A Software Based Sonar Ranging Sensor for Smart Phones

Daniel Graham, George Simmons, David T.Nguyen, Gang Zhou Senior Member IEEE

***Abstract*—We live in a three dimensional world. However, the smart phones that we use every day are incapable of sensing depth, without the use of custom hardware. By creating new depth sensors, we can provide developers with the tools that they need to create immersive mobile applications that take advantage of the 3D nature of our world. In this paper, we propose a new sonar sensor for smart phones. This sonar sensor does not require any additional hardware, and utilizes the phone's microphone and rear speaker. The sonar sensor calculates distances by measuring the elapsed time between the initial pulse and its reflection. We evaluate the accuracy of the sonar sensor by using it to measure the distance from the phone to an object. We found that we were able to measure the distances of objects accurately with an error bound of 12 centimeters.**

I. Introduction

Sensors on mobile devices have allowed developers to create innovative mobile applications. For example, the use of GPS localization allows developers to create applications that tailor their content based on the user's location [1]. Other sensors such as the proximity sensor help to improve the user's experience by disabling the touch screen when it detects that the user has placed the phone next to his or her ear. This prevents buttons from accidentally being pressed during a phone call [2]. Since the release of Android 1.5, Google has added application program interface (API) support for eight new sensors [3]. These sensors include: ambient temperature sensors, ambient pressure sensors, humidity sensors, gravity sensors, linear acceleration sensors and gyroscopic sensors.

Developing new and innovative sensors for smart phones will help open the field to new possibilities and fuel innovation. In particular, developing sensors that allow smart phones to perceive depth is key. Google's Advanced Technology and Projects Team share this vision. They are currently working on a smart phone that uses custom hardware to perceive depth. The project is nicknamed: "Project Tango" [4]. Engineers at NASA have also partnered with the Google team to attach these phones to robots that will be sent to the international space station [5]. However, Google's structured light sensor does not perform well in outdoor environments, since the light from the sun interferes with the sensor. In this paper we explore the possibility of using sonar to provide depth sensing capabilities in both indoor and outdoor environments and address two unique research questions: *1) How do we design a sonar sensor for smart phones using only the phone's existing hardware? 2) How do environmental factors such as noise, reverberation and temperature affect the sensor's accuracy?.*

The proposed sonar sensor uses the smart phone's rear speaker and microphone, and implements the sonar capabilities on a software platform. The software process is comprised of three major steps: 1) a signal generation step, 2) a signal capture step, and 3) a signal processing step. During the signal generation step, the phone's rear speaker emits a pulse. The pulse forms a pressure wave which travels through the air until it encounters an object, which then reflects the pulse and scatters it in multiple directions. During the signal capture step, the microphone captures the reflected pulse, and the distance to the object is determined by calculating the time between the pulse and its reflection.

However, factors such as noise and multipath propagation negatively affect the system's ability to accurately identify the reflected pulse. To address this we use a technique called pulse compression. Pulse compression is the process of encoding the pulse with a unique signature. This unique signature makes it easier to distinguish the pulse from external noise [6]. The pulse is recovered by calculating the cross correlation between the noisy signal and the pulse.

In addition to being corrupted by noise, a pulse may sometimes overlap with another pulse. This occurs when objects close to the system begin to reflect parts of the wave while it is still being transmitted. This limits the minimum distance at which an object can be detected. Encoding the pulse helps to reduce this distance by allowing the filtering process to distinguish between overlapping pulses.

The sonar sensor was evaluated using three metrics: accuracy, robustness, and real-time performance. The accuracy of the sonar sensor was evaluated by comparing the distances reported by our sensor with known distances. The sensor accurately measured distances within 12 centimeters. The robustness of the sensor was evaluated by comparing the sensor's accuracy under different noise and reverberation conditions in different environments. Finally, the sensor's real-time performance was evaluated by measuring the time that it takes to process a signal and return a measurement when different optimizations are applied. By using a collection of optimizations we were able to reduce the processing time from 27 seconds to under two seconds.

In-air sonar has been extensively studied in the literature and supports a vast array of sensing capabilities beyond simply ranging. State of the art systems can determine the 3D positions of objects [7] and can even ascertain properties of these objects [8]. However, these techniques cannot simply be ported to smart phones. Implementing these techniques on smart phones presents a collection of unique challenges and therefore requires a measured and systematic approach. In this paper we begin by exploring ranging applications.

The main contributions of the paper are:

- Presents a design and implementation of a sonar sensor for smart phones that does not require specialized hardware.
- Uses the smart phone's temperature sensor to improve the accuracy of the readings.
- Evaluates the sonar sensor under different reverberation and temperature conditions.

The remainder of the paper is structured as follows: Section 2 outlines the necessary background. Section 3 outlines the related work. Section 4 and 5 describe our approach and system design. Section 6 describes our evaluation. Finally, section 7 concludes the paper.

II. Background

A sonar system can be decomposed into three steps. Figure 1 shows a simulated example of these steps. During the first step, the system generates a pulse. This pulse travels through the air until it encounters an object. Once the pulse encounters an object, it is reflected by the object. These reflected waves then travel back to the system which records the reflected pulse. The time difference between the initial pulse and the reflected pulse is used to calculate the distance to the object. Since the speed of sound in air is known, the distance to an object can be calculated by multiplying the time difference between the initial pulse and the reflected pulse by the speed of sound, and dividing the result by two. We need to divide by two because the time difference between the reflected pulse and the initial pulse accounts for the time that it takes the wave to travel from the phone to the object and back.

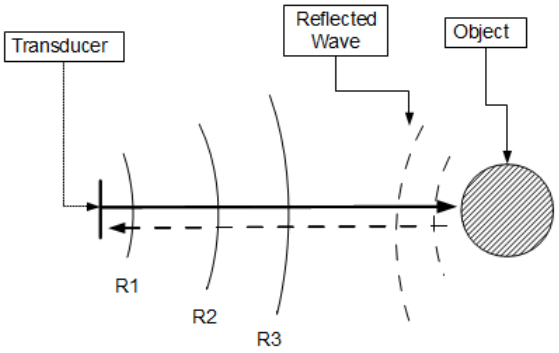
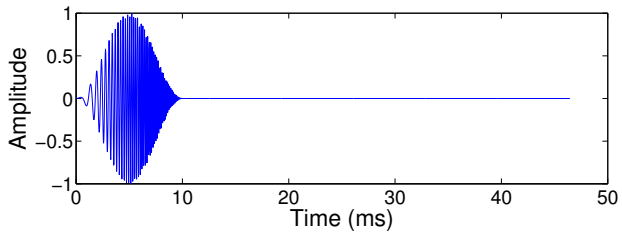


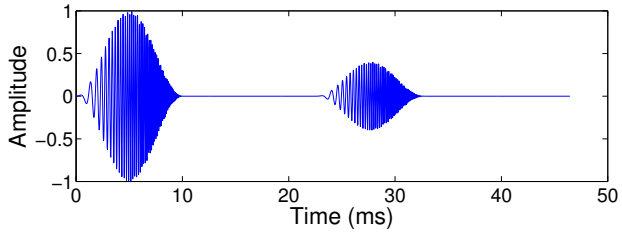
Fig. 1: This figure shows an overview of the process that the sonar system uses to calculate the distance from the system to an object.

The reflected pulse will contain noise from the environment. This noise is reduced by filtering the signal. Figure 2 shows the signals that are generated or recorded at each step.

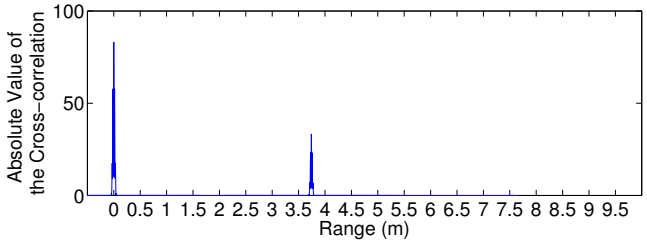
Figure 2a shows the signal that is transmitted from the phone's rear speaker, while figure 2b shows the signal that is received by the phone's microphone. In figure 2b the received signal contains both the initial pulse and the reflected pulse, the phone's microphone will pick up both the transmitted signal and its reflection. This is common in sonar systems where both the transmitter and receiver are located close to each other. Such sonar systems are called monostatic sonar systems. In figure 2b the second pulse represents the reflected pulse. Figure 2c shows the output of the filtering process. The peaks in the resulting filtered signal correspond to the location of the pulse in the original signal. The filtering process will be discussed in detail in section IV-D.



(a) Transmitted pulse



(b) Sound recorded



(c) Result of filtering process

Fig. 2: Figure(a) shows the original pulse that is transmitted. Figure(b) shows the signal that is received by the microphone, and Figure(c) shows the result of the filtering process. These figures are illustrative and have been simulated using a sample rate of 44.1kHz

In figure 2b the reflected signal has a smaller amplitude than the initial pulse because some of the energy has been lost. This is because as sound travels through free space its energy per square meter dissipates as a fixed amount of energy gets spread over a larger surface area. This dissipation is governed by the inverse wave square law [9]. Figure 1 provides a visual explanation of the inverse wave square law. As the wave moves from location $R1$ to $R3$, its energy density decreases since the same amount of energy is spread over a larger surface area.

As the wave travels further from the transmitter, its power density decreases. If an object is too far away, the energy density of the wave that encounters the object may not be enough to generate a reflected wave that can be picked up at the receiver. Distance is not the only factor in determining the amount of energy that is reflected. The amount of energy that is reflected is also determined by the composition and cross section of the object. Larger objects have larger cross

sections and therefore reflect more energy, while smaller objects have smaller cross sections and therefore reflect less energy. Because objects with larger cross sections reflect more energy, they can be detected at larger distances. However, objects with smaller cross sections can only be detected at smaller distances because they reflect less energy. Another key insight for sonar systems is that large flat surfaces act like mirrors and mostly reflect sound waves in the direction of their surface normal. This property is known as the mirror effect.

Sonar systems attempt to accommodate for these limitations by designing special speakers and microphones. To improve the range of sonar systems, speakers are designed so that they focus the speaker's output. The concept of focusing the speaker's output is known as the speaker's gain. Focusing the speaker's output allows sound waves to travel further, in a specified direction. Sonar systems also attempt to improve their range by being able to pick up weaker signals. Just as objects with large surface areas are able to reflect more energy, microphones with large surface areas are able to receive more energy. The concept of a microphone's surface area is known as the microphone's aperture. Once the wave reaches the microphone, it is only able to pick up a subsection of the waves energy. Sonar systems use an array of microphones to increase the receiving surface area thus increasing the microphone's aperture. Now that we have developed an intuition for sonar systems, we will compare the state of the art in-air sonar systems with our proposed system, highlighting the key differences and technological challenges that arise when implementing a sonar system on a smart phone.

III. Related work

In 1968 D. Dean wrote a paper entitled "Towards an air Sonar" in which he outlined some of the fundamental challenges of designing in-air sonar [10]. These challenges included acoustic mismatch and wind effects. Since Dean's paper several in-air sonar systems, have been develop for a variety of applications. These systems include: ultrasonic imaging [11], ultrasonic ranging for robots [12] and SODAR (SOnic Detection And Ranging) systems that measure atmospheric conditions [13]. However, all of these systems have been implemented using custom hardware. By using custom hardware these systems are able to address many of the challenges associated with in-air sonar systems. This is where our system is different. The sonar sensor that we proposed does not use any custom hardware and must compensate for the limitations of the commodity hardware in everyday smartphones.

The earliest occurrence of a smart phone based ranging sensor in the literature occurred in 2007 when Peng et al. proposed an acoustic ranging system for smart phones [14]. This ranging sensor allowed two smartphones to determine the distance between them by sending a collection of beeps. The sensor was accurate to within centimeters. The sensor is a software sensor and only uses the front speaker and microphone on the phone. Our sensor is different from the sensor in [14] because it allows smartphones to determine the distance from the phone to any arbitrary object in the environment.

In 2012, researchers at Carnegie Mellon University proposed a location sensor that allowed users to identify their specific location within a building [15]. The system proposed by the researchers used a collection of ultrasonic chirps that were emitted from a collection of speakers in a room. A smart phone would then listen for these chirps and use this information to locate a person in a room. The phone was able to do this by using the chirps from the speakers to triangulate itself. For example, if the smart phone is closer to one speaker than another it will receive that speaker's chirp before it receives the chirp from another speaker. Since the locations of the speakers are known and the interval of the chirps are also known, the phone is able to use the chirps to triangulate its location. Our system is different from this one, since it attempts to determine the location of the smart phone relative to another object.

Other researchers have also implemented in-air sonar systems on other unconventional systems. For example, researchers at Northwestern University have implemented a sonar system on a laptop [16]. Other researchers have also uploaded code to Matlab central that implements a sonar system on a laptop by using Matlab's data acquisition framework [17]. The closest sensor to the proposed sensor is an iphone application called sonar ruler [18]. The application measures distances using a series of clicks. The application does not filter the signal and requires the user to visually distinguish the pulse from the noise. Our sensor is different from the sonar ruler application because our sensor filters the signal and does not require the user to manually inspect the raw signal. Removing the need for user input allows the proposed sensor to be abstracted using an API. Being able to abstract the sensor using an API is important because it allows the sensor to be easily used by other applications.

IV. Design

The system is comprised of three major components: 1) a signal generation component, 2) a signal capture component and 3) a signal processing component. Figure 3 shows an overview of these components. The signal generation component is responsible for generating the encoded pulse. This component is comprised of two sub-components: a pulse/chirp generation component and a windowing component. The second component is the signal capture component. The signal capture component records the signal that is reflected from the object. The third component is the signal processing component. The signal processing component filters the signal and calculates the time between the initial pulse and its reflection. This component is comprised of two sub-components. The first component is the filtering component and the second subcomponent is the peak detection component. We discuss each component in detail in the following sections.

A. Generating the Signal

The signal generation process is comprised of two subprocesses. The first subprocess generates an encoded pulse, while

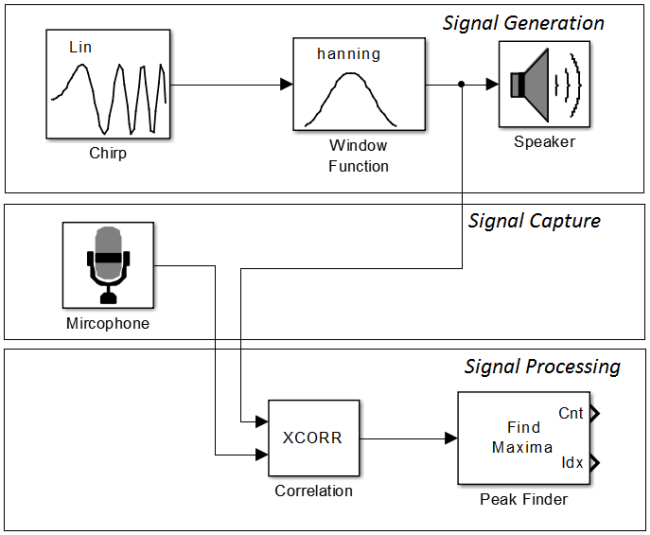


Fig. 3: The figure shows an overview of the sonar system's architecture.

the second subprocess shapes the encoded pulse. We discuss each part of the process in a separate subsection. We begin by discussing the pulse encoding process which is also called pulse compression.

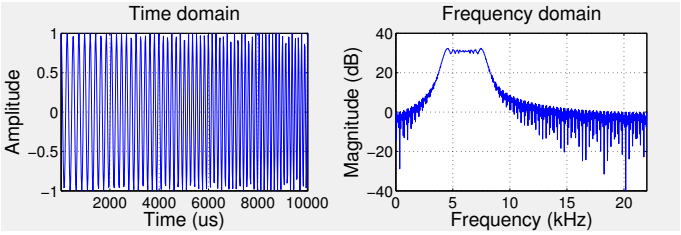
1) Pulse Compression: Pulse compression makes it easier to recover a pulse by encoding the pulse with a unique signature. The pulse can be encoded using amplitude modulation or frequency modulation. Amplitude modulation is the process of encoding a wave by increasing or decreasing the amplitude of sections of the wave, while frequency modulation is the process of varying the frequency of different sections of the wave. The state of the art pulse compression approach uses frequency modulation to create an encoded pulse, since frequency modulation is less susceptible to noise [19]. The encoded pulse is known as a linear chirp. A linear chirp is a signal whose frequency increases linearly from a starting frequency to an ending frequency.

Figure 4a shows an image of the linear chirp in the time and frequency domain. The signal was sampled at 44.1kHz, so each sample index represents $0.227\mu s$. The signal starts at a low frequency and progresses to a higher frequency.

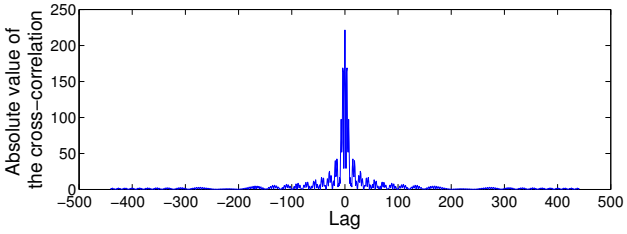
Now that we have discussed the intuition behind a linear chirp, we will look at how the signal is generated. A linear chirp can be expressed as a sinusoidal function. Equation 1 describes how the amplitude of a linear chirp signal varies with time. The value f_0 represents the initial frequency while the value k represents chirp rate (how quickly the frequency increases) and ϕ_o represents the phase of the pulse. The chirp rate can be determined using equation 2.

$$x(t) = \sin\left[\phi_0 + 2\pi\left(f_0 * t + \frac{k}{2} * t^2\right)\right] \tag{1}$$

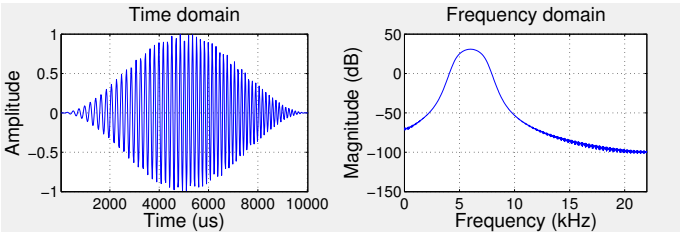
In equation 2 the values f_0 and f_1 represent the starting and ending frequencies, respectively. The value t_1 represents the duration of the pulse.



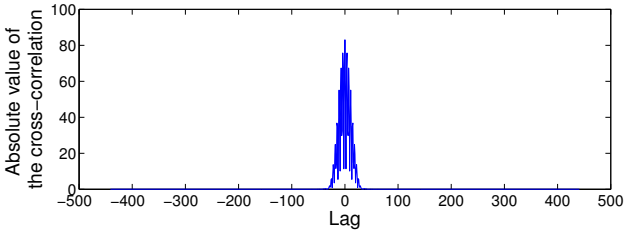
(a) Unwindowed linear chirp time and frequency domain sampled at 44.1kHz



(b) Unwindowed linear chirp autocorrelation showing sidelobes



(c) Windowed linear chirp time and frequency domain sampled at 44.1kHz



(d) Windowed linear chirp autocorrelation showing no sidelobes

Fig. 4: These figures show the time and frequency domain representation for the windowed and unwindowed linear chirp

$$k = \frac{f_1 - f_0}{t_1} \tag{2}$$

f_0	4kHz
f_1	8kHz
t_1	0.01s
ϕ_0	0
Sample Rate	44.1kHz

TABLE I: Chirp properties table

Equation 1 represents the continuous time representation of a linear chirp. However, the signal must be discretized before it can be played by the phone's speaker. The signal is discretized by sampling equation 1 at specific time intervals. The time intervals are determined based on the sample rate and the pulse duration. In our implementation, a linear chirp was generated using the parameters shown in table I. Though

we have selected a linear chirp with these properties it is important to note that other linear chirps can be used with different frequencies and sweep times.

Now that we have discussed how the encoded pulse is generated, we will discuss how the pulse is shaped in the next section. This process of shaping the pulse is known as windowing.

2) Windowing the Pulse: The windowing process is the second subprocess of the signal generating process. The windowing subprocess shapes the pulse by passing it through a window. Shaping the pulse improves the pulse's signal to noise ratio by improving the peak to side lobe ratio. This becomes clearer when we compare the autocorrelated representation of the windowed signal in figure 4d with the unwindowed signal in figure 4b. Notice that the autocorrelated representation of the windowed signal does not contain the additional peaks/sidelobes that are in the autocorrelated representation of the unwindowed signal.

The signal in figure 4c was windowed using a hanning window [20]. The pulse is shaped by multiplying it by a hanning window of the same length. The hanning window is described by equation 3. In equation 3, N represents the number of samples in the window and n represents the index of a sample. Since the pulse is $0.01s$ and the sample rate is 44.1kHz, the window has a length of 441 samples. The discretized pulse is shaped by performing an element-wise multiplication between the discretized pulse vector and the discretized hanning window. Finally, the discretized shaped pulse is sent to the speaker's buffer to be transmitted.

$$H[n] = 0.5 * (1 - cos(\frac{2 * \pi * n}{N - 1})) \tag{3}$$

B. Capturing the Signal

Once the system has transmitted the pulse, the next step is capturing the pulse's reflection. The signal capture component is responsible for capturing the signal's reflection. However, accurately capturing the signal possess two unique challenges. The first challenge is working with the constraints of the phone's sampling rate and the second challenge is concurrently managing the hardware's microphone and speaker buffers.

1) Sampling Constraints and Hardware Requirements: The range of frequencies that can be recovered by the phone is limited by the maximum sampling rate and frequency response of the hardware. This is because in order to recover a wave we must sample at more than twice the wave's frequency. This means that the frequencies that can be contained in the linear chirp are limited by the sampling rate of the microphone and speaker. The microphone and speaker on the nexus 4 has a maximum sampling rate of 44.1kHz. This means that without the use of compressive sensing techniques it is only possible to generate and record a maximum frequency of $22,050Hz$, since Nyquist sampling theorem states that we must sample at twice the frequency of signal that we are attempting to recover. To ensure that we remain within the sample range of most phones, we use a linear chirp that ranges from 4kHz to 8kHz. Limiting the frequency range of the linear chirp allows us to address the sampling constraints of the hardware. In addition to the sampling constraints of the hardware, the phone's speaker and microphone have frequency response constraints. This means that they are only able to generate and receive a limited range of frequencies. This frequency response depends heavily on the make and model of the microphone and speaker, which can vary drastically among devices. To mitigate this we select a frequency range for the chirp that is slightly above the range of the human voice. So most smart phone microphones and speakers should be able to transmit and receive the pulse.

2) The Concurrency Problem: State of the art sonar systems have the ability to concurrently manage the microphone and speaker buffers. Synchronizing the buffers is important for sonar systems because ideally the system would start recording immediately after it has finished sending the pulse. Starting the recording immediately after the pulse is transmitted provides a baseline for calculating the time difference between when the initial pulse was sent and when the reflected pulse was received. If the buffers are not well managed, the recording may contain the initial pulse, and the time index of the reflected pulse will not accurately represent the time between the initial pulse and the reflected pulse. The android operating system does not allow for real-time concurrent management of the microphone and speaker buffers so synchronizing them is challenging. This means that we must find a way to accurately calculate the time between the initial pulse and reflected pulse without managing the buffers in real-time.

We solve the concurrency problem by starting the recording before the pulse is transmitted. Starting the recording before transmitting the pulse allows us to record the initial pulse as well as the reflected pulse. We can then calculate the distance by calculating the time between the first pulse and the second pulse, since we have recorded both. This solution works because the proposed sonar system is monostatic which means that both the microphone and the speaker are located on the same device.

C. Processing the Signal

Now that we have explained how the sonar system generates a pulse and captures its reflection, we can discuss how the captured signal is processed. The process of analyzing the signal is comprised of two subprocesses. The first process is the filtering process. The signal is filtered by calculating the cross correlation between the known signal and the noisy signal. The result of the filtering process is passed to the peak detection process, which detects the peaks in the output and calculates the distance between each peak. The distance between peaks is used to calculate the distance between an object and the sonar system.

D. Frequency Domain Optimization

The cross-correlation process works by checking each section of the noisy signal for the presence of the known signal. Each section of the noisy signal is checked by calculating the sum of the element-wise multiplication between the known signature and the signal. Equation 4 describes this process.

The $\star$ notation represents the cross-correlation operation and, the value $f^{'}$ represents the complex conjugate of f.

$$(f \star g)[n] = \sum_{m=-\infty}^{\infty} f^{'}[m]g[n+m] \tag{4}$$

Calculating the cross-correlation of the signal can be computationally expensive. This is because, the algorithm that is normally used to calculate the cross-correlation of two signals has an algorithmic complexity of $O(n^2)$ (Assuming that both signals (f and g) have the same length). However, it is possible to optimize the algorithm by performing the computation in the frequency domain. Equation 5 shows how to compute the cross-correlation of two signals in the frequency domain. The $\mathscr{F}$ is the mathematical notation for the Fourier transform, and $\mathscr{F}^{-1}$ represents the inverse Fourier transform.

$$(f \star g) = \mathscr{F}^{-1}[\mathscr{F}[f]^{'} \cdot \mathscr{F}[g]] \tag{5}$$

Matches are represented by large peaks in the output of the filtering process. Higher peaks represent better matches. Figure 5a shows a noisy signal which contains the initial pulse and its reflection while figure 5b shows the output of the filtering process. Notice that the output of the filtering process contains two prominent peaks. These peaks correspond to both the initial pulse and its reflection.

Computing the correlation in the frequency domain allows for faster computation since the algorithm has a lower algorithmic complexity $O(nlog(n))$. The process can be further optimized since the Fourier transform of the known signal has to be computed only once. Reducing the number of samples will also result in faster computation times.

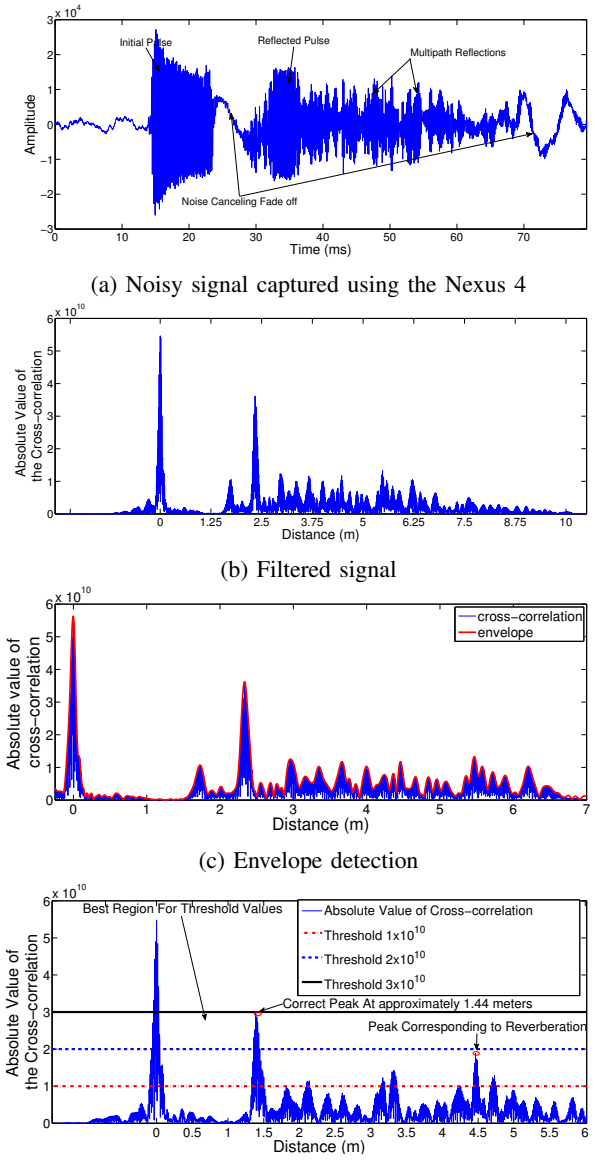
Now that we have discussed the filtering process we can discuss the process that is used to detect the peaks in the filter's output.

E. Peak Detection and Reverberation

Performing peak detection on the cross-correlation function is difficult, since the resulting function is jagged and contains several small peaks. To mitigate this we calculate the envelope of the cross correlation by calculating the analytic signal of the frequency domain multiplication of the pulse and the received signal. The analytic signal is calculated by setting the negative frequencies to zero and doubling the positive frequencies. Once the analytic signal has been calculated the absolute value of the inverse Fourier transform is calculated to determine the final envelope. Figure 5c shows an example of the envelope.

The output of the filtering process is a collection of peaks. The sonar system needs to automatically detect these peaks and calculate the distance between them. In an ideal case the signal would only contain as many peaks as there are objects in the wave's path and a simple peak detection algorithm could be applied. However, factors such as noise cause the filtered signal to contain other peaks.

To account for the noise in the filtered signal, we propose a new peak detection algorithm that selects the most prominent peaks. By only considering peaks above a fixed threshold it is possible to remove the number of peaks that correspond to noise. This threshold can be determined empirically. We define a peak as a point which is higher than its adjacent points.



(a) Noisy signal captured using the Nexus 4

(b) Filtered signal

(c) Envelope detection

(d) Threshold selection

Fig. 5: Figure (a) shows the noisy signal that was captured at approximately 2.5 meters from the wall. Figure (b) shows the filtered signal. Figure (c) shows the result of applying the envelope detection algorithm. Figure (d) shows the detection threshold values applied to another sample taken at 1.5 meters. A Nexus 4 smart phone was used to collect these readings by measuring the distance to an outdoor wall on a college campus.

We propose a peak detection algorithm for detecting the peaks in the cross-correlation result. The threshold that we chose for the Nexus 4 was $22 * 10^9$. This threshold provides the best trade-off between accuracy and range. We selected this threshold empirically by taking 10 samples in a room at different distances. We then selected the detection threshold that resulted in the least number of peaks above the line. The height of a peak is a reflection of the quality of the received

Algorithm 1: Peak Detection Algorithm

```
Input: array, threshold
Output: PeakArray
for i=0; i<array.length; i++ do
    if array[i] > array[i-1] and array[i] > array[i+1]
    then
        if array[i] ≥ threshold then
            PeakArray.add(i);
        end
    end
end
```

signal. The last peak corresponds to the object that is the furthest from the smart phone, while the first peak represents the object that is closest to the phone. Figure 5d shows the threshold superimposed on the filtered signal. Furthermore, figure 6 shows the error in meters verse the threshold that was selected.

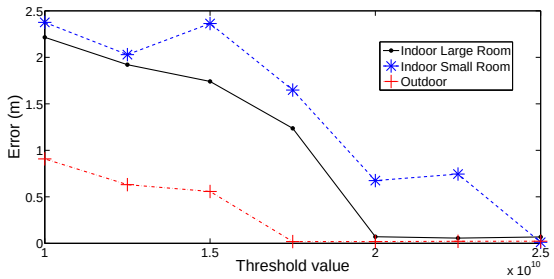


Fig. 6: The figure shows the error in meters vs. the threshold value. These readings were collected using a Nexus 4 smart phone.

We have considered a collection of approaches for determining the threshold including other functions that are more complex than the proposed linear one. However, since the height of the correlation peak also depends on the sensitivity of the microphone the user needs to be able to calibrate the device by adjusting the threshold. A simple linear function makes this calibration process easier. Figure 8a shows the user interface slider that is used to adjust this threshold.

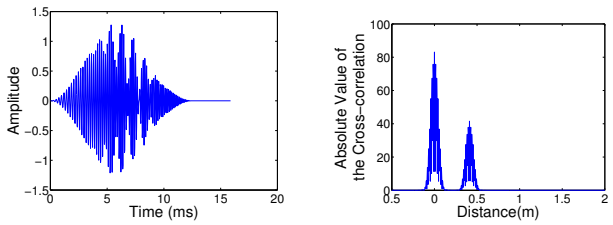
F. Increasing the minimum detection range

The minimum distance at which a sonar system can identify an object is limited. If objects are too close to the system they will reflect parts of the linear chirp while other parts are still being transmitted. This is because the linear chirp is not transmitted instantly but rather over a period of $10ms$.

Consider the example shown in figure 7(a). This figure shows an illustration of the signal that is received when a pulse is reflected from an object at $0.41m$, overlaps with the initial pulse. Notice that it is not possible to visually distinguish the initial pulse from its reflection. This is because as the first linear chirp is generated it begins to travel through the air until it encounters an object at 0.41 meters, $2.5ms$ later. The object will then reflect the waves and the reflected waves will arrive at the system $5ms$ later. This is problematic since the system is recording both the pulse and the reflection. The reflected signals will interfere with the signal that is being generated since the generated signal has a duration of $10ms$. This means that the sonar system cannot detect objects within a 1.65 meter radius.

There are two ways to decrease the minimum distance at which an object can be detected. The first method is to reduce the duration of the pulse. Reducing the duration of the pulse reduces the amount of time that subsequent reflections have to overlap with the pulse. However, reducing the duration of the pulse increases the signal to noise ratio. The second method is pulse compression.

Pulse compression allows the cross-correlation process to identify pulses even when they overlap without increasing the signal to noise ratio. Figure 7(b) shows the signal that results from two overlapping pulses and the output of the filtering process. The two peaks correspond to the correct location of the pulses. Notice that even-though the pulses overlapped the filtering process was able to recover two distinct peaks, because they were encoded. The only thing that limits the minimum distance now is the width of the autocorrelation peak.



(a) Time domain (b) Cross-correlation result

Fig. 7: The figures show an example of how pulse compression helps reduce the minimum distance at which objects can be detected. Figure (a) shows the combination of the two pulses that are received by the microphone. The first pulse is a linear chirp 4kHz to 8kHz with a sweep time of 10ms while the second pulse has the same parameters but is attenuated by a factor of 0.4. Figure (b) shows the result of the filtering process. All signals shown in this figure were sampled at a rate of 44.1kHz.

G. Temperature's Impact on the Speed of Sound

Environmental factors such as temperature, pressure and humidity affect the speed of sound in air. Because these factors affect the speed of sound in air, they also affect the accuracy of a sonar system. The factor that has the most significant impact is temperature [21] [22]. In this section, we show how the ambient temperature can significantly affect the accuracy of a sonar system. We also propose a method for increasing the system's accuracy by using the phone's temperature sensor to estimate the air's temperature.

Equation 6 from [23] describes the relationship between the speed of sound and the air's temperature. Where T_c represents

the air temperature and $v(T_c)$ represents the speed of sound as a function of air temperature.

$$v(T_c) \approx 331.4 + 0.6 * T_c \quad (6)$$

From equation 6 we can see that underestimating the temperature will result in a speed of sound that is slower than its actual speed. Underestimating the speed of sound will cause objects to appear further away than they actually are, while overestimating the temperature will overestimate the speed of sound thus causing objects to appear closer than they actually are.

Overestimating or underestimating the temperature by a single degree results in an error of 0.6 meters for every second of time that has elapsed. Therefore we can improve the accuracy of the sonar sensor by using the phone's temperature sensor. Several phones such as the Samsung S4 now have ambient temperature sensors [24].

H. Smartphone Application

Figure 8a shows a screen shot of the sonar application. The top graph shows the raw signal that was captured by the microphone. The number in the top left shows the distance in meters. The number below it shows the distance in feet. The graph at the bottom shows the absolute value of the filtered signal.The highest peak represents the initial pulse while the second highest peak represents the reflected pulse. The x-axis in both the top and bottom graphs represent the sample index. The graphs have been designed so that the user is able to zoom in the graphs by pinching the display. Subsequent readings are overlaid on the original graph. This allows the user to easily validate the peaks. The horizontal line in the bottom graph represents the detection threshold. Only peaks above this threshold are considered by the peak detection algorithm. The slide bar on the top left is used to adjust the threshold and the value in the box next to it displays the value of the detection threshold. New measurements are taken by pressing the "Measure Distance" button. If the output of the filtering process does not contain peaks above the threshold, the system will report a measurement of zero. The application will also display a message to the screen that asks the user to take another measurement. Since graphs for each measurement are overlaid on each other, the user may reset the display by simply rotating the phone.

To obtain a good reading the user must not cover the phone's microphone or rear speaker. If the user covers the phone's rear speaker, the sound from the rear speaker will be muffled and the chirp signal will be suppressed. If the user covers the phone's microphone, the reflected pulse will not be received. It is also important to calibrate sensor by adjusting the threshold as previously described. The source code and application package(APK) file for this application is available on github [1].

V. Performance Evaluation

We evaluate the sonar system using three metrics: accuracy, robustness and real-time performance. We evaluate the accuracy of the system by measuring known distances under different temperature and reverberation conditions. The accuracy of the system is then determined by comparing the known values to the measured values. All measurements that are reported in this section were collected using a Nexus 4 smart phone

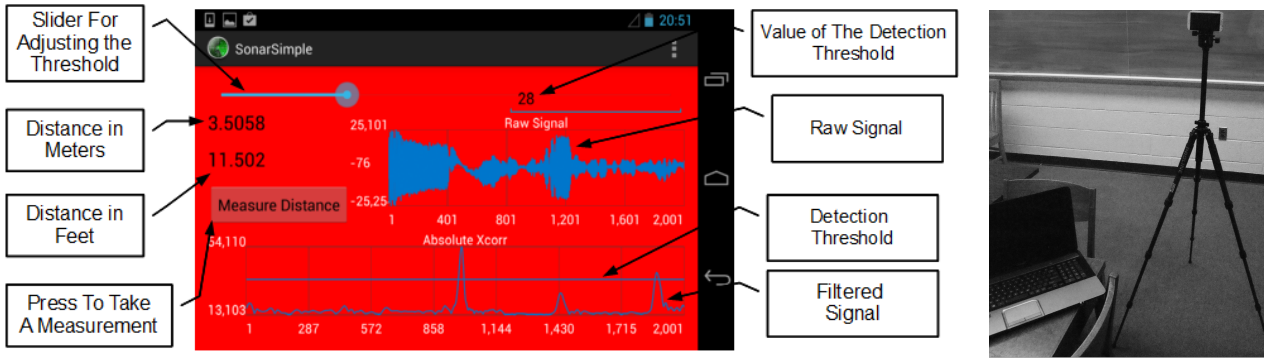
A. Evaluating the Impact of Temperature and Reverberation

In this subsection we explain the process that we used to concurrently measure the sonar system's accuracy and robustness. We measure the sonar sensor's accuracy by using it to measure known distances. Ten measurements were taken at distances between 1 and 4 meters. This range was divided into 0.5 meter increments: 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4. To ensure that the measurements were taken in the same position every time, the phone was placed on a tripod. The tripod was placed at a height of 1.53 meters. It is important to note that stability is not a requirement. The phone does not need to be held perfectly still to obtain a reading. However, since we are evaluating the accuracy of the sonar system, we wanted to ensure that the readings were taken at the same distance every time. Figure 8b shows a picture of the phone on the tripod. We also evaluated the sensor's accuracy in different environments. In particular, we examined the sensor's performance in environments that were expected to have different levels of reverberation and ambient noise. It is well known that reverberation affects the accuracy of sonar systems [25]. Reverberation occurs when a reflected signal is scattered in multiple directions and reflects off other objects in the environment. Some signals may arrive at the receiver later since they have taken a longer path. The signal processing component at the receiver must decide among these multiple reflections. We evaluate the sensor's robustness by repeating the measurement process above for environments that are expected to have different levels of reverberation and ambient noise. Figure 9 shows a picture of each of these environments.

Figure 10a shows the results of the outdoor measurements. The y-axis represents the measured distance, while the x-axis represents the known distance. The dotted line represents a perfect match between measured distances and the known distances. Ideally the sensor's readings should perfectly match this line. The bars represent the standard deviation of the measurements. Each point on the graph represents the average of the 10 readings. The solid black lines represent the readings that were taken using $330m/s$ as the speed of sound. The green lines represent the readings that were taken using the temperature adjusted speed of sound. Notice that most of the non-temperature adjusted readings are slightly below the ideal line. This is because the speed of sound is temperature dependent. Sound travels at $330m/s$ at -1^oC, however it travels faster at higher temperatures for example $343.6m/s$ at 21^oC. Since the system is assuming that sound waves are traveling slower than they actually are, the system under estimates the distance. Some smart phones have temperature sensors. These sensors can be used to improve the accuracy of the system.

The outdoor environment is expected to have the least reverberation, since it is an open environment in which other

[1] http://researcher111.github.io/SonarSimple



(a) Application screenshot

(b) The experimental setup

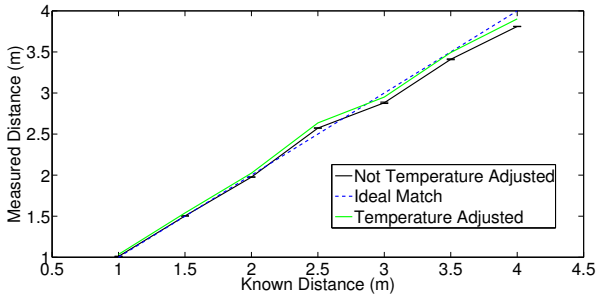
Fig. 8: Figure (a) shows a screen shot of the sonar sensor application running on the Nexus 4. Figure (b) shows the experimental setup that was used in the evaluation

reflecting surfaces are far away. Ten measurements were taken at fixed distances between 1 and 4 meters. Once the ten measurements have been taken, the tripod is moved to the next 0.5 meter increment. The process is repeated until measurements have been taken at all seven distances, for a total of 70 readings. When the measurements were taken, there was low ambient noise, light foot traffic and conversation.

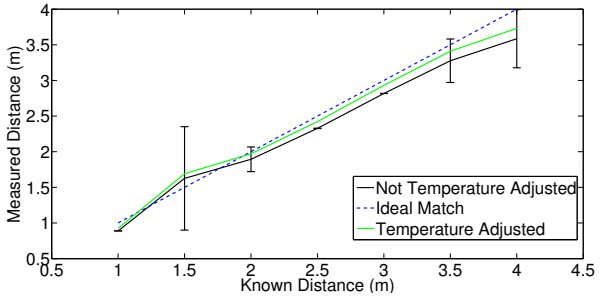
The second environment is a large indoor carpeted class room. This environment is expected to have more reverberation than the outdoor environment, since it contains several chairs and tables. For this experiment the tripod is set up facing a wall in the classroom. The measurement process is repeated until all 10 measurements were obtained for all seven distances. The measurements were plotted in a similar fashion to the outdoor results. Figure 10b shows a plot of these results. Notice the indoor measurements underestimate the ideal line even more than the outdoor environment. This is because the class room is warmer than the outdoor environment and therefore the sound waves are traveling faster. Notice that the measurement at 1.5 meters has a high standard deviation. This is due to the effects of reverberation. The indoor class room also contained ambient noise such as the rumbling of the air conditioning unit.

The final environment is a small indoor room that is not carpeted. The room is 3.2 meters by 3.1 meters and has a ceiling that is 2.9 meters high. All the walls in the room are solid brick. This room is expected to have the highest level of reverberation. Since the room is small and contains tables and chairs, the sound waves are expected to bounce off multiple objects thus resulting in a high level of reverberation and interference. Figure 10c shows the results for the experiments performed in the room. Notice the high standard deviation. This is expected since the room is expected to have a high degree of reverberation. The standard deviation is larger at smaller distances in the small room because as the phone is placed closer to the wall the reverberation effects are greater, due to the proximity of the neighboring walls. The room also contains ambient noise from the air conditioning unit.

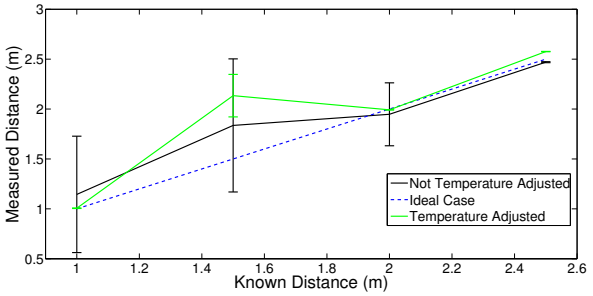
The results in Figure 10 lead us to conclude that this system works well in environments that have low reverberation such as outdoor environments and large rooms but does not work



(a) Outdoor



(b) Large room



(c) Small room

Fig. 10: These figures show the measurements that were taken in all three environments. Figure (a) shows the results for the outdoor environment. Figure (b) shows the results for the large carpeted indoor classroom and figure (c) shows the results for the small indoor room 3.2m by 3.1m.

(a) Outdoor

(b) Large room

(c) Small room

Fig. 9: These figures show pictures of all three environments. Figure (a) shows the outdoor environment. Figure (b) shows the large carpeted indoor classroom and figure (c) shows the small indoor room 3.2m by 3.1m.

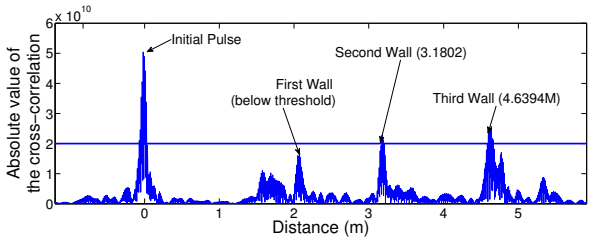
well in areas that have high reverberation such as small rooms.

B. Evaluating the Detection of multiple Objects

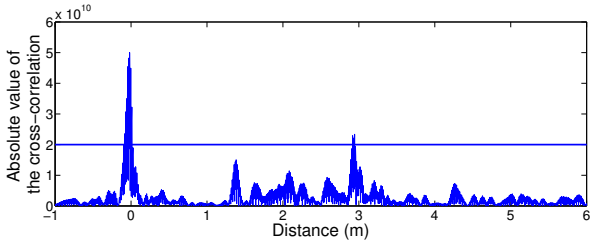
In this experiment, we use the sonar sensor to detect the distance from the phone to three walls that form the sides of a wheel chair ramp. Figure 11 shows a picture of the walls. The output of the filtering process is shown in figure 12a. Notice that, there are four main peaks. The first peak corresponds to the initial pulse while the second, third and fourth peaks correspond to the first, second and third walls, respectively. Each Wall is approximately 1.7 meters apart. The phone was placed 1.7 meters in front of the first wall so that the distances from the phone to each wall would be 1.7, 3.4 and 5.1 meters respectively. In figure 12a each peak is labeled with the distance it represents. It is important to note that this is a single sample and averaging the additional samples will increase accuracy as previously shown.

Fig. 11: This figure shows a picture of three the wall that form the sides of the wheel chair ramp.

The experiment was repeated but this time the phone was placed at an oblique angle to the wall, figure 12b shows the result of the experiment. Notice that the system does not detect the walls when the readings are taken at an oblique angle (approximately 140^o) to the wall. This is because of the mirror effect. Since large planar surfaces reflect sound in the direction of their surface normal, the reflected sound does not get reflected back to the phone, hereby preventing the walls from being detected.



(a) Directly facing the wall



(b) Oblique angle to the wall

Fig. 12: Figure (a) shows the output of the filtering process when the readings were taken directly facing the wall. Figure (b) shows the output of the filtering process when the readings were taken at an oblique angle to the wall. The horizontal line represents a detection threshold of $2*10^{10}$. These readings underestimate the target since they are not temperature adjusted, the speed of sound is assumed to be 330 m/s

C. Evaluating Real-time Performance and System Usage

In this subsection we evaluate the real-time performance of the sonar sensor. In particular we focus on the time that it takes to obtain a reading. The most computationally expensive part of processing a signal is calculating the cross-correlation between the captured signal and the known pulse. In this section we discuss three optimizations and evaluate how each optimization affects the system's real-time performance.

We focus on three optimization strategies: Opt 1) Performing the cross-correlation calculation in the frequency domain. Opt 2) Caching the frequency domain representation of the pulse. Opt 3) Only processing a subsection of the signal. Figure 13 summarizes the result of our experiments.

Calculating the cross-correlation in the time domain is

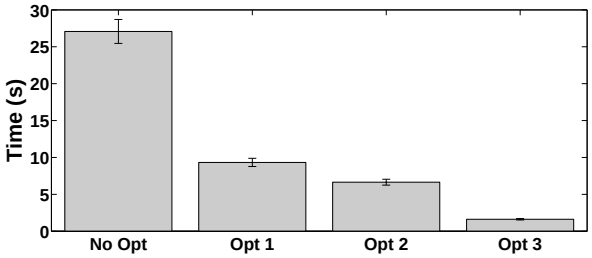


Fig. 13: The figure shows the amount of time that it takes to obtain a single sonar reading. No Opt represents not optimizations. Opt 1 represents the Frequency domain optimization, Opt 2 represents caching the FFT of the pulse and Opt 3 represents limiting the number of samples

computationally expensive and has an algorithmic complexity of $O(n^2)$. It takes an average of 27 seconds to return a result. However, calculating the cross-correlation in the frequency domain has an algorithmic complexity of $O(nlog(n))$. Performing the calculation in the frequency domain reduces that average time from 27.01 seconds to 9.33 seconds. This is equivalent to a 290% reduction in the amount of time that it takes to return a reading. However, it still takes over 9.33 seconds to return a result. Ideally we would like to get the response time to under 2 seconds. In an effort to reduce the time we introduced the second optimization Opt 2. This optimization reduces the processing time by caching the Fast Fourier transform of the pulse. Since the transform of the known pulse does not change, we only have to calculate its transform once. This reduces the average processing time by 2.68 seconds resulting in an average processing time to 6.65 seconds. However, this is still above the ideal value of 2 seconds.We further optimize the process by limiting the number of samples that we consider to 2048. This does not affect the accuracy of the system but it does limit the system's range to approximately 4 meters. Limiting the number of samples reduces the processing time to 1.62 seconds. This is 0.38 seconds below the ideal processing time of 2 seconds.

VI. Conclusion and Future Work

The proposed sonar sensor is comprised of three components: a signal generation component, a signal capture component and a signal processing component. Designing a sonar system for smart phones presented two unique challenges: 1) concurrently managing the buffers and 2) achieving real-time performance. We addressed the concurrency problem by starting the recording before transmitting the pulse. This allowed us to capture the pulse along with its reflected pulses. Doing this allowed us to determine the index of the pulse and reflections by filtering the signal. We addressed the real-time performance problem by reducing the algorithmic complexity of the filtering process from $O(n^2)$ to a $O(nlog(n))$ by performing the cross-correlation calculation in the frequency domain.

Finally, we evaluated our sonar sensor using three metrics: accuracy, robustness, and efficiency. We found that the system was able to accurately measure distances within 12 centimeters. We evaluated the robustness of the sensor by using it to measure distances in environments with different levels of reverberation. We concluded that the system works well in environments that have low reverberation such as outdoor environments and large rooms but does not work well in areas that have high reverberation such as small rooms. In the future, we plan to investigate strategies for improving the sonar sensor's accuracy in environments with high reverberation. We also evaluated the system's real-time performance. We found that by performing three optimizations we were able to reduce the computation from 27 seconds to under 2 seconds. In the future we will be releasing the sonar application on the android market. This will allow us to test the sensor's performance on different hardware platforms.

VII. Acknowledgement

This work was supported in part by U.S. National Science Foundation under grants CNS-1253506 (CAREER) and CNS-1250180.

References

[1] J. Nord, K. Synnes, and P. Parnes, "An architecture for location aware applications," in *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*. IEEE, 2002, pp. 3805–3810.

[2] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *Communications Magazine, IEEE*, vol. 48, no. 9, pp. 140–150, 2010.

[3] "Sensors Overview," http://developer.android.com/guide/topics/sensors/sensors_overview.html, 2013, [Online; accessed 22-November-2013].

[4] "Introducing project tango," https://www.google.com/atap/projecttango/#project, accessed: 2014-08-11.

[5] "Nasa sending google's project tango smartphone to space to improve flying robots," http://www.theverge.com/2014/4/19/5629254/nasa-google-partnership-puts-tango-smartphone-on-spheres-robots, accessed: 2014-08-11.

[6] J. Klauder, A. Price, S. Darlington, and W. Albersheim, "July 1960." the theory and design of chirp radars,"," *The Bell System Technical Journal*, vol. 39, no. 4.

[7] H. Akbarally and L. Kleeman, "A sonar sensor for accurate 3d target localisation and classification," in *ICRA*, 1995, pp. 3003–3008.

[8] J.-E. Grunwald, S. Schörnich, and L. Wiegrebe, "Classification of natural textures in echolocation," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 15, pp. 5670–5674, 2004.

[9] W. M. Hartmann, *Signals, sound, and sensation*. Springer, 1997.

[10] D. Dean, "Towards an air sonar," *Ultrasonics*, vol. 6, no. 1, pp. 29–32, 1968.

[11] M. P. Hayes, "Ultrasonic imaging in air with a broadband inverse synthetic aperture sonar," 1997.

[12] W. Burgard, D. Fox, H. Jans, C. Matenar, and S. Thrun, "Sonar-based mapping with mobile robots using em," in *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*. MORGAN KAUFMANN PUBLISHERS, INC., 1999, pp. 67–76.

[13] S. Bradley, "Use of coded waveforms for sodar systems," *Meteorology and Atmospheric Physics*, vol. 71, no. 1-2, pp. 15–23, 1999.

[14] C. Peng, G. Shen, Y. Zhang, Y. Li, and K. Tan, "Beepbeep: a high accuracy acoustic ranging system using cots mobile devices," in *Proceedings of the 5th international conference on Embedded networked sensor systems*, ser. SenSys '07. New York, NY, USA: ACM, 2007, pp. 1–14. [Online]. Available: http://doi.acm.org/10.1145/1322263.1322265

[15] P. Lazik and A. Rowe, "Indoor pseudo-ranging of mobile devices using ultrasonic chirps," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*. ACM, 2012, pp. 99–112.

[16] S. P. Tarzia, R. P. Dick, P. A. Dinda, and G. Memik, "Sonar-based measurement of user presence and attention," in *Proceedings of the 11th international conference on Ubiquitous computing*. ACM, 2009, pp. 89–92.

Daniel Graham is a fourth year student pursuing a Doctorate Degree in Computer Science at William and Mary. He is working with Dr. Gang Zhou, and his research interests include intelligent embedded systems and networks. He received his M.Eng. in Systems Engineering from the University of Virginia in 2011, and his B.S. from the University of Virginia in 2010.

George Simmons is a sixth year student pursuing a Doctorate Degree in Computer Science at William and Mary. He is working with Dr. Gang Zhou, and his research interests include embedded systems and phase-lock loops. He received his M.Sc. in Computer Science from the William and Mary, and his B.S. from the Massachusetts Institute of Technology.

David Nguyen has been working on his Ph.D. in Computer Science at the College of William and Mary, USA, since Fall 2011. He is working with Dr. Gang Zhou, and his research interests include wireless networking and smartphone storage. Before joining W&M, he was a lecturer in Boston for 2 years. He received his M.S. from Suffolk University (Boston, 2010), and his B.S. from Charles University (Prague, 2007). He was a visiting graduate student at Stanford University in 2009.

[17] R. Bemis, "Sonar demo," http://www.mathworks.com/matlabcentral/fileexchange/1731-sonar-demo, 2014.

[18] L. C. Corp, "Sonar ruler," https://itunes.apple.com/us/app/sonar-ruler/id324621243?mt=8, 2014.

[19] M. G. Crosby, "Frequency modulation noise characteristics," *Radio Engineers, Proceedings of the Institute of*, vol. 25, no. 4, pp. 472–514, 1937.

[20] S. K. Mitra, *Digital signal processing: a computer-based approach*. McGraw-Hill Higher Education, 2000.

[21] G. S. Wong and T. F. Embleton, "Variation of the speed of sound in air with humidity and temperature," *The Journal of the Acoustical Society of America*, vol. 77, p. 1710, 1985.

[22] J. Minkoff, "Signals, noise, and active sensors-radar, sonar, laser radar," *NASA STI/Recon Technical Report A*, vol. 93, p. 17900, 1992.

[23] "Speed of sound in air," http://hyperphysics.phy-astr.gsu.edu/hbase/sound/souspe.html, note = Accessed: 2014-12-05.

[24] "Samsung GALAXY S4 SPECS," http://www.samsung.com/latin_en/consumer/mobile-phones/mobile-phones/smartphone/GT-I9500ZKLTPA-spec, 2013, [Online; accessed 27-January-2014].

[25] D. A. Abraham and A. P. Lyons, "Simulation of non-rayleigh reverberation and clutter," *Oceanic Engineering, IEEE Journal of*, vol. 29, no. 2, pp. 347–362, 2004.

Gang Zhou Dr. Gang Zhou is an Associate Professor in the Computer Science Department at the College of William and Mary. He received his Ph.D. degree from the University of Virginia in 2007 under Professor John A. Stankovic. He has published more than 60 papers in the areas of sensor networks, ubiquitous computing, mobile computing and wireless networks. The total citations of his papers are more than 4300 according to Google Scholar, among which the MobiSys04 paper has been cited more than 780 times. He also has 13 papers each of which has been cited more than 100 times since 2004. He is an Editor of IEEE Internet of Things Journal. He is also an Editor of Elsevier Computer Networks Journal. Dr. Zhou served as NSF, NIH, and GENI proposal review panelists multiple times. He also received an award for his outstanding service to the IEEE Instrumentation and Measurement Society in 2008. He is a recipient of the Best Paper Award of IEEE ICNP 2010. He received NSF CAREER Award in 2013. He is a Senior Member of IEEE and a Senior Member of ACM.