EE 419 - Project 5

**The Hunt for Red October**

|  |
| --- |
| http://2.bp.blogspot.com/-yEo84M6AATM/USLgSVexuWI/AAAAAAAAHWY/omZRJOv4xnw/s1600/bloghuntoctober_4.jpghttps://tse2.mm.bing.net/th?id=OIP.M38731be086efa42c2e565f6fc901c380o0&pid=15.1http://cdn.hark.com/images/000/377/232/377232/original.jpghttps://s-media-cache-ak0.pinimg.com/736x/b4/ee/27/b4ee2792d87bffc62f3decc3063adf0d.jpg |

|  |  |
| --- | --- |
| **Names: Aiku Shintani & Chris Adams** | **Lab Date: 2/5/19** |
| **Bench #: 9** | **Section: 2** |

**Learning Objectives:**

**Use Matlab to process discrete-time signals, including:**

* **Correlation Signal Detection in the Presence of Noise**
* **Template Pattern Matching by Correlation Peak Amplitude**
* **Waveform Envelop from Hilbert Transform**
* **Time-of-Flight Determination from Correlation Peak Position**

**1) Evaluate The Available Data**

**Waveforms and Frequency Spectra of the Sonar Transmit Pulse and Potential Targets:.**

* 1. **Time-domain Waveforms** for all four example signals (one per subplot).

****

* 1. **DFT Frequency Spectrum** (magnitude only) for all four signals (one per subplot).

**

* Describe the differences that you see between the waveforms and spectra of the different targets.

**The frequency responses of the the sonar pulse, Typhoon sub, Akula sub, and LA sub all depict two peaks (centered around the same frequencies) which contain the majority of the energy of the signals. The peaks of the sonar pulse are uniform in their shape and resemble a delta type shape the most. The peaks of the LA sub response are similar to that of the sonar pulse except the shape is not as uniform (can visibly see when zoomed in). The peaks of the Typhoon sub response exhibit a slight dip in the magnitude in the center. The peaks of the Akula sub response is the least similar to the rest of the responses. At the center its peaks, the Akula sub response shows a large dip where the magnitude response goes to zero.**

* Which submarine has the “most stealthy” sonar signature (most difficult to distinguish from a rock)? Justify your choice.

**The LA sub has the “most stealthy” sonar signature as its magnitude response closely follows the shape of the response of the sonar pulse. The sonar signature of a solid rock formation (or wall) resembles that of the original transmitted pulse since the signal is reflected strongly. The LA sub can be considered the “most stealthy” since it could be mistaken by a DSP algorithm as a rock formation.**

* Matlab code that created the figures:

% load sonar pulse data

xn = TxPulse;

Mx = length(TxPulse);

% load submarine echo data

yn\_1 = TyphoonSubEcho; yn\_2 = AkulaSubEcho; yn\_3 = LosAngelesSubEcho;

My\_1 = length(yn\_1); My\_2 = length(yn\_2); My\_3 = length(yn\_3);

% plot time-domain waveforms

fs = 50e3; %sampling freq.

T = 0.05; %sapling period

t\_x = 0:T/(Mx-1):T; %acquire time axis for sonar pulse

t\_yn\_1 = 0:T/(My\_1-1):T; t\_yn\_2 = 0:T/(My\_2 - 1):T; t\_yn\_3 = 0:T/(My\_3 - 1):T; %acquire time axis for echo signals

figure(1)

subplot(4, 1, 1)

plot(t\_x, xn)

xlabel('Time (s)')

ylabel('Magnitude')

title('Time-domain Waveform: Sonar Pulse')

hold on

subplot(4, 1, 2)

plot(t\_yn\_1, yn\_1)

xlabel('Time (s)')

ylabel('Magnitude')

title('Time-domain Waveform: Typhoon Sub Echo')

subplot(4, 1, 3)

plot(t\_yn\_2, yn\_2)

xlabel('Time (s)')

ylabel('Magnitude')

title('Time-domain Waveform: Akula Sub Echo')

subplot(4, 1, 4)

plot(t\_yn\_3, yn\_3)

xlabel('Time (s)')

ylabel('Magnitude')

title('Time-domain Waveform: LA Sub Echo')

%take and then plot DFTs of each of the time domain signals

Fd\_x = (0:Mx-1)/Mx;

Fd\_y\_1 = (0:My\_1-1)/My\_1; Fd\_y\_2 = (0:My\_2-1)/My\_2; Fd\_y\_3 = (0:My\_3-1)/My\_3;

%acquire analog frequencies from digital ones

fa\_x = Fd\_x\*fs;

fa\_y\_1 = Fd\_y\_1\*fs; fa\_y\_2 = Fd\_y\_2\*fs; fa\_y\_3 = Fd\_y\_3\*fs;

fftx = fft(xn); %take the fft of the sonar pulse

fft\_y\_1 = fft(yn\_1); fft\_y\_2 = fft(yn\_2); fft\_y\_3 = fft(yn\_3); %fft of echo signals

figure(2)

subplot(4, 1, 1)

plot(fa\_x, abs(fftx)/Mx)

xlabel('Sample Index')

ylabel('DFT Magnitude')

title('Analog Frequency Frequency Response: Sonar Pulse')

hold on

subplot(4, 1, 2)

plot(fa\_y\_1, abs(fft\_y\_1)/My\_1)

xlabel('Sample Index')

ylabel('DFT Magnitude')

title('Analog Frequency Response: Typhoon Sub Echo')

subplot(4, 1, 3)

plot(fa\_y\_2, abs(fft\_y\_2)/My\_2)

xlabel('Sample Index')

ylabel('DFT Magnitude')

title('Analog Frequency Response: Akula Sub Echo')

subplot(4, 1, 4)

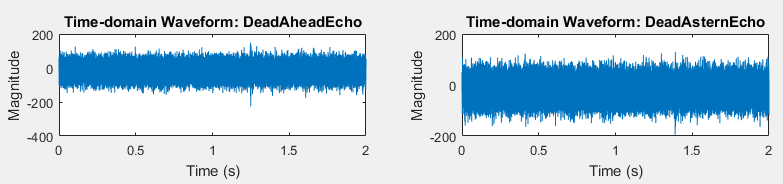
plot(fa\_y\_3, abs(fft\_y\_3)/My\_3)

xlabel('Sample Index')

ylabel('DFT Magnitude')

title('Analog Frequency Response: LA Sub Echo')

**Sonar Echo Waveforms From Two of the Eight Directions:**



* Describe the waveforms (both plotted data shapes and sounds), and whether or not you are able to **visually** spot or **audibly** hear any of the expected target echo waveforms in any of these longer sonar echo waveforms.

|  |  |  |  |
| --- | --- | --- | --- |
| **Direction** | **Relative Bearing**  **(degrees)** | **Visual (Waveform) Results** | **Audible (Listening) Results** |
| **Number of Targets and Distance to Each Target Seen**  (*nearby / mid range / far away*) | **Number of Targets and Distance to Each Target Heard**  (*nearby / mid range / far away*) |
| **Forward** | **0** | **Far Away** | **Far Away** |
| **Starboard Bow** | **45** | **Mid range** | **Mid range** |
| **Starboard Beam** | **90** | **Mid range** | **Mid range** |
| **Starboard Quarter** | **135** | **Mid range** | **Mid range** |
| **Aft** | **180** | **Far away** | **Far away** |
| **Port Quarter** | **225** | **Far away** | **Far away** |
| **Port Beam** | **270** | **None** | **None** |
| **Port Bow** | **315** | **Nearby** | **Nearby** |

* Matlab code that created the examples.

fs = 50e3; %sampling freq.

T = 2; %sapling period

% load echo data %get length %get x-axis

x1 = DeadAheadEcho; Mx1 = length(DeadAheadEcho); t\_x1 = 0:T/(Mx1-1):T;

x2 = StarboardBowEcho; Mx2 = length(StarboardBowEcho); t\_x2 = 0:T/(Mx2-1):T;

x3 = StarboardBeamEcho; Mx3 = length(StarboardBeamEcho); t\_x3 = 0:T/(Mx3-1):T;

x4 = StarboardQuarterEcho; Mx4 = length(StarboardQuarterEcho); t\_x4 = 0:T/(Mx4-1):T;

x5 = DeadAsternEcho; Mx5 = length(DeadAsternEcho); t\_x5 = 0:T/(Mx5-1):T;

x6 = PortQuarterEcho; Mx6 = length(PortQuarterEcho); t\_x6 = 0:T/(Mx6-1):T;

x7 = PortBeamEcho; Mx7 = length(PortBeamEcho); t\_x7 = 0:T/(Mx7-1):T;

x8 = PortBowEcho; Mx8 = length(PortBowEcho); t\_x8 = 0:T/(Mx8-1):T;

% plot time-domain waveforms

figure(1)

subplot(4, 2, 1)

plot(t\_x1, x1)

xlabel('Time (s)')

ylabel('Magnitude')

title('Time-domain Waveform: DeadAheadEcho')

hold on

subplot(4, 2, 3)

plot(t\_x2, x2)

xlabel('Time (s)')

ylabel('Magnitude')

title('Time-domain Waveform: StarboardBowEcho')

subplot(4, 2, 5)

plot(t\_x3, x3)

xlabel('Time (s)')

ylabel('Magnitude')

title('Time-domain Waveform: StarboardBeamEcho')

subplot(4, 2, 7)

plot(t\_x4, x4)

xlabel('Time (s)')

ylabel('Magnitude')

title('Time-domain Waveform: StarboardQuarterEcho')

subplot(4, 2, 2)

plot(t\_x5, x5)

xlabel('Time (s)')

ylabel('Magnitude')

title('Time-domain Waveform: DeadAsternEcho')

subplot(4, 2, 4)

plot(t\_x6, x6)

xlabel('Time (s)')

ylabel('Magnitude')

title('Time-domain Waveform: PortQuarterEcho')

subplot(4, 2, 6)

plot(t\_x7, x7)

xlabel('Time (s)')

ylabel('Magnitude')

title('Time-domain Waveform: PortBeamEcho')

subplot(4, 2, 8)

plot(t\_x8, x8)

xlabel('Time (s)')

ylabel('Magnitude')

title('Time-domain Waveform: PortBowEcho')

**2) Correlation Detection Function**

function Cxy = NormCrossCorrelate(xn, yn)

Where: rxy[k] = unnormalized Cross correlation between signals x, y

E = energy in each signal = sum of the squared signal values

rxx[0] = Auto-correlation of signal x at the 0 lag position = sum of the squared signal values

**Functional Requirements:**

* Your program should properly handle any length signals xn and yn.
* To run as fast as possible, you function should compute the cross-correlation using FFT’s (and zero padding to ensure linear cross-correlation (instead of circular cross-correlation) and to make the FFTs as fast as possible - similarly to how you carried out fast convolution in a previous lab). Do NOT use the Matlab xcorr( ) function!
* You do not need to break up long sequences into shorter blocks using overlap-add or overla-save block processing. Simply cross-correlate by processing both full signals at once.
* Be sure to normalize the cross-correlation results as shown in the equation above.
* Be sure to remove any extra length in your Cxy output from zero padding.
* Your function should plot the two input signals (xn, yn) and the Cxy results vs. index (3 subplots) in a single figure.

**Testing:** NormCrossCorrelate(XCorrTest, TxPulse)

1. **Test case results**: provide the plot created by your function for the specified test case, and interpret the results:



* 1. What was the peak cross-correlation value?

**Peak cross-correlation is 0.9999 ~ 1 (normalized)**

* 1. Where did it occur? **Index #: 46667 Time (sec): 0.93334sec**
  2. Was this peak value what you expected? **Yes, since cross correlation with itself is autocorrelation which has a peaked normalized value of 1. The fact that the peak is at 1 indicates that the pulse shapes are identical.**

1. **Explain your selection for the FFT size** used in your FFT signal processing: Why did you use the particular value chosen?

**FFT size was taken as the next power 2 of the sum of the number of samples of the inputs. This way, no information is lost when returning to the time domain.**

1. **Matlab function code listing** (with appropriate comments)

function [Cxy, index] =  NormCrossCorrelate(xn, yn)

%this function takes 2 inputs and takes the normalized cross correlation

%between the two

My = length(yn);       %compute lengths

Mx = length(xn);

M = My + Mx - 1;       %number of samples for fft is sum of lenghts - 1

n = 2^(nextpow2(M));   %for most efficient fft computation

extra = floor(n-M);    %to see how much is going to be zero padded

index = (0:M-1);

Xk = fft(xn, n);       %fft of x[n]

Yk = fft(yn, n);       %fft of y[n]

Ck = Xk.\*conj(Yk);     % to compute cross correlation conv. in time domain is multipl. in freq.

cn = real(ifft(Ck));   %compute the cross correlation of x and y

rxx = real(ifft(Xk.\*conj(Xk))); %compute autocorrelation of x

ryy = real(ifft(Yk.\*conj(Yk))); %compute autocorrelation of y

Cxy\_temp = cn/(sqrt(rxx(1)\*ryy(1))); %compute the normalized cross correlation of x and y

Cxy = Cxy\_temp(1:end-extra);

%Cxy = [Cxy\_temp((Mx/2):end-extra+1) Cxy\_temp(1:(Mx/2)-1)];

%Generate plots

    figure

    %make stem plot of input

    subplot(3, 1, 1)

    stem(0:(length(xn)-1), xn, '.', 'MarkerSize', 20, 'Linewidth', 2);

    xlabel('Sample Index')

    ylabel('Amplitude')

    title('x[n] Sequence')

    %make stem plot of unit sample response

    subplot(3, 1, 2)

    stem(0:(length(yn)-1), yn, '.', 'MarkerSize', 20, 'Linewidth', 2);

    xlabel('Sample Index')

    ylabel('Amplitude')

    title('y[n] Sequence')

    %make stem plot of output

    subplot(3, 1, 3)

    stem(index, Cxy, '.', 'MarkerSize', 20, 'Linewidth', 2);

    xlabel('Sample Index')

    ylabel('Normalized Amplitude')

    title('Cross Correlation Between x[n] and y[n]')

    xlim([0 length(Cxy)])

end

**3) Who’s Out There???**

**Test case results**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Direction** | **Relative**  **Bearing**  **(degrees)** | **Distance to the Target Found (meters)** | | | |
| **Russian Typhoon-Class Ballistic Missile Submarine** | **Russian Akula-Class Attack Submarine** | **US Los Angeles Class Attack Submarine** | **Rock / Wall** |
| **Forward** | **0** | - | - | - | 936.9825 |
| **Starboard Bow** | **45** | - | - | - | 723.7575 |
| **Starboard Beam** | **90** | - | - | - | 425.6325, 648.7275 |
| **Starboard Quarter** | **135** | - | 783.7275 | - | 929.7675 |
| **Aft** | **180** | 1042.7325 | - | - | - |
| **Port Quarter** | **225** | - | - | - | 1058.7525 |
| **Port Beam** | **270** | - | - | - | - |
| **Port Bow** | **315** | - | - | 351.7425 | - |

1. In addition, provide plots that support your conclusions for the distances and the type of target present.

To encompass the data with supporting plots, the correlation of three directions with each echo signal can be seen below:

**Starboard Bow (one object):**

|  |  |
| --- | --- |
| LA Sub Echo | Typhoon Sub Echo |
| Akula Sub Echo | Rock Echo |

**Starboard Quarter (two objects):**

|  |  |
| --- | --- |
| LA Sub Echo | Typhoon Sub Echo |
| Akula Sub Echo | Rock Echo |

**Port Beam (no objects):**

|  |  |
| --- | --- |
| LA Sub Echo | Typhoon Sub Echo |
| Akula Sub Echo | Rock Echo |

1. Describe what you hear when you listen to the cross-correlation results when there is a target present (both what the target sounds like, and what you hear during the rest of the signal).

**When the target is present the target sounds like a beap. For the rest of the signal duration the signal sounds like air is flowing through a hollow tunnel.**

**Draw a diagram (to scale)** showing the target positions relative to the SS CalPoly**. Indicate the position** of any valid target **on the following grid**. **Identify the target** (what type of sub, or rock/wall), and write in the numeric value of the **distance to the object**.

1. **Matlab code listing** (with appropriate comments)

**Matlab code listing** (with appropriate comments):

function [sub\_type, disp\_dist] = SonarDetect(xn)

%This funtion takes 1 input (xn) which must be an echo signal and will

%return the type of sub/if there is a rock in the path of the echo and at

%what distance the obkect is

load('SubSonar', '-mat')

%xn\_envelope = abs(hilbert(xn));

%compute cross correlatation the received echo signal with each of the

%echos from subs

[LAsub, LAindex] = NormCrossCorrelate(xn, LosAngelesSubEcho);

[Akulasub, Akulaindex] = NormCrossCorrelate(xn, AkulaSubEcho);

[Typhoonsub, Typhoonindex] = NormCrossCorrelate(xn, TyphoonSubEcho);

%compute cross correlation b/w received echo signal with the Txpulse to see

%if echo is caused by rock/wall

[Rock, Rockindex] = NormCrossCorrelate(xn, TxPulse);

%take the Hilbert Transform to accurately find the absolute peak of the

%cross-correlation results

LA\_envelope = abs(hilbert(LAsub));

Akula\_envelope = abs(hilbert(Akulasub));

Typhoon\_envelope = abs(hilbert(Typhoonsub));

Rock\_envelope = abs(hilbert(Rock));

%find the index where the cross-correlation is at its max

LA\_i = find(LA\_envelope == max(LA\_envelope), 1, 'first');

Akula\_i = find(Akula\_envelope == max(Akula\_envelope), 1, 'first');

Typhoon\_i = find(Typhoon\_envelope == max(Typhoon\_envelope), 1, 'first');

Rock\_i = find(Rock\_envelope == max(Rock\_envelope), 1, 'first');

%find the index of the secons peak of the cross-correlation result

LA\_envelope\_temp = LA\_envelope; Akula\_envelope\_temp = Akula\_envelope;

Typhoon\_envelope\_temp = Typhoon\_envelope; Rock\_envelope\_temp = Rock\_envelope;

%if (peak\_hil == 1)

    LA\_envelope\_temp(LA\_i - 500: LA\_i + 500) = 0;  %set max of col to 0

%elseif (peak\_hil == 2)

    Akula\_envelope\_temp(Akula\_i - 500: Akula\_i + 500) = 0;  %set max of col to 0

%elseif (peak\_hil == 3)

    Typhoon\_envelope\_temp(Typhoon\_i - 500: Typhoon\_i + 500) = 0;  %set max of col to 0

%else

    Rock\_envelope\_temp(Rock\_i - 500: Rock\_i + 500) = 0;  %set max of col to 0

LA\_ii = find(LA\_envelope\_temp == max(LA\_envelope\_temp), 1, 'first');

Akula\_ii = find(Akula\_envelope\_temp == max(Akula\_envelope\_temp), 1, 'first');

Typhoon\_ii = find(Typhoon\_envelope\_temp == max(Typhoon\_envelope\_temp), 1, 'first');

Rock\_ii = find(Rock\_envelope\_temp == max(Rock\_envelope\_temp), 1, 'first');

if(LA\_i > LA\_ii)

    LA\_temp = LA\_i;

    LA\_i = LA\_ii;

    LA\_ii = LA\_temp;

end

if(Akula\_i > Akula\_ii)

    Akula\_temp = Akula\_i;

    Akula\_i = Akula\_ii;

    Akula\_ii = Akula\_temp;

end

if(Typhoon\_i > Typhoon\_ii)

    Typhoon\_temp = Typhoon\_i;

    Typhoon\_i = Typhoon\_ii;

    Typhoon\_ii = Typhoon\_temp;

end

if(Rock\_i > Rock\_ii)

    Rock\_temp = Rock\_i;

    Rock\_i = Rock\_ii;

    Rock\_ii = Rock\_temp;

end

%organize the magnitude values of the max peaks into an array

peaks\_i = [LA\_envelope(LA\_i) Akula\_envelope(Akula\_i) Typhoon\_envelope(Typhoon\_i) Rock\_envelope(Rock\_i)];

peak\_hil = find(peaks\_i == max(peaks\_i), 1);       %find the index of greatest peak corresponding to cross corr w one of the subs

%organize the magnitude values of the second peaks into an array

peaks\_ii = [LA\_envelope(LA\_ii) Akula\_envelope(Akula\_ii) Typhoon\_envelope(Typhoon\_ii) Rock\_envelope(Rock\_ii)];

peak\_hil\_2 = find(peaks\_ii == max(peaks\_ii), 1);       %find the index of greatest peak corresponding to cross corr w one of the subs

% peaks\_temp = peaks; peaks\_temp(:,peak\_hil) = 0;

% peak\_hil\_2 = find(peaks\_temp == max(peaks\_temp), 1)

    % check if the highest value of cross correlation result is valid

    if (peaks\_i(peak\_hil) > 0.03)

        if (peak\_hil == 1)

            sub\_type = 'LosAngeles Sub';

            peak\_i = LAindex(LA\_i);

        elseif (peak\_hil == 2)

            sub\_type = 'Akula Sub';

            peak\_i = Akulaindex(Akula\_i);

        elseif (peak\_hil == 3)

            sub\_type = 'Typhoon Sub';

            peak\_i = Typhoonindex(Typhoon\_i);

        elseif (peak\_hil == 4)

            sub\_type = 'Rock/Wall';

            peak\_i = Rockindex(Rock\_i);

        end

    else

        sub\_type = 'Nothing in path';

    end

    if (peaks\_ii(peak\_hil\_2) > 0.03)

        if (peak\_hil\_2 == 1)

            sub\_type\_2 = 'LosAngeles Sub';

            peak\_ii = LAindex(LA\_ii);

        elseif (peak\_hil\_2 == 2)

            sub\_type\_2 = 'Akula Sub';

            peak\_ii = Akulaindex(Akula\_ii);

        elseif (peak\_hil\_2 == 3)

            sub\_type\_2 = 'Typhoon Sub';

            peak\_ii = Typhoonindex(Typhoon\_ii);

        elseif (peak\_hil\_2 == 4)

            sub\_type\_2 = 'Rock/Wall';

            peak\_ii = Rockindex(Rock\_ii);

        end

    else

        sub\_type\_2 = 'Nothing in path';

    end

%determine the distance to the detected sub or the rock/wall

fs = 50000; %sampling frequency

vs = 1500;  %speed of sound under water in m/s

if (~strcmp('Nothing in path', sub\_type))

    time\_tx\_rx = ((peak\_i + length(TxPulse)/2)/fs); %calculate time for tx and rx based off of the peak\_i index

    dist = (time\_tx\_rx/2)\*vs; %dist. to object is only half of time\_tx\_rx

else

    dist = 0;

end

if (~strcmp('Nothing in path', sub\_type\_2))

    time\_tx\_rx\_2 = ((peak\_i + length(TxPulse)/2)/fs); %calculate time for tx and rx based off of the peak\_i index

    dist\_2 = (time\_tx\_rx\_2/2)\*vs; %dist. to object is only half of time\_tx\_rx

else

    dist\_2 = 0;

end

    %if (dist <= (dist\_2 + 10) & dist <= (dist\_2 + 10))

        %if (

    %else

        %display distance

        disp\_dist = num2str(dist);

        disp(sub\_type)

        disp(strcat(disp\_dist, 'm'))

        %display distance

        disp\_dist\_2 = num2str(dist\_2);

        disp(sub\_type\_2)

        disp(strcat(disp\_dist\_2, 'm'))

   % end

end

**Conclusions:**

*Summarize one or two learnings about cross correlation and correlation detection that this project helped you understand better. Also describe any particular challenges that you had to overcome, and at least one suggestion for improvement of this lab in the future.*

**Name: Aiku Shintani**

**Conclusions:** Through the course of this project, a practical application of the FFT as well as statistics based signal processing methods were utilized. Cross correlation and auto correlation were the two statistical methods that were used to analyze sonar data. Correlation detection was used to detect objects (and determine their distance) surrounding a vessel in water. In order to do this, the echo signals from several known submarines were characterized initially. Then, the characterized submarine echos were cross correlated with other echo signals in all directions surrounding the vessel. The result of the cross correlation function was then used to determine the probability of there being any submarine, rock formation, or lack of an object. This project helped me understand a practical method of performing correlation detection, which we implemented via a custom normalized cross correlation function.

The most difficult task of this project was to be able to have the DSP program correctly predict the presence of more than 1 object. This task can easily be accomplished visually, through the plotted cross correlation results but it is not as straightforward in code. In the future this lab could improve if there were several additional hints as to why some of the plotting in step 1 was so important to the rest of the lab. It was clear after step 3 was completed but was less obvious when completing step 1.

**Name: Chris Adams**

**Conclusions:** From this project, the concept of using correlation to detect things like objects or notes to decode information was emphasized. It was amazing to see how correlating various echo signals across what looks like noise will allow detections of certain objects in the sonar’s path. Building on the FFT and DTMF decoder we worked on last week, cross correlation is another tool that we can utilize to better decode signals. One area of challenge associated with this project was to correctly obtain the distance of detected objects. When cross correlating, the resulting plots shifts the peaks down half the number of samples in the known echo signals (2500/2 = 1250). While the number of samples it was displaced is minor compared to the long echo, it bothered me that it was slightly incorrect. After trying to figure out how to fix it in the NormalCrossCorrelate function, I decided it was easiest to compensate in the detection function. Also, finding dual peaks took a lot of work. However, after completing this assignment, I learned there is a findpeaks function that would have saved us the trouble.

In the future, making it clear that there is a find peaks function and that people might need to compensate their results for more accurate distances might be useful. That way, students might not overlook accuracy and/or spend extra time writing conditionals to find peaks.