AMATH 482/582: HOMEWORK 1

SATHVIK CHINTA

Abstract. Using the Fast Fourier Transform algorithm alongside a Guassian Noise filter, we were able to succesfully de-noise acoustic data of a submarine located within Puget Sound and track its position over a 24 hour period of time.

1. Introduction and Overview

Imagine we are given the following problem: There is a submarine traveling thorugh Puget Sound. We've been tasked with finding the submarine and tracking it over a 24 hour period. We've been given a 64 x 64 x 64 grid of acoustic data points at 30 minute intervals, but the data is noisy. The submarine is also moving, so we need to determine its path if we want to find it.

Since we're dealing with acoustic frequencies, we should immediately be thinking of Fourier Transforms. These will be key in order for us to find the path of the submarine.

2. Theoretical Background

Imagine we have multiple frequencies being combined together into a single signal. Simply observing the shape of the signal would be very hard to interpret. However, we know that there are multiple underlying frequencies that have been combined in order to give us the output signal. The Fourier Transform is an effective method to decompose such a signal into its underlying frequencies. In our case, since we are dealing with noisy acoustic data underwater, we can stand to believe that there will be multiple frequencies (corresponding to the noise and the submarine). If we were to scrub away the noise, then the highest frequency would be the submarine.

We know the data is noisy. As such, we can use an interesting property of the Fourier Transform in order to help get rid of the noise. Let the function be represented by the following:

$$f(x) = g(x) + \xi$$

Where ξ is the noise and $f(x)$ is the submarine data. The clean data should then be represented by $g(x)$, so the question is how do we get rid of the noise. If we assume the noise is random in distribution with mean 0, then taking the Fourier Transform of the random variable will still be some random variable. As such, we can write this as

$$\hat{f(x)} = \hat{g(x)} + \xi$$

We have a distribution over time (one point at every 30 mintues for 24 hours). If we imagine that there are n number of such points, then if we were to take the average of this fourier transform over time, it would become

$$\frac{1}{n}\sum_{n=0}^{n-1} \hat{f(x)} = \frac{1}{n}\sum_{n=0}^{n-1} \hat{g(x)} + \frac{1}{n}\sum_{n=0}^{n-1} \xi$$

Date: January 28, 2022.

However, since ξ is randomly distributed with mean 0, the noise should approach 0 as well. So, this should simplify to

$$\frac{1}{n}\sum_{n=0}^{n-1} f\hat{(x)} = \frac{1}{n}\sum_{n=0}^{n-1} g\hat{(x)}$$

Effectively meaning that we have dealt with the noise. Once we have gotten rid of the noise, we should be able to scan through the fourier transform and find the highest frequency. Since there should be no other frequencies in the data, we can use this frequency to find the value of the central frequency (and it's x, y, z coordinates).

When we create a filter to denoise the data when finding the path, we can extend the Gaussian filter to three dimensions. In two dimensions, the filter is

$$G(x, y, \sigma) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Where σ is chosen by us. We can extend this definition to three dimensions by just adding another dimension z. In this case, the Gaussian Filter would become

$$G(x, y, z, \sigma) = e^{-\frac{x^2+y^2+z^2}{2\sigma^2}}$$

We will use this filter to attempt to de-noise along the path that the submarine takes. We can also center this around the max frequency in order to better reduce noise around the sub. Let (a, b, c) be the x, y, and z coordinates of the max frequency. Our Gaussian filter would then look like

$$G(x, y, z, \sigma) = e^{-\frac{(x-a)^2+(y-b)^2+(z-c)^2}{2\sigma^2}}$$

Also, the transform would be given to us from range -32 to 32, but we want our range to be from 0 till 64. So, we will add 32 to each dimension as well to get the final Gaussian filter as

$$G(x, y, z, \sigma) = e^{-\frac{(x-a+32)^2+(y-b+32)^2+(z-c+32)^2}{2\sigma^2}}$$

3. Algorithm Implementation and Development

For this assignment, I used Python. I used NumPy to do all mathematical operations and the Fourier Transforms. I used Matplotlib to plot the data and the path of the submarine, as well as any other plots.

Within numpy, I used the following functions:

- `np.reshape()` to reshape the data into a 3D array.
- `np.fft.fftshift()` to shift the fourier transform data.
- `np.fft.fftn()` to take the fourier transform of the data in n dimensions (3 in our case).
- `np.size()` to find the size of the data.
- `np.argmax()` to find the index of the max value in the fourier transform (used to find the central frequency).
- `np.abs()` to find the absolute value of the fourier transform.
- `np.real()` to find the real part of the fourier transform.
- `np.unravelIndex()` to convert the flat index of the central frequency to a tuple of indices, which was then fed into the Gaussian function
- `np.exp()` to take the exponential of the Gaussian function.
- `np.fft.ifftn()` to take the inverse fourier transform after applying the Gaussian function.
- `np.fft.ifftshift()` to shift the inverse fourier transform data.

Within matplotlib, I used the following functions:

`plt.plot()` to plot the data in two dimensions.
`plt.axes(projection = '3d')` to plot the data in three dimensions.

4. Computational Results

Averaging over time after taking the FFT in order to reduce noise, I go the highest frequency as 90. Looking for the x, y, z coordinates of the max frequency, I found the central frequency as (39, 49, 10) with value 90.

Through my testing, I found the ideal value of σ to be used in my Gaussian filter to be 15.

After computing the Gaussian filter and using the central frequency coordinates within them, I multiplied my filter by the result of the Fourier Transform. I then took the inverse Fourier Transform, and repeated the process for each data point (every 30 minutes for the entire 24 hour timespan). This resulted in the following path for the x-y plane:

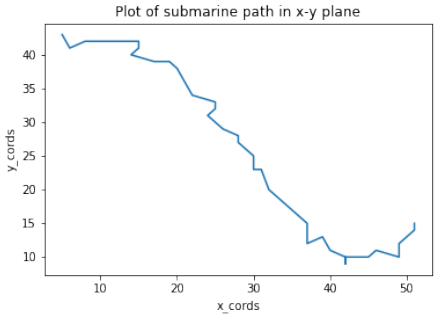

Figure 1. A plot of the x-coordinates and the y-coordinates of the submarine path over the entire 24 hours

We can now clearly see the path that the submarine is taking. The original data itself is in 3 dimensions, so while we can't paint the whole picture just yet (we will graph in 3 dimensions later on), we can see the general shape of the path that the submarine is taking.

This is what the path looks like in 3 dimensions:

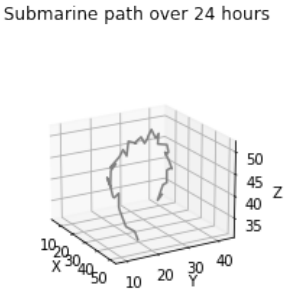

Figure 2. A plot of the of the submarine path over the entire 24 hours in all 3 dimensions

We can see that the path is very clear, with minimal (if any) noise. The jagged lines might be a result of the noise, but doing a bit of research into the subject reveals that submarines do in fact take rapid turns in order to avoid detection.

5. Summary and Conclusions

Even with very messy data in a higher dimension space, Fourier Transforms are still very effective at finding the most significant features. Guassian noise filters are also very effective, and applicable to a wide variety of use cases. We were succesfully able to find the path of the submarine inside of Puget Sound over the 24 hour time period.

Acknowledgements

I am thankful to Professor Hosseini for introducing us to the concept of Fourier Transforms and explaining their significance. I am also grateful to the YouTube Channel "3Blue1Brown" by Grant Sanderson for providing a visual explanation for the intuition behind Fourier Transforms/series, and how they can be used to find signals within data.

I am very thankful to my peers taking the class alongside me, they have helped me understand the material as well as provide a reference to compare my results against. I interacted with them both through Canvas discussion boards as well as Discord chat.

References

[1] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[2] B. Hosseini. High-dimensional extensions and amp; image processing. University of Washington (LOW 216), Jan 2022. AMATH 482/582.

[3] B. Hosseini. Signal processing with dft. University of Washington (LOW 216), Jan 2022. AMATH 482/582.

[4] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[4] [1] [3] [2]