Utilizing the Fourier Transform for Unknown Object Tracking

Carlos Sama Jan. 2021

Abstract

This report provides a solution for a specific instance of object tracking in which a moving, or stationary target is emitting an unknown frequency. The method presented utilizes the Fourier Transform to analyze frequencies of recorded sonar signals. Through a process of frequency averaging, and gaussian filtering, the unknown emitted frequency is determined and the location from which it was detected is re-traced.

Introduction

In order to determine spatial and environmental knowledge underwater, submarines must use sonar, which emits a specific frequency and listens for the echo response [1]. The problem presented involves attempting to track the motion of a submarine using sonar; however, the frequency of the sonar is unknown. Furthermore, due to countless other sources of sound emission, on or above the ocean, the recorded data is very noisy, and the exact signal location of the targeted submarine cannot be clearly identified. So, a method is implemented that can ignore background noise and identify the unknown frequency [2], which in turn allows for a targeted interpretation of the received sonar data meaning the submarine can now be tracked.

Theoretical Background

Initially, the recorded data provides no insight as to the whereabouts of the targeted submarine, or even if it is there. This is due to the abundance of noise in each signal. However, over an appropriate number of samples, the unwanted noise can be canceled out by averaging in the frequency domain. This is possible because of the random nature of white noise, which at some instances will have positive components, while in other instances have negative components, giving an overall average of a close to zero component. However, the sonar signal will continue to emit the same frequency throughout each sample and thus is unaffected by averaging.

In this example, to identify the submarine, forty-nine samples over a twenty-four-hour period are analyzed. The data is transposed into the frequency domain using the Discrete Fourier Transform (equation 1) where the frequency averaging process is applied [3].

(1)

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-i\frac{2\pi nk}{N}}$$

(2)

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{i\frac{2\pi nk}{N}}$$

However, the spatial information of the submarine is lost through the frequency domain transfer. Therefore, a gaussian filter is designed which highlights the submarines emitted frequency by mapping the frequency components outside of a specified threshold to zero. The filter is applied in the frequency domain and then each sample is transferred back into the spatial domain using the inverse Discrete Fourier Transform (equation 2). Consequently, the spatial signals are recreated using only the frequency information around the submarine, allowing its location to be clearly identified and tracked.

Algorithm Implementation

The defined algorithm was implemented in MATLAB. Provided data was analyzed as a three-dimensional, 64^3 element, square array which was tabulated across a provided distance. In total, forty-nine samples were analyzed. A for-loop iterated through each sample, transferring it into the frequency domain using the `fft` command, creating a three-dimensional array representation of the frequency space for that sample. The absolute values of each array were added on top of the prior array through the for-loop, creating a running sum of all frequency components for the forty-nine samples. The resulting array was divided by forty-nine, thus creating an average of the frequency components measured. Using an element wise implementation of the `max` command, the most significant frequency was found analytically from the averaged array. Furthermore, the array was shifted using `fftshift` and plotted using the `isosurface` feature, allowing for a qualitative interpretation of the significant frequency. A three-dimensional gaussian filter (equation 3) was centered on this frequency with location (x, y, z) and then individually multiplied to each of the sample's frequency domains (η).

(3)

$$f(x,y,z) = \exp\left[-\tau * \begin{pmatrix} (\eta_x - x)^2 + \cdots \\ (\eta_y - y)^2 + \cdots \\ (\eta_z - z)^2 \end{pmatrix}\right]$$

This filtered domain was then transferred back into the spatial domain using the `ifft` command. A similar process was repeated on the new spatial domain where the maximum of the absolute value of the data was identified in each of the forty-nine samples. These points were plotted for inspection and revealed a trajectory of the submarine.

Results

The averaged data was first visually inspected to find the sonar frequency. The frequency component absolute values are plotted in Figure 1, with the significant sonar frequency identified by the red dot.

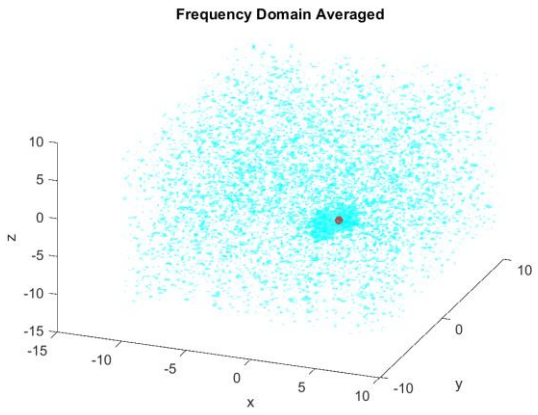

Figure 1 Frequency domain data, averaged from each sample.

The coordinates of the 3D frequency representation corresponding to the maximum point were identified as [5.3407, -6.9115, 2.1991]. A filter was then constructed around his point and encompasses a threshold of $\tau = 1$. This frequency area is illustrated in Figure 2 by the green bubble.

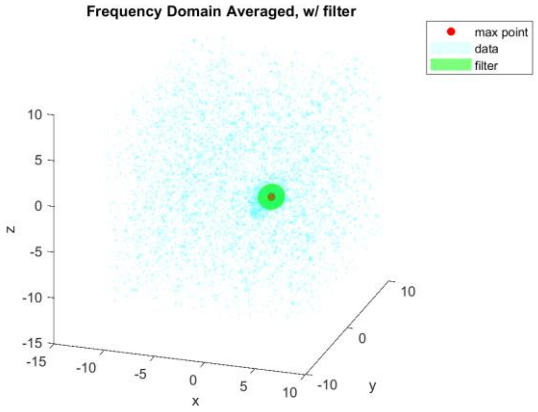

Figure 2 Averaged frequency data with a filter around the identified sonar frequency.

The filtered data, once transferred back into the spatial domain, created a significantly less noisy trace, and the maximum value was isolated and to yield the trajectory plotted in Figure 3. The identified coordinates for each sample are detailed in Table 1, in Appendix A.

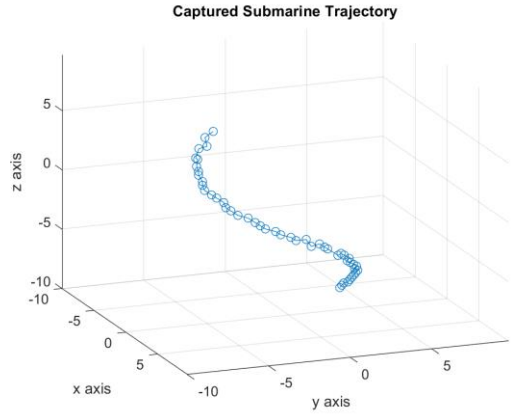

Figure 3 Plotted trajectory of the submarine from filtered data

The submarine appears to be moving in a helictical pattern but with a changing radius; however, in the more recent data, it appears to change x and y position by 0.625 units per hour. The submarine is also ascending at a constant rate of 0.625 units per hour. Since this trajectory is calculated from old data, to intercept the submarine, it's position must be predicted. If the goal is to destroy the submarine, depth charges should be released to explode around the coordinates [0.6250, -4.6875, 6.8750] at precisely thirty minutes after the last sonar signal was acquired.

Conclusion

By transforming data into frequency space, unique operations and analysis can be performed to achieve information not normally visible. The technique of averaging frequency data was able to eliminate white noise to the extent that an unknown signal frequency was clearly identified. And through the capabilities of the Fourier Transform, the information discovered in one domain was easily applied into the other, allowing a cloudy mess of data to reveal a well-defined signal. Thus, this methodology has major implications in fields of radar and sonar processing where objects must be detected from highly noisy signal measurements.

Appendix A

Utilized MATLAB Functions:

`fftn:` This function performs the Discrete Fast Fourier Transform on a provided N-dimensional array and returns the calculated frequency components in the same dimensioned array. This transform function was specifically necessary as the input data was in a 3-dimensional array and needed to be returned as such.

`ifftn:` This function performs the Inverse Discrete Fast Fourier Transform on a provided N-dimensional array and returns the calculated signal in the same dimensioned array. This transform function was specifically necessary as the input data was in a 3-dimensional array and needed to be returned as such.

`fftshift:` This function takes a provided array of frequency data and shifts the negative frequency components to the left of the zero so that the components are arranged in ascending order. It was used to better visualize the frequency data in 3D space as it was tabulated from negative to positive.

`max:` This function returns the maximum value and its index of a provided dataset. The operation can be further defined using the "[], 'all'" modifiers to return one maximum over an entire N-D dataset, whereas the initial functionality would return a maximum of each two dimensional set.

`isosurface:` This function was used to plot the 3-dimensional signal data. It uses a defined threshold to group similar points to create a 3-D surface. A threshold of 0.2 was used for plotting frequency data.

Table 1 estimated location of the submarine

X coordinate	Y coordinate	Z coordinate
0	3.125	-8.125
0.3125	3.125	-7.8125
0.625	3.125	-7.5
1.25	3.125	-7.1875
1.5625	3.125	-6.875
1.875	3.125	-6.5625
2.1875	3.125	-6.25
2.5	3.125	-5.9375
2.8125	3.125	-5.625
3.125	2.8125	-5.3125
3.4375	2.8125	-5
3.75	2.5	-4.6875
4.0625	2.1875	-4.375
4.375	1.875	-4.0625
4.6875	1.875	-3.75
4.6875	1.5625	-3.4375
5	1.25	-3.125
5.3125	0.9375	-3.125
5.3125	0.3125	-2.5
5.625	0	-2.1875
5.625	-0.3125	-1.875
5.9375	-0.9375	-1.875
5.9375	-1.25	-1.25
5.9375	-1.875	-1.25
5.9375	-2.1875	-0.9375
5.9375	-2.8125	-0.625
5.9375	-3.125	-0.3125
5.9375	-3.75	0
5.9375	-4.0625	0.3125
5.9375	-4.375	0.625
5.625	-4.6875	0.9375
5.625	-5.3125	1.25
5.3125	-5.625	1.5625
5.3125	-5.9375	1.875
5	-5.9375	2.1875
4.6875	-6.25	2.5
4.6875	-6.5625	2.8125
4.375	-6.875	3.125
4.0625	-6.875	3.4375
4.0625	-6.875	3.75
3.4375	-6.875	4.0625
3.4375	-6.875	4.375
3.125	-6.875	4.6875
2.5	-6.5625	5
2.1875	-6.5625	5
1.875	-6.25	5.625
1.5625	-5.625	5.625
1.25	-5.625	6.25
0.9375	-5	6.5625

Appendix B

MATLAB CODE

```
clear all; close all; clc
load subdata.mat % Imports the data as the 262144x49 (space
by time) matrix called subdata

L = 10; % 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]; % scale frequencies
to the period of the spatial domain (1 = period length...)
ks = fftshift(k);
[X,Y,Z]=meshgrid(x,y,z);
[Kxs,Kys,Kzs]=meshgrid(ks,ks,ks);
[Kx,Ky,Kz]=meshgrid(k,k,k);

Unt_avg = zeros(n,n,n);
for j = 1:49
          Un(:,:,:)= reshape(subdata(:,j),n,n,n);
          Unt = fftn(Un);
          Unt_avg = Unt_avg + Unt;
end
Unt_avgs = abs(fftshift(Unt_avg)) / 49;
M = max(Unt_avgs,[],'all');
[a, b, c] = ind2sub([n,n,n], find(Unt_avgs == M));
M_idx = [a, b, c];

figure(1)
hold on
[r,c,v] = ind2sub(size(Unt_avgs),find(Unt_avgs > 75));
%scatter3(ks(c),ks(r),ks(v),'o','filled')
scatter3(ks(M_idx(2)), ks(M_idx(1)),
ks(M_idx(3)),'o','r','filled')
xlabel('x')
ylabel('y')
zlabel('z')
p = patch(isosurface(Kxs,Kys,Kzs,Unt_avgs/M, 0.2));
p.FaceColor = 'cyan';
p.EdgeColor = 'none';
p.FaceAlpha = 0.1;

tau = 1;
filter = exp(-tau*((Kx - ks(M_idx(2))).^2+(Ky -
ks(M_idx(1))).^2+(Kz - ks(M_idx(3))).^2));
p1 = patch(isosurface(Kxs,Kys,Kzs,fftshift(filter), 0.2));
p1.FaceColor = 'green';
p1.EdgeColor = 'none';
p1.FaceAlpha = 0.5;

traj = zeros(49,3);
for j = 1:49
          Un(:,:,:)= reshape(subdata(:,j),n,n,n);
          Unt =  fftn(Un);
          Untf = Unt.*filter;
          Unf = ifftn(Untf);
          val = max(abs(Unf),[],'all');
          [a, b, c] = ind2sub([n,n,n], find(abs(Unf) ==
          val));
          traj(j,:) = [x(a), y(b), z(c)];
end
figure(2)
plot3(traj(:,1),traj(:,2),traj(:,3),'o-');
xlabel('x axis')
ylabel('y axis')
zlabel('z axis')
xlim([x(1),x(end)]);
ylim([y(1),y(end)]);
zlim([z(1),z(end)])
```

References

[1] “Quick Tutorial on How Fish Finder Works.” *Deeper Smart Sonar: Castable Fish Finder Perfect for Any Fishing Type*, deepersonar.com/us/en_us/how-it-works/how-sonars-work#:~:text=A%20sonar%20device%20sends%20pulses,and%20then%20bounce%20back%20up.

[2] Bramburger, Jason. “Assignment 1: A Submarine Problem.” AMATH 482, University of Washing Applied Mathematics Department. Jan. 2021

[3] “FFT.” *MATLAB & Simulink*, The MathWorks, www.mathworks.com/help/signal/ug/discrete-fourier-transform.html.