Save Dog By UltraSound Analyzing

Yiling Sun

AMATH 482 HW01 Winter 2019

abstract

This report describes the method of analyzing noisy ultrasound data with a real marble-in-dog's-intestines case. Our data is obtained concerning the spatial variations in a small area of the intestines where the marble is suspected to be. Data is denoised in frequency domain by the method of averaging and then applied Guassian filter. By doing so, we target the object and keep its dynamic in time domain. The useful signal extracted from data gives the trajectory of marble and the last location of marble is given by the 20th data set.

1. Introduction and Overview

A marble is somewhere in my dog fluffy's intestine. The vet was trying to locate the marble and break it with the aid of ultrasound. But fluffy kept moving when doing the ultrasound, the spatial data is so noisy that we cannot get the location of marble immediately from it. So, our task is to take away those noise from the spatial data and obtain the location of marble, otherwise I would lose my dog.

2. Theoretical Background

To analyze the data with noise, we need Fast Fourier Transform to translate between time and frequency domain, averaging method to denoise the data and Guassain filter to find the useful information.

2.1 Fast Fourier Transform(FFT)

Fourier Transform translates the data between time domain and frequency domain. Basically, it can represent function as sum of sines and cosines. The Fourier transfrom and its inverse are defined as

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} f(x)\, dx \tag{1}$$

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} F(k)\, dk \tag{2}$$

, where $e^i kx = coskx + isinkx$(Euler's formula) is the Kernel describing the oscillatory behavior, and k is the wave number.

In the view of Matlab, the commands for Fourier Transform and its reverse are **fft(u)** and **ifft(u)** with operation account $O(n \log n)$. To be praticle in Matlab, finite number of points frequencies on finite domain is required. Also, there are two restrictions worth-noticing. First, based on the characteristic of $coskx+isink$, the periodic boundary condition is necessary. Second, the Fourie mode of frequencies is in 2π periodic domain. Data need to be scaled into $2\pi/L$ periodic domain beforehand.

For higher dimension application, we use the Matlab command **fftn(u)** and **ifftn(u)**.

2.2 Filtering

To examine our spectual data, we apply two types of filtering technique, filtering by averaging and the Gaussain Filter.

(a) Averaging

Based on the idea that the random white noise has zero mean. In idea situation, if we add all data up, the sum of white noise would go to zero and only useful signal is left.

(b) Gaussain Filter

The Gaussain Filter is one of the simplest filter to eliminate unnecessary noise. Because we only interest in the information around specific frequencies, so we can filtering out the information with other frequencies component. The basic Guassain filter is in the form of

$$F(k) = exp(-\tau(k - k_0)^2), \tag{3}$$

where τ is the bandwidth of the filter and k is the wave number. It extracts information around the wavenumber k_0, which is the center-frequency of the desired signal field.[1] Furthermore, it can be applied to higher dimension appliction by adding k variables.

3. Algorithm Implementation and Development

1. Find the center frequency

The following steps are built on the assumption that the spectral signature remains unchanged as the marble moves over the time domain. So, we can find one fixed maximum frequency point as the frequency center.

Prepare data

First of all, we set the half-length of spacial domain to 15 and Fourier mode to 64, and rescale our frequency domain by $2pi/2L$, since FFT assumes 2π periodic domain. We create the time and frequency grid with these set span x,y,z and kx,ky,kz.
Then, with 20 sets of 3 dimension data take at 20 moments, we reshape them into 64*64*64 grids. Then, we transform them from time space to frequency domain row by row in **for loop**. Because this transformation is in higher dimension, so we apply **fftn(u)** instead of fft(u).

Averaging data

Still in this for loop, we add these 20 sets frequency data up into a new set of data. This new data set contains the cleaned up spectrum in 64*64*64 grid. We need to arrange data into right position in frequency space by **fftshift** for the convenience of finding the frequency center in frequency domain. Then, we get our idea averaging data by dividing it with 20 and normalizing it.

Looking for center

The location of maximum value in the frequency domain is the frequency signature created by the marble. I find the center by the command **ind2sub(find())** and translate the exact location back into the grid of frequency space.

[1]Description from 582 notes by J. Nathan Kutz

2. Find the path of marble

Prepare data

The filter is built on frequency space with the center frequency. In order to do the data filtering, we need to take the data back to frequency space and shift it to the ordered position row by row in a new for loop.

Guassian filter

From the first part, we find the marble's location information in frequency space. Hence, we can build the Guassian filter centered on the found frequency center,

$$F(k) = exp(-0.2 * ((k - k_x)^2 + (k - k_y)^2 + (k - k_z)^2)). \tag{4}$$

The raw data in frequency domain is filtered by this Guassain filter centered around the wavenumber k_x, k_y, k_z with a bandwidth parameter $\tau = 0.2$. Bandwidth 0.2 is chosen here because it is appropriate for the nature of our data.

Then, we can get 20 set of data with desired location information. Then, we take the frequency information back to time space by **ifftn** and normalize them to find the position of the maximum values in each data set. We assumed that the positions of the maximum frequencies in each set are the positions of marble at that time.

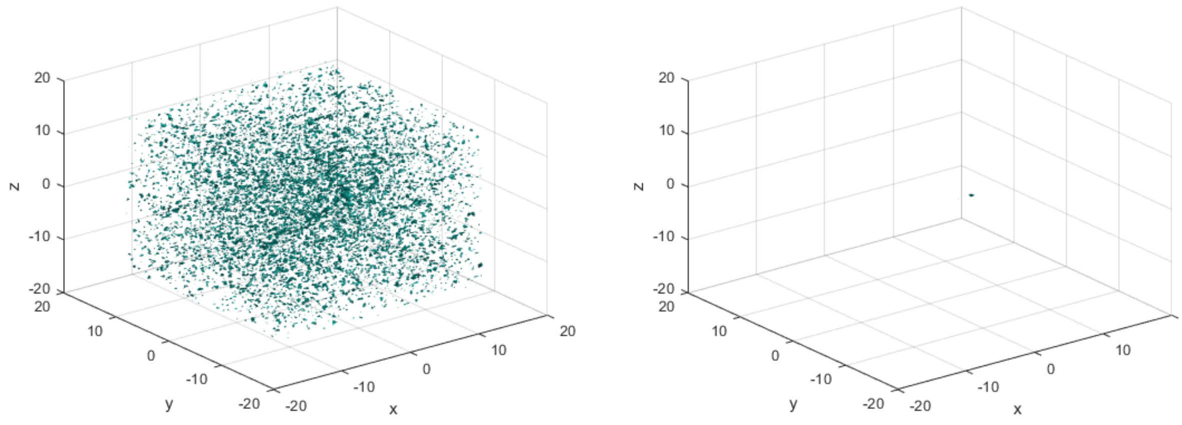

Figure 1: average data with isovalue 0.4 and 0.8 in frequency spectrum

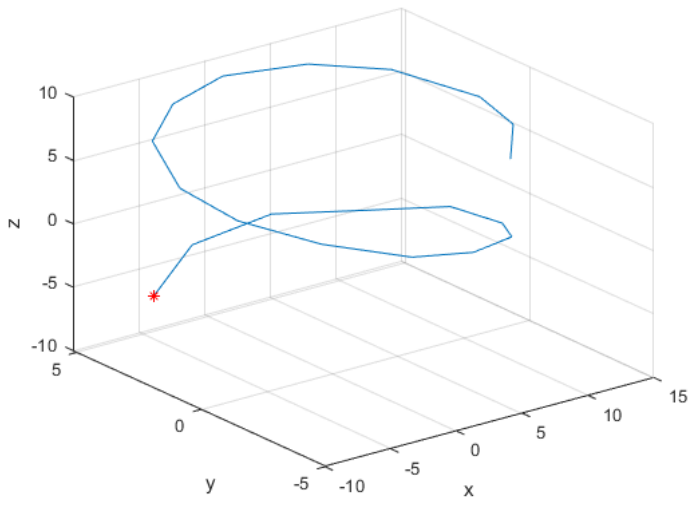

Figure 2: trajectory of marble in time domain

4. Computational Results

By adding up these 20 sets of raw data in frequency domain and taking the average of them, we get one average image of data without most of the noise. FIG. 1 are produced by the Matlab command **isosurface**. It is still hard to find the maximum frequency with isovalue 0.4, so I increase the isovalue from 0.4 to 0.8, then I get one clear point that is the frequency center,

$$\mathbf{x = 1.8850,\ y = \text{-}1.0472,\ z = 0} \tag{5}$$

Then our filter is built based on this frequency center. By applying the filter on 20 data sets, I get 20 positions of marble. In FIG.2, I use **plot3** to plot these 20 points, which are connected to form the path of the marble movement. Also, I point out the last point with red star, its location is

$$\mathbf{x = \text{-}5.6250,\ y = 4.2188,\ z = \text{-}6.0938} \tag{6}$$

5. Summary and Conclusions

Analyzing the nosiy ultrasound data by applying the averaging method and Guassain filter in frenquency domain can give the location information. For our case, we find 20 locations of marble in 3D grid with given 20 sets of data. The trajectory with these 20 locations in order is the suspected path of marble movement in dog's intestine. Also, the marble is observed to keep twisting down. With these location information, we can further make prediction on the next position of marble. Once we take back the location information to the small area in the intestines, we can use an intense acoustic wave to focus on that location and breakup the marble to save my dog.

Appendix A: MATLAB functions used and brief implementation explanation

Y = fftn(X) returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm.[2]

X = ifftn(Y) returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm.[3]

Y = fftshift(X) rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.[4]

B = reshape(A,sz1,...,szN) reshapes A into data in N dimension and each dimension has sz1,...,szN elements

a = max(A) return the maximum element in A

a = abs(A) return the absolute value of each element in A

A = zeros(a,b,...) create a*b*.. matrix that all elements are 0

sz = size(A) return the length of each dimension in A

[I,J] = ind2sub(siz,IND) returns the matrices I and J containing the equivalent row and column subscripts corresponding to each linear index in the matrix IND for a matrix of size siz. siz is a vector with ndim(A) elements (in this case, 2), where siz(1) is the number of rows and siz(2) is the number of columns.[5]

fv = isosurface(X,Y,Z,V,isovalue) computes isosurface data from the volume data V at the isosurface value specified in isovalue. That is, the isosurface connects points that have the specified value much the way contour lines connect points of equal elevation.[6]

plot3(x,y,z,...) plots points with x,y,x coefficient in 3D and connects points up with line.

[2]Description from MathWorks
[3]Description from MathWorks
[4]Description from MathWorks
[5]Description from MathWorks
[6]Description from MathWorks

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);

U=Undata;
Usize = size(U,1);

% add up and take average
usum = zeros(n,n,n);
for j = 1:Usize
    u(:,:,:) = reshape(U(j,:),n,n,n);
    ut = fftn(u);
    usum = usum + ut;
end
uave = abs(fftshift(usum))/Usize; %% take average
uaven =  uave/max(uave(:)); %% normalize

figure(1)
close all, isosurface(X,Y,Z,abs(uaven),0.4)
xlabel('x'),ylabel('y'),zlabel('z')
axis([-20 20 -20 20 -20 20]), grid on


figure(2)
isosurface(X,Y,Z,abs(uaven),0.8)
xlabel('x'),ylabel('y'),zlabel('z')
axis([-20 20 -20 20 -20 20]), grid on

%find the frequency location

% find the center of frequency location
[xc,yc,zc]=ind2sub(size(uave),find(uave==max(uave(:))));
% locate it in frequency space
```

```
xcf=Kx(xc,yc,zc);
ycf=Ky(xc,yc,zc);
zcf=Kz(xc,yc,zc);


% build Gaussain filter
filter = exp(-0.2*((Kx-xcf).^2+(Ky-ycf).^2+(Kz-zcf).^2));

% trait every the marble movement in space
path = zeros(20,3);
for j = 1:Usize
    u(:,:,:) = reshape(U(j,:),n,n,n);
    ut = fftshift(fftn(u));
    utf = fftshift(filter.*ut);
    uf = abs(ifftn(utf));
    un = uf/max(uf(:));
    [xc,yc,zc]=ind2sub(size(un),find(un==max(un(:))));
    path(j,1)=X(xc,yc,zc);
    path(j,2)=Y(xc,yc,zc);
    path(j,3)=Z(xc,yc,zc);
end
figure(3)
plot3(path(:,1),path(:,2),path(:,3)),grid on
hold on
plot3(path(20,1),path(20,2),path(20,3),'r*')
xlabel('x'),ylabel('y'),zlabel('z')
```