Using Spectral Filtering Techniques on Ultrasound Images to Locate a Marble Swallowed by a Dog

Ashley Batchelor

Abstract

Ultrasound imaging was used to locate a marble swallowed by a dog. I analyzed a series of 20 three dimensional images of the dog's digestive system with 64x64x64 voxels. In order to remove noise, I first averaged all 20 images in the frequency domain to locate a peak frequency profile, and then spectrally filtered each of the images in the frequency domain, using a gaussian filter around the peak frequency profile. I was able to locate the marble as a peak within each of the 20 filtered images.

Sec. I. Introduction and Overview

A very useful technique for images which contain a high level of noise is to attenuate the noise through filtering in the frequency domain, e.g. with bandpass filtering. In ultrasound imaging applications various fluids may produce unwanted noise which may make it difficult to locate a desired object in an image. With a sufficient number of images, it is possible to use averaging of Fourier transformed images to identify a frequency profile corresponding to the desired object. In a single image, this frequency profile may have an energy value which is of comparable magnitude to noise. However, by averaging over many images, noise is averaged out and attenuated, and a distinct peak emerges which is indicative of the object's frequency profile.

Sec. II. Theoretical Background

In one dimension, an example of a signal f(x) may be represented by a Gaussian curve centered about x_0 with a variance σ which represents a desired signal which is coupled with a white noise component:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-x_0)^2/2\sigma^2} + noise \qquad \text{Eq.1.}$$

Alternatively, f(x) may be represented by a sech function or other similar curve. A measurement of this signal may include noise with an amplitude which is similar to the Gaussian curve.

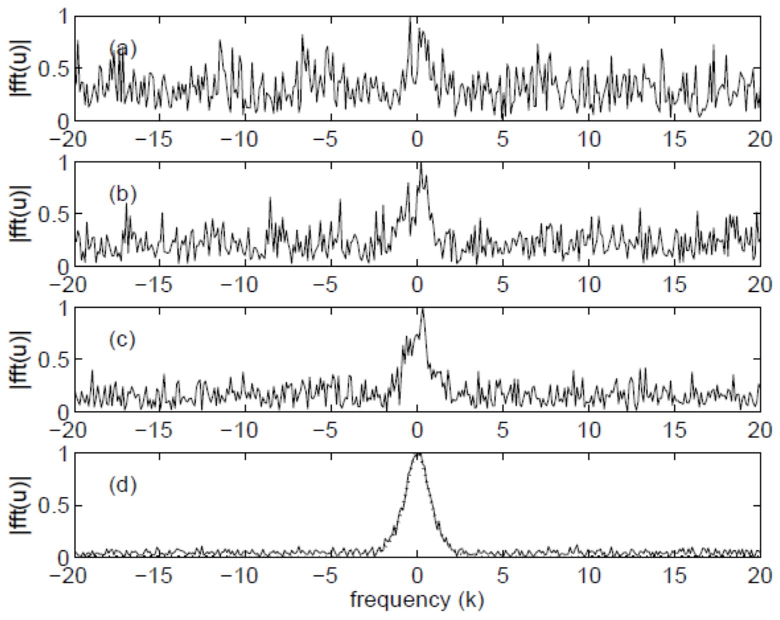

Figure 1: FFT content of averaged signals containing random noise and a spectral peak about k=0, for a single signal, 2 signals, 5 signals, and 100 signals. Kutz, J. Nathan. Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data, Oxford University Press, 2013: page 318.

Such a measurement may be represented in the frequency domain as a Fourier transform:

$$F(x) = \Im(f(x)) \qquad \text{Eq.2.}$$

In the presence of noise, a single measurement may be indistinguishable from noise in both the time domain and the frequency domain. Averaging multiple signals with a Gaussian (or similar) peak coupled white noise yields a composite signal with a discernible peak in the frequency domain, as shown in the example of Fig.1. Averaging reduces the amplitude of the noise and increases the amplitude of the signal peak.

A bandpass filter may be applied to noisy signals based on the location k_0 of the signal peak. A very simple bandpass filter is a Gaussian curve in the frequency domain:

$$G(k) = \exp(-\tau(k - k_0)^2) \qquad \text{Eq.3.}$$

where the Gaussian curve is centered about k_0 with a bandwidth τ. The filter may be applied as a product

$$H(k) = F(k)G(k) \qquad \text{Eq.4.}$$

Taking the inverse Fourier transform gives a filtered noise-attenuated signal in the spatial domain (or time domain, in the case of applications using time-based signals).

$$h(x) = \Im^{-1}(H(k)) \qquad \text{Eq.5.}$$

A sufficiently accurate frequency profile of a desired signal may be determined from averaged frequency domain measurements of noisy signals, since the noise will average to low values, while a well-behaved signal will increase. Thus, by determining the location of a frequency peak corresponding to a desired signal, a location k_0 to center a bandpass filter $G(k)$ may be determined.

The preceding methods may be broadened to multiple dimensions. For three dimensions, a filtered signal $h(x, y, z)$ may be determined by averaging instances of three dimensional measurements $f(x, y, z)$ in the frequency domain to determine a center (k_{x0}, k_{y0}, k_{z0}) for a bandpass filter $G(k_x, k_y, k_z)$. As one example, this filter may be a three-dimensional Gaussian:

$$G\left(k_x, k_y, k_z\right) = \exp(-\tau((k_x - k_{x0})^2 + \left(k_y - k_{y0}\right)^2 + (k_z - k_{z0})^2) \qquad \text{Eq.6.}$$

Alternatively, a variety of window functions centered about $\left(k_{x0}, k_{y0}, k_{z0}\right)$ may also be suitable for bandpass filtering which is similar to a Gaussian filter.

Sec. III. Algorithm Implementation and Development

In this application, the initial data was a MATLAB file in the form of a two-dimensional array of 20x262,144 complex values. I reshaped the array into 20 instances of three-dimensional images with a resolution of 64x64x64 voxels. The images were taken over a sequence of time in a cubical volume with a 30 mm width spanning L=-15 mm to L=15 mm in the x, y, and z directions. The images I analyzed of the dog's digestive system contained a high amount of noise from moving fluids. I performed a three-dimensional Fast Fourier Transform (FFT) on each image and summed and scaled those 20 transformed images. I used 64 Fourier modes in my calculations. I scaled the wavenumbers by a factor of $2\pi/L$ for the FFT calculations. I located a maximum value of the peak of the data in this averaged frequency domain image, which provided a frequency profile of the marble. It should be noted that a more robust centroid calculation of this peak may improve accuracy. For example, such a centroid may be calculated from mean in the neighborhood of the peak, a Gaussian fit, or a Gaussian fit with higher order standardized moments (e.g. skewness, kurtosis, etc.). For simplicity, I used the indices of a maximum value in the averaged frequency domain image to determine the frequency profile of the marble. Next, I spectrally filtered each of the 20 transformed images with a Gaussian filter centered about the maximum value from the averaged frequency domain image and then applied an inverse FFT to provide a set of 20 noise-attenuated images.

From the 20 noise-attenuated images, I located a maximum value of the image and from its indices within the image, I determined a position of the marble within that image. I then plotted the trajectory of the marble through the course of the 20 images and determined a position within the 20th image for where an acoustic wave could be focused to break up the marble.

Sec. IV. Computational Results

Fig.2. shows an example of a noisy initial image, displayed as a slice in the z =-6.093 plane which lacks a distinct peak corresponding to the marble. My averaged Fourier transformed image indicated a peak in frequency space corresponding to [Kx,Ky,Kz] = [1.8850,-1.4072,0.000]. Fig.3. shows a slice in the Z=0.000 plane showing a color mapped peak around Kx =1.8850, Ky = -1.4072. Fig.4. shows a Gaussian filter (according to Eq.3.) which I applied to the noisy images (according to Eq.4.) about the characteristic frequency of wavenumber [Kx,Ky,Kz] = [1.8850,-1.4072,0.000], with a bandwidth $\tau = 0.2$. Fig.5. shows the 20th noise-attenuated image (according to Eq.5.) indicating a position of the marble. Fig.6. shows a plot of the trajectory of the marble in a coordinate system defined by the field of view of the ultrasound sensor. I determined that an acoustic wave should be focused to break up the marble at a position [x,y,z] = [-5.625,4.219,-6.093] in the 20th image.

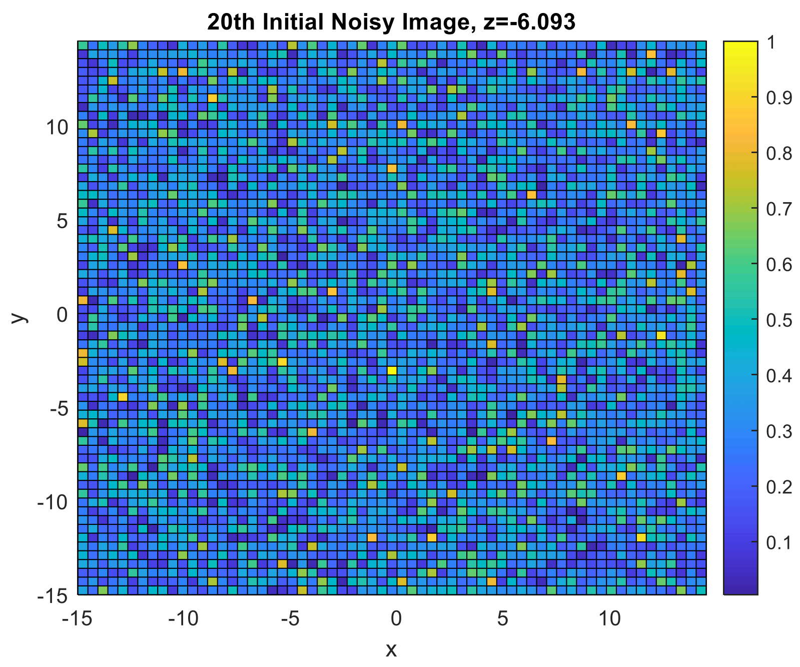

Figure 2: A noisy image with no visible indication of the marble.

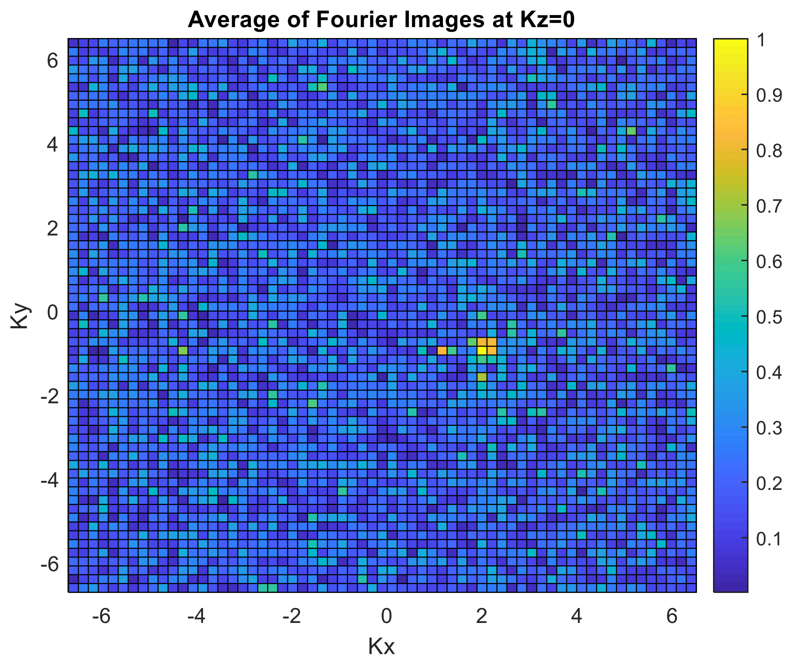

Figure 3: Composite frequency domain image showing a peak frequency profile.

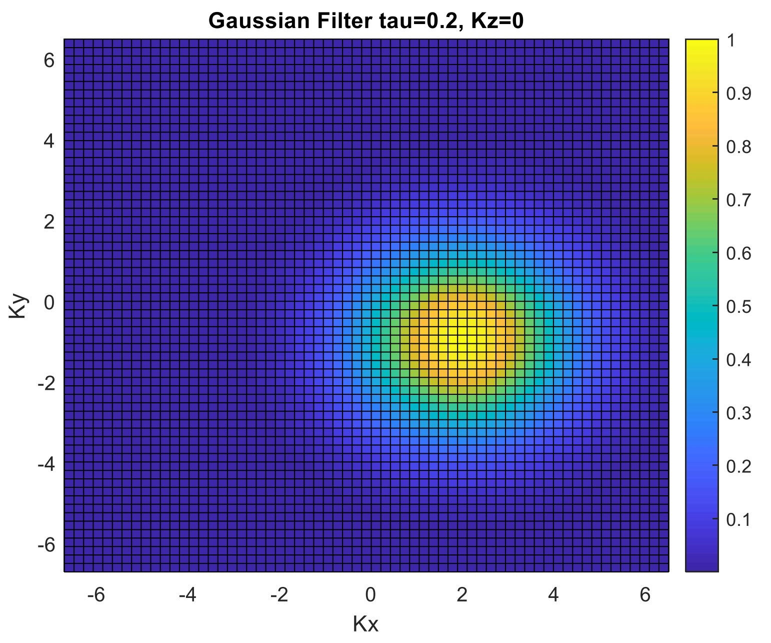

Figure 4: Gaussian filter centered on the frequency wavenumbers of the peak of Figure 2.

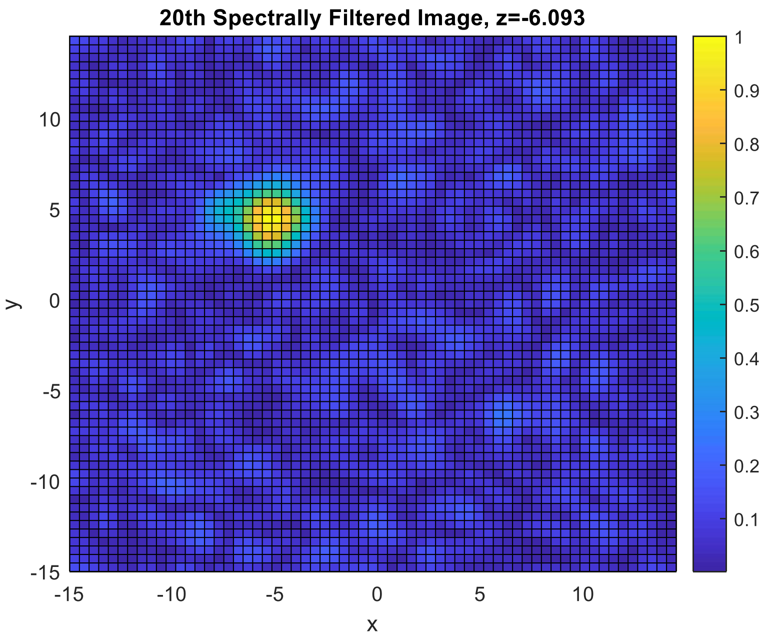

Figure 5: Noise-attenuated image from the data of Figure 2, visibly showing the marble.

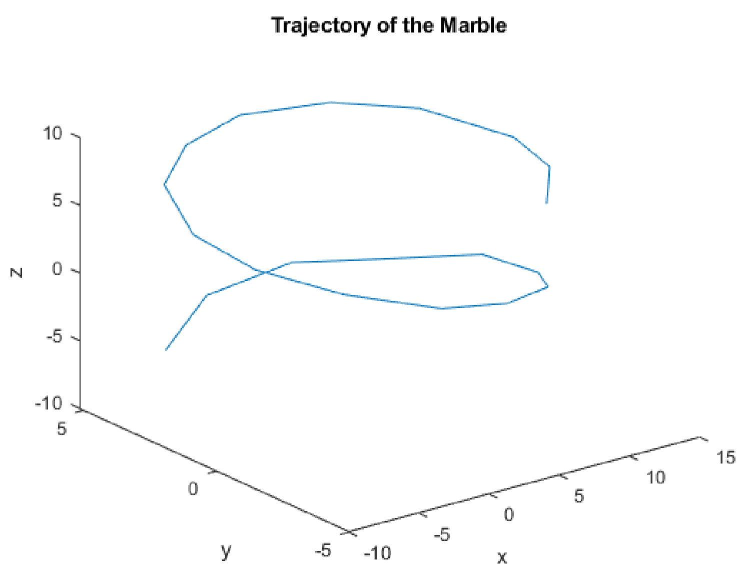

Figure 6: Trajectory of the marble through the 20 ultrasound images.

Sec. V. Summary and Conclusions

Bandpass filtering in the frequency domain is an effective way to attenuate noise within an image or within other types of signals (e.g. radar), which improves desired signal characteristics which may not otherwise be discernable in a noisy image. Using such techniques, I was able to locate and track a marble in a series of noisy ultrasound images in which the marble was not directly visible.

Appendix A MATLAB functions used and brief implementation explanation

fftn() – This is the N-Dimensional Fast Fourier Transform.

fftshift() – This shifts the FFT results such that the zero frequency components are at the center of the array.

ifftn() – This is the inverse N-Dimensional Fourier transform.

max() – This returns the maximum value within an image as Ill as its indices.

ind2sub() – This converts the index value returned by max() from a single number into a vector of indices.

plot3() – This returns a 3-dimensional plot, which I used for plotting the trajectory of the marble.

squeeze() – This was useful for taking an instance of a 3-D image out of an array of such images and giving it the appropriate dimensions for a Fourier transform.

Appendix B MATLAB codes

```
clear all; close all; clc;
load Testdata
L=15; % spatial domain
n=64; % Fourier modes
x2=linspace(-L,L,n+1); x=x2(1:n); y=x; z=x;
k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1]; ks=fftshift(k);

[X,Y,Z]=meshgrid(x,y,z);
[Kx,Ky,Kz]=meshgrid(ks,ks,ks);

UnAve = zeros(n,n,n);

%Take a Fourier transform of each image and average them in the Fourier
%Domain, store the Fourier transformed images and the spatial domain images
%in arrays for later
for j=1:20
  Un(:,:,:)=reshape(Undata(j,:),n,n,n);
  Uninstance(j,:,:,:)=Un(:,:,:);
  Ut=fftshift(fftn(Un));
  Utinstance(j,:,:,:)=Ut(:,:,:);
  UnAve = UnAve + Ut;
end
UnAveAbs = abs(UnAve);
UnAveAbs = UnAveAbs / max(UnAveAbs(:));

%Find the max of the averaged spectra
[C,I] = max(UnAveAbs(:));
[I1,I2,I3] = ind2sub(size(UnAveAbs),I);

%Create a filter around the max value of the averaged spectrum
KxFilt = ks(I2);
KyFilt = ks(I1);
KzFilt = ks(I3);
filter = exp(-0.2*((Kx-KxFilt).^2+(Ky-KyFilt).^2+(Kz-KzFilt).^2));
```

```
%Filter the images in the Fourier domain, and take the inverse transform
for j = 1:20
    UtFilt(j,:,:,:)=filter.*(squeeze(Utinstance(j,:,:,:)));
    UnFilt(j,:,:,:)=ifftn(UtFilt(j,:,:,:));
end


%Calculate a position for each image and plot its motion through the image
%sequence
GraphData = zeros(3,20)
for j = 1:20
    UnFiltInstance = abs(squeeze(UnFilt(j,:,:,:)));
    [C,In]=max(UnFiltInstance(:));
    [In1,In2,In3] = ind2sub(size(UnFiltInstance),In);
    xLoc = x(In2);
    yLoc = y(In1);
    zLoc = z(In3);
    GraphData(1,j)=xLoc;
    GraphData(2,j)=yLoc;
    GraphData(3,j)=zLoc;
end

%Plot a slice of the 20th unfiltered image
UnSlice = squeeze(UnFiltInstance(:,:,In3));
pcolor(x,y,UnSlice);
xlabel('x')
ylabel('y')
title('20th Initial Noisy Image, z=-6.093');

%Plot a slice of the Fourier averaged image
UnAveAbsSlice = squeeze(UnAveAbs(:,:,I3));
pcolor(ks,ks,UnAveAbsSlice);
xlabel('Kx')
ylabel('Ky')
title('Average of Fourier Images at Kz=0');

%Plot a slice of the Gaussian filter
filtSlice = squeeze(filter(:,:,I3));
pcolor(ks,ks,filtSlice);
xlabel('x')
ylabel('y')
title('Gaussian Filter tau=0.2, Kz=0');

%Plot a slice of the 20th filtered image
UnFiltSlice = squeeze(UnFiltInstance(:,:,In3));
pcolor(x,y,UnFiltSlice);
title('20th Spatially Filtered Image, z=-6.093');

%Plot the marble trajectory
plot3(GraphData(1,:),GraphData(2,:),GraphData(3,:));
xlabel('x')
ylabel('y')
title('Trajectory of the Marble');
```