Washington University / University of Missouri- St. Louis

Department: Electrical Engineering

Ultrasound Elasticity Imaging

Project Report

Senior Design Project

(JEE4980)

Professor Jason W. Trobaugh, D.Sc

December 16, 2008

Group members

Russo, John

Dake, Emmanuel

Abousamra, Taufik

Table of content

Listing of Figures

Appendix A

Appendix B

Appendix C

Abstract

Nowadays, ultrasound imaging is a not just reduced to looking at images. It can be used to explore motions of tissues, blood flow, detecting unknown areas under the skin of a living human. We researched an add-on tool that computes elasticity of tissues of certain areas from a set of ultrasound frames. This tool could be used for many purposes but most importantly for tumor detection. Tumors in a human body have a special characteristic that is its stiffness. We are taking advantage of the high stiffness level on tumors to find them in an ultrasound set of frames. To do that, we developed an algorithm that calculates the displacement of each point in an ultrasound image relative to previous image taken of a tissue enduring compression. The derivative of the resulting displacement data is taken and that would produce a strain or stiffness set of data that can be presented as an image. This image will highlight the areas where there is a sudden change in the stiffness of the tissues

Introduction

Background

Medical imaging involves the exposition of an object to some form of energy and creating an image as a result of how the input energy interacts with the object.

Ultrasound elasticity is a technique that has been used to detect tumors in tissues. One special characteristic of tumors is that they have a higher stiffness relative to the normal tissue of the human body.

The strain image is the derivative of a displacement image. It is an index of hardness or softness of the region under consideration. By finding the strain indices of all the different layers of a tissue in a human member, a tumor could be detected because of its abnormally high strain index relative to the rest of the tissue.

Problem or need

Finding the strain of a tissue can not be done from just one ultrasound image. We had to collect a set of ultrasound images of the tissue that were taken sequentially while compressing the tissue with the transducer. In this process, the stiffer parts of the tissue will compress less than the softer parts. We had to develop an algorithm that will compute the strain based on a certain set of ultrasound data.

Purpose of report

The purpose is to write a program to calculate displacement and strain from a set of raw ultrasound data using Matlab. We want to see the differences in consistency of displacement and strain to detect any changes in a given substance sample. This technology is applicable in the medical field to detect tumors and other discrepancies in human tissue.

Overview

In this project we wrote program using Matlab to improve Ultrasound Elasticity Imaging. We researched Elasticity Imaging and combed that knowledge with past semesters Matlab code; we then improved the code by writing a Matlab function that improves the speed or clarity of past code. Our code takes the cross correlation of two signals, and then finds the displacement. We then filtered the displacement and took the derivative to obtain our strain.

Methodology:

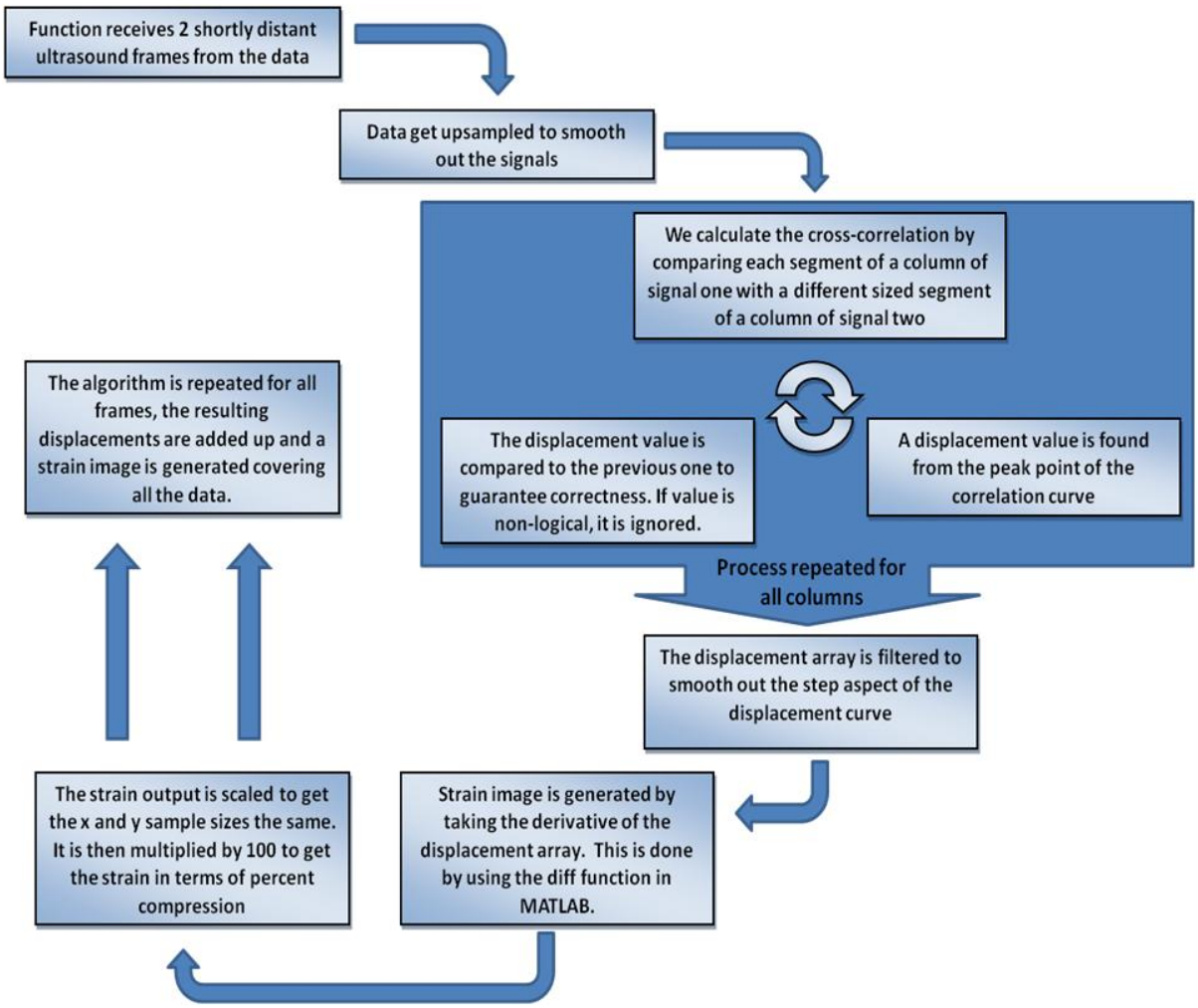
The purpose of this course is to improve upon the codes of previous semesters. There are a number of ways to achieve that goal. Our group chose to write a code from scratch; this will gave us a greater understanding of the project and a better understanding of MATLAB. The advantages of writing our own code are; we can easily debug problems and understand the inner workings of the code. This would be harder to conceptually do if we used past groups' work.

Our code will takes the cross correlation of two signals, and then finds the displacement. After we are satisfied with this displacement, we run the displacement through a filter. Our code then takes the derivative of the filtered displacement to obtain a strain image.

Finally, we compute the percent strain compression. Our code is an improvement upon past groups' work in the following ways: it is a more straight forward, simplistic code, it improves upon the speed of past semester code, and it gives the percent compression of the test substances. The percent compression is a very important part of our code design because it puts the strain data into something quantitative. In the past groups' work, they just compared how the graphs looked, and qualitatively decided if they were on the right track. Putting the strain images into percent compression allows us to see how much the test substance is compressed compared to its original size. This gives us numerical data that is useful, and no other group has done this yet.

Code description:

Our code uses a command we called ‘callcorr2’. The user passes two signals, the lower and upper x-axis values, and the lower and upper y-axis values through callcorr2, and this outputs the displacement. The code takes a signal (Sig1) with a specified window size (r1) and another signal (Sig2) with a different specified window size (r2). These signals are up sampled by five times, and the new signals are called s1 and s2. We then use two for loops to accumulate data for our images. The outer loop gives the y range from y1 (lower bound) to y2 (upper bound), and the inner loop give the x range from x1*up sample rate (lower bound) to x2*up sample rate (upper bound). The x range is incremented in steps of 5*up sample rate; this makes our code run faster, and it causes our images to be smoother in some cases. The rows are incremented by a variable called ‘sample’, and the columns are incremented by a variable called ‘line’. In the inner loop signal one is shifted by the difference in window size by putting zeros in front of the signal. This causes signal one and signal two to line up. Our code then takes the shifted signal one over a range of plus or minus its window size for each x value (r1) and compares it with signal two over a range of plus or minus its window size for each x value (r2). The correlation and lags of these two signals are found running those signals through a function called xcorr; xcorr computes cross correlation between two signals. Our program then tries to find the max correlation by matching up the peaks of the two signals, and records that max correlation along with its max correlation location. The displacement of the line and sample for those signals is the lags of the max correlation locations. We found that there were some very large spikes in our displacement curves because sometimes the xcorr function would match up the wrong peaks. We corrected these spikes, or blips as we called them, by using an if else loop. If a displacement value was plus or minus ten from the previous displacement value, our loop replaced that value with the previous displacement value. After displacement values were taken for every line and sample, we ran those values through a filter. This was important because it made our choppy displacement curve into a continuous curve; we needed a continuous curve so that we could take the derivative of that curve. Our filter used a window size of twenty and added an array of ones divided by the window size to get a moving average over that window size. This moving average smoothed out our displacement curve, and then we got the strain of our data by using a difference function (diff) on the filtered displacement. In order to get the percent compression of the strain, we divided the difference of our filtered displacement by five; this made our x and y sample sizes the same. Then we multiplied this by one hundred to get the strain into a percent. We then summed the strain data to over small increments to get a total strain compression for the image that spanned all of the frames.



Percent Compression:

The percent compression of an object can be calculated by taking the size of the object before it was compressed, subtracting it from the size of the object after it was compressed, and then dividing the whole thing by the original size of the object. We used this method by looking at the tofu videos on the class website. We measured the anomalies before and after compression and used the above technique to find percent compression. We checked to see if our code was working correctly by looking at the displacement curve for a column of data over a small span. The rise over run of the curve is the percent compression in that localized area. There were different slopes for the part of the tofu that had an anomaly and the part that did not, we took the slope of interest and found the percent compression. Assuming that the strain compression was constant, we multiplied the small span by a number that would give us the full data span. The compression from the video the compression from our displacement curve calculations matched, and that is how we knew we were on the right track. After this we summed up the percent strain

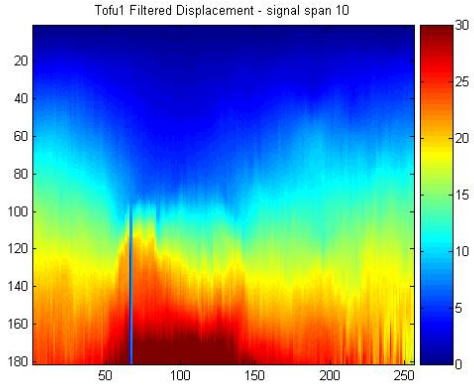
compressions by using a signal span of ten at a time. We would make total displacement equal to the total displacement plus the next span of ten until we covered all one hundred and sixteen frames. At the end, the percent compression of all the frames summed to together equaled what we calculated using the video. The percent compression is helpful because it gives us the ability to quantitatively know how accurate our program is working; this will really take the guess work out of judging the correctness of strain images, and therefore, it improves upon last semester projects.

Results

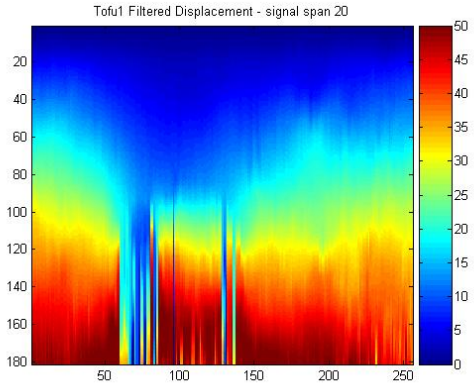
Filtered Displacement and Strain:

The filtered displacements for tofu1 and tofu2 for different signal spans are show below.

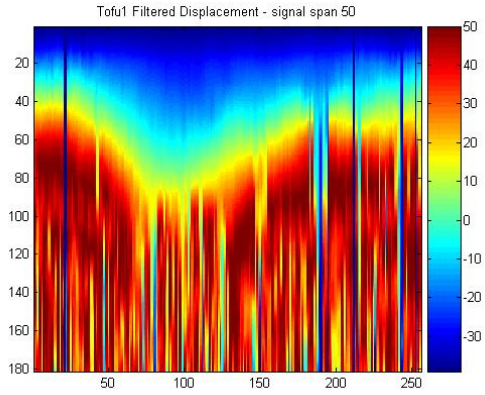
Tofu1:



*Tofu1 Filtered Displacement Signal Span 10

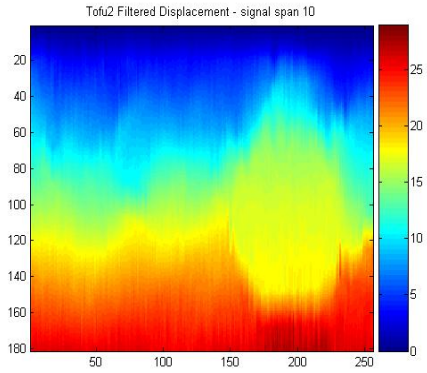


*Tofu1 Filtered Displacement Signal Span 20

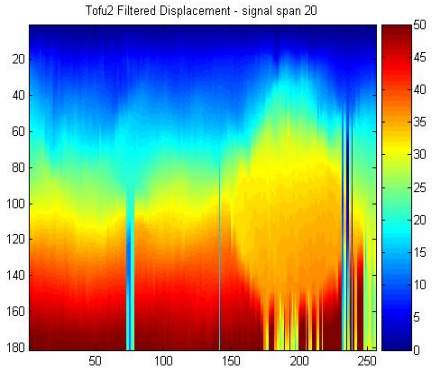


*Tofu1 Filtered Displacement Signal Span 50

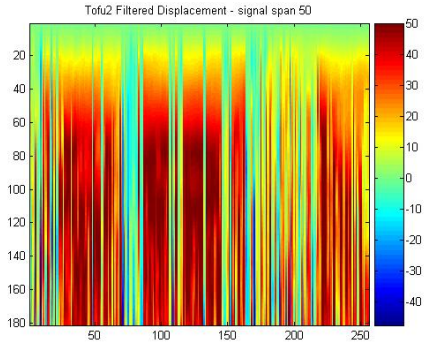
Tofu2:



*Tofu2 Filtered Displacement Signal Span 10



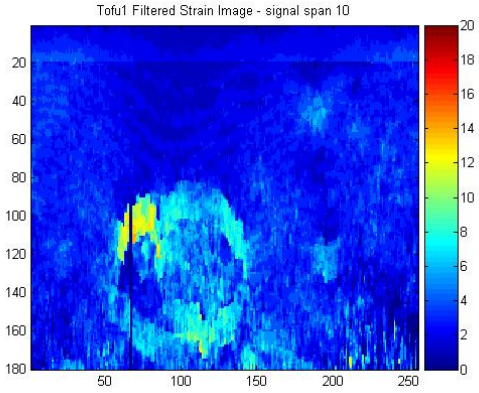
*Tofu2 Filtered Displacement Signal Span 20



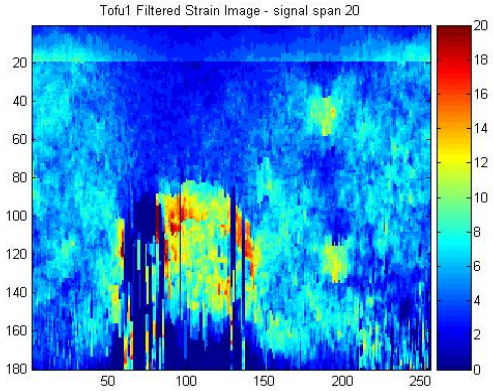
*Tofu2 Filtered Displacement Signal Span 50

The strains for tofu1 and tofu2 for different signal spans are shown below.

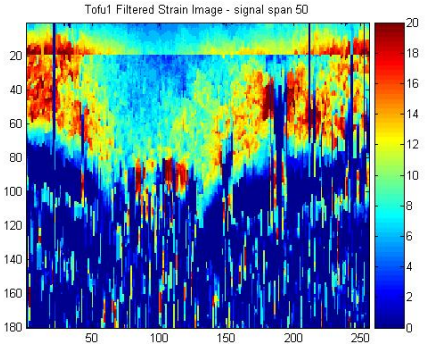
Tofu1:



*Tofu1 Strain Signal Span 10

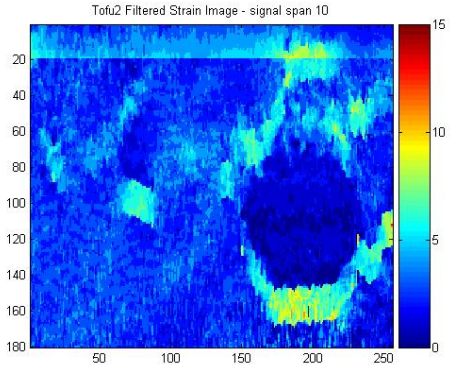


*Tofu1 Strain Signal Span 20

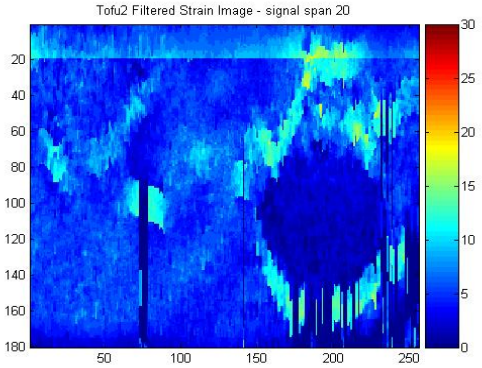


*Tofu1 Strain Signal Span 50

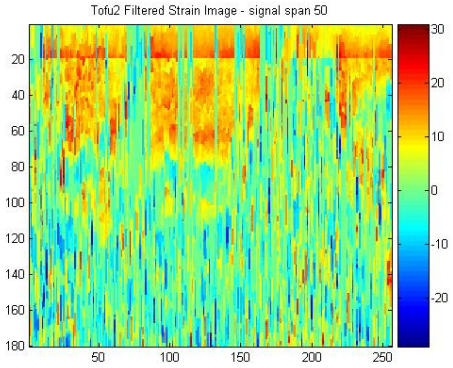
Tofu2:



*Tofu2 Strain Signal Span 10



*Tofu2 Strain Signal Span 20



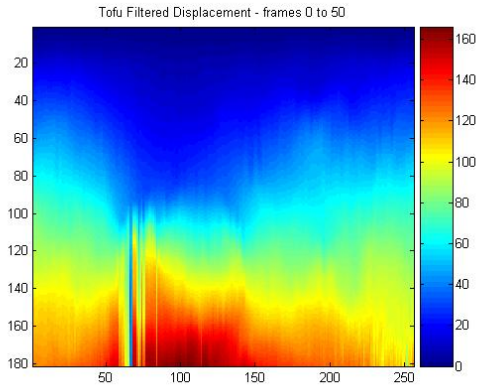
*Tofu2 Strain Signal Span 50

As can be seen, these results get worse and worse as the signal span increases. They are almost unreadable after a signal span of thirty. In order to get an accurate displacement and strain image over all one hundred and sixteen frames of tofu1 and tofu2, we will compound the percent compression strain in small increments to get a final displacement and strain. This is possible because the displacement and strain should, hypothetically, be linear. Therefore, each small span should have the same displacement and strain if their signal spans are equal.

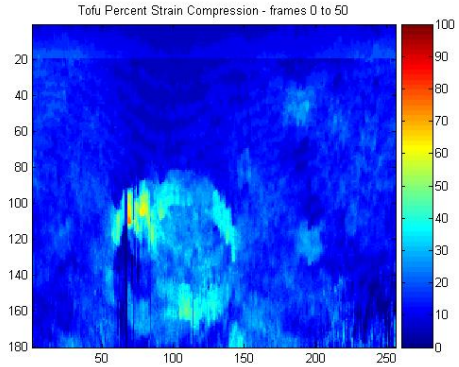
Percent Compression Strain and Filtered Displacement:

The percent compression strains and filtered displacements for tofu1 and tofu2 are shown below.

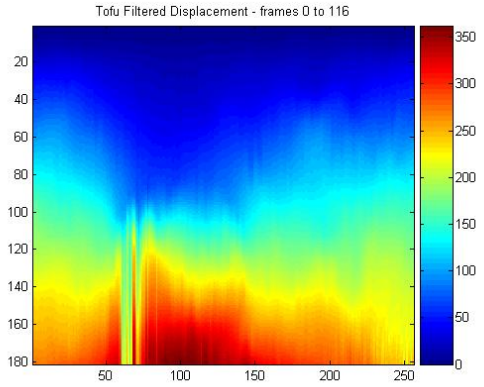
Tofu1:



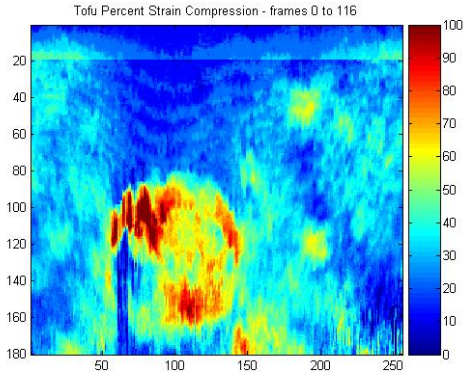
*Tofu1 Filtered Displacement Signal Span of 50



*Tofu1 % Compression Strain Signal Span 50

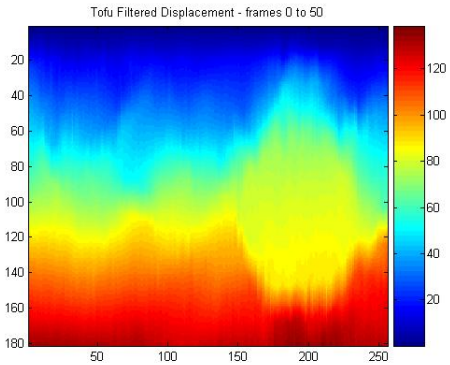


*Tofu1 Filtered Displacement Signal Span of 116

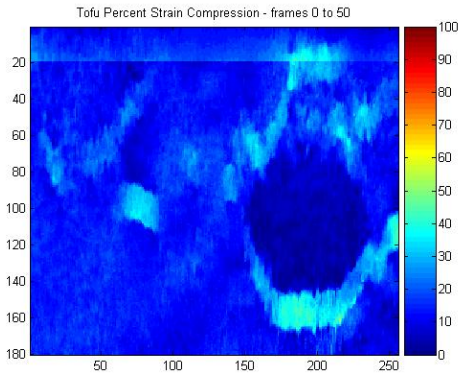


*Tofu1 % Compression Strain Signal Span 116

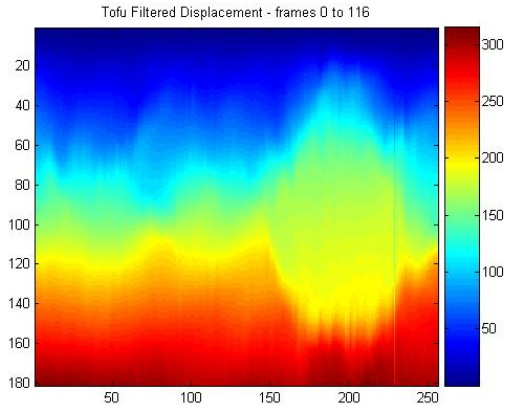
Tofu2:



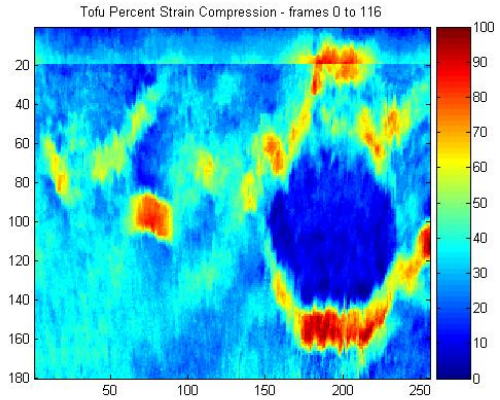
*Tofu2 Filtered Displacement Signal Span of 50



*Tofu2 % Compression Strain Signal Span of 50



*Tofu2 Filtered Displacement Signal Span of 116



*Tofu2 % Compression Strain Signal Span of 116

As you can see, this percent compression method gives much smoother displacement and strain results over a much larger signal span. Also it gives us a quantitative way to look at strain.

Conclusion

In Conclusion, our code has improved upon past semester work by making strain analysis quantitative. Also, we have improved the time of our program; it now runs at about sixteen seconds. These two contributions will make strain easier to analyze over a wide range of data signals, and debugging will take a shorter amount of time. I believe that this will go a long way in advancing ultrasound strain images.

Recommendations

Given More Time:

If we had more time to work on this project we would plot a percent compression ratio of the hard and soft tofu of tofu1 and tofu2. This would further give us quantitative data that shows how the soft compresses in relation to the hard, and vice versa. Another aspect we would have explored if time allowed is the effect different types of filters would have on the displacement and strain. Also, it would be beneficial to work on a normalized cross correlation; I believe this would work a lot better than the xcorr function, and it would help eliminate blips. We didn't have time to apply this percent strain compression technique to other data samples, and if we had more time, we would have looked into using this method over a wide range of test substances.

Appendix A

```
function displacement = callcorr2(Sig1, Sig2, x1, x2, y1, y2);
%x1 and x2 are the upper and lower boundaries of x-axis, must be larger
% than r1 and r2.
%y1 and y2 are the upper and lower boundaries of y-axis
usr = 5; %upsamplerate
r1 = (15*usr);   %region window size for s1
r2 = (25*usr);   %region window size for s2
s1 = resample(Sig1,usr,1); %upsampled signal 1
s2 = resample(Sig2,usr,1); %upsampled signal 2
maxlags = (r2-r1); %range that maxlags will check
[rows, cols]=size(s1); %makes sure that the cols and rows match up
if rows > cols
   in1 = s1'; %inputs 1 and 2 equal signals 1 and 2 inverted
   in2 = s2';
else
   in1 = s1; %inputs 1 and 2 equal signals 1 and 2
   in2 = s2;
end
line = 0;
for y = y1:y2
   sample = 0;       %segment of signal row
   line = line + 1;  %increments the column
   prev = 0;
   for x = x1*usr:usr*5:x2*usr
      sample = sample + 1;
      in1shift = [zeros(1,(r2-r1)) in1(y,x-r1:x+r1)];
      [corr,lags]=xcorr(in1shift,in2(y,x-r2:x+r2),ceil(maxlags));
      [maxcorr,maxcorrlocation] = max(corr);
      displacement(line,sample) = lags(maxcorrlocation);
      current = lags(maxcorrlocation);
      [row, col] = size(corr);
      if current < prev - 10
         current = prev;
      elseif current > prev + 10
         current = prev;
      else
         current = current;
      end
      displacement(line,sample) = current;
      prev = current;
   end
end
windowSize = 20;
FilterDisp = filter(ones(1,windowSize)/windowSize,1,displacement');
Strain = (diff(FilterDisp)/5)*100;  %divide by 5 to get the x and y samples sizes the same, and then
multiple by 100 to get %comp
figure(7)
imagesc(Strain);
colorbar
figure(8)
imagesc(FilterDisp);
colorbar
```

Final Demo Code:

```
 totaldisplacement = callcorr2(b_data000, b_data010, 100, 1000, 1, 256);
figure(1)
imagesc(totaldisplacement')
colorbar
title('Tofu Unfiltered Displacement - frames 0 to 10')
windowSize = 20;
TotalFilterDisp = filter(ones(1,windowSize)/windowSize,1,totaldisplacement');
figure(2)
imagesc(TotalFilterDisp)
colorbar
title('Tofu Filtered Displacement - frames 0 to 10')
PercentStrain = (diff(TotalFilterDisp)/5)*100;
figure(3)
imagesc(PercentStrain, [0 100])
colorbar
title('Tofu Percent Strain Compression - frames 0 to 10')
M(1) = getframe;
%pause

totaldisplacement = totaldisplacement + callcorr2(b_data010, b_data020, 100, 1000, 1, 256);
figure(1)
imagesc(totaldisplacement')
colorbar
title('Tofu Unfiltered Displacement - frames 0 to 20')
windowSize = 20;
TotalFilterDisp = filter(ones(1,windowSize)/windowSize,1,totaldisplacement');
figure(2)
imagesc(TotalFilterDisp)
colorbar
title('Tofu Filtered Displacement - frames 0 to 20')
PercentStrain = (diff(TotalFilterDisp)/5)*100;
figure(3)
imagesc(PercentStrain, [0 100])
colorbar
title('Tofu Percent Strain Compression - frames 0 to 20')
M(2) = getframe;
%pause

totaldisplacement = totaldisplacement + callcorr2(b_data020, b_data030, 100, 1000, 1, 256);
figure(1)
imagesc(totaldisplacement')
colorbar
title('Tofu Unfiltered Displacement - frames 0 to 30')
windowSize = 20;
TotalFilterDisp = filter(ones(1,windowSize)/windowSize,1,totaldisplacement');
figure(2)
imagesc(TotalFilterDisp)
colorbar
title('Tofu Filtered Displacement - frames 0 to 30')
PercentStrain = (diff(TotalFilterDisp)/5)*100;
figure(3)
imagesc(PercentStrain, [0 100])
colorbar
title('Tofu Percent Strain Compression - frames 0 to 30')
```

```
M(3) = getframe;
%pause

totaldisplacement = totaldisplacement + callcorr2(b_data030, b_data040, 100, 1000, 1, 256);
figure(1)
imagesc(totaldisplacement')
colorbar
title('Tofu Unfiltered Displacement - frames 0 to 40')
windowSize = 20;
TotalFilterDisp = filter(ones(1,windowSize)/windowSize,1,totaldisplacement');
figure(2)
imagesc(TotalFilterDisp)
colorbar
title('Tofu Filtered Displacement - frames 0 to 40')
PercentStrain = (diff(TotalFilterDisp)/5)*100;
figure(3)
imagesc(PercentStrain, [0 100])
colorbar
title('Tofu Percent Strain Compression - frames 0 to 40')
M(4) = getframe;
%pause

totaldisplacement = totaldisplacement + callcorr2(b_data040, b_data050, 100, 1000, 1, 256);
figure(1)
imagesc(totaldisplacement')
colorbar
title('Tofu Unfiltered Displacement - frames 0 to 50')
windowSize = 20;
TotalFilterDisp = filter(ones(1,windowSize)/windowSize,1,totaldisplacement');
figure(2)
imagesc(TotalFilterDisp)
colorbar
title('Tofu Filtered Displacement - frames 0 to 50')
PercentStrain = (diff(TotalFilterDisp)/5)*100;
figure(3)
imagesc(PercentStrain, [0 100])
colorbar
title('Tofu Percent Strain Compression - frames 0 to 50')
M(5) = getframe;
%pause

totaldisplacement = totaldisplacement + callcorr2(b_data050, b_data060, 100, 1000, 1, 256);
figure(1)
imagesc(totaldisplacement')
colorbar
title('Tofu Unfiltered Displacement - frames 0 to 60')
windowSize = 20;
TotalFilterDisp = filter(ones(1,windowSize)/windowSize,1,totaldisplacement');
figure(2)
imagesc(TotalFilterDisp)
colorbar
title('Tofu Filtered Displacement - frames 0 to 60')
PercentStrain = (diff(TotalFilterDisp)/5)*100;
figure(3)
imagesc(PercentStrain, [0 100])
```

```
colorbar
title('Tofu Percent Strain Compression - frames 0 to 60')
M(6) = getframe;
%pause

totaldisplacement = totaldisplacement + callcorr2(b_data060, b_data070, 100, 1000, 1, 256);
figure(1)
imagesc(totaldisplacement')
colorbar
title('Tofu Unfiltered Displacement - frames 0 to 70')
windowSize = 20;
TotalFilterDisp = filter(ones(1,windowSize)/windowSize,1,totaldisplacement');
figure(2)
imagesc(TotalFilterDisp)
colorbar
title('Tofu Filtered Displacement - frames 0 to 70')
PercentStrain = (diff(TotalFilterDisp)/5)*100;
figure(3)
imagesc(PercentStrain, [0 100])
colorbar
title('Tofu Percent Strain Compression - frames 0 to 70')
M(7) = getframe;
%pause

totaldisplacement = totaldisplacement + callcorr2(b_data070, b_data080, 100, 1000, 1, 256);
figure(1)
imagesc(totaldisplacement')
colorbar
title('Tofu Unfiltered Displacement - frames 0 to 80')
windowSize = 20;
TotalFilterDisp = filter(ones(1,windowSize)/windowSize,1,totaldisplacement');
figure(2)
imagesc(TotalFilterDisp)
colorbar
title('Tofu Filtered Displacement - frames 0 to 80')
PercentStrain = (diff(TotalFilterDisp)/5)*100;
figure(3)
imagesc(PercentStrain, [0 100])
colorbar
title('Tofu Percent Strain Compression - frames 0 to 80')
M(8) = getframe;
%pause

totaldisplacement = totaldisplacement + callcorr2(b_data080, b_data090, 100, 1000, 1, 256);
figure(1)
imagesc(totaldisplacement')
colorbar
title('Tofu Unfiltered Displacement - frames 0 to 90')
windowSize = 20;
TotalFilterDisp = filter(ones(1,windowSize)/windowSize,1,totaldisplacement');
figure(2)
imagesc(TotalFilterDisp)
colorbar
title('Tofu Filtered Displacement - frames 0 to 90')
PercentStrain = (diff(TotalFilterDisp)/5)*100;
```

```
figure(3)
imagesc(PercentStrain, [0 100])
colorbar
title('Tofu Percent Strain Compression - frames 0 to 90')
M(9) = getframe;
%pause

totaldisplacement = totaldisplacement + callcorr2(b_data090, b_data100, 100, 1000, 1, 256);
figure(1)
imagesc(totaldisplacement')
colorbar
title('Tofu Unfiltered Displacement - frames 0 to 100')
windowSize = 20;
TotalFilterDisp = filter(ones(1,windowSize)/windowSize,1,totaldisplacement');
figure(2)
imagesc(TotalFilterDisp)
colorbar
title('Tofu Filtered Displacement - frames 0 to 100')
PercentStrain = (diff(TotalFilterDisp)/5)*100;
figure(3)
imagesc(PercentStrain, [0 100])
colorbar
title('Tofu Percent Strain Compression - frames 0 to 100')
M(10) = getframe;
%pause

totaldisplacement = totaldisplacement + callcorr2(b_data100, b_data110, 100, 1000, 1, 256);
figure(1)
imagesc(totaldisplacement')
colorbar
title('Tofu Unfiltered Displacement - frames 0 to 110')
windowSize = 20;
TotalFilterDisp = filter(ones(1,windowSize)/windowSize,1,totaldisplacement');
figure(2)
imagesc(TotalFilterDisp)
colorbar
title('Tofu Filtered Displacement - frames 0 to 110')
PercentStrain = (diff(TotalFilterDisp)/5)*100;
figure(3)
imagesc(PercentStrain, [0 100])
colorbar
title('Tofu Percent Strain Compression - frames 0 to 110')
M(11) = getframe;
%pause

totaldisplacement = totaldisplacement + callcorr2(b_data110, b_data116, 100, 1000, 1, 256);
figure(1)
imagesc(totaldisplacement')
colorbar
title('Tofu Unfiltered Displacement - frames 0 to 116')
windowSize = 20;
TotalFilterDisp = filter(ones(1,windowSize)/windowSize,1,totaldisplacement');
figure(2)
imagesc(TotalFilterDisp)
colorbar
```

```
title('Tofu Filtered Displacement - frames 0 to 116')
PercentStrain = (diff(TotalFilterDisp)/5)*100;
figure(3)
imagesc(PercentStrain, [0 100])
colorbar
title('Tofu Percent Strain Compression - frames 0 to 116')
M(12) = getframe;
%pause

figure(4)
movie(M,10,4)
caxis([0 100])
colorbar
```

Appendix B

Members

Russo, John- worked on coding, debugging, researching and presenting

Abousamra, Taufik – worked on coding, debugging, diagrams and presenting

Dake, Emmanuel – worked on debugging, researching and documentation

Appendix C

References

1. http://classes.engineering.wustl.edu/jee4980/
2. A.D. Wilcox, Engineering Design for Electrical Engineers, Prentice Hall, Englewood Cliffs, N.J., 1990.
3. Professor Jason W. Trobaugh, D.Sc