Identification of a Foreign Object Obscured by Noise: A Case Study in Killion, the Dog

Kristen Drummey

Department of Physiology and Biophysics, University of Washington

Github: github.com/kldrummey/AMATH582

Abstract

The Fourier transform (FT) is an essential tool in analyzing and processing signals, particularly those that are obfuscated by noise. In this case study, the fast FT (FFT) was used to determine the path and location of a marble that was swallowed by Killion, a Golden Retriever with a poor sense of what is edible and what is not. Twenty ultrasounds measurements of Killion's digestive tract were taken to determine the location of the marble so that it could be destroyed using an acoustic wave. Using an FFT, the spatial data was transformed and averaged so as to determine the maximum values, and therefore likely frequency signature, of the marble. Using this frequency signature, a Gaussian filter was created and applied to the transformed data, which was then transformed back into the spatial domain to determine the location of the marble in Killion's intestinal tract at each measurement. This case study illustrates the powerful practical applications of the FFT in analyzing noisy data.

1 Introduction and Overview

The Fourier transform (FT) is uniquely useful in decomposing spatial or temporal signals into their component frequencies. This transformation allows for determination of an object's frequency signature, which can be used to create filters that can clean noisy data, allowing for more precise analysis of the signal of interest.

In this study, ultrasound measurements were taken of Killion the dog's intestinal tract after it was discovered that he had swallowed a marble. In order to allow Killion to live a long and happy doggy life, the location of the marble needed to be determined so that it could be broken up with a strong acoustic wave. Unfortunately, as with many real-world measurements, the ultrasound data was noisy, completely obscuring the position of the marble during each of the twenty measurements taken. By transforming the spatial ultrasound signals into the frequency domain, averaging the data, and constructing a Gaussian filter, the location of the marble was determined and Killion was able to recover.

Figure 1: Killion, the dog in question.

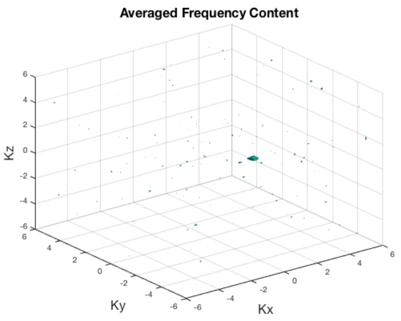

Figure 2: Transformed data, averaged over 20 measurements.

2 Theoretical Background

Fourier proposed that any function, f(x), can be portrayed by a series of sines and cosines, as shown below in Equation 1 and defined on the interval $x \in (-\pi, \pi]$.

$$f(x) = \frac{a_0}{2} + \sum (a_n \cos nx + b_n \sin nx) \tag{1}$$

This ability allows for functions collected outside of the frequency domain to be broken down into their component frequencies through the equation below:

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

The beauty of the FT is that it permits any function, even those that are discontinuous, to be displayed as a series of the function's frequencies. By breaking a function down into its component frequencies, several methods can be applied to the data in order to denoise it and extract a signal of interest. For example, for a signal that is obscured by Gaussian white noise, transforming the signal and noise into the frequency domain allows for averaging of the signal. Since the mean of a Gaussian is equal to zero, averaging across enough measurements will force the noise to go to zero, allowing for a more clear picture of the signal of interest. Additionally, transforming a function into the frequency domain allows for filtering of the signal around the center frequency.

Transformed data is also able to be inverted back into its original domain, using the inverse FT shown in Equation 3. This invertibility means that functions can be filtered and processed in the frequency domain and transformed back to show how the data looks in the original domain where it was collected (e.g., the spatial or temporal domain).

$$f(x) = \frac{1}{\sqrt{2\pi}} \int e^{ikx} F(k) dk \tag{3}$$

A final key component of the FT's functionality is that it can perform transforms very quickly when used on an $O(N \log N)$ scale, rather than on a $O(N^2)$ scale. This difference in scaling allows for the operation count to increase linearly instead of exponentially, greatly decreasing the computational time and effort devoted to transforming a function.

3 Algorithm Implementation and Development

Ultrasound measurements were taken at twenty time points. The raw data acquired by the ultrasound contains a lot of noise, likely due to movement of the dog and obstruction of clean measurements of the marble by the interior of the body. In order to clean up the data and determine the path of the marble, the following steps were taken:

1. Denoise the data by averaging, assuming the noise is Gaussian-distributed. (Algorithm 1)
2. Determine frequency signature of the marble and construct a Gaussian filter. (Algorithm 2)
3. Apply filter to raw data to determine the x/y/z coordinates of the marble at each of the twenty time points. (Algorithm 3)

Algorithm 1: Averaging the Transformed Data

```
Import data from Testdata.mat
for $j = 1 : 20$ do
    Extract measurement $j$ from Undata and reshape
    Take the FFT of the data for each of 20 trials
    Add each FFT and divide by 20
end for
```

Algorithm 2: Construction of Gaussian filter

```
Determine maximum value of averaged data and its index
Use index to find maximum frequency values in Kx,Ky,and Kz
Construct 3-D Gaussian filter using maximum values found above.
```

Algorithm 3: Filter data and find path of the marble

```
for $j = 1 : 20$ do
    Multiply raw data by Gaussian filter
    Pull out spatial coordinates for each of 20 ultrasound measurements
    Plot path of marble and determine coordinates for 20th measurement
end for
```

4 Computational Results

The spatial domain (`L=15`) and Fourier modes (`n=64`) were established and used to construct domains and frequency components. Since the MATLAB `fft()` function assumes that the signals are on a 2π periodic domain, the frequency components were restructured using the previously defined spatial domain and Fourier modes.

Assuming that the noise in the sample is Gaussian distributed allows us to do several things. First, the sample can be averaged over several trials to determine the maximum frequency values. Since a Gaussian has mean of zero, if the noise is Gaussian distributed then averaging it over several trials will cause the level of the noise to approach zero. This allows us to pull out the maximum values, which we can attribute to the frequency signature of the marble. The result of averaging the raw data over twenty trials is shown in Figure 2.

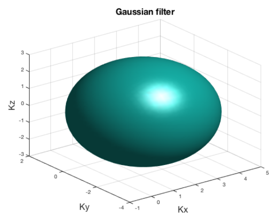

Figure 3: Gaussian filter constructed using maximum values determined by averaging transformed data.

Averaging the raw data does not completely get rid of the noise but does show a clear maximum in the frequency domain. Determining the maximum value and its index allowed for indexing in the Kx, Ky, and Kz axes. These coordinates (`fx,fy,fz`) were determined to be (`1.8850,-1.0472,0`). Using these frequency maximums, a Gaussian filter was constructed using the formula:

$$filter = exp(-0.2 * ((Kx - fx)^2 + (Ky - fy)^2 + (Kz - fz)^2)) \quad (4)$$

Figure 3 shows the structure of this filter. As expected with a 3-D Gaussian filter, its structure is that of a sphere.

After confirmation that the filter looked appropriate, it was applied to the reshaped raw data that had undergone the FFT. After application of the filter, the inverse FFT was applied to return the filtered data into the spatial domain. Figures 4 and 5 show the result of filtering the data in the frequency and spatial domains, respectively.

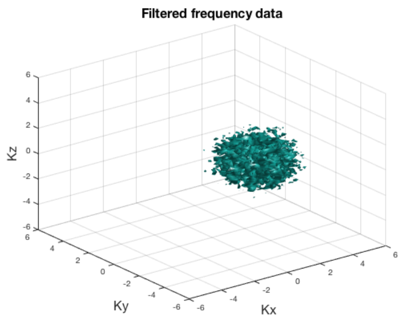

Figure 4: Frequency data after filtering with Gaussian filter

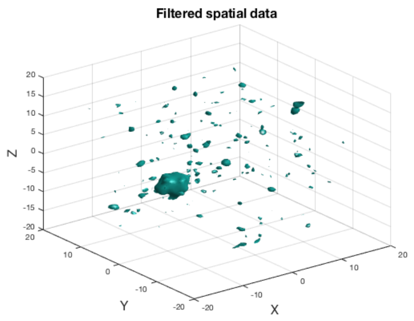

Figure 5: Spatial data after filtering with Gaussian filter in frequency domain, then using the inverse FFT.

Once the data was filtered, the path of the marble was determined over the twenty trials of data that were collected. The marble's path is shown in Figure 6 and appears to be spiraling downward over time. The (X, Y, Z) coordinates of the marble at the twentieth time point was determined to be (`-5.625,4.219,-6.094`). By determining this measurement, an acoustic beam can be focused at those coordinates to break up the marble and save Killion.

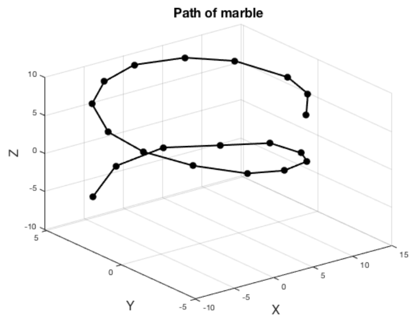

Figure 6: Path of the marble over 20 measurements.

5 Summary and Conclusions

The FFT is an essential tool in signal processing and has a variety of practical applications, ranging from detecting enemy jets on radar to pinpointing the location of a potentially deadly marble in a dog's intestines. As shown in this case study, the FFT can be a powerful tool for pulling usable signal out of a mess of unusable noise.

Fourier analysis takes advantage of the fact that any function, even those that are discontinuous, can be broken down into component sines and cosines. By transforming functions into the frequency domain, Fourier analysis creates the opportunity for precise filtering and denoising of a signal of interest. For Killion the dog, this analysis proved to be extremely useful for locating a marble that he had swallowed that had been obscured by noise in the ultrasound measurements that were taken. Fourier analysis is an incredibly useful tool for signal processing and analysis, one that even produces life-saving results.

6 References

1. Kutz, J.N. (2013). Data-Driven Modeling Scientific Computation: Methods for Complex Systems and Big Data. Chapter 13. Oxford University Press. 1st. Ed.

Appendix A MATLAB Functions

Notable functions used in this analysis:

- `y = linspace(x1,x2,n)` returns a vector of `n` points between `x1` and `x2`.
- `[X,Y,Z] = meshgrid(x,y,z)` returns a grid of coordinates based on the values listed in the vectors x, y, and z, where X is a matrix of rows copied from x, Y is a matrix with columns copied from y, and Z is a matrix of pages copied from z.
- `fftn(u)` returns the n-dimensional fast Fourier transform of the input, u
- `fftshift(u)` returns the fast Fourier transform of the input, u, shifted to the zero-frequency component
- `ifftn(u)` returns the n-dimensional inverse Fourier transform of the input, u
- `ifftshift(u)` returns the n-dimensional inverse Fourier transform of the input, u, shifted to the zero-frequency component
- `isosurface(X,Y,Z,V,i)` computes and plots an isosurface plot, using X,Y, and Z as grid axes, V as the volume at each point in those axes, and i as the isovalue
- `reshape()` reshapes
- `abs(u)` returns the absolute value of u
- `[M I]=max(u)` returns the maximum value (M) and its index (I) of the input, u
- `for...end` constructs a for logic loop, in which the program iterates over the given number of instances and performs the function contained within the loop
- `plot3(x,y,z)` produces a visualization of the vectors x, y, and z
- `zeros(n,n)` produces a matrix of zeros, with size n x n
- `xlabel,ylabel,zlabel` labels the x, y, or z axes of a plot
- `axis` sets the axes limits of a plot

Appendix B MATLAB Code

```
%AMATH582 Winter Quarter 2020
%Homework Assignment #1 - Kristen Drummey

%% ESTABLISH PARAMETERS
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; %domain discretization
k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1]; %frequency components of FFT
ks=fftshift(k); %unshifted frequency components

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

%Reshape data into 64x64x64 matrix with 20 time points.
for j=1:20
```

```
   Un(:,:,:)=reshape(Undata(j,:),n,n,n);
   close all, isosurface(X,Y,Z,abs(Un),0.4)
   axis([-20 20 -20 20 -20 20]), grid on, drawnow
   pause(1)
end

%% AVERAGE DATA OVER 20 TRIALS

%Average data over 20 trials.
ave=zeros(n,n,n); %Setup matrix of zeroes to populate

for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Utn=fftn(Un); %3-dimensional FFT on reshaped data
    ave(:,:,:)=ave+Utn(:,:,:);
end

ave=ave/j; %divide populated matrix by the number of trials to determine average
Utns=fftshift(ave); %shift averaged data
Utnnorm=abs(Utns)/max(abs(Utns(:))); %normalize data for plotting by dividing by max value

figure(1)
close all, isosurface(Kx,Ky,Kz,Utnnorm,0.6)
axis([-6 6 -6 6 -6 6]), grid on, drawnow
xlabel('Kx','Fontsize',[16]); ylabel('Ky','Fontsize',[16]); zlabel('Kz','Fontsize',[16])
title('Averaged Frequency Content','Fontsize',[16])

%% CONSTRUCT GAUSSIAN FILTER AND FILTER DATA

%Determine frequency signature of the marble.
[M I]=max(Utns(:));
fx=Kx(I); fy=Ky(I); fz=Kz(I);

%Gaussian filter using frequency signature
filter=exp(-0.2*((Kx-fx).^2 + (Ky-fy).^2 + (Kz-fz).^2));

%Double check that the filter looks correct - Gaussian filter should be
%circular (or in this case, spherical)
figure(2)
close all; isosurface(Kx,Ky,Kz,abs(filter),0.2);
grid on, drawnow
xlabel('Kx','Fontsize',[16]); ylabel('Ky','Fontsize',[16]); zlabel('Kz','Fontsize',[16])
title('Gaussian filter','Fontsize',[16])

%Filter data across 20 trials
for a=1:20
    Un(:,:,:)=reshape(Undata(a,:),n,n,n); %reshape original data
    Ut(:,:,:)=fftn(Un); %apply n-dimensional FFT to data
    Uts(:,:,:)=fftshift(Ut); %shift transformed data
    Utsf(:,:,:)=filter.*Uts; %apply Gaussian filter to transformed data
    Unf(:,:,:)=ifftn(ifftshift(Utsf));  %revert filtered data to spatial domain
end

%Plot filtered frequency data
```

```
figure(3)
close all; isosurface(Kx,Ky,Kz,(abs(Utsf)/max(abs(Utsf(:)))),0.2);
axis([-6 6 -6 6 -6 6]),grid on, drawnow
xlabel('Kx','Fontsize',[16]); ylabel('Ky','Fontsize',[16]); zlabel('Kz','Fontsize',[16])
title('Filtered frequency data','Fontsize',[16])

%Plot filtered spatial data
figure(4)
close all; isosurface(X,Y,Z,(abs(Unf)/max(abs(Unf(:)))),0.2);
axis([-20 20 -20 20 -20 20]),grid on, drawnow
xlabel('X','Fontsize',[16]); ylabel('Y','Fontsize',[16]); zlabel('Z','Fontsize',[16])
title('Filtered spatial data','Fontsize',[16])

%% DETERMINE AND PLOT PATH OF THE MARBLE
%Plot path of marble
xcoord=zeros(20,1); %establish empty vectors to populate with x,y,z coordinates of the marble
ycoord=zeros(20,1);
zcoord=zeros(20,1);

for a=1:20
    Un(:,:,:)=reshape(Undata(a,:),n,n,n); %reshape original data
    Ut(:,:,:)=fftn(Un); %apply n-dimensional FFT to data
    Uts(:,:,:)=fftshift(Ut); %shift Ut
    Utsf(:,:,:)=filter.*Uts; %apply Gaussian filter to transformed data
    Unf(:,:,:)=ifftn(ifftshift(Utsf)); %revert filtered data to the spatial domain
    [Ms,Is]=max(Unf(:)); %find max value and its index in the filtered spatial data
    xcoord(a)=X(Is); %determine X,Y,Z max values at each of 20 time points
    ycoord(a)=Y(Is);
    zcoord(a)=Z(Is);
    plot3(xcoord,ycoord,zcoord,'.k','MarkerSize',[30],'LineWidth',2,'LineStyle','-') %plot coordinates
    grid on
    xlabel('X','Fontsize',[16]); ylabel('Y','Fontsize',[16]); zlabel('Z','Fontsize',[16]);
    title('Path of marble','Fontsize',[16])
end

%Location of marble at 20th time point
twenty=[xcoord(20),ycoord(20),zcoord(20)];
```