Developing a Loudspeaker Beamforming Array to Steer Sound in Three-Dimensions

Sarkis Ter Martirosyan[1] and David Bell[1,2]

[1] *Thomas Jefferson High School for Science and Technology (TJHSST)*
[2] *TJHSST Microelectronics Lab Director*

September 27, 2018

Analyzing the performance of acoustic beamforming has ramifications for radio-frequency beamforming, which is a core component of the budding 5th-Generation Wireless Systems standard. By taking advantage of the constructive and destructive properties of sinusoidal signals in the free-field, beamforming makes it possible to direct sound using static emitters. Beamforming can be applied to both acoustic waves (with loudspeaker emitters) and electromagnetic waves (with antenna emitters). In this article, I develop and explore a system that allows for the beamforming of an arbitrary audio signal to solve the problem of personalized sound.

Contents

1 Introduction

The goal of this research was to explore various beamforming algorithms and their performance by developing a mixed hardware-software system for acoustic beamforming. The observations and conclusions drawn from my research on acoustic beamforming are valuable to exploring the personalized sound problem. Furthermore, my acoustic analysis of beamforming algorithms and performance also applies to radio-frequency signals, as the fundamental underlying mathematics is identical for both electromagnetic and acoustic waves, with the only modification being a change from the speed of sound to the speed of light.

1.1 Problem Definition

To develop a system that uses a loudspeaker array to aim sound at a singular "bright point," a location in the three-dimensional free-field where the signal will be present. All the other points in our field to be "dark points," or points where our signal will be absent (Druyvesteyn and Garas, 1997; Elliott et al., 2012).In this research project, we defined the absence of our signal as the absence of any sound (i.e. silence).

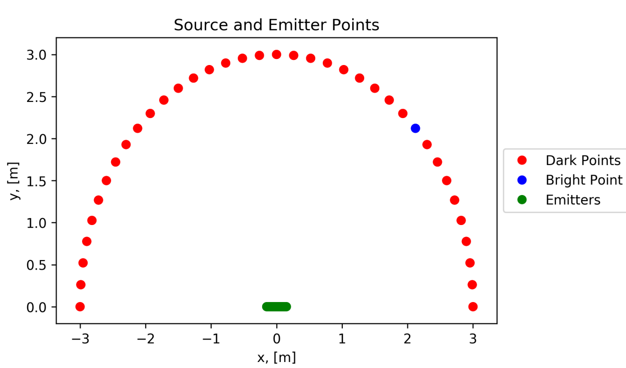

Figure 1

1.2 Mathematical Problem Parameters

The beamforming problem of personalized sound presented in this article is defined by a matrix of M control points $\mathbf{X} \in \mathbb{R}^{3\times M}$ where

$$\mathbf{x}_m = \begin{pmatrix} \rho\cos\frac{\pi(m-1)}{M-1} \\ \rho\sin\frac{\pi(m-1)}{M-1} \\ 0 \end{pmatrix}$$

as the control points are arranged on a semicircle with a radius of ρ meters[1].

Furthermore, a matrix of L emitter points (loudspeakers) is defined as $\mathbf{Y} \in \mathbb{R}^{3\times L}$ where

$$\mathbf{y}_\ell = \begin{pmatrix} (\ell - \frac{L+1}{2})\Delta \\ 0 \\ 0 \end{pmatrix}$$

where Δ is the spacing between loudspeakers in meters[2].

Finally, we can define a frequency-domain representation of a monosignal $\mathrm{s} \in \mathbb{S}_N$, where N is our sampling window[3].

1.3 Complex Mathematical Quantities

The fundamental matrix transformation that defines beamforming is

$$\mathbf{p}(\omega) = \mathbf{Z}(\omega)\mathbf{q}(\omega)s(\omega) \quad (1)$$

where ω is the angular velocity and is defined as $\omega = 2\pi f$, where f is the temporal frequency (Olivieri et al., 2016).

The vector of target sound pressures $\hat{\mathbf{p}} \in \mathbb{C}^M$ is the same regardless of what ω we evaluate it at, and is

[1]For this research project $M = 37$ and $\rho = 3$

[2]For this research project $L = 16$ and $\Delta = 0.02$

[3]For this research project $N = 1024$, meaning the signal processing is performed on 1k sized windows

$$\hat{\mathbf{p}} = \begin{pmatrix} 0 \\ \vdots \\ p_B \\ \vdots \\ 0 \end{pmatrix} \tag{2}$$

where p_B signifies the bright point. If the beam was aimed at $90°$, B would be set to 19. The value of p_B is 1, signifying that we want sound to be there, while all the dark points are assigned a value of 0, meaning we don't want sound there.

The vector of loudspeaker weights is $\mathbf{q}(\omega) \in \mathbb{C}^L$ and varies based on the frequency. The aim of the beamforming algorithms detailed in Section 2 is to solve for $\mathbf{q}(\omega)$. In practice, I defined a matrix $\mathbf{Q}$, whose columns are values of $\mathbf{q}(\omega)$ at each frequency of a discrete Fourier transform with N bins on a signal sampled at a rate of 44100 kHz.

The quantities $\mathbf{p}(\omega)$ and $\mathbf{q}(\omega)$ are related by the linear transformation in Equation 1. The elements of $\mathbf{Z}(\omega) \in \mathbb{C}^{M\times L}$ are defined as

$$\mathbf{Z}_{m,\ell}(\omega) = Z(\mathbf{x}_m, \mathbf{y}_\ell, \omega) = \frac{e^{-j\frac{\omega}{c}\|\mathbf{x}_m - \mathbf{y}_\ell\|}}{4\pi\|\mathbf{x}_m - \mathbf{y}_\ell\|} \tag{3}$$

where $j = \sqrt{-1}$ and $\|\cdot\|$ represents the l2-norm operator.

2 Algorithms

In my research project, I explored two widely used beamforming algorithms, Delay and Sum ("DAS") and Pressure Matching ("PM").

2.1 Delay and Sum

Delay and Sum employs the use of constructive and destructive interference to produce unitary pressure at the bright point, and is calculated using this linear transformation

$$\mathbf{q}_{DAS}(\omega) = \mathbf{\Gamma}(\omega)\mathbf{z}_B^\dagger(\omega) \tag{4}$$

where $[\cdot]^\dagger$ is the complex conjugate transpose and $\mathbf{z}_B(\omega) \in \mathbb{R}^{1\times L}$ is the row vector of $\mathbf{Z}(\omega)$ corresponding to the bright point (Fink and Prada, 2001). $\mathbf{\Gamma} \in \mathbb{R}^{L\times L}$ is a diagonal matrix of the form

$$\mathbf{\Gamma} = \begin{bmatrix} \gamma_1 & 0 & \dots & 0 \\ 0 & \gamma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \gamma_L \end{bmatrix}$$

where

$$\gamma_\ell = \frac{16\pi^2\|\mathbf{x}_B - \mathbf{y}_\ell\|}{L}$$

.

2.2 Pressure Matching

Save for Delay and Sum, much of modern beamforming is focused on the optimization of a cost function, referred to as "super-directive beamforming." Pressure Matching is a form of super-directive beamforming, making use of normal equations to solve a least-squares problem (Kirkeby and Nelson, 1993).

All the quantities below have a frequency dependence on ω.

In Pressure Matching, there are two quantities that we are trying to minimize:

1. the complex target field error magnitude $\|\mathbf{e}_{PM}\| = \mathbf{e}_{PM}^\dagger \mathbf{e}_{PM}$
2. the input energy to the emitter (loudspeaker) array $E_\mathbf{q}$

These quantities are defined as follows.

$$\mathbf{e}_{PM} = \hat{\mathbf{p}} - \mathbf{p} = \hat{\mathbf{p}} - \mathbf{Z}\mathbf{q} \tag{5}$$

$$E_\mathbf{q} = \mathbf{q}^\dagger\mathbf{q} \tag{6}$$

The final cost-minimization relationship is defined as

$$\min_{\mathbf{q}} J_{PM} = \min_{\mathbf{q}}(\mathbf{e}_{PM}^{\dagger}\mathbf{e}_{PM} + \beta_{PM}E_{\mathbf{q}}) \qquad (7)$$

where β_{PM} is a Tikhnonov regularization parameter. The solution to this minimization problem statement is

$$\mathbf{q}_{PM} = (\mathbf{Z}^{\dagger}\mathbf{Z} + \beta_{PM}\mathbf{I})^{-1}\mathbf{Z}^{\dagger}\hat{\mathbf{p}} \qquad (8)$$

2.3 Predicted Algorithm Performance

The two algorithms explored in my research have countering areas of strength and weakness. Delay and Sum has a very low computational complexity and is guaranteed to preserve the integrity of the signal at the bright point. This consistency and speed comes at the price of a broader, less well-directed beam. Pressure Matching, on the other hand, suffers from a significantly higher computational complexity and requires intense regularization to preserve the integrity of the original signal. However, despite having these weaknesses, Pressure Matching is able to form an extremely tight and well-directed beam to the bright point, with little peripheral aliasing.

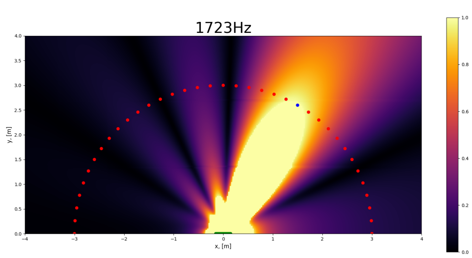

Figure 2: *Pressure Matching*

3 System

To test the performance of the algorithms presented in Section 2, I designed a beamforming system and necessary infrastructure. The system is composed of hardware and software subsystems.

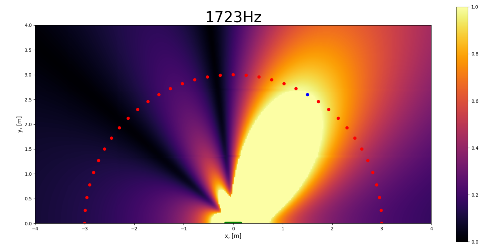

Figure 3: *Delay and Sum*

3.1 Hardware

The core of my hardware subsystem was a laptop, where I performed all of my signal processing. The hardware subsystem was able to accept data either through a .WAV file or in real time through a microphone. Then, after processing, the laptop used two USB-hubs and eight USB-audio cards to playback through a 16×1 loudspeaker array.

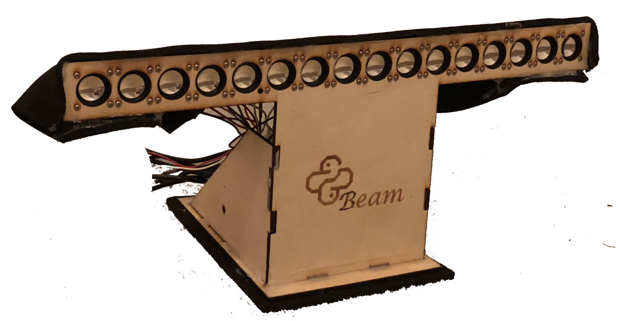

Figure 4: *The 16×1 loudspeaker beamforming array used*

3.2 Software

All of my software was written in the Python programming language and made extensive use of the SciPy and NumPy libraries for their highly optimized signal processing and linear algebra routines. I employed the PyAudio library both for reading data from a microphone and to playback through 16 loudspeakers. Finally, I developed the PyBeam library, which contained methods encapsulating the process of generating and applying beamforming filters to an incoming audio signal. The PyBeam library is not only able to process signals and drive a 16×1 array like I used, but is also able to scale upwards and downwards to any possible array configuration (Ter Martirosyan, 2018).

4 Results

In testing my system, I hoped that my data would corroborate the expectations presented in Section 2.3. If not, my beamforming system would be proven unsuccessful.

Fortunately, most of the testing parameters set forward in Section 2.3 were easily verified. Both algorithms replicated the original signal faithfully at the bright point. However, Pressure Matching took longer to compute than Delay and Sum did and required intense regularization, as expected.

This means that, ultimately, the success of my beamformer rested on whether my results corroborated two aspects of beamforming: "amplitude" beamforming and "phase" beamforming.

4.1 Amplitude Beamforming

Amplitude Beamforming refers to the amplitude of the signal at the bright point being significantly larger than the amplitude of the signal at any dark point. Amplitude Beamforming is what most people think beamforming is, and the only form of beamforming humans can detect unaided by any equipment. Accomplishing Amplitude Beamforming is the primary objective of this research project, as that is the only component of beamforming relevant to the problem of personalized sound.

As is evident from the plots in Figure 5, Amplitude Beamforming was successfully acheived. Both algorithms were effective at beamforming sound to the bright point when directed at both $90°$ and $60°$. As predicted, Pressure Matching outperformed Delay and Sum in terms of acoustic contrast between bright and dark points.

4.2 Phase Beamforming

Phase Beamforming is another component of beamforming less important to acoustic beamforming, but vitally relevant to radio-frequency beamforming. With modern modulation techniques for telecommunication, like Quadrature Amplitude Modulation and Amplitude Phase Shift Keying, data is contained not only in amplitude but in phase. Losing phase integrity essentially renders the data being transmitted nonsense. Since the goal of my research project was to not only to analyze and solve the personalized sound problem but to also gain intuition on radio-frequency beamforming, exploring Phase Beamforming made sense. With both algorithms, we want the signal at the bright point to be in-phase with the input signal.

The plots in Figure 6 indicate the expected performance and measured performance like the Amplitude Beamforming plots in Figure 5 along with a 5 centimeter confidence interval. The interval takes into account error in microphone placement $\pm$5 centimeters from the target radius of 3 meters. From these plots, it is clear that the data mostly falls in the 5 centimeter confidence interval, with the data being in-phase at the bright point, where Phase Beamforming dictates it should fall.

4.3 Next Steps

That is not to say my beamformer is perfect. There is significant room for improvement, as while signal directivity was accomplished, the dark points were not silent. Possible improvements for my beamformer include:

- Better rear noise cancellation
- Better emitter isolation
- Better noise elimination

In terms of my testing methodology, it was far from perfect—as was my family's living room, where I did all my testing. If I were to repeat this testing, I would either test outdoors or in an anechoic chamber, both places that have methods for the removal of excess acoustic energy. Without these methods, excess acoustic energy bounces and ricochets off of walls and hard surfaces, interfering with beamforming signals in unwanted ways.

Finally, I would expand my beamformer to operate in the three-field by adding another layer of dimensionality to my beamforming array (2D array vs. my current 1D array) as depicted in Figure 7. This would allow my beamformer, for example, to aim at the listener's head rather than at in their general direction.

4.4 Conclusions

With the success of Amplitude Beamforming and Phase Beamforming verified, that means my beamforming hardware and software structure is successful at beamforming an acoustic signal and has promise for solving the personalized sound problem.

Bibliography

Druyvesteyn, WF and John Garas (1997). "Personal sound". In: *Journal of the Audio Engineering Society* 45.9, pp. 685–701.

Elliott, Stephen J et al. (2012). "Robustness and regularization of personal audio systems". In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.7, pp. 2123–2133.

Fink, Mathias and Claire Prada (2001). "Acoustic time-reversal mirrors". In: *Inverse problems* 17.1, R1.

Kirkeby, Ole and Philip A Nelson (1993). "Reproduction of plane wave sound fields". In: *The Journal of the Acoustical Society of America* 94.5, pp. 2992–3000.

Olivieri, Ferdinando et al. (2016). "Theoretical and experimental comparative analysis of beamforming methods for loudspeaker arrays under given performance constraints". In: *Journal of Sound and Vibration* 373.Supplement C, pp. 302 –324. ISSN: 0022-460X. DOI: https://doi.org/10.1016/j.jsv.2016.03.005. URL: http://www.sciencedirect.com/science/article/pii/S0022460X16002340.

Ter Martirosyan, Sarkis (2018). *PyBeam*. https://github.com/smtm1209/PyBeam.

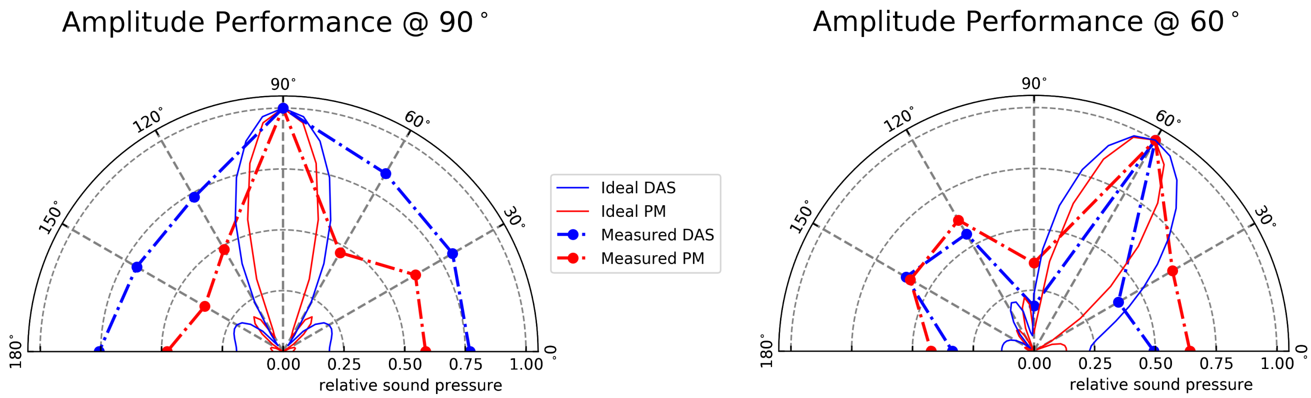

Figure 5: *Amplitudes at* $90°$ *and* $60°$

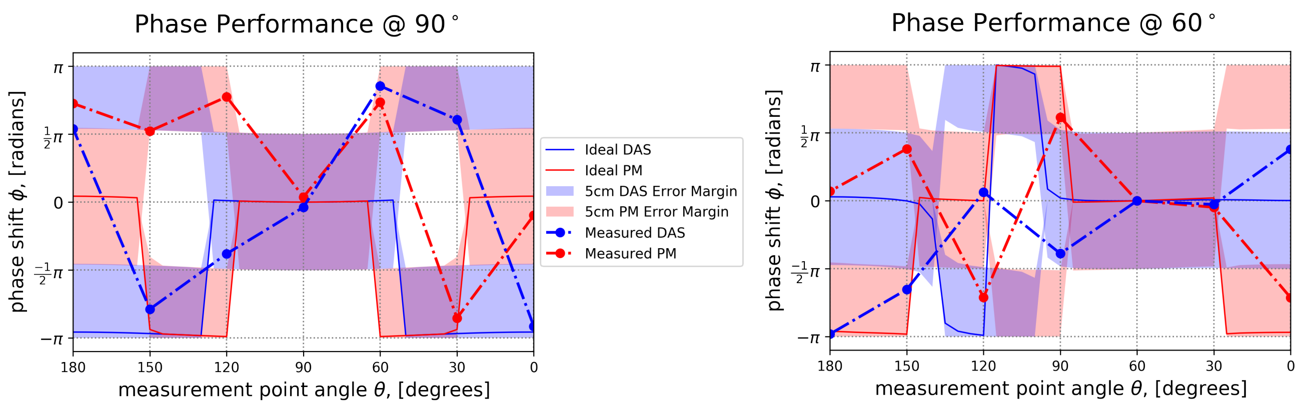

Figure 6: *Phase Shifts at* $90°$ *and* $60°$

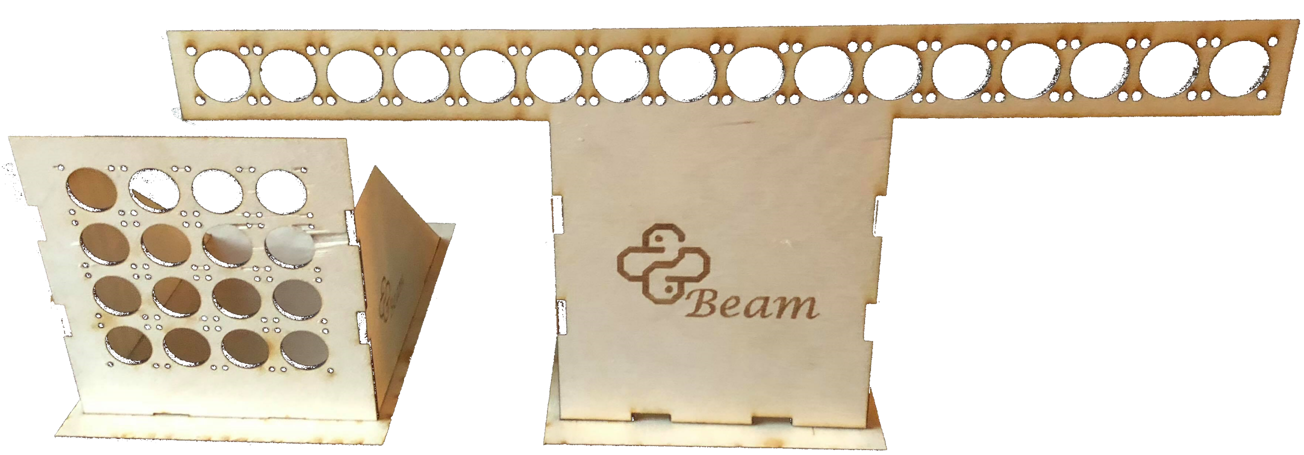

Figure 7: *Example 2D and 1D loudspeaker arrays*