An Ultrasound Problem

Fourier Techniques for Spacio-Temporal Localization in Noisy Ultrasound Data

Dino de Raad

April 11, 2019

Abstract

Noise filtration techniques are applied to raw ultrasound data of a marble in a dog's intestinal tract. Theory on Fourier analysis is introduced. An algorithm is developed taking advantage of signal averaging across data realizations, and using a Gaussian filter to find the path of the marble. Finally, the end point of that path is used for the targeting of a high intensity wave to break up the marble, saving the dog's life.

I Introduction and Overview

We would like to save Fluffy, a dog (Figure 1) who has swallowed a marble. We have access to (noisy) ultrasound data of his intestinal tract, where the marble is suspected to be. Using the Fourier transform to analyze this data, the position of the marble can be determined. Once we know where the marble is, we will use an intense acoustic wave to break the marble into smaller pieces, allowing for easier digestion. The paper will summarize the theory behind techniques, and later implements these techniques in algorithms.

II Theoretical Background

The main approaches taken to analyze the data are simply techniques for refining the Fourier transform. The Fourier transform in 3D is given by:

$$F(\mathbf{k}) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-i\mathbf{k}\mathbf{x}} f(\mathbf{x}) d\mathbf{x}$$

The inverse is similarly:

$$f(\mathbf{x}) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{i\mathbf{k}\mathbf{x}} F(\mathbf{k}) d\mathbf{k}$$

Computationally, these are implemented as the n-dimensional Fast Fourier transform and inverse transform (`fftn`, `ifftn` in MATLAB). The FFT assumes that

Figure 1: My cat, Cream (It is common knowledge that `cat = fft(dog)`).

the input domain will have 2^n points, and that the signal is 2π periodic. This necessitates that we scale the domain of the transform (the wavenumbers k) by $\frac{2\pi}{L}$.

Transforming the signal reveals many properties of the signal, especially when analyzing a relatively noise-free, static signal. However, in our problem, there is a lot of noise obscuring the signal of interest (i.e. the marble). In this problem, the noise is presumably a product of the other internal organs and structures of the body, and movement of the dog. The following techniques help to remove it:

1. **Signal Averaging**. Since we have access to numerous time slices of the data, we can look at how the transformed data stays similar over time, and let that key us into which part of the frequency domain is associated with the marble's path. Over many successive averages, noise will cancel out while the signal of interest, being unchanging in frequency, will be the only information remaining.

2. **Filtering**. Since we want to know (in each time slice) where the marble resides inside of Fluffy, we want to ideally reduce the noise in the signal before returning the signal to the time domain so that it is clearer what is marble and what is Fluffy. Thanks to the previous technique, we will have known where the signal resides in the frequency domain, so now we can focus on removing all of the other data unrelated to the marble. I've elected to use a 3D Gaussian filter, although many other filter options are available. The Gaussian is of the form

$$G(\mathbf{k}) = \frac{1}{\sqrt{(2\pi)^3}\sigma^3} e^{-\frac{1}{2\sigma^3}\left((k_x-k_{x0})^2+(k_y-k_{y0})^2+(k_z-k_{z0})^2\right)}$$

with standard deviation σ and central point k_{j0} on the jth coordinate axis in the frequency domain (variable $\mathbf{k}$ with components k_x, k_y, k_z. Importantly, when we multiply the Gaussian by the signal in the frequency domain, only the frequencies of interest contribute significantly when the filtered signal is brought back to the time domain, since values further from the frequencies at the center will be multiplied by smaller and smaller values further from the central frequency, and thus impact the filtered results less and less.

III Algorithm Implementation and Development

Algorithm 1

The purpose of this algorithm is to denoise the data. The details are as follows:

1. Reshape the initial data structure `Undata` from 20x262144 to 20x64x64x64; this structure corresponds to the 20 time-steps and 64 discretizations of Cartesian space in each axis.

2. Apply the Fast Fourier Transform in n dimensions (`fftn`) to the reshaped data along the time axis and shift the data to the appropriate frequency axes via the `fftshift`.

```
Unt = fftshift(fftn(Un));
```

3. Take the absolute value of the transformed data, then average that data along the frequency domain.

```
UntavgABS = squeeze(mean(abs(Unt),1));
[cfABS, cfI] = max(UntavgABS(:));
```

4. Find the maximum value and index of the Central Frequency in the averaged data (size 64x64x64).

```
cfkx = Kx(cfIKx, cfIKy, cfIKz);
cfky = Ky(cfIKx, cfIKy, cfIKz);
cfkz = Kz(cfIKx, cfIKy, cfIKz);
```

5. Create a Gaussian on the frequency axes centered at the coordinates of the Central Frequency. Multiply it by every time slice of the transformed, shifted data.

```
s = 2^0.5;
Gfilter = (2*pi)^(-3/2)*s^-3 * ...
exp(-(1/(2*s^3))*((Kx - cfkx).^2 + (Ky - cfky).^2 + ...
(Kz - cfkz).^2));
% Replication of the filter for multiplication
G = zeros(size(Unt));
```

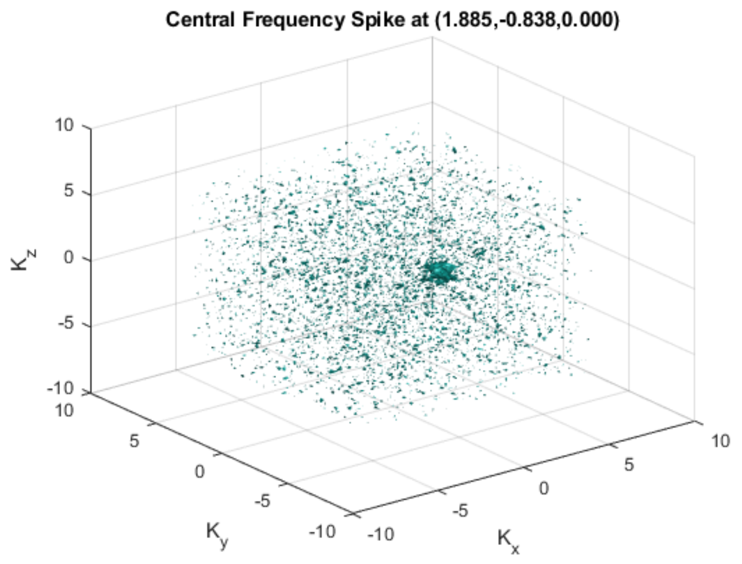

Figure 2: The wavenumbers U_t plotted as an isosurface, where the isovalue is $.725 * max(U_t)$. A large cluster of high-magnitude wavenumbers is apparent. As seen in figures 2 and 3, the marble undergoes a helix motion which is well described by these wavenumbers; no z contribution, and opposite signed x and y contribution

```
for j = 1:T
    G(j,:,:,:) = Gfilter;
end
```

6. Unshift (`ifftshift`) the data then bring it back into the time domain using the Inverse Fast Fourier Transform in n dimensions (`ifftn`). This is the denoised data.

   ```
   UnF = ifftn(ifftshift(G.*Unt));
   ```

Algorithm 2

The purpose of the algorithm is to record the approximate positions of the marble at each of time slices. While not too complicated, this data is crucial since the final position of the marble is exactly the information we need to save Fluffy.

1. For every time slice:

   - Take the 3D slice corresponding to that time,

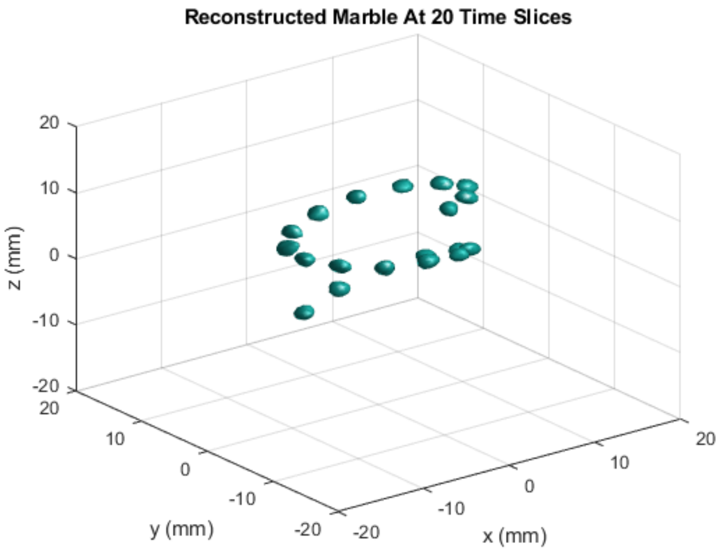

Figure 3: A depiction of the marble through time extracted from the denoised data.

- Find the coordinates of the maximum on the Cartesian axes,
- Record its position.

2. The final entry in the recorded positions is the position of the marble in the 20th time slice.

IV Computational Results

Our first priority is to save Fluffy, so we need to know where the marble is at the end of its path. The coordinates of this end point are

$$(x_{end}, y_{end}, z_{end}) = (-5.6250, 4.2188, -6.0938)$$

in the given x, y, z coordinate axes. In the case that the time axis data is reversed,

$$(x_{start}, y_{start}, z_{start}) = (4.6875, -4.6875, 9.8438)$$

are the coordinates of the starting point of the marble's path. This answer, of course, varies slightly with our choice of filter width (what is meant here by filter width is the parameter σ in the 3D Gaussian). However, we can have some confidence in this answer because the changes in the ending value are marginal until the filter becomes to wide or too narrow, at which point the outline of the marble becomes muddled and the path deviates from the helix path. As we

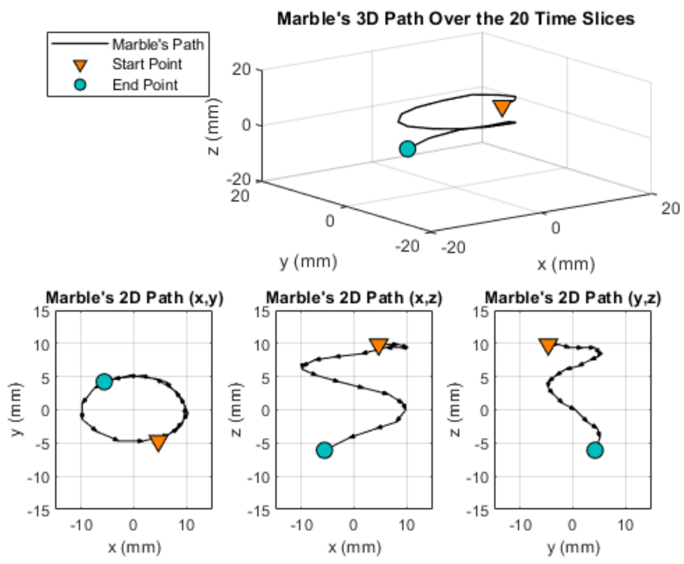

Figure 4: The path of the marble in 3D and 2D cross-sections for each of the coordinate axes. The arrows show at their tip where the marble is at the corresponding time slice. The marble begins at the starting marker (orange triangle) and ends at the ending marker (blue dot).

can see, with our filter width the marble appears as a spheroid (Figure 3) and the path it takes is helictical (Figure 4), which is consistent with the associated wavenumbers in the frequency domain (see note on Figure 2).

V Summary and Conclusions

We are able to both locate the marble and identify the path it takes over the course of the 20 time instances available in the data, using signal averaging and filtering to remove noise. This means, in principle, that if new data about the marble were given, we would be able to locate the marble in that new data. However, we must attribute the ease of determining central frequencies associated with the marble to its motion being easily described by static frequencies; that is, the helictical motion is easily characterized in the Fourier space since it is a periodic motion.

Appendix A MATLAB functions

Appendix B MATLAB codes

```
clearvars -except Undata
close all; clc;
if exist('Undata', 'var')==0
    load Testdata
end

L=15; % spatial domain
n=64; % Fourier modes
T = 20;
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);

Un = zeros([T n n n]);

% Creation of the 4D array representing the 3D ultrasonic image
% at the 20 time slices (dimension 1)
for j=1:T
    Un(j,:,:,:)=reshape(Undata(j,:),n,n,n);
end

% Averaging process; we average along the time axis
Unt = fftshift(fftn(Un));

UntavgABS = squeeze(mean(abs(Unt),1));

[cfABS, cfI] = max(UntavgABS(:));

figure(1)
isosurface(Kx,Ky,Kz,UntavgABS/cfABS,.725)
   axis([-20 20 -20 20 -20 20]./2), grid on, %drawnow

xlabel('K_x')
ylabel('K_y')
zlabel('K_z')

[cfIKx, cfIKy, cfIKz] = ind2sub([n n n], cfI);

% 3D coordinates of the central frequency (presumably the marble)
```

```
cfkx = Kx(cfIKx, cfIKy, cfIKz);
cfky = Ky(cfIKx, cfIKy, cfIKz);
cfkz = Kz(cfIKx, cfIKy, cfIKz);

title(sprintf('Central Frequency Spike at (%.3f,%.3f,%.3f)', cfkx,cfky,cfkz))

% hold on
% m = scatter3(cfkx,cfky,cfkz,25,[0 0 0]);
% m.MarkerFaceColor = [1 0 0];

% 3D Gaussian filter centered at the marble's frequency in k (frq) space
s = 2^0.5; % range of 2^-(0.5, 0.5) give good results (same start, end)
Gfilter = (2*pi)^(-3/2)*s^-3 * ...
    exp(-(1/(2*s^3))*((Kx - cfkx).^2 + (Ky - cfky).^2 + (Kz - cfkz).^2));

% Replication of the filter for multiplication
G = zeros(size(Unt));
for j = 1:T
    G(j,:,:,:) = Gfilter;
end

% Filtered data
UnF = ifftn(ifftshift(G.*Unt));

figure(2)

for j = 1:T
    Ua = abs(squeeze(UnF(j,:,:,:)));
    Uam = max(Ua(:));
    isosurface(X,Y,Z,Ua/Uam,.75)
    hold on
end

axis([-20 20 -20 20 -20 20]), grid on
xlabel('x (mm)')
ylabel('y (mm)')
zlabel('z (mm)')

title('Reconstructed Marble At 20 Time Slices')

% Extract the path from the data
mx = zeros([20 1]); my = mx; mz = mx;

for j = 1:T
    Unf = squeeze(UnF(j,:,:,:));
    [~, I] = max(Unf(:));
    [my(j), mx(j), mz(j)] = ind2sub(size(Unf), I);
end

fig = figure(3);
sp1 = subplot(2,3,[1 2 3]);
p = plot3(x(mx), y(my), z(mz), 'Color', [0 0 0], 'LineWidth', 1);

hold on

markersize = 75;
arrowstyle = 'plain';

h = scatter3(x(mx(1)), y(my(1)), z(mz(1)), markersize, [0 0 0], 'v');
h.MarkerFaceColor = [1 0.5 0];
g = scatter3(x(mx(end)), y(my(end)), z(mz(end)), markersize, [0 0 0], 'o');
g.MarkerFaceColor = [0 0.75 0.75];

legend('Marble''s Path', 'Start Point', 'End Point', 'Location', 'West', 'FontSize', 16)

xlabel('x (mm)', 'FontSize', 16)
ylabel('y (mm)', 'FontSize', 16)
zlabel('z (mm)', 'FontSize', 16)

title('Marble''s 3D Path Over the 20 Time Slices', 'FontSize', 32)

axis([-20 20 -20 20 -20 20]), grid on

sp3 = subplot(2,3,4);
u = zeros(size(x)); v=u; w=u;

xvecs = diff(x(mx)); yvecs = diff(y(my)); zvecs = diff(z(mz));
xvecs = [xvecs 0]; yvecs = [yvecs 0]; zvecs = [zvecs 0];

u(mx) = xvecs; v(my) = yvecs; w(mz) = zvecs;

q = quiver(x(mx),y(my),xvecs,yvecs,0,'Color',[0 0 0],'ShowArrowHead','off');

qU = q.UData;
qV = q.VData;
qX = q.XData;
qY = q.YData;


headWidth = 2;
headLength = 3;
LineLength = 1; %0.08;
```

```
for ii = 1:length(qX)-1

    ah = annotation('arrow',...
        'headStyle',arrowstyle,'HeadLength',headLength,'HeadWidth',headWidth);
    set(ah,'parent',gca);
    set(ah,'position',[qX(ii) qY(ii) LineLength*qU(ii) LineLength*qV(ii)]);

end

hold on


k = scatter(x(mx(1)), y(my(1)), markersize, [0 0 0], 'v');
k.MarkerFaceColor = [1 0.5 0];
l = scatter(x(mx(end)), y(my(end)), markersize, [0 0 0], 'o');
l.MarkerFaceColor = [0 0.75 0.75];

legend('Marble''s Path', 'Start Point', 'End Point', 'Location', 'NorthWest')

xlabel('x (mm)', 'FontSize', 16)
ylabel('y (mm)', 'FontSize', 16)
title('Marble''s 2D Path (x,y)', 'FontSize', 24)
axis([-15 15 -15 15]), grid on

sp4 = subplot(2,3,5);

q2 = quiver(x(mx),z(mz),xvecs,zvecs,0,'Color',[0 0 0],'ShowArrowHead','off');

qU = q2.UData;
qW = q2.VData;
qX = q2.XData;
qZ = q2.YData;


for ii = 1:length(qX)-1

    ah = annotation('arrow',...
        'headStyle',arrowstyle,'HeadLength',headLength,'HeadWidth',headWidth);
    set(ah,'parent',gca);
    set(ah,'position',[qX(ii) qZ(ii) LineLength*qU(ii) LineLength*qW(ii)]);

end

hold on
```

```
k = scatter(x(mx(1)), z(mz(1)), markersize, [0 0 0], 'v');
k.MarkerFaceColor = [1 0.5 0];
l = scatter(x(mx(end)), z(mz(end)), markersize, [0 0 0], 'o');
l.MarkerFaceColor = [0 0.75 0.75];

legend('Marble''s Path', 'Start Point', 'End Point', 'Location', 'NorthWest')

xlabel('x (mm)', 'FontSize', 16)
ylabel('z (mm)', 'FontSize', 16)
title('Marble''s 2D Path (x,z)', 'FontSize', 24)
axis([-15 15 -15 15]), grid on



sp21 = subplot(2,3,6);

q3 = quiver(y(my),z(mz),yvecs,zvecs,0,'Color',[0 0 0],'ShowArrowHead','off');

qV = q3.UData;
qW = q3.VData;
qY = q3.XData;
qZ = q3.YData;


for ii = 1:length(qY)-1

    ah = annotation('arrow',...
        'headStyle',arrowstyle,'HeadLength',headLength,'HeadWidth',headWidth);
    set(ah,'parent',gca);
    set(ah,'position',[qY(ii) qZ(ii) LineLength*qV(ii) LineLength*qW(ii)]);

end

hold on


k = scatter(y(my(1)), z(mz(1)), markersize, [0 0 0], 'v');
k.MarkerFaceColor = [1 0.5 0];
l = scatter(y(my(end)), z(mz(end)), markersize, [0 0 0], 'o');
l.MarkerFaceColor = [0 0.75 0.75];

legend('Marble''s Path', 'Start Point', 'End Point', 'Location', 'NorthWest')

xlabel('y (mm)', 'FontSize', 16)
ylabel('z (mm)', 'FontSize', 16)
```

```
title('Marble''s 2D Path (y,z)', 'FontSize', 24)
axis([-15 15 -15 15]), grid on

xend = x(mx(end)), yend = y(my(end)), zend = z(mz(end))
```