Filtering intestinal ultrasound data to isolate the location of a foreign body *in vivo*

Arman Naderi

Introduction and Overview:

Modern medical imaging systems depend on the ability to extract information of interest, such as the size and location of a foreign body, from information collected incidentally, such as noise. Some imaging systems can be designed to inherently filter some of this ineffectual information. For example, ultrasound systems can be designed with a specific transducer shape and/or material that only allows certain frequencies of energy to be sent and received. However, this design feature can in and of itself be detrimental if the information of interest is also filtered out. A more flexible method to extract information is via image post-processing algorithms. Through this approach, a data set can be manipulated through a variety of filters and algorithms to effectively extract useful information. In this paper, I demonstrate the utility of a Gaussian filter applied to ultrasound data to isolate the location of a marble within the intestines of a dog.

A dog presented to the emergency veterinary hospital after swallowing a marble. After initially palpating the animal's gut, the attending veterinarian suspected that the marble had entered a small area of the intestines. Using ultrasound imaging, data was collected at the location of interest. However, due to the restless nature of the animal, internal fluid movement in the intestines generated highly noisy data. The veterinarian suggested that the malady could be treated if the marble was ablated through the use of high frequency acoustic wave. To prevent the high frequency acoustic wave from being sent to the wrong location, I first identified the center frequency signature generated by the marble, then utilized this signature to isolate the trajectory of the marble over time, and finally isolate the location of the marble at the latest timepoint. To understand the mechanism by which I accomplished this, a theoretical understanding of spatial and frequency domain analysis is required.

Theoretical Background:

Ultrasound imaging relies on the ability to emit and receive sound waves. After a sound wave is emitted at a known frequency, the frequency of the sound wave that returns after reflecting off of objects *in vivo* can generate an image of the space occupied by the area being probed. Fundamental to this technique is signal analysis in both the spatial and frequency domains. In this case, I have received spatial data from the dog's ultrasound. In order to eliminate much of the noise from the spatial domain of the data, I will have to convert, and manipulate, the data in the frequency domain. The Fourier Transform (FT) (1.a) and is used to convert a signal in the spatial domain into the frequency domain; the Inverse Fourier Transform (IFT)(1.b) is used to convert a signal from the frequency domain back into the spatial domain.

$$F(k) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-ikx} f(x)dx \qquad (1.a)$$

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-ikx} F(k)dk \qquad (1.b)$$

The utility of this specific transform is limited in that it is only applicable for signals that are continuous for all real numbers x. In digital signal analysis, where datasets are discrete, the

Direct Fourier Transform (DFT) (2.a) can be used to represents signals as sums of sines and cosines.

$$\hat{x}_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn} \text{ and } x_n = \frac{1}{N}\sum_{k=0}^{N-1} \hat{x}_k \cdot e^{\frac{i2\pi}{N}kn} \qquad (2.a)$$

However, the DFT is outclassed in modern computational digital signal analysis, by the Fast Fourier Transform (FFT) because of its efficient computational complexity: O(N log N) instead of O(N2). By converting the spatial domain signal into the frequency domain, I can isolate the frequency signature of the marble by averaging out the noise across the timepoints by which the ultrasound was collected. Then utilizing this frequency signature as a basis for filtering even more noise, I will implement a Gaussian filter to isolate the location of the marble in spatial domain for each timepoint.

The Gaussian filter I will utilize is a spectral filter, meaning that it scales out unwanted frequencies in the frequency domain. The filter, as shown in 3.a, operates simply as a scalar function by which the frequency signal is multiplied by.

$$F(k) = e^{-\tau((k_x-k_{x_0})^2+(k_y-k_{y_0})^2+(k_z-k_{z_0})^2)}$$

Tau describes the width of the filter around $k_{x_0}, k_{y_0}$and k_{z_0}, which are the frequencies of interest in the x, y, and z axes.

Algorithm Implementation and Development:

The first goal was to identify the center frequency of the marble. To do this, the algorithm I developed averages the frequency domain signal of the data set across the twenty different timepoints at which data was collected. Assuming that all of the noise collected by the ultrasound is white noise - that is noise that is equal in intensity at random frequencies – then by averaging the data, the algorithm I developed can effectively isolate the marble's frequency signature. The algorithm then converts this average frequency domain signal back into the time domain to identify at which index in each principal axes the maximum frequency occurs. The algorithm then associates the index of the frequency signature of the marble to its corresponding frequency domain coordinates. These coordinates are the center frequency. The center frequency is then applied to the Gaussian filter which is used to scale the frequency domain signal from every timepoint. With this center frequency, the Gaussian filter removes any frequency domain noise not associated with the marble.

Computational Results:

The computational results from the algorithm identifies the center frequency which is 1.8550, -1.0472, and 0 for the x-, y-, and z-directions, respectively. The trajectory by which the marble travels is shown in Figure 1. The isosurface plot, which identifies the three-dimensional surfaces at which the filtered frequency signals equals a value of 0.4, is shown in Figure 2. The final position of the marble is -5.6250 4.2188 and -6.0938, in the x, y and z directions.

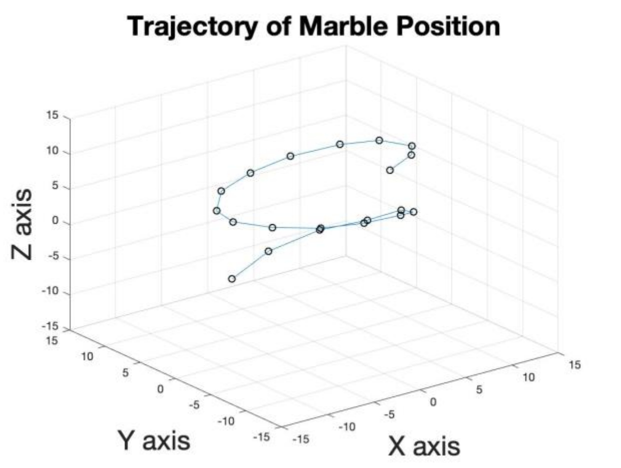

Figure 1: The trajectory of the marble's position, as plotted by the location of the maximum frequency of the marble after filtering for noise with a Gaussian filter at each time point.

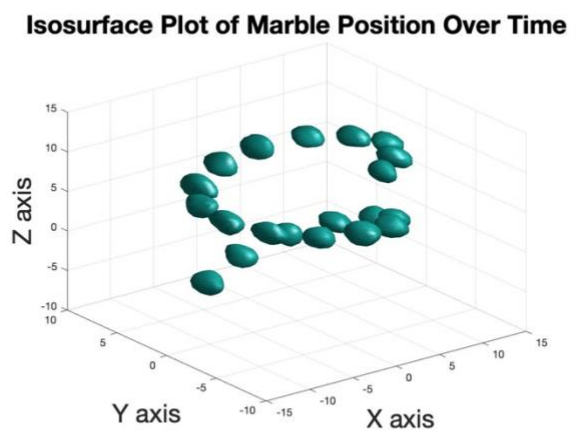

Figure 2: The location and size of the marble as plotted by the location of the maximum frequency of the marble after filtering for noise with a Gaussian filter at each time point, and using an isosurface plot value of 0.4.

Appendix A:
Incomplete

Appendix B:

```
clear all; close all; clc;
load Testdata

% spatial domain
L = 15;
% Fourier modes
n = 64;
%number of time points in Undata
timepoints = size(Undata,1);
x2 = linspace(-L,L,n+1); x = x2(1:n); y = x; z = x;
%freq components for fft scaled to 2 pi
k = (2*pi/(2*L)) * [0:(n/2-1) -n/2:-1];
%shifted freq components to be mathematically correct
ks = fftshift(k);

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

% Go to the frequency domain with fftn, average across time points
avefftUn=zeros(64,64,64);

for j=1:timepoints
    Un(:, :, :) = reshape(Undata(j,:),n,n,n);
    avefftUn(:, :, :) = avefftUn(:, :, :) + fftn(Un);
end
```

```
% Average of the frequency domain of Un, now shifted to be mathematically
% correct
avefftUn=fftshift(avefftUn)./timepoints;

% Go back to the spatial domain of Un, to locate frequency space index
aveUn=abs(ifftn(avefftUn));

[maxfreq, maxfreqindex] = max(abs(avefftUn(:)));
[maxKxindex, maxKyindex, maxKzindex] = ind2sub(size(avefftUn), maxfreqindex);

%Identifying the center frequency for each principal axis
Kx0 = Kx(maxKxindex, maxKyindex, maxKzindex);
Ky0 = Ky(maxKxindex, maxKyindex, maxKzindex);
Kz0 = Kz(maxKxindex, maxKyindex, maxKzindex);

%Filter data using a Gaussian filter
tau = 0.2;
gausfilter = exp(-tau*(((Kx-Kx0).^2)+((Ky-Ky0).^2)+((Kz-Kz0).^2)));

%Creating matrices to store the location
x_pos = zeros(1,timepoints);
y_pos = zeros(1,timepoints);
z_pos = zeros(1,timepoints);

%Applying the filter to each timepoint in the frequency domain, must
%reshape the data again and shift it at each timepoint

for j=1:timepoints
    Un(:, :, :) = reshape(Undata(j,:), n, n, n);
    fftUnfilter = gausfilter.*fftshift(fftn(Un));
    %filtered frequency in spatial domain
    Unfilter = abs(ifftn(fftUnfilter));

    [maximum, index] = max(Unfilter(:));
    [filterKxindex, filterKyindex, filterKzindex] =
ind2sub(size(fftUnfilter), index);

    x_pos(1,j) = X(filterKxindex, filterKyindex, filterKzindex);
    y_pos(1,j) = Y(filterKxindex, filterKyindex, filterKzindex);
    z_pos(1,j) = Z(filterKxindex, filterKyindex, filterKzindex);

    isosurface(X,Y,Z, Unfilter,0.4), grid on
    hold on
    title('Isosurface Plot of Marble Position Over Time', 'Fontsize', 24)
    xlabel('X axis', 'Fontsize', 24)
    ylabel('Y axis', 'Fontsize', 24)
    zlabel('Z axis', 'Fontsize', 24)
end

figure()
plot3(x_pos, y_pos, z_pos, 'ko')
hold on
plot3(x_pos, y_pos, z_pos)
```

```
title('Trajectory of Marble Position', 'Fontsize', 24)
xlabel('X axis', 'Fontsize', 24)
ylabel('Y axis', 'Fontsize', 24)
zlabel('Z axis', 'Fontsize', 24)
axis([-L L -L L -L L]), grid on, drawnow

figure()
isosurface(X, Y, Z, Unfilter, 0.4)
title('Marble Position at 20th Timepoint', 'Fontsize', 24)
xlabel('X axis', 'Fontsize', 24)
ylabel('Y axis', 'Fontsize', 24)
zlabel('Z axis', 'Fontsize', 24)
axis([-L L -L L -L L]), grid on, drawnow

breakup_pos = [x_pos(1, j), y_pos(1, j), z_pos(1, j)]
```