A SHOT IN THE DARK

Point-Source Localization of a Gunshot

Group 9, 18-551, Fall 2009

Pranay Jain | pranayj@andrew.cmu.edu
Arda Orhan | ayo@andrew.cmu.edu
Oren Wright | omw@andrew.cmu.edu

Table of Contents

1. OVERVIEW

1.1. THE PROBLEM

Localization of a point-source is not a new problem, and has been widely studied, especially in the fields of radar and acoustics. Acoustic point-source localization of a sound has numerous applications, from voice tracking to beam steering. Specifically, identifying the location of a gunshot is a useful tool for law enforcement. Having sensor systems installed in high-traffic areas can give early warning of a crime in progress and increase the available information at the scene of a crime.

The problem necessitates precise audio processing and real-time array sampling in order to identify the location of a gunshot. A gunshot is broadband and highly localized in time; both factors introduce unique difficulties to the point-source localization problem.

1.2. THE SOLUTION

Two or more microphones in an array can be used to capture a sound, and then the dissimilarities in the acquired waveforms can yield a point-source location. Commercial gunshot-localization solutions are offered by companies like QinetiQ and ShotSpotter. Solutions normally involve one or two stages[1]; two-stage solutions involve first estimating time delays between microphones and then using the time delays for localization.

The critical steps of an implementable two-stage solution are real-time source detection and accurate estimation of time delays between microphones. Source detection requires real-time sampling of multiple inputs in order to obtain relevant data. The detection process is further complicated because a gunshot is broadband, limiting the effectiveness of simple filters.

Time delays estimated with the acquired data must be accurate, a process made difficult by the signal being highly localized in time, which limits the available data and therefore degrades accuracy. Synchronization issues between waveforms can also degrade accuracy, so system-induced delays introduced during the detection and acquisition of data must be compensated for.

1.3. PRIOR WORK AND NOVELTY

Two past projects in 18-551 have been done on point-source localization with a microphone array. The first project, “MASLA: Microphone Arrays for Source Location Applications,” in spring of 2004, used an array of four microphones and the older EVM board to try to track a speaker’s voice in real-time[2]. The second project, “Find That Sexy Noise: An Exploration into Acoustical Location,” in fall of 2007, attempted to build upon the first by using an array of eight microphones to improve the accuracy of localization[3]. Both groups were only partially successful, due mainly to system synchronization issues.[2, 3]

Elements of the project new to 18-551 are a working multichannel system, localization with limited data, and a near field acoustic solution. To our knowledge, this project is the first in the course with more than two audio inputs working in real-time. The synchronization issues of past projects were caused by multiple real-time requirements, PC interfacing problems, and system design flaws; by limiting the number of real-time components in our system design and sampling all microphones through the same simple input process, we were able to sample four microphones in real-time.

Past source localization projects used voice signals, lengthy in time, as system inputs[2, 3]. The signal of a gunshot is highly localized in time. Because of this only one instantiation of the source is possible; an accurate time delay estimate is difficult with no averaging available. Working with limited data reduces the number of necessary computations, but requires algorithms sufficient enough to yield accurate results without sampling a large amount of data.

Acoustic wave problems can be solved in either the far field or the near field. Past projects only computed solutions in the far field; we computed solutions in both. The far field and near field are discussed more extensively in section 2.3.

1.4. SYSTEM DESIGN

A signal representing a gunshot is sampled by an array of four microphones input into the Educational DSP DSK_AUDIO4 audio daughter card at 48 kHz. The input of each microphone is sampled and stored in a circular buffer on the C67 DSP chip. As the microphones are being sampled, one microphone signal is run through a real-time detection algorithm. The buffers are constantly overwritten until an incoming signal is recognized as a gunshot.

The real-time detection algorithm uses a simple matched filter to filter inputs and then checks the maximum absolute value of the filtered signal against a predetermined threshold. If the value exceeds the threshold, the system will stop capturing new data and will send the acquired signals to the localization algorithm.

The localization algorithm is a two-stage algorithm, the first stage being computing the time delay estimates between each microphone using the generalized cross-correlation (GCC) algorithm. The second stage uses time delay estimates in conjunction with the known microphone positions to locate the source. Localization is computed in both the far field, using a simple algorithm, and in the near field, using the Gillette-Silverman (GS) algorithm. Lastly, the point-source estimates and other pertinent data are sent to the PC, as shown in Fig. 1. A Matlab script presents the data to the user in a tractable fashion.

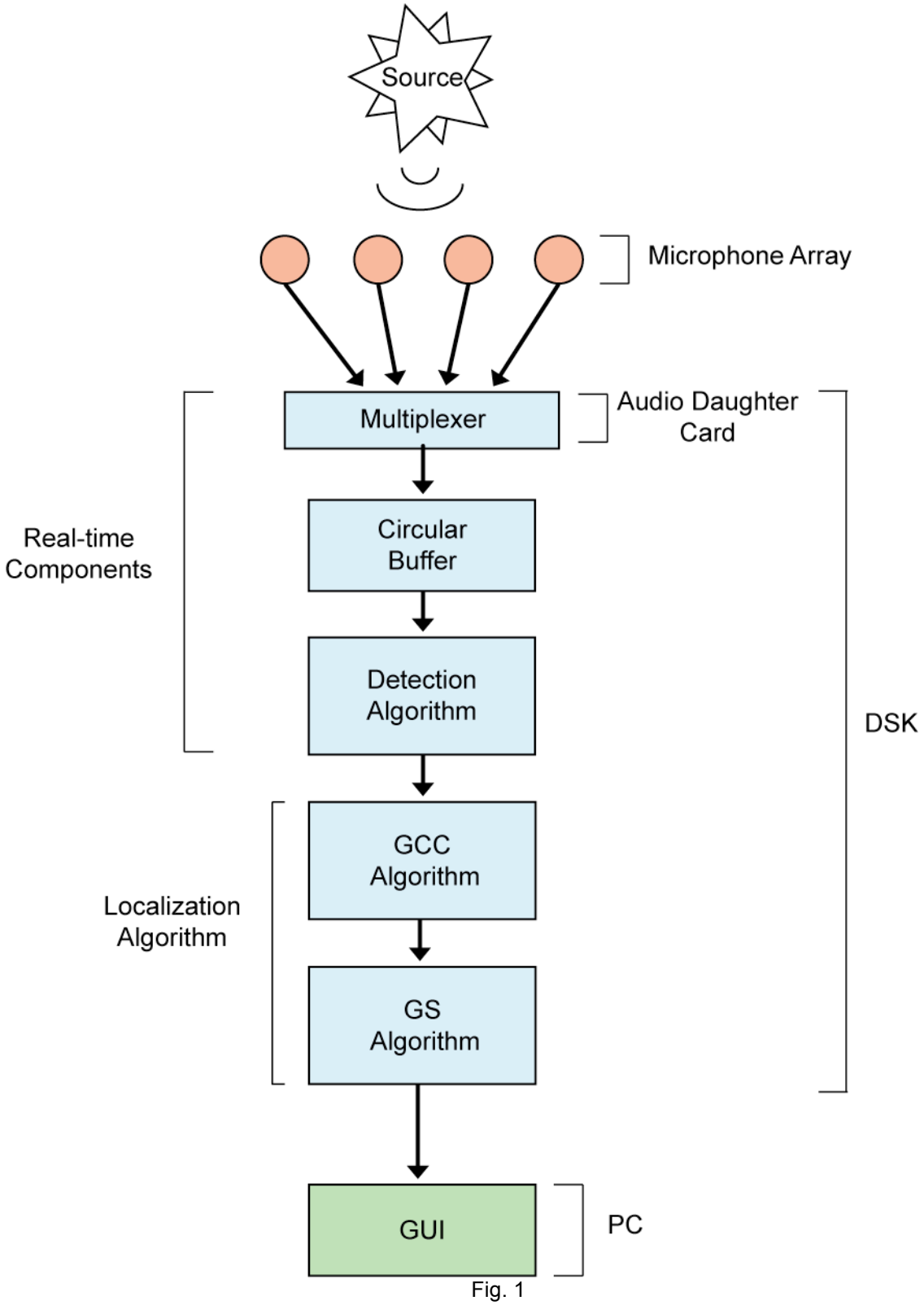
Source
Microphone Array
Multiplexer
Audio Daughter Card
Real-time Components
Circular Buffer
Detection Algorithm
DSK
GCC Algorithm
Localization Algorithm
GS Algorithm
GUI
PC

Fig. 1

2. ALGORITHMS

2.1. REAL-TIME DETECTION

2.1.1. Overview

Detection involves filtering sampled data in real-time and identifying the sound of a gunshot. We use a matched filter in our detection algorithm, because that is the optimal linear filter for maximizing the signal to noise ration when filtering against additive white Gaussian noise.

A predetermined matched filter is convolved with the incoming signal in real-time, and the absolute maximum value of the convolution is checked against an empirically derived threshold value. When the value exceeds the threshold, a gunshot has been detected and the microphone waveforms are input to the localization algorithm.

It is important to note that checking the convolution value against a threshold is somewhat crude, but was deemed appropriate due to the real-time requirements.

2.1.2. Implementation

The detection and acquisition of data was done in an interrupt routine that ran with every new data sample. Because of the real-time requirements, both the detection algorithm and the circular buffer were designed with simplicity in mind.

$$(x_1 * h)[n] = \sum_{m=n-l}^{n} x_1[n-m]h[m]$$

Eqn. 1

The detection algorithm uses a direct convolution as shown in Eqn. 1, with x_1 being the input waveform, h the matched filter, n the sample number, and l the length of the matched filter.

The matched filter length l needed to be long enough to contain the frequency content of a gunshot but short enough so that the direct convolution could be performed in real-time. After testing, a length of 100 samples was decided to be a reasonable compromise.

Once a gunshot is detected, the system continues to acquire new data for half of the circular buffer length, so that the buffer contains gunshot data both before and after the maximum value.

The circular buffer length needed to be able to hold the maximum difference in time acquirable by the microphone array. This is represented in Eqn. 2, where c is the speed of sound and f_s is the sampling frequency. For a practical implementation, the buffer length should greatly exceed this value.

$$\frac{\text{maximum extent of array}}{c} \times f_s << \text{minimum buffer length}$$

Eqn. 2

With the 140-sample requirement met, the circular buffer length, similar to the matched filter length, needed to be long enough to allow for accurate time delay estimation in the localization algorithm but short enough so that all data could be stored in internal memory to facilitate real-time operation. Analyzing gunshot data, a buffer size of 1024 samples for each microphone was decided upon. At the 48 kHz sampling rate, this allows for 21.33 ms of data, which is enough to capture a gunshot. An example of acquired data can be seen in Fig. 2.

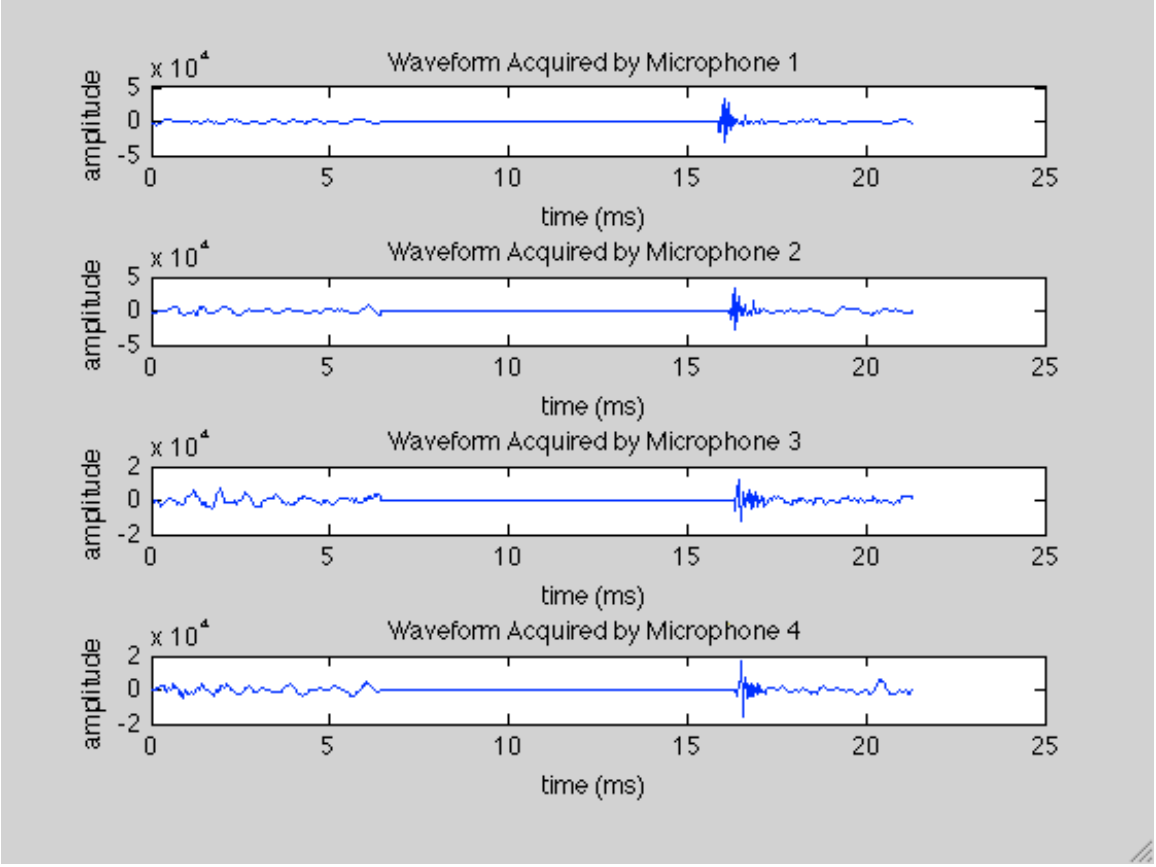

Fig. 2

The entire interrupt routine was profiled at 222 clock cycles with detection and 186 cycles without detection, so the entire routine takes approximately 8.8 μs and the detection algorithm on its own takes approximately 1.36 μs.

2.2. GENERALIZED CROSS CORRELATION

2.2.1. Overview

The first stage of localization is to compute the time delay estimate (TDE) for each unique pair of microphones. The generalized cross correlation (GCC) algorithm, a common method of determining time delays[4], computes the cross correlation function to obtain a TDE. The cross correlation can be approximated as shown in Eqn. 3.

$$R_{x_1x_2}(\tau) = E\left[x_1(t)x_2(t-\tau)\right]$$
$$\hat{R}_{x_1x_2}(\tau) = \frac{1}{T-\tau}\int_t^T x_1(t)x_2(t-\tau)dt$$

Eqn. 3

The GCC algorithm uses this equation, where x_1 and x_2 are microphone waveforms and T is the correlation interval, to approximate the cross correlation as a function of time shift τ. The TDE is simply the abscissa value τ at which the cross correlation function peaks[4]. The GCC has multiple weightings[4] that can alter performance, but in the interest of time we use no weighting.

With four microphones in the array, there are three unique TDE values to compute and input to the second localization stage.

2.2.2. Implementation

In practice, it is more efficient to implement the GCC algorithm using fast Fourier transforms rather than a direct correlation function. The runtime of a correlation function is $O(n^2)$ while the runtime of a FFT is $O(n\log_2(n))$, where n is the number of elements being operated on.

$$X_1 = \Im\{x_1\}$$
$$X_2 = \Im\{x_2\}$$
$$G_{X_2X_1} = X_2 \times X_1^{*(a)}$$
$$\hat{R}_{x_2x_1}(\tau) = \Im\left\{G_{X_2X_1}\right\}^{-1}$$

Eqn. 4

The FFT implementation of the GCC algorithm is shown in Eqn. 4, where X_1, X_2 are the fast Fourier transforms of the waveforms in the time domain after they have been zero-padded. Zero-padding is necessary because the length of the correlation is the sum of the two functions being correlated minus one.

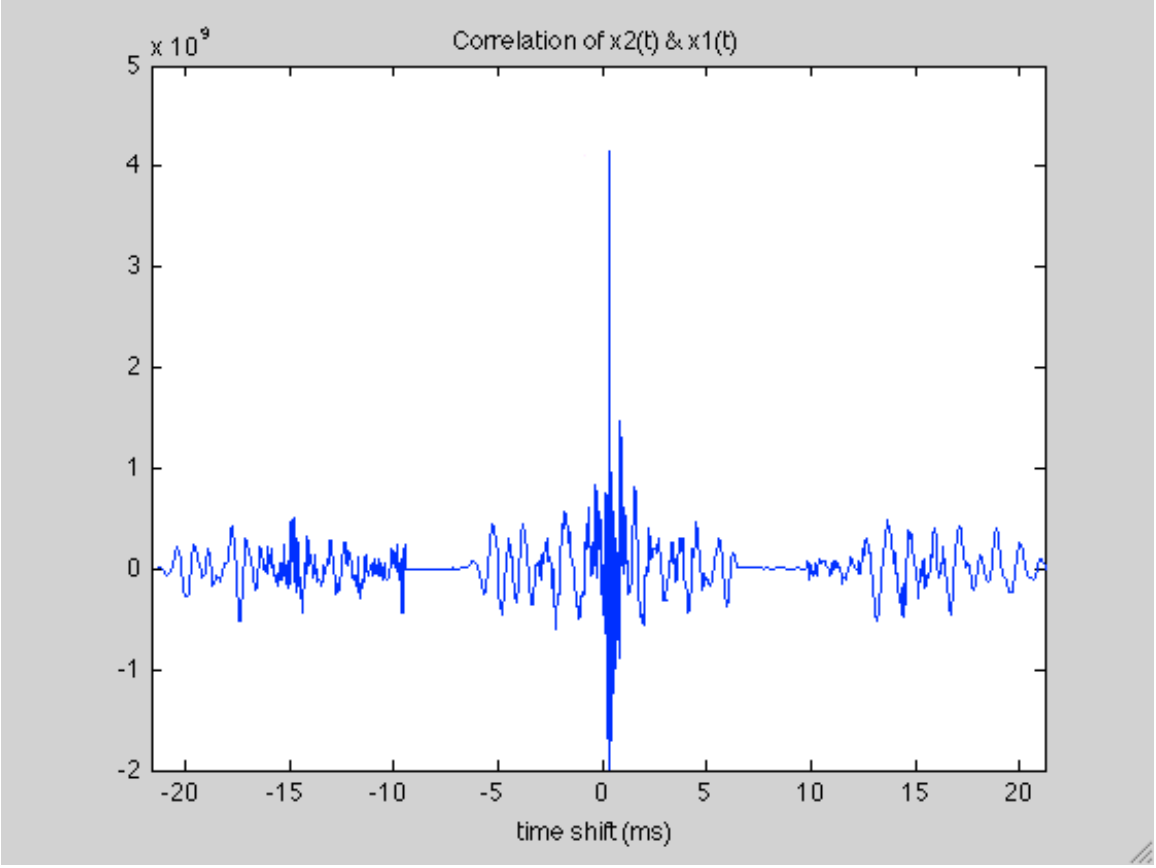

Fig. 3

Fig. 3 shows an example of a cross correlation between two input signals; the peak, and therefore the time delay between signals, of this function is at 0.2917 ms.

In an effort to improve TDE accuracy, we inserted a spectral interpolation step after step (a) of Eqn. 4 to effectively upsample the cross correlation. After multiplying X_2 by the complex conjugate of X_1, the product is zero-padded in the *middle* of the frequency domain (to preserve conjugate symmetry[5]). This zero-padding in the frequency domain results in an increased sampling rate in the time domain[5]. By doubling the signal-length through zero-padding the frequency domain, we upsample our cross correlation by a factor of two.

2.3. GILLETTE-SILVERMAN ALGORITHM

2.3.1. Overview

The second stage of localization is to take the TDE values computed in the first stage and use them to determine the location of the point-source. We use the Gillette-Silverman (GS) algorithm to solve this problem. It is a fairly recent algorithm that takes both the predetermined microphone locations and the TDE values from the first stage and estimates the position of the source in the near field using linear algebra.[1]

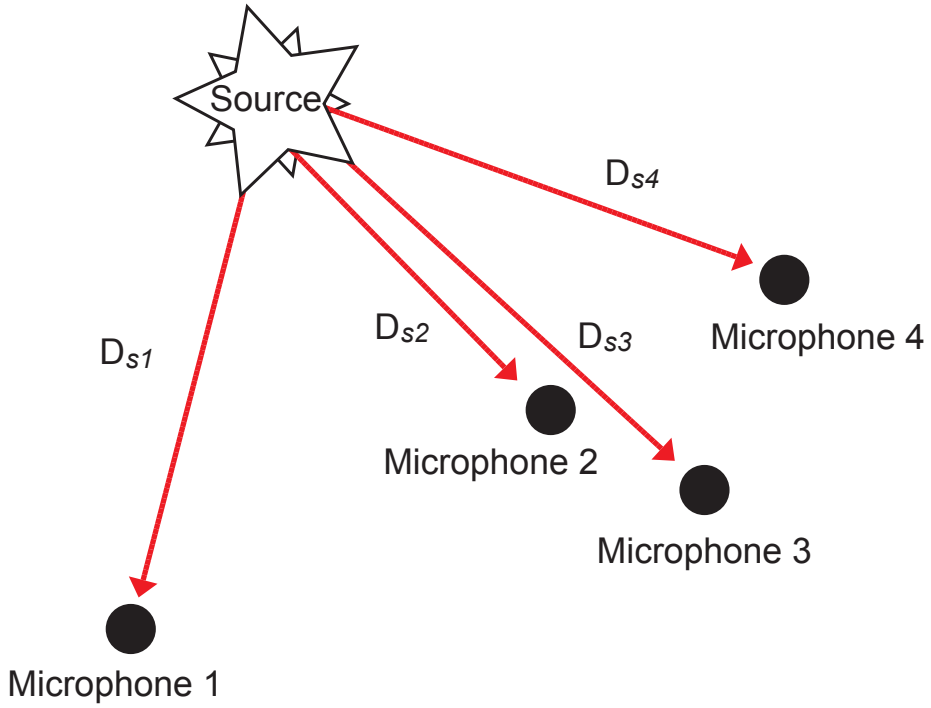

Fig. 4

Using simple geometry in a three-dimensional Cartesian coordinate system, depicted in Fig. 4, it can be shown that,

$$D_{sm}^2 = \left(x_s - x_m\right)^2 + \left(y_s - y_m\right)^2 + \left(z_s - z_m\right)^2$$

$$x_{m1}x_{s1} + y_{m1}y_{s1} + z_{m1}z_{s1} + d_{m1}D_{s1} = \left(x_{m1}^2 + y_{m1}^2 + z_{m1}^2 - d_{m1}^2\right)/2$$

Eqn. 5

where $\left(x_m, y_m, z_m\right)$ is the location of microphone m in Cartesian coordinates and the distance between microphone m and the point-source s is equal to D_{sm}. The differential distance between microphone m and the reference microphone can be expressed by $d_{m1} = \left(D_{sm} - D_{s1}\right)$. If four microphones are used in a planar array (such that $z_{m1} = 0$) to locate a source in two-dimensional space, the system of equations can be reduced to a matrix equation $\mathbf{Ax} = \mathbf{w}$ shown in Eqn. 6, where $\mathbf{x}$ is the source vector to be determined. The point-source location can be found by solving the system of equations.

$$\begin{bmatrix} x_{21} & y_{21} & d_{21} \\ x_{31} & y_{31} & d_{31} \\ x_{41} & y_{41} & d_{41} \end{bmatrix} \times \begin{bmatrix} x_{s1} \\ y_{s1} \\ D_{s1} \end{bmatrix} = \begin{bmatrix} \left(x_{21}^2 + y_{21}^2 - d_{21}^2\right)/2 \\ \left(x_{31}^2 + y_{31}^2 - d_{31}^2\right)/2 \\ \left(x_{41}^2 + y_{41}^2 - d_{41}^2\right)/2 \end{bmatrix}$$

Eqn. 6

It is important to note that the GS algorithm requires a minimum of one more microphone than what other methods require[1], but because it provides a linear closed-form solution in the near field, we consider this desirable. In addition, the geometry of a planar solution cannot disambiguate a source at z_s from its reflection at $-z_s$.

In addition to the near field solution provided by the GS algorithm, a secondary far field solution is computed for comparison. The far field solution assumes planar waveforms, so only the direction of the source in relation to the microphone array can be estimated as shown in Fig. 5.

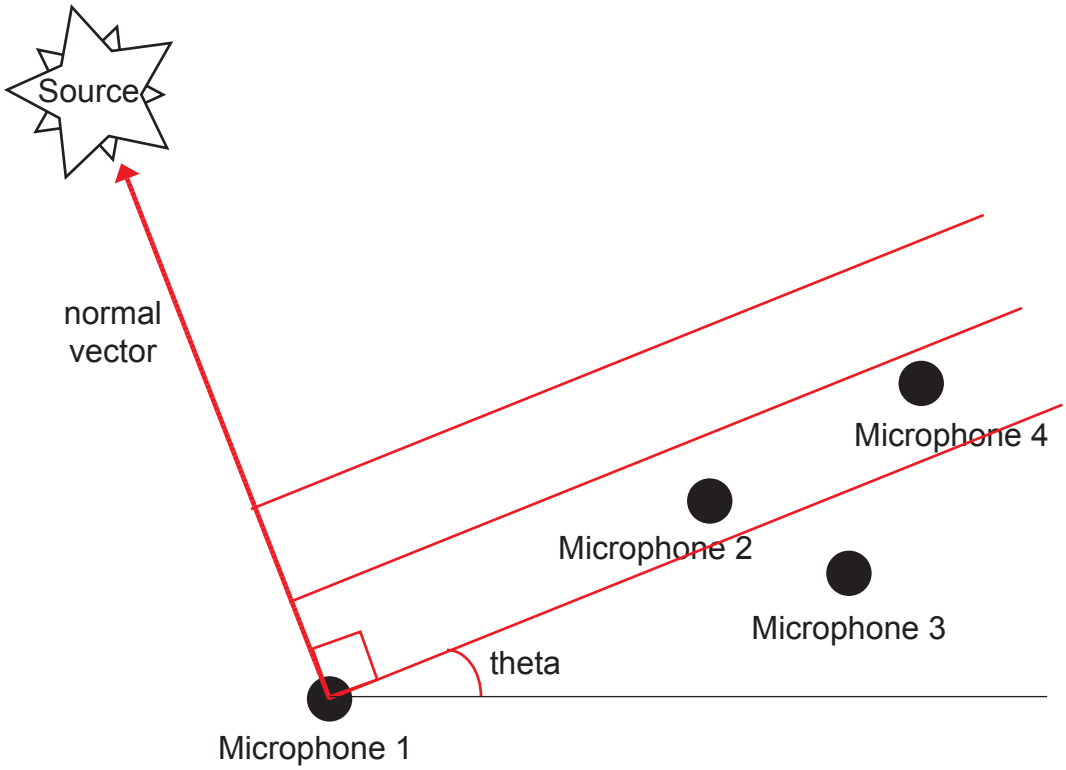

Fig. 5

Like Eqn. 5, simple geometry can be used to show that,

$$x_{m1}\cos\theta\sin\phi + y_{m1}\sin\theta\sin\phi + z_{m1}\cos\phi = -d_{m1}$$

Eqn. 7

where ϕ is the azimuthal angle of arrival and θ is the elevation angle. These angles describe a vector normal to the incoming planar wavefront. As with the near field equations, the location of microphone m in Cartesian coordinates is given by (x_m, y_m, z_m) and the differential distance between the reference microphone and microphone m is given by d_{m1}. If four microphones are used in a planar array (such that $z_{m1} = 0$) to locate a source in two-dimensional space, a matrix equation can be constructed as shown in Eqn. 8.

$$\begin{bmatrix} x_{21} & y_{21} \\ x_{31} & y_{31} \\ x_{41} & y_{41} \end{bmatrix} \times \begin{bmatrix} \cos\theta \sin\phi \\ \sin\theta \sin\phi \end{bmatrix} = -\begin{bmatrix} d_{21} \\ d_{31} \\ d_{41} \end{bmatrix}$$

Eqn. 8

Similar to the GS algorithm, solving the system of equations provides the direction of the source. Note that Eqn. 8 is overdetermined, because a far field solution only requires three microphones. Once again, the planar array geometry cannot disambiguate the angle ϕ from the angle $\pi - \phi$.

2.3.2. Far Field Versus Near Field

Far Field	**Near Field**
Assumes planar waves	Uses spherical waves
Linear problem	Non-linear problem
Only allows for source direction estimate	Allows for source position estimate
Accurate only if distance from array to source >> extent of array[6]	Accuracy degrades with distance from array to source

Fig. 6

The important differences between far and near field are listed in Fig. 6. A near field solution is normally non-linear and most localization methods that operate in the near field are iterative approximations[1]. The GS algorithm, which allows for a linear, closed-form solution, mitigates the computational requirements and provides for a solution altogether more elegant than other near field methods.

It is desirable to solve the point-source localization problem in the acoustic near field for two reasons. The more significant of these is that a far field solution only allows for an estimate of the point-source direction rather than position. The second reason is that a far field approximation requires that $r >> a$, where a is the maximum extent of the sensor array and r is the distance traveled from the source. A factor between three and ten must separate r from a.[6]

The tradeoff is normally that a near field solution is far more computationally intense than a far field approximation, but this is not true with the GS algorithm.

2.3.3. Implementation

Both the far field and near field systems of equations are implemented by using a matrix math library and finding the inverse matrix $\mathbf{A}^{-1}$, where $\mathbf{A}$ the is the first matrix in Eqn. 6 or Eqn. 8, to solve $\mathbf{x} = \mathbf{A}^{-1}\mathbf{w}$. For the GS algorithm, the inverse matrix is found using an adjugate matrix of determinants, as determinants are simple to compute. For the far field solution, the inverse matrix can be hard-coded due to the fact that all values are predetermined, making computation trivial.

3. SIGNALS & HARDWARE

3.1. INPUT SIGNALS

It stands to reason that creating a real gunshot as an input signal was out of the scope of this project, so at first we attempted to make a database of recorded gunshot sounds. A handful of gunshots that were sampled at an adequate frequency without significant clipping were found on the site of the Freesound Project[7]. However, playing these gunshot sounds over a set of speakers was impractical due to volume requirements and the need to frequently move the source for testing.

A gunshot is high energy, highly localized in time, and broadband. Most energy is centered at approximately 1 kHz and low energy content can reach 20 kHz. We decided that a loud hand clap or a clap of wooden blocks were both adequate substitutes for a gunshot because they recreated the significant gunshot characteristics; a comparison of frequency content is shown in Fig. 7.

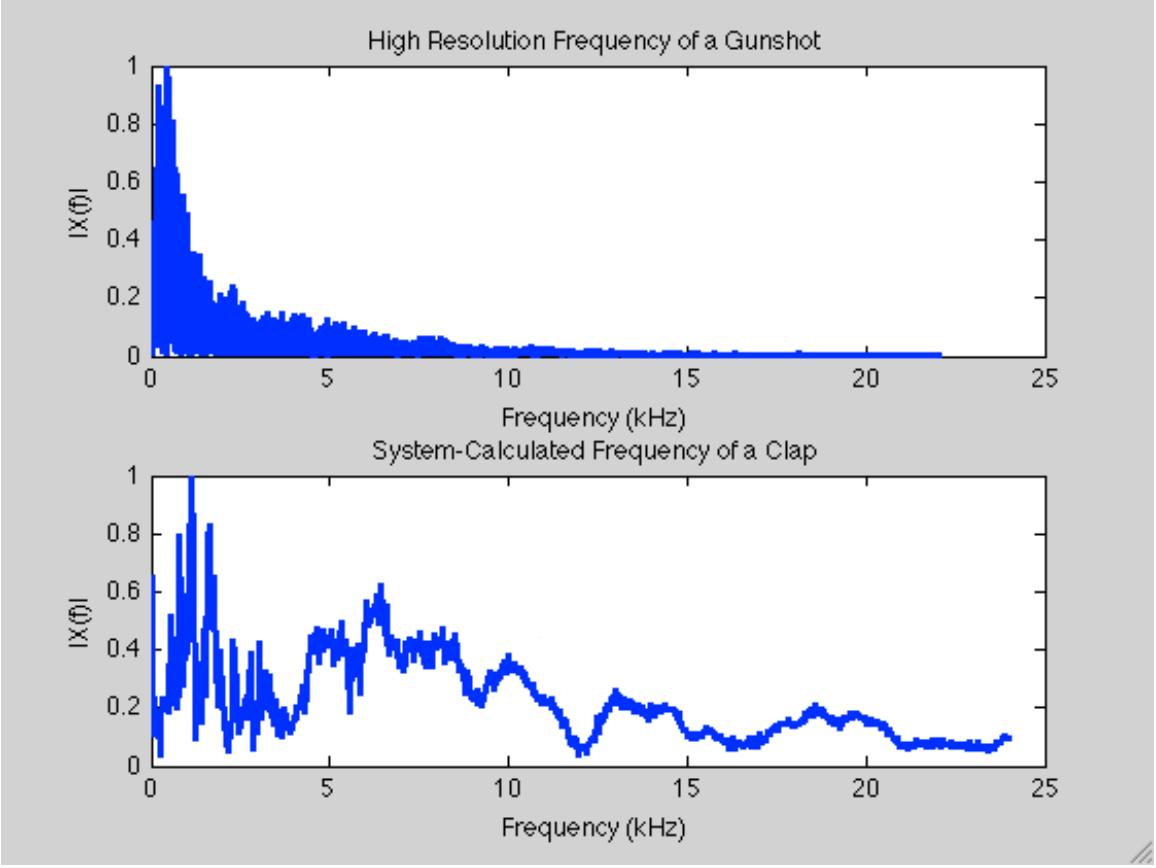

Fig. 7

3.2. MICROPHONE ARRAY

We chose to create a planar array of four randomly spaced microphones at a maximum extent of 1 m. We chose Audio-Technica ATR3350 condenser lavalier microphones for their omnidirectionality and broadband frequency response[8]. The planar geometry was chosen because, unlike a linear array, it allows for no ambiguity in two dimensions, and its structure is simple to create compared to a non-planar, three-dimensional array. Four microphones is the minimum number required to use the GS algorithm in two dimensions.

The GS algorithm will not function if the matrix $\mathbf{A}$, described in section 2.3.1, is singular. Certain configurations of the array microphones, such as uniform spacing, result in a singular matrix. For microphones with random spacing, the matrix is virtually always nonsingular[1]. Because of this, we chose a non-uniform spacing shown in Cartesian coordinates in Eqn. 9.

Microphone 1 (reference) = (0m, 0m)
Microphone 2 = (0.5m, 0.3m)
Microphone 3 = (0.8m, 0.2m)
Microphone 4 = (0.9m, 0.436m)

Eqn. 9

The array was constructed with plywood, and holes were drilled in measured positions. Microphones were placed through the holes from below and clipped into place in the plane.

3.3. AUDIO DAUGHTER CARD

Initially we hoped to avoid using external hardware, which requires dismantling the normal LAN network used by Prof. Casasent and the 18-551 staff. It was quickly established that external hardware would be necessary, and we decided on the Educational DSP DSK_AUDIO4 audio daughter card recommended by Prof. Casasent.

The audio daughter card plugs into the J3 peripheral connector of the DSK, freeing the external memory interface connector. The card has four mono microphone inputs, two stereo inputs, and two stereo outputs; it has 16-bit ADC and DAC and can achieve a maximum sampling rate of 48 kHz. This card is split into two parallel system which each contain two mono inputs, and one stereo input and output. The inputs also have a 20db pre-amp that can be toggled on and off.[9]

The sampling delay between audio daughter card inputs was measured as only a single clock cycle, approximately 40 ns, so we system-induced delays were deemed negligible.

An important resource was Educational DSP employee Mike Morrow, morrow@educationaldsp.com, who corresponded with us via email and helped us understand how signals are read through the card onto the DSK. We asked him where in provided code we would be able to read data into the DSK, which allows us to then use the data in our algorithms. He explained to us how we could use the interrupt service routine to access data and store it in four separate variables.

Interrupt routines the DSK McBSPs must be used to move input data into the DSK internal memory. This process is highlighted by the code in Fig. 8. Bolded text represents our own additions to the generic interrupt routines provided by Educational DSP.

```
volatile union {
    unsigned int uint;
    short channel[2];
    } TxData;

// a single interrupt is used to service the transmit and receive
// sections of both codecs
interrupt void McBSP1_Rx_ISR()
{
    McBSP *port;
    float fdata0, fdata1, fdata2, fdata3;

    port  = McBSP1_Base;
    TxData.uint = port->drr; // get input data from serial 1
```

```
    // process data here
    fdata0 = TxData.channel[0];
    fdata1 = TxData.channel[1];

    port->dxr = TxData.uint; // send output data to serial 1

    port  = McBSP0_Base;
    TxData.uint = port->drr; // get input data from serial 0
    // process data here
    fdata2 = TxData.channel[0];
    fdata3 = TxData.channel[1];

    port->dxr = TxData.uint; // send output data to serial 0
}
```

Fig. 8

3.4. C67 DSK

3.4.1 Attempted Configurations

Once it became apparent that external hardware was necessary, and we decided to use the audio daughter card, we needed a method of transmitting data to the PC; the required removal of the LAN setup necessitated an alternative method.

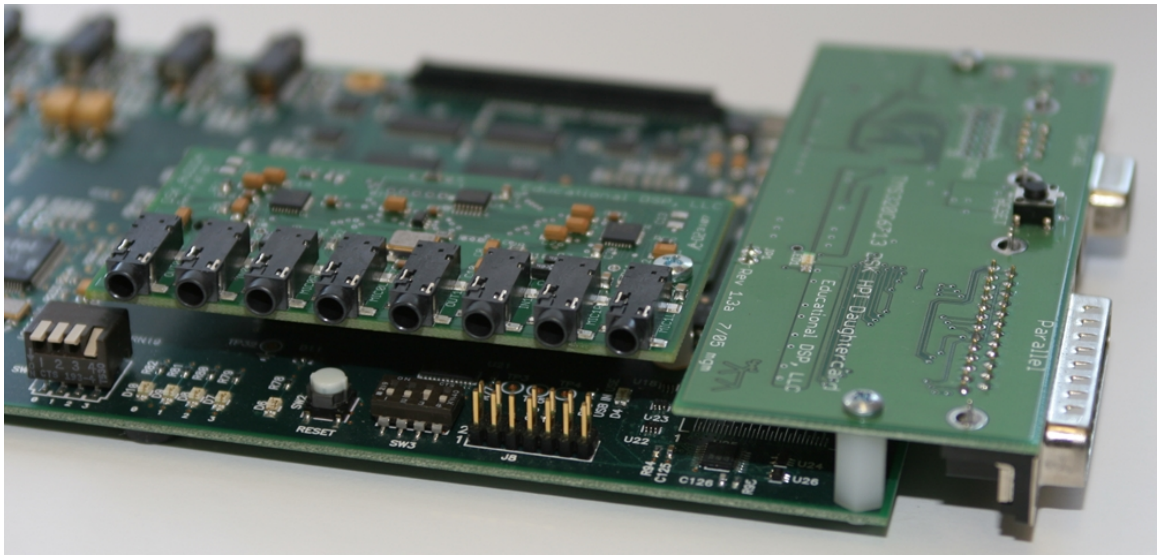

Fig. 9

The Educational DSP DSK6XXXHPI daughter card allows USB, parallel, or serial communication between the DSK and the PC[10]. This was used together with the audio daughter card, where the audio daughter card read in inputs to the DSK and the HPI daughter card sent outputs to the PC. This configuration, shown in Fig. 8, was used for the majority of the project. The HPI daughter card included starter code and useful GUI tools, but was ultimately removed in the final configuration.

3.4.2. Final Configuration

The final configuration ended up abandoning the HPI daughter card for various reasons. Only the GUI was run on the PC and no computations made on the PC were necessary for use on the DSK, and the HPI daughter card ended up unnecessarily complicating the data flow; because of this we decided to simplify the system and communicate directly with the PC.

The HPI daughter card can only access memory on the DSK if the system is no longer processing data. Essentially only one daughter card can access memory at a time. Using the HPI daughter card and the audio daughter card in unison would require an additional interrupt service routine. As we were only transmitting a small amount of data to the PC, communication via the ordinary USB port on the DSK, which requires a small amount of code compared to communication via the HPI daughter card, seemed more appropriate.

The other system elements remained as originally designed, and the system data flow is shown by Fig. 1 in section 1.4. All data was stored in internal memory.

3.5. HPI DAUGHTER CARD AND PC INTERFACE

As stated in section 3.4.2, the HPI daughter card was replaced with a direct interface to the PC. A simple C file read data output by the DSK and wrote the data to text files that were then read by a Matlab script. The script displayed the information to the user in a tractable fashion. Fig. 10 shows a graph of the microphone array and the far and near field estimates displayed by the script.

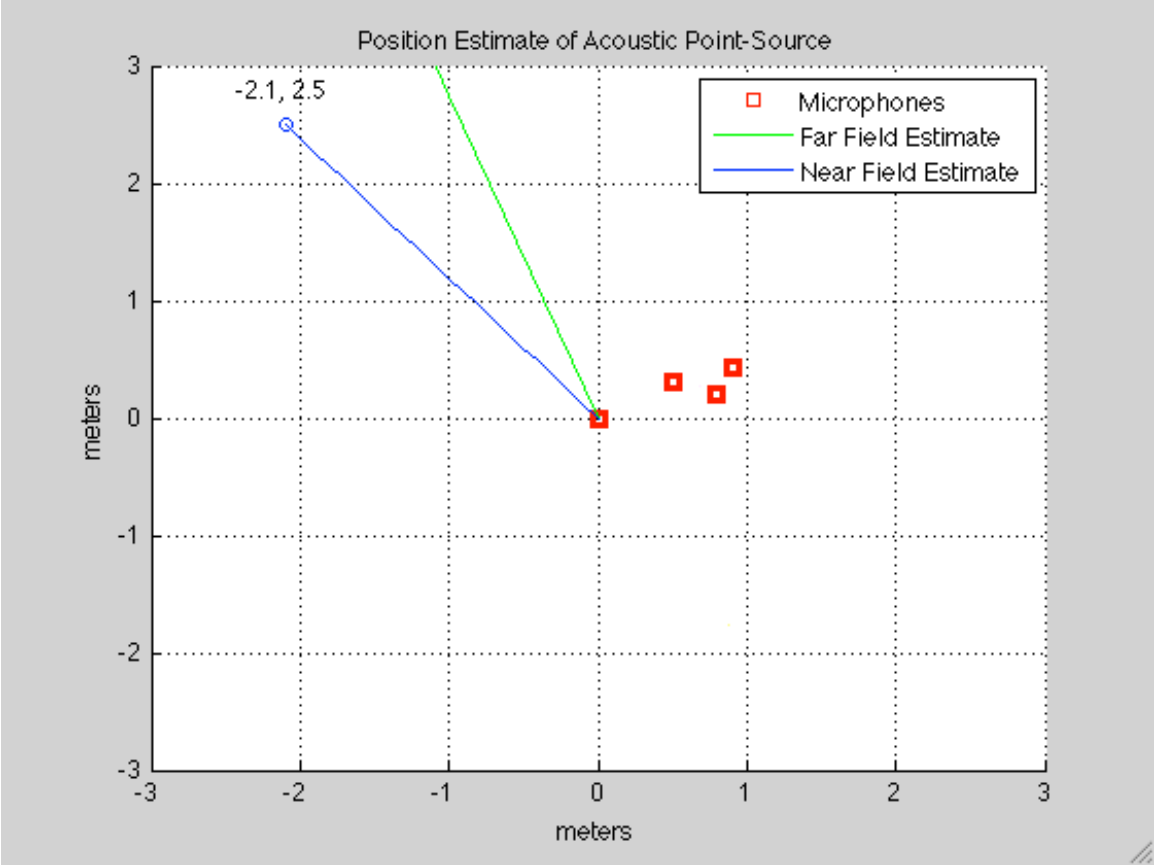

Fig. 10

4. ERROR ANALYSIS

4.1. ARRAY STRUCTURE

As stated in section 3.2, a microphone array should be randomly spaced to ensure a nonsingular matrix. It stands to reason that even with well-positioned microphones, particular source positions can produce a matrix that is singular or close to singular. In testing we found that certain regions in space would fail to produce accurate results. This is shown in Fig. 11. Actual point-source positions are shown in blue and positions approximated by the GS algorithm are shown in red. Trial 14, which has a point-source position in the second quadrant and near the x-axis, produces a very poor result. The condition number of the matrix in trial 14 is 6.4×10^3, corroborating the idea that certain spatial regions can produce a matrix that is especially sensitive to errors in data.

These sensitivities can be mitigated by improving the placement of microphones and increasing the number of microphones in the array.

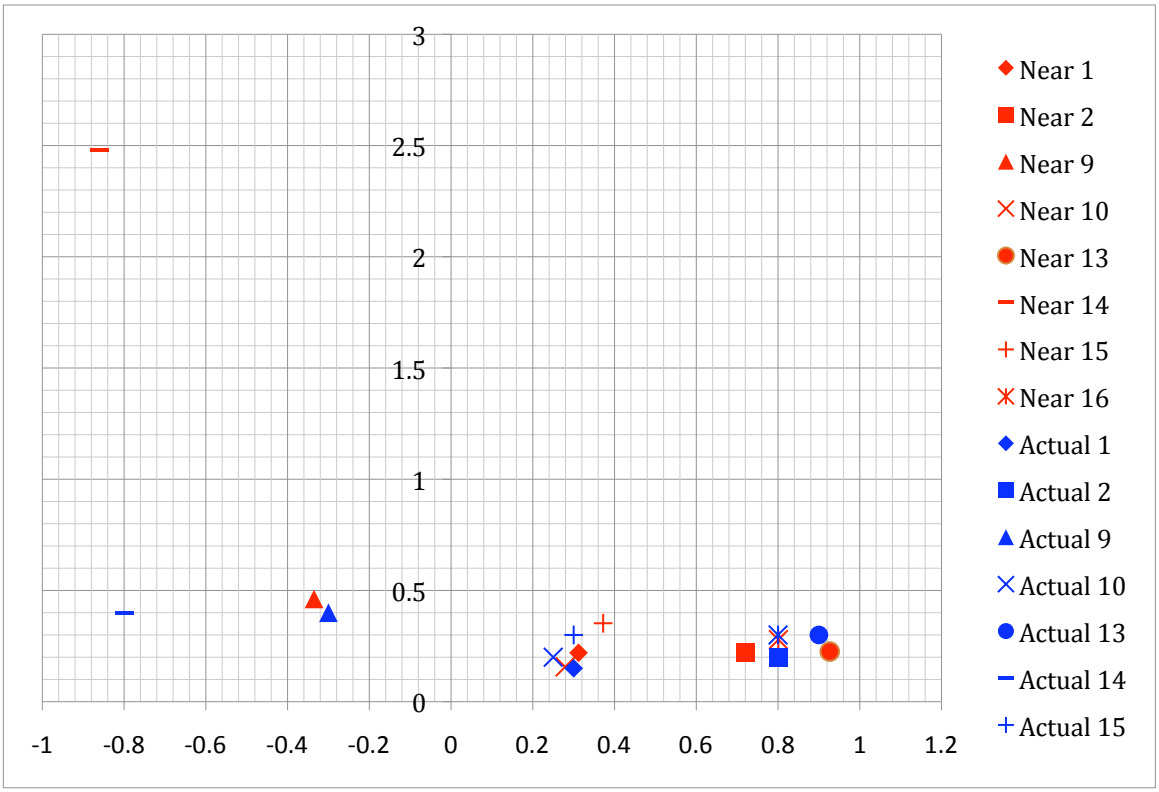

Fig. 11

In Fig. 12, the angles of the near field and far field estimations of the trials shown in Fig. 11 are compared with the actual angle between source and reference microphone. The far field estimation has poor accuracy in many of the trials due to the proximity of the source.

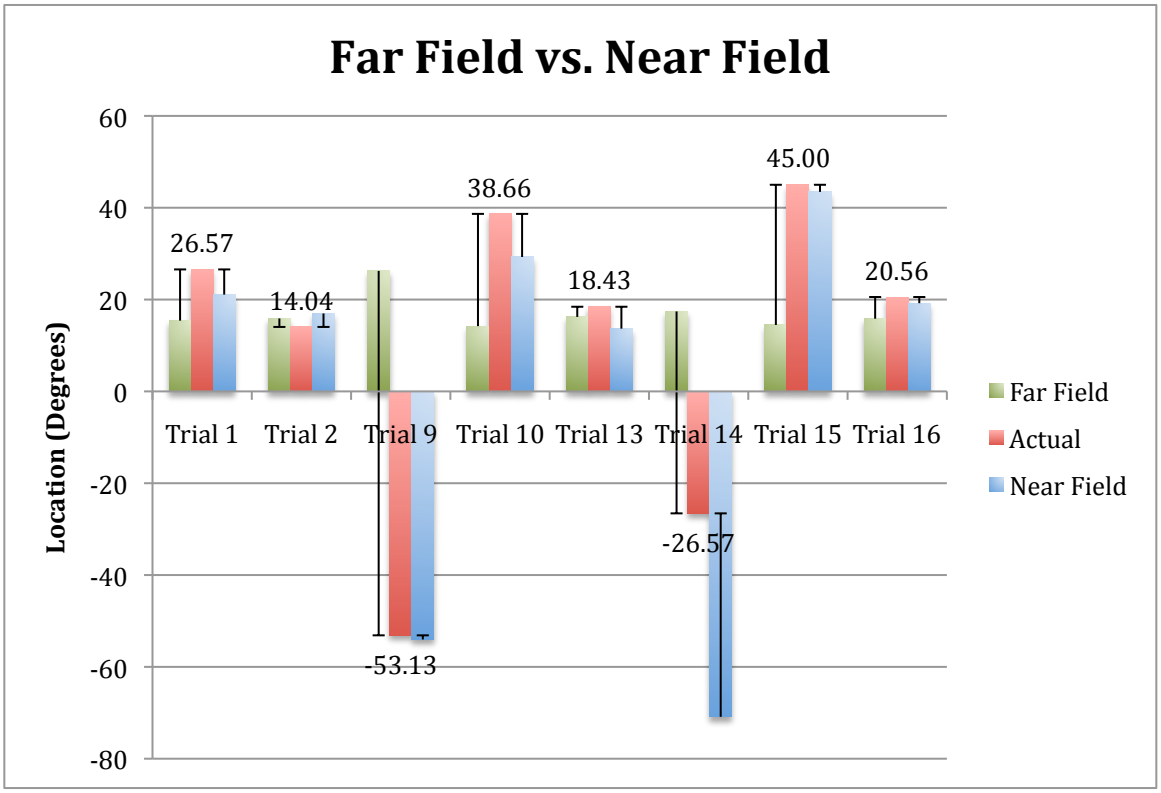

Fig. 12

4.2. RANGE-ACCURACY DEGRADATION

It is easy to show that the accuracy of the near field solution degrades linearly with the distance from the array to the point-source. Consider a source that is a distance L. For simplicity, assume that the distance is normal to the array. Suppose that the microphone farthest from the center of the array is at a distance K, so that L and K form a right triangle with a hypotenuse of length $\sqrt{L^2 + K^2}$. The differential time delay between the two distances can then be estimated using the differential distance, as shown in Eqn. 10.

$$d = \sqrt{L^2 + K^2} - L$$

$$d = L\sqrt{1 + K^2/L^2} - L$$

$$d \cong \frac{K^2}{2L} \quad (a)$$

$$\Delta t \cong \frac{K^2}{2L} \times \frac{1}{c}$$

Eqn. 10

Step (a) arises from the use of a Taylor series assuming that $K^2/L^2 << 1$, used to make the math more tractable. Suppose that $K = 0.5$ m and $K = 5$ m, the

approximated function gives $d \cong 0.025$ m and $\Delta t \cong 72.89$ μs. This is between three and four sampling periods at a sampling rate of 48 kHz, which appears reasonable, but we want to be able to reliably differentiate a range of 5 m from say, 5.05 m (1% range resolution). The difference in differential time delays between 5 m and 5.05 m is only $\Delta(\Delta t) \cong 0.722$ μs. This $\Delta(\Delta t)$ can be approximated with a Taylor series as shown in Eqn. 11 and a graph of difference of differential time delay versus source range L versus is shown in Fig. 13. Note that the graph assumes $K = 0.5$ m for simplicity.

$$\Delta(\Delta t) \cong \frac{\partial(\Delta t)}{\partial L} \times \Delta L$$

$$\Delta(\Delta t) \cong \frac{-\Delta t}{L} \times \Delta L$$

Eqn. 11

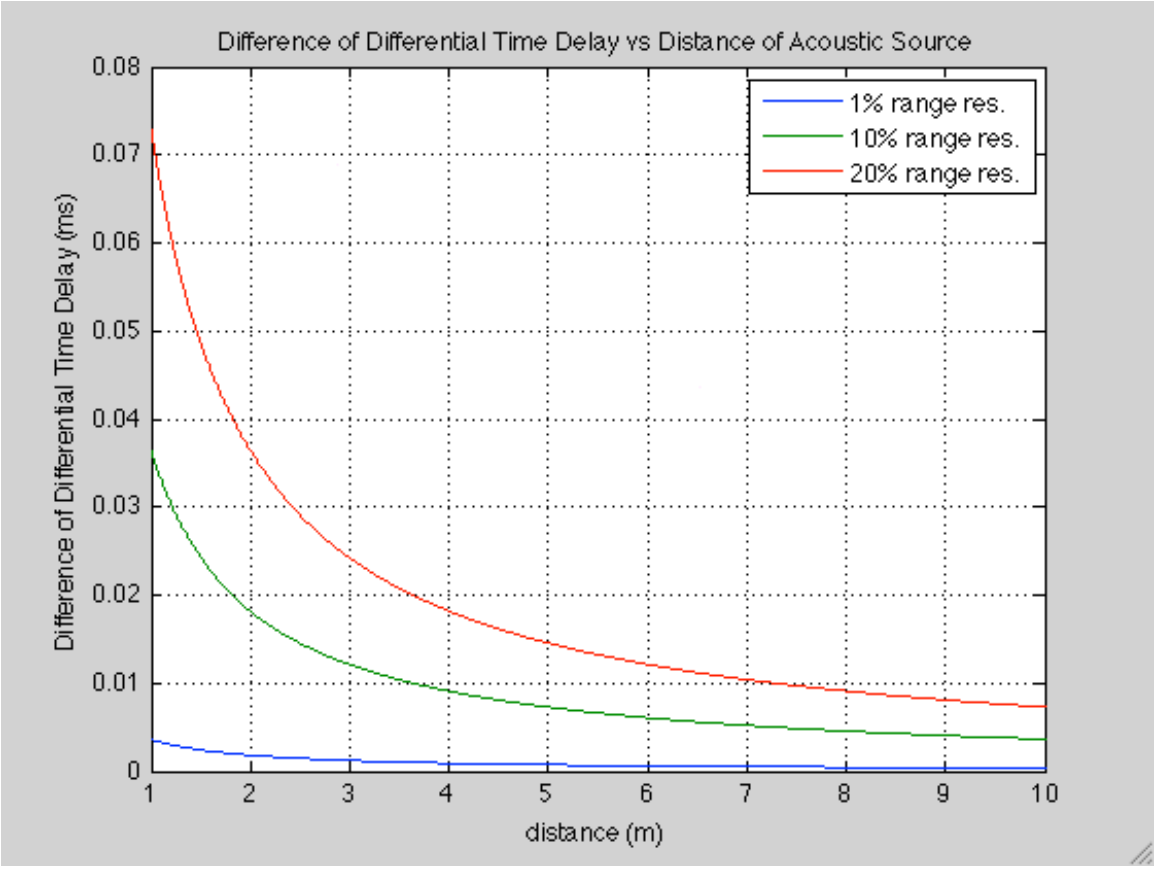

Fig. 13

4.3. PRACTICAL LIMITATIONS

4.3.1. Sampling Time

A sampling rate of 48 kHz means that the TDE values can only be accurate up to 20.83 µs; interpolation can decrease this value but obviously cannot eliminate it.

4.3.2. Signal Length

The system only uses signals that are 1024 samples in length, limiting the accuracy of TDE values. It is important to note, though, that even with infinite system memory, a gunshot still only offers a small amount of data, impairing the precision of TDE values.

4.3.3. Matched Filter Length

Like the signal length and the GCC algorithm, the matched filter length affects the accuracy of the real-time detection algorithm.

4.3.4. Array Construction

We discovered higher estimation error for both near and far field solutions when the source is in the plane of the microphone array. Unlike the regions discussed in section 4.1, we postulate that this is due to impediments introduced by the physical structure of the array rather than the matrix created by the source and microphone positions, as these matrices were no closer to being singular than those in regions with low estimation error.

4.4. MISTAKES

Two significant mistakes were made during the project, both involving the matched filter. The amplitude values of the filter and the signal were not normalized, rendering the matched filter ineffective. The filter was also not time-reversed, as is in a convolution; while this doesn't affect the threshold-checking method as much as the first error, it can invalidate more sophisticated detection methods. Unnoticed, both errors hampered development but were finally corrected by the end of the project.

Additionally, a minor GUI error was made in calculating the angle of the far field solution, but this was also corrected by the end of the project.

5. DEMONSTRATION

To demonstrate the project, we simulated gunshots in space around the microphone array and showed viewers the results on the PC. Sounds were made both by clapping hands and clapping blocks of wood, and these sources were positioned both within the region of the array and as far away as 5 m. Solutions were estimated in both the far and near field. A demonstration abstract is shown by Fig. 14.

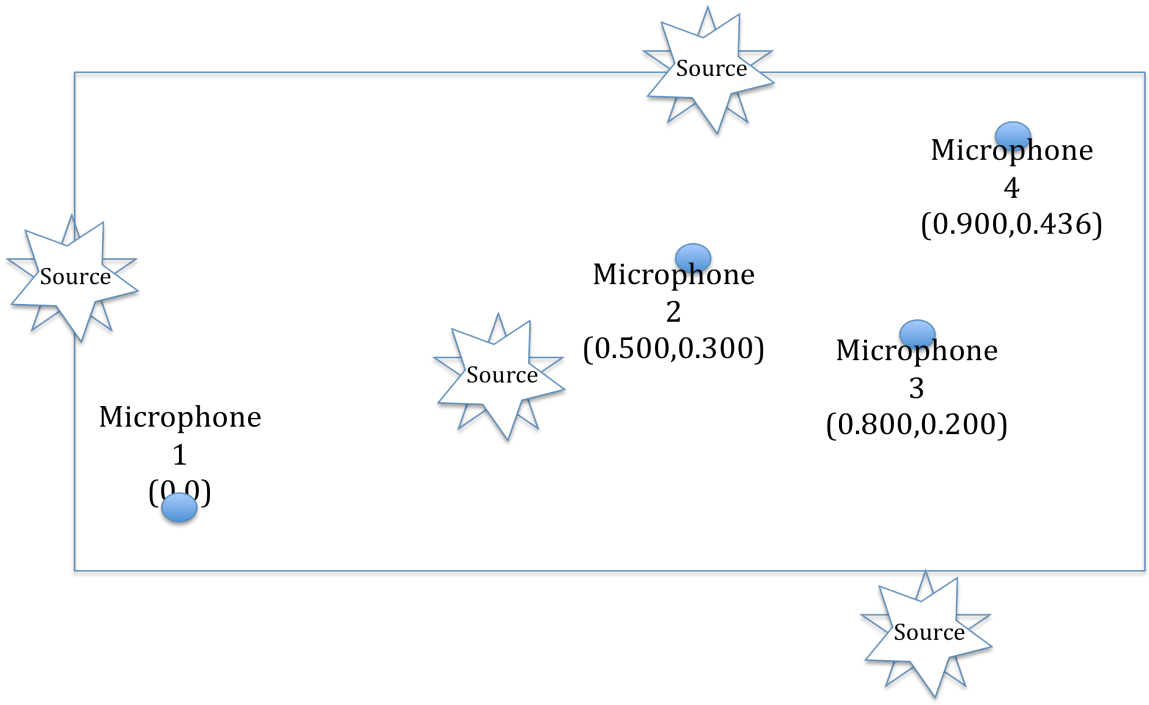

Fig. 14

This is an effective proof of concept for a larger scale product, as the problem scales linearly. The two significant alterations needed in a scale increase would be a change in microphones, as higher quality would be necessary, and a reevaluation of system-induced delays, as these may no longer be negligible. Even if no longer negligible, system-induced delays can be compensated for as long as they are known.

6. FUTURE WORK

Improving localization accuracy is a clear and probable focus for future work. We recommend two methods of doing this, improving the system algorithms and increasing the number of microphones used.

Numerous areas of the system can be improved. The detection process certainly deserves more work and filtering techniques could be made more sophisticated. Other methods of TDE computation could be explored and new interpolation techniques, such as a quadratic best-fit approximation, could be attempted.

With no improvements to the localization algorithms, both near and far field solutions can be made more precise by increasing the number of array microphones. Educational DSP sells a board that allows four audio daughter cards to be input to a single DSK, meaning up to sixteen microphone inputs could be used. This would certainly be useful for multichannel projects, and would certainly require a great deal of new code.

Another improvement in future work could be the development of a more robust method to test the system. Clapping blocks of wood together allows for only a rough estimate of the actual point-source location, and a more sophisticated system could give more accurate data to compare with the system's estimations.

7. SCHEDULE

Week	Task	Person	Remarks
5 Oct	7-10 page proposal	All	Oral and Written Proposals are due
12 Oct	Hardware purchases		Proposal feedback
19 Oct	Review proposal feedback Capture microphone data	All	
26 Oct	Basic USB code to interface DSK and PC	Arda	
26 Oct	GUI – basic interface on PC	Arda	
26 Oct	Implement external hardware and circular buffer Figure out how to collect data from microphone array Resolve synchronization issues	Pranay Oren	
26 Oct	Algorithm specifications for matched filter and GCC and GS algorithms Matlab testing	Oren	
2 Nov	Improve GUI	Arda	
2 Nov	Implement and test localization algorithms	Pranay Oren	
2 Nov	Implement detection algorithms	Pranay Arda	
9 Nov	Test detection algorithms Begin testing final setup	All	
9 Nov	Prepare for oral update	All	Project Oral Updates
16 Nov	All components working Solve possible issues / debug	All	
23 Nov	Solve possible issues / debug	All	
30 Nov	Prepare demo	All	Final Orals in class, Demos at night
7 Dec	Final written report	All	Final written reports due

8. REFERENCES

[1] Gillette, Matthew D. and Silverman, Harvey F. “A linear closed-form algorithm for source localization from time-differences of arrival.” IEEE Signal Processing Letter, Vol. 15, 2008.

[2] “MASLA: Microphone Arrays for Source Location Applications.” Group 1, Spring 2004, 18-551.

[3] “Find That Sexy Noise: An Exploration into Acoustical Location.” Group 4, Fall 2007, 18-551.

[4] Knapp, Charles H. and Carter, G. Clifford. “The generalized correlation method for estimation of time delay.” IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-24, No. 4, 320-327 (1976).

[5] Lyons, Rick. “How to Interpolate in the Time Domain by Zero-Padding in the Frequency Domain.” 1999. http://www.dspguru.com/dsp/howtos/how-to-interpolate-in-time-domain-by-zero-padding-in-frequency-domain

[6] Beranek, Leo L. *Acoustics*. Cambridge: Bolt Beranek and Newman, Inc. 1954.

[7] The Freesound Project. 2009. http://www.freesound.org/

[8] “ATR3350 Condenser Lavalier Omnidirectional Microphone.” Audio-Technica. 2009.
http://www.audio-technica.com/cms/wired_mics/9c6eca17168eef6f/index.html

[9] “DSK_AUDIO4.” Educational DSP. 2009.
http://www.educationaldsp.com/stockproduct_dsk_audio4.htm

[10] “DSK6XXXHPI Daughtercard.” Educational DSP. 2009.
http://www.educationaldsp.com/stockproduct_dsk6xxxhpi.htm