Computer Vision Project1 Report
Binfeng Xu (Bill)

Spatial Filtering

First, I applied the Gauss Filter and Median Filter to the noisy puppy image based on 3 different filter size (3, 9 and 27). The results are:

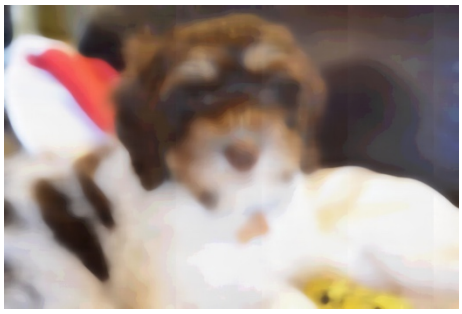
Original Noisy Puppy Image:

Gauss Filter (with size 3, 9, and 27):



Median Filter (with size 3, 9, and 27):





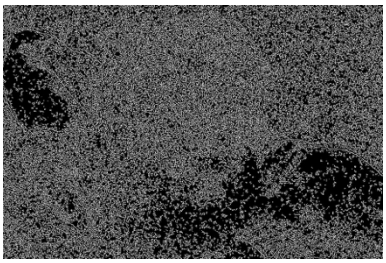
We can notice a tradeoff between denoising and image resolutions. The bigger filter size, the more blurring the image will be after transformation. This effect is extremely obvious on Median Filter, where a lot of details are removed while increasing filter size. The Gauss Filter is

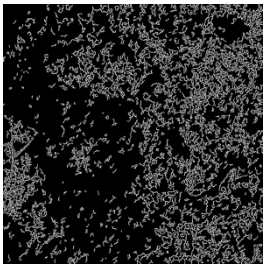
better in terms of keeping details, but the such extent is determined by how large its variance (sigma) is.

Now I use Canny Filter to perform edge detection to the noisy puppy image, the original puppy image and the forest image. The results look like:

Noisy Puppy | Original Puppy | Forest



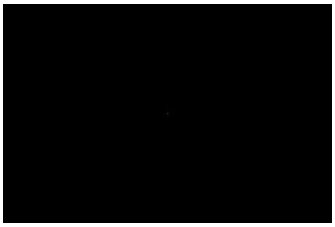
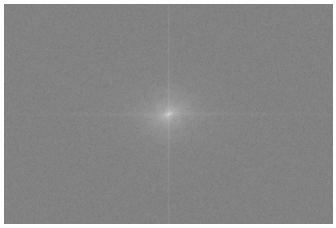


The noisy picture contains a lot of unrelated details and make the edge detection noisy as well. Probably one way to improve the result is denoising using Gauss Filters before edge detection.

Frequency Analysis

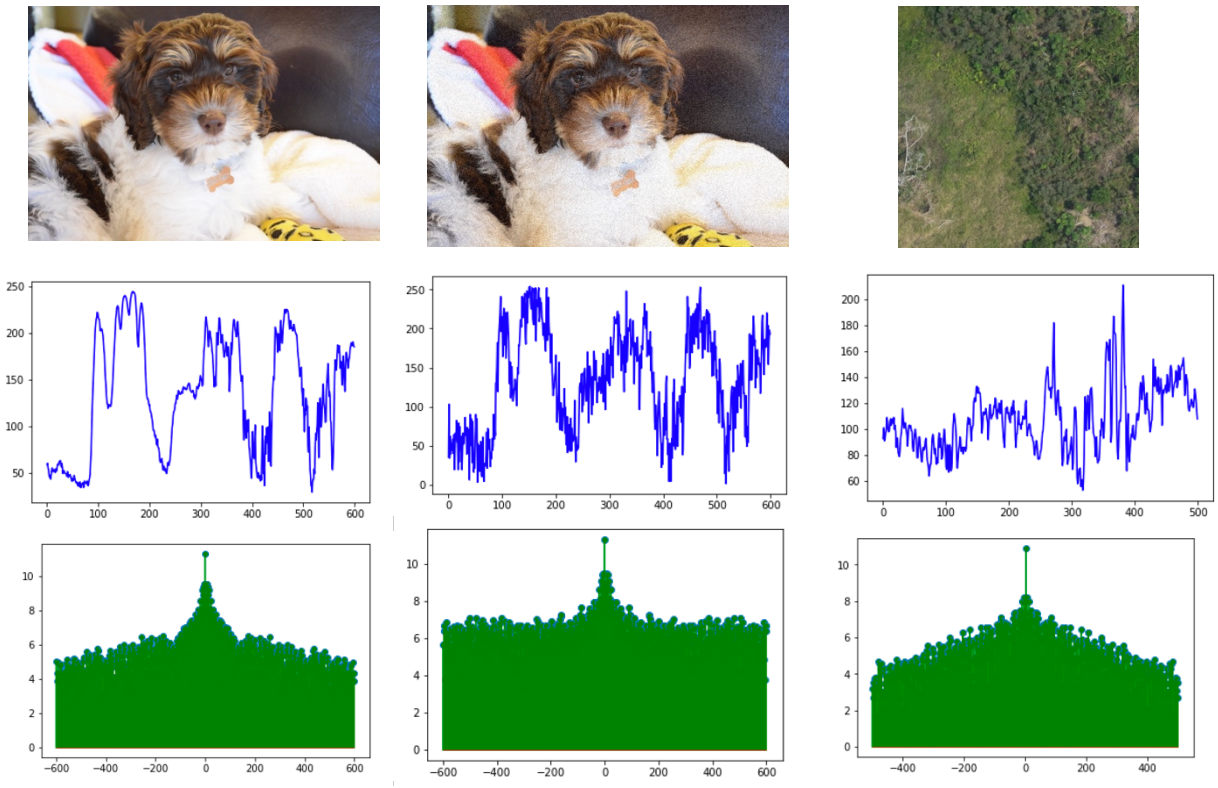
Using 2D Fourier Transformation, we and get a 3D plot of magnitude in frequency domain. The results look like this:

Original Image | Magnitude | Log Magnitude

We can also transform the image using only a column of it, to visualize magnitude decay rate. The first row are original images, the second row are their intensity profile of first column, and the third row are Fourier Transformation magnitude plots.

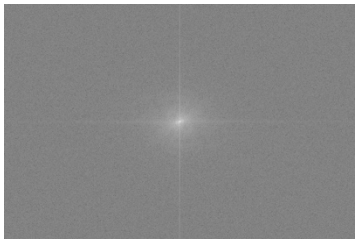
Original Puppy | Noisy Puppy | Forest



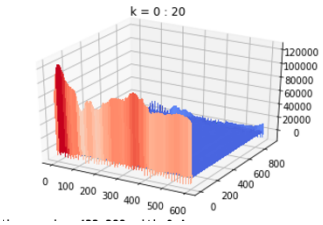
There doesn't seem to be obvious relationship between image contents and magnitude decay rate. Both puppy images decay a little bit slower than the forest, possibly because that forest image is sparser in terms of contents.

For the noise comparison, even though not obvious, the magnitude for noisy puppy image does have more fluctuations than original one. Also, it decays a little bit slower than the original.

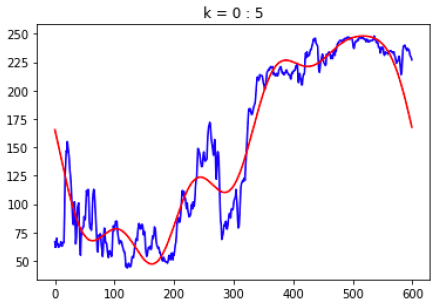
We can notice from the 2D magnitude plot that magnitude around the axes is higher than others. This is because that a small k value in Fourier Transform function tends to result in higher magnitude.

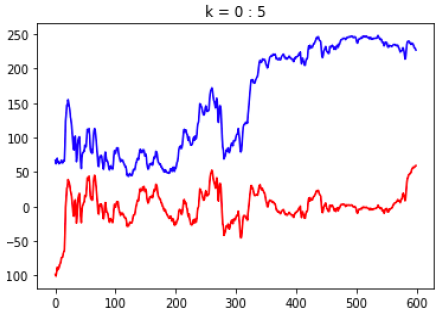


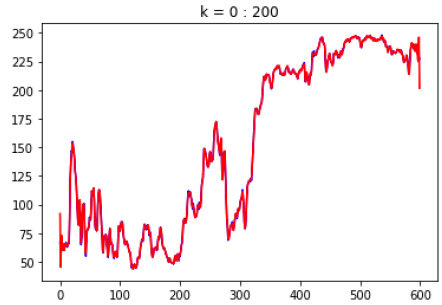
The inverse Fourier Transform effect looks like this:
Since it is too hard to interpret, we will look at the 1D effect on only 1 column.

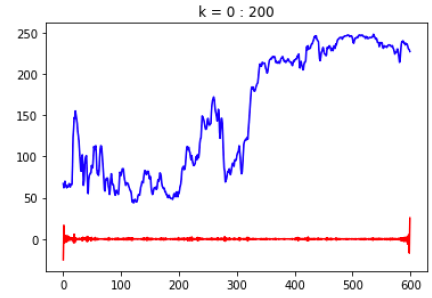


1D inverse DFT:







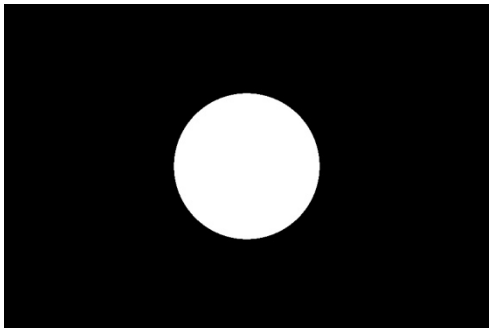


The left columns shows the results of zeroing out centers (low frequency) of frequency domain. The right columns shows the results of zeroing out margins(high frequency). As k increases, the former method fit the origin frequencies better, the later method diverges more and more from the original frequency. So when smoothing the image, we should zeroing out low frequencies from frequency domain.

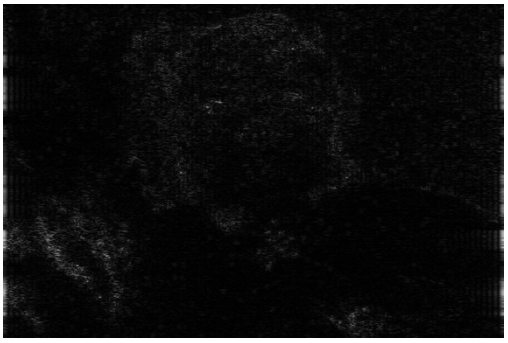
Frequency Filtering

Now we are using both Lowpass and Highpass filters on both original approach and Butterworh's approach.

The lowpass and highpass filters are visualized as:



Their effect on the different approaches looks like:

	Lowpass	Highpass
Zero-Out		
Butte-rworth		

For original zeroing out approach, highpass filter removes a lot of details while only keeping high frequency elements such as edges. However, such effect does not seem obvious in Butterworth's approach.
Using the same filter, we can see that Butterworh's approach seems to have better smoothing results in terms of that it contains more details.