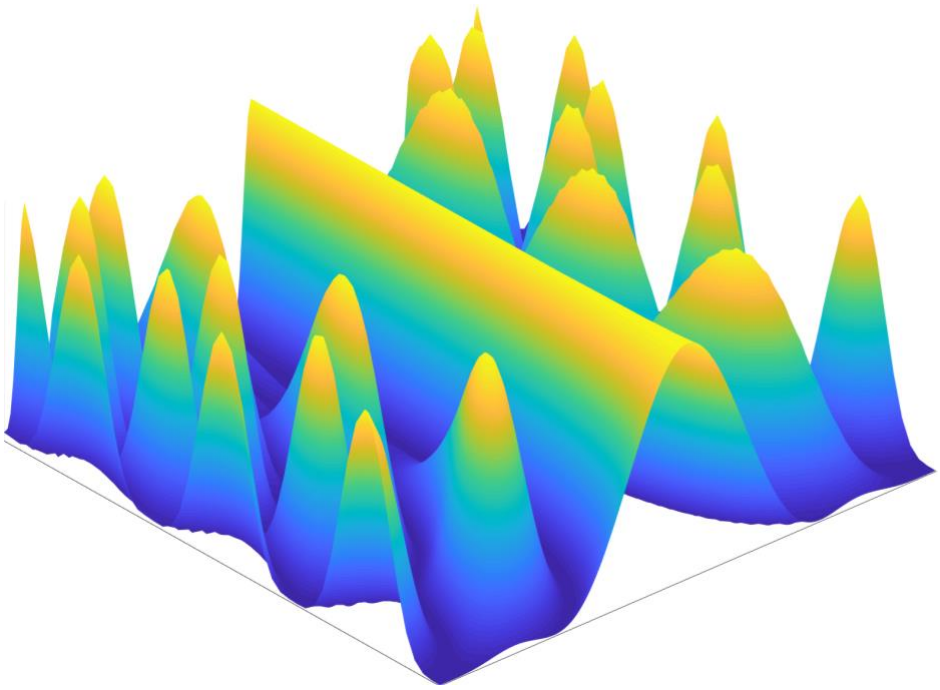
Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project Report 2019



Project Title:	**Signal Processing for Circular Microphone Array Beamforming**
Student:	**Libang Liang**
CID:	**01204497**
Course:	**EE3**
Project Supervisor:	**Prof P.A. Naylor**
Second Marker:	**Prof A. Manikas**

Table of Contents

Abstract *4*

1. Introduction *5*

1.1. Aim **5**

1.2. Objectives **5**

1.3. Report Structures **6**

2. Technical Background *7*

2.1. Acoustic Wave Propagation in a Room **7**

2.2. Near-field and Far-field **8**

2.3. Array Geometry **9**

2.4. Discrete Representation of Model **9**

2.5. Evaluation of Beamformers **10**

2.6. Continuous Array Spatial Filtering **12**

2.6.1. Linear Arrays [9] 13

2.6.2. Circular Arrays [9] 14

2.6.3. Conclusion and Array Geometry Comparison 15

2.7. Beamforming Algorithms **15**

2.7.1. Delay-and-Sum Beamformer 16

2.7.2. Superdirective Beamformer 16

2.7.3. Spherical Isotropic Noise Model 17

3. Implementations and Results *18*

3.1. Software Tools **18**

3.2. Direction of Arrival **20**

3.3. Array Geometry and Source Positions **20**

3.4. Simulative Continuous Linear and Circular Array with No Beamforming **21**

3.5. Linear Array Delay-and-Sum Beamformer **26**

3.5.1. Spatial Aliasing and Grating Lobes 27

3.5.2. Effects of spacing 29

3.5.3. Effects of Number of Elements 31

3.5.4. Steering Angle Analysis 33
3.5.5. Speech From Different Direction 35
3.6. Circular Array Delay-and-Sum Beamformer 36
3.6.1. Effects of Radius 39
3.6.2. Effects of Number of Elements 40
3.6.3. Spatial Aliasing Analysis 41
3.6.4. Steering Angle Analysis 42
3.6.5. Speech from Different Direction 44
3.6.6. Elevation Study 49
3.7. Superdirective Beamformer 54
4. Testing 56
4.1. Manipulation on frequency component 56
4.2. Extra Gain Added 56
5. Conclusion 58
5.1. Evaluation 58
5.2. Further Work 58
Bibliography 60
Appendices 62

Abstract

Microphone array beamforming is an important technique which attracts much attention nowadays due to current increasing demand in smart speaker in consumer market as well as digital meeting transcription and translation technology in business field. Alternative speech enhancement which use single microphone is a proven technique and widely used in many mobile devices. However, the use of microphone array beamforming gives one the opportunity to exploit the fact that the source of the desired signal and the noise sources are physically separated in space. And there are many common scenarios where the desired signal and unwanted noise are spatially separated. So investigating microphone array beamforming technique is worthwhile.

The aim of this project is to investigate the feasibility and robustness of processing human speech through microphone array beamforming via simulation. The project involved verifying beamforming algorithms and comparing the performances for different array geometry, microphone number and interspacing between microphones. This project also conclusively evaluated the performance of microphone array with realistic spatial arrangement specific for a room and its corresponding beamforming algorithms via simulation.

1. Introduction

Beamforming is a spatial filtering technique that widely used in sensor arrays. This technique enable the array to focus on specific direction when receiving or emitting signals by having a constructive interference at desired angle while a destructive interference otherwise [1]. This technique was widely investigated in wireless communication, radar, sonar, geophysical and medical imaging and astrophysics. In wireless communication, beamforming is commonly used in signal emitter. Take 5G technology as an example, beamforming can help by focusing a signal in a concentrated beam that points only in the direction of a user, rather than broadcasting in many directions at once. This approach can strengthen the signal's chances of arriving intact and reduce interference for everyone else [2]. On the other hand, as a receiver, microphone arrays employed beamforming technique can achieve multiple purposes such as extracting voice input from ambient noise in electronic devices, locating objects, high fidelity original recordings and environmental noise monitoring [3].

With the fast development of the smart speaker industries in recent years, the demand for microphone array and beamforming technology is in a great escalation [4]. In smart speaker usage scenarios, which is also a typical microphone arrays environment, the desired speech signal originates from a talker's mouth, and is corrupted by interfering signals such as other talkers and room reverberation [5]. The aim of beamforming in this case is to capture the most undistorted speech signal which will be later used in speech recognition. Compare to single microphone with speech enhancement technique, the use of microphone array beamforming gives one the opportunity to exploit the fact that the source of the desired signal and the noise sources are physically separated in space. And there are many common scenarios where the desired signal and unwanted noise are spatially separated. So investigating microphone array beamforming technique is worthwhile.

1.1. Aim

The aim of this project is to investigate the performance of different beamforming algorithms for circular microphone array as well as how array geometry parameters would change the behavior of the beampattern.

1.2. Objectives

The objective for this project is to answer two questions: "Is the performance of a microphone array applicable?", "Is circular microphone array or linear microphone array better off in real

life application?" To answer this question, the objective can be broken down in to number of deliverables:

Linear Array Delay-and-Sum Beamforming simulation

Linear microphone array will be set up in simulating environment and delay-and-sum beamforming algorithms will be implemented, variation of the performance of the beamforming algorithm when element spacing and element number change will be studied.

Circular Array Delay-and-Sum beamforming simulation

Circular microphone array will be set up in simulating environment and delay-and-sum beamforming algorithms will be implemented, variation of the performance of the beamforming algorithm when array radius and element number change will be studied.

Superdirective Beamforming Algorithm simulation

Both circular microphone array and linear microphone array will be set up in simulating environment and superdirective beamforming algorithms will be implemented. Comparison between delay-and-sum beamformer and superdirective beamformer will be carried out.

1.3. Report Structures

The rest of the report is structured as follows:

Technical Background: This section covers the background knowledge required for this project, such as, acoustic wave, assumptions, derivations of the beamformers, etc..

Implementations and Results: This section includes the verification of theoretical knowledge, implementation of each beamforming algorithms and performance comparison between each beamformers, and varying array geometries and parameters. Most of the objectives are achieved here.

Testing: This section includes methods and tests used during implementation.

Conclusion: This section contains analyzation of the simulation result, scientific conclusion and assumption obtained from the experiment. This section also includes evaluation whether the project objectives set in Interim Report are met. This section also gives brief conclusion on the project and discuss the potential of how could this project be taken further in the future.

2. Technical Background

2.1. Acoustic Wave Propagation in a Room

In order to understand how microphones array operates, we first knowing how sound propagates in space

Acoustic wave equation at position(x, y, z)(shown in Figure 1) at time instant t, acoustic pressure $p(x, y, z, t)$ is derived by Richard Feynman [6]:

$$\nabla^2 p(x, y, z, t) = \frac{1}{c^2}\frac{\partial^2 p(x,y,z,t)}{\partial t^2} = \frac{1}{c^2}\frac{\partial^2 p(r,t)}{\partial t^2} = \nabla^2 p(\boldsymbol{r}, t), \tag{2.1}$$

where $\boldsymbol{r} = (x, y, z)^{\boldsymbol{T}}$, c is the speed of sound. The four-dimensional Fourier Transform [7] of $p(x, y, z, t)$ gives

$$\begin{aligned} P(\boldsymbol{k}, \omega) &= \int_{-\infty}^{+\infty} (\int_{-\infty}^{+\infty} p(\boldsymbol{r}, t) e^{-i\omega t} dt) e^{\boldsymbol{k}^T \boldsymbol{r}} d\boldsymbol{r} \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} p(\boldsymbol{r}, t) e^{-(i\omega t - \boldsymbol{k}^T \boldsymbol{r})} dt\, d\boldsymbol{r}, \end{aligned} \tag{2.2}$$

where $\boldsymbol{k}$ is the wave vector that gives directional and frequency information

$$\begin{aligned} \boldsymbol{k} = -k\begin{pmatrix} k_x \\ k_y \\ k_z \end{pmatrix} == -\frac{2\pi}{\lambda}\begin{pmatrix} k_x \\ k_y \\ k_z \end{pmatrix} = -\frac{2\pi f}{c}\begin{pmatrix} k_x \\ k_y \\ k_z \end{pmatrix} = -\frac{\omega}{c}\begin{pmatrix} k_x \\ k_y \\ k_z \end{pmatrix} = \\ -\frac{\omega}{c}\begin{pmatrix} \sin(\theta)\cos(\phi) \\ \sin(\theta)\sin(\phi) \\ \cos(\theta) \end{pmatrix}. \end{aligned} \tag{2.3}$$

Where $k = \frac{2\pi}{\lambda}$ is called wavenumber, λ is the wavelength, θ and ϕ are the elevation and azimuth angles shown in Figure 1, give the directional information. To conclude, the wave vector gives information about the direction of propagation and wave length of the wave.

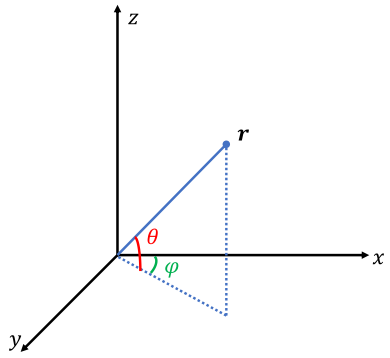


Figure 1 Position System

Consider a microphone is placed at position $\boldsymbol{r}$ in a room. It actually records infinite of these acoustic pressures coming from all direction that superpose each other. These could be wanted signal, unwanted signal, noise and reverberation of signal and unwanted signal. So the signal obtained by the microphone can be written as

$$s_0(\boldsymbol{r}_0, t) = \sum_{j=0}^{\infty} A_j e^{i(\omega_j t - \boldsymbol{k}_j{}^T \boldsymbol{r}_0)}. \tag{2.4}$$

Where A_j is the amplitude of j acoustic pressure.

If it is assumed that the room is anechoic, and only source signal is present, then the signal captured by the micro phone at position $\boldsymbol{r}_0$ at time instant t could be written as

$$s_0(\boldsymbol{r}_0, t) = A_0 e^{i(\omega_0 t - \boldsymbol{k}_0{}^T \boldsymbol{r}_0)}. \tag{2.5}$$

2.2. Near-field and Far-field

The near-field and far-field concepts are relevant to the physical distance between the signal and the receiver.

Far-field is defined as beginning at a distance of two wavelengths away from the sound source, and extends outward to infinity. In far-field, the spherical shape of the sound waves is large enough so we could approximate the wave front as a plane wave. Signal captured by a microphone array at far-field could be formulated as

$$X(\omega, \boldsymbol{r}_n, \eta) = D(\omega, \boldsymbol{r}_n, \eta) S(\omega) + N_n(\omega). \tag{2.6}$$

Where $S(\omega)$ is the source signal, $N_n(\omega)$ is the additive noise the n-th microphone collected, $D(\omega, \boldsymbol{r}_n, \eta)$ is the transfer function for sound travel from source position to n-th microphone position:

$$D(\omega, \boldsymbol{r}_n, \eta) = \frac{1}{\rho} e^{-i\omega ||\boldsymbol{r}_n|| \cos(\phi_s - \phi_n)/c}. \tag{2.7}$$

Where ρ is the average distance between the source and microphones, it could be seen that the signal captured by the microphone is attenuated by the amount of $\frac{1}{\rho}$. ϕ_s is the average angle between the source and the array, ϕ_n is the angle spacing between each microphone. So the captured signal is delayed by $||\boldsymbol{r}_n|| \cos(\phi_s - \phi_n)/c$ seconds.

On the other hand, near-field is the area close to the source. In this region the attenuations and delays between each signals captured by microphones vary heavily with the microphones' position, so average approach is not valid for near-field scenarios. Attenuations and delays is calculated by considering individual microphone position:

$$D(\omega, \boldsymbol{r}_n, \eta) = \frac{1}{||\boldsymbol{r}_s - \boldsymbol{r}_n||} e^{-i\omega ||\boldsymbol{r}_s - \boldsymbol{r}_n||/c}. \tag{2.8}$$

2.3. Array Geometry

Microphones can be placed in different patterns spatially to achieve different characteristics. Most researched geometries are Uniform Circular Array, Uniform Linear Array, Non-uniform Linear Array and Uniform Rectangular Array, shown in Figure 2. The geometry of the array can be treated as choosing different ways to sample the spatial information.

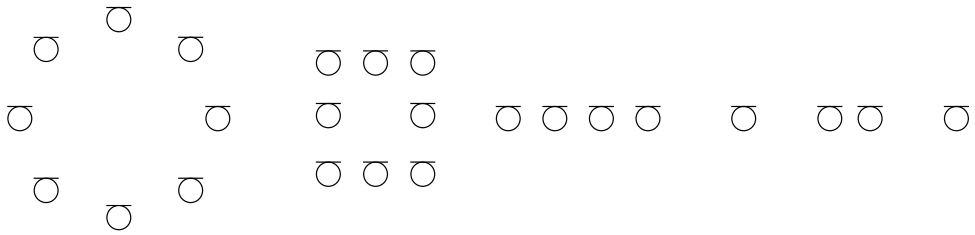


Figure 2 Different Array Geometries: from left to right: Uniform Circular Array, Uniform Rectangular Array, Uniform Linear Array, Non-uniform Linear Array

2.4. Discrete Representation of Model

In real life, signal captured by microphones are sampled and quantized, so discrete model needs to be introduced.

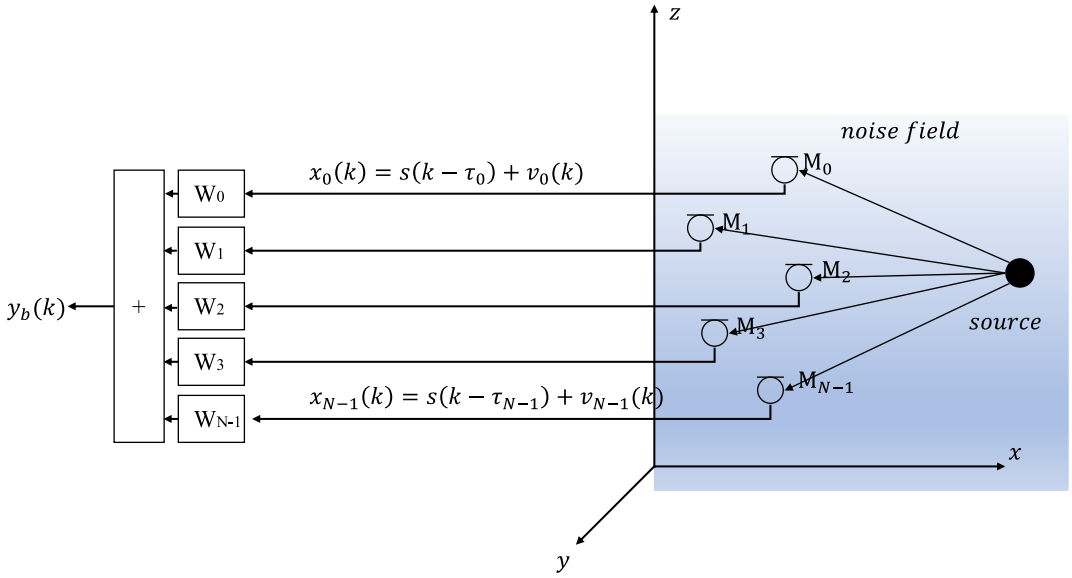


Figure 3 Signal model consisting of noise field and desired source signal

Signal model shown in Figure 3 illustrates that one sample of the discrete input sequence $x(k)$ at each sensor n is the sum of the delayed and attenuated version of the source signal $a_i s(k-\tau)$ and noise component $v_i(k)$ with arbitrary spatial statistics.

$$\begin{pmatrix} x_0(k) \\ x_1(k) \\ \vdots \\ x_{N-1}(k) \end{pmatrix} = \begin{pmatrix} a_0 s(k-\tau_0) \\ a_1 s(k-\tau_1) \\ \vdots \\ a_{N-1} s(k-\tau_{N-1}) \end{pmatrix} + \begin{pmatrix} v_0(k) \\ v_1(k) \\ \vdots \\ v_{N-1}(k) \end{pmatrix}. \tag{2.9}$$

Writing in matrix form gives:

$$\boldsymbol{x}(k) = \boldsymbol{\alpha s}(k - \boldsymbol{\tau}) + \boldsymbol{v}(k). \tag{2.10}$$

Transforming this in to discrete frequency domain using Fourier Transform leads to

$$\boldsymbol{X}(e^{j\Omega}) = \boldsymbol{S}(e^{j\Omega})\boldsymbol{d} + \boldsymbol{V}(e^{j\Omega}). \tag{2.11}$$

Where $\boldsymbol{d}$ represents the attenuation and delays in frequency domain, which is $\boldsymbol{\alpha}$ and $\boldsymbol{\tau}$ in time domain. This parameter depends on spatial location of the sensors as well as the direction of the desired signal. So $\boldsymbol{d}$ could be expressed as

$$\boldsymbol{d^T} = \left[a_0 e^{-j\Omega\tau_0}, a_1 e^{-j\Omega\tau_1}, a_2 e^{-j\Omega\tau_2} \dots a_N e^{-j\Omega\tau_N}\right]. \tag{2.12}$$

According to the model in Figure 3, the output signal could be expressed as

$$Y_b(e^{j\Omega}) = \sum_0^{N-1} W_n{}^*(e^{j\Omega}) X_n(e^{j\Omega}) = \boldsymbol{W^H X}, \tag{2.13}$$

where W_n is the frequency-domain coefficients of the beamformer of the sensor n at the frequency Ω. The inverse Fourier transform gives the time domain output $y_b(k)$.

2.5. Evaluation of Beamformers

To get a better understanding of the features of the different designs of optimal beamformers, different measures are used to analyze beamformers' performances.

Array-Gain

Array-Gain(AG) evaluates the enhancement of the signal-to-noise ratio(SNR) of the output(whole array) when one sensor's SNR changes. It is defined as

$$G = \frac{SNR_{Array}}{SNR_{Sensor}}. \tag{2.14}$$

The SNR of one sensor is the ratio of the power spectral densities(PSD) of the signal Φ_{SS} and the average noise $\Phi_{V_aV_a}$:

$$SNR_{Sensor} = \frac{\Phi_{SS}}{\Phi_{V_aV_a}}. \tag{2.15}$$

Since the input-output relationship is given by $\boldsymbol{Y} = \boldsymbol{W^H X}$, the PSD relationship is then:

$$\Phi_{YY} = \boldsymbol{W^H \Phi_{XX} W}, \tag{2.16}$$

when there is only desired signal is present, the output is:

$$\Phi_{YY\,Signal} = \Phi_{SS}|\boldsymbol{Wd}|^2. \tag{2.17}$$

Assuming only noise is present, the output is:

$$\Phi_{YY\,Noise} = \Phi_{V_aV_a}\boldsymbol{W^H \Phi_{VV} W}, \tag{2.18}$$

where $\boldsymbol{\Phi_{VV}}$ is the normalized cross power spectral density matrix of the noise, where the normalization factor is set so the trace of the matrix is equal to N, so

$$SNR_{Array} = \frac{\Phi_{SS}|\boldsymbol{Wd}|^2}{\Phi_{V_aV_a}\boldsymbol{W^H \Phi_{VV} W}}. \tag{2.19}$$

Rearrange (2.14) gives:

$$G = \frac{|\boldsymbol{W}^H \boldsymbol{d}|^2}{\boldsymbol{W}^H \boldsymbol{\Phi}_{\boldsymbol{VV}} \boldsymbol{W}}. \tag{2.20}$$

Assuming that the noise filed is homogeneous, $\boldsymbol{\Phi}_{\boldsymbol{VV}}$ can be replaced by the coherence matrix:

$$\boldsymbol{\Gamma}_{\boldsymbol{VV}} = \begin{pmatrix} 1 & \Gamma_{V_0V_1} & \Gamma_{V_0V_2} & \cdots & \Gamma_{V_0V_{N-1}} \\ \Gamma_{V_1V_0} & 1 & \Gamma_{V_1V_2} & \cdots & \Gamma_{V_1V_{N-1}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Gamma_{V_{N-1}V_0} & \Gamma_{V_{N-1}V_1} & \Gamma_{V_{N-1}V_2} & \cdots & 1 \end{pmatrix}. \tag{2.21}$$

$\Gamma_{V_nV_m}(e^{j\Omega})$ is the coherence function of noise in different sensors:

$$\Gamma_{V_nV_m}(e^{j\Omega}) = \frac{\Phi_{V_nV_m}(e^{j\Omega})}{\sqrt{\Phi_{V_nV_n}(e^{j\Omega})\Phi_{V_mV_m}(e^{j\Omega})}}. \tag{2.22}$$

Thus,

$$G = \frac{|\boldsymbol{W}^H \boldsymbol{d}|^2}{\boldsymbol{W}^H \boldsymbol{\Gamma}_{\boldsymbol{VV}} \boldsymbol{W}}. \tag{2.23}$$

Transforming the PSD matrix of the noise to coherence matrix allows a better comparison between different noise field.

Directivity Index

Directivity Index(DI) describes the ability of the array to suppress a diffuse noise field and enhance the signal from the desired direction. It is defined as

$$DI(e^{j\Omega}) = 10log_{10}\left(\frac{|H(e^{j\Omega},\varphi_0,\theta_0)|^2}{\frac{1}{4\pi}\int_0^{\pi}\int_0^{2\pi}|H(e^{j\Omega},\varphi,\theta)|^2 \sin(\theta)\, d\varphi d\theta}\right). \tag{2.24}$$

This definition can be interpreted as the ratio of the transfer function at desired direction(look-direction) $[\varphi_0, \theta_0]$ of the array with respect to the average transfer function over all directions(using spatial integration).

Front-to-Back Ratio

For the applications, the look-direction is always the same(conference or orchestras). For these cases, calculating front-to-back ratio(FBR) is better than DI. This is because the wanted signal is always at the same direction, all other directions could be considered unwanted noise.

The formal definition of FBR uses again the transfer function

$$FBR(e^{j\Omega}) = \frac{\int_{\theta_0-\frac{\pi}{2}}^{\theta_0+\frac{\pi}{2}}\int_{\varphi_0-\frac{\pi}{2}}^{\varphi_0+\frac{\pi}{2}}|H(e^{j\Omega},\varphi_0,\theta_0)|^2 \sin(\theta)\, d\varphi d\theta}{\int_{\theta_0+\frac{\pi}{2}}^{\theta_0+\frac{3\pi}{2}}\int_{\varphi_0+\frac{\pi}{2}}^{\varphi_0+\frac{3\pi}{2}}|H(e^{j\Omega},\varphi,\theta)|^2 \sin(\theta)\, d\varphi d\theta}. \tag{2.25}$$

White Noise Gain

White noise gain measures the ability of the array to suppress spatially uncorrelated noise.

For spatially uncorrelated noise, the coherence function $\boldsymbol{\Gamma}_{\boldsymbol{VV}\,|\boldsymbol{uncorr}} = \boldsymbol{I}$, insert in the AG, we

get:

$$WNG(e^{j\Omega}) = \frac{|\boldsymbol{W}^H \boldsymbol{d}|^2}{\boldsymbol{W}^H \boldsymbol{W}}. \tag{2.26}$$

On a logarithmic scale, positive values states an attenuation of uncorrelated noise, negative shows an amplification.

2.6. Continuous Array Spatial Filtering

This section introduces some basic ideas for how microphone array achieves directivity characteristics and some simple derivation on outputs behaviors where outputs are obtained from linear or circular array when the signal are coming from different direction. Although in simulation and real life it is not possible to have a continuous microphone array, the following derivation gives basic understanding on microphone arrays and their corresponding algorithm. Consider a sound wave originates from a far field source perpendicular to a continuous linear array, when it reaches the receivers, it could be considered as a plane wave, i.e. a wave whose value, at any moment, is constant over any plan that is perpendicular to a fixed direction in space [8]. As shown in Figure 4, when the plane wave is perpendicular to a linear array, the array output is maximized. Therefore far field directivity can be achieved using a continuous linear array.

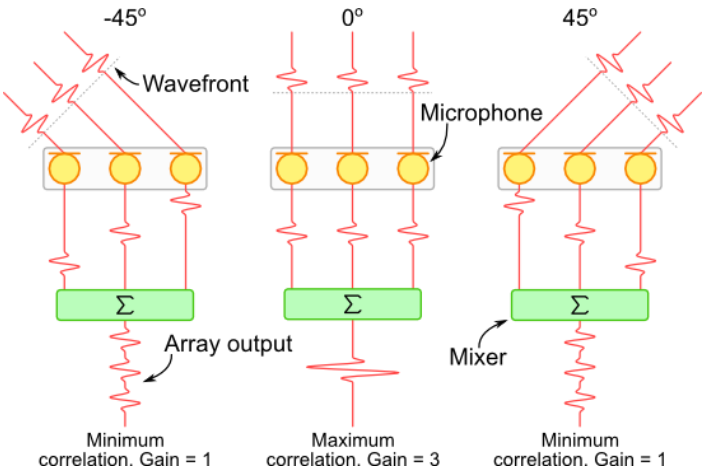


Figure 4 Plane waves incident on linear microphone arrays

However, if the continuous array has a circular geometry the signals will no longer be in-phase when summed, therefore the continuous circular array will not show the directivity obtained when using a linear array. Mathematical proof is given below.

2.6.1. Linear Arrays [9]

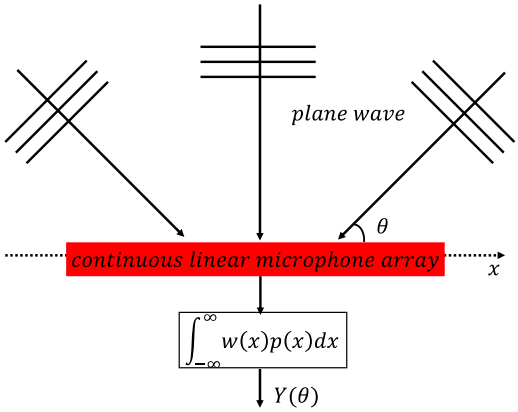


Figure 5 Linear Continuous Array Model

Shown in Figure 5, a plane wave with an angle θ (elevation is set to be 90 degree) and wavelength λ, passes through a continuous linear array, with an assumed infinite length, position on the x-axis. The output of the system is given by

$$\int_{-\infty}^{\infty} w(x)p(x)dx, \tag{2.27}$$

where $w(x)$ is the weighting function that represent the contribution of each element of the array to the output of the system and $p(x)$ is the acoustic pressure:

$$p(x) = e^{-ikx\sin\theta} = e^{-ik_x x}. \tag{2.28}$$

k is the wavenumber of the plane wave and it is defined in (2.3), $|k| = \frac{2\pi}{\lambda}$, λ is the wavelength of the acoustic plane wave. The wavenumber component in the x-direction is given by $k_x = k\sin\theta$.

Assuming in reality the linear array has finite length L. Then the weighting function could be write as a rectangular aperture function

$$w(x) = \begin{cases} 1/L, -L/2 < x < L/2 \\ 0, otherwise \end{cases}. \tag{2.29}$$

This is the simplest form of weighting functions where output can be interpreted as taking the average of each element in the array. Substituting the (2.28) and (2.29) into (2.27) gives

$$Y(k_x) = \frac{1}{L} = \int_{-L/2}^{L/2} e^{-ik_x x}\,dx = \frac{1}{ik_x L}(e^{ik_x L/2} - e^{-ik_x L/2}), \tag{2.30}$$

which gives an relationship between output and wavenumber k_x. Using the fact that $e^{ia} - e^{-ia} = 2i\sin a$, the output becomes

$$Y(k_x) = \frac{\sin[(kL/2)\sin\theta]}{(kL/2)\sin\theta}. \tag{2.31}$$

This is a "sinc" function, its zero crossings closest to the origin are given by

$$\left(\frac{kL}{2}\right)\sin\theta = \pm\pi. \tag{2.32}$$

When the plane wave passes the linear microphone array at angle θ. Using the definition of

wavenumber $k = \frac{2\pi}{\lambda}$,

$$\theta = \pm \sin^{-1}(\lambda/L) \approx \lambda/L \quad (\lambda \ll L) \tag{2.33}$$

This equation indicates that when the plane wave come at angle $\pm \sin^{-1}(\lambda/L)$, the output $Y(k_x)$ is zero, which means the directivity of the main lobe of the linear array is situated within $-\lambda/L$ and λ/L. Thus it can be conclude that the width of the main lobe is inversely proportional to array length. Longer the array, better the directivity(when frequency of the plane wave keep constant).

2.6.2. Circular Arrays [9]

A similar arrangement is setup except for the linear array is replaced by a circular array. The array again receives a plane wave from angle θ with wavelength λ.

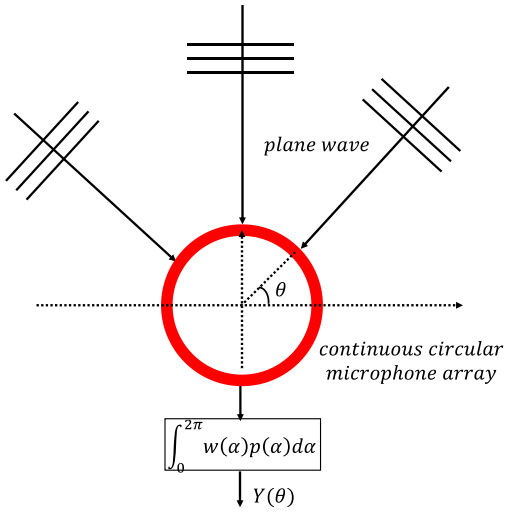


Figure 6 Circular Continuous Array Model

The output of the array is given by integrating the product of the weighting function and received signal at all element from 0 to 2π:

$$\int_0^{2\pi} w(\alpha)p(\alpha)d\alpha. \tag{2.34}$$

The complex pressure of a plane wave travelling in two-dimensions can be written as [10]

$$p_p(\boldsymbol{x}) = p_p(\boldsymbol{y})e^{-i\boldsymbol{k}(\boldsymbol{x}-\boldsymbol{y})}, \tag{2.35}$$

where $p_p(\boldsymbol{y})$ is also a complex pressure at position $\boldsymbol{y}$ with wavenumber vector $\boldsymbol{k}$ with modulus k.

Now assuming $p_p(\boldsymbol{y}) = A$ and $\boldsymbol{y}$ is at origin $(0,0)$, $\boldsymbol{x}$ is on circular array which radius is r. Then the coordinate of $\boldsymbol{x}$ is $(r\cos a, r\sin a)$, when a plane wave incidents on the circular array with angle θ shown in Figure 6, then,

$$p_p(\boldsymbol{x}) = Ae^{-ikr(\cos a \sin\theta + \sin a \cos\theta)}. \tag{2.36}$$

Using the trigonometric property, $\cos(a-\theta) = (\cos a \sin\theta + \sin a \cos\theta)$, (2.36) becomes

$$p_p(\boldsymbol{x}) = Ae^{-ikr\cos(a-\theta)}. \tag{2.37}$$

Uniform weighting function for a circular array can be written as

$$w = \frac{1}{2\pi r} \quad 0 < a < 2\pi. \tag{2.38}$$

Substitute above derivation into the output equation gives

$$Y(a) = \frac{A}{2\pi r}\int_0^{2\pi} e^{[-ikr\cos(a-\theta)]}\, da. \tag{2.39}$$

From the mathematical property

$$J_n(z) = \frac{1}{2\pi i^n}\int_0^{2\pi} e^{iz\cos a} e^{ina}\, da. \tag{2.40}$$

Assuming $n = 0$ and θ is constant, the output becomes

$$Y(kr) = \frac{AJ_0(-kr)}{r}, \tag{2.41}$$

where $J_0(-kr)$ is the Bessel function of the first kind of order zero [11], see Figure 7.

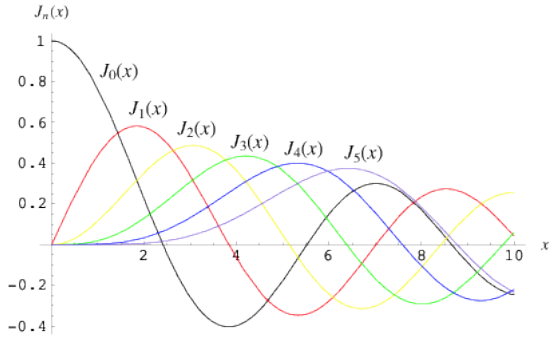


Figure 7 Bessel Function

This is an important result showing that the output does not depends on the angle of the coming plane wave, in other words, if a continuous circular array has uniform filter weighting the array is omni-directional and has no directivity.

2.6.3. Conclusion and Array Geometry Comparison

The derivation and simulation in [12], [13] demonstrates that the directivity response of continuous circular microphone array with no beamforming technique implemented(simply summing elements' output) has homogeneous directivity. Linear array with the same setting exhibits two main lobes which width varies with wavelength microphone-spacing ratio.

2.7. Beamforming Algorithms

Beamforming is one of the simplest and most robust means of spatial filtering [14], this technique is widely used in discriminating between signals based on physical locations of the signal sources. Simple beamformers like Delay-and-Sum Beamformer, Superdirective

Beamformers will be introduced in this section.

2.7.1. Delay-and-Sum Beamformer

The Delay-and-Sum Beamformer(DSB) delays the output of each elements to co-phase the sound arriving from a given. This technique has proved very effective for hands-free sound capture in moderately reverberant auditoria rooms [15]. And the weighting function of the DSB is

$$\boldsymbol{W} = \frac{1}{N}\boldsymbol{d}, \tag{2.42}$$

where $\boldsymbol{d}$ is the direction vector in (2.12).

2.7.2. Superdirective Beamformer

Superdirective Beamformer(SDB) aims to minimize the power of the output signal of the array, given that the gain at desired direction is 1, i.e.

$$\boldsymbol{W}^H\boldsymbol{d} = 1. \tag{2.43}$$

The signal from all direction except for the look direction is considered as noise, so this method could be interpreted as noise minimization process. Then the problem can be formalized as:

$$\min_{\boldsymbol{W}} \boldsymbol{W}^H\boldsymbol{\Phi}_{\boldsymbol{XX}}\boldsymbol{W} \quad subject\ to\ \boldsymbol{W}^H\boldsymbol{d} = 1. \tag{2.44}$$

Using the Lagrange-multiplier [16] or gradient computation, the solution could be obtained:

$$\boldsymbol{W} = \frac{{\boldsymbol{\Phi}_{\boldsymbol{VV}}}^{-1}\boldsymbol{d}}{\boldsymbol{d}^H{\boldsymbol{\Phi}_{\boldsymbol{VV}}}^{-1}\boldsymbol{d}}, \tag{2.45}$$

where $\boldsymbol{d}$ is the direction vector in (2.12), $\boldsymbol{\Phi}_{\boldsymbol{VV}}$ is the normalized cross power spectral density matrix of the noise.

Assuming a homogeneous noise field the solution is a function of the coherence matrix:

$$\boldsymbol{W} = \frac{{\boldsymbol{\Gamma}_{\boldsymbol{VV}}}^{-1}\boldsymbol{d}}{\boldsymbol{d}^H{\boldsymbol{\Gamma}_{\boldsymbol{VV}}}^{-1}\boldsymbol{d}}. \tag{2.46}$$

Where $\boldsymbol{\Gamma}_{\boldsymbol{VV}}$ is the coherence matrix depends on different type of noise, solution for spherical isotropic noise is introduced later in 2.7.3. This is the well-known solution for superdirective beamformers, also known as Minimum Variance Distortionless Response(MVDR) solution.

Figure 8 shows performance comparison between DSB and SDB, it could be noticed that the DSB shows no directivity at low frequency range, while the SDB demonstrates relatively constant directivity throughout whole frequency range of interest.

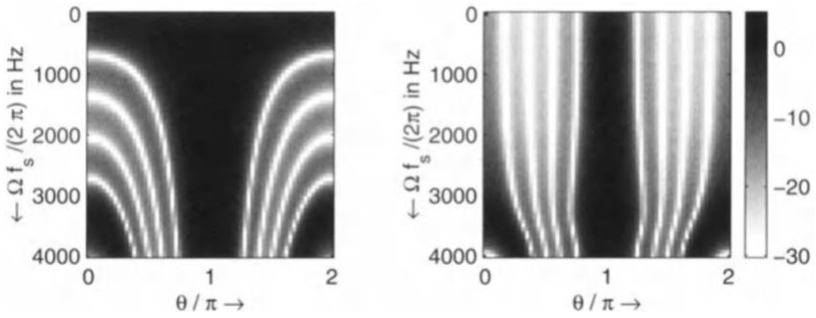


Figure 8 DSB vs. SDB

It can be noticed the MVDR-solution, when the spatial uncorrelated noise is applied, which coherence matrix is $\boldsymbol{\Gamma_{VV}} = \boldsymbol{I}$, is

$$\boldsymbol{W} = \frac{1}{N}\boldsymbol{d}. \tag{2.47}$$

In reality, microphones are not perfect and so they generate noise themselves and add to the signals recorded. Gilbert and Morgan have proposed a method for solving this self-noise amplification problem [17] under a WNG constraint [18], which is adding a scalar μ to the diagonal of the coherence matrix:

$$\boldsymbol{W} = \frac{(\boldsymbol{\Gamma_{VV}}+\mu\boldsymbol{I})^{-1}\boldsymbol{d}}{\boldsymbol{d}^{\boldsymbol{H}}(\boldsymbol{\Gamma_{VV}}+\mu\boldsymbol{I})^{-1}\boldsymbol{d}}. \tag{2.48}$$

Typical value of μ are set to -10dB to -30dB [14]. In order to preserve the values of the coherence matrix smaller than one, adding a scalar μ to diagonal values is equivalent to divide non-diagonal values by $1+\mu$. μ can be interpreted as the ratio of the self-noise, also known as sensor noise power σ^2 to the ambient noise power Φ_{VV}.

2.7.3. Spherical Isotropic Noise Model

Different noise field will result in different coherence matrix in (2.46). For the data independent beamformer investigated in this project, it is required to use different noise field models in different situations for beamformer such as superdirective beamformer. Coherence matrix can be obtained from the normalized cross power spectral density matrix.

While the diagonal elements have values of one, for spherical isotropic noise, the non-diagonal elements are given by [14]

$$\Gamma_{V_nV_m} = \frac{sinc\left\{\frac{\Omega f_s l_{nm}}{c}\right\}}{1+\frac{\sigma^2}{\Phi_{VV}}}, n \neq m. \tag{2.49}$$

Where Ω is the frequency, f_s is the sampling frequency, l_{nm} is the distance between receivers.

3. Implementations and Results

This section concentrates on the implementation of simulation of microphone arrays and corresponding beamforming algorithms in MATLAB. Before implementation basic set up includes package used named RIR Generator, array geometry and source positions are introduced and explained. Firstly continuous array with no beamforming is simulated to illustrate microphone array behaviors. Then simple Delay-and-Sum Beamformer are implemented. During simulation, parameters like microphone element inter-spacing, element number and radius for circular array geometry are varied. How these parameters modify beamformer's behavior are investigated. Finally, Superdirective Beamformer is implemented and analyzed. To conclude, different array geometry and different beamformers are studied, their performance with changing array parameters are also explored.

3.1. Software Tools

The model simulation in this project is carried out by MATLAB Simulink together with a MEX-file(MATLAB executable file) named Room Impulse Response Generator(RIR Generator) by Audio Labs [19]. This script is able to generate multichannel Room Impulse Responses using the image method. This function enables the user to control the reflection order, room dimension and microphone directivity. By simply indicating the source and receiver positions in the room, transfer function that describes the input-output relationship could be obtained. The reason for using a MEX-file instead of standard MATLAB function is due to computation time, using MEX-functions is around 100 times faster than the standard MATLAB code [20]. In addition this file also considered fractional sampling issue so there is no need to concern about fractional sampling error in the remaining project.

RIR Generator Experiment

Consider signal model in Figure 3 in a room, sound pressure is generated by a source and propagates in all direction. When the sound pressure passes through the microphone, its amplitude level is recorded. The transfer function, which is defined as the ratio of signal recorded by the microphone and the original source signal, is affected by many parameters such as distance between the receiver and the source, room reverberation time and room geometry, etc.. Room Impulse Response Generator(RIR Generator) [19] developed by Emanuel A.P. Habets can generate the impulse response by indicating these parameters. In all simulations below, source position is an information that already pre-known.

Investigation on Reverberation Time

To get a better understanding of RIR Generator, one parameter named reverberation time needs to be well explained. Reverberation time is defined as the time required for the sound to "fade away", or realistically, to be lower than certain level such as 60dB. Sound in a room will repeatedly bounce off surface such as the floor, walls, ceiling, windows or tables. When the reflections superpose together, reverberation is created. Several reverberation time is tested and corresponding impulse response are generated. Figure 9 illustrates that the relationship between impulse responses and reverberation time(in second) using the RIR Generator. With a larger reverberation time, the impulse response, or transfer function becomes 'noisier' after the main impulse happens.

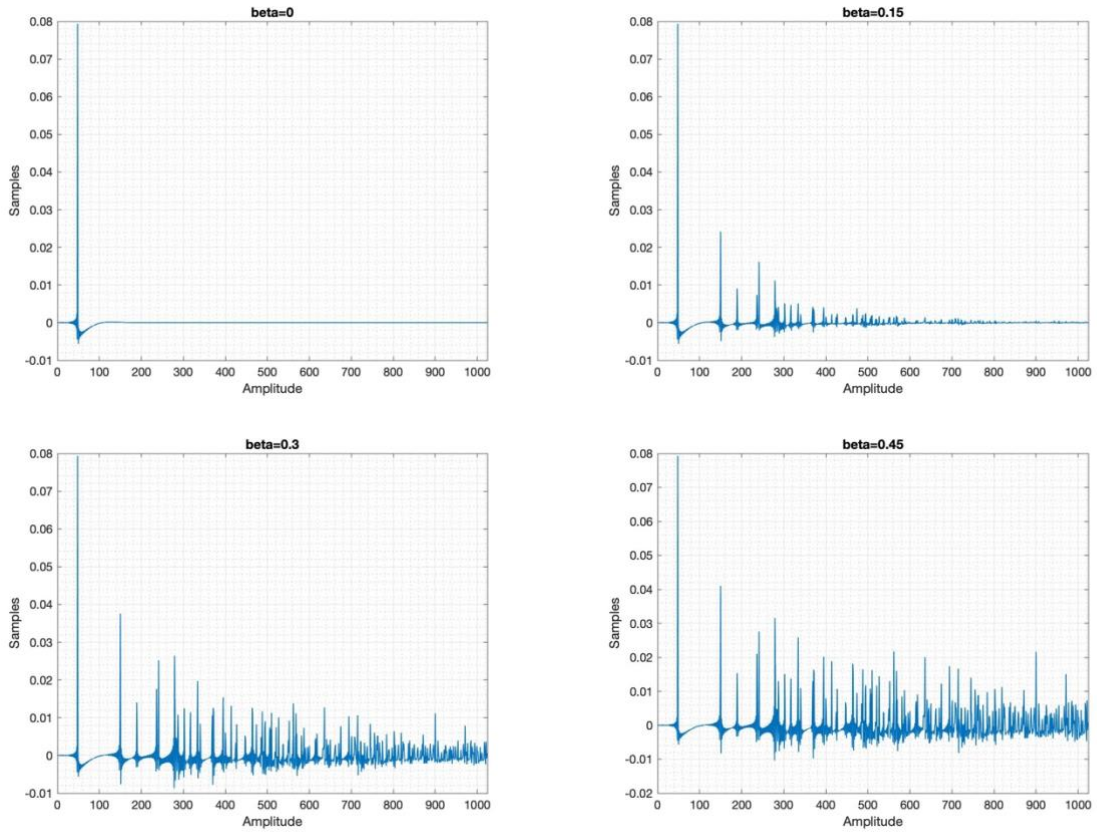


Figure 9 Impulse Responses vary with Reverberation Time(beta), with room dimension(5,5,3), source position(2.5,3.5,1.5), receiver position(2.5,2.5,1.5), number of samples = 1024.

When source is a sinusoid of frequency 500Hz, the signals recorded by the microphone is shown below in Figure 10, this is obtained by convoluting the impulse response and the source signal. It can be shown that signal with no reverberation time has the smallest amplitude and amplitude fluctuation. This can be interpreted as there is no superposition between direct signal and reflected signals when reverberation time is zero second, since the reflected signal

immediately fade away. As the reverberation time becomes larger, the reflected signal becomes larger and so does the superposed signal recorded. And also the pattern(amplitude fluctuation) gets more complex as the number of reflections increases as reverberation time increases.

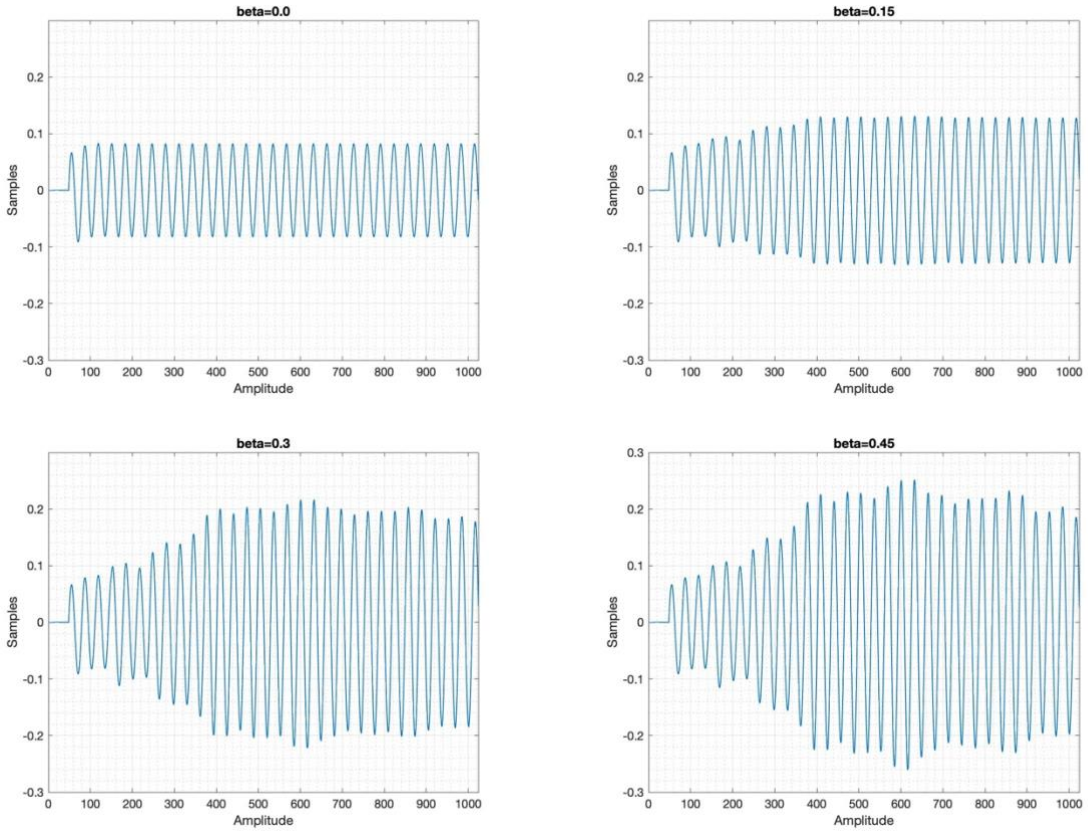


Figure 10 Signal Recorded when Source is 500Hz Sine Wave

3.2. Direction of Arrival

Given the source position and microphone positions, direction of arrival in (2.12) can be calculated. Far-field and Near-field model can be assumed using derivation in 2.2. To reduce computation complexity, Far-field model is assumed in all simulations below. For different array geometry, different direction of arrival will can be calculated, each will be explained in corresponding sections.

3.3. Array Geometry and Source Positions

As shown in (2.1), microphone position and source position are needed in order to calculate the transfer function that describe the propagation between source and receiver(microphone). Function `createMicrophonePositionArray` and `calculateSourcePositionArray` are used to generate microphone array and source positions.

The result are shown in Figure 11, for Uniform Linear Array(ULA), the microphone position generating function can control the number of microphone and the spacing between. Given the

room dimension as an input to the function, the array is placed at the center of the room. Similarly, the Uniform Circular Array(UCA) microphone generating function can adjust the number of microphone and the radius of the circular array.

The source generating function aims to generate source array, which would be used to emit signal one at a time. This is used to calculate variation of the gain if the source is at different angle with respect to the array. The source generating function is able to change the radius and the resolution(how many source for one circle). Again given the room dimension, it is placed at the center of the room.

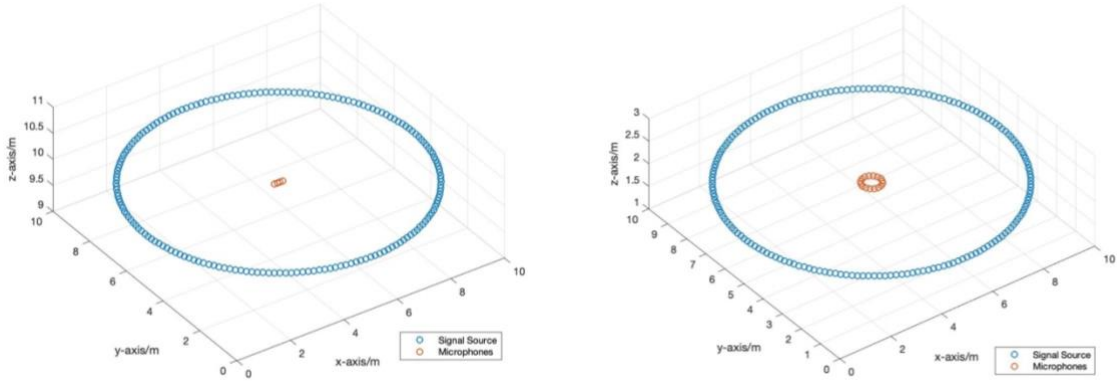


Figure 11 Microphone Array and Source Geometry(left: uniform linear array, right: uniform circular array)

3.4. Simulative Continuous Linear and Circular Array with No Beamforming

This simulation aims to prove that the continuous linear and circular array beampatterns described in 2.6 is correct. Since in simulation it is not possible to implement continuous arrays, so in order to imitate a continuous array as realistic as possible, the number of microphone is chosen to be relatively large, for example, 50. For linear array, the microphone inter-element spacing is set to be $1cm$, so the total array length is $0.01 \times (50 - 1) = 0.49m$. For circular array, the radius is set to be $0.1m$, the unwrapped array length, which is the perimeter of the circle, can then be calculated: $C = 2\pi r = 2 \times \pi \times 0.1 \approx 0.628m$.

The array geometry as well as source position are shown in Figure 12, the room dimension is set to be $(10m, 10m, 4m)$ in x, y, z axis. For simplicity, the reverberation time $beta$ is set to equal $0s$. Both circular and linear microphone array are placed at the center of the room and source position is $5m$ away from the array center.

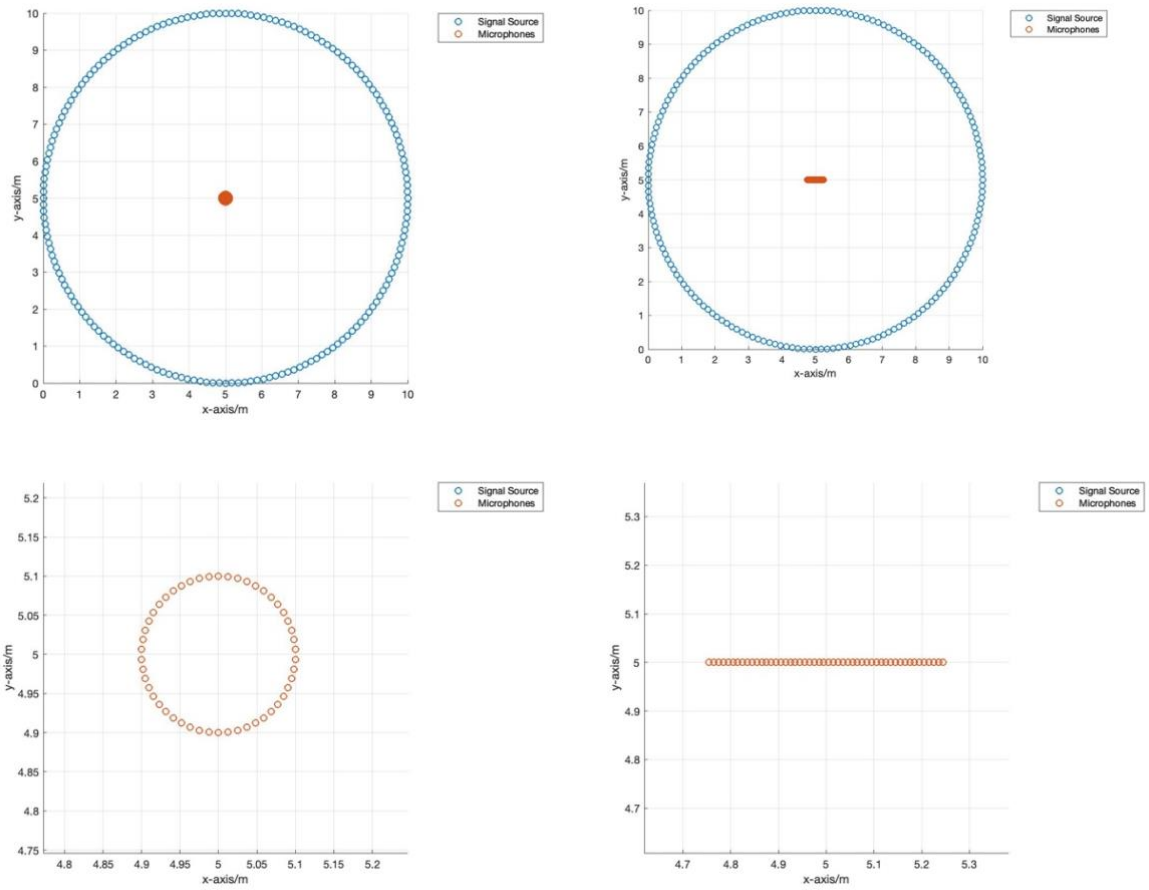


Figure 12 Circular and Linear Array Geometry(Upper: Microphone Array and Source Position, Down: Microphone Array Zoom in)

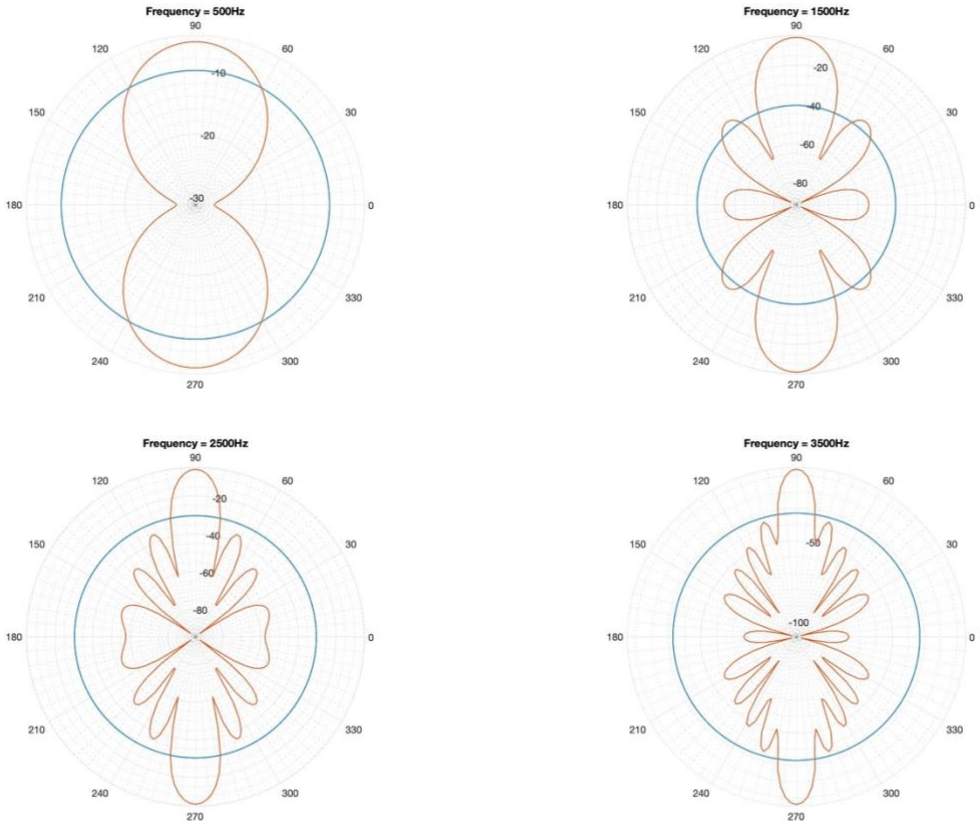
After setting up the source and receiver position, impulse responses are calculated based on each source position and each microphone position, then outputs for each microphone are calculated using convolution, outputs are summed to gives the array output. The algorithm are described below:

Pseudocode

```
source signal = sin(2*pi*frequency*t)
response list = []
for source position = 1 to end:
        output = 0;
        for microphone position =1 to end:
                transfer function = RIRGenerator( source position, microphone position, beta)
                output = output + conv(transfer function, source signal)
        end
        response list.append(output)
end
plot(response list)
```

The sampling frequency used is 16kHz. Each source position is spaced at 2° interval, so total 180 sources. Six frequencies(500Hz, 1500Hz, 2500Hz, 3500Hz, 4500Hz, 6500Hz) are tested and the corresponding beampatterns for both linear and circular array are shown in Figure 13.

It is clearly seen that the continuous circular microphone array has homogeneous directivity along all angles, since a uniform response is obtained for all six frequencies. For continuous linear array, two main lobes can be observed which are in opposite direction. As frequency increases, the number of side lobes increases and at the same time the widths of the main lobes and side lobes decrease. It could be also seen that as frequency changes, the gain of circular array also changes. However, Figure 13 does not show the exact pattern. More investigation is needed.



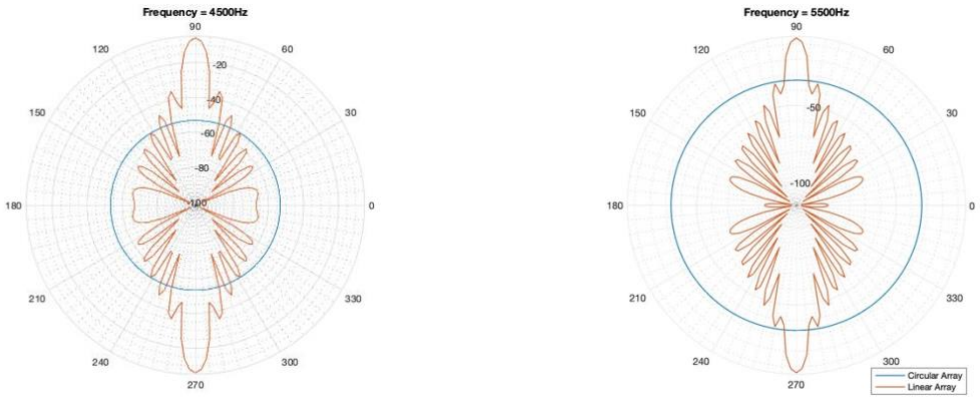


Figure 13 Beampatterns of ULA and UCA for 6 different frequencies

Investigation on gain-frequency relationship

Figure 14 and Figure 15 depict the beampatterns of continuous linear and circular array without beamforming, signals in each microphone are summed instead. $x\ axis$ represents change of azimuth angle of the source, $y\ axis$ represents frequency changes. For this project of interest, which is human speech, frequency tested varies from 400Hz to 3000Hz. The color bar on the right indicates how strong the signal is. Stronger signal tends to become yellow, while weaker becomes dark blue.

For continuous linear array beampattern shown in Figure 15, it could be observed that the beam width become narrower as frequency increases.

It can be clearly seen that for a given frequency, a continuous circular array has uniform directivity, which is also demonstrated in Figure 13. But along the frequency UCA has a much higher gain at lower frequency. It can be also noticed that there is a trough at frequency between 1000Hz and 1500Hz, as shown in Figure 14. To further investigate this problem, circular arrays with different radius are simulated, the results are shown in Figure 16. It can be concluded that larger radius gives lower frequency where first side lobe occurs, while larger radius also results in smaller range of omni-directivity region at low frequency range.

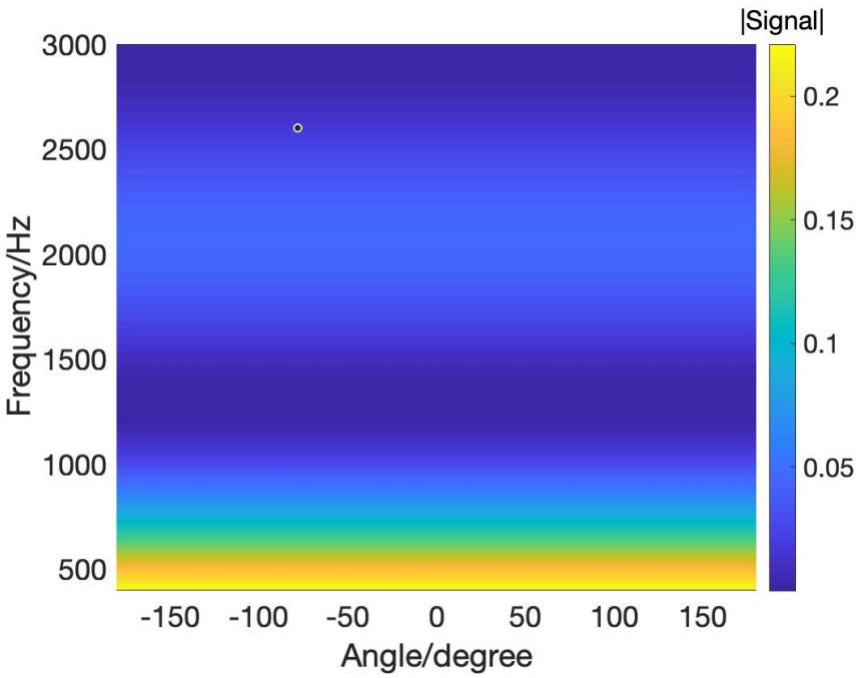


Figure 14 Continuous Circular Array Beampattern

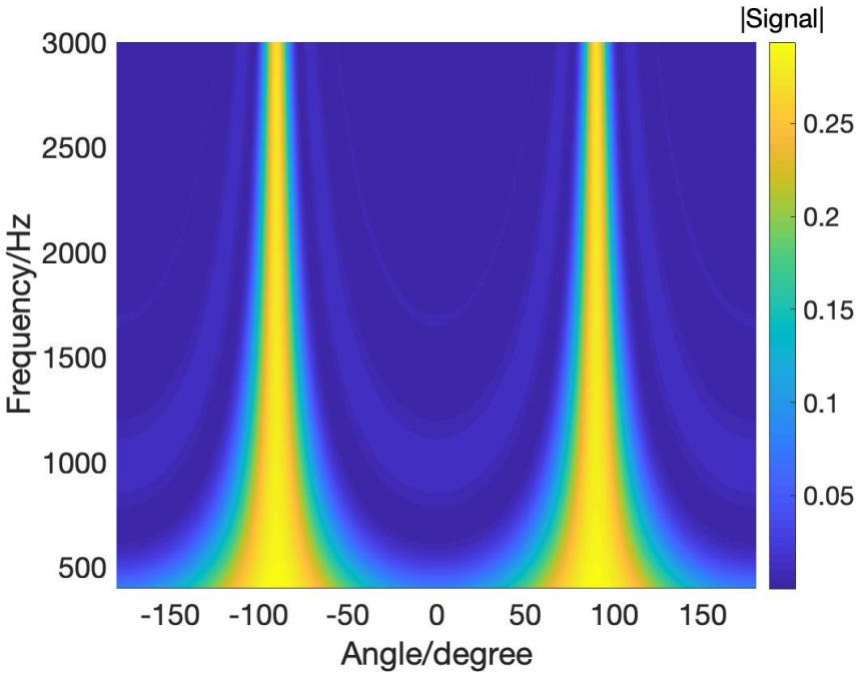


Figure 15 Continuous Linear Array Beampattern

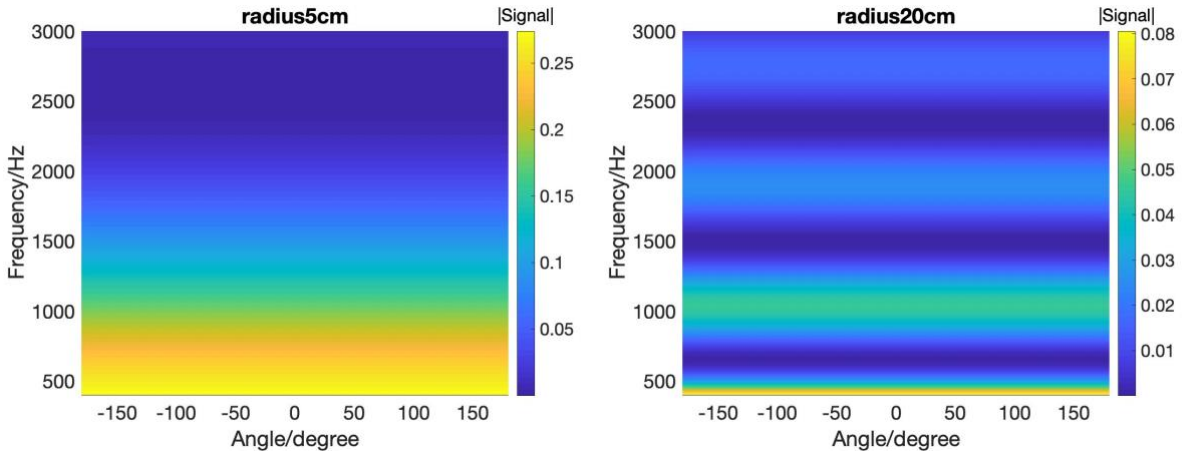


Figure 16 Continuous Circular Array with Different Radius

3.5. Linear Array Delay-and-Sum Beamformer

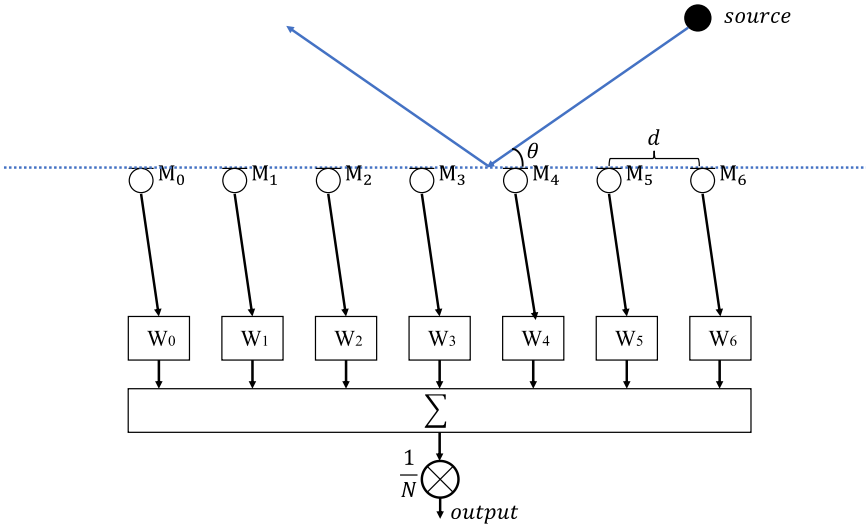


Figure 17 Discrete Linear Microphone Array with Delay-and-Sum Beamforming

In this section discrete linear microphone array with delay-and-sum beamforming is investigated. First, linear array and source positions are specified using the same method described in 3.4. Then delay-and-sum beamforming algorithm are implemented using the pseudocode shown below.

Pseudocode

```
source signal = sin(2*pi*frequency*t)
response list = []
for source position = 1 to end:
        mic output = fft(conv(impulse response, source signal))
        output = w .* mic output
        output = sum(output)
        response list.append(output)
end
plot(response list)
```

Where `mic output` is a matrix that contains all individual microphone output. `w` is the weighting function matrix which includes all microphone weighting function shown in Figure 17, i.e. $\boldsymbol{W} = [W_0, W_1, W_2, \dots, W_{N-1}]$. Operator `.*` represents element-wise multiplication. `sum()` adds up all rows.

In order to calculate the weighting function matrix, the direction array described in (2.12) and 3.2 needs to be calculated. Consider Figure 17, a sound source $s(t)$ arrives ULA at the direction given by θ. In order to produce an output that has a constructive interference at the angle of θ, the delay or advance for the n-th microphone needed is

$$\Delta_n = \left(\frac{d}{c}\right) \times \left(n - \frac{N+1}{2}\right) \times \cos\theta. \tag{3.1}$$

Where d is the spacing between microphones, $c = 340m/s$ is the speed of sound, N is the number of micrphone elements. This can be interpreted as taking the centre of the array as reference, microphone signals are delayed and advanced depending on the distance between microphone positions and the reference position.

(3.1) taking the Fourier transform gives

$$\boldsymbol{W}(\omega, n) = \exp\left(-j\omega \left(\frac{d}{c}\right) \times \left(n - \frac{N+1}{2}\right) \times \cos\theta\right). \tag{3.2}$$

Then the weighting function matrix can be calculated using

$$\boldsymbol{W} = \frac{1}{N}\begin{pmatrix} e^{-\boldsymbol{j}\omega_0 \frac{d}{c}\left(0-\frac{N+1}{2}\right)\cos\theta} & \dots & e^{-\boldsymbol{j}\omega_k \frac{d}{c}\left(0-\frac{N+1}{2}\right)\cos\theta} \\ e^{-\boldsymbol{j}\omega_0 \frac{d}{c}\left(1-\frac{N+1}{2}\right)\cos\theta} & \dots & e^{-\boldsymbol{j}\omega_k \frac{d}{c}\left(1-\frac{N+1}{2}\right)\cos\theta} \\ \vdots & \ddots & \vdots \\ e^{-\boldsymbol{j}\omega_0 \frac{d}{c}\left(N-\frac{N+1}{2}\right)\cos\theta} & \dots & e^{-\boldsymbol{j}\omega_k \frac{d}{c}\left(N-\frac{N+1}{2}\right)\cos\theta} \end{pmatrix}. \tag{3.3}$$

3.5.1. Spatial Aliasing and Grating Lobes

Using the algorithm derived above, it was found that different beam patterns can be obtained for different frequencies, shown in Figure 18. It can be observed that there are two side lobes

which are increasing in amplitude in dB as frequency increases, at around $f = 6000Hz$, the gain of the side lobes are larger than the gain at main lobes.

To explain this phenomenon, spatial aliasing needs to be introduced. In discrete spatial sampling, spatial aliasing arises when microphones captured signals' phase shifts are exactly integer number cycles for given direction of arrival at given frequency, shown in Figure 19, at angle φ, M_0 and M_1 receive far-field signal of frequency f where phase shift is exactly one cycle. When applying delay-and-sum beamforming algorithm, although this is not the steering angle, a constructive overlapping greatly increase the gain at this angle at this frequency.

For multi-element microphone array, a grating lobe arises due to the identical phase shift in all microphones, i.e. all microphones have integer number of cycles of phase shift with respect to others. This means that all signals recorded by all microphones are constructively overlapping and the gain at this angle at this frequency would be even higher than gain at look direction.

For one-dimensional linear array with equispaced elements, the position of the grating lobes can be calculated as

$$\sin(\theta) = \frac{\lambda}{d} \tag{3.4}$$

$$\theta = \sin^{-1}\left(\frac{k\lambda}{d}\right), where\ \lambda = \frac{c}{f} \tag{3.5}$$

$$\theta = \sin^{-1}\left(\frac{kc}{fd}\right). \tag{3.6}$$

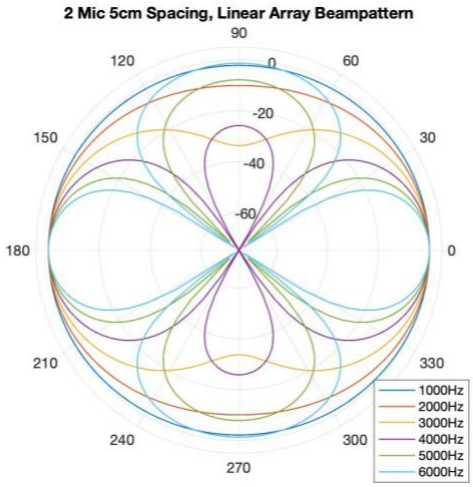


Figure 18 2 Mic 5cm Spacing Linear Array with DSB Beampattern

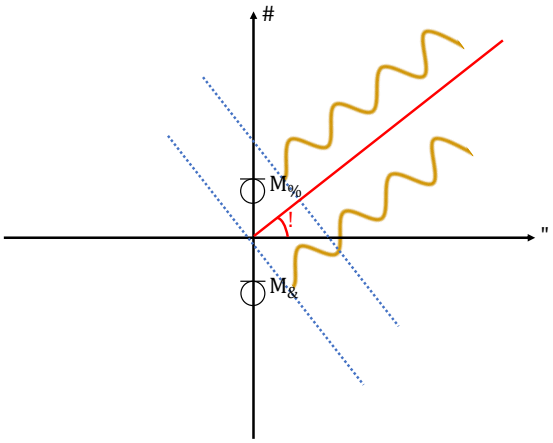


Figure 19 Identical Phase Shift

3.5.2. Effects of spacing

This section investigates the relationship between the microphone spacing and beam pattern.

As shown in Figure 20, a two-element linear microphone array with delay-and-sum beamforming is implemented, the spacing varies from $5cm$ to $105cm$.

In Figure 20, it can be observed that all plots have two main lobes situate at $-90°$ and $90°$. This is due to plotting reference and the power recorded have a $90°$ difference. The steering angle is still set to be $0°$.

From the figure, it can be observed that an increase in the microphone spacing without a change of the number of microphones would have

1. A decrease in the main, side and grating lobe width,
2. A decrease in frequency that grating lobe occurs,
3. A decrease in spatial aliasing frequency.

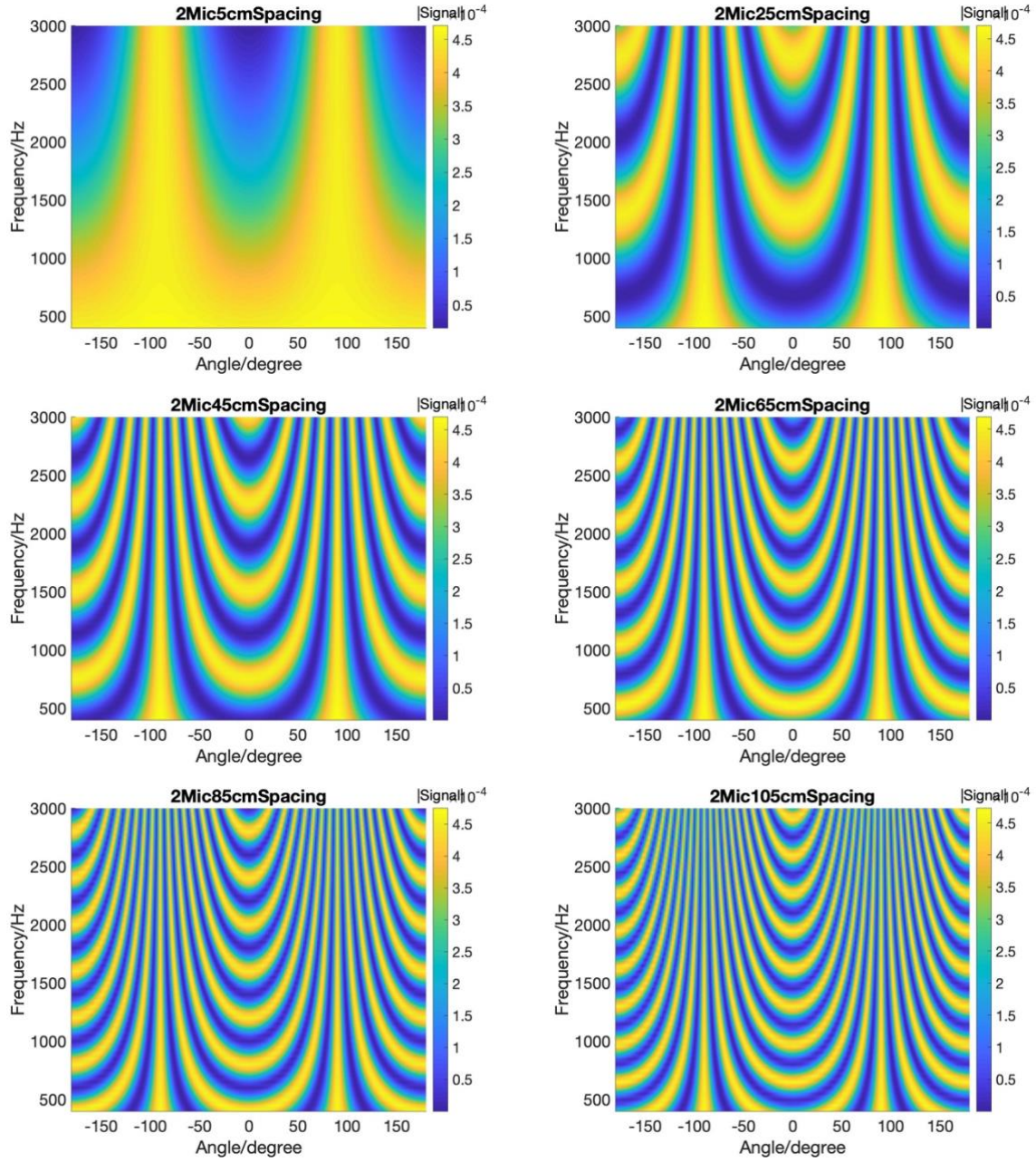


Figure 20 Beampatterns vary with microphone spacing(number of mic =2)

In Figure 21, when spacing increases to 45cm, it can be observed that the main lobe beam width at high frequency are already hard to see, in other word, distorted. This is a warning for designing microphone array, spacing which are too large may results in not capturing signal at high frequency.

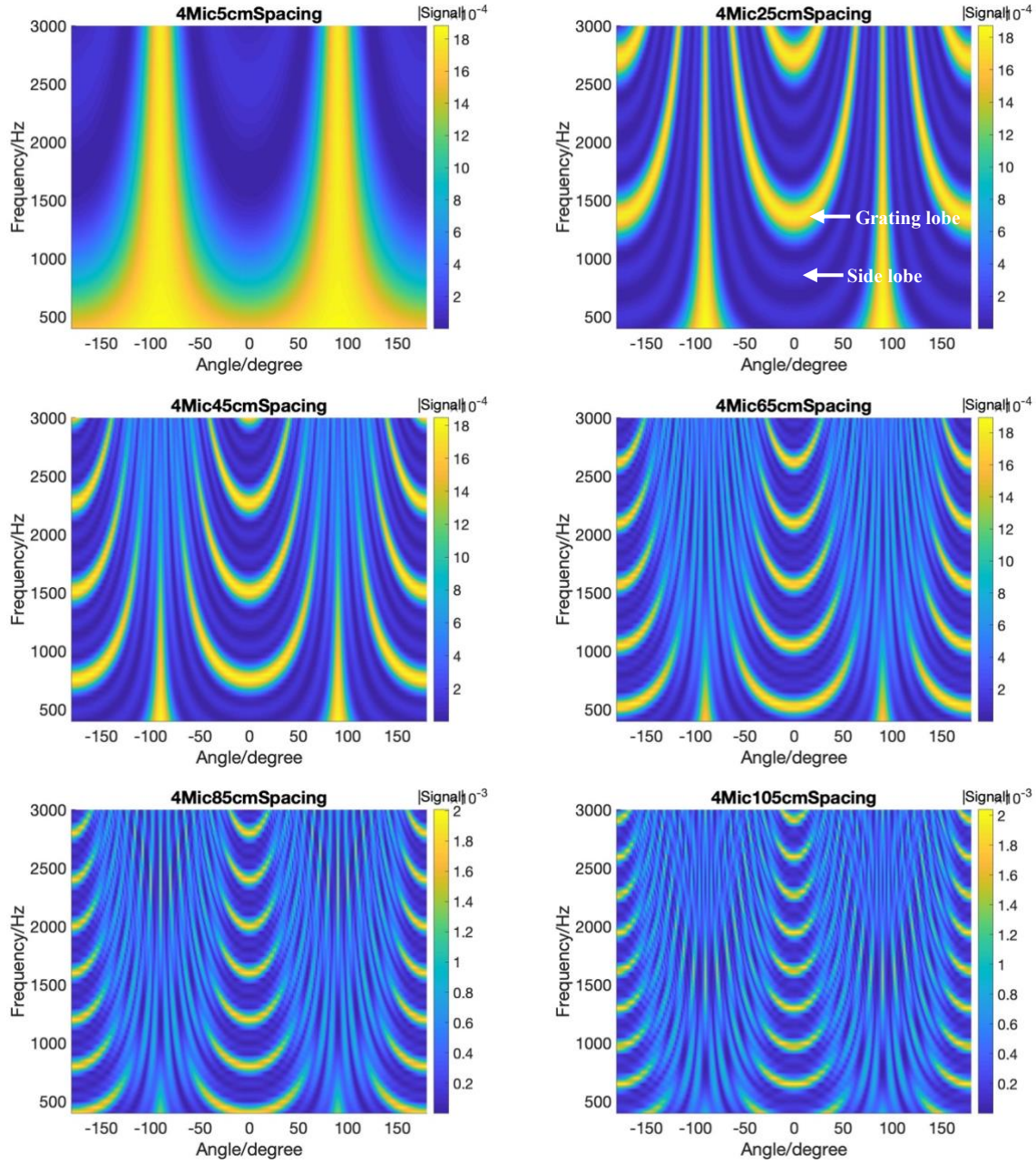


Figure 21 Beampatterns vary with microphone spacing(number of mic =4)

3.5.3. Effects of Number of Elements

Comparing plots in Figure 22, some characteristics can be conclude is an increase in the number of microphones without changing the microphone spacing leads to

1. An increase number of side lobes,
2. A decrease in main and side lobes beam width,
3. An increased attenuation of the side lobes.

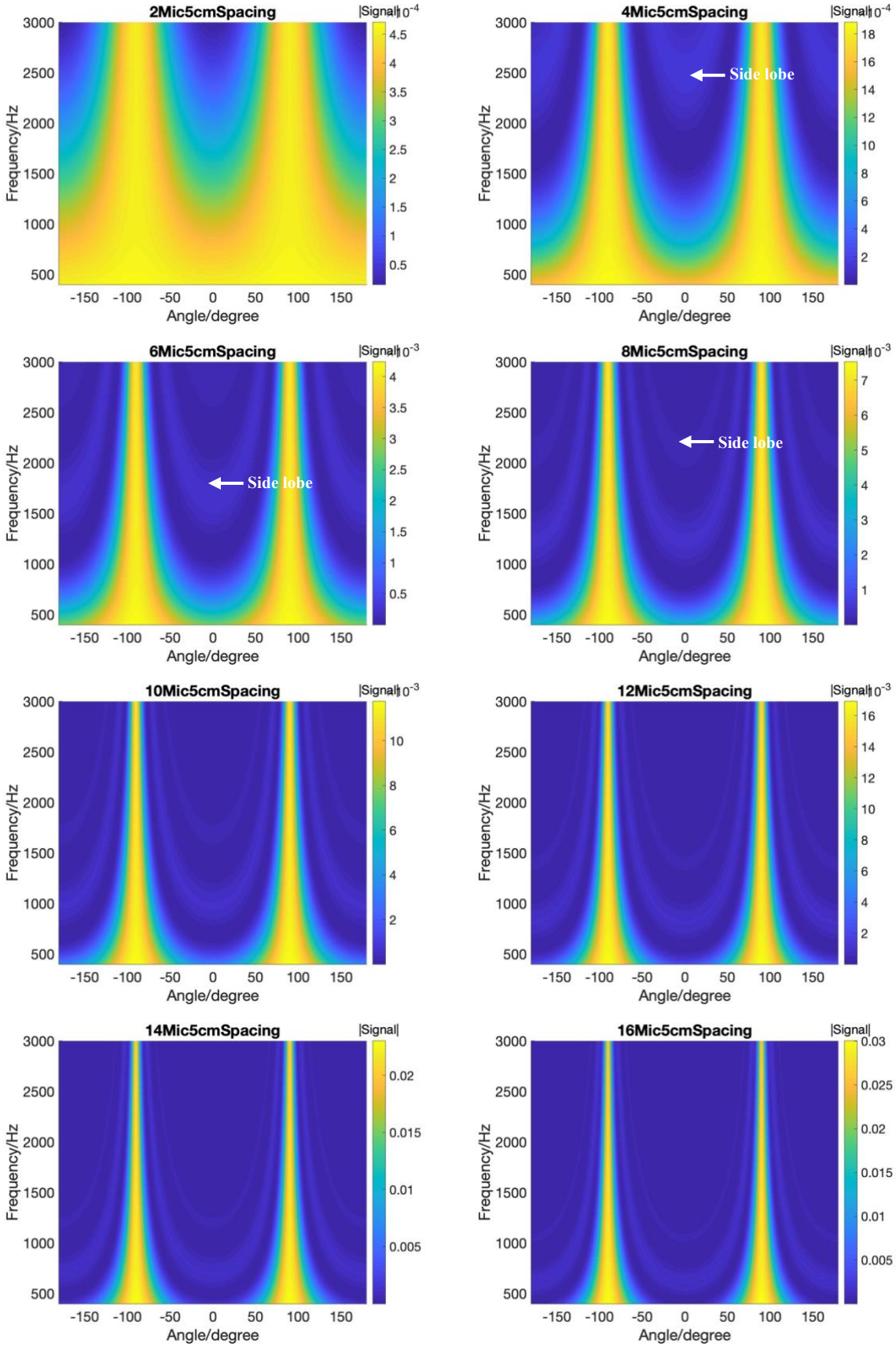


Figure 22 Beampatterns vary with microphone number(spacing of mic =5cm)

To investigate how number of microphone will affect the grating lobes, which are defined in 3.5.1, a larger spacing is needed for given range of frequency of interest.

In Figure 23, the beam patterns show that an increase in the number of microphones without changing the microphone spacing leads to a decrease in the beam width of the grating and side lobes, but it does not change the frequency that the first grating lobe appears.

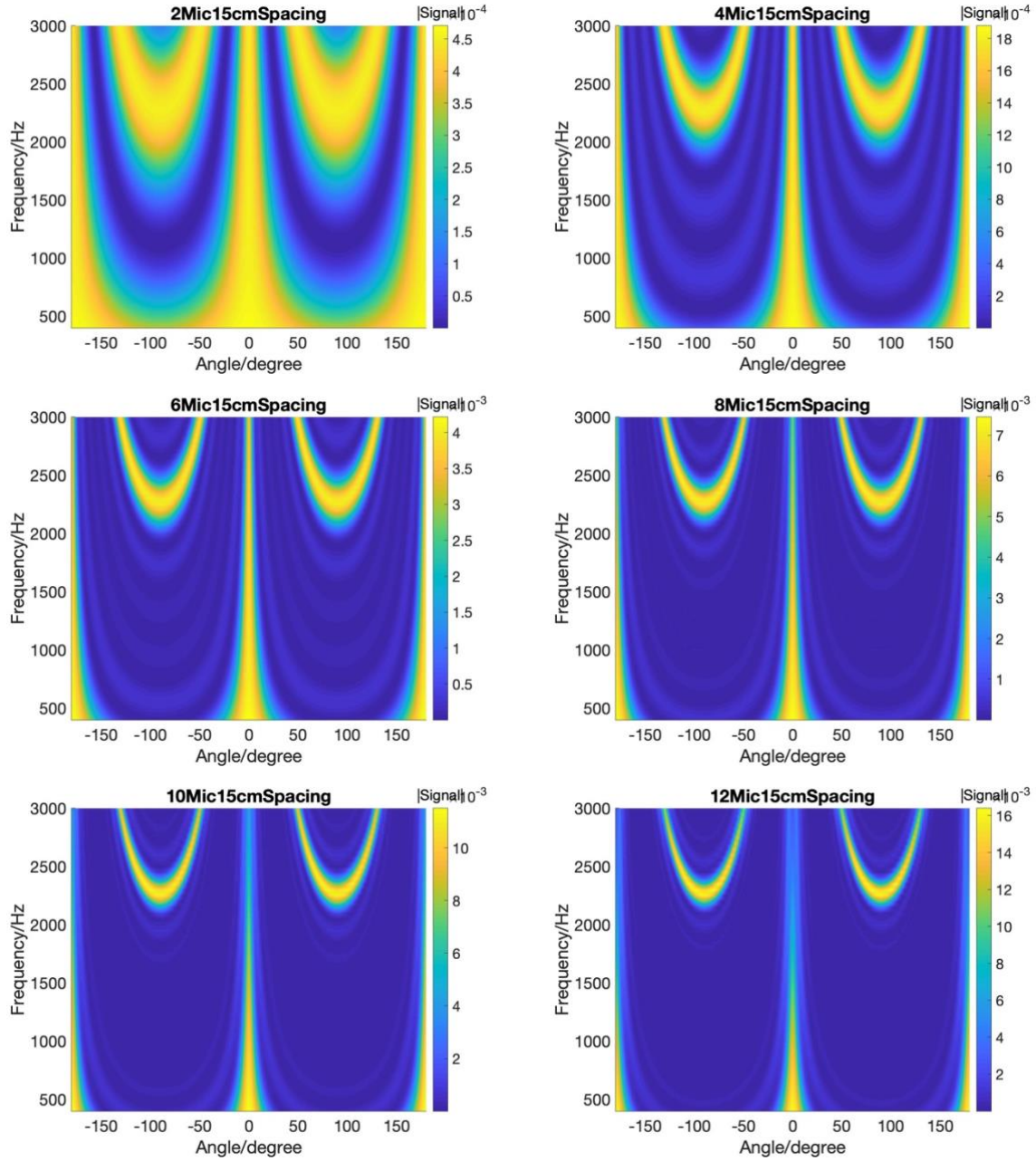


Figure 23 Beampatterns vary with microphone number(spacing of mic =15cm)

3.5.4. Steering Angle Analysis

This section investigates if linear array with applied algorithm could focus on certain direction while rejecting signals from other directions.

It can be seen that both front and back main lobes shift as steering angle changes, take 30°

steering angle as an example, the front main lobe shift right for 30° and the back main lobe shift the same amount but to the left. As steering angle increases up to 90°, front and back main lobes coincide at 90° and form a main lobe with larger beam width. As steering angle continues to increase, the front and back main lobe switch around.

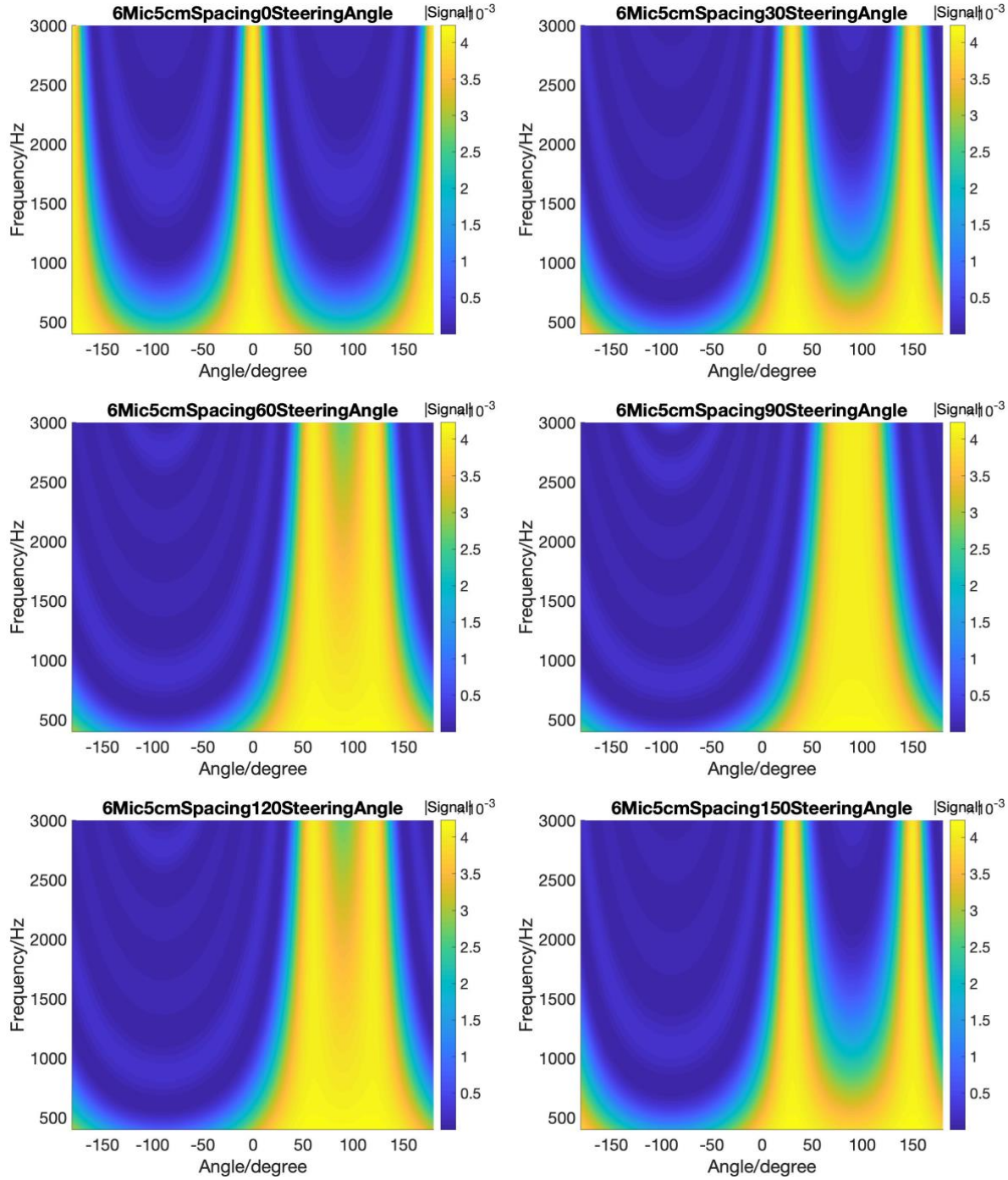


Figure 24 Linear Array Steering Angle Analysis

3.5.5. Speech From Different Direction

This section uses speech signal as input, which source are placed at different direction with respect to the linear array. Their outputs are plotted in time domain, as shown in Figure 25. These figures clearly indicate the front back ambiguity since the output reaches highest amplitude when signal angle are at 0° and 180°.

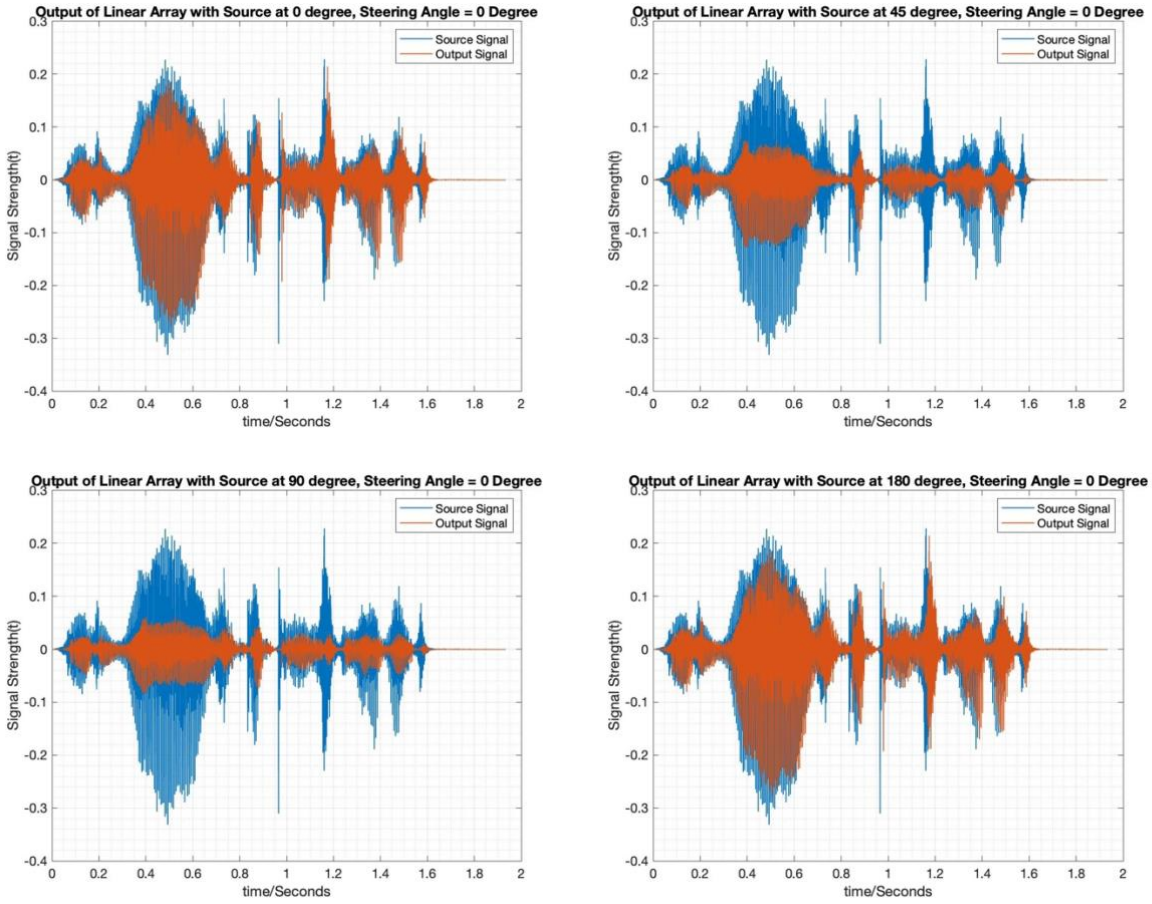


Figure 25 8 Mic 5cm Spacing Linear Array Time Domain Output(Source Angle = 0, 45, 90, 180 Degree)

Then competing signal are added to test whether the array can focus at the steering direction and rejecting unwanted signal from other direction. As shown in Figure 26, when competing source is located at 90°, the output signal is less affected by the unwanted source. However, when the unwanted source is situated at the back main lobe, the linear array fails to separate source and unwanted signals.

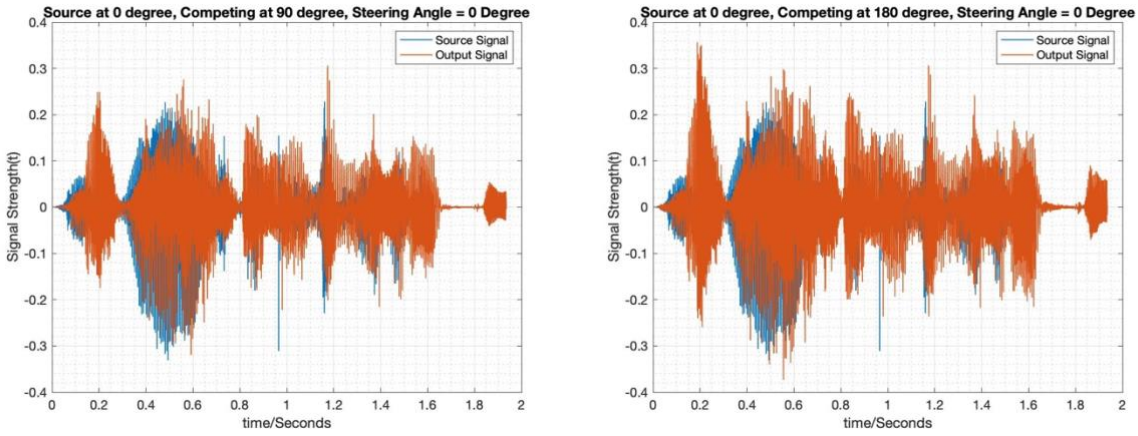


Figure 26 Linear Array Output when Competing Source is Added

3.6. Circular Array Delay-and-Sum Beamformer

Circular geometry are widely used in microphone array products since it does not gives front-back ambiguity. This section uniform circular array with delay-and-sum beamforming is implemented and corresponding performance and characteristics are investigated.

First, the microphone array and source position are set up again using method described in 3.4. Figure 27 shows a model where a 8-element circular microphone array is placed at center of a room. A sound source which is at the same plane as the array propagates source signal at angle φ with respect to the circular array center.

Similar to the delay-and-sum algorithm used in uniform linear array in 3.5, the algorithm implemented is shown below

Pseudocode

```
source signal = sin(2*pi*frequency*t)
response list = []
for source position = 1 to end:
        mic output = fft(conv(impulse response, source signal))
        output = w .* mic output
        output = sum(output)
        response list.append(output)
end
plot(response list)
```

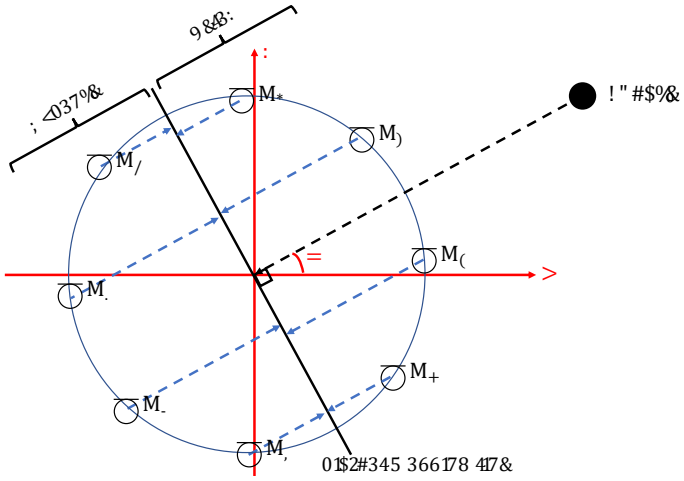


Figure 27 Two-dimensional Circular Array Arrangement

However, the direction array for circular array is different from direction array used in linear array. This is because for linear array, the reference mapping point where the signals are delayed or advanced to is a single point at the center of the linear array. However, for circular array, signals are mapped to a line which is perpendicular to the source direction that passes through the center of the circular array, which is the virtual mapping line shown in Figure 27. The virtual mapping line split the plane into two half circles, the half circle close to the source are delayed and the other part are advanced. As a result, there would be a constructive overlapping when summing all signals at the direction of the source after time shifting(delay and advance). Blue dashed lines shown in Figure 27 represent the amount of delay or advance required for each microphone.

Calculation can be done on the exact time that the signal recorded is needed for delay or advance. In Figure 28, the delay that M_1 needed to map to the line is

$$\tau_1 = \frac{r \times \cos(\varphi - \alpha)}{c}. \tag{3.7}$$

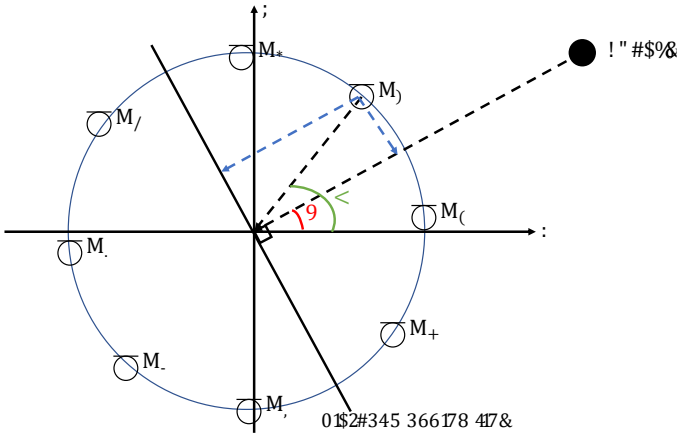


Figure 28 Calculating Direction Array for Circular Array

Where τ is the delay in seconds, r is the radius of the circular array, $c = 340m/s$ is the speed of sound. It can be generalize for the n-th microphone M_n

$$\tau_n = \frac{\mathrm{r}\times\cos(\varphi-\frac{2\pi n}{N})}{c}, n = 0 \dots (N-1). \tag{3.8}$$

Where N is the number of elements.

According to (3.8), taking the Fourier transform gives

$$\boldsymbol{W}(\omega, n) = \exp(-j\omega\frac{\mathrm{r}\times\cos(\varphi-\frac{2\pi n}{N})}{c}). \tag{3.9}$$

Where ω is the angular frequency.

Then, the delay-and-sum weighting function for circular array can be written as

$$\boldsymbol{W} = \frac{1}{N}\begin{pmatrix} e^{-j\omega_0\frac{d}{c}\cos\varphi} & \dots & e^{-j\omega_k\frac{d}{c}\cos\varphi} \\ e^{-j\omega_0\frac{d}{c}\cos(\varphi-\frac{(1)2\pi}{N})} & \dots & e^{-j\omega_k\frac{d}{c}\cos(\varphi-\frac{(1)2\pi}{N})} \\ \vdots & \ddots & \vdots \\ e^{-j\omega_0\frac{d}{c}\cos(\varphi-\frac{(N-1)2\pi}{N})} & \dots & e^{-j\omega_k\frac{d}{c}\cos(\varphi-\frac{(N-1)2\pi}{N})} \end{pmatrix}. \tag{3.10}$$

Using the algorithm derived above, an 8-microphone uniform circular array with 5cm radius with delay-and-sum beamforming are implemented. Different beam patterns can be obtained for different signal frequencies, shown in Figure 29. Just like in Uniform Linear Array, the main lobes beam width decrease as signal frequency increases. The amount of side lobes also

increases. The front-back ambiguity is solved.

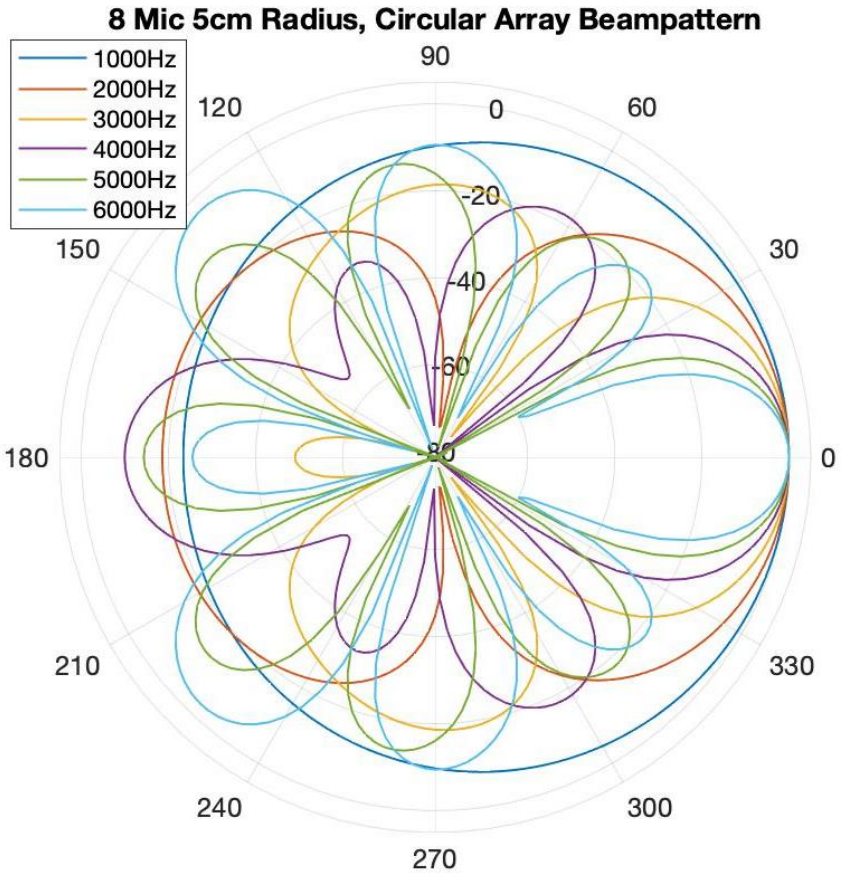


Figure 29 8 Mic 5cm Radius Circular Array with DSB Beampattern

3.6.1. Effects of Radius

This section investigates the relationship between the circular microphone array radius and beam pattern.

As shown in Figure 30, a four-element circular microphone array with delay-and-sum beamforming is implemented, the radius varies from $5cm$ to $105cm$. Similar to the result in 3.5.2, the beam width decreases as radius increases. But it can be noticed that the side lobes are no longer have the same shapes as in linear array, the pattern become arbitrary. This is because the microphone pairs that cause spatial aliasing are no longer in the same line, different spatial aliasing effects from different microphone pairs superpose together and form this arbitrary lobe pattern, more explanation will be given in 3.6.3. The other characteristics to note is that at large radius, the main lobe totally disappear(Figure 30, 85cm and 105cm radius). So when designing circular microphone array, the size of radius needs to be carefully considered, tradeoff between small beam width and the strength of the main lobe needs to be considered.

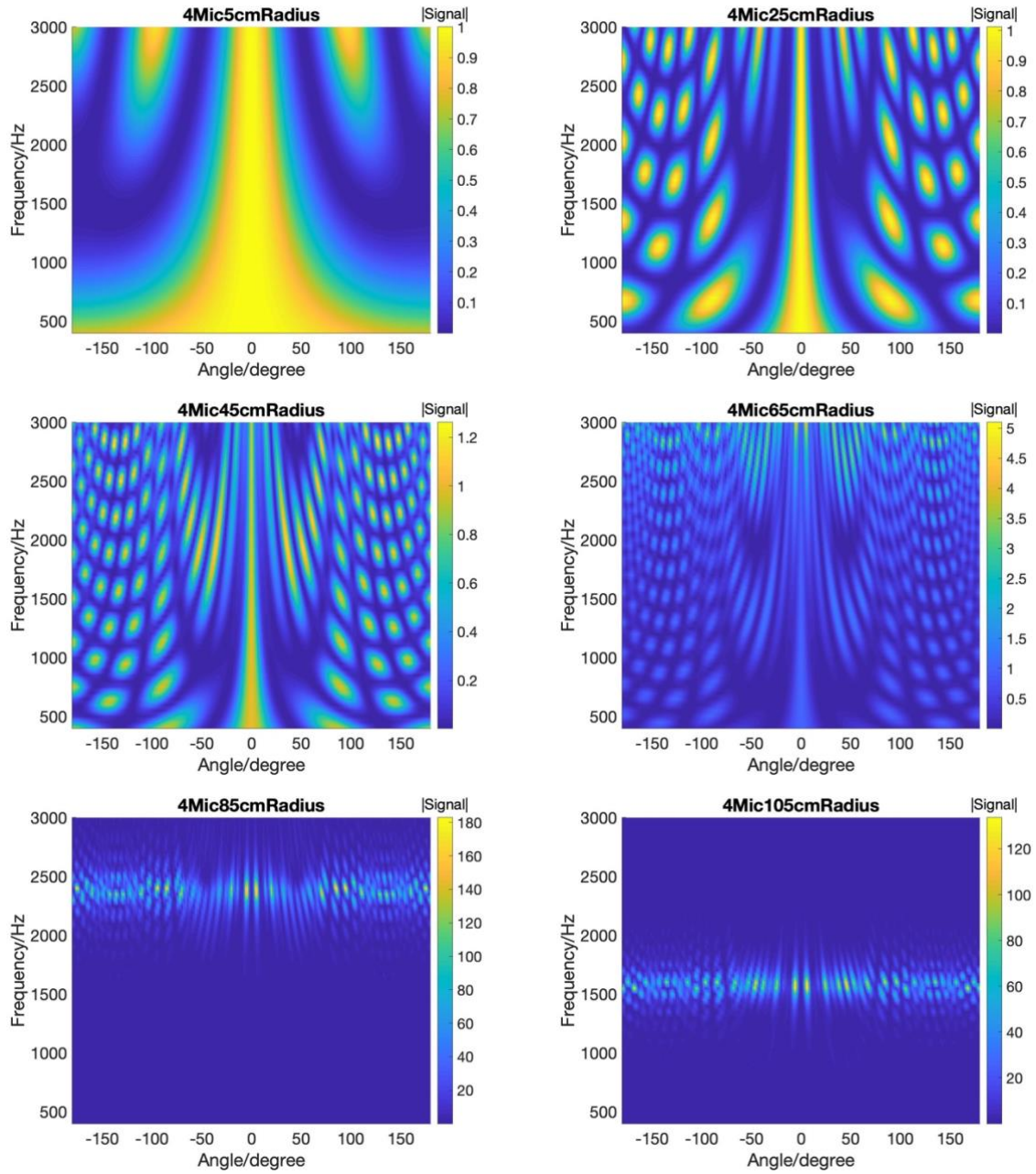


Figure 30 Beampatterns vary with array radius(number of mic =4)

3.6.2. Effects of Number of Elements

This section investigates how changing the number of elements would affect the circular array beampattern. Comparing plots in Figure 31, some characteristics can be conclude is an increase in the number of microphones without changing the circular array radius leads to

1. A decrease in main lobe beam width(only from 2-element to 4-element)
2. More complex pattern in arbitrary grating region
3. A decrease in amplitude of arbitrary grating region

4. An increase of number of side lobes near main lobe

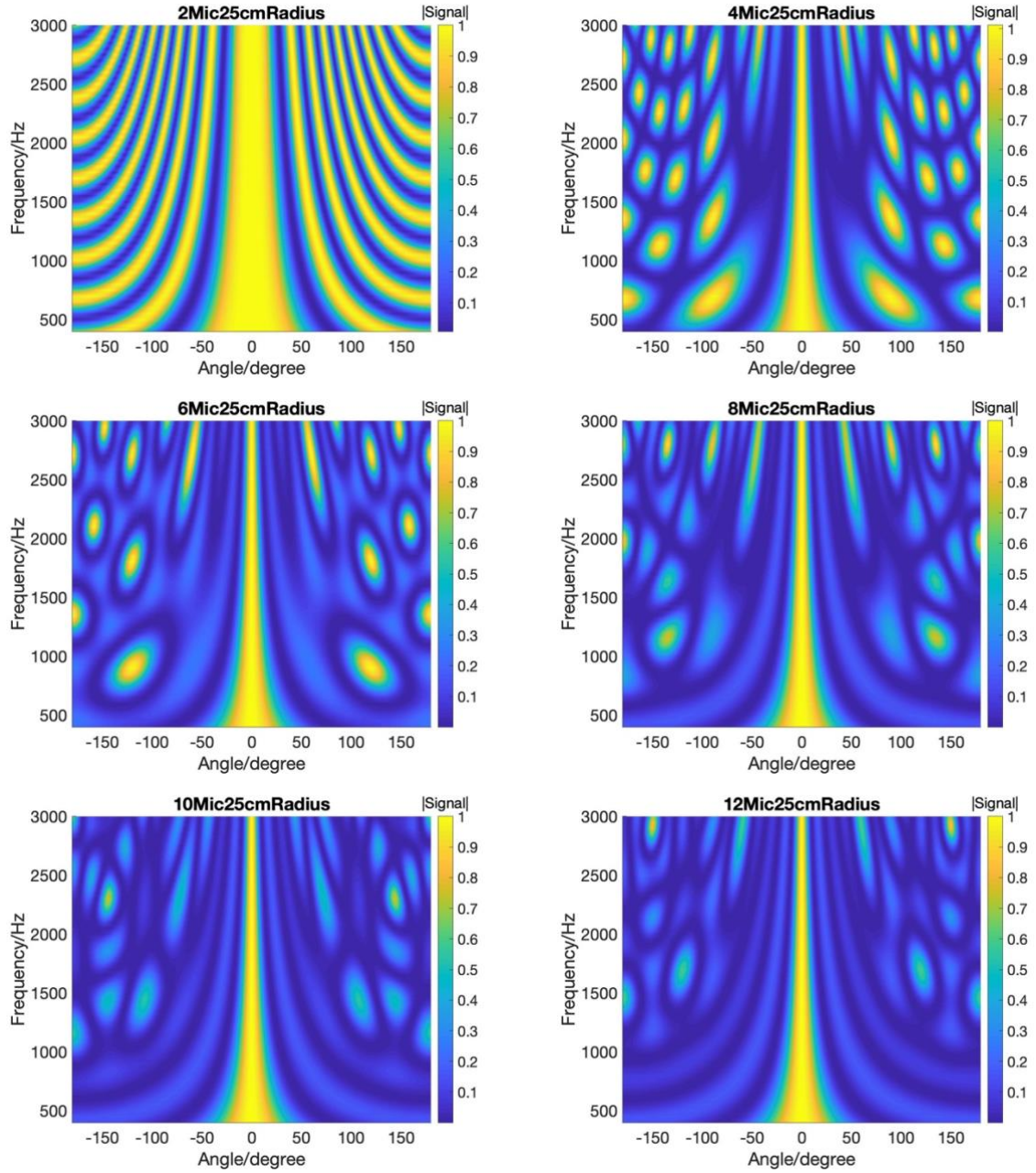


Figure 31 Circular Array Beampatterns vary with microphone number(radius of array =25cm)

3.6.3. Spatial Aliasing Analysis

As talked in 3.5.1, spatial aliasing arises when the phase shift of two or more microphones have a integer number of cycles difference. For circular array, the microphone-pair combination become much more complex. For example, a four-element circular array shown in Figure 32 has 6 microphone pairs, which are not in the same line as in ULA, thus, all of the pairs need to be analyzed individually. This is also the reason why the side lobes region become arbitrary. Same in ULA, grating lobes occur when phase shifts of all microphone pairs are different from

integer number of cycles.

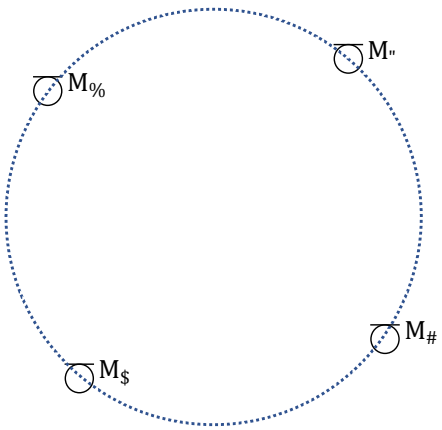


Figure 32 All Possible Microphone Pairs That Cause Spatial Aliasing

3.6.4. Steering Angle Analysis

This section studies if the applied algorithm can steer at different direction. As said before the source location is an pre-known information, so estimating the direction of arrival is not a topic in this project. However, given that knowing the source position, the algorithm needs to demonstrate the ability to steer at it. Figure 33 shows this ability: A six-element uniform circular array with 10cm radius is implemented. Different steering angles are applied and the beampatterns below shows that they all accurately steer at the angle required. One interesting thing to note is the arbitrary grating dots(or superposed grating lobes) shift in frequency when steering angle changes.

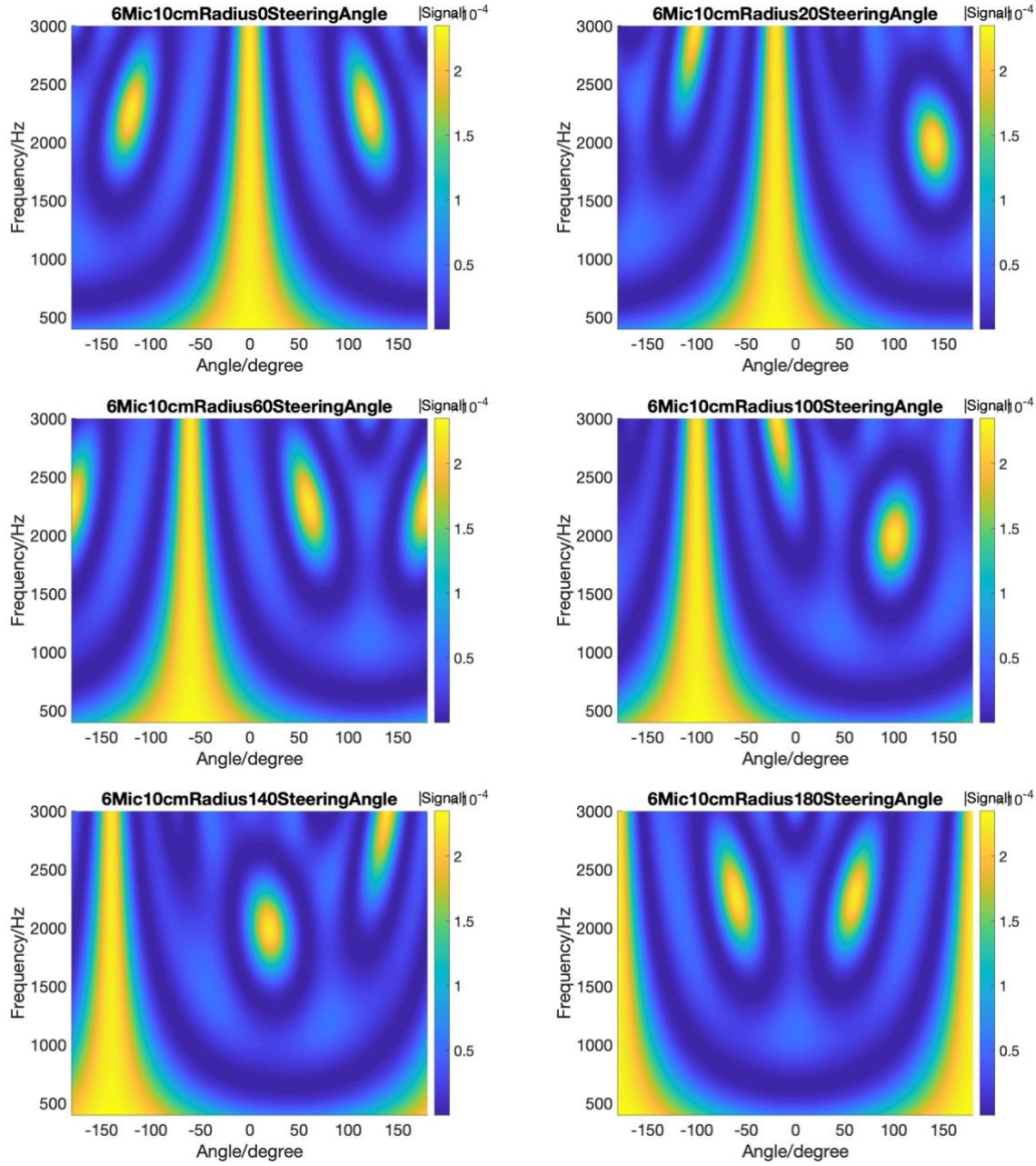


Figure 33 Steering At Different Angles(6 Mic 10cm Radius)

Then, a 2-element circular array with 10cm radius is implemented. 2-element circular array is a special case for circular array. As shown in Figure 34, a large main lobe beam width is observed when the steering angle is 0°. As the steering angle increases, the main lobe actually splits into two main lobes and exhibits front-back ambiguity. The beam width of two split main lobes has much smaller beam width. The grating lobes also slip in from high frequency toward low frequency between the gap of two main lobes.

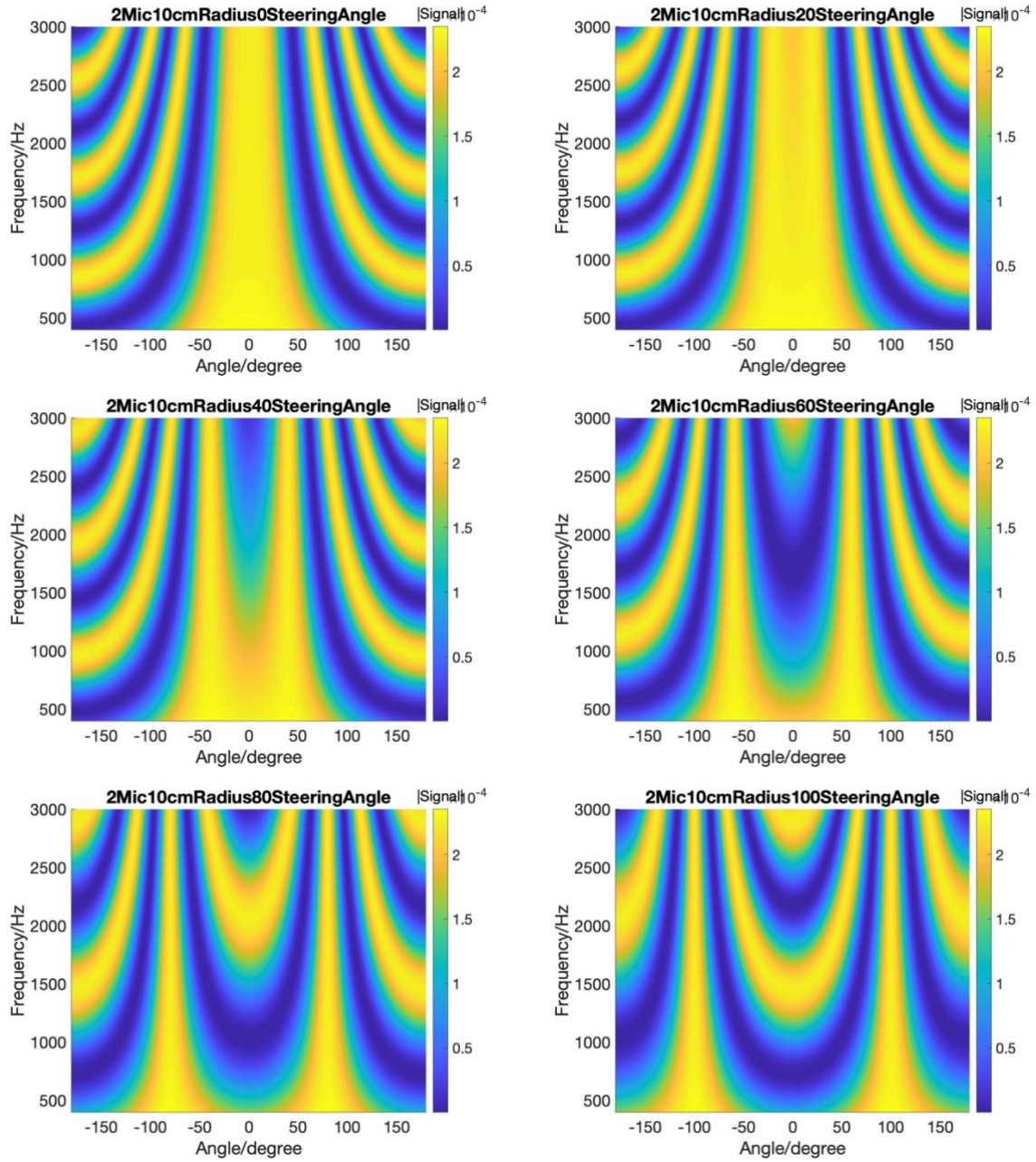


Figure 34 Steering At Different Angles(2 Mic 10cm Radius)

3.6.5. Speech from Different Direction

To examine the performance of designed microphone array in real environment, speech signal are used as the source signal and tested from different directions. First, same speech signal are used to test whether the microphone record higher amplitude in the steering angle than other directions. Then two different speech signals where one is at the steering direction and is defined as desired signal, and the other signal are recognized as a competing signal placed at an arbitrary direction other than desired direction. Outputs are plotted in time domain.

The speech signal resources uses Open Speech and Language Resources(OpenSLR) [21], which is a corpus of 16kHz read English speech. Speech signal used is randomly chosen from this corpus.

Firstly, polar plot similar to Figure 18 and Figure 29 is generated to show the beampattern when the input signal is speech signal which contains complex multiple frequencies, shown in Figure 39. The speech signal used is 121-127105-0033.flac(frequency and time domain shown in Figure 35), which is a 2 seconds woman speech contains with speech “It was the beauty of it.”. The polar plot shows that the circular array exhibits strong directivity and has an approximately 15dB different between steering direction and the direction oppose it.

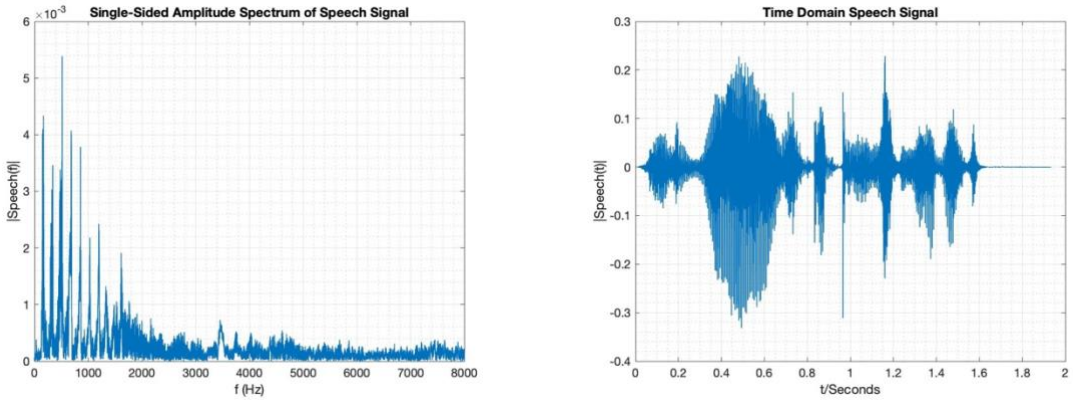


Figure 35 Frequency and Time Domain Plot of Input Speech Signal

It can be seen that most of the power situates between 0Hz to 2000Hz. However, Figure 36 shows that the beam pattern at frequency between 0Hz to 250Hz is almost uniform, which has not directivity for this frequency range. While at higher frequency, the beam pattern becomes desirable, main lobe has much higher strength.

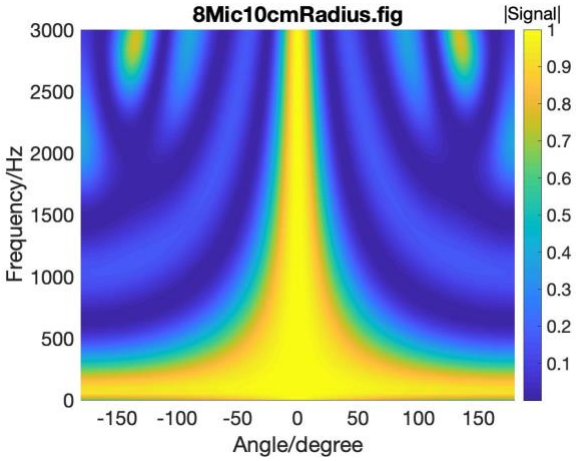


Figure 36 Circular Array Beampatterns(8 Mic 10cm Radius)

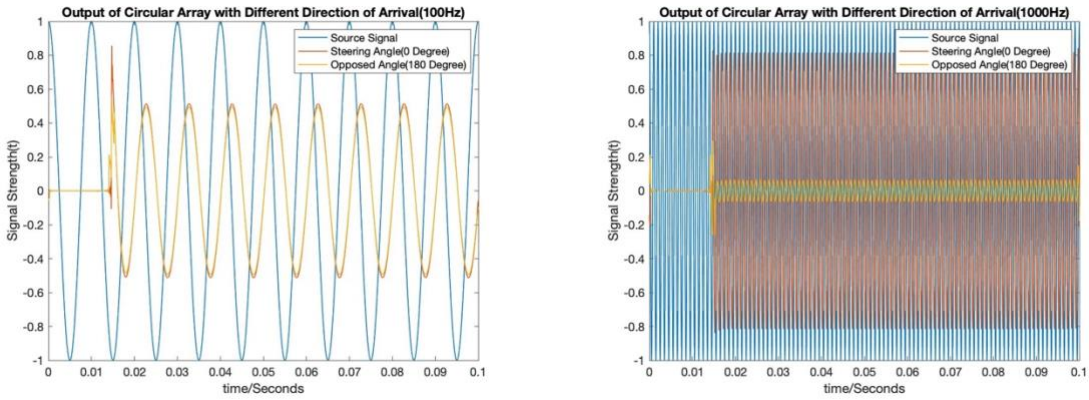


Figure 37 Outputs When Signal Are Sinusoid Wave(Left: 100Hz, Right: 1000Hz)

So the time domain output when the input signal is speech signal is plotted in Figure 38. It can be seen clearly that the signal placed at steering direction has higher amplitude than the opposed direction, average power of the output signal can be calculated:

Signal	Power
At Steering Angle(0°)	0.0011
At Opposed Angle(180°)	2.956×10^{-4}

So the signal to interference ratio can be calculated: $10\log(\frac{0.0011}{2.956\times10^{-4}}) \approx 13.14dB$, Which is close to the beam pattern plotted in Figure 39.

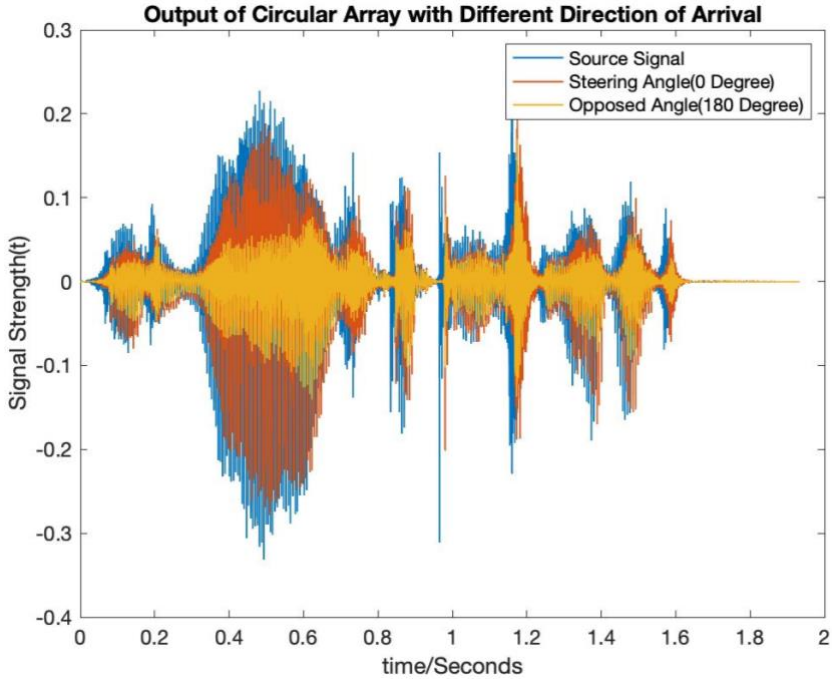


Figure 38 Time Domain Outputs Comparison(Speech Signal)

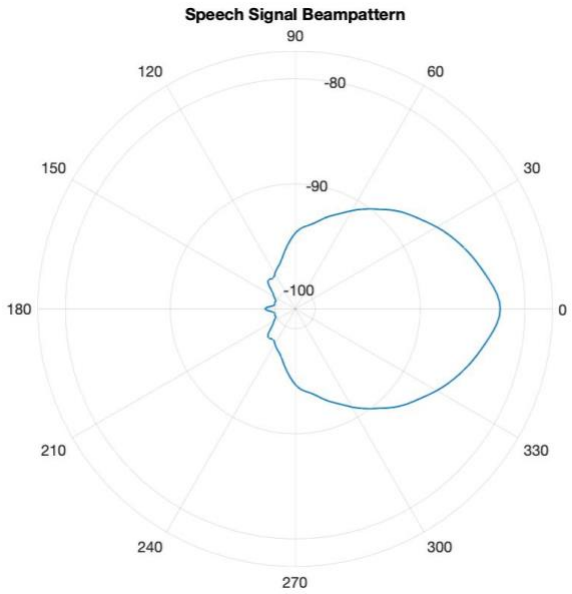


Figure 39 Polar Plot When Input is Speech(8 Microphone 10cm Radius)

Then, two different speech signal are placed at the same plane of the array, one at the look direction φ_1 and the other at some arbitrary direction φ_2, to test if the microphone array can focus on the speech at the look direction. The signal model is shown in Figure 40.

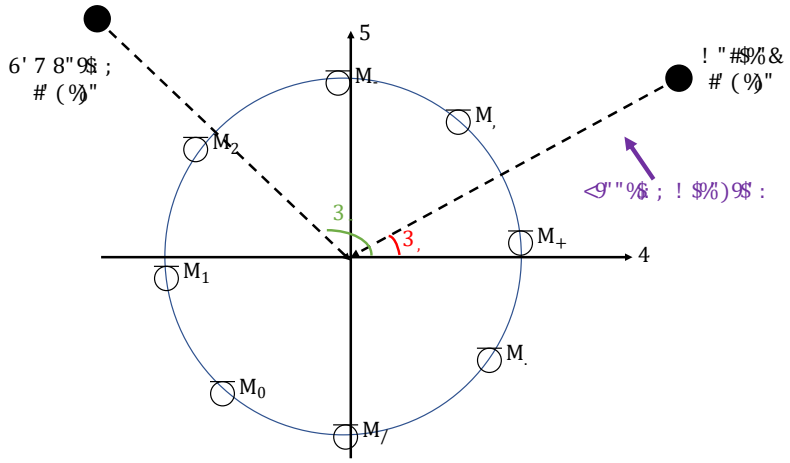


Figure 40 Signal Model for Competing and Desired Signals

Now consider single microphone M_n where two source signals $s_d(t)$ and $s_c(t)$ are arriving from different angles, as shown in Figure 41. Different transfer functions $h_d(t)$ and $h_c(t)$ could be obtained using RIR Generator since signal positions is known. So the output of microphone M_n can be written as the sum of two convoluted source signals, assuming no noise is present:

$$x_n(t) = \sum_{i=0}^{1} s_i(t) * h_i(t). \tag{3.11}$$

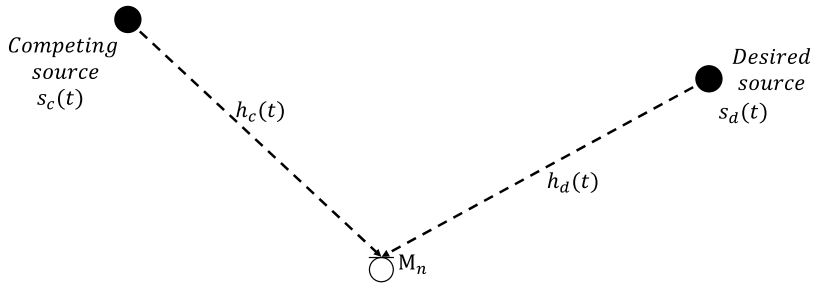
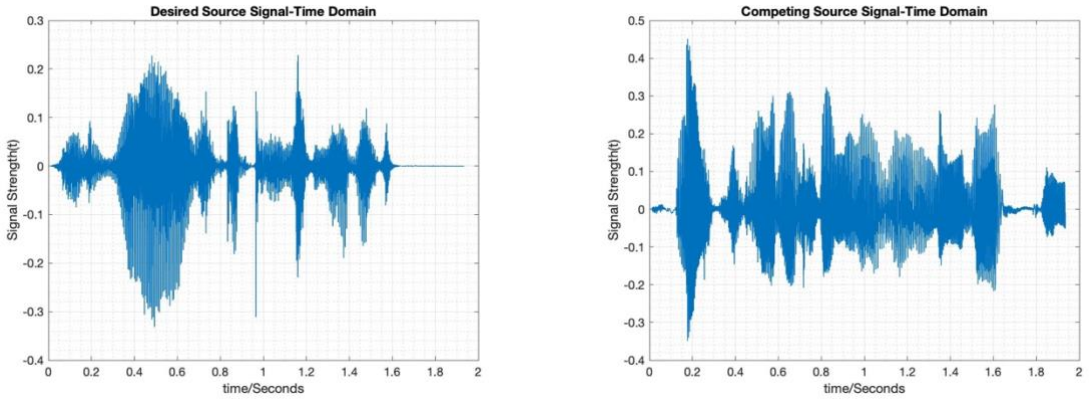


Figure 41 Single Microphone Capturing Signals

To simplify calculation, competing source are cut such that it has the same array size as the desired source signal. Desired source signal and competing source signal after this process are shown in Figure 42.



Desired Signal Speech Content: "It was the beauty of it."	Competing Signal Speech Content: "Mr. Edison was a leader far…(cutted)"

Figure 42 Desired Source Signal and Competing Signal in Time Domain

Placing the desired source signal at steering direction(0°) and competing signal at opposed direction(180°). Using the derivation above, the output signal of the array can be obtained, which is shown in Figure 43. It shows that the output signal has very similar shape to the desired signal. Using *sound()* function in MATLAB, the output signal sounds like the desired signal with a small amount of competing signal added.

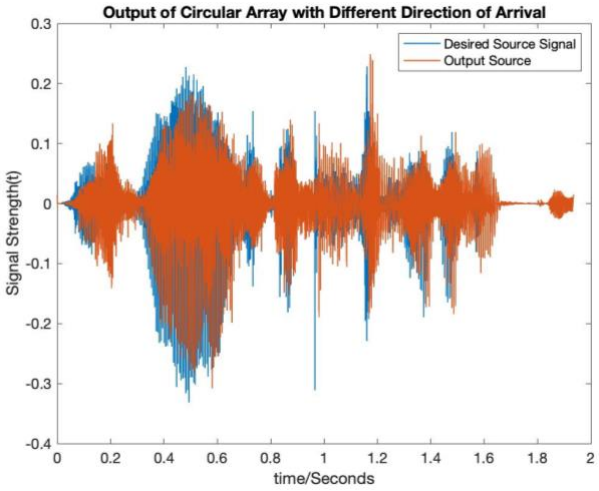


Figure 43 Output and Source Comparison

To show the performance of using circular array, single omni-directional microphone which is placed at the center of the room with same source signals configuration is implemented, for comparison. The result is shown in Figure 44. It can be shown that the output signal is just the sum of desired and competing signal, testing by *sound()* function demonstrates that output signal is just adding two sources together.

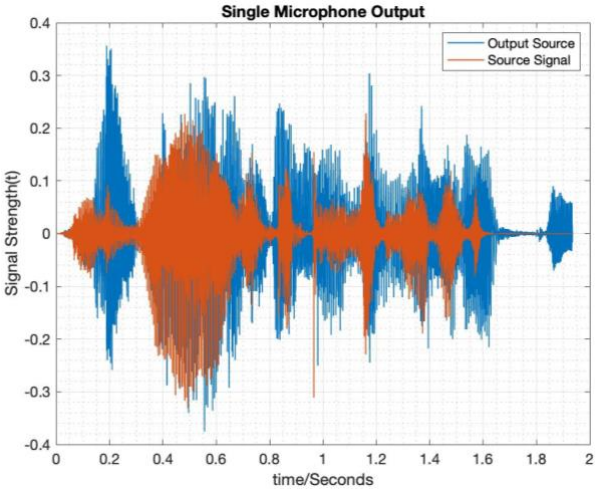


Figure 44 Single Omni-directional Microphone Output

Also compare to linear array performance, which is shown in Figure 26, the circular array does not suffer from front back ambiguity, the output signal is also a better replication of the desired signal.

3.6.6. Elevation Study

So far all the investigations are done based on the fact that the source and the array are at the same height, this is obviously unrealistic in real life. Take smart speaker scenario as an example, average standing Mr. and Ms. usually have heights of 1.78m and 1.64m [22], while the smart

speaker are usually placed on a table, usually with a height of 0.9m. So how the system behaves when elevation is taken into account is also a crucial problem.

To start with, we need to understand that the direction array described in (3.10) are no longer valid since a lack of elevation information.

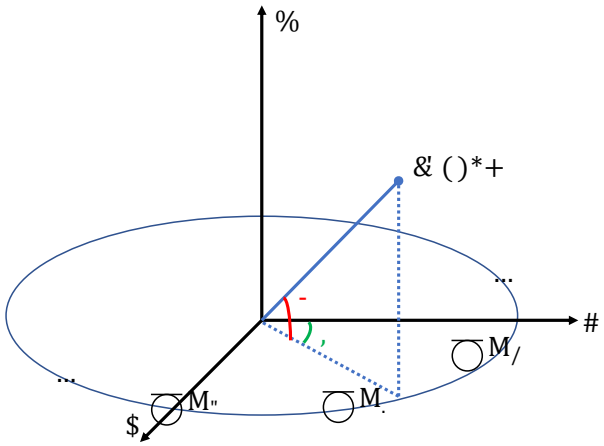


Figure 45 Three Dimensional Array Arrangement

Consider the three-dimensional circular array system shown in Figure 45. Compare to two dimensional coordinates system in Figure 27, this system is only extended by an additional angle θ [23], so the weighting function can be reconstructed as

$$\boldsymbol{W} = \frac{1}{N}\begin{pmatrix} e^{-j\omega_0\frac{d}{c}\cos\varphi\cos\theta} & \cdots & e^{-j\omega_k\frac{d}{c}\cos\varphi\cos\theta} \\ e^{-j\omega_0\frac{d}{c}\cos(\varphi-\frac{(1)2\pi}{N})\cos\theta} & \cdots & e^{-j\omega_k\frac{d}{c}\cos(\varphi-\frac{(1)2\pi}{N})\cos\theta} \\ \vdots & \ddots & \vdots \\ e^{-j\omega_0\frac{d}{c}\cos(\varphi-\frac{(N-1)2\pi}{N})\cos\theta} & \cdots & e^{-j\omega_k\frac{d}{c}\cos(\varphi-\frac{(N-1)2\pi}{N})\cos\theta} \end{pmatrix} \tag{3.12}$$

Using the weighting function derived above, beampatterns which vary with both elevation and azimuth for a given signal frequency can be plotted. Figure 46 shows the beampatterns of a uniform circular array of 16 elements and 25cm radius, where the elevation steering angle varies from 0° to 90°. The azimuth steering angle is kept at 0°. It can be seen that the main lobe exhibits "up-down" ambiguity, i.e. there are two main lobes located in positive and negative elevation angles. For elevation steering angle near 0°, two lobes "stick" together and as a result for the angles between ±elevation steering angle display very high gain.

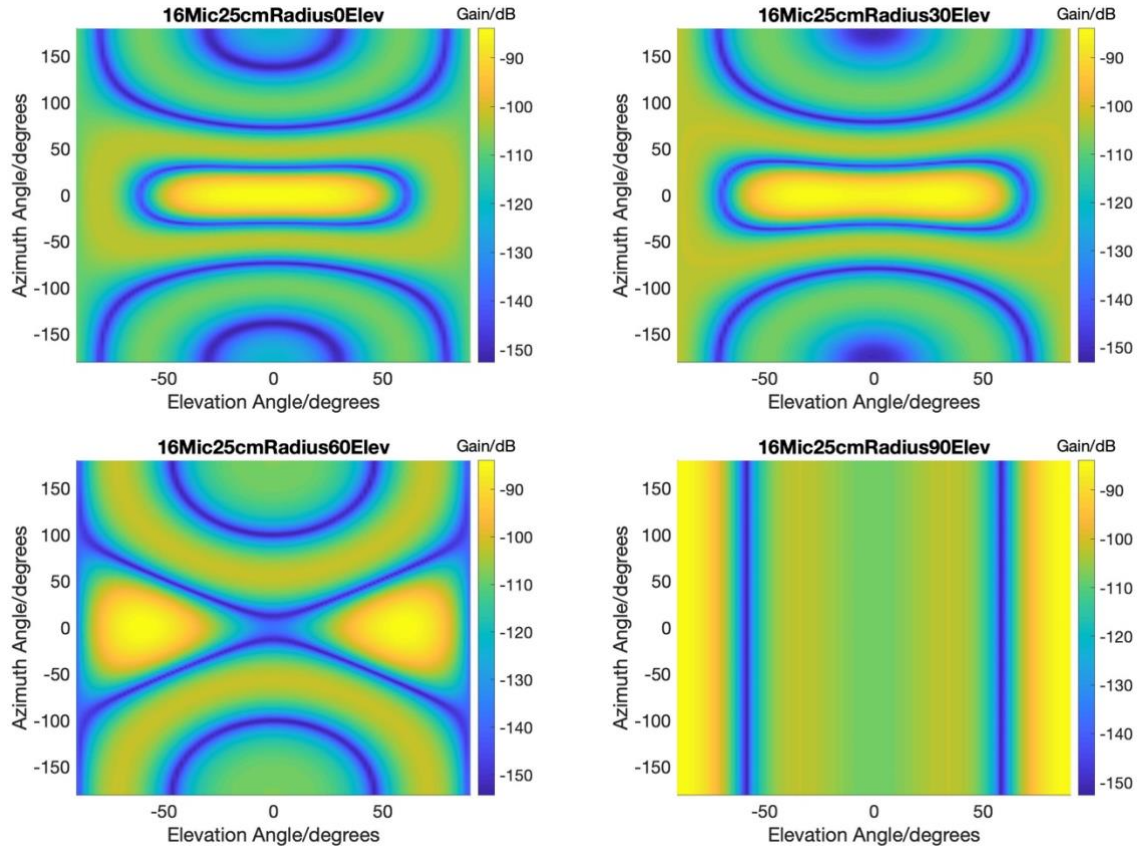


Figure 46 Beampatterns vary with Elevation and Azimuth(Signal Frequency = 1000Hz)

To investigate how would signal frequency influence the elevation beam pattern, just similar to the study for azimuth beam pattern done before, the following attempts on varying signal frequency is done.

The sinusoid's frequency varies from 1000Hz to 6000Hz, the elevation steering angle is set to be 30°, azimuth steering angle 0°. The beampatterns are shown in Figure 48. In the previous study in how azimuth beam width vary with signal frequency, it can be predicted that the higher the signal frequency, the smaller the azimuth beam width. In Figure 48, the same pattern can be observed since the main lobe becomes thinner as frequency increases. In terms of elevation, the beam width also become thinner as the linkage between two main lobes at ±elevation steering angle becomes weaker and weaker and finally the main lobes are separated. On the other hand, the pattern for side lobes and grating lobes become much more complex as signal frequency increases. This can be interpreted as: at low signal frequency, the microphone pairs which have short distance apart do not exhibit spatial aliasing effects, the spatial aliasing effects at this time is caused by long distance microphone pairs(Figure 47). As frequency increases, the spatial aliasing effects of this short distance microphone pairs occurs so more aliasing effects superpose together and thus a more complex pattern.

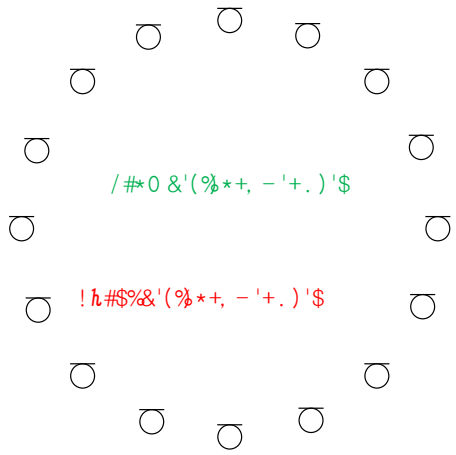


Figure 47 Short and Long Distance Microphone Pairs

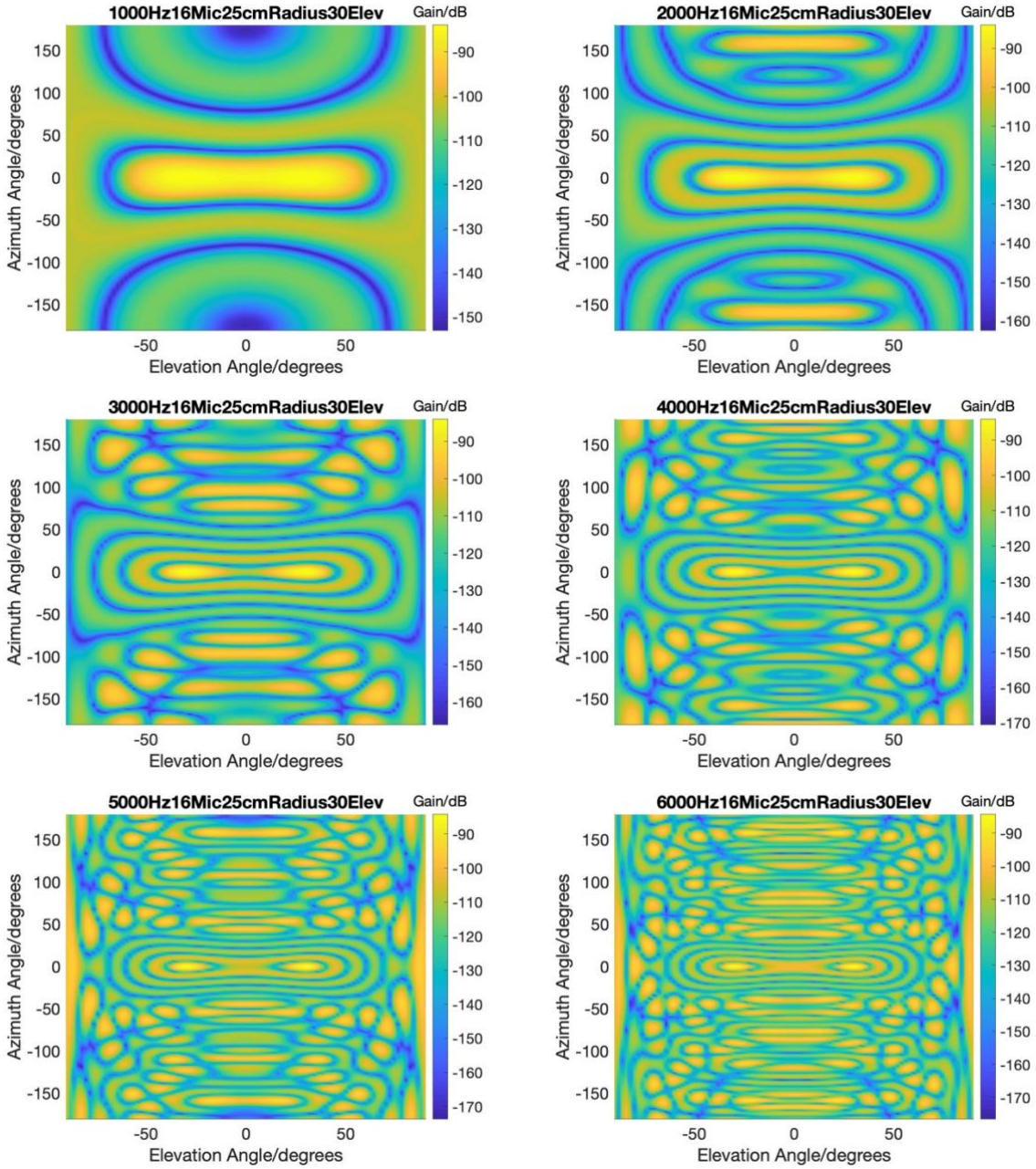


Figure 48 Beampatterns(Elevation) Vary With Signal Frequency

Then, how varying circular array radius would affects elevation beam patterns is studied. Similar to azimuth beam pattern, an increase in array radius will result in

1. A decrease in elevation beam width,
2. A more complex pattern in the side and grating lobes region.

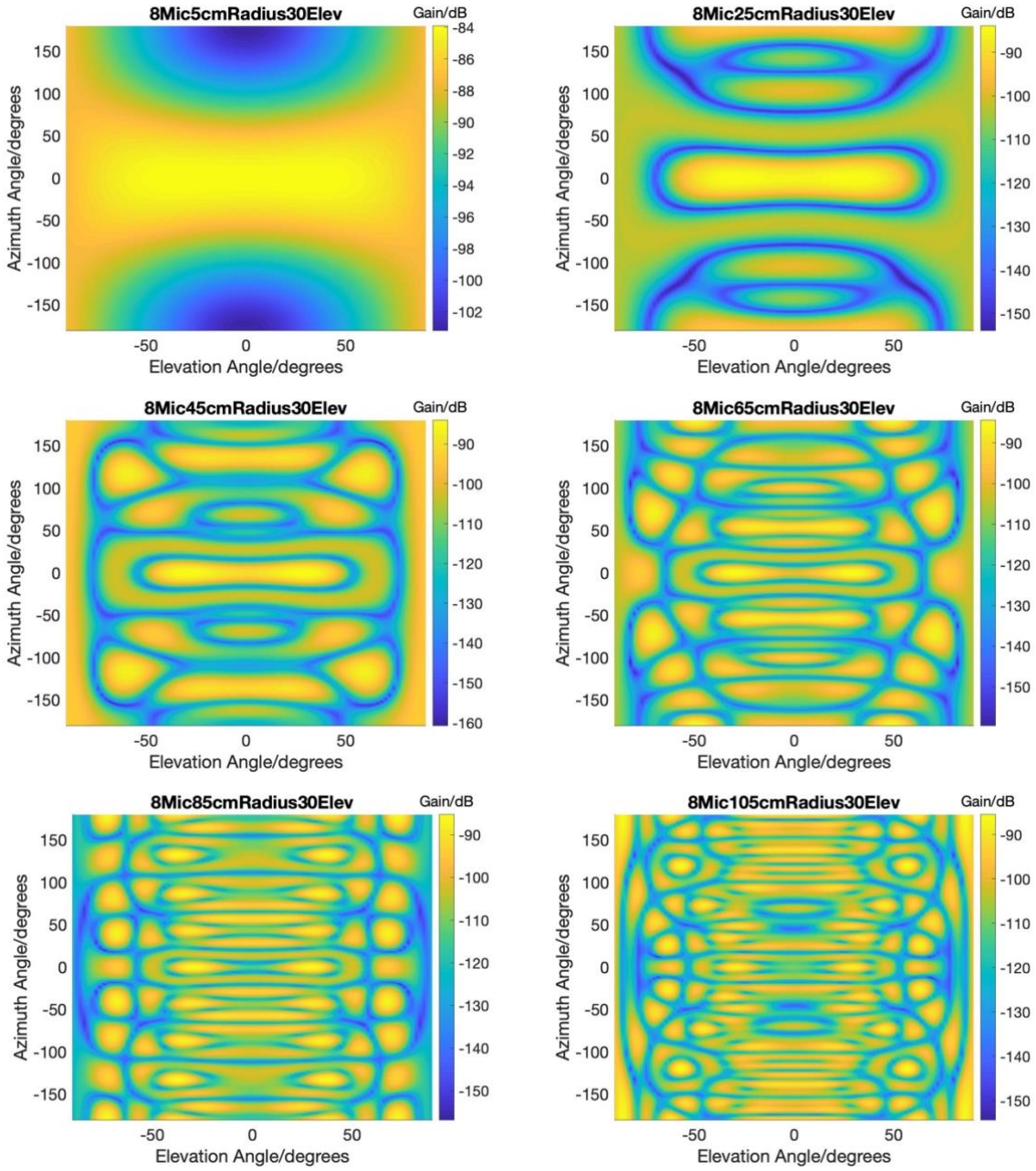


Figure 49 Beampatterns(Elevation) Vary With Circular Array Radius

Finally, varying microphone number is observed to have little change for main lobes' behavior. But it does decreases the strength of the side and grating lobes, as shown in Figure 50.

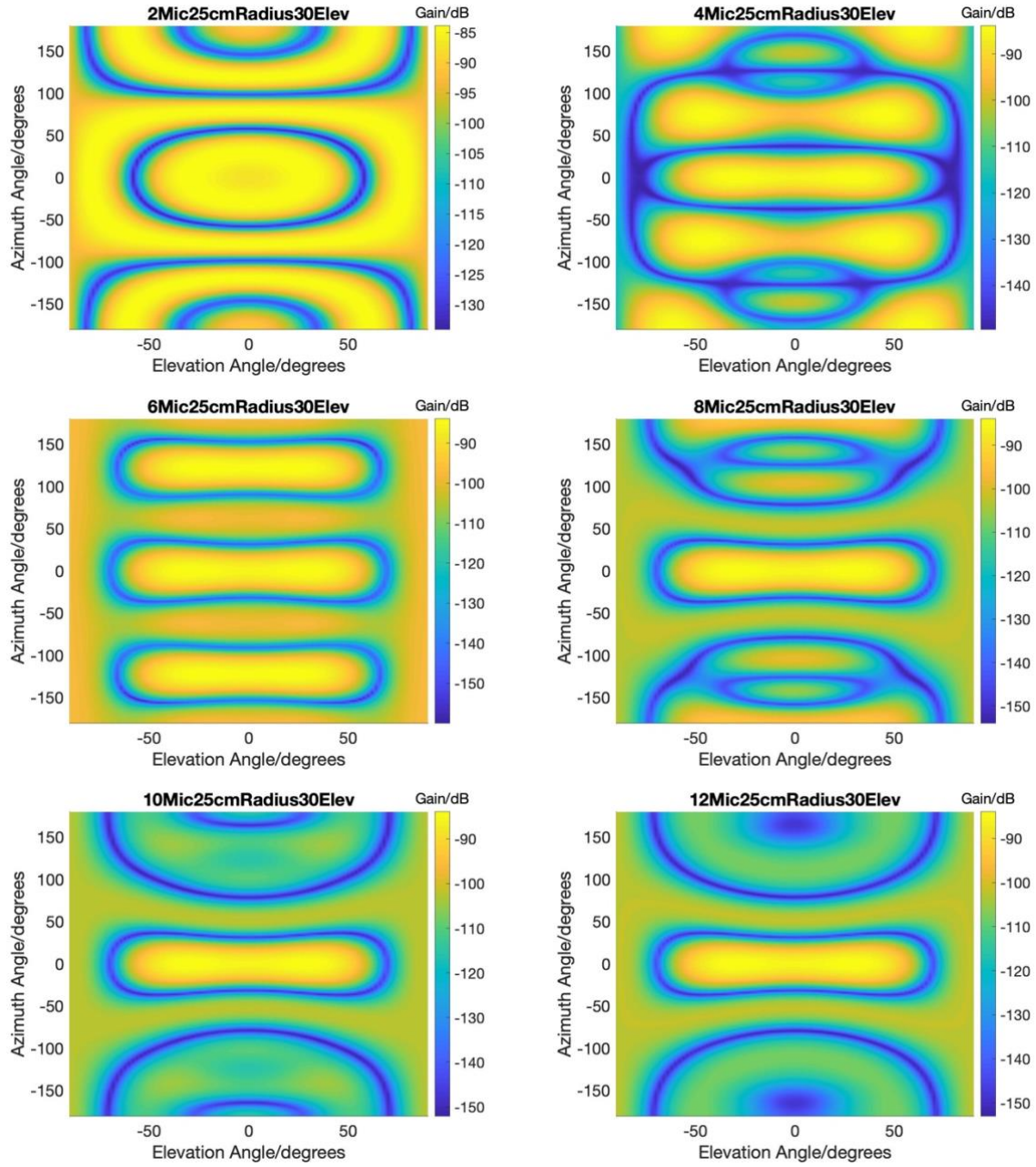


Figure 50 Beampatterns(Elevation) Vary With Number of Microphones

3.7. Superdirective Beamformer

In this section, superdirective(MVDR) beamforming algorithm is implemented. MVDR beamforming is a data dependent algorithm, which ideally, the noise is known. This is clearly impossible in reality, however, to investigate the circular array with this type of beamforming, noise is a known information while implementing the beamforming algorithm.

In this simulation, VOICEBOX [24] is used in order to add noise to each microphone output. First, after calculating the output of each microphone in the array, a acoustic noise is added to

of signal-to-noise ratio of 20dB is added to each output, using the command in VOICEBOX package: *v_addnoise*.

Then the noise arrays added in each microphone can be extracted. Using this noise arrays, the covariance matrix at each frequency can be calculated using the equation

$$\boldsymbol{R}_{nn}(\omega) = \boldsymbol{n}[\omega]\boldsymbol{n}[\omega]^{H}. \tag{3.13}$$

Where $\boldsymbol{n}[\omega]$ is the noise amplitude at frequency ω. $\boldsymbol{n}[\omega]\boldsymbol{n}[\omega]^{H}$ gives a $N \times N$ matrix and N is the number of microphones. The covariance matrix is then scaled by dividing its expected value.

After obtaining the noise covariance matrix, equation in (2.45) or (2.46) can be used. The time domain output of both MVDR and DSB algorithm implemented are shown in Figure 51, when only desired signal and white noise exist. One can be observed is that two algorithms result in almost the same performance in reducing the white noise. It can be shown that when the array is exposed in white noise, the MVDR beamformer becomes DSB, which means the DSB is the best beamformer when the noise at each channels are uncorrelated.

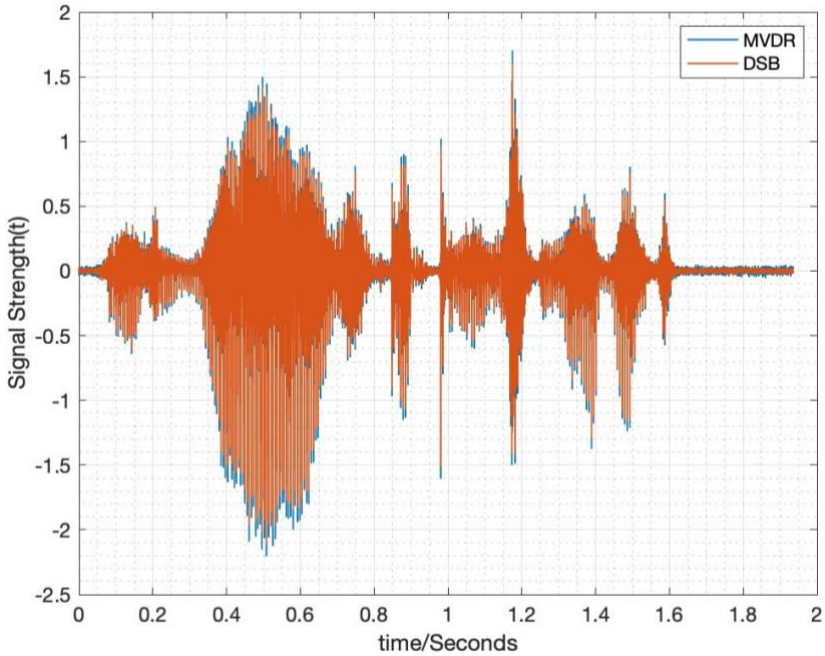


Figure 51 Comparison Between MVDR and DSB When 20dB White Noise Is Added

4. Testing

During simulation process, some approaches are used to get better visualization or accommodate software immanent characteristic.

4.1. Manipulation on frequency component

Function to split frequency around(direction array and fft)

When calculating the direction array stated in (2.12), the frequency used for calculation is $[-8000, \dots, 8000]$Hz, which includes the negative frequency components. However, when calculating each microphone outputs in frequency domain using the *fft()* function, the frequency components contain frequency of $[0, \dots, 16000]$Hz. Using the positive and negative frequency symmetric property for a real signal, the output signal is adjusted such that it can has the same frequency range as the direction array in order to multiply with the direction array during applying the beamforming algorithm. Figure 52 shows the result of this process, after this process, the zero frequency is located at the center of the vector, where it is the first element before the manipulation.

In order to use the *ifft()* function to transform the signal to time domain after beamforming, the same approach is used to transfer back the frequency range to $[0, \dots, 16000]$Hz.

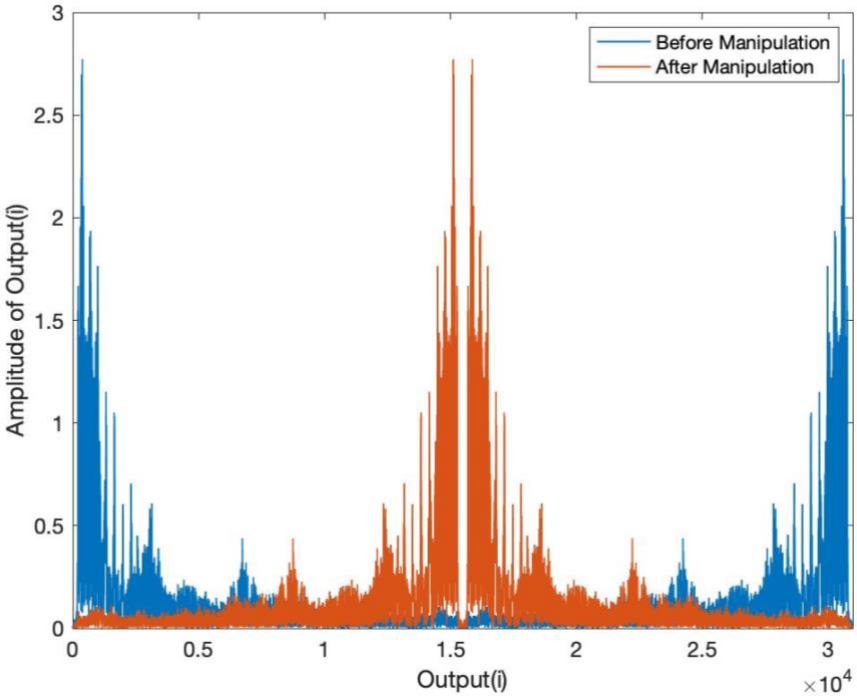


Figure 52 Before and After Manipulating the Frequency Content of the A Microphone Output Signal

4.2. Extra Gain Added

In some simulations to produce output signal(e.g. Figure 38, Figure 43, Figure 44), due to long

distance between the source and receiver, the signal outputs by the beamformer is attenuated and the amplitude is very small. An additional gain is added after implementing beamforming algorithm to amplify the output signal amplitude back to the original level. In Figure 53, the output before and after adding a gain of 50 as well as the source signal is plotted.

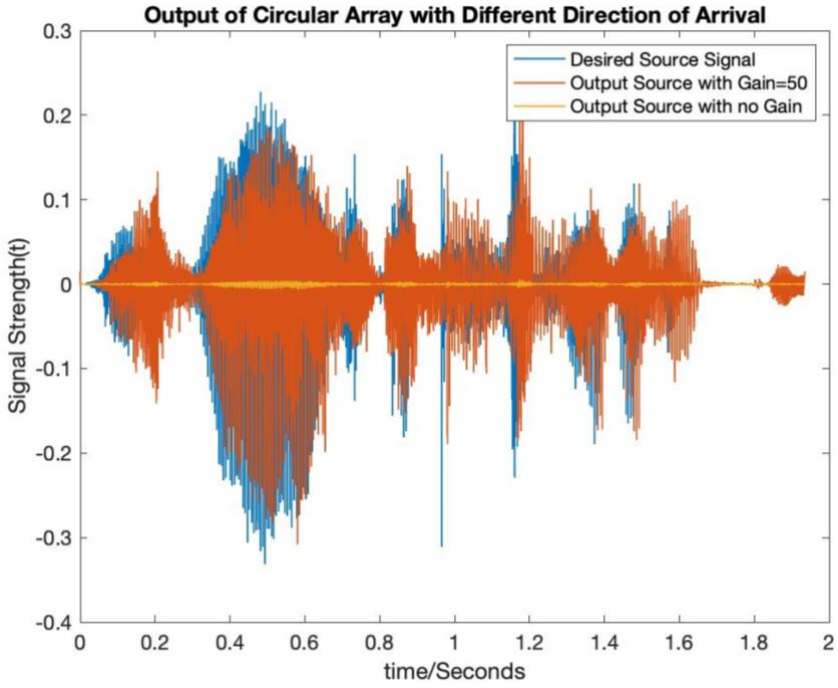


Figure 53 50 Gain vs. No Gain

5. Conclusion

5.1. Evaluation

The aim of this project was to investigate the characteristics and performance of two types of microphone array: Linear Microphone Array and Circular Microphone Array. Also to answer two questions asked in objectives section: “Is the performance of a microphone array applicable?” and “Is circular microphone array or linear microphone array better off in real life application?”. In real life application, most of the microphone array in smart speaker on the market is circular. The simulation result in this project validated that microphone array has generally better performance than single microphone, and circular array has generally better performance than linear array, given that the source position is known.

Both circular and linear array with delay-and-sum beamforming algorithm were implemented. Parameters such as number of elements and microphone spacing are varied to study the geometry-performance relationship. Beampatterns and time domain outputs were plotted to visualized performances. For circular array, elevation steering angle was also analyzed. All simulation shows that circular array is more robust and more applicable in real life application, since it overcomes front-back ambiguity and the main lobe at 90° does not exhibit a wider width.

Circular array with MVDR beamformer was implemented, the characteristics that when white noise was the only noise exist, the MVDR beamformer becomes DSB was proven.

In summary, the questions asked were answered, microphone array’s peculiarity was investigated. This projects demonstrated that if source position is pre-known, circular microphone array can achieve better performance compared to linear array and single microphone.

5.2. Further Work

In summary, this project successfully answer the objectives question, however, more improvements can be done if more time is available. Some of them are listing below:

- Estimation on direction of arrival can be studied. This project assumed that the source direction is a pre-known information, however, in realistic application, most of the time this information is unknown and estimation is needed. The accuracy of estimation would directly affects the performance of the microphone array.

- It can be noticed that the directivity for both linear and circular microphone array at low frequency are omni-directive, this can be improved by adjusting the spatial parameters such as number of elements or spacing between element. However, more can be done on the algorithm plane.
- In MVDR beamforming simulation, more realistic noise environment such as bubble noise can be added to discriminate the performance between MVDR and DSB.
- To evaluate the performance more realistically, physical microphone array should be built and tested, different room dimensions and noise environments can more practically reveal the actually performance of two types of array and their performance differences in different situations.

Bibliography

[1] B. V. Veen and K. Buckley, “Beamforming: a versatile approach to spatial filtering,” *IEEE ASSP Magazine,* vol. 5, no. 2, pp. 4-24, April 1988.

[2] A. Nordrum and K. Clark, “5G Bytes: Beamforming Explained,” [Online]. Available: https://spectrum.ieee.org/video/telecom/wireless/5g-bytes-beamforming-explained. [Accessed 25 May 2019].

[3] “Microphone array - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Microphone_array. [Accessed 25 May 2019].

[4] J. Koetsier, “Smart Speaker Users Growing 48% Annually, To Hit 90M In USA This Year,” Forbes, [Online]. Available: https://www.forbes.com/sites/johnkoetsier/2018/05/29/smart-speaker-users-growing-48-annually-will-outnumber-wearable-tech-users-this-year/#6f88bddb5dde. [Accessed 2 May 2019].

[5] M. Brandstein and D. Ward, "Constant Directivity Beamforming," in *Microphone Arrays: Signal Processing Techniques and Applications*, Berlin, Springer, 2001, pp. 3-4.

[6] R. Feynman, “47 Sound. The wave equation,” [Online]. Available: http://www.feynmanlectures.caltech.edu/I_47.html. [Accessed 25 May 2019].

[7] “Chapter 8 n-dimensional Fourier Transform,” [Online]. Available: https://see.stanford.edu/materials/lsoftaee261/chap8.pdf. [Accessed 25 May 2019].

[8] “Plane wave - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Plane_wave. [Accessed 5 May 2019].

[9] C. F. Cardoso, "Signal Processing for Circular Microphone Arrays," Southampton, 2007.

[10] S. Elliott and P. A. Nelson, “Active noise control,” *IEEE Signal Processing Magazine,* pp. 12-35, 1993.

[11] “Bessel Function of the First Kind,” Wolfram MathWorld, [Online]. Available: http://mathworld.wolfram.com/BesselFunctionoftheFirstKind.html. [Accessed 6 May 2019].

[12] A. Abdulla, H. M. Luai, Z. M. Sami, A. Saif, A. Nazar and W. Luis, “Linear and Circular Microphone Array for Remote Surveillance: Simulated Performance Analysis,” in *BCS International IT Conference 2013*, Abu Dhabi, 2013.

[13] C. F. Cardoso, “Signal Processing for Circular Microphone Arrays,” University of Southampton, 2007.

[14] J. Bitzer and K. U. Simmer, “Superdirective Microphone Arrays,” in *Digital Signal Processing*, Berlin, Springer, 2001.

[15] J. L. Flanagan and E.-E. Jan, “Microphone arrays for speech processing,” in *Proceedings of ISSE'95 - International Symposium on Signals, Systems and Electronics*, San Francisco, 1995.

[16] “Lagrange multiplier - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Lagrange_multiplier. [Accessed 2 June 2019].

[17] “Microphone noise - The basics about self-noise in mics,” DPA Microphones, [Online]. Available: https://www.dpamicrophones.com/mic-university/the-basics-about-noise-in-mics. [Accessed 6 June 2019].

[18] E. N. Gilbert and S. P. Morgan, “Optimum design of directive antenna arrays subject to random variations,” *The Bell System Technical Journal,* pp. 637-663, 1955.

[19] E. Habets, “AudioLabs - RIR Generator,” Audio Labs, [Online]. Available: https://www.audiolabs-erlangen.de/fau/professor/habets/software/rir-generator. [Accessed 2 May 2019].

[20] E.¨ Habets, “Room Impulse Response Generator,” 20 September 2010.

[21] V. Panayotov and D. Povey, “openslr.org,” [Online]. Available: http://www.openslr.org/12. [Accessed 10 June 2019].

[22] J. Amos, “Dutch men revealed as world's tallest - BBC News,” BBC, 26 July 2016. [Online]. Available: https://www.bbc.co.uk/news/science-environment-36888541. [Accessed 4 June 2019].

[23] T. C. Pichler, “Circular Differential Microphone Arrays,” Graz University of Technology, Graz, 2016.

[24] M. Brookes, “VOICEBOX,” Imperial College London, [Online]. Available: http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html. [Accessed 18 June 2019].

Appendices

Code Snippet: Create Uniform Linear Array Coordinates

```
function microphone_position_array =
createMicrophonePositionArray(numberOfMicrophones,arrayLength,z,roomX,roomY
)
microphone_position_array = [];
spacing = arrayLength/(numberOfMicrophones-1);
for microphoneIndex = 0:numberOfMicrophones-1
    microphonePosition = [(roomX-arrayLength)/2+microphoneIndex*spacing
roomY/2 z];
    microphone_position_array =
[microphone_position_array;microphonePosition];
end
end
```

Code Snippet: Create Uniform Circular Array Coordinates

```
function microphone_position_array =
createMicrophonePositionArray(numberOfMicrophones,Radius,z,roomX,roomY)
microphone_position_array = [];
angleSpacing = 2*pi/numberOfMicrophones;
for microphoneIndex = 0:numberOfMicrophones-1
    angle = microphoneIndex*angleSpacing;
    microphonePosition = [roomX/2+Radius*sin(angle)
roomY/2+Radius*cos(angle) z];
    microphone_position_array =
[microphone_position_array;microphonePosition];
end
end
```

Code Snippet: Create Source Coordinates

```
function source_position_array =
calculateSourcePositionArray(degreeNeeded,z,r,resolution, roomX, roomY)
source_position_array = [];
for source_angle =
0*degreeNeeded/360:2*pi*(degreeNeeded/360)/degreeNeeded*resolution:2*pi*deg
reeNeeded/360
    source_position = [roomX/2-r*cos(source_angle+pi/2) roomY/2-
r*sin(source_angle+pi/2) z];
    source_position_array = [source_position_array;source_position];
end
end
```

Code Snippet: Plot Source And Microphone Array

```
%%%%% scatter the spatial arrangement if needed %%%%%
figure;
scatter3(source_position_array(:,1),source_position_array(:,2),source_posit
ion_array(:,3));
hold on;
scatter3(r(:,1),r(:,2),r(:,3));
hold off;
```

```
legend('Signal Source','Microphones');
hold off;
xlabel("x-axis/m");
ylabel("y-axis/m");
zlabel("z-axis/m");
```

Code Snippet: Full Code For Plotting Beampatterns of Continuous Linear Array

```
c = 340;                        % Sound velocity (m/s)
fs = 16000;                     % Sample frequency (samples/s)
L = [10 10 4];                    % Room dimensions [x y z] (m)
beta = 0;                     % Reverberation time (s)
n = 1024;                       % Number of samples

t = 0:1/fs:0.1; % 0.1 sec of signal input

%%%% Create numbers of microphones in the centre of the room with total
%%%% length and equally spaced
numberOfMic = 50;
micSpacing = 0.01;
r = createMicrophonePositionArray(numberOfMic,micSpacing*(numberOfMic-
1),L(3)/2,L(1),L(2));
degreeNeeded = 360; % angle needed to plot
resolution = 2; % how dense the source will be placed
source_position_array =
calculateSourcePositionArray(degreeNeeded,L(3)/2,5,resolution); % calculate
source position

set(0, 'DefaultLineLineWidth', 1); %%%set default line width
angleIndex = -degreeNeeded/2:resolution:degreeNeeded/2;
plotFrequencyUpperRange = 3000;
plotFrequencyLowerRange = 0;
source_frequency_array =
plotFrequencyLowerRange:50:plotFrequencyUpperRange;

patternToPolarPlot_matrix = [];
%%%% iterate source frequency and angle to plot 2D graph
for i = 1:length(source_frequency_array)
    source_input = cos(2*pi*source_frequency_array(i)*t);
    powerArrayAtDifferentAngle =
calculatePowerAtDifferentAngle(source_position_array,source_input, c, fs,
r, L, beta, n);
    patternToPolarPlot = abs(powerArrayAtDifferentAngle);
    patternToPolarPlot_matrix =
[patternToPolarPlot_matrix;patternToPolarPlot];
end
%%% plot 2D graph
figure;
surf(angleIndex,source_frequency_array,patternToPolarPlot_matrix);
change_figure_config(plotFrequencyLowerRange, plotFrequencyUpperRange);

function h_array = calculateImpulseResponses(c, fs, r, s, L, beta, n)
h_array = [];
microphoneNumbers = size(r,1);
for microphpneIndex = 1:microphoneNumbers
    h = rir_generator(c, fs, r(microphpneIndex,:), s, L, beta, n);
    h_array = [h_array; h];
end
end
```

```
function powerOfSignal = calculatePower(inputSignal)
if ~isvector(inputSignal)
    error('Input must be a vector')
end
powerOfSignal = sum(inputSignal.^2)/length(inputSignal);
end

function output_array =
calculateAllMicrophonesOutput(inputSource,impulseResponses)
output_array = [];
microphoneNumbers = size(impulseResponses,1);
for microphpneIndex = 1:microphoneNumbers
    output = filter(impulseResponses(microphpneIndex,:),1,inputSource);
    output_array = [output_array;output];
end
end

function source_position_array =
calculateSourcePositionArray(degreeNeeded,z,r,resolution)
source_position_array = [];
for source_angle = 0:2*pi*resolution/360:2*pi*degreeNeeded/360
    source_position = [5+r*cos(source_angle) 5+r*sin(source_angle) z];
    source_position_array = [source_position_array;source_position];
end
end

function microphone_position_array =
createMicrophonePositionArray(numberOfMicrophones,arrayLength,z,roomX,roomY
)
microphone_position_array = [];
spacing = arrayLength/(numberOfMicrophones-1);
for microphoneIndex = 0:numberOfMicrophones-1
    microphonePosition = [(roomX-arrayLength)/2+microphoneIndex*spacing
roomY/2 z];
    microphone_position_array =
[microphone_position_array;microphonePosition];
end
end

function powerArrayAtDifferentAngle =
calculatePowerAtDifferentAngle(source_position_array,source_input, c, fs,
r, L, beta, n )
powerArrayAtDifferentAngle = [];
sourceNumbers = size(source_position_array,1);



for sourceIndex = 1:sourceNumbers

    impulseResponsesArray = calculateImpulseResponses(c, fs, r,
source_position_array(sourceIndex,:), L, beta, n);%%% get impulse responses
at this position of s and r

    output_array = calculateAllMicrophonesOutput(source_input,
impulseResponsesArray);

    timeDomainLength = length(output_array(1,:));
    frequencyDomainLength = fs*(0:((timeDomainLength-
1)/2))/(timeDomainLength-1);
    frequencyDomainLength = [fliplr(-
frequencyDomainLength),frequencyDomainLength(2:end)];
```

```
    final_output_fft_array = fft(output_array.').';%calculate each output's
fft
    micNum = length(r(:,1));
    for i = 1:micNum %%%rearrange the frequency component
        fftOutput = final_output_fft_array(i,:);
        final_output_fft_array(i,:) =
[fftOutput(802:1601),fftOutput(801),fftOutput(1:800)];
    end
    spacing = abs(r(1,1) - r(2,1));
    directionArray = calculateDirectionVector(r, spacing, fs, c,
frequencyDomainLength);

    w = directionArray;
    system_output = w .* final_output_fft_array;
    system_output = sum(final_output_fft_array);

    system_output =
[system_output(802:1601),system_output(801),system_output(1:800)];

    timeDomainOutput = ifft(system_output);

    final_output_power =
sum(abs(timeDomainOutput).^2)/length(timeDomainOutput); %%% calculate final
output power

    powerArrayAtDifferentAngle =
[powerArrayAtDifferentAngle,final_output_power];
end
end

function final_output_fft = calculateOutputFFT(outputArrayInTimeDomain)
final_output_fft = [];
outputArrayNumber = size(outputArrayInTimeDomain,1);
for outputIndex = 1:outputArrayNumber
    final_output_fft = [final_output_fft;
fft(outputArrayInTimeDomain(outputIndex,:))];
end
end


%%% far field in the book
%%% take first microphone position in the array as the reference position
function directionArray = calculateDirectionVector(microphonePositionArray,
spacing, fs, c, frequencyDomainLength)
directionArray = [];
mircrophoneNumber = size(microphonePositionArray,1);
frequencyNumber = length(frequencyDomainLength);
for microphoneIndex = 1:mircrophoneNumber
    singleDirectionVector = []; %%% reset singleDirectionVector
    angle_to_use = (microphoneIndex-1)*2*pi/mircrophoneNumber;
    for frequencyIndex = 1:frequencyNumber %% for different frequcny
component in one microphone output
        steering_angle = 0/360*2*pi;
        singleDirectionVector = [singleDirectionVector,divider*exp(-
1*1i*frequencyDomainLength(frequencyIndex)*2*pi/c*spacing*(microphoneIndex-
(mircrophoneNumber+1)/2)*cos(steering_angle_radian-pi/2))];
    end
    directionArray = [directionArray;singleDirectionVector]; %%% append
microphones direction vector
end
```

```
end

function
change_figure_config(plotFrequencyLowerRange,plotFrequencyUpperRange)

view(0,90)
set(gca,'FontSize',18)
xlim([-180 180]);
ylim([plotFrequencyLowerRange plotFrequencyUpperRange]);
shading interp  %%%remove grid
xlabel('Angle/degree');
ylabel('Frequency/Hz')
colorbar;
title(colorbar,'|Signal|')

end
```

Code Snippet: Weighting Function for Uniform Linear Array with DSB

```
function directionArray = calculateDirectionVector(microphonePositionArray,
spacing, fs, c, frequencyDomainLength, steering_angle)
directionArray = [];
mircrophoneNumber = size(microphonePositionArray,1);
frequencyNumber = length(frequencyDomainLength);
steering_angle_radian = steering_angle/360*2*pi;
for microphoneIndex = 1:mircrophoneNumber
    singleDirectionVector = []; %%% reset singleDirectionVector
    angle_to_use = (microphoneIndex-1)*2*pi/mircrophoneNumber;
    for frequencyIndex = 1:frequencyNumber %% for different frequcny
component in one microphone output
        singleDirectionVector = [singleDirectionVector,exp(-
1*1i*frequencyDomainLength(frequencyIndex)*2*pi/c*spacing*(microphoneIndex-
(mircrophoneNumber+1)/2)*cos(steering_angle_radian-pi/2))];
    end
    directionArray = [directionArray;singleDirectionVector]; %%% append
microphones direction vector
end
end
```

Code Snippet: Weighting Function for Uniform Circular Array with DSB (Plane)

```
function directionArray = calculateDirectionVector(microphonePositionArray,
radius, c, frequencyDomainLength, steeringAngle)
directionArray = [];
mircrophoneNumber = size(microphonePositionArray,1);
frequencyNumber = length(frequencyDomainLength);
for microphoneIndex = 1:mircrophoneNumber
    singleDirectionVector = []; %%% reset singleDirectionVector
    steering_angle = steeringAngle/360*2*pi;
    angle_to_use = (microphoneIndex-1)*2*pi/mircrophoneNumber;
    divider = 1/mircrophoneNumber;
    for frequencyIndex = 1:frequencyNumber %% for different frequcny
component in one microphone output
        frequency = frequencyDomainLength(frequencyIndex);
        singleDirectionVector = [singleDirectionVector,(divider)*exp(-
1*1i*2*pi*frequency/c*radius*(cos(steering_angle-angle_to_use)))];
    end
    directionArray = [directionArray;singleDirectionVector]; %%% append
microphones direction vector
```

```
end
end
```

Code Snippet: Weighting Function for Uniform Circular Array with DSB (Elevation)

```
function directionArray = calculateDirectionVector(microphonePositionArray,
radius, c, frequencyDomainLength, steeringAngle, elevationSA)
directionArray = [];
mircrophoneNumber = size(microphonePositionArray,1);
frequencyNumber = length(frequencyDomainLength);
for microphoneIndex = 1:mircrophoneNumber
    singleDirectionVector = []; %%% reset singleDirectionVector

    steering_angle = steeringAngle/360*2*pi;
    angle_to_use = (microphoneIndex-1)*2*pi/mircrophoneNumber;
    for frequencyIndex = 1:frequencyNumber %% for different frequcny
component in one microphone output
        singleDirectionVector = [singleDirectionVector,exp(-
1*1i*frequencyDomainLength(frequencyIndex)*2*pi/c*radius*cos(elevationSA)*(
cos(steering_angle-angle_to_use)))];
    end
    directionArray = [directionArray;singleDirectionVector]; %%% append
microphones direction vector
end
end
```

Code Snippet: Reading Source Signal (Speech File)

```
% Read Desired Source
desired_source_filename = '121-127105-0033.flac';
[source_input,Fs] = audioread(desired_source_filename);
source_input =source_input(6720:end);
source_input =source_input';

% Read Competing Source
competing_source_filename = '2300-131720-0014CompetingSignal.flac';
[competing_input,Fs] = audioread(competing_source_filename);
competing_input =competing_input(4335:end);
competing_input =competing_input';
competing_input = competing_input(1:length(source_input));
```

Code Snippet: Add Noise to Outputs of Microphones

```
function [output,noise]= calculateNoise(input,fs,SNR)
output = zeros(size(input));
noise = zeros(size(input));
for i = 1:length(input(:,1))
    output(i,:) = (v_addnoise(input(i,:),fs,SNR,'k'))';
    noise(i,:) = output(i,:) - input(i,:);
end
end
```

Code Snippet: Calculate the Covariance Matrix

```
function coherenceMatrix= calculateCoherenceMatrix(noiseMatrix)
coherenceMatrix =
zeros(length(noiseMatrix(:,1)),length(noiseMatrix(:,1)),length(noiseMatrix(
1,:)));
```

```
noiseMatrix = fft(noiseMatrix.').';
numOfInput = length(noiseMatrix(1,:));
micNum = length(noiseMatrix(:,1));
for i = 1:micNum %%%rearrange the frequency component
    noiseRow = noiseMatrix(i,:);
    noiseMatrix(i,:) = [noiseRow((numOfInput-
1)/2+2:numOfInput),noiseRow((numOfInput-1)/2+1),noiseRow(1:(numOfInput-
1)/2)];
end

for f = 1:1:length(noiseMatrix(1,:))
    a = noiseMatrix(:,f) * noiseMatrix(:,f)';
    avg = sum(a,'all')/numel(a);
    a = a/avg;%scale
    coherenceMatrix(:,:,f) = a;%filling the output matrix
end
end
```

Code Snippet: Calculate the Weighting Function(MVDR)

```
w=zeros(length(directionArray(:,1)),length(directionArray(1,:)));
for i = 1:1:length(directionArray(1,:))
    noisevv = squeeze(coherenceMatrix(:,:,i));
    inv_noisevv = inv(noisevv);
    direction = directionArray(:,i);
    nominator = inv_noisevv * direction;
    denominator = (conj(direction') * inv_noisevv *direction);
    w(:,i) = (nominator/denominator)';
end
```