Tracking a Marble in a Dog's Stomach

Aidan Sleavin

The problem that is being dealt with is a dog has eaten a marble that needs to be located in its intestines. Ultrasound data of the innards of the dog have been taken but the data is too noisy and difficult to read because of the dog's internal fluids. This data must be made clear in order to locate the marble in the dog's digestive tract so it may be destroyed.

Overview

In the process of taking the ultrasound data of the dog, the data from the dog's intestines were recorded at twenty different points in time. For each of these points the ultrasound had the resolution to take data of points in a 64x64x64 grid in physical space around the area of interest. With this data the frequency off the marble from the ultrasound device can be detected and the marble tracked. The problem with the data is because of the internal fluids of the dog are constantly moving the data has too much noise to immediately pick out the frequency and thus the location of the marble. This data can be made less noisy by using Fourier transforms to look at the data from the perspective of the frequency it creates and averaging over all noisy data to find the marbles frequency. Once the frequency is found a filter in put onto all 20 data points in time in the frequency domain in order to exaggerate the marbles center frequency. Once this is done the inverse Fourier transform is implemented which will result is a clear marble in the spatial domain. The program Matlab is being used to do all analysis and calculation of the data taken by the ultrasound.

Theory

In order to detect the frequency of the marble a Fourier transform must be used. Fourier transforms use sine and cosine, which are periodic function, to transform a plot into frequency space, the specific transform used is known as the Fast Fourier transform and will be referred to at (FFT). The FFT will take the data from a point in time and transform it where the XYZ location is a given frequency, and the magnitude at the point is how strong that signal is. For example, in frequency space the point (1,0,0) would describe a frequency of 1 in the x direction and 0 in both y and z. If the data from the ultrasound had zero noise there would be a distinct peak at the point that describes the frequency for the marble, but the data obscures this peak. In order to discover the marbles frequency, the data must be averaged across all 20 times.

To do this the data at each time is transformed into frequency space using the FFT and added together. Because the background noise is more or less be random the uncontrollable noise will dissipate out leaving the center frequency of the marble to stand out. This is the

importance of taking many ultrasound readings, with too few readings there would not be enough data to effectively average out the noise. To average all that needs to be done is to add each array of points together for the twenty times and divide by the number of arrays added together (twenty). From here the marbles center frequency can be found at the point in the frequency domain with the largest magnitude.

Before converting back into spatial domain from frequency domain for each of the twenty different time points a filter should be applied to the data. The purpose of the filter is to exaggerate the data around the already known center frequency so when transformed back into spatial domain the location of the ball becomes clearer. In one dimension the filter would take the form of a gaussian centered at the center frequency. In the third dimension this can be expanded to be a gaussian for each direction giving another 64x64x64 array with the largest intensity at the center frequency that can by multiplied to the data in the frequency domain. One important not on the filter is that it is not magically exaggerating a random center frequency to give and erroneous reading. If the filter is applied to an incorrect location that does not match with the marbles center frequency the path of the ball won't be tracked.

Implementation

To begin the data must be set up and ordered in a way Matlab and the viewer can digest, this happens in the first ten lines of code in Appendix B (All lines of code referenced will be found in Appendix B). A variable L is created to define the bounds of physical space where the data lives which is 15, this number is a given by the data from the ultrasound. As well as a variable n is created, this is the number of points along each direction the ultrasound recorded (64), this leads to each set of data in time being an nxnxn array. In the next two lines the locations for all xyz coordinates in spatial as well as frequency domain are being set up. In the spatial domain for x,y, and z the first point is at -L or -15 and goes along a constant interval until one space before L, this is due to the periodicity of using the FFT. When creating the coordinate points in frequency domain the points are evenly spaced and scaled by as factor of $\frac{2*\pi}{2*L}$ this is because the FFT expects to view a period of 2π and must be scaled to be 2L which is the length of each side of the space in question. Using the meshgrid command the points for physical and frequency space are created to be 3 dimensional arrays. [X, Y, Z] relating to physical space, [Kx, Ky, Kz] relating to frequency space. Now that the file Testdata is loaded in and the locations for frequency and physical space that is being worked in is set up the data can begin to be analyzed.

As said earlier the first step is to average the data in frequency space. This process is done from lines 13 to 19. To start a for loop is created which runs from 1 to 20 each one corresponding to a different point in time. On line 15 the data from the Testdata which has a variable named "Undata" is loaded and the physical nxnxn array for the correct time is taken out and assigned to "Un". The reshape command must be used because "Undata" is a

20x262144 matrix, where each column is a different time and the row being all the data that fits into a nxnxn array. An FFT using the fftn command is taken of Un and added to the "ave" variable. By the time the for loop is done each point in time has been reshaped to Un, transformed into frequency space, and added to "ave". After the loop ave is made to be the absolute value of itself and divided by 20 (the number of points in time we have data for). The absolute value if for the ease of viewing data as a lot of imaginary numbers get created due to the FFT, but it won't change the final outcome of the data.

Next after obtaining the nxnxn array with the averaged data the maximum value which corresponds to the center frequency of the marble, and more importantly its location in the matrix, must be found. Using the max command on line 21 the variable "C" and "I" are created, which respectively relate to the magnitude of the max point and the index location in the matrix that refers to the point. As a sanity check it can be nice to see that "C" is indeed much larger than the background noise though the exact number is not the primary concern. On line 29 through 34 the code calculates the average magnitude across all points in the "ave" set which turns out to be 56.0753 where "C" is 271.8673. This shows that the point we have calculated is substantially larger than the average and gives more confidence to the point. Using the index location "I" we can find the location of the center frequency in frequency space by indexing it in Kx, Ky, and Kz to receive the values and assign to Xo, Yo, and Zo respectively. The point [Xo, Yo, Zo] is the center frequency for the marble.

The next step done on line 37-39 the filter in the form of the three-dimensional gaussian is created. The equation for a one-dimensional gaussian g goes $g = e^{-\tau*(k-ko)^2}$ where k is the variable, ko is the center of the gaussian, and τ is the rate at which the gaussian decays, with larger numbers being quicker decays and more localizes gaussian. This can be extended in three dimensions to $g = e^{-\tau*((Kx-Xo)^2+(Ky-Yo)^2+(Kz-Zo)^2)}$. Where in this case [Kx, Ky, Kz] are each the nxnxn array for xyz values in frequency space and [Xo,Yo,Zo] is the center frequency. The value of tau doesn't need to be too exact, for this procedure tau being less than .1 starts to give too wide of a filter for good data, so a value of .3 is picked.

The last step done on lines 48-60 is the filtering of the data and turning it back into the spatial domain, lines 57-59 are for visualization and will be talked about in results. Like in the for loop created for averaging this process starts much the same loop, the loop is run through twenty times for each point in time and begins by making a "Un" that is reshaped data. "Un" is then run through the fftn process to output a it's FFT. This FFT is then multiplied by the filter to amplify the effects around [Xo, Yo, Zo], then run through the inverse FFT, ifftn in Matlab. This takes the data from the frequency space and puts it back into the spatial domain, this array being called "UnF". This will be clear enough due to the filter that the position of the marble is easier to pick out. The max command is used on the now filtered data in physical space to pick out the index location ("Im"), of the marble in "UnF". Like in lines 22-24 "Im" is indexed through [X, Y, Z] to get our location for the marble. This data is saved to Xm, Ym, and Zm which is the x, y, and z location respectively for the marble at each time. This process is run through 20 times

for each distinct time we have data for the marble. After which the complete path of the marble can be plotted (See Figure 1)

Results

There are Two main results of interest from this analysis. The first being the center frequency of the marble, which is a point in frequency space and in the code is defined as

[Xo, Yo, Zo]. This point is [1.8850, -1.0472, 0], this means that the center frequency of the marble is 1.8850 in the x direction, -1.0472 in the y direction, and has no frequency signature in the z direction.

The second important result being the plot of all known locations of the marble, but more importantly the one at time 20. The most recent time is the most important because it is the location which a strong acoustic ray can be focused to break up the marble in the dog's intestines. This point is defined in the code as [Xm(20), Ym(20), Zm(20)] and has value,

[-5.625, 4.219, -6.094] in physical space and is shown on figure 1 as the red circle, the green circle is the start.

To better visualize the results an isosurface command can be used in physical space to see the marble lines 57-59. The value at the center of the ball is about .54 so the value the surface is plotting must be smaller as no larger value exist. The closer to .54 the surface is the smaller the marble will appear to be. This can be seen in figure 2 and 3, with the surface depicting .5 creates a much smaller depiction as the value gets closer to the center. As the number the isosurface represents the marble will look bigger and bigger until more blobs appear in other locations.

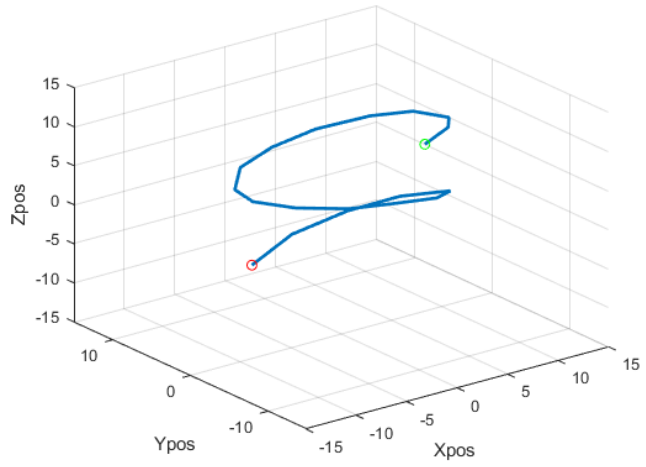
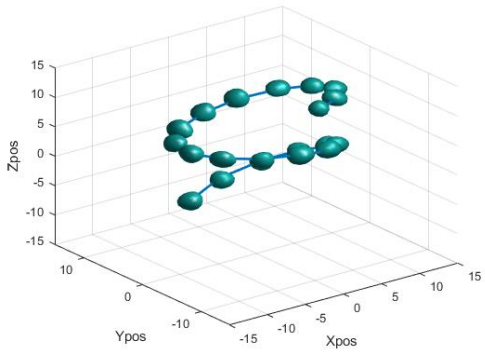


Figure 1



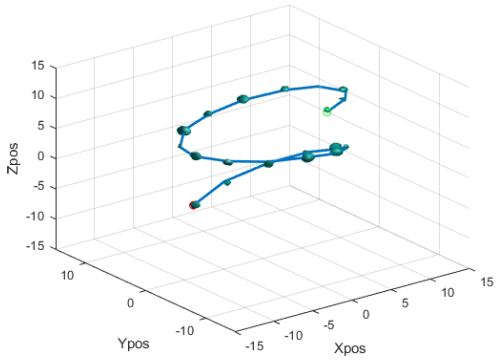


Figure 2 isosurface= .5

Figure 3 isosurface= .35

Conclusion

In summary using a 64x64x64 array of data taken at twenty points in time from ultrasound the location of a marble can be found. Using the Fourier transform to analyze the data in frequency space, all twenty points of data can be averaged to find the center frequency of the marble. A filter centered around the center frequency can then be multiplied to the FFT from each point in time then converted back to the spatial domain to have a clear location of the marble. The marble can be visualized using an isosurface plot looking at a value near that of the marble. Using this data an intense acoustic wave can be focused at the now known location of the marble to break it up and thus saving the dog it was lodged in.

Appendix A. list of used Matlab commands and their functions.

meshgrid- This command takes in a list of values in three directions and makes an array of points to grid for those coordinates.

reshape- Will take data in vector form a turn it into a three dimensional array based off given parameters

fftn- Will create a FFT for the nth dimension, output dimensions will be the same as input

max – the max command will output the maximum value in a vector. If it is a 3d array named array it may be written as [a, b] = max(array(:)), and a will be max value, b will be the index location corresponding to a such that array(b)=a

ifftn- takes the inverse FFT for nth dimension, output dimensions will be same as input

isosurface- takes three-dimensional data and creates a surface along the imputed value, where all values along the surface and the same as the input value.

Appendix B. the numbers at the beginning of each line are for reference

```
1   clear; close all; clc;
 2  load Testdata
 3
 4  L=15; % spatial domain
 5  n=64; % Fourier modes
 6  x2=linspace(-L,L,n+1); x=x2(1:n); y=x; z=x;
 7  k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1];
 8
 9  [X,Y,Z]=meshgrid(x,y,z);
10  [Kx,Ky,Kz]=meshgrid(k,k,k);
11
12
13  ave=zeros(n,n,n);%creates array to turn into average
14  for j=1:20
15      Un(:,:,:)=reshape(Undata(j,:),n,n,n);
16      Unt=fftn(Un);%makes an FFT
17      ave = ave+Unt;%adds the FFT to the average
18  end
19  ave = abs(ave)./20;
20
21  [C,I] = max(ave(:)) %C is max value I is the location index
22  Xo=Kx(I)
23  Yo=Ky(I)
24  Zo=Kz(I)
25
26  %this creates a number for the average magnatude of all
points in frequency
27  %space so for a sanity check one can see that C is in deed
much greater
28  %than the background noise
29  avemag=0;
30  timesthrough=0;
31  for i= 1:n*n*n
32      avemag= avemag+ave(i);
33  end
34  avemag=avemag/(n*n*n)
35
36
37  %this creates a filter for the data at point Xo,Yo,Zo
38  tau=.3;
39  Filter= exp(-tau*((Kx-Xo).^2+(Ky-Yo).^2+(Kz-Zo).^2));
40
41  %creating positons for the marble in different times
42  Xm=zeros(1,20);
```

```
43  Ym=zeros(1,20);
44  Zm=zeros(1,20);
45
46  %goes through each of the 20 time points takes the FFT
applies the filter
47  %then uses iFFT to get a clear location on the marble
48  for j=1:20
49      Un(:,:,:)=reshape(Undata(j,:),n,n,n);
50      Unt=fftn(Un);
51      UnFt=Unt.*Filter;
52      UnF=abs(ifftn(UnFt));
53      [M,Im] = max(UnF(:));%Im is the marble index location in
space
54      Xm(1,j)=X(Im);
55      Ym(1,j)=Y(Im);
56      Zm(1,j)=Z(Im);
57      isosurface(X,Y,Z,abs(UnF),.5)%plotting the marble as an
isosurfate
58      axis([-20 20 -20 20 -20 20]), grid on, drawnow
59      hold on
60  end
61
62
63  %plotting path as well as start in green and end in red
64  plot3(Xm,Ym,Zm,'linewidth',2)
65  hold on
66  plot3(Xm(20),Ym(20),Zm(20),'ro',Xm(1),Ym(1),Zm(1),'go')
67  axis([-15 15 -15 15 -15 15]), grid on, drawnow
68  xlabel('Xpos')
69  ylabel('Ypos')
70  zlabel('Zpos')
```