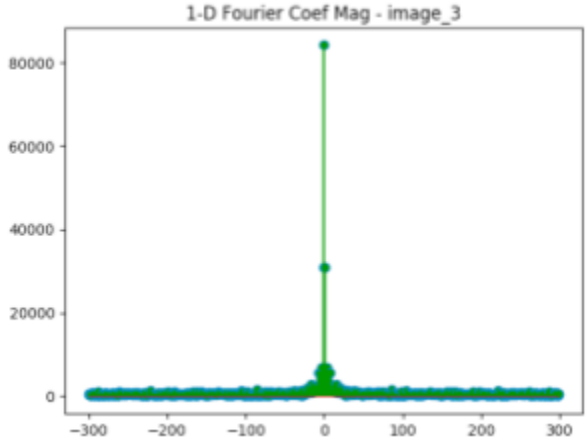
2-D Magnitude w/ log

- 4.2 Frequency Analysis

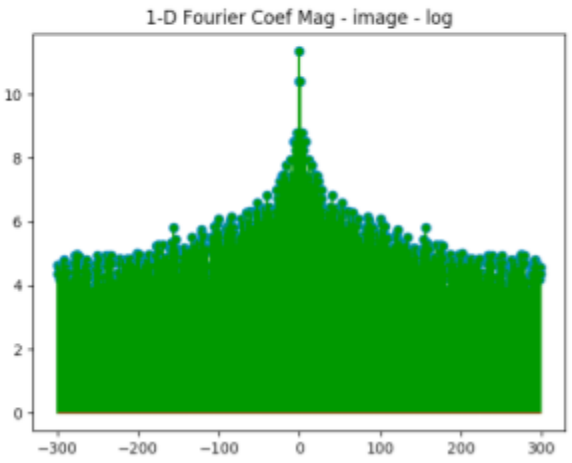
In this part, I first plot the 1-D Fourier coefficients along the x-axis. I apply 1-D Fourier Transformation on original image (`image`), noisy image (`image_3`), and a different image (`image_2`). Below are the 2-D plots showing magnitude of Fourier coefficients.

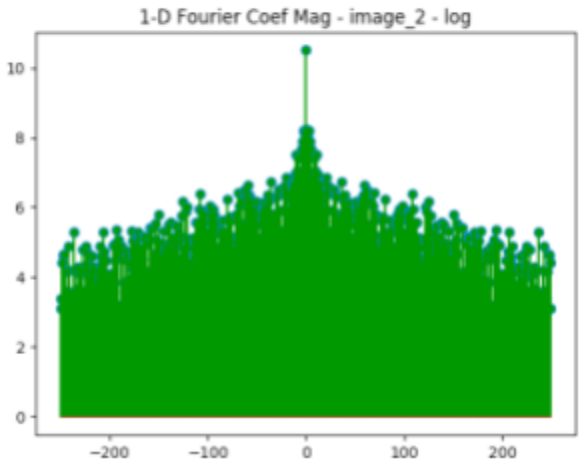




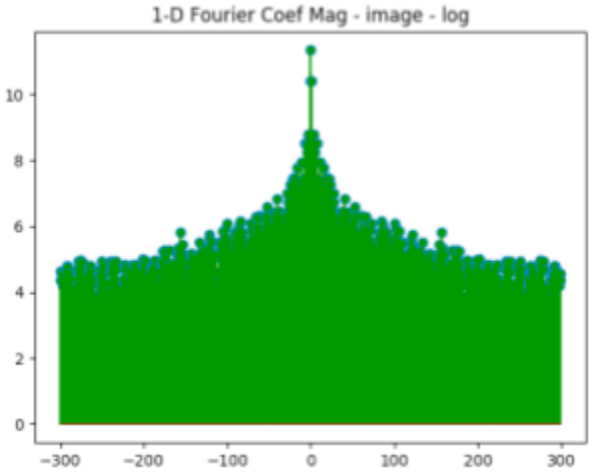


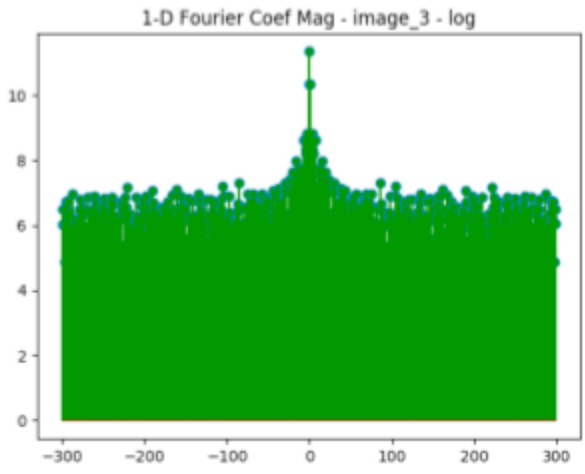
To see if there is any relationship between image content and the decay of Fourier coefficient, compare the plot for `image` and `image_2`. As shown below by the logged Fourier magnitude plots, the 2 plots are different in their rates of decay.



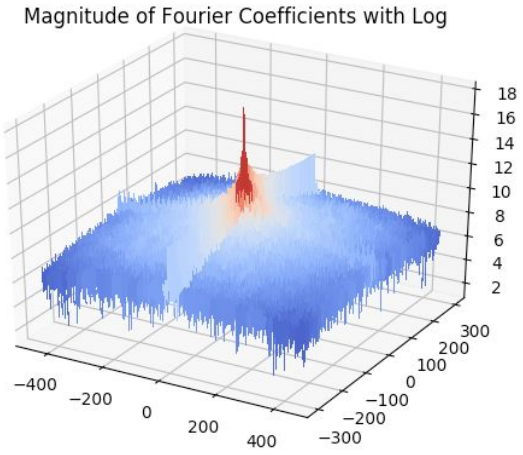


Then I apply log on the magnitudes to better visualize the contribution of mid and high frequencies. Based on the plot below for original image (`image`) and noisy image (`image_3`), the effect of noise on the magnitude of Fourier coefficients is that it gives the magnitude plot a flatter pattern. This is because noisy images has more high frequencies, thus the values are high even for areas away from the center.





Some show higher (brighter) values around the axis, away from the center of image. This is due to the fact that the frequencies on x- and y-axis represent horizontal and vertical frequencies and they have higher contribution to the overall frequencies of the image. This situation is illustrated by plots like this:

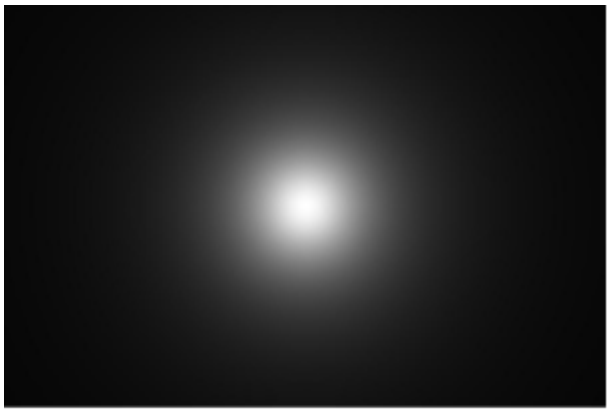


When the area away from the center of the Fourier coefficient is zeroed out, the situation resembles ideal low-pass. When the center of the Fourier coefficient is zeroed out, the situation resembles high-pass. Zeroing out area away from the center is useful for denoising / smoothing

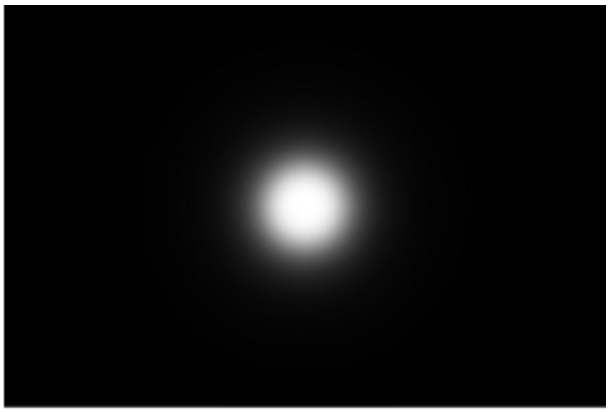
because it zeros out high frequencies. Similarly, zeroing out the center highlights the high frequencies, which is useful for edge detection.

- **5. Frequency Filtering**
- 5.1 Implementation

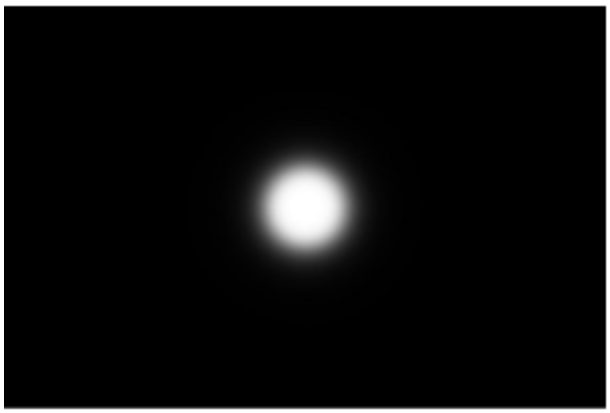
In this part, I use Butterworth approach for filtering. The filters' Fourier coefficients are dampened as the following image shows.



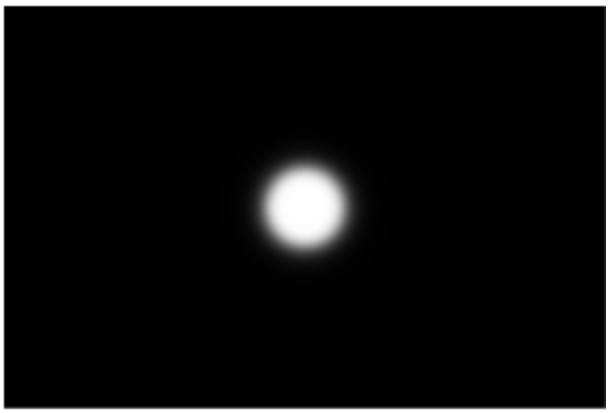
Butterworth n = 1



Butterworth n = 2

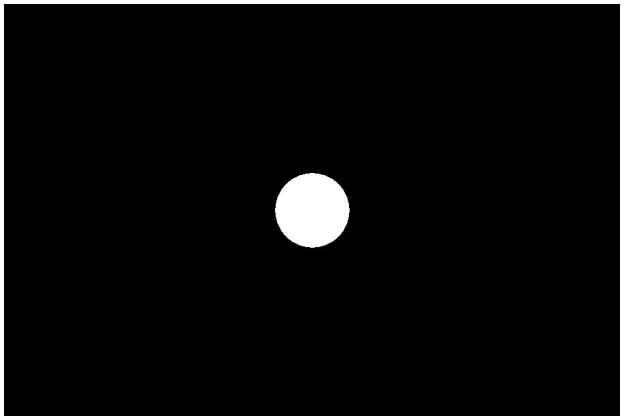


Butterworth n = 3



Butterworth n = 4

The ideal low pass filter are shown below.



The 4 Butterworth filters result in the filtered images below. The cut-off frequency I choose is 0.1 using command `D0 = 0.1 * D.max()`.

Butterworth n = 1

Butterworth n = 2

Butterworth n = 3

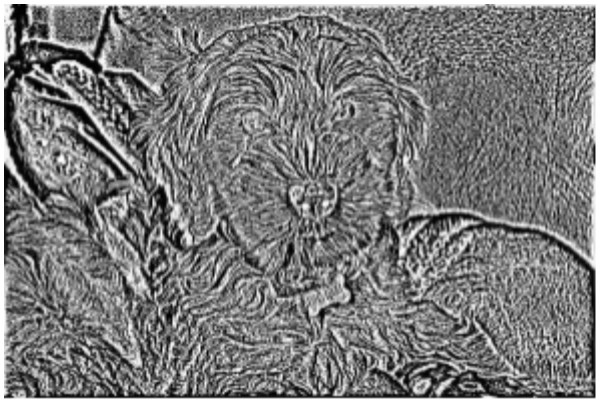
Butterworth n = 4

The ideal low-pass results in the image below

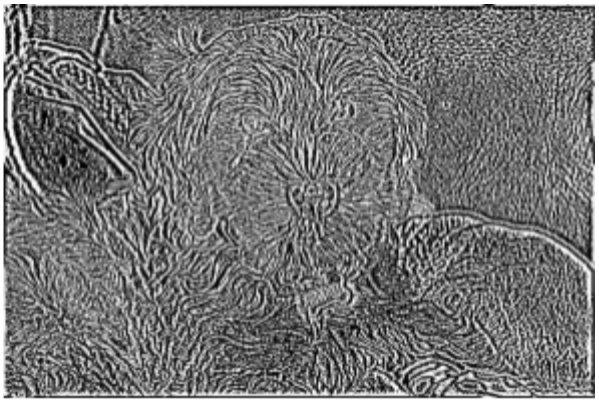
We can tell that for Butterworth filters, the image is more blurry when n is larger. The image even shows ripple patterns for the ideal low-pass image.

- 5.2 Low-pass and High-pass Filtering

I use the original image minus the low-pass images to create the high pass images that highlights the edges of the images. For high-pass images resulting from each of the Butterworth images, the results are shown below.



Butterworth n = 1



Butterworth n = 2

Butterworth n = 3

Butterworth n = 4

The high-pass image resulting from the ideal low-pass image is shown below

We can see that this one resulting from the ideal low-pass shows the clearest edges and least amount of noise among the 5 high-pass images.