Bradley University
Department of Electrical and Computer Engineering

Acoustic Fossil Imaging

Project Report

By
Matt Kaiser and John Lewis

Advisors
Dr. James H. Irwin
Mr. José Sánchez

May 10, 2004
Abstract

Often fossils are found buried within rocks of similar composition to the fossil. The paper covers a technique for imaging a fossil by nondestructive testing in three dimensions. Basic ultrasonic theory involving material density, and curvature is discussed. Frequency of transmission (50 kHz & 2 MHz) and transmission media are also discussed. Next, there is an in-depth discussion on the hardware needed to produce a usable signal. Discussion continues into two different forms of data acquisition. A web based approached, using a Tektronix® TDS3012B, is discussed in greater detail. MATLAB® is used to analyze and visually represent the data. Limited resolution images were constructed from several data sets ranging from a simple echo return to a multi-layered steel washer. While fossils were not imaged, only gain in the receiver prevented this.
# Table of Contents (In Progress)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Objective</td>
<td>1</td>
</tr>
<tr>
<td>Theory</td>
<td>1</td>
</tr>
<tr>
<td>Measurement System</td>
<td>7</td>
</tr>
<tr>
<td>Hardware</td>
<td>7</td>
</tr>
<tr>
<td>Data Acquisition</td>
<td>9</td>
</tr>
<tr>
<td>Software</td>
<td>10</td>
</tr>
<tr>
<td>Conclusions</td>
<td>17</td>
</tr>
<tr>
<td>Appendix I – Main MATLAB® Function</td>
<td></td>
</tr>
<tr>
<td>Appendix II – Depth MATLAB® Function</td>
<td></td>
</tr>
<tr>
<td>Appendix III – Classification MATLAB® Function</td>
<td></td>
</tr>
<tr>
<td>Appendix IV - Dual-Stage Common Emitter Amplifier</td>
<td></td>
</tr>
<tr>
<td>Appendix V – Original Project Proposal</td>
<td></td>
</tr>
<tr>
<td>Appendix VI – Original Functional Description</td>
<td></td>
</tr>
<tr>
<td>Appendix VII – Original System Block Diagram</td>
<td></td>
</tr>
<tr>
<td>Appendix VIII – Data Sheet</td>
<td></td>
</tr>
<tr>
<td>Appendix IX – Reson Transducer Plots</td>
<td></td>
</tr>
</tbody>
</table>
**Introduction**

The use of acoustics in non-destructive testing (NDT) has many applications. In the area of geology there is a need for NDT of fossils buried in rock. When a geologist suspects there is a fossil buried inside a rock there are several important determinations that need to be made. First, is there actually a fossil inside the rock? If there is then this conclusion can reduce the time that might be spent on rocks that contain no fossils. Second, what is the shape of the fossil? Knowing the shape and size of the fossil is important to fossil recovery, since this information reduces the chance of damaging the fossil.

**Objective**

The objective of the project is to make advancements in non-destructive testing which will allow fossils buried within rocks to be found. Advancement would allow geologists to be more productive and efficient, allowing the possibility of uncovering an above average number of fossils. The significance of the research includes not only preventing chipping and damaging of fossils, but also has the possibility of being useful in such fields as land mine detection.

**Theory**

Acoustic wave propagation is similar to electromagnetic waves on a transmission line. A transmission line can contain multiple loads at different points in the line, where each of these loads creates an impedance mismatch. With every impedance mismatch a reflection occurs. This means that some of the original signal on the line is transmitted back to the source. Acoustic energy works in a very similar fashion. When the acoustic energy is transmitted it propagates through a medium. When there is a change in medium a reflection occurs. This change in medium is analogous to an impedance mismatch on a transmission line.
In air materials were chosen for the reflectivity of acoustical energy. Generally dense and non-porous objects reflected very well. This is due to the impedance mismatch the object had with its medium, the air. To observe the penetrating power of the acoustics in air the group used facial tissue. A metal object was placed inside of a tissue box. A thin layer of tissue was placed across the top of the box. It was demonstrated that the acoustical energy would reflect off the top layer of tissue as well as penetrate the tissue. The acoustic wave would then hit the metal object and return another echo.

![Figure 1 - Tissue & Metal Acoustic Returns](image)

Figure 1 shows the two distinct echoes spaced apart in time. The first echo is from the tissue, and the second echo is from the metal object under the tissue. This acoustical penetration in air worked well for a thin layer of tissue. However, if it is desired to use acoustical waves to penetrate metals or possibly even rock, another medium is necessary.

The Reson transducer (see Appendix IX for specifications) was designed to be used underwater. The water was necessary to provide the proper impedance match with the sensor, and to provide a low loss absorption medium to couple the sound energy into the sample. Once
in water, objects like metals and some plastics would allow acoustical energy to penetrate. Therefore, the mismatch between the metal and water caused a reflection to occur. The metal would also pass the energy through it and into the next object in its path making imaging multiple metal objects very easy. Sand and plaster do not pass the energy as effectively as the metal. The sand is very granular, with very sharp edges, which can cause the acoustical energy to scatter and not return an accurate echo. The sand will often return what appear to be multiple echoes, but there is no object buried beneath the sand. This happens due to the scattering effect of the sand. In Figure 2 there appear to be multiple echoes. There were some instances when there were multiple echoes present. However, there should have only been one. There was no object buried in this sand sample. If there were an object buried within the sand, current software would not be able to distinguish between the sand reflections and the object reflections.

Figure 2 - Scattering Echoes in Sand
Plaster presents another problem. The plaster is a very porous material. When the plaster was submerged in water for the first time, hundreds of air bubbles appeared. It was found that acoustical energy had an extremely tough time penetrating the plaster. The plaster was acting like acoustical foam does in air, it traps the sound waves in its tiny air pockets. In plaster these tiny air pockets bounce the sound around until it is attenuated and provides no real penetrating power. All that could be seen was the front reflection from the plaster. Problems may arise in rocks with scattering as well as other some other difficulties. Another important aspect to acoustic wave reflection is the curvature of the object.

When acoustical waves are dispersed at a relatively flat object they tend to reflect directly back to the source. However, if the object is curved or angled, the waves will be deflected away from the source. It was important to discover how this would affect imaging objects that were not flat. A simple experiment was setup using a 1” ring binder. The binder was used to create an angled surface. With one 50 [kHz] air type transducer placed directly above the binder, acoustic energy was pulsed out. It was determined that at angles of less than five degrees the object could still be detected with one sensor. However, at angles greater than this the object returned no usable echo to the source. To solve this problem a second sensor was used.
As seen in Figure 3, this sensor was spaced a set distance from the pulsing sensor. It was designed to pick up the deflected pulse from the binder. This technique increased the observable angle to 17 degrees. This was an improvement over having just one sensor.

Another interesting experiment was done with two pieces of metal one on top of the other. Figure 4 shows a metal washer (light gray) on top of a metal block (dark gray). The acoustic energy penetrates through both pieces of metal. The return arrows show the echoes that were caused by the initial penetration.
Figure 5 shows the oscilloscope plot of the reflections as measured. This demonstrates the capability to detect very small impedance mismatches even with objects of the same density. The mismatch from one piece of metal to the next will not be that great when they are placed on top of other in this way. The smaller echoes after the main echoes in Figure 5 resulted from echoes that got caught bouncing in the metal and reflected back out as secondary echoes. They are not as strong due to the attenuation from being trapped in the metal. The smaller echo in the green in Figure 5 is most likely a weld on the metal washer, which did not provide as large of an impedance mismatch. The echo in Figure 5 highlighted in light gray is stronger than the previous echo because there was a greater impedance mismatch going between the two metals than there was at the welded junction.
Measurement System

The measurement system consists of an X-Y table, an immersion tank, an immersion transducer, a function generator, and oscilloscope for data acquisition. All of the signal processing is done in MATLAB®. The processing of data is currently done offline, however, using a data acquisition card will allow for real-time data analysis.

Hardware

Initial acoustic work was done using a 50 [kHz] Polaroid transducer in air. This allowed techniques for measuring the distance, size, and shape of an object to be developed. The same principles could then be applied to the immersion transducer.

When the 2 [MHz] Reson transducer was acquired for underwater testing, it came with no supporting hardware. Therefore, it was necessary to implement hardware that could properly drive the transducer. Typically acoustic transducers are complemented by a pulser/receiver. This is a device capable of pulsing a set amount of cycles at a particular frequency. Once the
pulse is sent out, the device listens for a return echo. The echo is then passed through an amplifier before being sent back to the user.

![System Block Diagram](Image)

Figure 6 – System Block Diagram

Figure 6 shows the system block diagram. The pulsing was taken care of with a HP8012A function generator. The function generator was used in burst mode, therefore, it was setup to burst five cycles of 2 [MHz] every 10 [Hz]. By only pulsing every 0.1s this allowed plenty of time for the echo to be returned to the transducer. Five cycle pulses were chosen to create an easily discernable signal.

Once the echo was returned to the transducer, it needed to be amplified to make sure all echoes were seen above the noise threshold. Several amplifiers were attempted before finding one that was acceptable. The first amplifier that was build was a single stage common emitter amplifier. This worked well, but did not provide the desired amplification. Experiments were
done with the cascode and darlington pair amplifiers. Both amplifiers proved difficult to implement. The amplifier design that was chosen was a cascaded common emitter amplifier (see Appendix IV). By cascading the stages the gain is multiplied. The net result was a gain of 45dB at 2 [MHz] (see Appendix IV). This was enough amplification to ensure that the software would be able to detect low amplitude reflections. The only problem encountered with this design was the DC blocking input capacitors. These affected the poles of the amplifier. A value was chosen to ensure that the amplifier operated properly at 2 [MHz].

**Data Acquisition**

The initial data acquisition system used a TDS340 Tektronix® oscilloscope and a personal computer. The TDS340 takes 1,000 data points with individual timestamp per acquisition. The data is transferred between the oscilloscope and the computer through a RS-232 serial communication line. To decipher the data a software package called Wavestar® was downloaded from Tektronix®. This enabled comma separated variable data to be stored into a folder for later signal processing. The move away from this method of data acquisition was due to a time limit on the trial Wavestar® software.

The current method used for acquiring data utilizes the Ethernet. The system uses a Tektronix® TDS3012B oscilloscope with the e*Scope web-based remote control and a personal computer. The data acquisition process starts by letting the oscilloscope acquire its own IP address. The user gains access to the oscilloscope's built-in web control by typing in the proper IP address, user name, and password in an Internet browser. Complete control is granted to the personal computer and data files for a specific oscilloscope channel can then be saved onto the computer. Each data file is saved in comma separated variable format for easy analysis. The files
are saved with the prefix “Data” and the suffix contains the row number and the column number. For example, “Data429.csv” represents the data set of “y” direction four and “x” direction twenty-nine. The data files contain 20,000 data values. Each data file contains 10,000 timestamp values with 10,000 corresponding voltage values. Once all of the data is saved, the processing can then be done in MATLAB®.

**Software**

All of the signal processing is done using MATLAB® offline. For ease of debugging, functions were written modularly. There are three modules of MATLAB® code: the main program, the depth function, and the classification function. Figure 7 shows the flowchart for the main program.

![Figure 7 - Main Software Flowcharts](image-url)

**Figure 7 - Main Software Flowcharts**
The main program first prompts the user to enter in the current temperature. This portion of code needed to be modified when the project switched from air to water transmission. The equation for the speed of sound changed from Equation 1 in air to Equation 2 in water.

\[ 331.3 \text{ [m/s]} + \text{temperature [°C]} \times 0.6 \text{ [m/s/°C]} \]

**Equation 1 - Speed of Sound in Air**

\[ 1403 \text{ [m/s]} + \text{temperature [°C]} \times 3.9 \text{ [m/s/°C]} \]

**Equation 2 - Speed of Sound in Water**

After the temperature is entered, MATLAB® prompts the user to enter in the last number for the data set. For example, if the last data set obtained was “Data529” then the user would enter in “5”, “2”, and “9”. The coding used to do this can be seen in Figure 8 where MATLAB® sees 48 plus a string is equal to ASCII zero.

\[ x = \text{importdata(‘data’,i+48,m+48,n+48,’.csv’)}; \]

**Figure 8 – Importdata Function**

The next feature of the software is optional. The image can be “padded” by zero-valued depth reading to make the image looked smaller to get a better overall view. This feature is currently turned off but has been used in the past. The software then processes the first data set.
The software then checks to make sure that the next data set to be processed does not exceed the bounds of the user entered information. The first data file is then loaded into a variable and checked to make sure that the file is the proper size to ensure the data file was not truncated in the data acquisition routine. A voltage versus time scan, or an A-scan, of the data can then be plotted through MATLAB® if desired. Four example A-scans can been seen in Figure 9.

![Four A-Scans with Three Reflections Each](image)

**Figure 9 - Four A-Scans with Three Reflections Each**

Now that the data set is imported, it is passed to the “depth” function, which determines the height of an object. The depth function can calculate up to three different distinct thicknesses within certain guidelines. One of these guidelines is that the data point must be past a certain timestamp. This is because the initial send pulse from the transducer is large and needs to be thrown away because it tells us nothing. The next guideline is that the software stops reading data before the platform-return. The platform that the object is sitting on has its own timestamp, which needs to be ignored. The programmer does this by entering in the timestamp of the
platform into the depth function to signal the end of the data set. The next guideline is the threshold variable. If the return voltage breaks this threshold then the software has detected an object as can be seen in Figure 10. The depth software flowchart can be found in Figure 11.

![Figure 10 - Typical A-Scan with Threshold Shown in Yellow](image)

![Figure 11 - Depth Function Flowchart](image)
The final guideline only applies to multiple reflection objects. When there is more than one reflection, the side-reflections and residues left from the first echo, could possibly trigger as the second echo. To avoid this from occurring, the multiple echo depth calculations wait for a five microsecond delay. Once a data point can meet all of the guidelines its timestamp is stored into a variable. Then its depth is calculated using Equation 3.

\[
[(\text{speed} \cdot \text{of} \cdot \text{sound} \cdot x \cdot (p/2.0)) - 0.3 - (\text{speed} \cdot \text{of} \cdot \text{sound} \cdot x \cdot (\text{depth} / 2.0))] \\
x = 39.37117874 \text{ [in/m]} = \text{conversion from meters to inches} \\
p= 0.00012107 \text{ [sec]} = \text{platform echo return} \\
\text{depth} = \text{time stamp of echo in seconds} \\
\]

**Equation 3 - Basic Depth Calculation Code**

The 0.3 inches in the above equation represents the correction of the 0.00012107 seconds, the "z" distance of the transducer from the platform. After processing each data point of a single data set, the depth function can return up to three depth values. Each of these values has a timestamp set as a variable that is returned to the main software. After the depth is calculated, the program calls the function “classify” which then places the depth value in the correct place according to a timestamp threshold. See Figure 12 for the complete classify flowchart.
The classify function has set values of where three classes are located in time by a data point number set by the programmer. A range is also set for each class and is currently at 400 data points. For the data set last run this equated out to be 4 microseconds. The code checks the timestamp return of the depth function to see in which class it is located. If the timestamp falls within the range of one of the three classes, then the depth value is placed into the responding array. If the timestamp does not fall in one of the three class ranges, then it is ignored as noise. The arrays of the three image arrays are then sent back to the main program. After this process is done, the software goes through the same steps until there is no more data to be processed.

The end of the main program involves plotting the three dimensional images. This is done using the built-in surface function. The software then plots the three image arrays seen in Figure 13.
The software then plots the three averaged image arrays and the averaged overlain plot as seen in Figure 14. The software only counts non-zero height points in the averaging process. When the height is not zero the software increments a counter and adds to the total sum of the objects depth. To add some better clarity of the profile of the images, a two dimensional plot of the side of the average image arrays overlain is plotted as seen in Figure 15.
Conclusions

The team finished the following tasks: hardware implementation, software implementation, data acquisition system, web site development, and a basic setup. Through these tasks the team was able to image objects with up to three distinct layers containing fairly homogeneous materials. Non-homogeneous materials were not able to be imaged, as well as objects buried beneath sand. Both three dimensional plots and two dimensional plots can now be obtained through the software.

For curved and angled objects more transducers are needed. To be able to penetrate into materials more power needs to be provided to the transducer. Another method of deeper penetration involves lowering the frequency of the transducer. Future research includes adding more power to the transducer, adding an array of transducers, and improving the current setup.
Appendix I – Main MATLAB Routine

% John Lewis last modified 4/27/04
% This code takes in Data scans and process the A-Scans to form 3-D images
% All of the Data sets need to be titled as "data101.csv" and so on.
% The user is asked to enter the last data point in the set as well as the
% temperature.

clear; %Clear out everything before we get started

% Ask user for some information
temp=input('Version 4.22:Enter the Temperature(C):') %Current temperature
%Now ask for the number of data sets
first=input('Enter in the maximum first Data number (e.g. Data101-Data528 enter in 5):')
second=input('Enter in the maximum second Data number:')
third=input('Enter in the maximum third Data number:')

% Temperature calculation
speed_of_sound=331.3+temp*.6; %Equation for the speed of sound with temperature
speed_of_sound=1403+3.9*temp %Equation for the speed of sound in water
lengthtic=0; %Unused counter
close all; %Close out all of those annoying Matlab windows

% set some flags
first2=0; %store time index of "first" echo
second2=0; %store time index of "Second" echo
third2=0; %store
loop_flag=0; %flags to setup classes for waveforms
loop_flag2=0; %flags to setup classes for waveforms
loop_flag3=0; %flags to setup classes for waveforms

% Set even more flags
f=0; %flag for breaking out when we exceed data set
y=0; %I dont think its ever used
i=0; %flag for Most sig data index
n=0; %flag for second most sig data index
m=0; %flag for least sig
a=0; %just a counter for the whole data set

% Used for averaging of the images
sum_boxer=0; %summation of second image
av_boxer=0; %average of second image
count_boxer=0; %count of non-zero numbers of second image
sum_img=0; %summation of first image
av_img=0; %average of first image
count_img=0; %count of non-zero numbers of first image
sum_wow=0;              %summation of third image
av_wow=0;               %average of third image
count_wow=0;            %count of non-zero numbers of third image

%Counter
count=0;
four=10*second+third;       %combine the tens and ones place of the limit of the data set

%Make sure that the 3-D data is cleared
img=zeros(first,four);    %first image
boxer=zeros(first,four);  %second image
wow=zeros(first,four);    %third image

%This is just filler to make things look nice
%It adds arrays of zeros to the front and back end of the data set
% for i=1:5
%   for m=0:9
%     if f==9
%       m=9;
%       f=2;
%       break
%   end
%   else
%     for n=0:9
%       if n==0 & m==0
%         n=1;
%       end
%       deep=0;
%       g=m*10+n;
%       img(i,g)=deep;
%       if m==2 & n==8
%         n=9;
%         f=9;
%       end
%       break
%     end
%   end
% end
% end
% end
%end
%
% for i=11:13
%   for m=0:9
%     if f==9
%       m=9;
%       f=2;
% break
% else
% for n=0:9
%     if n==0 & m==0
%         n=1;
%     end
% deep=0;
% g=m*10+n;
% img(i,g)=deep;
%     if m==2 & n==8
%         n=9;
%         f=9;
%     %
%     break
% end
% end
% end
% end
% end
% end
% end

% Start taking in data
% Start with the most sig at one and go until the end (user entered)
for i=1:first
    for m=0:9  % now move on to the next most sig data fig and go to end
        % Check to see if we are at the end of the data set, if so break out
        if f==9  % this gets set on line on line 226 col 18
            m=9;  % set the second most sig to 9 to make sure we get out
            f=2;  % change the flag to not nine
            break
        end
    else
        for n=0:9  % start at least sig fig
            if n==0 & m==0  % the min for the data set is data101 so make sure n=1;  % that it is never at data100
                end
            a=1+a;  % here is the running count
            x=importdata(['data',i+48,m+48,n+48,:csv']);  % Bring in the data
            r=i*100+m*10+n  % Display "Data" number
            figure(r),plot(x);  % Display A-Scans
            % Make sure the data set is the proper size
            large=size(x);

            % Code to be used to amplify a second return echo
            % if first2>second2
            %    index4=first2;
            % else
            %    index4=second2;
% check to see if the data set is all in place
if large == [10000 2]
    % Call the function "depth" to calculate the depth and to
    % find buried objects
    [lengthtic,distsensmeasure,deep,deept,deepto,distsensorcalc,index,index2,index3] = depth(x,speed_of_sound,lengthtic);
else
    % if there is not enough data in the data set then there has
    % been in error in data acquisition and the depth is set at 0
    'ERROR!!!@',r
    deep=0;
    deept=0;
    deepo=0;
end

% set values of depth into "img" vector
s=1; % store the most sig
% s=s+5; % used if the filler code is used

% now we need to make sure each echo is classified in the proper place
% [img,boxer,wow,loop_flag,loop_flag2,loop_flag3,first2,second2,third2] = classify(s,g,img,boxer,wow,deep,deept,deepto,loop_flag,loop_flag2,loop_flag3,first2,second2,third2,index,index2,index3);

% troubleshooting purposes only
% figure(r),plot(x); % A-Scans can be displayed for

% Review Edge of A-Scans
if (i==2 & m==0 & n==8)
    figure(1),subplot(221),plot(x),title('Data208')
elseif (i==2 & m==0 & n==9)
    figure(1),subplot(222),plot(x),title('Data209')
elseif (i==2 & m==1 & n==0)
    figure(1),subplot(223),plot(x),title('Data210')
elseif (i==2 & m==1 & n==1)
    figure(1),subplot(224),plot(x),title('Data211')
end
%Troubleshooting A-scan
%if (i==2 & m==0 & n==3)
%   figure(100),plot(x),title('Data203')
%end

%%%Find averages of images

%Start out by storing
val=boxer(s,g);
val2=img(s,g);
val3=wow(s,g);

%Start averaging for the second image
%Assume the object does not have holes and only
%Take the average of objects with depth > 0
if val>0
   if g==1 %tricky, the code runs dataX01's twice
      val=val/2.; %because of this divide its value in half
   end
   sum_boxer=val+sum_boxer; %keep a running sum
   count_boxer=count_boxer+1; %keep a counter
end

%Averaging for the first image(same as second)
if val2>0
   if g==1
      val2=val2/2.;
   end
   sum_img=val2+sum_img;
   count_img=count_img+1;
end

%Averaging for the third image(same as second)
if val3>0
   if g==1
      val3=val3/2.;
   end
   sum_wow=val3+sum_wow;
   count_wow=count_wow+1;
end

%We leave the loop
if m==second & n==third
   n=9; %Set the third data variable to 9, the limit
f=9; %Set flag so we leave the loop
break
end
end
end

av_boxer=sum_boxer/count_boxer %Take the average for the second image
av_wow=sum_wow/count_wow %Take the average for the third image
av_img=sum_img/count_img %Take the average for the first image

%final images
figure(2),surf(img),title('First Image + Second Image')
%Plot the second image on top of the first image
hold on, surf(boxer)
figure(3),surf(img),title('First Image')
figure(4),surf(boxer),title('Second Image')
figure(5),surf(wow),title('Third Image')
figure(6),surf(wow),title('All Three Images')
hold on,surf(boxer)
hold on,surf(img)

%Now I would like to plot all of the averaged images
%This is the same as the above code except with averaging features
for i=1:first %Most sig
    for m=0:9 %2nd most sig
        %Check to see if we are at the end of the data set
        if f==9 %again just like above (line 287)
            m=9;
            f=2; %set flag to non-nine
            break
        else
            for n=0:9 %least sig
                if n==0 & m==0 %dataX00 protection
                    n=1;
                end
                g=10*m+n; %DataXYY where g=YY
                s=i; %DataXYY where X=i=s
                %Now go through and pick either average or 0
                %First for the second image
                if boxer(s,g)==0 %if the value is a zero then set
                    boxer2(s,g)=0; %the averaged array to a zero
                else
                    %else
boxer2(s,g)=av_boxer;  %if non-zero then fill the array with avg.
end
%Now for the first image
if img(s,g)==0    %if the value is a zero then set
    img2(s,g)=0;  %the averaged array to a zero
else
    img2(s,g)=av_img;  %if non-zero then fill the array with avg.
end
%Now for the third image
if wow(s,g)==0    %if the value is a zero then set
    wow2(s,g)=0;  %the averaged array to a zero
else
    wow2(s,g)=av_wow;  %if non-zero then fill the array with avg.
end
if m==second & n==third  %Check if we are done or not
    n=9;                  %set flags to break out
    f=9;
    break
end
end
end
end
end

%Plot all of the averaged images
figure(7),surf(img2),title('Smooth First Image');
figure(8),surf(boxer2),title('Smooth Second Image');
figure(9),surf(wow2),title('Smooth Third Image');
figure(10),surf(wow2),title('All Three Smoothed Images')
hold on,surf(boxer2)
hold on,surf(img2)

%Add some 2-D plots
wow2=wow2(first,:);
img2=img2(first,:);
boxer2=boxer2(first,:);
figure(11),plot(wow2,'rd'),title('2-D Plot of all 3 Echos')
hold on, plot(boxer2,'bs')
hold on, plot(img2,'mp')
Appendix II – Depth Function

% John Lewis last modified 4/27/04
% This function gets called to calculate the depth of the object(s)
% [passback]=function(passto)
function [lengthtic,distsensmeasure,deep,deep2,deepto,deepsensorcalc,index,index2,index3] =
  depth(x,speed_of_sound,lengthtic,index4)

N=10000;  %old=1000 new=10,000 This is the number of data points
index=1000000;  %Set starting point for the depth index
index2=1000000;
index3=1000000;
widthtic=0;  %Marker

% Store the depth values
r=0;  %these are the three indices for the depth values
tr=0;
tro=0;% Set markers
u=0;  %flags for cancelling noise
tu=0;
tuo=0;
% Setup some flags
flag=0;  %used to make sure we need to move forward
flag2=0;  %used to make sure we need to move forward again
dead=0;  %used to make sure there is more than noise
dead2=0;  %used as a flag again for noise
delay=10;  %delay time
found_second=0;  %gets set but does nothing

% Code to be used to amplify the second return echo.
% startme=index4-200;
% stopme=index4+200;

% for K=startme:stopme
%  x(K+10000)=x(K+10000)*1.0;
% end

for I=1:N,
  if x(I) > .001 & x(I+10000) >= .124 & x(I) < .00287 %old=1000 new=10,000
    u=u+1;  %check for a threshold three times
  end
end

if u>=3 && flag==0  %if the threshold is broken thrice then we have something
    index=I-3;      %move back three steps in time
    r=x(index);     %capture it
    flag=1;         %set a flag
    %I=10000;     %for old scope=1000 and for new=10,000
    %break
end
end
if flag==1;     %if we have something for a first object...
    %air        delay=x(index)+.0006;
    %wait until the first pulse has ended with delay
        delay=x(index)+.000005;
end
%Now check for a second image!
if x(I) > delay & x(I) < 0.00010986 & x(I+10000) >= .005 & found_second==0
    'I am in the loop!!!'; %troubleshooting
    tu=tu+1;    %again with the noise
    if tu>=3
        dead=1;        %flag means more than noise
        %break
    end
end
%If there is something there besides the noise we plot it
if dead==1 %& x(I+10000) >= .02 & x(I) < .000120 MORE THAN NOISE
    index2=I-3;     %step back thrice
    tr=x(index2);   %capture
    flag2=1;        %set flag
    dead=0;         %set another flag
    found_second=1; %and we dont use this
    %I=10000;
    %break
end
if flag2==1;    %was flag set about 7 lines ago
    %air        delay=x(index)+.0006;
    %wait until the second pulse has ended
        delay=x(index2)+.000005; %delay
end
%Now check for a third image!
if x(I) > delay & x(I) < 0.00010986 & x(I+10000) >= .005 & flag2==1
    'I am in the loop!!!';
    tuo=tuo+1; %noise
    if tuo>=3 %break noise
        dead2=1;    %set flag if we broke the noise
        %break
    end
%If there is something there besides the noise we plot it
if dead2==1 & x(I+10000) >= .005 & x(I) < 0.00010986
    index3=I-3; %step back thrice
    tro=x(index3); %capture
    I=10000; %break out
    break
end
end

%distsensorcalc=r./2.*speed_of_sound*39.37007874;
distsensmeasure=13373.5*(r./2.);

%Calculate first object depth
%make sure there is a value in r
if r~=0
    deep=-
    (speed_of_sound*39.37007874*((r./2.)))+speed_of_sound*39.37007874*(.00012107/2.0)-.3; %8.0349 is the wooden board
elseif r==0
    deep=0;
end

%Calculate second object depth
if tr==0
    deept=0;
elseif tr ~=0
    %deept=(speed_of_sound*39.37007874*(.002075-(tr./2.)))-
    speed_of_sound*39.37007874*(.002075-.00289/2); %8.0349 is the wooden board
    deept=-
    (speed_of_sound*39.37007874*((tr./2.)))+speed_of_sound*39.37007874*(.00012107/2.0)-.23; %8.0349 is the wooden board
end

%Calculate the third object depth
if tro==0
    deepto=0;
elseif tro ~=0
    %deept=(speed_of_sound*39.37007874*(.002075-(tro./2.)))-
    speed_of_sound*39.37007874*(.002075-.00289/2); %8.0349 is the wooden board
    deepto=-
    (speed_of_sound*39.37007874*((tro./2.)))+speed_of_sound*39.37007874*(.00012107/2.0)-.3; %8.0349 is the wooden board
end

%Unused code from here on down!
for I=1:N,
    if x(I) > 0.00003976 & x(I+10000) >= .004 & x(I)<0.00010986 %old=1000 new=10,000
        u=u+1;
        if u>=3
            lengthtic=lengthtic+1;
            l=10000; %for old scope=1000 and for new=10,000
            break
        end
    end
end

return

%Now for width
widthtic=input('Enter the number of echos for the width:')
echo_per_width=input('Enter the number of echos per inch(e.g. 2 or 4):')

width=2*widthtic/(echo_per_inch*distssensmeasure*tan(5.5*pi/180))
clc

%width
%length
%depth
%volume=length*deep*length
Appendix III – Classification Function

%Classification code
%John Lewis 4-27-04
%This code places up to 3 waves in the proper places

function [img, boxer, wow, loop_flag, loop_flag2, loop_flag3, first2, second2, third2] = classify(s, g, img, boxer, wow, deep, deept, deepo, loop_flag, loop_flag2, loop_flag3, first2, second2, third2, index, index2, index3);

%Setup flags for classification
%These are not really needed
flag_1 = 0;
flag_2 = 0;
flag_3 = 0;
flag_4 = 0;
flag_5 = 0;
flag_6 = 0;
flag_7 = 0;
flag_8 = 0;
flag_9 = 0;

%Code that does not yet work to find its own first, second, and third echo
% if index<10000 & index>0 & loop_flag==0
%     first2=index;
%     loop_flag=1;
% end
%
%  if index2<10000 & index2>0 & loop_flag2==0
%     loop_flag2=1;
%     if index2>first2-100 & index2<first2+100  %swap indexes if we are in the first
%     %second2=index;
%     %index range
%     %first2=index2;
%     else
%        second2=index2;
%     'STOP ME HERE!!'
%     end
% end
%
% if index3<10000 & index3>0 & loop_flag3==0;
%     loop_flag3=1;
%     if index3>first2-100 & index3<first2+100;
%     %third2=index;
elseif index3>second2-100 & index3<second2+100;
% third2=index2;
else
third2=index3;
end

Set the first, second, and third wave classes
first2=4900;
second2=5400;
third2=5900;
%third2=6000;
%third2=4900;

%Image storage
%The depth value needs to go into one of the three classes
%This code determines which class it goes into
%There are 9 possibilities

%Setup second image
%check to see if it is in the second wave range
%the 200 is a range number
if index2<second2+200 & index2>second2-200;
    boxer(s,g)=deept;
    flag_2=1;
end
%Check to see if it is in the first wave range
if index<second2+200 & index>second2-200;
    boxer(s,g)=deep;
    flag_3=1;
end
%Check to see if it is in the third wave range
if index3<second2+200 & index3>second2-200;
    boxer(s,g)=deepto;
    flag_6=1;
end

%Setup first image
if index2<first2+200 & index2>first2-200;
    img(s,g)=deept;
    flag_4=1;
end

if index3<first2+200 & index3>first2+200;
   img(s,g)=deepto;
   flag_7=1;
end
if index<first2+200 & index>first2-200;
   img(s,g)=deep;
   flag_1=1;
end

%Setup third image
if index2<third2+200 & index2>third2-200;
   wow(s,g)=deept;
   flag_8=1;
end
if index<third2+200 & index>third2-200;
   wow(s,g)=deep;
   flag_9=1;
end
if index3<third2+200 & index3>third2-200;
   wow(s,g)=deepto;
   flag_5=1;
end
end
Appendix IV – Dual-stage common emitter amplifier schematic and frequency response plot

Figure 1 - Dual-Stage Amplifier

Figure 2 - Amplifier Frequency Response
Bradley University  
Department of Electrical and Computer Engineering

Acoustic Fossil Imaging

Project Proposal

By  
Matt Kaiser and John Lewis

Advisors  
Dr. James H. Irwin  
Mr. José Sánchez

December 11, 2003
Project Summary
The acoustic imaging system will consist of three major components: the ultrasonic sensor, sensor controller, and personal computer. The ultrasonic sensor will transmit and capture the sound wave while the personal computer will display an image on a computer screen. The block diagram for this system can be seen in Figure 1. The personal computer will use MATLAB to process the signals provided by the ultrasonic sensor. The goal of the project is to be able to image fossils buried within a rock on the PC screen. The key component of the imaging system will be the ultrasonic sensor.

Figure 1: High-Level System Block Diagram

Function of System Inputs and Outputs
Ultrasonic Sensor/Controller
A high frequency ultrasonic sensor will be used to image the objects. An immersion type transducer will be used for the ultrasonic transmitter. The benefit of imaging an object in water instead of air is that an immersion type transducer is impedance matched to water. The sensor will have to operate at a frequency of around one megahertz in order to properly propagate and penetrate the water and sand. Objects will return echoes that will be captured with a receiver. The sensor will scan the tank using a control mechanism.

Data Acquisition
After the ultrasonic sensor takes readings the data will be sent to a personal computer for data manipulation. An oscilloscope will be used in order to bring the sensor data into the personal computer. Once the data is in the personal computer, MATLAB will be used to perform the signal processing. Measuring the time delay of the echoes received by each receiver will allow the object(s) position to be determined. The strength of the echoes will determine that there is some object buried beneath the sand. The processed information will then be displayed
graphically via the personal computer monitor. Figure 2 shows the data acquisition and computation routine.

![Figure 2: Data Acquisition and Computation Routine](image)

**System Level Block Diagram**

![Figure 3: Overall Block Diagram](image)
Sub-System Level Block Diagrams

Figure 4: Sensor Block Diagram

1) Acoustic Sensor
   - Input: Signal generator set at 10Hz TTL waveform
   - Output: Acoustical pulse with return echo
   - Description: A device that sends a burst of 16 50 [kHz] acoustical pulses

2) Acoustical Foam
   - Input: Acoustical pulse with return echo
   - Output: Filtered acoustical pulse with return echo
   - Description: The acoustical foam narrows the dispersion angle of the sensor to provide more accurate results. This blocks faint echoes from the surroundings.

Figure 5: Oscilloscope Block Diagram

3) Oscilloscope
   - Input: Filtered acoustical pulse with return echo from sensor
   - Output: Acoustical data plotted to the screen
   - Description: The oscilloscope provides a means to view the data graphically

4) Format Output
   - Input: Acoustical data from the oscilloscope screen
   - Output: Raw data in comma separated variable form
   - Description: This allows the graphical data to be put into a form that can be imported to a PC with MATLAB
5) Web Interface (e*scope)
   Input: Raw data in comma separated variable form
   Output: Raw data in comma separated variable form
   Description: The web interface allows the data to be moved via ethernet. This saves the user from having to store the data to multiple disks.

6) PC with MATLAB
   Input: Raw data in comma separated variable form
   Output: a) Visual display of the object in three dimensions on the screen  
           b) Basic dimensions of the object for user information
   Description: The signal processing will allow the user to see the object in three dimensions and will give other useful data about the object (e.g. length, width, and height)

7) Import Data
Input: Raw data in comma separated variable form
Output: Raw data stored in a two dimensional array
Description: This converts the data into a useable form for MATLAB manipulation

8) Correlate Data

Input: Two dimensional array with raw data
Output: Correlated data
Description: Cross correlating the data, from the transmit echo and return echo, allows the signals to be analyzed to determine the location of multiple echoes within one return pulse

9) Dimension Determination

Input: Correlated data
Output: Length, width, and height of a specified object or group of objects
Description: Using time delays and the amount of returned echoes the dimensions of an object can be determined

10) 3D Plot

Input: Object dimensions
Output: A graphical representation of the object in three dimensional space
Description: This allows the user to view a specific object in three dimensions

### Specification sheet

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Supply Voltage</td>
<td>4.5</td>
<td>5</td>
<td>6.8</td>
<td>V</td>
</tr>
<tr>
<td>Input Frequency</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>Hz</td>
</tr>
<tr>
<td>Input Voltage</td>
<td>0</td>
<td>-</td>
<td>4</td>
<td>V</td>
</tr>
<tr>
<td>Transmit Frequency</td>
<td>-</td>
<td>49.4</td>
<td>-</td>
<td>kHz</td>
</tr>
<tr>
<td>Transmit Pulses</td>
<td>-</td>
<td>16</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Transmit Sensitivity</td>
<td>110</td>
<td>-</td>
<td>-</td>
<td>dB</td>
</tr>
<tr>
<td>Receive Sensitivity</td>
<td>-42</td>
<td>-</td>
<td>-</td>
<td>dB</td>
</tr>
<tr>
<td>Operating Temperature</td>
<td>0</td>
<td>40</td>
<td>40</td>
<td>°C</td>
</tr>
<tr>
<td>Operating Distances</td>
<td>.5</td>
<td>-</td>
<td>35</td>
<td>Ft</td>
</tr>
<tr>
<td>Vertical Directivity Pattern*</td>
<td>0</td>
<td>-</td>
<td>60</td>
<td>°</td>
</tr>
</tbody>
</table>

*This directivity pattern is for the current sensor and is extremely large. Future sensors will have a smaller pattern.

### User Interface

In the current interface, the user must move the sensor by hand. An oscilloscope then records the data. The user must then save the data in comma separated variable format using Tektronix web interface software. Once all of the data is collected, the user must then run a MATLAB m-file to
extract and manipulate the data. Future plans include using a controls mechanism for better accuracy and also to eliminate the need for the user to move the sensor by hand.

**Schedule**

<table>
<thead>
<tr>
<th>Date</th>
<th>Planned Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 27</td>
<td>Implement new sensor</td>
</tr>
<tr>
<td>February 3</td>
<td>Work on better test platform</td>
</tr>
<tr>
<td></td>
<td>Work on sensor hardware</td>
</tr>
<tr>
<td>February 10</td>
<td>Acquire data and process it</td>
</tr>
<tr>
<td>February 17</td>
<td>Work on more complex calculation routines (e.g. Cross-Correlation)</td>
</tr>
<tr>
<td>February 24</td>
<td>Attempt to image objects buried with objects</td>
</tr>
<tr>
<td>March 2</td>
<td>Continue complex imaging</td>
</tr>
<tr>
<td>March 9</td>
<td>Possibly implement more sensors if necessary and increase resolution</td>
</tr>
<tr>
<td>March 16</td>
<td>Spring Break</td>
</tr>
<tr>
<td>March 23</td>
<td>Testing</td>
</tr>
<tr>
<td>March 30</td>
<td>Testing</td>
</tr>
<tr>
<td>April 6</td>
<td>Final software and hardware implementations and testing</td>
</tr>
<tr>
<td>April 13</td>
<td>Make adjustments to improve system</td>
</tr>
<tr>
<td>April 20</td>
<td>Prepare for Expo</td>
</tr>
<tr>
<td>April 27</td>
<td>Prepare final presentation</td>
</tr>
<tr>
<td>May 4</td>
<td>Finishing touches on documentation</td>
</tr>
<tr>
<td>Late May</td>
<td>Present to ASA in New York</td>
</tr>
</tbody>
</table>

**Parts**

The project requires some form of data acquisition, which is currently a Tektronix oscilloscope. Also in constant use is a Hewlett Packard single power supply and a Krohn-Hite function generator. Currently, a Panasonic ultrasonic transducer is being used. This will change soon but no transducer(s) has been selected as of now.

**References**


Appendix VI

Acoustic Fossil Imaging

Functional Description
By
Matt Kaiser and John Lewis

Advisors: Dr. James H. Irwin and Mr. José Sánchez

Project Description
The acoustic imaging system will consist of three major components. The ultrasonic sensor, sensor controller, and PC will be essential in the imaging system. The sensor and sensor controller will be inputs to the system. The PC using MATLAB software will be the output. The block diagram for this system can be seen in Figure 1. MATLAB will be used to do all of the signal processing. With all of this hardware, the goal is to be able to image fossil buried within a rock. However, preliminary work will be done using a fish tank with sand at the bottom. Objects will be buried in the sand, and the imaging system will attempt to locate the objects. Besides just locating the object, the object will be imaged. This implies that an accurate picture of the buried object can be portrayed on the PC screen. One of the key components to the system is the ultrasonic sensor.

Function of System Inputs and Outputs

Ultrasonic Sensor/Controller

A high frequency ultrasonic sensor will be used to attempt to image the objects. The sensor will have to operate at a high frequency in order to properly propagate and penetrate the water and sand. The transmitter will return echoes that can be measured with a receiver. The receiver can either be mounted with the sensor or as a separate unit. Also, multiple receivers can be used for better triangulation. The transmitter and receiver(s) can be mounted at fixed points for data acquisition. It is also possible that the sensor could scan up and down the tank using a control mechanism.

Data Acquisition

The data will be sent to a PC for data manipulation. A data acquisition card will be acquired in order to bring the sensor data in the PC. Once the data is in the PC, MATLAB can be used to do the signal processing. There are multiple ways to process the data. Measuring the time delay of the echoes received by each receiver will allow the objects position to be determined. The strength of the echoes will determine that there is some object buried beneath the sand. The processed information will then be displayed graphically via the PC monitor.
Figure 3: High-Level System Block Diagram
Project Description
The goal of the project is to be able to image fossils in rock. At the beginning, the group will investigate imaging objects in air and underwater. The project will be accomplished using an acoustic sensor along with data acquisition and signal processing tools. Figure 1 represents the high-level block diagram for the system.

Sub-System Level Block Diagrams
The overall system block diagram, in figure 1, can be broken down into smaller sub-system block diagrams. The acoustic sensor can be broken down into two separate blocks, see figure 2. The oscilloscope block can also be broken down into two blocks, see figure 3. The PC subsystem and the software flow chart can be seen respectively in figure 4 and figure 5. The software flow chart, in figure 5, is to be used in order to digitally process the data in MATLAB.

1) Acoustic Sensor
Input: Signal generator set at 10Hz TTL waveform
Output: Acoustical pulse with return echo
Description: A device that sends a burst of 16 50 [kHz] acoustical pulses

2) Acoustical Foam

Input: Acoustical pulse with return echo
Output: Filtered acoustical pulse with return echo
Description: The acoustical foam narrows the dispersion angle of the sensor to provide more accurate results. This blocks faint echoes from the surroundings.

3) Oscilloscope

Input: Filtered acoustical pulse with return echo from sensor
Output: Acoustical data plotted to the screen
Description: The oscilloscope provides a means to view the data graphically

4) Format Output

Input: Acoustical data from the oscilloscope screen
Output: Raw data in comma separated variable form
Description: This allows the graphical data to be put into a form that can be imported to a PC with MATLAB

5) PC with MATLAB

Input: Raw data in comma separated variable form
Output: a) Visual display of the object in three dimensions on the screen
b) Basic dimensions of the object for user information
Description: The signal processing will allow the user to see the object in three dimensions and will give other useful data about the object (e.g. length,
6) Import Data

Input: Raw data in comma separated variable form  
Output: Raw data stored in a two dimensional array  
Description: This converts the data into a useable form for MATLAB manipulation

7) Correlate Data

Input: Two dimensional array with raw data  
Output: Correlated data  
Description: Cross correlating the data, from the transmit echo and return echo, allows the signals to be analyzed to determine the location of multiple echoes within one return pulse

8) Dimension Determination

Input: Correlated data  
Output: Length, width, and height of a specified object or group of objects  
Description: Using time delays and the amount of returned echoes the dimensions
of an object can be determined

9) 3D Plot

<table>
<thead>
<tr>
<th>Input:</th>
<th>Object dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>A graphical representation of the object in three dimensional space</td>
</tr>
<tr>
<td>Description:</td>
<td>This allows the user to view a specific object in three dimensions</td>
</tr>
</tbody>
</table>
## Appendix VIII – Data Sheet

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Amplifier</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power Supply Voltage</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>V</td>
</tr>
<tr>
<td>Gain</td>
<td>-</td>
<td>45</td>
<td>-</td>
<td>dB</td>
</tr>
<tr>
<td><strong>Transducer</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmit Frequency</td>
<td>1.8</td>
<td>2</td>
<td>2.2</td>
<td>MHz</td>
</tr>
<tr>
<td>Transmit Pulses</td>
<td>-</td>
<td>5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pulse Repetition</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>Hz</td>
</tr>
<tr>
<td>Transmit Sensitivity</td>
<td>-</td>
<td>187</td>
<td>188</td>
<td>dB</td>
</tr>
<tr>
<td>Receive Sensitivity</td>
<td>-</td>
<td>-207</td>
<td>-204</td>
<td>dB</td>
</tr>
<tr>
<td>Operating Temperature</td>
<td>2</td>
<td>-</td>
<td>35</td>
<td>°C</td>
</tr>
<tr>
<td>Operating Distance</td>
<td>-</td>
<td>-</td>
<td>700</td>
<td>m</td>
</tr>
<tr>
<td>Vertical Directivity Pattern</td>
<td>0</td>
<td>2.2</td>
<td>-</td>
<td>°</td>
</tr>
<tr>
<td>Input Power (1% D.C.)</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>W</td>
</tr>
<tr>
<td><strong>Data Acquisition</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Points Per Sample</td>
<td>-</td>
<td>10,000</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Software</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample Reference</td>
<td>101</td>
<td>-</td>
<td>999</td>
<td>-</td>
</tr>
</tbody>
</table>
Appendix IX – Reson Transducer Plots

**Figure 9 - Directivity Pattern of Reson Transducer**

**Figure 10 - Receiving Sensitivity of Reson Transducer**
**Figure 11 - Transmitting Sensitivity of Reson Transducer**

**Figure 12 - Impedence of Reson Transducer**