

Audio Localization

By:

Elizabeth Gregory

Joseph Cole

Audio Localization

By:

Elizabeth Gregory

Joseph Cole

Online:

< <http://cnx.org/content/col10250/1.1/> >

C O N N E X I O N S

Rice University, Houston, Texas

This selection and arrangement of content as a collection is copyrighted by Elizabeth Gregory, Joseph Cole. It is licensed under the Creative Commons Attribution 1.0 license (<http://creativecommons.org/licenses/by/1.0>).

Collection structure revised: December 15, 2004

PDF generated: October 30, 2009

For copyright and attribution information for the modules contained in this collection, see p. 33.

Table of Contents

1	Audio Localization Problem Motivation and Goal	1
2	Beamforming Theory	3
3	Design Decisions for Audio Localization Implementation	9
4	MATLAB Simulation of Audio Localization	11
5	Hardware Implementation for Audio Localization	13
6	Software Implementation of Audio Localization	17
7	Results and Discussion on Audio Localization	19
8	Conclusions on Audio Localization Project	23
9	Appendix for Audio Localization Project	25
	Index	32
	Attributions	33

Chapter 1

Audio Localization Problem Motivation and Goal¹

1.1 Possible Application Areas

Audio localization has quite a few applications in the real world. In particular, such a system can be used in a home automation system to identify a person's location. Also, this project has applications in sonar and radar systems.

1.2 Objective

In this project we would like to:

- Accurately determine the origin of a sine wave to within 22.5°
- Adjust to real-time change in the signal location

However, first we need to set a few parameters. Firstly, our field-of-view is the half plane in front of the array and only deals with the azimuth, not the elevation. Secondly, as we will learn in Beamforming Theory, the signal will have to originate in the far-field. Figure 1.1 (The Setup) shows how our set-up will work.

¹This content is available online at <http://cnx.org/content/m12512/1.2/>.

The Setup

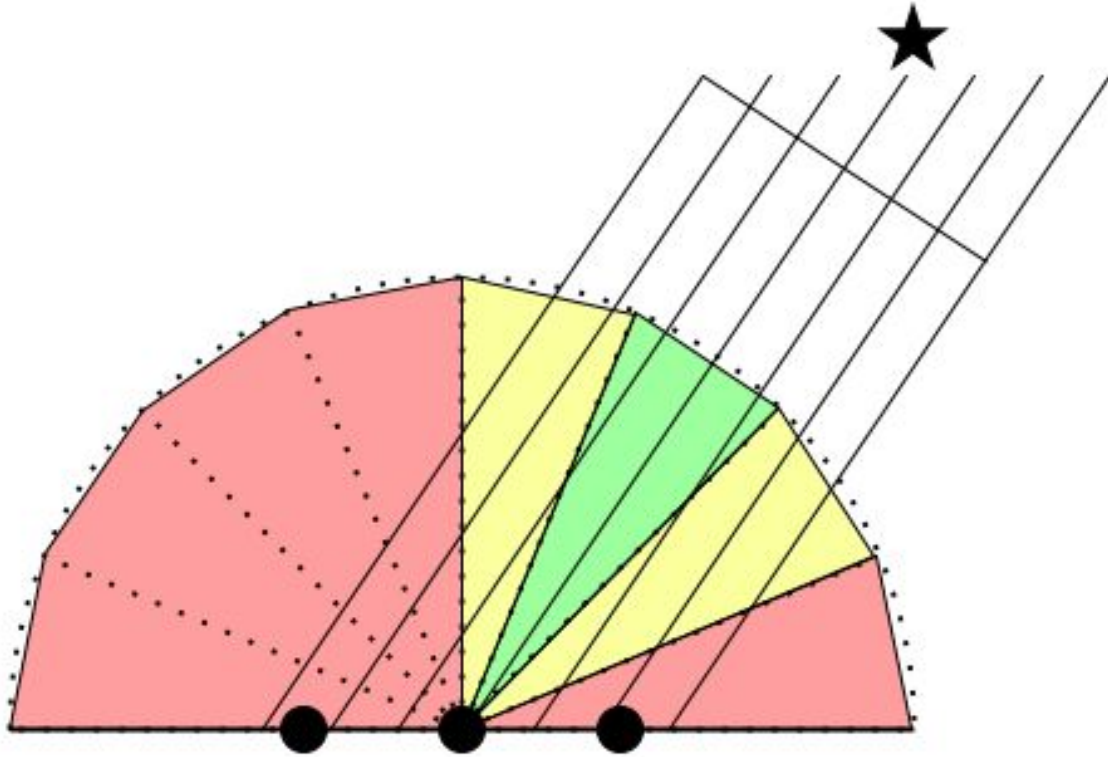


Figure 1.1: The star represents our far-field source. The black circles represent microphones. The area shaded in green represents the correct region. The areas shaded in yellow represent our margin of error. The areas shaded in red represent the incorrect areas.

Chapter 2

Beamforming Theory¹

The Geometry

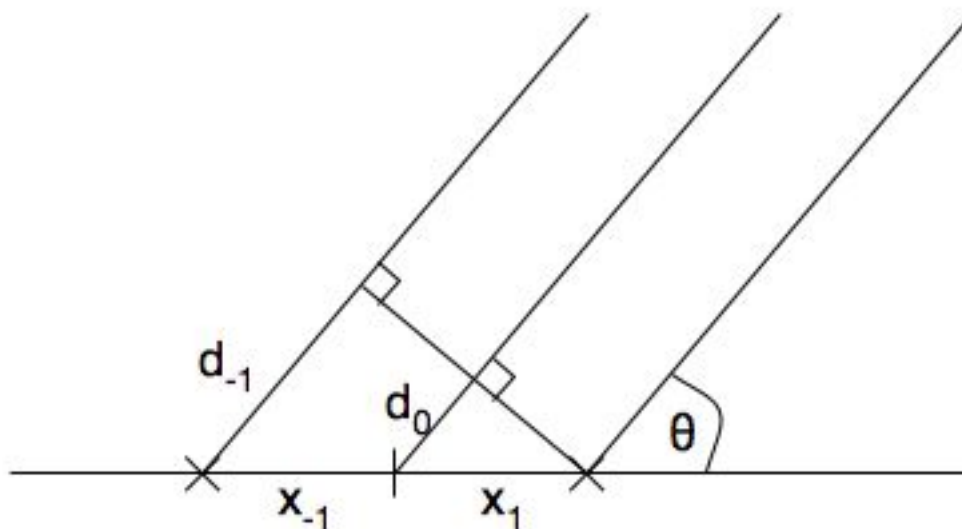


Figure 2.1

We used delay-and-sum beamforming in order to determine the direction of origin for our 500 Hz test signal. Beamforming takes advantage of the fact that the distance from the source to each microphone in the array is different, which means that the signal recorded by each microphone will be phase-shifted replicas of each other. The amount of phase-shift at each microphone in the array can be calculated by thinking about the geometry of the situation, shown in Figure 2.1 (The Geometry). In our case, we are assuming that the source is in the far-field, which means that the source is far enough away that its spherical wavefront appears planar at our array. The geometry is much simpler with that assumption, and (2.1) shows the calculation for the extra time it takes to reach each microphone in the array relative to the array center. Figure 2.2 (Out of

¹This content is available online at <http://cnx.org/content/m12516/1.1/>.

Phase Signals As Seen by a 3-Microphone Array) shows an example of the out of phase signals that might be recorded by a three microphone array.

$$\Delta_m = \frac{x_m \cos(\theta)}{c} \quad (2.1)$$

Out of Phase Signals As Seen by a 3-Microphone Array

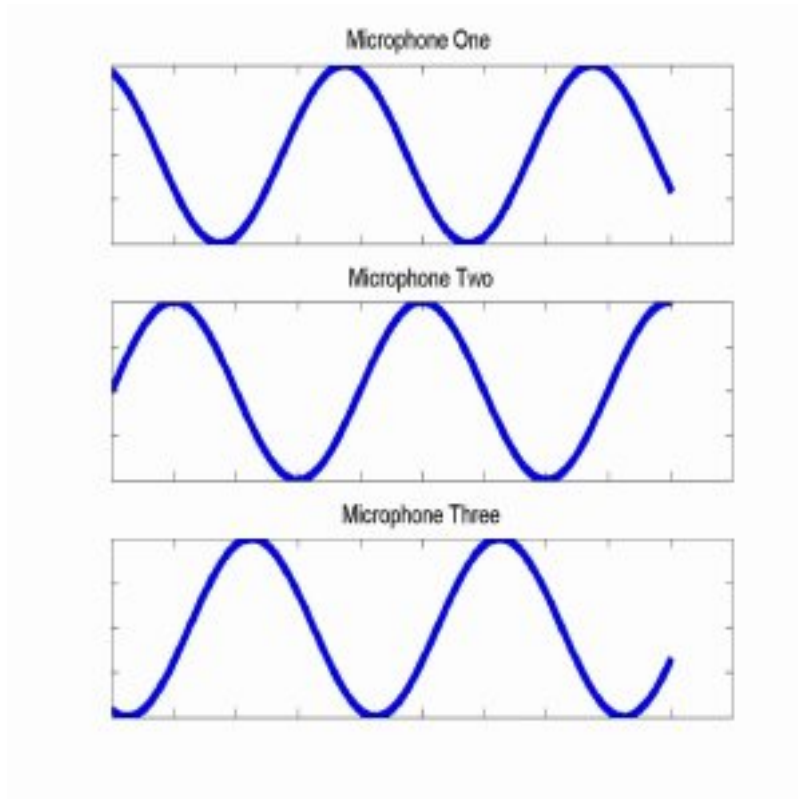


Figure 2.2

In order to determine the direction of origin of a signal, we have to add a time delay to the recorded signal from microphone that is equal and opposite of the delay caused by the extra travel time. That will result in signals that are perfectly in-phase with each other. Summing these in-phase signals will result in constructive interference that will amplify the result by the number of microphones in the array. The question is how to know what time delay to add that will produce the desired constructive interference. The only solution is to iteratively test time delays for all possible directions. If the guess is wrong, the signal will destructively interfere resulting in a diminished output signal, but the correct guess will result in the signal amplification described above.

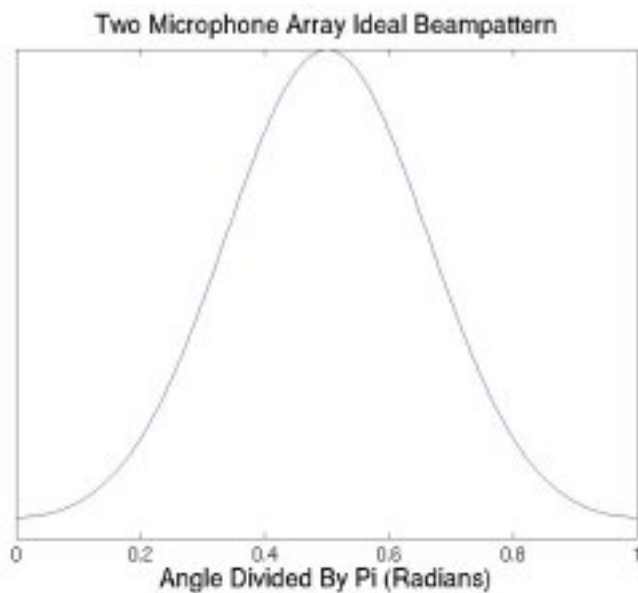


Figure 2.3: The beampattern for a signal arrive from $\pi/2$, as seen by a two-microphone array.

We can plot the resulting output amplitudes as a function of test angles to produce a beampattern for the array. A typical beampattern for a signal arriving from the $\frac{\pi}{2}$ direction is shown in Figure 2.3 for a two microphone array. Naturally, the peak is located at $\frac{\pi}{2}$ because time delays from that region produced the most constructive interference. Test values further from the true angle resulted in diminished output signals. If the source originates from a different direction, such as $\frac{\pi}{3}$ as shown in Figure 2.4, the peak moves to the new location.

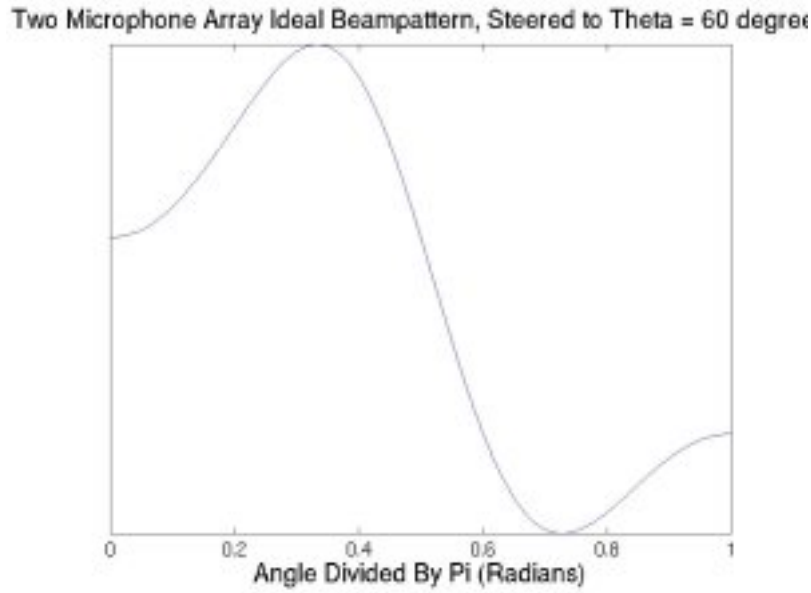


Figure 2.4: The comparison of a beampattern for a two-microphone array when at $\pi/3$.

The peak width is partially determined by the spacing of the microphones in the array. Figure 2.5 shows that as the spacing is increased, the peak width decreases. That trend will continue until the array length reaches the optimal length for the source frequency used. This length is half the wavelength of the source signal as shown in the Design Decisions section.

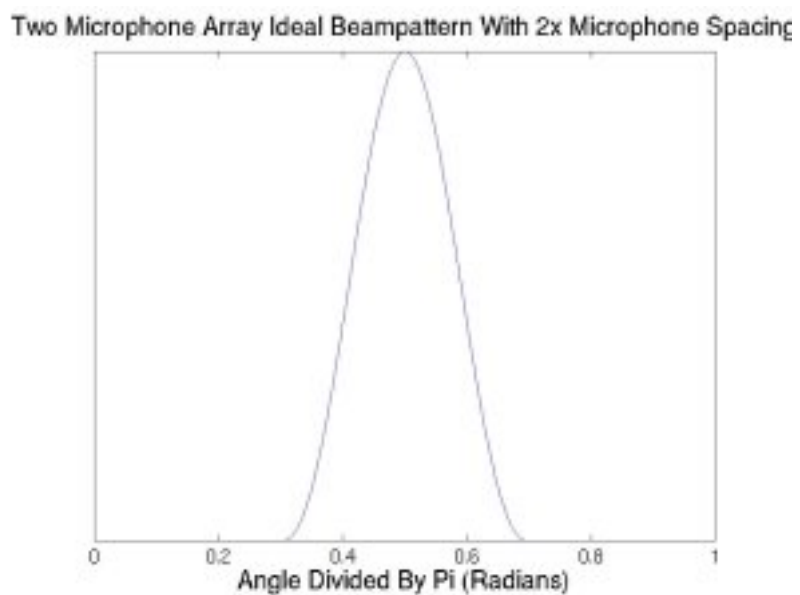


Figure 2.5: Beampattern with an increased array spacing.

Figure 2.6 shows the affect of adding more microphones to the array. The most interesting feature is the appearance of side lobes in the beampattern. However, the global peak value is still located at the true origination angle.

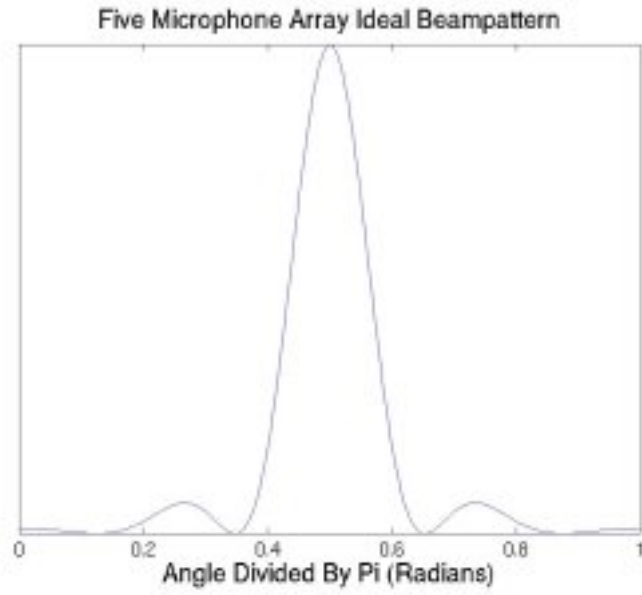


Figure 2.6: Beam pattern with more microphones

Chapter 3

Design Decisions for Audio Localization Implementation¹

With the theory out of the way, we have to face the real world and set more constraints.

3.1 The Number of Microphones

In this project, we are using the TI TMS320C6211 DSK board. This board has two channels that sample at 48 kHz and another channel that samples at 8 kHz. Since we are not interpolating our signals, the sampling frequency is increasingly critical, so we can only use **two microphones**. Figure 3.1 (Beampattern with Reduced Sampling Frequency) shows what happens to the beampattern when we use reduced sampling frequency.

¹This content is available online at <<http://cnx.org/content/m12513/1.4/>>.

Beampattern with Reduced Sampling Frequency

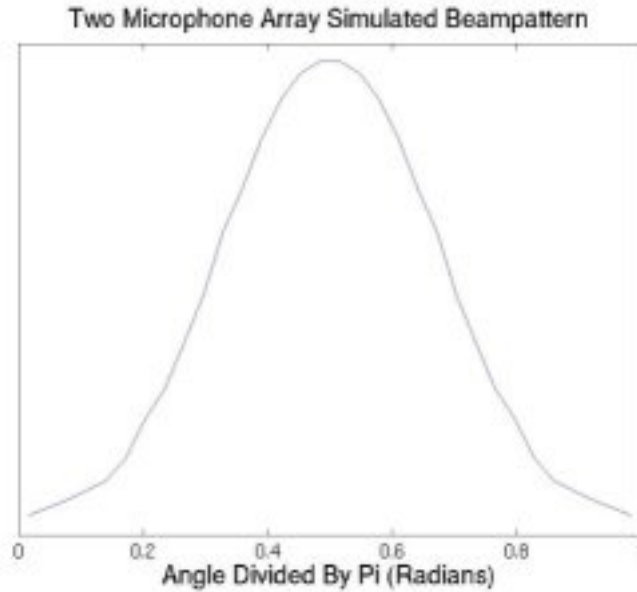


Figure 3.1

3.2 Array Spacing

In order to find the best array spacing (theoretically), we have to go through a few calculations. In our most extreme case, the signals will be hitting our array from 180° . This gives us perfect destructive interference, as our phase difference will be π . Therefore, we only need half of the wavelength:

$$d = \frac{\lambda}{2} \quad (3.1)$$

We can find λ by dividing the frequency of our sine wave (500 Hz) by the speed of sound (346.287 m/s):

$$\begin{aligned} \lambda &= \frac{c}{f} \\ &= \frac{346.287}{500} \\ &= 0.69 \end{aligned} \quad (3.2)$$

and our array spacing d ends up being 0.345 m.

Chapter 4

MATLAB Simulation of Audio Localization¹

$$\sin(\omega(t - \tau_2) + \phi_2) + \sin(\omega(t - \tau_1) + \phi_1) = 2\sin\left(\omega t + \frac{\phi_1 + \phi_2 - \omega(\tau_1 + \tau_2)}{2}\right) \cos\left(\frac{\phi_1 - \phi_2 - \omega(\tau_1 - \tau_2)}{2}\right) \quad (4.1)$$

(4.1) shows that the sum of two sinusoids that are out of phase is just another sinusoid with an amplitude directly related to the phase difference. Our goal is to adjust the phase difference by adding time delays to the incoming signals so as to maximize the amplitude of the output. The maximum occurs when the phase difference is zero, because the signals will add constructively. Once the maximum is found, the time delays used to achieve it tell us from which direction the signals originated.

Since we are working with discrete-time signals, the time delays we tried were limited by the sampling frequency of the DSP boards, which is 48 kHz. By dividing the desired time delay by that sampling period and rounding to the nearest integer, we converted our trial time delays into indices that could be used to select the correct sample out of the buffer.

So, the algorithm for delay-and-sum beamforming is straightforward, but there is room for a little bit of creativity in finding the amplitude of the summed sinusoids. We experimented with two methods to accomplish that task. We will call the first method "amplitude extraction," and the second "signal integration."

¹This content is available online at <<http://cnx.org/content/m12510/1.2/>>.

Amplitude Extraction Flow Diagram

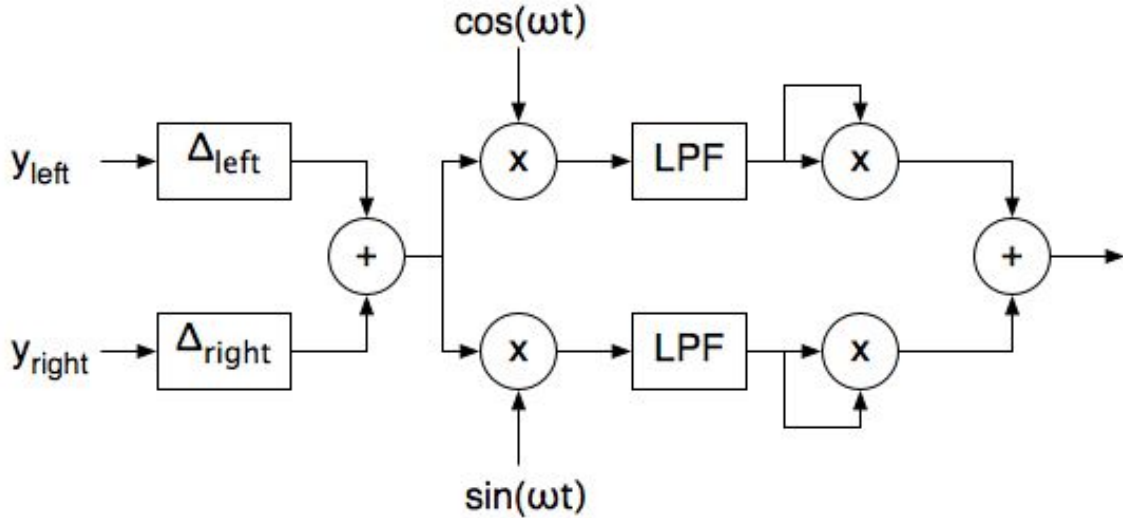


Figure 4.1

A flow diagram for the amplitude extraction method is shown in Figure 4.1 (Amplitude Extraction Flow Diagram). We split the signal into two parts and multiply one part by $\sin(\omega t)$ and the other by $\cos(\omega t)$. By low-pass filtering the results, we obtain the DC part of the signal which contains the amplitude information. This algorithm has the obvious disadvantage that it is dependent on knowing the frequency of the incoming signal so that the correct w is used in the multiplication step. In spite of that, we were originally selected it for implementation on the DSP board because it showed extremely robust performance in the presence of loud noise. Adding gaussian white noise with a variance of 1 to a signal in the range $[-1, 1]$ had no affect on the performance of the beamformer in our Matlab simulation. Unfortunately, the algorithm is too slow to be used in real time. Evaluating two low-pass filters for every time delay combination tried was simply not practical.

The signal integration method is much simpler computationally, which made it a better choice for our final implementation. We only had to square the beamformer output to make all the numbers positive, and sum the results over approximately one cycle of the incoming signal. The sum is similar to an integral over one period of the signal, except that the samples aren't multiplied by the sampling period to make an "area." Our matlab simulation showed that the algorithm should work, but that it is somewhat more sensitive to noise than amplitude extraction.

We were unable to try either of our Matlab simulations with real signals recorded from our microphone array because we had difficulty making stereo recordings. The computers we used defaulted to recording from the microphone input (which is mono) instead of the line-in input, and we didn't have administrator access to change the settings.

Chapter 5

Hardware Implementation for Audio Localization¹

5.1 TI TMS320C6211 DSK Board

You can find a description of the board on the TI website².

To receive the signals, we decided to use the McBSP1 receive interrupt so we could read the values from our two 16 bit channels simultaneously.

¹This content is available online at <<http://cnx.org/content/m12514/1.3/>>.

²<http://www.ti.com>

5.2 Microphone Array

The Microphones and the Circuit

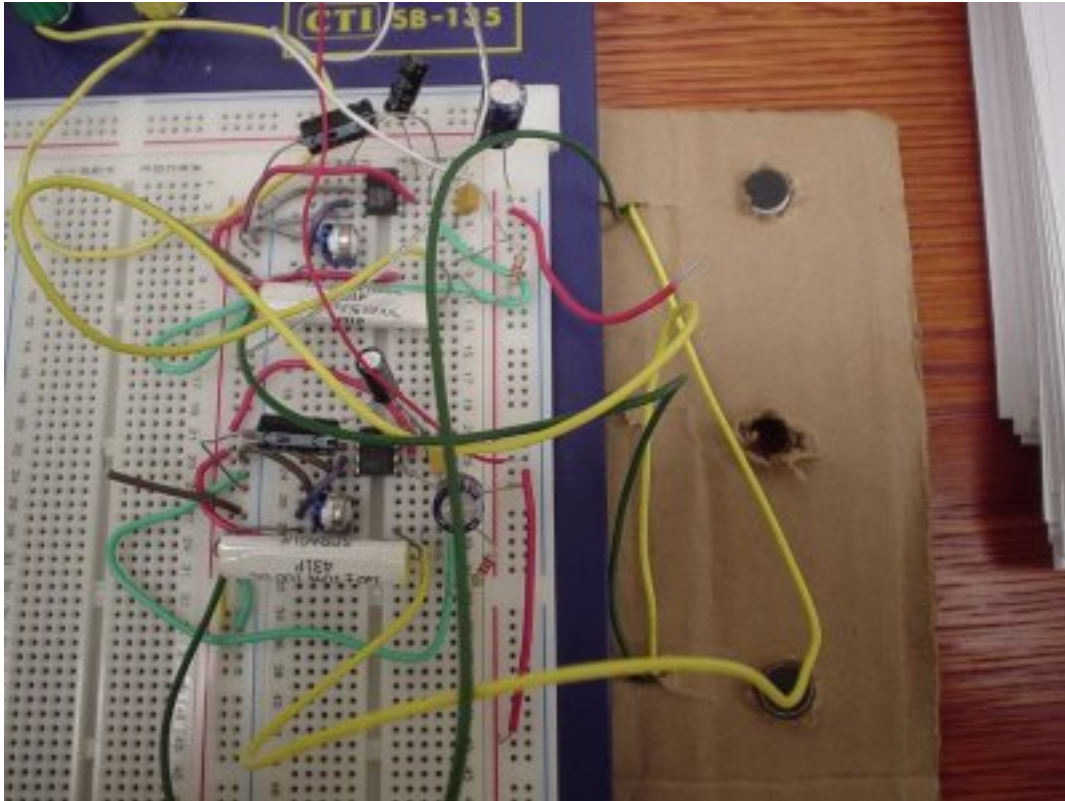


Figure 5.1

As shown in Figure 5.1 (The Microphones and the Circuit), we used two electret microphones spaced 10 cm apart. Each microphone was amplified with identical non-inverting op-amp circuits (LM386³), as shown in Figure 5.2 (The Microphone Amplifier Circuit). This circuit includes a low-voltage audio power amplifier and internal feedback with fixed gain.

³<http://www.njr.co.jp/pdf/de/de05001.pdf>

The Microphone Amplifier Circuit

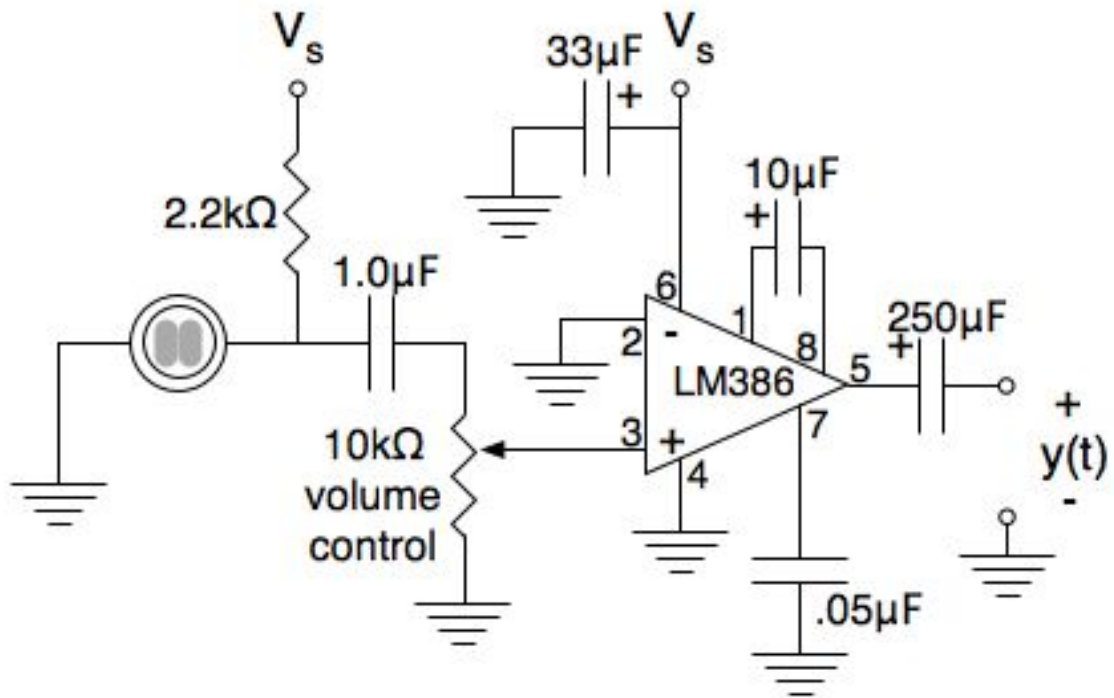


Figure 5.2

Chapter 6

Software Implementation of Audio Localization¹

6.1 Code Description

Our final implementation for the DSP board is written in C, and consists of two major functions, the interrupt service routine and the main function. The samples arrive and are put into a buffer for each microphone channel by the McBSP1 receive interrupt routine. This function keeps track of how full the buffers are, and sets a flag when they are full. The main function waits for the buffer full flag, and processes the contents of the buffers when the flag is set. The algorithm is the same as the signal integration method described in the Matlab Simulation section. The code keeps an average of the last 128 region codes selected, which is the value that is output on the DSP board's LEDs.

6.2 Signal Input Options

For our signal input, we have two options:

1. Ideal Signal generated in Matlab
2. Real signal from the microphone array.

To thoroughly test this algorithm, we tried both options, starting with the ideal signal, since this would help us in debugging. Later, we tried using a real signal, generated from a computer across the room. Unfortunately, within this option lies the danger of acoustics; the answer the algorithm gives us will even depend on the number of people in the room!

¹This content is available online at <<http://cnx.org/content/m12519/1.1/>>.

Chapter 7

Results and Discussion on Audio Localization¹

7.1 Matlab Simulation Results

In running our simulation, we discovered that while our region output was not always correct, we maintained accuracy within 22.5° . The algorithm seemed to have the most difficulty on the edges of the field of view. Our exact results can be found in Table 7.1: Without Noise and Table 7.2: With Noise, SNR=2.

Without Noise

True Region	True Angle	Estimated Region
0	$\frac{\pi}{16}$	0
1	$\frac{3\pi}{16}$	0
2	$\frac{5\pi}{16}$	2
3	$\frac{7\pi}{16}$	3
4	$\frac{9\pi}{16}$	4
5	$\frac{11\pi}{16}$	5
6	$\frac{13\pi}{16}$	6
7	$\frac{15\pi}{16}$	7

Table 7.1

¹This content is available online at <http://cnx.org/content/m12520/1.1/>.

With Noise, SNR=2

True Region	True Angle	Estimated Region
0	$\frac{\pi}{16}$	1
1	$\frac{3\pi}{16}$	2
2	$\frac{5\pi}{16}$	2
3	$\frac{7\pi}{16}$	2
4	$\frac{9\pi}{16}$	6
5	$\frac{11\pi}{16}$	7
6	$\frac{13\pi}{16}$	6
7	$\frac{15\pi}{16}$	6

Table 7.2

7.2 Results with the DSK

In testing our DSK algorithm, we started with the ideal signal, generated using Matlab. The results had more error than in Matlab, but seemed reasonable at the time. However, in multiple trials of the same ideal signal, the DSK responded differently each time, indicating that our algorithm still needed a bit of work.

When we tried to implement the same algorithm using a real signal, generated from a computer across the room, we received very poor results. In the end, the DSK could tell whether the signal came from the left or the right, but only when the lab was quiet and empty. Also, the program was very sensitive to alternate signal paths and the general acoustics of the room. All in all, our program was not very reliable, as shown in Table 7.3: C Results.

C Results

True Region	True Angle	Estimated Region
0	$\frac{\pi}{16}$	4
1	$\frac{3\pi}{16}$	3
2	$\frac{5\pi}{16}$	6
3	$\frac{7\pi}{16}$	3
4	$\frac{9\pi}{16}$	5
5	$\frac{11\pi}{16}$	3
6	$\frac{13\pi}{16}$	4
7	$\frac{15\pi}{16}$	6

Table 7.3

7.3 Areas of Improvement

All in all, we need to start the improvement by doing a better job designing and checking the array beam pattern. Also, we need a better algorithm for integration. For this algorithm, we need to know how much of a cycle is needed, how integration length affects accuracy, and how to deal with non-periodic signals. Finally,

we need a better understanding and control over the room acoustics, as well as more time to fully test the algorithm.

Chapter 8

Conclusions on Audio Localization Project¹

While in Matlab, our Objectives seemed pretty reasonable, in the end, this project seems to be far beyond the scope of an end-of-the semester project. While we were able to determine while side the signal was coming from, we did not get anything near the resolution we were expecting. From here, we need to re-analyze our algorithms, and go through the theory again, perhaps. Also, we would need to look into acquiring a "control" room, whose acoustics we would be able to account for.

However, in the end, we learned quite a few lessons:

- Always double-check your math.
- Don't try to simplify your equations too soon.
- Sometimes, you really **do** need to just shut down the program and restart.

¹This content is available online at <<http://cnx.org/content/m12515/1.2/>>.

Chapter 9

Appendix for Audio Localization Project¹

9.1 Matlab Code

- new_sim2:

```
function new_sim2(theta_true)
% Values, Vectors, and a Matrix
%theta_true = pi./2;
degree = 32;

c = 346.287;           % speed of sound in air
N = 150;              % length of the sample buffer
Fs = 48000;           % sampling frequency
f = 500;              % frequency of sine wave
M = 2;               % number of microphones
dist = .1;
t = [0:N-1]./Fs;      % time axis
m = (M-1)./2;         % array center
x = dist.*[-m:m];     % microphone location on the x axis
omega = 2*pi*f;       % commonly used value

theta_test = [1:2:2*degree-1]*pi/(2*degree); % test vector of theta values
%theta_test = pi./2;
divisor = degree/8;   % region divisor
length_t = length(theta_test); % length of the delay vector
A = zeros(1,length_t); % initialize A vector

delay_true = x.*cos(theta_true)./c; % actual delay
delay_test = x'*cos(theta_test)./c; % test matrix of delay values
samples = round(2.*delay_test*Fs)./2; % number of samples to shift in testing
index = samples - ones(M,1)*min(samples) + 1;

% Signal Simulation
```

¹This content is available online at <http://cnx.org/content/m12517/1.1/>.

```

for j = 1:M
    y(j,:) = sin(omega*(t-delay_true(j)));
end

% Region Approximation

for i = [1:length_t]
    for j = [1:M]
        y_delay(j,:) = y(j,[index(j,i)+50:index(j,i)+100]);    %delay y1 by the 1,i value using in
    end
    z = sum(y_delay);

    A(i) = sum(z.^2);
end

aa = find(A == max(A));
region = floor(aa(1)./divisor)
theta_range = [region-1 region]*pi/8;
if(0)
figure
plot(theta_test/pi,A)
end

```

- sim_input3:

```

function [region,theta_range] = sim_input3(theta_true,degree)

% Values, Vectors, and a Matrix

c = 346.287;           % speed of sound in air
N = 150;              % length of the sample buffer
Fs = 44100;          % sampling frequency
f = 500;             % frequency of sine wave
M = 2;              % number of microphones
%dist = .32;         % distance between microphones
dist = .5;
t = [0:N-1]./Fs;     % time axis
m = (M-1)./2;       % array center
x = dist.*[-m:m];   % microphone location on the x axis
cutoff = 50;        % cutoff frequency of filter
omega = 2*pi*f;     % commonly used value
theta_test = [1:2*degree-1]*pi/(2*degree); % test vector of theta values
divisor = degree/8; % region divisor
length_t = length(theta_test); % length of the delay vector
A = zeros(1,length_t); % initialize A vector
B = fir1(40,cutoff/Fs,'low'); % lowpass filter
delay_true = x.*cos(theta_true)./c; % actual delay
delay_test = x.*cos(theta_test)./c; % test matrix of delay values
samples = round(2.*delay_test*Fs)./2; % number of samples to shift in testing
index = samples - ones(M,1)*min(samples) + 1;
cos_base = cos(omega*t(1:N));

```



```

#include "index.h"

#define N 100 //sample buffer length
#define N2 128 //averaging buffer length
#define DEGREE 32 //number of theta test values to try
#define SUMSAMP 70 //number of theta test values to try

float Fs = 16000.0; //irrelevant since jumper in 3-4

int* lights = (int*)0x90080000;

short buffer1[N] = {0};
short buffer2[N] = {0};
int buffer3[N2] = {0};
int buf_full = 0;
int buf_index = 0;

interrupt void c_int12()      //McBSP1 receive ISR
{

int sample = input_leftright_sample();

    buffer1[buf_index] = (short)(sample >> 16);
    buffer2[buf_index] = (short)sample;

buf_index++;
if(buf_index == N) {
buf_index = 0;
buf_full = 1;
}

return; //return from interrupt
}

void main()
{
int i;
int j;
int k = 0;
int z, test_amp;
int region;
int out;
int sum = 0;

int max_theta_index = 0;
int max_amplitude = -1;

for(k=0;k<N2;k++) buffer3[k] = 0;
k = 0;

```

```

comm_intr();          //init DSK, codec, McBSP
    while(1) {
if(buf_full) {
buf_full = 0;
max_amplitude = -1;

    for(i=0; i<DEGREE; i++) {
test_amp = 0;

for(j=0; j<SUMSAMP; j++) {
    z = buffer1[index[0][i]+j] + buffer2[index[1][i]+j];
test_amp += (z * z) >> 15;
    }

    if(test_amp > max_amplitude) {
max_amplitude = test_amp;
max_theta_index = i;
}
    }

    region = max_theta_index >> 2;
sum -= buffer3[k];
buffer3[k++] = region;
sum += region;

    if(k == N2) {
k = 0;
out = sum >> 7;
/*if(out > 3)
out = 0;
else
out = 7;*/
*lights = out << 24;
}
}
}

```

- vectors_11.asm:

*Vectors_11.asm Vector file for interrupt-driven program

```

.ref _c_int12    ;ISR used in C program
.ref    _c_int00 ;entry address
.sect    "vectors" ;section for vectors
RESET_RST: mvl .S2 _c_int00,B0 ;lower 16 bits --> B0
    mvkh .S2 _c_int00,B0 ;upper 16 bits --> B0
    B .S2 B0 ;branch to entry address
NOP      ;NOPs for remainder of FP
NOP ;to fill 0x20 Bytes

```

```
NOP
NOP
NOP
NMI_RST: .loop 8
NOP      ;fill with 8 NOPs
.endloop
RESV1: .loop 8
NOP
.endloop
RESV2: .loop 8
NOP
.endloop
INT4: .loop 8
NOP
.endloop
INT5: .loop 8
NOP
.endloop
INT6: .loop 8
NOP
.endloop
INT7: .loop 8
NOP
.endloop
INT8: .loop 8
NOP
.endloop
INT9: .loop 8
NOP
.endloop
INT10: .loop 8
NOP
.endloop

INT11: .loop 8
NOP
.endloop

INT12: b _c_int12 ;branch to ISR
.loop 7
NOP
.endloop

INT13: .loop 8
NOP
.endloop
INT14: .loop 8
NOP
.endloop
INT15: .loop 8
NOP
```

```
.endloop
```

9.3 Project Executable Files

- `project.out`²
- `project.paf`³
- `project.pjt`⁴

²<http://cnx.org/content/m12517/latest/project.out>

³<http://cnx.org/content/m12517/latest/project.paf>

⁴<http://cnx.org/content/m12517/latest/project.pjt>

Index of Keywords and Terms

Keywords are listed by the section with that keyword (page numbers are in parentheses). Keywords do not necessarily appear in the text of the page. They are merely associated with that section. *Ex.* apples, § 1.1 (1) **Terms** are referenced by the page they appear on. *Ex.* apples, 1

- | | |
|------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------|
| A appendix, § 9(25) | H hardware, § 5(13) |
| B beamforming, § 1(1), § 2(3), § 3(9), § 4(11),
§ 5(13), § 6(17), § 7(19), § 8(23), § 9(25) | L LM386D, § 9(25) |
| C C, § 6(17)
code, § 9(25) | M Matlab, § 4(11)
microphone, § 5(13) |
| D design, § 3(9)
digital signal processing, § 2(3)
digital single processing, § 1(1)
DSP, § 2(3), § 7(19), § 8(23) | R real-time, § 2(3), § 6(17)
results, § 7(19) |
| F far-field, § 1(1) | S simulation, § 4(11)
software, § 6(17) |
| | T TMS320C6211, § 5(13)
trigonometric identities, § 9(25) |

Attributions

Collection: *Audio Localization*

Edited by: Elizabeth Gregory, Joseph Cole

URL: <http://cnx.org/content/col10250/1.1/>

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Audio Localization Problem Motivation and Goal"

By: Elizabeth Gregory, Joseph Cole

URL: <http://cnx.org/content/m12512/1.2/>

Pages: 1-2

Copyright: Elizabeth Gregory, Joseph Cole

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Beamforming Theory"

By: Elizabeth Gregory, Joseph Cole

URL: <http://cnx.org/content/m12516/1.1/>

Pages: 3-8

Copyright: Elizabeth Gregory, Joseph Cole

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Design Decisions for Audio Localization Implementation"

By: Elizabeth Gregory, Joseph Cole

URL: <http://cnx.org/content/m12513/1.4/>

Pages: 9-10

Copyright: Elizabeth Gregory, Joseph Cole

License: <http://creativecommons.org/licenses/by/1.0>

Module: "MATLAB Simulation of Audio Localization"

By: Elizabeth Gregory, Joseph Cole

URL: <http://cnx.org/content/m12510/1.2/>

Pages: 11-12

Copyright: Elizabeth Gregory, Joseph Cole

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Hardware Implementation for Audio Localization"

By: Elizabeth Gregory, Joseph Cole

URL: <http://cnx.org/content/m12514/1.3/>

Pages: 13-15

Copyright: Elizabeth Gregory, Joseph Cole

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Software Implementation of Audio Localization"

By: Elizabeth Gregory, Joseph Cole

URL: <http://cnx.org/content/m12519/1.1/>

Page: 17

Copyright: Elizabeth Gregory, Joseph Cole

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Results and Discussion on Audio Localization"

By: Elizabeth Gregory, Joseph Cole

URL: <http://cnx.org/content/m12520/1.1/>

Pages: 19-21

Copyright: Elizabeth Gregory, Joseph Cole

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Conclusions on Audio Localization Project"

By: Elizabeth Gregory, Joseph Cole

URL: <http://cnx.org/content/m12515/1.2/>

Page: 23

Copyright: Elizabeth Gregory, Joseph Cole

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Appendix for Audio Localization Project"

By: Elizabeth Gregory, Joseph Cole

URL: <http://cnx.org/content/m12517/1.1/>

Pages: 25-31

Copyright: Elizabeth Gregory, Joseph Cole

License: <http://creativecommons.org/licenses/by/1.0>

Audio Localization

This course has been created as an introduction to audio localization, and how beamforming can be applied in a real-time environment.

About Connexions

Since 1999, Connexions has been pioneering a global system where anyone can create course materials and make them fully accessible and easily reusable free of charge. We are a Web-based authoring, teaching and learning environment open to anyone interested in education, including students, teachers, professors and lifelong learners. We connect ideas and facilitate educational communities.

Connexions's modular, interactive courses are in use worldwide by universities, community colleges, K-12 schools, distance learners, and lifelong learners. Connexions materials are in many languages, including English, Spanish, Chinese, Japanese, Italian, Vietnamese, French, Portuguese, and Thai. Connexions is part of an exciting new information distribution system that allows for **Print on Demand Books**. Connexions has partnered with innovative on-demand publisher QOOP to accelerate the delivery of printed course materials and textbooks into classrooms worldwide at lower prices than traditional academic publishers.