Signal Denoise and Tracking by Fourier Transform

Crystal Yang

ABSTRACT

Detecting and recognizing frequency that emitted by crafts is becoming more and more important nowadays. How to eliminate the noise causes by target's surrounding become the key part in signal processing. Fourier transform is a formular that takes a function of time and transform it into a function of frequency which allows for further analysis. In this article, I am going to use fast Fourier transform to perform signal processing on a collected acoustic data that emits by a submarine and track its position after eliminating the noise.

1 INTRODUCTION

There is a collected 3-D dataset that captures the activity of a submarine somewhere in Puget Sound, Seattle. The new technology can capture the special acoustic frequency data emits by the submarine, and thus I can make use of the Fourier Transform to analyze the data and locate the submarine. The data is capture during a 24-hour range, and every half hour there is a collection of the acoustic frequency. The dimension of the dataset is 262144 by 49, where the 49 represent the time dimension, and the 262144 represents the location information in 3D space. A lot of noise is captured at the same time; thus, I am going to eliminate the noise by averaging the frequency and create a filter around the center point to localize the submarine.

2 THEORY BACKGROUND

Fourier Transform is widely used in signal processing, we can get the function of frequency by applying Fourier Transform on a function of time. In this instance with the submarine, the function of time is not given, what I have is the data points that contains the location information formatted in 3D Cartesian coordinate, thus, direct Fourier Transform cannot be used, and I am going to perform Fast Fourier Transform on the data. Fast Fourier Transform (FFT) is particular for discrete data and the algorithm behind it is the Discrete Fourier Transform (DFT). On each data point we get $\{x_0, x_1, x_2, \ldots, x_N\}$, the discrete transform is a sequence of number given by:

$$\hat{x}_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n e^{\frac{2\pi i k n}{N}}$$

Note the 2π in this equation which means that the FFT assumes that the signal is periodic. Thus, we need to rescale the signal when plotting the submarine's location. To eliminate the noise, I make use of the filtering over the time domain and averaging over the frequency domain. To use filter to track the path of submarine, I need to find the center in the frequency domain,

and this center is not changing in frequency, thus, I will use FFT to convert the data into frequency and average over each time point to find the center point.

In averaging, I am going to make use of the property of white noise. White noise is the noise that will be captured when we collect frequency data in natural setting. Thus, we can think that in this set of data, there are also a lot of white noise contained which will distracting us from recognize the signal emitted by the submarine. One important property of white noise is that the center of their frequency is at zero, which means that if we average over repeated realizations, we will eventually get rid of all the white noise in the data. Thus, averaging is performed for this reason as well.

3 ALGORITHM IMPLEMENTATIONS AND DEVELOPMENT

First, to find the center location in frequency space, we need to average over the 49 realizations of the signal in the frequency domain. To split the location information stored in the row of the dataset (recall the dimension of the dataset 'subdata.mat' is 262144 by 49), we noticed that that the 262144 can be split into 64x64x64, which constitute the 3D space of the dataset and Fourier mode we should use is 64. To use Fast Fourier Transform transforming the signal into the frequency domain, we can use the built-in function fft() in MATLAB to achieve this, and note that the dimension here is 3, thus, we should use the fftn() to indicate that higher dimension is used. After summing all the frequency realizations in the frequency domain, we average the signal by 49 to get the centralized frequency. Grab the index of the max value in this centered frequency to extract the coordinate. When plotting the centralized frequency, we need to normalize it to make it visible because without rescaling, the true frequency could be way larger than the bound of the graph which will result in not seeing the graph.

Second, we want to eliminate the noise by filtering the signal around the center we find above. The filter used here is a Gaussian filter that looks like this in 3D space:

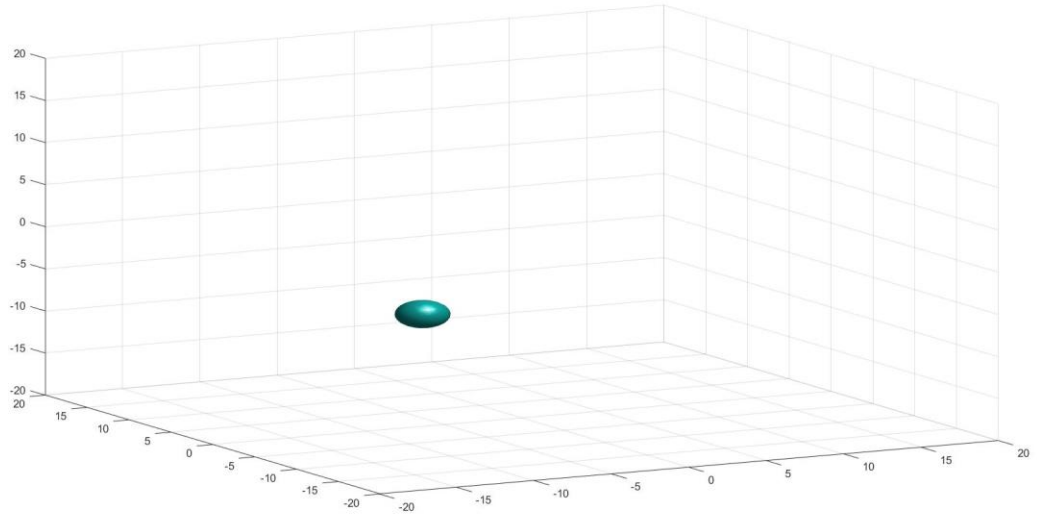

Figure 1: Gaussian filter

For every realization in frequency domain, we create this filter to capture the centered signal so that the signal outside this filter is treated as noise and thus ignored. The question of how large we should set this filter to comes naturally, this will depend on the size of the submarine we are tracking. Since we do not know the size of the submarine in this case, to make sure that the trajectory of the submarine is visible and clear, we set the default relative volume of this filter to be 0.7 (isovalue) in MATLAB. After we get the centralized frequency data, we will use inverse Fourier transform to transform the frequency information into time information and represent it in 3D time dimension.

After we are able to capture the trajectory of this submarine in time domain, in every realization, we also want to keep the 2D location information so that one will be able to track the submarine looking down the surface of the ocean.

4 COMPUTATIONAL RESULTS

According to the averaging frequency signal, the center location of the frequency is at (5.3407, -6.9115, 2.1991) in frequency domain.

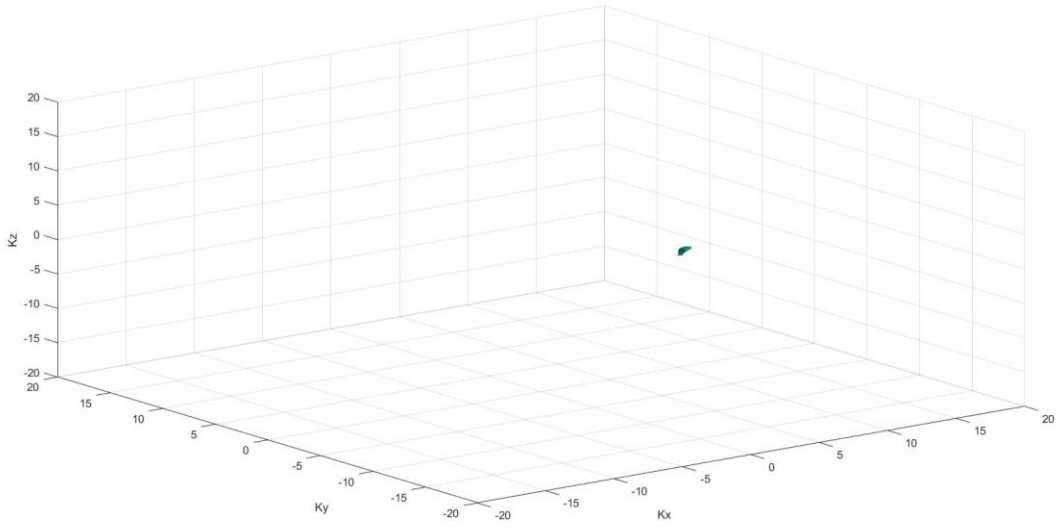

Figure 2: Frequency center

The trajectory of the submarine in 3D dimension:

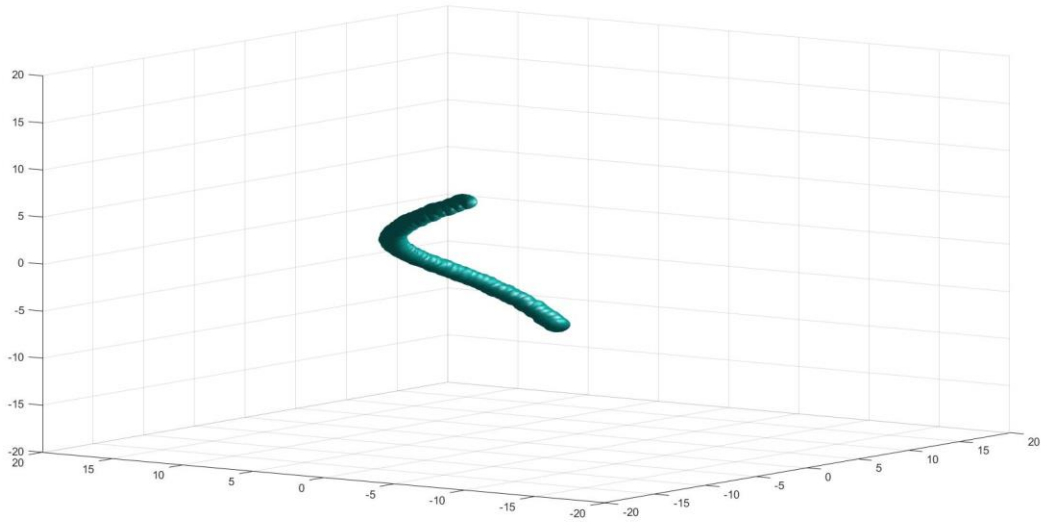

Figure 3: Trajectory of the submarine

The trajectory of the submarine looking from above:

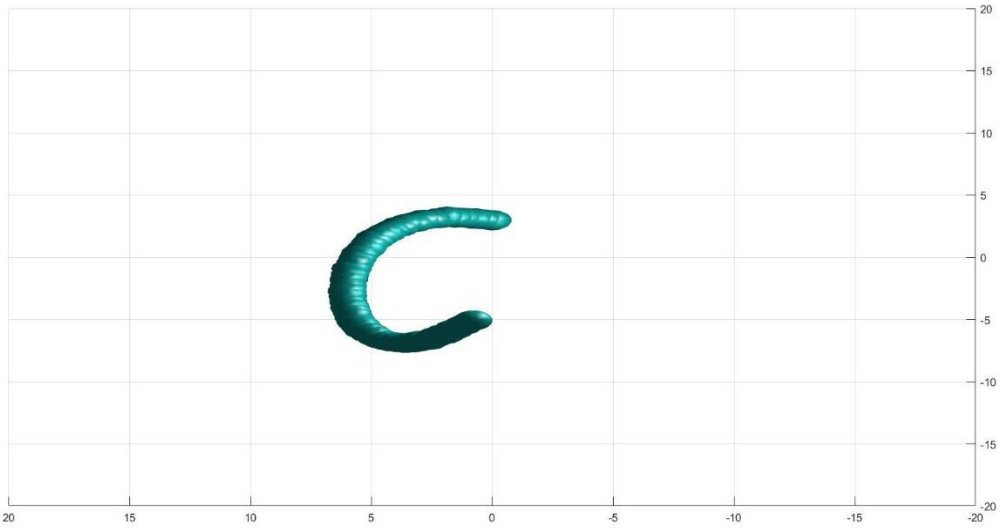

Figure 4: Bird's-eye view of submarine

To send a P-8 Poseidon subtracking aircraft to track the location of the submarine when looking down the submarine above the ocean, the final location that we get from the signal is at (-5.026,0.9425), thus, we should send the aircraft at this location for it to continuously tracking the submarine (see Appendix C. for complete table of location information)

5 SUMMARY AND CONCLUSIONS

From the results shown in the above section, we can see the clear trajectory of the submarine, thus, we successfully track the submarine from the location information given, and by averaging over the frequency domain, we can get the centralized location of the frequency to use this to eliminate the noise. The method to use Fast Fourier Transform to analyze the acoustic frequency helps us to denoise the signal and track the location of the submarine. The trajectory of the submarine is circular, and the submarine is moving toward the surface of the ocean.

APPENDIX A. MATLAB functions used and brief implementation explanation.

`fftn`(): this function calculate the discreate Fourier Transform of the input vector in n dimensions;

`ifftn()`: calculate the inverse dicreate Fourier Transform of the input, turn the function of frequency back to function of time

`fftshift()`:this function shifts the zero component frequency to the center of the image so that we can easily graph it;

`isosurface(X,Y,Z,V,isovalue)`: this function takes the vectors that represent the dimention and plot the vectors 'V' in the XYZ dimension, the specified isovalue represents the volume of V in this dimension;

`[row,col] = ind2sub(sz,ind)`: this function grab the element at the index of size 'sz' vector, and return the index represent row number and col number;

APPENDIX B. MATLAB Code

```
clear all; close all; clc
load subdata.mat

L = 10;
n = 64;
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); % Create the frequency
domain

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

for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    M = max(abs(Un),[],'all');
%     close all, isosurface(X,Y,Z,abs(Un)/M,0.7)
%     axis([-20 20 -20 20 -20 20]), grid on, drawnow
    pause(1)
end

%% Averaging the signal
ave(:,:,:) = zeros(n,n,n);

for j = 1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    Unt = fftn(Un); % fft1() when only one dinmension
    ave = ave + Unt;
end
ave = abs(fftshift(ave))/49;
m = max(abs(ave),[],'all');
isosurface(Kx,Ky,Kz,abs(ave)/m,0.7);
xlabel('Kx')
ylabel('Ky')
zlabel('Kz')
axis([-20 20 -20 20 -20 20]), grid on, drawnow

[m,ind] = max(ave(:));
[ind_x,ind_y,ind_z] = ind2sub([n,n,n],ind);
center_Kx = Kx(ind_x,ind_y,ind_z);
center_Ky=  Ky(ind_x,ind_y,ind_z);
center_Kz = Kz(ind_x,ind_y,ind_z);


%% Filtering the signal and getting the coordinate
tau = 0.2;
filter = fftshift(exp(-tau*((Kx-center_Kx).^2 + (Ky-center_Ky).^2 + (Kz-
center_Kz).^2)));

for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);

    Untf = fftshift(filter.*fftn(Un)); % apply a filter in the time spectrum
    Unf = ifftn(Untf);
    M = max(abs(Unf),[],'all'); % for normalization
```

Appendix C. Table of 2D location information of the submarine

Time	X	Y	Time	X	Y
1	3.141593	0	26	-2.82743	5.969026
2	3.141593	0.314159	27	-3.14159	5.969026
3	3.141593	0.628319	28	-3.45575	5.969026
4	3.141593	1.256637	29	-4.08407	5.969026
5	3.141593	1.570796	30	-4.39823	5.969026
6	3.141593	1.884956	31	-4.71239	5.654867
7	3.141593	2.199115	32	-5.34071	5.654867
8	3.141593	2.513274	33	-5.65487	5.340708
9	3.141593	2.827433	34	-5.96903	5.340708
10	2.827433	3.141593	35	-5.96903	5.026548
11	2.827433	3.455752	36	-6.28319	5.026548
12	2.513274	3.769911	37	-6.59734	4.712389
13	2.199115	4.08407	38	-6.59734	4.39823
14	1.884956	4.39823	39	-6.9115	4.08407
15	1.884956	4.712389	40	-6.9115	3.769911
16	1.570796	5.026548	41	-6.9115	3.455752
17	1.256637	5.026548	42	-6.9115	3.455752
18	0.628319	5.340708	43	-6.9115	2.827433
19	0.314159	5.340708	44	-6.59734	2.513274
20	0	5.654867	45	-6.28319	2.199115
21	-0.62832	5.654867	46	-6.28319	1.884956
22	-0.94248	5.969026	47	-5.96903	1.570796
23	-1.25664	5.969026	48	-5.34071	1.256637
24	-1.88496	5.969026	49	-5.02655	0.942478
25	-2.19911	0			